Dynamic Bandwidth Allocation for an OFDMA based RF Network-on-Chip

Eren Unlu

CentraleSupélec

Nº d'ordre : 2016-06-TH

CentraleSupélec

École Doctorale MATISSE

« Mathématiques, Télécommunications, Informatique, Signal, Systèmes Electroniques »

THÈSE DE DOCTORAT

Domaine : STIC

Spécialité : Télécommunications

Soutenue le 21/06/2016

Présentée par :

Eren UNLU

Allocation Dynamique de Bande Passante pour l'Interconnexion RF d'un Résau-sur-Puce

---

Directeur de thèse :	Christophe Moy	*Professeur de IETR/CentraleSupélec*
Composition du jury :		
Président du jury :		
Rapporteurs :	Fabien Clermidy	*Chercheur CEA-Leti*
	Lionel Torres	*Professeur de Université de Montpellier 2*
Examinateurs :	Myriam Ariaudo	*Maître de Conférences à ENSEA de Cergy/ETIS*
	Sébastien LeBeux	*Maître de Conférences à Ecole Centrale de Lyon*
	Yves Louët	*Professeur de IETR/CentraleSupélec*

Contents

Abstract

With increasing silicon manufacturing capabilities, it is expected that chip multiprocessors (CMPs) with thousands of cores will be ready before 2030. With increasing number of cores, the main bottleneck for the CMPs is to sustain the communication between these cores and traditional Network-on-Chip (NoC) interconnects start to be not sufficient, as number of routers to traverse for packets, distances and congestion increase. Recently, optical and RF communications have been proposed to overcome these challenges for on-chip interconnects.

Wired RF interconnect seems the most feasible one compared to wireless RF connection or optical interconnects, as it is full CMOS compatible and required high operating frequency for signal modulation is available. Proposed state-of-the-art RF or optical on-chip interconnects require high amount of CMOS or optical components to generate orthogonal channels and lack the effective bandwidth reconfiguration capacity. *Wired RF Network-on-Chip Reconfigurable-on-Demand (WiNoCoD)* project aims to break this bottleneck by proposing an Orthogonal Frequency Division Multiple Access (OFDMA) based on-chip RF interconnect, for the first time to the best of our knowledge. This approach enables a fully flexible RF architecture managed in a pure digital manner.

This thesis, which contributes to WiNoCoD chip, proposes a 3-level hierarchical architecture for a 2048-core generic CMP and OFDMA based RF interconnect. In this work, we propose dynamic decentralized and centralized bandwidth reconfiguration schemes for this interconnect, concerning the very specific constraints and requirements of the CMP environment. Built on this framework, several effective RF bandwidth allocation algorithms are proposed and examined using realistic stochastic on-chip traffic models. This dynamic approach can achieve remarkably low latencies even under traffic loads near the network capacity, without requiring elevated computational complexity. In addition to these, a novel bandwidth allocation algorithm is introduced, which intends to decrease transmission latency by exploiting the bimodal nature of on-chip cache-coherency packets. We have shown that, this specialized infrastructure can decrease average latency up to 10 times further compared to previously proposed algorithms in this thesis under lower traffic intensity and longer cache lines. We also consider the utilization of dynamic modulation orders as an additional degree of freedom and propose intelligent selection algorithms which seek optimal delay-energy trade-off. For instance, with this approach average power can be decreased up to 15 times under certain circumstances, by relaxing average latency requirement to a few symbols long. Information theoretic capacity analysis of this interconnect is done for different transmission line topologies.

Résume

L'essor des technologies micro-électroniques permet d'envisager que les puces de traitement intégreront plusieurs milliers de coeurs dans la prochaine décennie. Le principal défis de conception de ces puces se situe alors au niveau des échanges de données entre les cœurs cars les interconnexions NoC (Network on Chip) traditionnelles atteignent leurs limites en termes de congestion et de temps de latence. Les communications RF guidées préférés aux connexions RF non guidées pour leur plus faible puissance d'émission requise ou optiques en raison de leur compatibilité avec la technologie CMOS, qui désormais peut supporter les fréquences de fonctionnement exigées pour moduler les signaux RF. Mais, les solutions envisagées jusque là nécessitent une circuiterie importante pour générer des canaux RF orthogonaux, avec des capacités limitées en termes de flexibilité d'allocation de ces canaux. Pour y remédier, projet WiNoCoD (Wired RF Neetwork-on-Chip Reconfigurable-on-Demand) propose pour la première fois une approche basée sur l'OFDMA (Orthogonal Frequency Division Multiple Access) pour l'accès à l'interconnexion RF intra-puce. En effet, cette approche permet une gestion purement numérique de l'allocation des ressources fréquentielles de communication, apportant ainsi une grande flexibilité exploitable à la volée. La puce WiNoCoD propose une architecture générique à 2048 coeurs de traitement répartis en trois niveaux de hiérarchie dont les éléments du niveau supérieur sont reliés par une interconnexion RF basée sur l'OFDMA. Dans cette thèse, nous proposons des schémas de reconfiguration de l'allocation des ressources fréquentielles centralisés et décentralisés adaptés aux contraintes spécifiques de l'environnement intra-puce à très grand nombre de coeurs. Plusieurs algorithmes d'allocation de fréquence sont étudiés dans le cas de modèles stochastiques de trafic réalistes. L'approche dynamique proposée permet d'obtenir des performances remarquables en termes de latence, dans des conditions de trafic à la limite de la capacité du NoC, pour une faible complexité de calcul. En outre, un nouvel algorithme d'allocation est introduit pour décroître la latence de transmission en tenant compte de la nature bimodale des paquets de cohérence de cache d'une telle puce. Nous montrons que cela permet de diminuer la latence moyenne de livraison des paquets d'un facteur 10 en comparaison des autres solutions étudiées dans un contexte d'intensité de trafic modérée et pour des lignes de cache plus longues. Nous étudions aussi l'utilisation dynamique d'ordres de modulation comme un degré de liberté supplémentaire et nous proposons un algorithme de sélection du meilleur compromis entre consommation d'énergie et délai de livraison des paquets. Nous montons ainsi que la puissance moyenne peut être améliorée d'un facteur 15 en relaxant le délai de livraison de quelques symboles. Une analyse de capacité au sens de la théorie de l'information est aussi effectuée pour différentes topologies de la ligne guidée.

Dedicated to my parents

Chapter 0

Résumé en Français : Allocation Dynamique de Bande Passante pour l'Interconnexion RF d'un Résau-sur-Puce

Chapitre 1 : Introduction

L'accélération des traitements numériques a été portée depuis un demi-siècle par la diminution de la géométrie des transistors, mais cette tendance est dorénavant freinée en raison de problèmes thermiques et lithographiques liés aux dimensions nanométriques actuelles. Par ailleurs, le compromis entre puissance de calcul et consommation a poussé l'utilisation de plusieurs processeurs en parallèle sur une même puce pour exécuter des applications à forte exigence en termes de calculs. Aujourd'hui, il est fréquent de voir des puces avec plusieurs cœurs dans nos téléphones, ordinateurs, serveurs, terminaux ADSL, appareils photo numériques, etc.. En électronique embarquée, on parle de systèmes sur puces (System-on-Chip ou SoC) où les unités sont généralement de natures différentes (hétérogènes). Dans le domaine informatique, cela se traduit par la mise en parallèle de plusieurs voire nombreuses unités de traitement identiques (homogène), comme dans les GPU (Graphical Processing Units), processeurs superscalaires, etc.. Dans un avenir proche, on estime à quelques milliers le nombre de cœurs présents sur une seule puce. Ces systèmes sont appelés généralement avec la dénomination anglaise Chip Multiprocesseurs (CMPs) ou processeurs ManyCore.

Avec l'augmentation du nombre de cœurs dans une même puce, le problème des communications entre les cœurs devient prépondérant. Il est devenu impossible d'implanter des fils dédiés point à point entre tous les coeurs et des problèmes de congestion apparaissent avec les bus conventionnels. Les chercheurs ont introduit un nouveau paradigme connu sous le nom de réseau sur puce (Network-on-Chip ou NoC), où la couche de communication est détachée des coeurs et la transmission entre cœurs est effectuée par paquets via des routeurs, comme dans un réseau. Même si les NoC ont démontré leur performance en termes de latence et de bande passante, ils atteignent à leur tour des limites à partir de plusieurs dizaines de cœurs.

Récemment, les interconnexions optiques et radiofréquence (RF) sur puce ont été proposées pour fournir une solution à l'apparition de ce goulot d'étranglement. Ces interconnexions utilisent des ondes électromagnétiques pour transmettre des signaux à des vitesses plus proches de celle de la lumière, à la différence du câblage de cuivre classique utilisé jusqu'ici. Cependant, ces deux architectures, optiques et RF, ont besoin d'implanter des circuits spécialisés dans les émetteurs-récepteurs de chaque noeud (en regroupe plusieurs cœurs entre eux car connecter chaque cœur serait trop complexe) qui a un accès au canal de communication. En raison de la nature statique des architectures d'émission réception jusqu'à présent identifiées dans la littérature, l'allocation dynamique des canaux de communication à différents noeuds, en fonction de leur demande instantanée de bande passante est impossible. Cependant, en raison du trafic très fluctuant généré par les applications entre les cœurs, les approches RF et optiques apportent jusqu'à présent une solution limitée, ou surdimensionnée entre les capacités qu'elles apportent et celles qui peuvent être exploitées effectivement en cours de fonctionnement.

Afin de remédier à ces limitations, le projet WiNoCoD (Wired RF Network-on-Chip Reconfigurable-on-Demand) a été initié grâce au financement de l'Agence Nationale de Recherche (ANR). Les partenaires du projet sont ETIS-ENSEA, LIP6 -UPMC, NXP Semiconductors et IETR-CentraleSupélec. Ce travail de thèse contribue au projet WiNoCoD. La contribution majeure du projet WiNoCoD pour la communauté scientifique des réseaux-sur-puce est son interconnexion de communication RF basée sur l'OFDMA (Orthogonal Frequency Division Multiplexing Access). Contrairement aux interconnexions sur puce existantes, la modulation OFDM, support de l'OFDMA, permet de générer de nombreux canaux orthogonaux grâce à un nombres de circuits analogiques réduit (non pas une chaîne RF à bande étroite par canal, mais une seule chaîne RF globale à large bande). En outre, l'encodage des données sur les canaux de fréquences orthogonales est une procédure purement numérique en OFDM et sa capacité intrinsèque de diffusion est un atout très important en raison des caractéristiques particulières du trafic des données dans un réseau sur puce.

Les contributions de cette thèse sont :

- Une structure de contrôleur de RF est proposée pour l'interconnexion OFDMA de WiNoCoD.
- Plusieurs algorithmes d'allocation de bande passante efficaces (distribués et centralisés) sont proposés et leurs performances évaluées, concernant les demandes et contraintes très spécifiques de l'environnement sur-puce.
- Un protocole innovant pour l'arbitrage des sous-porteuses pour des longueurs bimodales de paquets sur-puce, qui ne nécessite aucune signalisation supplémentaire est introduit.
- Une évaluation de l'utilisation des ordres de modulation élevés est étudiée en fonction du compromis entre délai et consommation d'énergie.
- Les algorithmes proposés ne sont pas restreints aux limites de WiNoCoD, mais peuvent être étendus à d'autres interconnexions RF basées sur l'OFDMA pour les architectures CMP ou des réseaux à haute vitesse.

Chapitre 2 : L'Ere des «1000 cœurs» et le défi des réseaux-sur-puce

2.1 Multiprocesseurs

Les CMPs offrent une nouvelle réponse aux limitations des monoprocesseurs en utilisant de nombreux cœurs relativement simples plutôt qu'un seul cœur puissant, ce qui augmente les performances à la fois en termes de puissance de calcul et d'efficacité énergétique en exploitant le parallélisme. Un cœur est une unité de traitement arithmétique qui effectue les opérations logiques et de commande, qui est généralement composé de deux entités distinctes: unité arithmétique et logique (ALU) et une unité de commande (CU).

2.1.1 Caches et Mémoire

Dans un processeur multi-cœurs, chaque cœur dispose d'une mémoire pour les données et les instructions, sous la désignation Mémoire cache de niveau 1 (L1 - Level 1). La mémoire cache (antémémoire) est composée de registres temporaires pour un volume relativement faible de données, qui stocke les copies de la mémoire principale et qui

offre un accès très rapide pour le cœur de traitement. En fonction de l'architecture, il peut y avoir des niveaux plus élevés de caches tels que L2 et L3. Cependant, toutes les architectures ne partagent pas cette hiérarchie. Par exemple, le CMP qui est envisagé dans ce travail de thèse a seulement un cache L1 pour chaque cœur et une mémoire RAM partagée et distribuée physiquement entre tous les coeurs, qui peut être considéré comme un cache L2.

2.1.2 Protocole de Cohérence de Cache

Les cœurs d'un multiprocesseur peuvent accéder à tout endroit dans la mémoire partagée. Cependant, quand un cœur modifie les données dans un emplacement d'adresse, il peut y avoir déjà des copies des caches simultanées dans d'autres cœurs. Par conséquent, lorsque les nouvelles données sont écrites, les cœurs (qui utilisent cette adresse) risquent d'avoir une copie erronée pour cette ligne d'adresse. Par conséquent, ces cœurs doivent être informés du changement de contenu. Ce problème d'incohérence est connu sous le terme de cohérence de cache. Il existe différents protocoles pour résoudre le problème de la cohérence de cache, mais les plus largement connus sont "espionnage de bus" et "espionnage à répertoire". Dans l'espionnage, à chaque fois qu'un cœur ne peut pas retrouver les données dans son cache, il diffuse une requête de lecture sur le bus reliant tous les cœurs et les caches. Comme chaque contrôleur de cache des autres cœurs écoute ces émissions, ils invalident les copies dans leurs caches avec l'étiquette de la ligne d'adresse dans cette demande. La deuxième approche est le protocole par répertoire (directory), où les répertoires sont responsables de la mise à jour et de mémoriser les états et les propriétaires de blocs mémoire (lignes d'adresse). Chaque cœur, qui veut extraire la copie d'une adresse dans la mémoire principale, doit se référer au répertoire en premier lieu. Afin d'orchestrer l'exécution d'applications et fournir la cohérence des caches, les cœurs et les éléments de mémoire doivent communiquer entre eux. Ces messages de cohérence de cache constituent la base du profile des communications sur une telle puce.

2.2 Réseau-sur-Puce

2.2.1 Des Bus au Réseau-sur-Puce

Quand le nombre d'éléments reliés à un même bus augmente, la charge capacitive et résistance parasite augmentant également, ce qui est une cause supplémentaire de retard dans les transmissions sur le bus. Dans les architectures submicroniques classiques, les informations numériques sont transmises entre les noeuds par des fils de cuivre en

augmentant ou en diminuant la tension électrique sur ces fils. En outre, comme le nombre d'unités qui veut accéder au bus augmente, la bande passante par unité diminue, ce qui cause d'autant plus de risques de congestion. A l'échelle des CMPs, avec un nombre de cœurs de plus en plus important, jusqu'à atteindre plusieurs milliers d'unités, un cadre de communication de manière paquetisée a été introduit, connu sous le nom de Réseau-sur-puce (NoC).

2.2.2 Topologies de NoC

Depuis les recherches initiales sur les NoC, différentes topologies ont été proposées : 2D-grille, 2D-tore, octogone etc. (Fig 0.1). Malgré ses avantages, comme nous approchons d'une ère avec des centaines, voire des milliers de cœurs, ces NoC classiques sont aux prises avec un problème d'extensibilité. Par exemple, on peut voir que la distance entre les deux cœurs les plus éloignés dans un réseau de grille est $2\sqrt{N_{cores}}$, où N_{cores} est le nombre de cœurs.

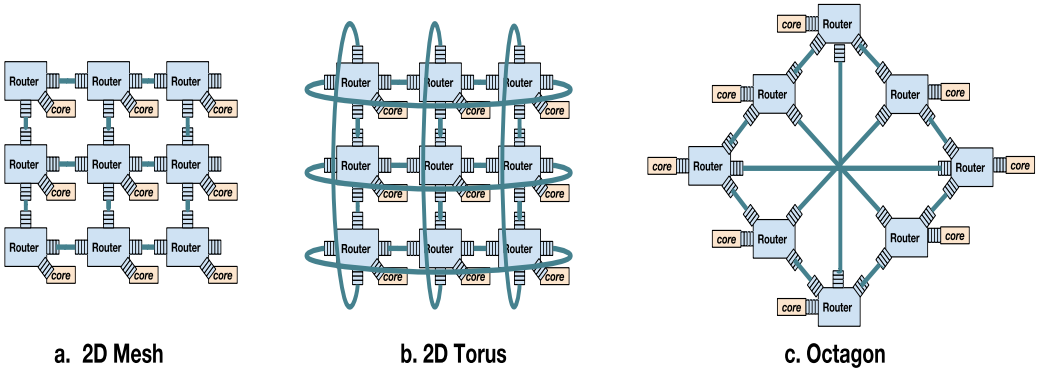


FIGURE 0.1: Trois des topologies les plus classiques pour les NoCs 2D, où les éléments sont interconnectés via des routeurs tamponnées: (a) grille (b) tore (c) octogone

2.2.3 Tera-Scale Multi-core Processor Architecture (TSAR)

Il est important de mentionner ici le projet TSAR (Tera-Scale Multicore processeur Architecture) [1]. Parmi les partenaires de ce projet, l'UPMC-LIP6 et NXP Semiconductors sont aussi des partenaires du projet WiNoCoD. Dans un sens, ce projet peut être considéré comme une base primordiale de WiNoCoD, comme ses principes architecturaux en termes de mémoire et de protocole de cohérence de cache a de nombreuses caractéristiques communes avec TSAR. La différence fondamentale du projet WiNoCoD est son infrastructure NoC. En effet, TSAR a été modélisé avec un réseau de grille-2D classique pour la communication entre les tuiles (unité atomique de 4 cœurs avec sa RAM, un contrôleur de répertoire).

2.3 Interconnexion RF et Optiques

Avec un horizon où des milliers de cœurs seront concentrés sur une même puce, l'industrie des semi-conducteurs a compris que les NoCs filaires classiques sont loin de fournir les besoins attendus en termes de latence, de bande passante et de contrainte de puissance électrique. Ainsi a émergé l'idée de communiquer par ondes électromagnétiques à une vitesse plus proche de celle de la lumière. L'International Technology Roadmap for Semiconductors (ITRS) affirme que les interconnexions optiques et RF représentent l'avenir pour satisfaire les besoins à long terme de bande passante, de latence, la puissance, considérant que l'on s'attend à ce que le nombre de cœurs sur une puce dépasse plusieurs milliers avant la fin de la prochaine décennie (Fig. 0.2).

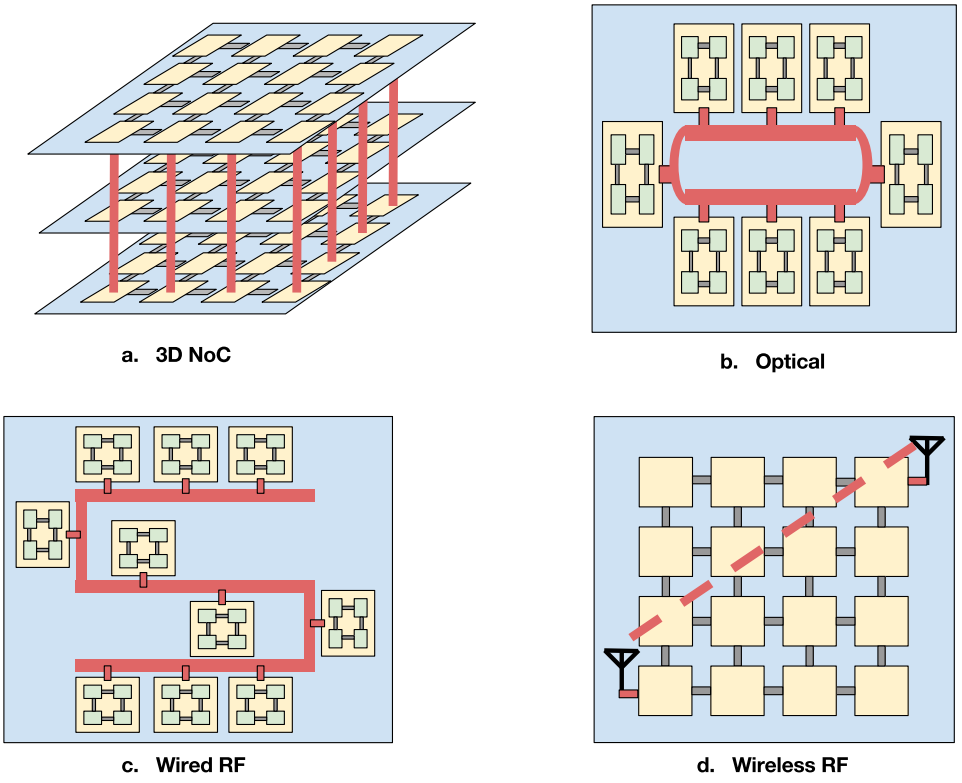


FIGURE 0.2: L'illustration d'options d'interconnexion sur puce innovante récemment proposées

2.3.1 Interconnexions Optiques

Les développements récents de la nanophotonique ont permis l'implantation d'éléments optiques tels que des guides d'ondes denses, des filtres, des modulateurs etc. sur une seule puce. Toutefois, ces tentatives sont encore dans la phase de début. Récemment, [2] a démontré sa faisabilité dans le cas d'un simple multiprocesseur avec 2 cœurs, avec

une interconnexion optique sur puce complète. Pour permettre le transfert simultané de plusieurs signaux sur le même guide d'ondes, des canaux orthogonaux (longueurs d'onde) sont générés. Une source laser sur la puce ou hors de la puce génère et fait circuler l'énergie photonique sur un guide d'onde dédié, capable de véhiculer toutes les longueurs d'onde utilisées dans le système. Des résonateurs en micro-anneaux (microring resonators) sont utilisés en tant que modulateurs pour la transmission de données et en tant que filtres pour la réception. Donc, nous pouvons comprendre que nous avons besoin d'implanter un grands nombre de ces modules pour créer tous les canaux orthogonaux nécessaires, ce qui est encombrant et consommateur d'énergie d'une part et peu évolutif d'autre part. Chacun de ces micro-anneaux est fabriqué pour une longueur d'onde spécifique, qui est déterminée par traitement thermique, et par différence de quantité de charge électrique injecté, ou en faisant varier le rayon de l'anneau au cours du processus de fabrication. Ainsi, ce système est statique par nature et il est impossible de redistribuer la bande dynamiquement entre les nœuds de traitement. ATAC et Corona sont deux exemples de l'état de l'art des architectures optiques sur puce.

Les interconnexions photoniques sont considérées comme une technologie efficace pour réduire la latence de manière significative grâce à leur grande bande passante, tout en offrant une faible consommation électrique. Cependant, leur praticabilité est mise en doute, au moins pour un avenir proche, à cause du bruit du couplage de guide d'ondes et de la taille relativement importante des composants optiques pour les intégrer en grands nombres dans une puce. La fabrication de composants photoniques sur puce est encore confrontée à de nombreux défis. Il impose la juxtaposition de deux technologies differents : la technologie CMOS pour l'architecture de traitement et la technologie optique pour l'interconnexion. En outre, il n'existe pas d'éléments de stockage optique, ainsi un tel système dépend d'une infrastructure électrique additionnelle. Comme mentionné précédemment, les NoC optiques nécessitent des sources laser constantes soit sur la puce, soit hors puce, avec un guide d'onde dédié séparé.

2.3.2 Interconnexions RF

En raison des inconvénients des interconnexions optiques, les chercheurs ont orienté leurs investigations vers les interconnexions Radio Fréquence (RF), qui utilisent encore les ondes électromagnétiques (EM). La fréquence de transition des transistors CMOS est toujours dans une tendance d'augmentation exponentielle de génération en génération, permettant désormais d'envisager des fréquences maximales jusqu'à 1 THz. Cela positionne les composants RF CMOS comme des candidats naturels pour les émetteurs

récepteurs sur puce à haute fréquence. L'avantage de cette approche est sa compatibilité CMOS avec la partie traitement de la puce (cœurs, mémoires, etc.). De plus, c'est une technologie beaucoup plus mature par rapport à l'optique sur puce.

Il y a deux propositions distinctes pour les interconnexions RF: par la propagation en espace libre (sans fil mais avec une antenne d'émission et une antenne de réception) ou la propagation guidée (RF filaire ou sur guide d'ondes). Pour la RF sans fil, l'idée est de générer des liaisons à haut débit entre les cœurs distants, afin de réduire la latence et la congestion, sans la nécessité d'un milieu de propagation supplémentaire dédié. Le défi majeur dans ce paradigme est la difficulté de caractériser les effets de la propagation, ainsi que la fabrication de petites antennes en nanotechnologies avec des caractéristiques électromagnétiques adéquates. La viabilité des antennes sur puce sans fil n'est pas encore démontrée et les propositions innovantes qui apparaissent, telles que les antennes en nanotubes de carbone, ne sont pas encore des technologies matures. Ainsi, la propagation RF via une ligne de transmission guidée (RF filaire) a reçu plus d'attention dans la communauté de la recherche sur puce par rapport à son homologue sans fil. Comme la distance de communication est faible, la méthode de couplage capacitif efficace peut être utilisée pour réaliser la transmission. En outre, le guide d'ondes permet une atténuation réduite, donc une puissance de transmission, qui est la plus consommatrice d'énergie dans un système de communication, limitée au plus juste (et non dispersée dans toutes les directions). C'est l'option qui sera considérée dans ce travail.

2.4 Caractéristiques du Trafic sur Puce

Comme première étape, les chercheurs se sont appuyés sur des modèles de trafic synthétiques primitifs pour évaluer leurs conceptions, mais ces modèles se révèlent souvent trop naïfs pour assurer la validité des simulations. D'autre part, il y a un nombre trop limité d'applications sur multicœurs, telles que celles fournies par PARSEC ou Splash-2. Un modèle statistique réaliste universel d'émulation du trafic d'un NoC est essentiel. Il existe différents modèles stochastiques de trafic pour la simulation sur puce dans la littérature, utilisant la notion d'auto-similarité, phénomène ayant pour origine la hiérarchie de cache dans les systèmes à mémoire partagée. Nous nous basons sur ces modèles de l'état de l'art dans notre travail.

Chapitre 3 : Projet WiNoCoD et Interconnexion RF filaire basée sur l'OFDMA

Nous avons vu que les interconnexions RF filaires apparaissent comme le candidat le plus réaliste à court terme. Cependant, tout comme leurs homologues optiques, les interconnexions RF proposées dans l'état de l'art reposent sur des circuits analogiques pour générer des canaux fréquenciels, ce qui limite leurs capacités d'allocation dynamique des ressources, notamment en termes de coût, de surface occupée et de consommation. Pour surmonter tous ces désavantages et fournir une réelle avancée en termes de bande passante reconfigurable, le projet WiNoCoD (Wired RF Network-on-Chip Reconfigurable on Demand) a été initié en 2013, en partenariat avec l'ANR, par ETIS-ENSEA, LIP6-UPMC, Supelec-IETR et NXP Semiconductors.

3.1 L'architecture sur Puce de WiNoCoD

3.1.1 Niveaux de Communication

Le circuit issu de WiNoCoD est un multiprocesseur massivement parallèle et générique de 2048 cœurs de traitement. Un principe de mémoire partagée est adopté, où tout l'espace d'adresse est accessible par tous les cœurs (NUMA - Non Uniform Memory Architecture). Il existe 3 niveaux hiérarchiques principaux, avec un type d'infrastructure de communication particulier pour chaque niveau. Au niveau le plus bas se trouve la tuile, constituée de 4 cœurs, 2 Go de RAM (une sous-partie de la mémoire globale du système) et un contrôleur de mémoire. Tous ces éléments sont reliés entre eux par un crossbar switch. Au niveau suivant, 16 tuiles sont interconnectés par un réseau en quadrillage (mesh 2D). Ceux-ci sont appelés tuilesets (ou grappes de tuiles) et il y a 32 tuilesets au total. Au plus haut niveau, ces 32 tuilesets sont interconnectés par la ligne de transmission RF. Par exemple, si un cœur veut transmettre des informations à un autre cœur dans un tuileset différent, le message doit traverser les 3 différentes couches de communications. L'architecture de WiNoCoD avec 2048 cœurs est illustrée sur la Fig. 0.3.

3.1.2 Détails du Protocole de Cohérence de Mémoire Cache

Nous employons dans WiNoCoD un protocole de cohérence de cache hybride par répertoire (Distributed Hybrid Cache Coherency Protocol), comme dans TSAR [1]. Une approche similaire est adoptée pour ATAC [3], qui est une architecture à 1024 cœurs interconnectés en optique. Une approche d'écriture transversale (Write Through) est adoptée,

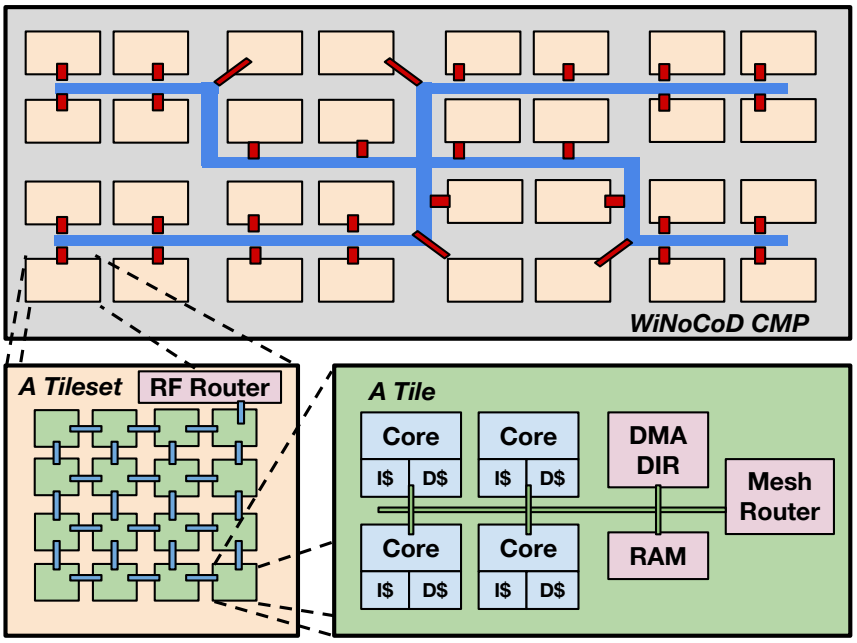


FIGURE 0.3: L'architecture à 3 niveaux de WiNoCoD, avec 2048 cœurs au total

c'est-à-dire que dans le cas où un cœur veut écrire une donnée à une ligne d'adresse, il transmet une demande d'écriture dans le répertoire responsable de cette ligne d'adresse. Par conséquent, si la ligne d'adresse destinataire est associée au répertoire de la même tuile, le message ne va pas à l'extérieur de la tuile, et utilise uniquement le crossbar switch. De même, si la ligne d'adresse est dans une tuile différente mais dans le même tuileset, il utilise uniquement le réseau en quadrillage. Si la ligne d'adresse est dans une tuile d'un tuileset différente, alors il doit utiliser l'interconnexion RF. Dans le protocole DHCCP de WiNoCoD, lorsque le nombre de cœurs partagants d'une ligne d'adresse dépasse un certain seuil (par exemple 8), le répertoire commence à garder le nombre de partageurs, plutôt que de garder les identificateurs (ID) de chaque cœur explicitement. Dans ce cas, pour chaque invalidation, le répertoire diffuse un message à chacun des 2048 cœurs et compte le nombre de messages d'ACK pour vérification. C'est la seule façon d'être en mesure de suivre des milliers de partageurs possibles. Le répertoire dans une tuileset est représenté sur la Fig. 0.4.

3.2 Notions Préliminaires sur l'OFDMA

L'OFDM (Orthogonal Frequency Division Multiplexing) est une technique de modulation qui transforme un signal de bande passante large en plusieurs canaux orthogonaux plus étroits. L'OFDM code ainsi l'information numérique sur le domaine des fréquences plutôt que dans le domaine temporel. Pour le mettre en oeuvre, les données numériques

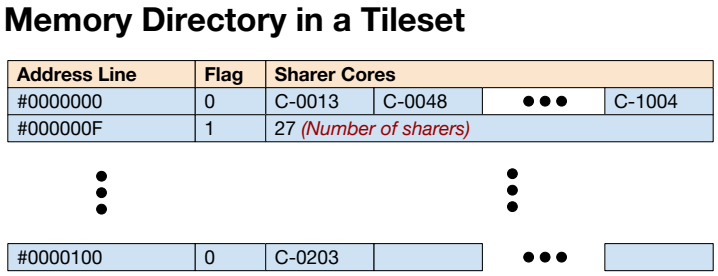


FIGURE 0.4: Context du répertoire de la mémoire dans une tuileset

sont tout d'abord mises en symboles de constellations BPSK, QPSK, M-QAM, etc., et chaque symbole de la constellation est associé à une sous-porteuse. La sous-porteuse est l'unité de fréquence atomique dans un signal OFDM, ou en d'autres termes c'est l'ensemble de toutes les sous-porteuses à bande étroite qui sont émises en parallèle. C'est une transformée de Fourier discrète inverse (IDFT) qui est appliquée au vecteur de N symboles pour les positionner sur N sous-porteuses en parallèle, où chacun d'eux est maintenant un nombre complexe associé au symbole de la constellation codée (chaque nombre complexe représente un certain nombre de bits). Le résultat de cette transformation donne un vecteur de N nombres complexes. Après, ce vecteur de N points est sérialisé et converti en un signal dans le domaine temporel. Ce signal s'est appelé un symbole OFDM. A la réception, l'inverse de ces opérations est effectuée. Les blocs élémentaires d'un émetteur-récepteur OFDM sont montrés sur la Fig. 0.5.

Les avantages de l'OFDM peuvent être énumérés comme suit :

- Robustesse contre les canaux sélectifs en fréquence et l'égalisation est simple pour chaque sous-porteuse.
- Génération de canaux fréquentiels par un processus numérique, d'où l'économie de nombreux circuits analogiques.
- Une bande passante reconfigurable grâce à une mise en œuvre numérique.
- Haute efficacité spectrale grâce à des sous-porteuses fenêtrées par des fonctions sinc orthogonales.

L'OFDMA (Orthogonal Frequency Division Multiple Access), d'autre part est un schéma d'accès multiple basé sur l'OFDM, avec tous les avantages de la couche physique de l'OFDM dont une bande passante reconfigurable par utilisateur. Un utilisateur encode l'information sur ses sous-porteuses allouées à la transmission comme expliqué

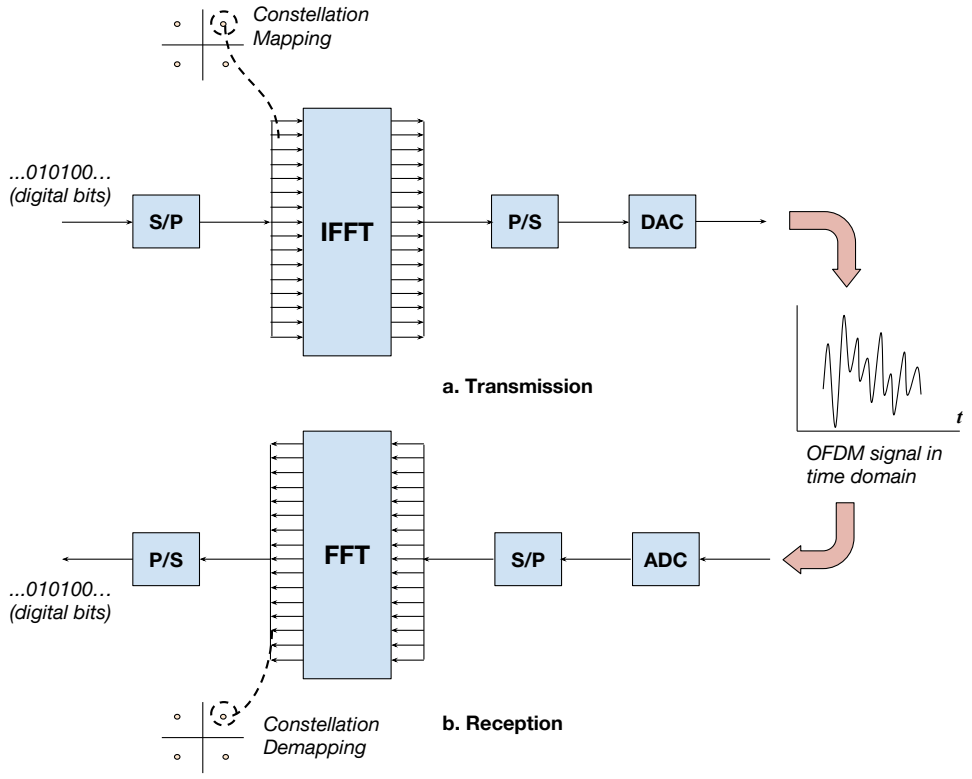


FIGURE 0.5: Les blocs de base d'un émetteur-récepteur OFDM

précédemment, et maintient les autres inactives (zéro avant IFFT) pour les laisser libres à d'autres utilisateurs. Cette procédure est mise en oeuvre dans le domaine numérique, simplement en manipulant le vecteur de bits avant l'IFFT. Cette tâche peut être réalisée par un microprocesseur ou un circuit numérique. En outre, comme chaque noeud dans le système doit décoder un symbole OFDM en entier pour extirper le signal le concernant, l'OFDMA est doté de capacités de diffusion intrinsèques.

3.3 L'Interconnexion RF guidée de WiNoCoD basée sur l'OFDMA

La technologie des convertisseurs fournis par notre partenaire NXP a une bande passante de 20 GHz. En raison des contraintes d'adaptation et de la propagation guidée, le spectre le plus approprié a été choisi entre 20-40 GHz. Il est décidé d'avoir 1024 sous-porteuses OFDM, ainsi des modules FFT et IFFT de 1024 points sont nécessaires. Comme nous avons la bande passante est de 20 GHz, avec 1024 sous-porteuses l'espacement de fréquence entre les sous-porteuses est de 19.53 MHz, soit une durée symbole de T = 1 / 19.53 MHz = 51.2 nanosecondes.

3.3.1 Contrôleur RF

Un paquet qui doit être envoyé par une tuile dans la ligne guidée RF est routé à travers le réseau en quadrillage du tuilset vers le contrôleur RF de ce tuileset. Ces paquets sont traités si nécessaire, c'est-à-dire fragmentation ou défragmentation, extraction ou insertion d'informations telles que l'identifiant de la source, etc., puis insérés dans la file d'attente de transmission. Le contrôleur RF prend certaines décisions selon des informations qu'il possède de l'état de sa file d'attente (Queue State Information - QSI) de transmission (TX) et les informations de trafic en provenance d'autres tuilesets qu'il peut recevoir directement des autres tuilesets, soit lui être fournis par une unité centrale intelligente (Central Intelligent Unit-CIU). Le contrôleur RF modifie en fonction la position et l'encodage en symboles des données dans son buffer précédent la IFFT et configure ainsi son occupation des sous-porteuses du symbole OFDM qui va être transmis. Chaque tuileset fait de même pour chaque symbole OFDM et les porteuses d'un symbole OFDM sont ainsi réparties entre tous les tuilesets. La répartition de l'occupation des sous-porteuses entre les tuilsets est l'enjeu de la thèse et plusieurs algorithmes vont être proposés dans le manuscrit. L'un ou l'autre pourra être utilisé en changeant le programme du ou des processeurs du contrôleur RF gérant le buffer d'émission.

3.3.2 Tête analogique RF

Cette partie est le travail de thèse de Frédéric Drillet et Lounis Zerioul du laboratoire ETIS. Dans la tête d'émission analogique RF, les mélangeurs associent un signal OFDM en bande de base avec la porteuse issue de l'oscillateur local. Le mélange se produit dans un MOSFET, dont la grille et le drain sont respectivement alimentés par l'oscillateur local et le signal en bande de base. La sortie de fréquence plus élevée est récupérée dans la source de transistor MOSFET. Grâce aux sorties différentielles du convertisseur numérique-analogique (CNA), deux modulateurs-I/Q sont combinés pour supprimer les replicas de fréquences. Ensuite, ce signal est amplifié par un amplificateur et injecté sur la ligne RF par un couplage à transistors qui a été préféré à un couplage capacitif classique.

En réception, le signal reçu de la ligne de transmission est amplifié par un amplificateur à faible bruit (LNA) et envoyé aux mélangeurs et l'oscillateur local de 30 GHz pour obtenir les composantes en phase (I) et en quadrature (Q). Des filtres passe-bas (LPF) sont également utilisés puis les composantes I et Q sont converties dans le domaine numérique par les convertisseurs analogique-numérique (CAN). Apres une conversion série vers parallèle, ce vecteur de valeurs I et Q est transposé dans le domaine fréquenciel par un module de transformée de Fourier rapide (FFT). Les symboles de la constellation

qui en résultent sont extraits sous forme de bits; sérialisés et finalement sont fournies au contrôleur RF en réception. La tête d'émission et de réception analogique d'un tuileset est illustrée sur la Fig. 0.6.

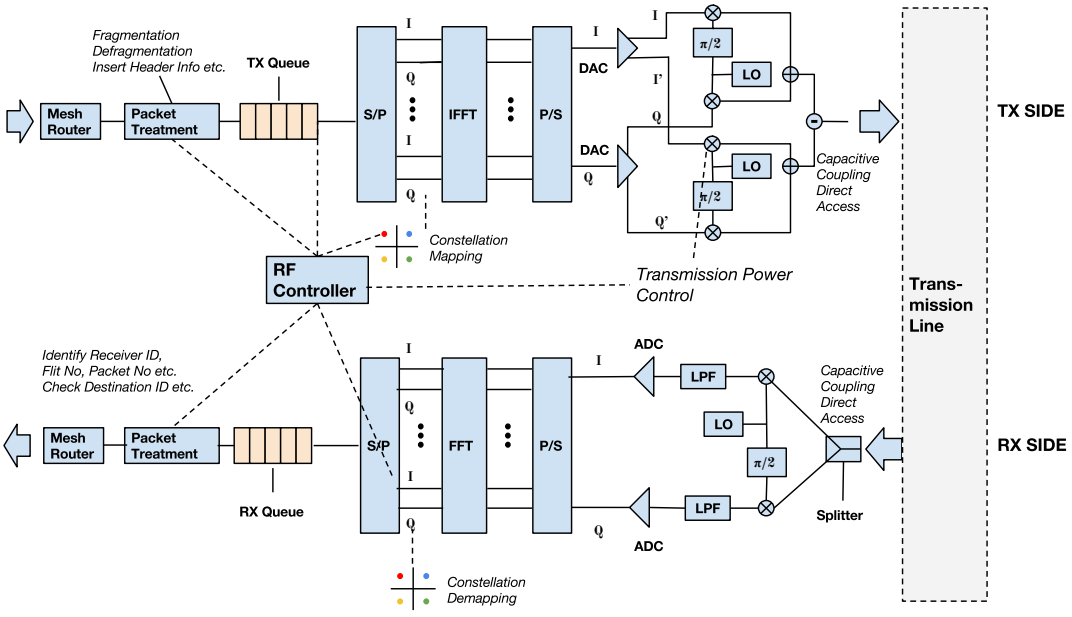


FIGURE 0.6: La tête numérique et la tête analogique RF pour la transmission et la réception dans un tuileset

Les modules FFT et IFFT sont conçus en technologie CMOS 120 nm. Il a été estimé que la superficie de chacun de ces modules est de 0,31 mm2 et la consommation d'énergie de 67.5 mW. La durée de calcul pour FFT / IFFT est considérée comme une latence du pipeline de la communication. Chacun des CAN et CNA sont conçus avec la technologie 120 nm et ont une superficie estimée de 0,12 mm^2 et leur consommation d'énergie de 81 mW.

L'étude de la ligne de transmission s'est effectuée dans le cadre de la thèse de Mohamad HAMIEH du laboratoire ETIS. Une ligne de transmission en forme de croix, comme illustré sur la Fig. 0.3 a été conçue pour minimiser la distance entre les deux tuilesets plus éloignés du CMP. Dans cette configuration, la distance maximale entre deux noeuds est de 80 mm. La forme en croix donne également un avantage supplémentaire, en fournissant une réponse en fréquence plate sur la bande passante de 20 GHz. Sur la gamme de fonctionnement, entre 20 et 40 GHz, l'atténuation attendue (issue des simulations) est de -10 dB à -42 dB entre les deux nœuds les plus éloignés et relativement linéaire sur toute la gamme de fréquence. Ceci peut paraitre variable mais comparativement au cas aérien qui connait beaucoup de variations fréquentielles et temporelles, on peut faire l'approximation ici de une certaine stabilité. On retrouve cette quasi-invariabilité en fréquence entre tous les autres noeuds, avec des atténuations plus faibles au fur

et à mesure que la distance décroît entre les deux nœuds considérés. Cela simplifie l'utilisation de l'OFDM puisque toutes les sous-porteuses subiront la même atténuation quelle que soit la fréquence utilisée dans le système. Seule la position relative entre les nœuds fera varier l'atténuation qui donc sera prévisible. Le choix des porteuses ne nécessitera pas ainsi de modifier la puissance de transmission, mais seulement le choix de l'ordre de modulation. Une ligne coplanaire en technologie CMOS à 0.25 micromètre fournie par NXP semiconductor est utilisée. Un mécanisme d'accès par transistor est préféré la celui d'un couplage capacitif classique pour accéder à la ligne de transmission depuis les têtes analogiques d'émission et de réception des tuilesets. Le signal est transmis entre la ligne guidée de transmission et la tête RF sans contact physique. Cette méthode réduit le phénomène des réflexions et de l'atténuation fluctuante sur différentes fréquences (désadaptation).

Chapitre 4 : Problème d'allocation de la bande passante d'un RF NoC

4.1 Interconnection OFDMA de WiNoCoD et motivation

Nous avons vu que chaque tuileset a un point d'accès à la ligne de transmission via un modulateur/démodulateur OFDMA. Grâce à la capacité de diffusion intrinsèque de l'OFDMA, chaque paquet envoyé n'a pas besoin d'être dupliqué pour être diffusé, comme toutes les transmissions effectuées par un utilisateur sont reçues par tous les autres (SWMR). Ceci augmente considérablement la capacité du système compte tenu de l'exigence particulière, dans le cas d'un système à cohérence de cache de répertoire, en termes de nombre de paquets de diffusion / multidiffusion. Une sous-porteuse sur un symbole OFDM peut être considérée comme un élément servant à la transmission de l'information (1,2, etc. bits sur la base de l'ordre de modulation utilisé BPSK, QPSK, etc.). Comme mentionné précédemment, la motivation principale derrière l'utilisation de l'OFDMA sur puce est sa capacité de reconfiguration, qui se réfère à la facilité de changer les utilisateurs de sous-porteuses, ou le nombre de porteuses par utilisateur, au cours du temps. Par conséquent, notre problème peut être formulé sous la forme de l'attribution des sous-porteuses au cours du temps (par exemple sur une durée de un ou plusieurs symboles OFDM) aux différentes files d'attente de transmission des tuilesets. Dans l'hypothèse où les files d'attente de réception en sortie de la ligne de transmission peuvent être vidées avec une rapidité élevée, en tout état de cause supérieure à celle d'arrivée des paquets, nous pouvons affirmer que l'allocation dynamique de sous-porteuses devrait diminuer les latences dans les files d'attente du côté de la transmission. Il est à noter que cette hypothèse est très légère, puisque cela signifie que les paquets reçus en sortie de

la ligne de transmission peuvent être transmis dans le réseau intra-grille d'un tuileset à haute vitesse, ce qui est le cas car le réseau en quadrillage est un réseau parallèle sur 32 ou 64 bits fonctionnant habituellement à plusieurs centaines de mégahertz de fréquence d'horloge. Il peut donc drainer 64 bits toutes les nanosecondes à 1 GHZ d'horloge, soit bien plus que des données sur 64 bits (ou un multiple de 64 bits inférieur à 10) arrivant de la ligne de transmission toutes les 51.2 ns. Par conséquent, dans ce projet, nous cherchons des mécanismes d'arbitrage de l'allocation des sous-porteuses afin de minimiser le retard dans les files d'attente de transmission des tuilesest, en entrée de la ligne de transmission, et les probabilités de dépassement de capacité des mémoires tampons ou files d'attente de l'émetteur des tuilesets vers la ligne de transmission. Les détails de la spécification des protocoles proposés dans le cadre de cette thèse, tels que la structure détaillée des bits d'un paquet, etc., ne sont pas compris dans ce travail de thèse. Un processus générique de mécanisme d'allocation de la bande passante est prévu, qui peut être utilisé pour tout type d'architecture massivement multicoeurs, au-delà du cas de WiNoCoD.

4.2 Ordonnancement dynamique de la bande passante pour les files d'attente parallèles

La problème présenté précédemment est référencé dans la littérature comme "l'ordonnancement multi-utilisateur multi-serveur" ou "l'ordonnancement des files d'attente parallèles". La recherche sur les politiques d'ordonnancement multi-utilisateurs pour le cas multi-serveurs intègre les domaines de la théorie des files d'attente, la théorie des réseaux, l'optimisation stochastique et les processus de décision markoviens. Considérant la nature de notre interconnexion OFDMA, nous allons étudier les algorithmes les plus efficaces en termes de politiques dynamiques d'ordonnancement. Ce type d'approches réassignent habituellement les serveurs aux files d'attente à chaque (ou chaque multiple) intervalle de temps, basé sur l'état instantané du système.

Pour le cas où il ya un seul serveur, qui doit être attribué à l'une des K files d'attente à chaque intervalle de temps (multi-utilisateur/serveur unique), il a été prouvé que la politique d'allocation prioritaire à la plus longue file d'attente (Longest Queue First: LQF) fournit la plus faible latence moyenne, sachant que les arrivées des paquets aux files d'attente sont indépendantes et identiquement distribuées. L'optimalité de cet algorithme pour le cas de plusieurs serveurs est toujours valable sous l'hypothèse d'arrivées indépendantes. L'inconvénient de cet algorithme est sa complexité de calcul. Il procède par itérations sur les N serveurs, et à chaque itération, le serveur effectue une itération pour toutes les files d'attente K. Après l'affectation, les longueurs des files d'attente sont mises à jour. Même si sa mise en œuvre dans WiNoCoD est impossible, en raison de la

puissance de calcul requise excessive et, plus important en raison du fait que l'algorithme a besoin de connaître le temps d'attente instantanée de chaque paquet dans le système, nous utilisons cet algorithme comme référence pour comparer les performances de nos politiques d'allocation de bande passante.

4.3 Allocation de la bande passante dans WiNoCoD

L'innovation de l'interconnexion RF de type OFDMA de WiNoCoD nécessite de développer de nouvelles techniques. La durée relativement longue des symboles OFDM à l'égard de temps d'arrivée des paquets dans les files d'attente d'émission des tuilesets, les longueurs des paquets et les exigences en termes de retard extrêmement strictes rendent cet environnement vraiment unique par rapport aux protocoles de communication existants, basés sur l'OFDMA. Par conséquent, dans cette section, nous introduisons certaines notions préliminaires liées à ce contexte particulier, et qui auront un impact sur l'allocation de la bande passante dans WiNoCoD.

4.3.1 Bloc de ressources et trames

La première notion que nous présentons est le bloc de ressources (Resource Block: RB), qui définit un groupe de sous-porteuses adjacentes sur un seul symbole. Considérant que nous avons 1024 sous-porteuses, il est évident qu'une granularité très fine pour allouer la bande passante avec une seule ou quelques sous-porteuses est coûteuse et inutile. Elle est notamment coûteuse en termes d'adressage et nécessitera l'échange de nombreux bits d'adresse pour informer les contrôleurs RF des tuilesets de leur allocation respective. Nous avons choisi de définir un RB pour servir exactement 1 paquet court (1 flit - 64 bits), en supposant que la modulation QPSK est utilisée par défaut. Un RB correspond donc à 32 sous-porteuses. Il est aussi à noter que comme le projet WiNoCoD prévoit l'utilisation de 1024 porteuses et 32 tuilesets, il y a en moyenne un RB de 32 porteuses disponible par tuileset. C'est une configuration qui pourra être utilisée par défaut, au lancement notamment, avec un RB d'émission pour chaque tuileset.

Le spectre de 20 GHz est donc divisé en RBs de 32 porteuses soit 625 MHz. Etudions maintenant les politiques de répartition des RB entre les tuilesets et le processus de reconfiguration associé. Comme nous l'avons examiné précédemment, les algorithmes efficaces d'allocation de bande passante nécessitent d'utiliser l'information de longueur des files d'attente (QSI). En outre, il est nécessaire que cette information soit récente, afin d'atteindre de faibles délais de livraison des paquets et de faibles tailles de files d'attente au niveau de l'émetteur d'accès à la ligne de transmission, géré par le contrôleur RF de chaque tuileset. Cependant, l'échange de l'information de QSI entre les tuileset à

chaque symbole n'est pas réaliste en raison des besoins de signalisation excessive qu'il provoquerait d'une part. D'autre part, des paquets peuvent arriver à la file d'attente d'émission pendant la durée d'un symbole OFDM de 51,2 ns. L'information de QSI n'est donc par définition jamais complètement à jour quand elle est envoyée. Nous envisageons un système, où les QSIs (ou d'autres indicateurs de trafic local, en tenant compte d'autres algorithmes possibles) sont diffusés par les tuilesets tous les T symboles et la nouvelle répartition des sous-porteuses est effectuée tous les T symboles. Grâce à la capacité de diffusion de l'interconnexion OFDMA, les QSIs qui sont envoyés par chaque tuileset peuvent être reçus par tous les autres. Nous appelons «trame» ce groupe de T symboles. Ainsi, à la fin, nous avons un système d'allocation en mode pipeline, c'est-à-dire décalé d'une trame au moins. L'allocation est calculée au début d'une trame selon des informations de QSI envoyées T symboles avant par les autres tuilsets, comme le montre la Fig. 0.7.

Dans le cas spécifique de WiNoCoD, où 1 symbole est d'environ 50 ns, le temps de calcul apparaît comme un paramètre important. Cette exigence temporelle stricte et peu orthodoxe fait de ce problème d'attribution, qui est examiné dans cette thèse, un travail original par rapport aux systèmes classiques. En effet, il n'est pas possible d'amortir sur une durée inférieure à celle d'un symbole, non seulement le temps des calculs d'un algorithme d'allocation, mais aussi le retard cumulé du temps de reconfiguration des sous-porteuses, du traitement des paquets, la synchronisation, le temps de propagation etc. D'où la nécessité de gérer le problème en trames de T symboles.

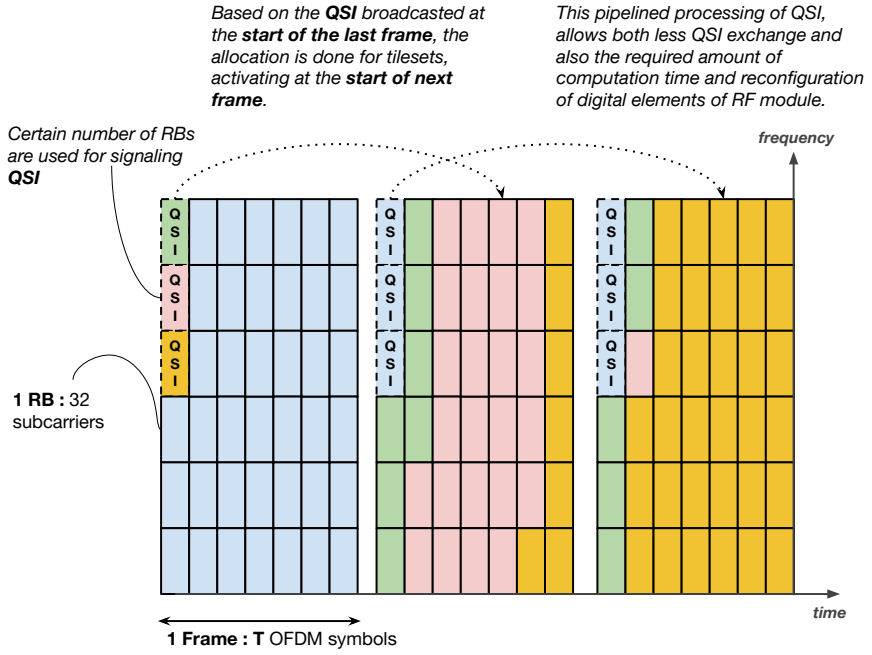


FIGURE 0.7: Les trames et blocs de ressources (RBs) dans WiNoCoD

4.3.2 Allocation décentralisée et centralisée

Les mécanismes d'arbitrage d'allocation des sous-porteuses que nous avons développé pour l'interconnexion WiNoCoD, peuvent être divisés en deux catégories : décentralisés et centralisés. Chacune des approches a des avantages et des inconvénients. Si une allocation centralisée est préférée, une seule unité centrale intelligente (Central Intelligent Unit: CIU), qui peut être soit un circuit numérique soit un microprocesseur simple, peut être installé à l'intérieur de la tête RF d'un tuileset (avant les modules IFFT/FFT). De cette façon, il peut utiliser les modulateur/démodulateur OFDM de ce tuileset. L'unité centrale intelligente doit diffuser sur la ligne guidée le nombre de RBs alloués à chaque tuileset. Ceci est une surcharge de signalisation supplémentaire évidente, ce qui n'est pas souhaitable afin d'épargner au maximum la bande passante sur la ligne guidée pour les échanges de données entre tuilesets.

Un autre inconvénient est la robustesse. D'une part si le CIU connaît une panne, toute la puce est en panne. Mais l'approche proposée ici peut permettre de choisir n'importe quel tuileset pour héberger le CIU, ce qui atténue légèrement cet effet (l'étude de capacité du dernier chapitre tempèrera cette affirmation en révélant qu'il faut mieux que le CIU soit au centre la la ligne guidée). D'autre part, nous avons positionné la réponse du CIU sur des sous-porteuses réservées sur quelques symboles avant la fin de la trame. Nous nous réservons $T_{configuration}$ symboles de temps pour que les processeurs de la tête RF des tuilesets puissent recevoir et démoduler le nombre de RBs alloués à chaque tuileset dans la trame suivante et reconfigurer leurs transmissions en conséquence. Nous proposons deux approches différentes pour l'attribution des RBs de chaque trame entre les tuilesets: l'allocation «en fréquence» ou «en temps».

Cependant, par moments, la somme des demandes de RB de tuilesets peut être beaucoup plus faible que le nombre total de RBs dans une trame. Dans ces cas là, il peut exister des RBs inutilisés. Considérant que des nouveaux paquets peuvent arriver dans les files d'attente d'émission des tuilesets pendant ces symboles inactifs, on répartit les RBs vides uniformément entre les tuilesets, ce qui permet à un tuileset d'utiliser un RB par symbole pour tout paquet qui arriverait pendant une trame. La nature numérique de la répartition des RBs en OFDMA rend ce processus trivial.

4.3.3 Encodage et signalisation des QSIs

Un autre aspect important est l'encodage des QSIs des tuilesets au début d'une trame. En cas de constellation QPSK, nous avons 2048 bits disponibles dans chaque symbole OFDM. Sur la base de nos simulations, l'utilisation de 8 bits par tuileset et par trame

pour coder les QSI s'est révélé offrir le meilleur compromis en termes de performance globale. Avec 8 bits, il est possible de coder 256 niveaux différents. Bien sûr, la file d'attente vide (0 QSI) doit aussi être codée. En regardant les résultats de simulation, nous avons vu que le nombre de flits dans une file d'attente de transmission dépasse très rarement 255 dans les scénarios évalués. Par conséquent, toutes les valeurs de longueur de QSI entre 0 et 255 sont directement codées, et si le nombre de flits est supérieur à 255, il est codé par 255. Bien sûr, on peut choisir différentes approches et granularité de codage des QSI. Chaque valeur de QSI pourrait aussi représenter plusieurs flits. Il y a un compromis évident entre une surcharge de signalisation de QSI et la précision de l'algorithme d'allocation. Cependant, à partir de nombreuses possibilités, et en prenant en compte des configurations d'interconnexion réalistes, nos simulations ont révélé que l'encodage sur 8-bits proposé semble viable et efficace.

Les contraintes très spécifiques des interconnexions sur puce en OFDMA imposent de respecter certaines exigences. Compte tenu de la durée relativement longue des symboles par rapport aux exigences de retard sur les paquets, même une latence de quelques symboles est importante, contrairement aux cas classiques d'utilisation de l'OFDM. Dans ce contexte particulier, la dynamique de la file d'attente peut changer radicalement en quelques symboles. Comme expliqué ci-dessus, en raison des contraintes de temps de calcul et de surcharge de signalisation, la répartition est effectuée en ≪pipeline≫ entre trames. En d'autres termes, la répartition se fait avec sur des QSI ayant T symboles de retard. Compte tenu du trafic temporellement très hétérogène, nous introduisons la notion de "QSI Définitif" (Definitive QSI - DQSI). Au début de chaque trame, avant de coder son QSI sur les sous-porteuses réservées, chaque tuileset connaît déjà le nombre de RBs attribué dans la trame actuelle. Par conséquent, plutôt que de coder directement le QSI, il soustrait le nombre de RBs déjà alloué de son QSI courant. Une fois que le nombre minimum de flits dans la file d'attente est déterminé, nous pouvons également prendre en compte le nombre de flits qui arriveront au cours du processus d'allocation (sur la durée de la trame actuelle). Pour ce faire, les tuilesets doivent estimer le nombre de flits qui arriveront pendant une trame. Cela peut se faire en utilisant un filtre à moyenne mobile. Nous appelons ce modèle comme "QSI prédit" (*Expected Queue State Information - EQSI*).

Avant de présenter nos algorithmes, il est essentiel d'introduire les techniques et les scénarios que nous allons utiliser pour les évaluer. Nous utilisons OMNeT++, un simulateur d'événements discrets qui est utilisé largement auprès la communauté réseau sur puce. OMNeT++ est utilisé pour émuler la dynamique des files d'attente de transmission des 32 tuilesets. Les simulations sont exécutées avec un pas temporel égal à celui d'un symbole OFDM, qui est donc l'unité atomique temporelle. L'objectif est l'amélioration des performances de l'interconnexion RF en termes de latence de délivrance des paquets

présents dans les files d'attente d'émission des tuilest. la première métrique d'intérêt que nous essayons d'améliorer est la latence moyenne. La latence moyenne d'un réseau sur puce, pour différents modèles de trafic et des charges de trafic différentes, est la mesure la plus significative de sa performance. L'importance du retard des paquets dans un NoC est particulièrement sensible pour les applications temps réel. Par conséquent, notre interconnexion RF est testée en premier pour cette métrique. En outre, pour chaque simulation; nous traçons la courbe de la probabilité de dépassement de limite de latence, qui indique la probabilité qu'un paquet dépasse une limite de latence spécifiée, D_0 : $P(D > D_0)$. La troisième métrique d'intérêt, est la courbe de probabilité de dépassement de la limite de longueur de queue : $P(L > L_0)$.

Cependant, pour tester les performances du NoC, les benchmarks de référence actuels en termes de génération de trafic sur la ligne guidée sont loin de représenter la charge générée par les futurs algorithmes qui seront embarqués sur des architectures de plusiuers milliers de cœurs de traitement. Par conséquent, pour évaluer la validité de notre interconnexion OFDMA et des algorithmes d'arbitrage des ressources fréquentielles proposés, nous avons choisi d'utiliser des modèles de trafic stochastiques et synthétiques.

Chapitre 5 : Algorithmes d'Allocation de bande passante de WiNoCoD

5.1 Allocation Série avec QSI Directe

Le première algorithme que nous proposons est relativement simple à mettre en œuvre. Après que les valeurs des QSIs soient diffusées dans la ligne guidée, puis acquises par chaque tuileset, le contrôleur RF de ces derniers alloue simplement les RBs en série dans la prochaine trame (matrice de RBs), en itérant sur les valeurs de QSI issues de chaque tuilesets (puisque chaque tuileset peut décoder toutes les valeurs émises par tous les autres tuilesets). L'algorithme se termine lorsqu'il n'y a plus de RBs vides ou si toutes les demandes des tuilesets sont servies. Pour le cas où le QSI total de tous les tuilesets est inférieur au nombre total de RBs dans la trame, les RBs restants sont partagés entre les tuilesets en utilisant la matrice d'allocation par défaut. On peut se demander si l'allocation des RBs peut également être faite unité par unité, en itérant sur chaque tuilesets pour chaque RB à allouer ou avec une approche complètement différente. Cependant, limiter le nombre d'itérations est essentiel. Considérant que le budget temporel est de quelques centaines de cycles de processeur pour le calcul de l'arbitrage de la bande passante entre les tuilesets, itérer sur 32 tuilesets n'est réalisable qu'une seule fois.

Comme l'algorithme d'allocation de QSI en série est de faible complexité (de calcul), on peut le dupliquer dans chaque tuileset pour un surcoût négligeable et on s'affranchit des problèmes déjà évoqués, inhérents à l'utilisation d'une unité centralisée. Cependant, nous effectuons la même expérimentation que pour le cas décentralisé, pour des raisons de cohérence. Tout au long de cette thèse, quand nous comparons l'approche centralisée à l'approche décentralisée, nous utilisons toujours un temps de reconfiguration $T_{configuration} = 2$ symboles. Par exemple, lorsque l'on compare une longueur de trame décentralisée de 4 symboles (principalement en raison du temps de calcul), nous utilisons une longueur de trame de 6 symboles pour l'approche centralisée. Rappelons que, dans l'approche centralisée, nous avons des RBs réservés pour la transmission des paquets de réponse.

Par exemple, les Fig. 0.8 et Fig. 0.9 montrent les variations de la latence moyenne en fonction du taux d'injection dans le cas d'un trafic réaliste non-uniforme (DPBPP) pour l'allocation série, avec l'approche centralisée et décentralisée, pour différentes longueurs de trame.

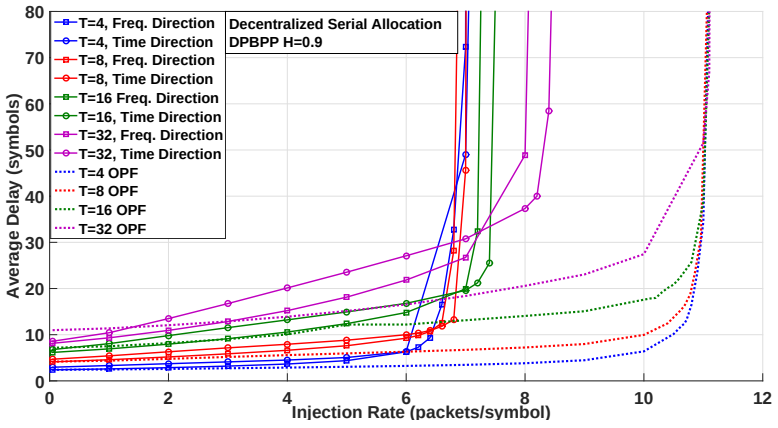


FIGURE 0.8: Latence moyenne en fonction de l'augmentation du taux d'injection pour un trafic réaliste non-uniforme (DPBPP) pour l'allocation serie avec l'approche décentralisée pour différentes longueurs de trame.

5.1.1 Allocation Série à Deux Itérations

L'algorithme serie ne permet pas d'approcher de près les performances idéales en termes de capacité du réseau. Une allocation de bande passante très fréquente, donc avec des longueurs de trames très courtes, provoque une augmentation significative de la latence moyenne. Nous avons observé que cet effet contradictoire est dû au faible nombre de RBs à allouer pour des longueurs de trames courtes. En utilisant la même priorité pour tous les tuilesets, certains vont prendre les RBs, et vont priver de ressources d'autres tuilesets qui en auraient besoin. Afin de résoudre ce problème d'équité, et d'augmenter

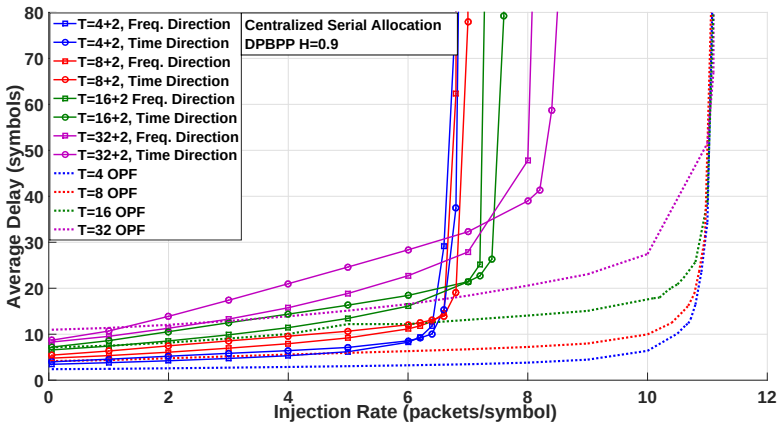


FIGURE 0.9: Latence moyenne en fonction de l'augmentation du taux d'injection sous le trafic réaliste non-uniforme (DPBPP) pour l'allocation serie avec l'approche centralisée pour différentes longueurs de trame.

l'équité entre les files d'attente de transmission des tuilesets, nous avons établi une modification de l'algorithme série. Cette fois, la répartition des RBs est effectuée en 2 étapes, comme le nom de l'algorithme l'indique. Tout d'abord après avoir reçu les QSIs de chaque tuileset, les contrôleurs RF des tuilesets les additionnent et divisent le résultat obtenu par le nombre de tuilesets afin d'obtenir le "QSI moyen" de la trame courante. A la première étape de l'algorithme, une itération sur les 32 tuilesets est effectuée comme dans l'allocation série précédente. Toutefois, lors de cette première itération, seulement les tuilesets qui ont un QSI plus grand que le "QSI moyen" courant peuvent prendre des RBs. Après cette première itération, les valeurs de QSI sont mises à jour en soustrayant le nombre de RBs déjà alloués à cette itération. À la deuxième étape, la même itération est effectuée sur les valeurs mises à jour de QSI des tuilesets, mais cette fois avec l'algorithme d'allocation série par défaut. De cette manière, nous nous assurons que les tuilesets qui ont vraiment besoin de RB vont obtenir leur part. Cette extension de l'algorithme d'allocation de QSI série directe peut être considérée comme une tentative pour compenser les déséquilibres dans les files d'attente.

Les Fig. 0.10 et Fig. 0.11 montrent la variation de la latence moyenne en fonction du taux d'injection sous le trafic réaliste non-uniforme (DPBPP) pour l'allocation série avec deux itérations, dans le cas de l'approche centralisée et décentralisée, pour différentes longueurs de trame.

5.1.2 Allocation Série avec DQSI et EQSI

Nous avons vu qu'il était possible d'augmenter la capacité du réseau en utilisant une modification de l'algorithme série, en le passant à deux iterations. Dans l'approche décentralisée, chaque tuileset calcule sa propre valeur de DQSI ou EQSI en utilisant le

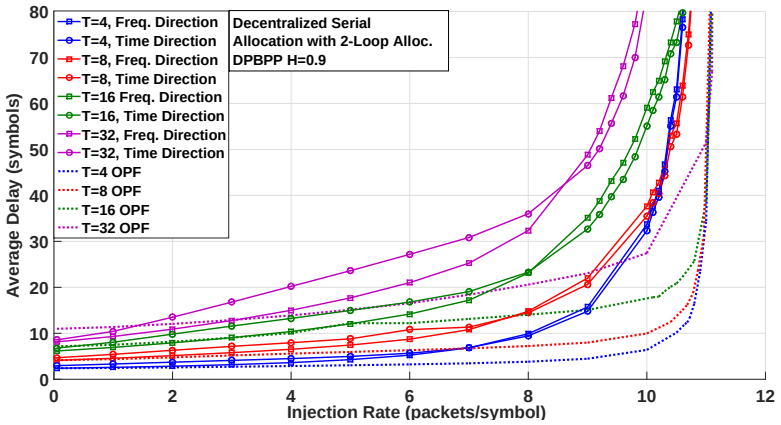


FIGURE 0.10: Latence moyenne en fonction de l'augmentation du taux d'injection pour un trafic réaliste non-uniforme (DPBPP) pour l'allocation serie (2 iterations) avec l'approche décentralisée pour différentes longueurs de trame.

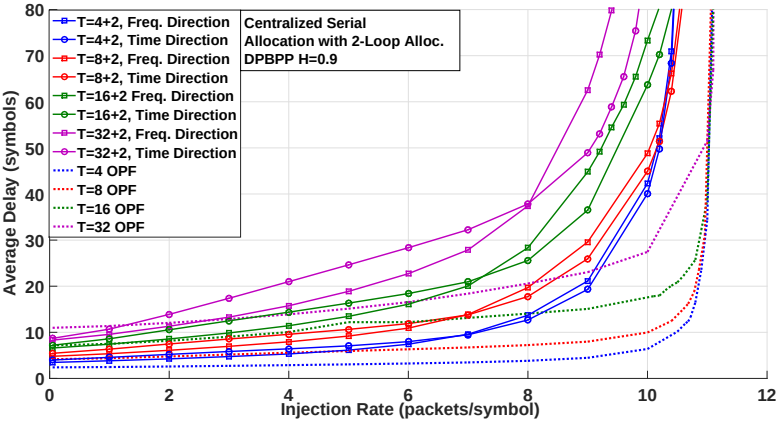


FIGURE 0.11: Latence moyenne en fonction de l'augmentation du taux d'injection pour un trafic réaliste non-uniforme (DPBPP) pour l'allocation serie (2 iterations) avec l'approche centralisée pour différentes longueurs de trame.

nombre de RBs actuellement alloués et/ou instantanés. Ensuite, cette valeur est diffusée sur le premier symbole de la trame suivante. Mais dans l'approche centralisée, seuls les tuilesets diffusent leurs valeurs de QSI instantanés et c'est le CIU qui est responsable du calcul du DQSI/EQSI de chaque tuileset.

Les Fig. 0.12 et Fig. 0.13 montrent la variation de la latence moyenne en fonction du taux d'injection dans le cas d'un trafic réaliste non-uniforme (DPBPP) pour l'allocation série avec l'algorithme DQSI/EQSI, pourl'approche centralisée et décentralisée, et pour différentes longueurs de trame.

En comparant ces résultats de l'allocation série simple et directe à ceux de l'allocation

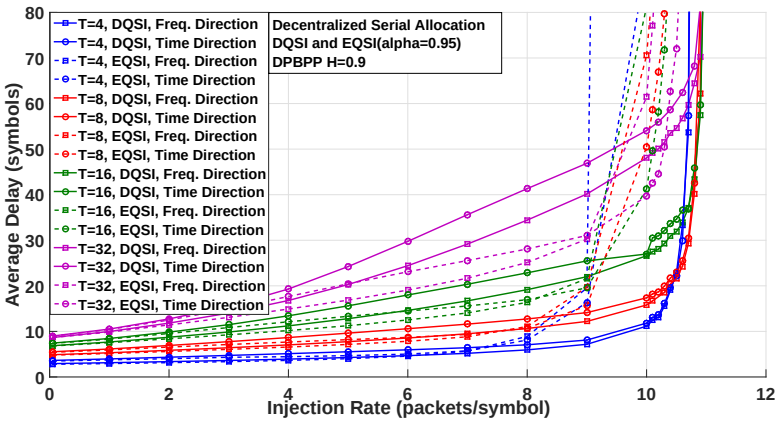


FIGURE 0.12: Latence moyenne en fonction de l'augmentation du taux d'injection pour un trafic réaliste non-uniforme (DPBPP) pour l'allocation serie (DQSI/EQSI) avec l'approche décentralisée pour différentes longueurs de trame.

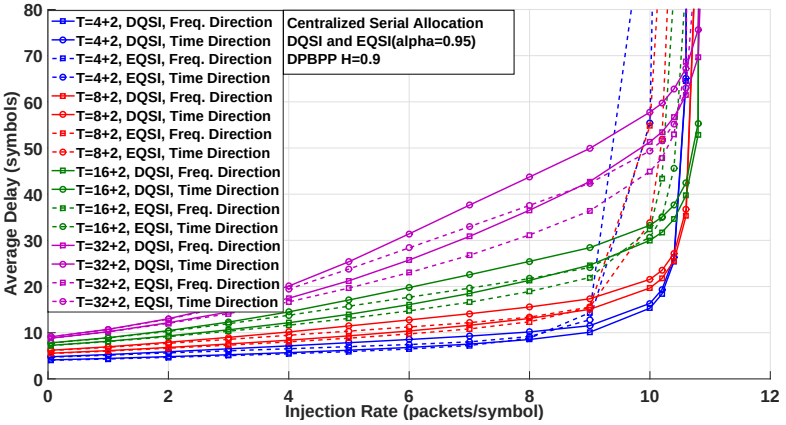


FIGURE 0.13: Latence moyenne en fonction de l'augmentation du taux d'injection pour un trafic réaliste non-uniforme (DPBPP) pour l'allocation serie (DQSI/EQSI) avec l'approche centralisée pour différentes longueurs de trame.

en deux itérations, le résultat le plus remarquable est de constater la diminution substantielle de la latence moyenne, simplement en utilisant DQSI. Il n'élimine pas seulement l'allocation de ressources inutiles, mais il compense également l'iniquité en raison du petit nombre de RBs.

5.2 Allocation QPS

5.2.1 Allocation proportionelle aux longueurs des files d'attente

Dans la section précédente, nous avons proposé et examiné la faisabilité de probablement l'option la plus simple, sous la forme de l'algorithme série avec QSI directe, où les tuilesets diffusent leur QSI et les RBs sont arbitrés séquentiellement en une seule (ou deux)

itération pour la prochaine trame. La principale motivation derrière cette pratique est d'être en mesure d'effectuer l'opération d'allocation en quelques centaines de nanosecondes, car s'y ajouteront tous les autres retards cumulatifs supplémentaires découlant de la propagation, la synchronisation, la transformation, etc.. C'est une contrainte stricte imposée, par les particularités de l'interconnexion OFDMA et de l'environnement sur puce quand une bande passante de 20 GHz ou plus est utilisée (ce qui diminue la durée des symboles OFDM à nombre de porteuses constant).

Dans cette section, nous évaluons la viabilité, dans notre contexte, de l'ordonnancement proportionnel aux longueurs des files d'attente (*Queue Proportional Scheduling-QPS*). Il ajoute une complexité supplémentaire limitée par rapport à l'allocation série. L'idée est d'allouer les RBs dans les trames de manière proportionnelle à la valeur des QSIs des tuilesets.

Les Fig. 0.14 et Fig. 0.15 montrent les variations de la latence moyenne en fonction du taux d'injection dans le cas d'un trafic réaliste non-uniforme (DPBPP) pour l'allocation QPS, pour l'approche centralisée et décentralisée, et pour différentes longueurs de trame.

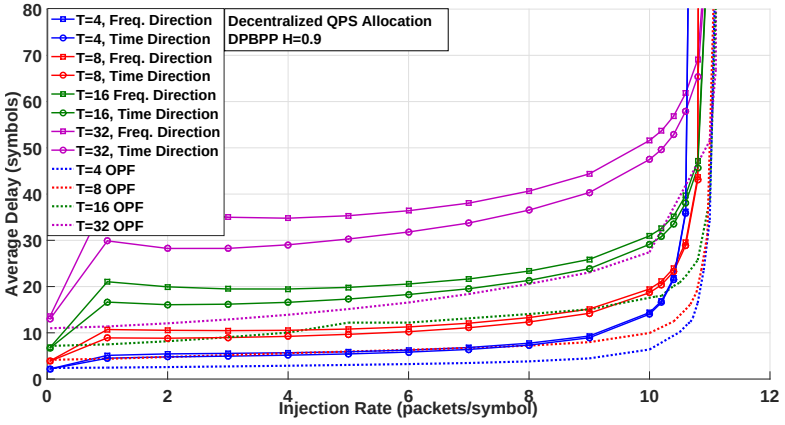


FIGURE 0.14: Latence moyenne en fonction de l'augmentation du taux d'injection pour un trafic réaliste non-uniforme (DPBPP) pour l'allocation QPS avec l'approche décentralisée pour différentes longueurs de trame.

L'algorithme QPS offre une bien meilleure performance pour les grandes valeurs de taux d'injection, en particulier à proximité de la limite de capacité du medium, par rapport à l'allocation série directe. En effet, allouer les RBs proportionnellement aux QSIs, élimine naturellement la probabilité que certains nœuds «épuisent» tous les RBs d'une trame et «affament» les autres nœuds. Même si l'allocation série avec DQSI fournit de meilleurs résultats (en particulier pour des trames de courte longueur), QPS apparaît comme une approche plus évolutive et équitable.

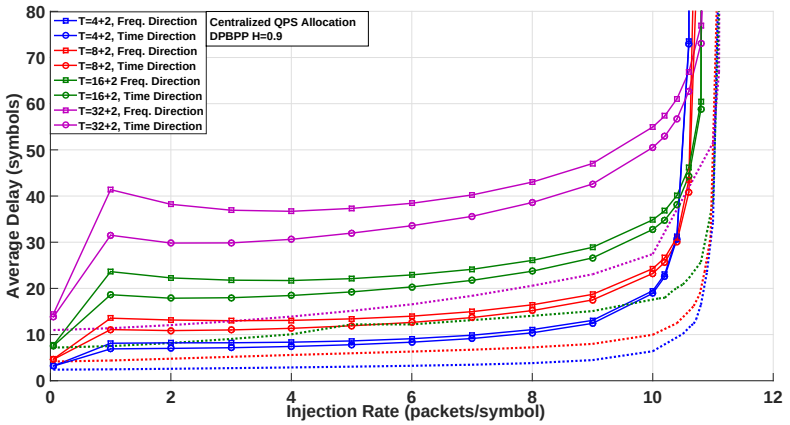


FIGURE 0.15: Latence moyenne en fonction de l'augmentation du taux d'injection pour un trafic réaliste non-uniforme (DPBPP) pour l'allocation QPS avec l'approche centralisée pour différentes longueurs de trame.

5.2.2 Allocation QPS avec DQSI et EQSI

Nous cherchons à accroître la performance de l'algorithme QPS en utilisant DQSI ou EQSI, en particulier pour les taux d'injection faibles. Comme pour les cas précédents, pour les approches décentralisées et centralisées, le calcul de DQSI et EQSI diffère.

Les Fig. 0.16 et Fig. 0.17 montrent la latence moyenne en fonction du taux d'injection dans le cas d'un trafic réaliste non-uniforme (DPBPP) pour l'allocation QPS (avec DQSI/EQSI), pour l'approche centralisée et décentralisée, et pour différentes longueurs de trame.

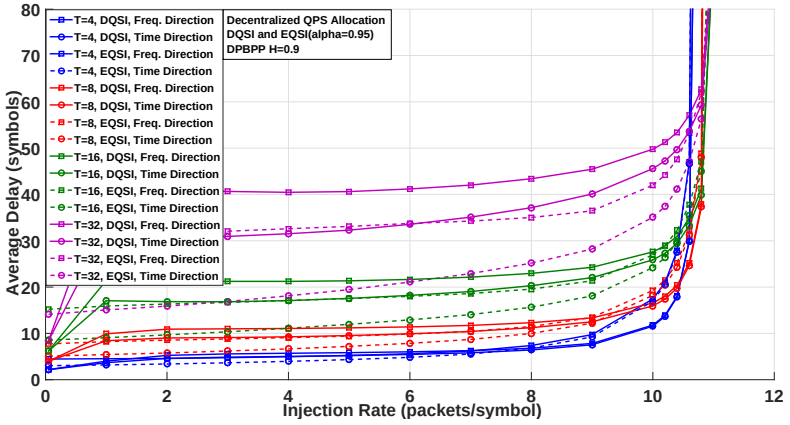


FIGURE 0.16: Latence moyenne en fonction de l'augmentation du taux d'injection pour un trafic réaliste non-uniforme (DPBPP) pour l'allocation QPS (DQSI/EQSI) avec l'approche décentralisée pour différentes longueurs de trame.

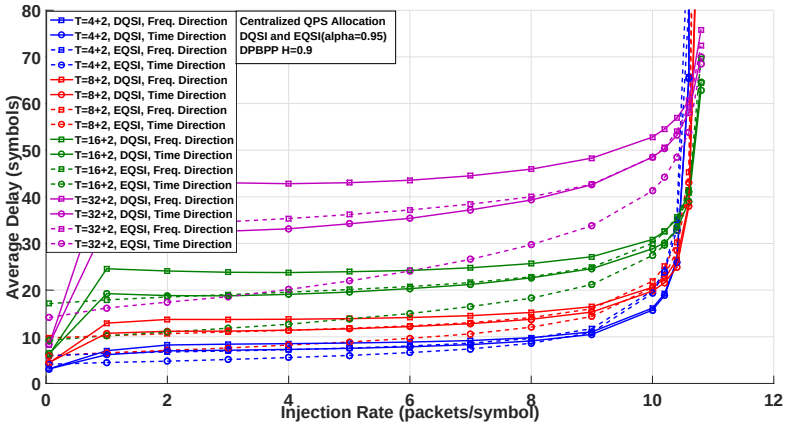


FIGURE 0.17: Latence moyenne en fonction de l'augmentation du taux d'injection pour un trafic réaliste non-uniforme (DPBPP) pour l'allocation QPS (DQSI/EQSI) avec l'approche centralisée pour différentes longueurs de trame.

5.2.3 Classification arbresencte des algorithmes d'allocation proposés

La Fig. 0.18 propose une classification des algorithmes introduits dans cette thèse sous forme d'arbre. A gauche sont représentés les approches d'allocation statiques (comme celui pouvant être utilisé par défaut avec une allocation équilibrée entre tous les tuilestes à 32 sous-porteuse chacun). A droite figurent les algorithmes d'allocation de bande passante dynamique proposés dans cette thèse. Ceux de ce chapitre 5 sont basées sur une file d'attente de transmission unique ("single queue") et se répartissent en deux grandes familles : série et QPS. Dans le chapitre 6, nous présentons un nouveau type d'algorithmes avec deux files d'attente de transmission, l'une pour les paquets courts de 1 flit (paquet de contrôle de cache et en-tête d'un paquet long) et l'autre pour les données des paquets longs ou "payloads" (ligne de cache). Ces derniers algorithmes sont illustrés en grisé sur l'extrême droit de la Fig. 0.18.

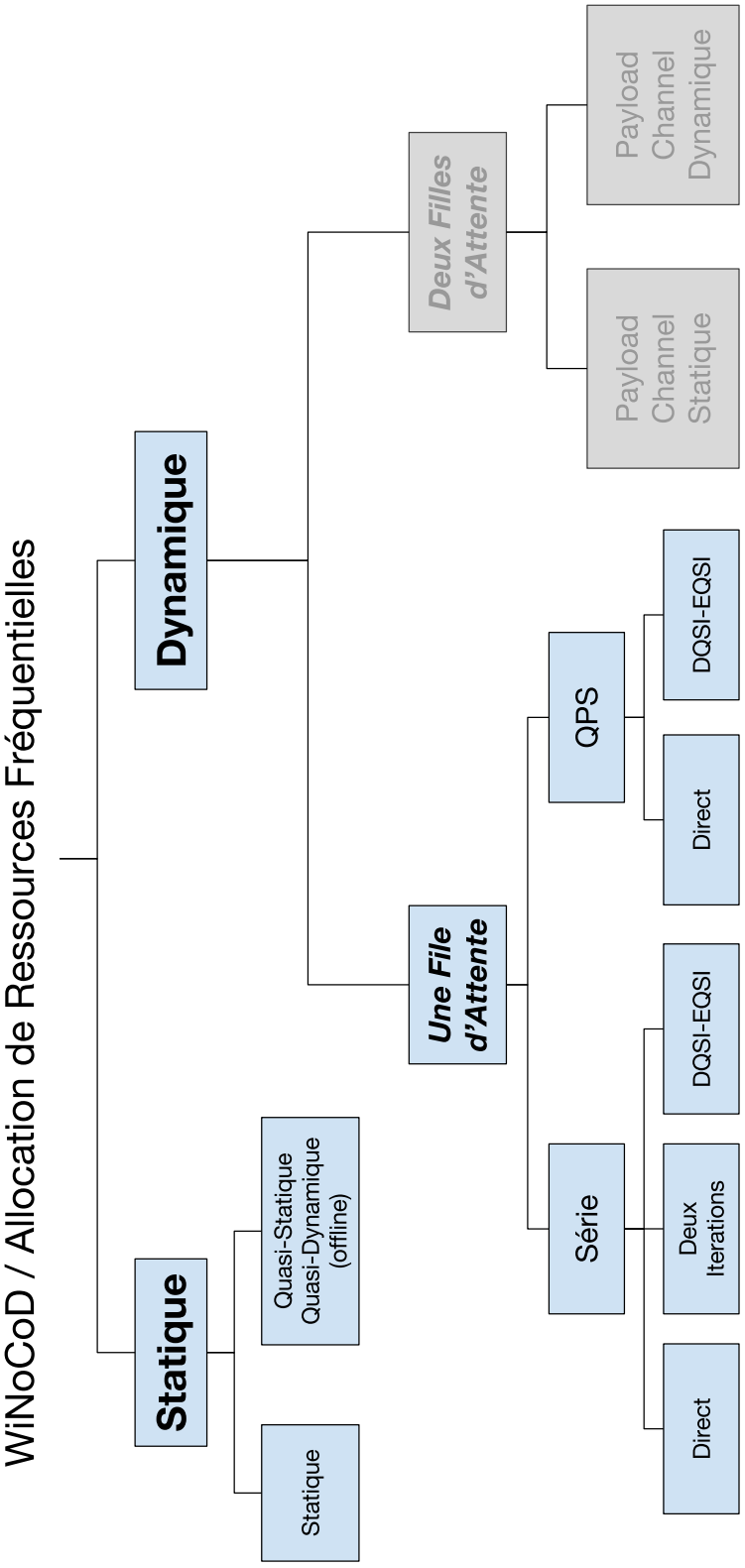


FIGURE 0.18: Arbre de classification des algorithmes d'allocation des ressources fréquentielles proposés dans cette thèse.

Chapitre 6 : Algorithme «Payload Channel»

Le trafic de cohérence de cache sur une puce à grand nombre de coeurs a des caractéristiques très spécifiques par rapport aux systèmes classiques. Comme mentionné précédemment, une de ces caractéristiques est la longueur des paquets bimodaux. Ce mécanisme de cohérence de cache pour les architectures à plusieurs milliers de cœurs avec mémoire partagée impose un nombre élevé d'échange de messages de diffusion, et par conséquent une charge de trafic sur le la ligne de transmission RF filaire, ce qui est le prix à payer pour l'évolutivité. Il existe deux types de paquets de cohérence de cache circulant dans un réseau sur puce :

- Des paquets de contrôle courts (demandes pour lire une ligne d'adresse etc.)
- Des paquets longs qui portent les lignes de cache ou charge utile («payload» / demandes pour écrire à une ligne d'adresse, réponse pour la lecture d'une ligne d'adresse, etc.)

Dans le cadre de cette étude, la longueur d'un paquet court est choisie et fixée à 64 bits, ce qui peut changer selon l'architecture. Les paquets longs sont composés d'un en-tête de commande (header) de 64 bits, qui contient les informations nécessaires à la communication, comme les identificateurs (ID) de destination et de source, le type des paquets, etc., et les données en ligne de cache (charge utile ou «payload»). La taille de la ligne de cache est ici choisie et fixée à 64 octets (512 bits), soit des paquets longs de 576 bits. Compte tenu de la modulation QPSK utilisée par défaut, en cas d'attribution statique et uniforme de 32 sous-porteuses par tuileset, chaque tuileset reçoit ou émet 1 RB de 64 bits par symbole, un paquet long prend 9 symboles pour être transmis, même sans délai d'attente dans la file d'attente d'émission. Considérant qu'un symbole OFDM dure un peu plus de 50 ns, on peut comprendre le goulot d'étranglement que les paquets longs créent pour l'interconnexion sur puce.

La taille de la ligne de cache a un impact significatif sur la performance du système de mémoire partagée. En cas de mémoires caches très grandes, utiliser de grandes lignes de cache est essentiel. Plus les lignes de cache sont grandes, plus cela diminue le taux de défaut de cache en général. Nous pouvons prédire que la meilleure performance sera acquise pour de plus grandes lignes de cache dans les architectures à milliers de coeurs avec mémoire partagée. En outre, une interconnexion reconfigurable et efficace, donc à faible latence, amortit la difficulté de transport de grandes lignes de cache, ce qui augmente les performances du système.

C'est en prenant tous ces aspects en considération que nous proposons un nouvel algorithme d'allocation de la bande passante pour l'interconnexion RF OFDMA de WiNoCoD.

L'objectif est de diminuer considérablement le temps de latence des paquets longs, en transmettant les lignes de cache ou charges utiles (payload) en un seul symbole. Sont exploités ici la capacité de diffusion intrinsèque et de reconfigurabilité de l'OFDMA. Ici, nous choisissons de considérer une taille différente pour les paquets longs, afin de simplifier notre explication, mais aussi parce que les paquets longs de charge utile ont avantage à regrouper un grand nombre de flits. A la différence des lignes de cache utilisés dans la première phase du projet (64 octets - 512 bits), cet algorithme est optimisé spécialement pour les lignes de cache 256 octets (2048 bits).

Dans ce cas, l'algorithme statique transmet les paquets longs dans une durée minimale égale à 33 symboles même sans délai d'attente en file d'attente (plus de 1650 ns), qui est très prohibitif.

6.1 : Algorithme «Payload Channel» avec Bande Passante de Base Statique

Notre algorithme repose sur l'idée de transmettre des charges utiles de 2048 bits de paquets longs dans un seul symbole OFDM, en utilisant des 1024 sous-porteuses modulées en QPSK (toute labande passante). Les paquets de contrôle ont une taille de 64 bits (un seul flit), ce qui correspond à 70-80% de tous les paquets et les paquets longs ont un flit d'en-tête qui précède la charge utile (ligne de cache). Notre algorithme original «payload channel» exploite ces bits indicateurs pour permettre la transmission de la charge utile des paquets longs dans un seul symbole OFDM, sans utiliser de surcharge de signalisation supplémentaire.

Initialement, 32 sous-porteuses sont allouées à chaque tuileset, et nous les nommons canaux de base (*home channel*). Comme la modulation QPSK est utilisée, ces canaux de base peuvent transporter 1 flit par symbole, soit un paquet court de contrôle ou un en-tête de paquet long. Tout d'abord, à la différence des architectures et solutions précédemment présentées, chaque tuileset a deux files d'attente de transmission différentes au niveau de leurs interfaces RF. Chaque fois qu'un nouveau paquet long arrive à l'interface RF pour être émis sur l'interconnexion RF, il est segmenté en deux sous-parties :

- Son en-tête, qui est mis en mémoire dans la file d'attente primaire de paquets courts
- Sa charge utile ou "payload", qui est mise en mémoire la file d'attente secondaire de charge utiles à la manière d'une FIFO.

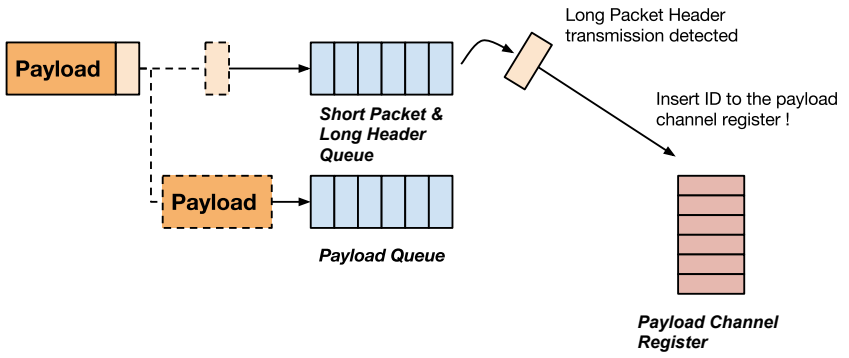


FIGURE 0.19: Deux files d'attente de transmission séparées pour l'algorithme «payload channel».

Ceci est illustré sur la Fig. 0.19. La motivation de l'emploi de deux files d'attente séparées pour les en-têtes et paquets courts d'une part, et les charges utiles d'autre part, est d'éviter l'incohérence entre les en-têtes transmis et de leurs charges utiles, mais aussi de permettre la transmission d'un nouveau paquet avant de transmettre la charge utile d'un paquet précédent.

Quand un tuileset transmet un en-tête de paquet long sur son canal de base (home channel), chaque tuileset peut le recevoir et décoder son contenu, grâce à la capacité de diffusion intrinsèque de l'interconnexion OFDMA. Chaque tuileset peut utiliser de simples unités de traitement pour vérifier les bits indicateurs des paquets et ainsi déterminer s'il s'agit d'un paquet long ou non. Quand les contrôleurs RF des tuilesets détectent un en-tête de paquet long, ils enregistrent le tuileset-ID correspondant (tuileset qui souhaite transmettre sa charge utile dans un seul symbole) en insérant l'ID dans le *registre de la charge utile* (*payload register*). Le Registre de la charge utile est une file d'attente simple de type FIFO au niveau de l'interface RF de chaque tuileset, qui met en mémoire les IDs des tuilesets qui veulent transmettre des charges utiles. Le contenu des registres de la charge utile est identique dans tous les tuilesets, afin d'éviter des incohérences.

Toutefois, l'acquisition du flit d'en-tête, le traitement et la reconfiguration de la bande passante prennent un certain temps en raison des délais de propagation, de synchronisation et de calcul des retards. Par conséquent, nous avons tenu compte d'un temps d'attente de 1 symbole (50 ns) pour l'ensemble de ces retards et l'activation de l'algorithme. Notez également que, plusieurs en-têtes de paquets longs de différents tuilesets peuvent être détectés dans un symbole. Leurs identifiants sont enregistrés dans les registres de la charge utile via leur Tuileset-ID reçus dans le même symbole. Au début de chaque symbole, les tuilesets contrôlent leurs registres de charge utile. Si le registre de la charge utile est vide, cela signifie qu'il n'y a pas de tuileset qui veut actuellement transmettre

une charge utile. La transmission reprend alors sa configuration par défaut d'un RB par tuileset. Toutefois, si le registre de la charge utile n'est pas vide, le premier ID dans le registre transmet sa charge utile en prenant la place des canaux de base. Le système retourne à la configuration de canaux de base, que si il n'y a pas tuileset reste dans le registre de charge utile. Cette procédure est illustrée sur la Fig. 0.20 pour un scénario.

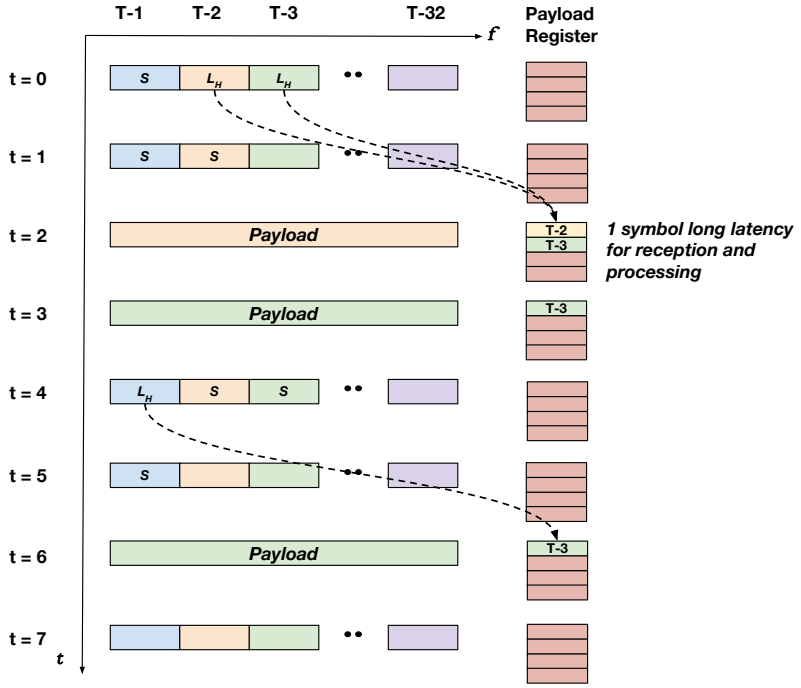


FIGURE 0.20: La procédure de transmission pour l'algorithme ≪payload channel≫ est illustrée pour un scénario.

Fig. 0.21 montre la latence moyenne de livraison des paquets en fonction de l'augmentation du taux d'injection pour notre algorithme ≪payload channel≫ et un cas statique (où chaque tuileset a un RB dans chaque symbole), sous trafic Poissonien uniforme. Nous avons également tiré une approximation analytique de la latence moyenne de cet algorithme, en utilisant les principes de la théorie des files d'attente de l'état de l'art. Notre algorithme innovant peut diminuer la latence moyenne jusqu'à 10 fois dans certains cas, grâce à son mécanisme simple, mais efficace.

6.2 : Algorithme ≪Payload Channel≫ avec Bande Passante de Base Dynamique

Le déséquilibre spatial du trafic de cohérence de cache a été discuté précédemment. Notre algorithme ≪payload channel≫ a besoin d'une modification afin de faire face à cette situation. À cette fin, nous proposons l'algorithme ≪ payload channel ≫ avec une

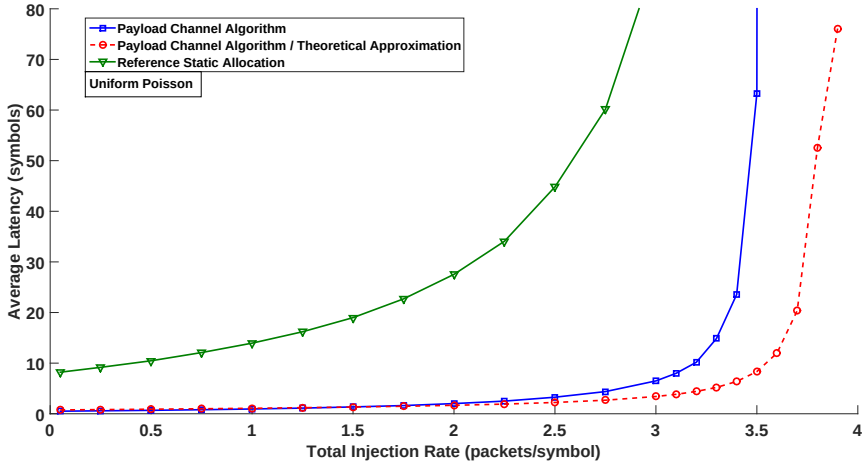


FIGURE 0.21: Latence moyenne en fonction de l'augmentation du taux d'injection pour un trafic poisson non-uniforme pour l'algorithme «payload channel».

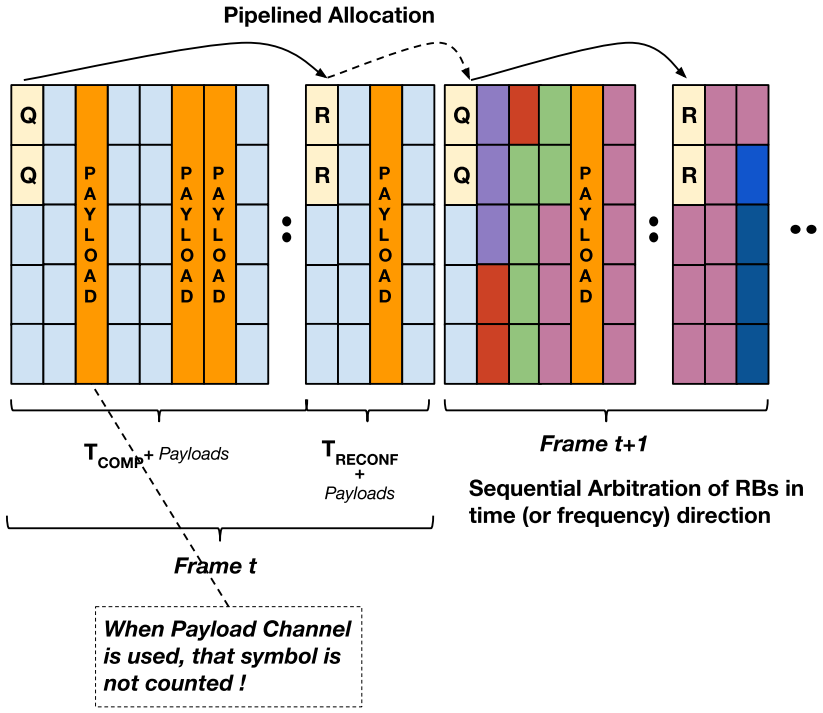


FIGURE 0.22: Arbitrage de RBs dans une trame pour l'algorithme «payload channel» dynamique.

bande passante de base dynamique, en fusionnant l'algorithme « payload channel » avec l'algorithme QPS centralisé, qui a été expliqué dans le chapitre 5. L'idée est de changer le nombre de canaux de base de chaque tuileset, à l'aide d'une unité centrale intelligente (CIU), de la même manière que dans les mécanismes dynamiques d'algorithmes précédents. De la même manière qu'évoqué précédemment, sur le premier symbole de chaque trame, chaque tuileset diffuse son QSI. Puis le CIU calcule les QSIs prédit (EQSI) des tuilesets, et en déduit l'allocation des RBs (canaux de base dans ce cas) pour les tuilesets selon l'algorithme QPS. Cependant, dans le cas où le canal de charge utile n'est pas utilisé, selon les messages envoyés en réponse par le CIU sur certains symboles et sous-porteuses pré-déterminés, les tuilesets reconfigurent leur allocation des sous-porteuses pour les canaux de base à la trame suivante. Cette procédure est illustrée en détail sur la Fig. 0.22.

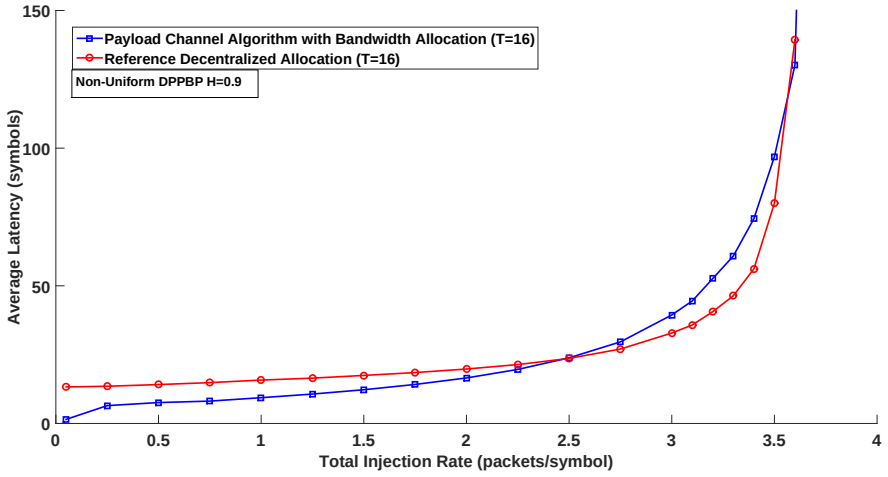


FIGURE 0.23: Latence moyenne en fonction de l'augmentation du taux d'injection un trafic réaliste non-uniforme (DPBPP) pour l'algorithme «payload channel» dynamique.

Fig. 0.23 montre l'évolution de la latence moyenne en fonction de l'augmentation du taux d'injection pour l'algorithme «payload channel» dynamique et l'algorithme QPS simple, dans le cas d'un trafic DPBPP non-uniforme, pour une longueur de trame de 16 symboles. Nous remarquons que nous pouvons diminuer la latence moyenne de paquets de manière drastique en exploitant les longueurs bimodale des paquets sur puce.

Chapitre 7 : Choix Dynamique de l'Ordre de Modulation

En règle générale pour les communications numériques, on sait qu'il existe une relation exponentielle entre la puissance de transmission et le nombre de bits par symbole transmis, pour un taux d'erreur donné ou un rapport signal à bruit donné. Un autre avantage de l'utilisation de l'OFDMA dans WiNoCoD est la possibilité de modifier les ordres de

modulation dynamiquement sur différents symboles et différentes sous-porteuses. Dans le cadre du projet WiNoCoD, quatre ordres de modulation ont été considérés: BPSK, QPSK, 16-QAM et 64-QAM, soit 1,2,4 et 6 bits par sous-porteuse respectivement. Dans notre étude, nous utilisons l'équation exponentielle, qui repose sur la relation issue de la théorie de l'information entre ordre de modulation et puissance. Cela permet de s'affranchir d'un taux d'erreur binaire spécifique, mais implique aussi l'utilisation de techniques de codage de canal qui ne sont pas considérées dans le cadre de cette étude. En outre, nous avons décidé d'employer les ordres de modulation jusqu'à 256-QAM, incluant également 8-PSK, 32-QAM et 128-QAM.

Nous avons développé deux algorithmes inspirés de la littérature mais modifiés pour nos besoins. Le premier algorithme vise à sélectionner dynamiquement l'ordre de modulation le plus bas (par conséquent, la puissance d'émission minimale), tout en essayant de respecter une limite sur la latence maximale d'un paquet. En raison de la nature multi-canaux de WiNoCoD, nous avons redéfini cet algorithme, où il peut être également considéré pour des communications génériques. Par exemple, sur la Fig. 0.24, on voit qu'avec l'algorithme proposé, on peut diminuer la consommation d'énergie moyenne jusqu'à 5 fois, mais cela au détriment de la latence maximale qui connait alors une augmentation de quelques dizaines de symboles. Notre deuxième algorithme définit une limite maximale sur la latence moyenne. Par exemple, dans cet exemple spécifique, la Fig. 0.25 montre que nous pouvons diminuer la consommation moyenne d'énergie jusqu'à 4 fois, en augmentant la latence moyenne de quelques symboles.

Une étude de capacité au sens de la théorie de l'information a aussi été menée.

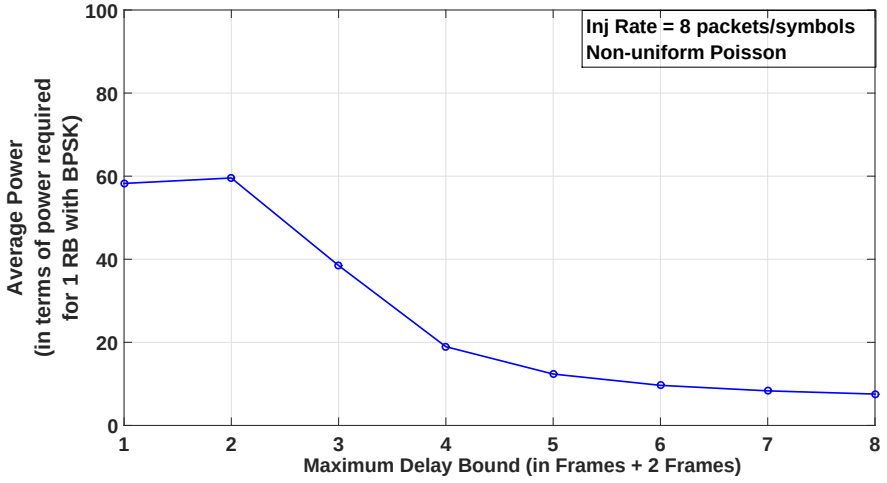


FIGURE 0.24: Le compromis entre latence maximale et puissance moyenne avec l'algorithme de choix dynamique de l'ordre de modulation pour un trafic Poisson non-uniforme.

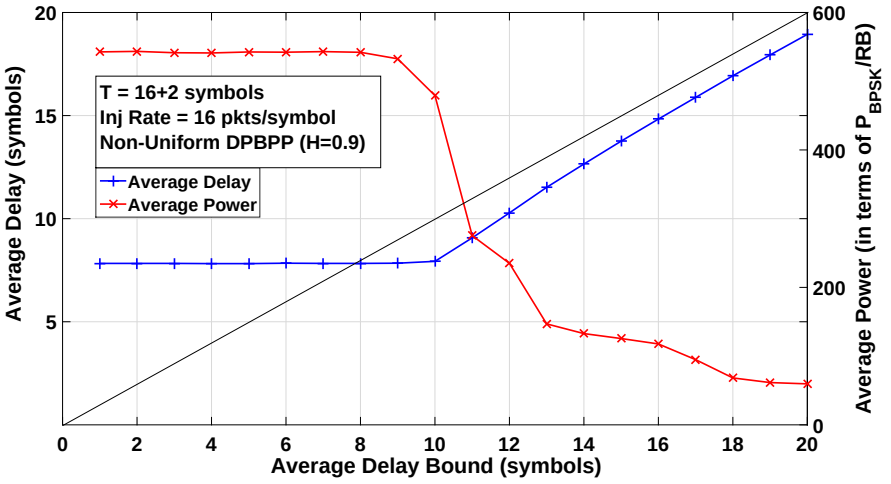


FIGURE 0.25: Le compromis entre latence moyenne et puissance moyenne avec l'algorithme de choix dynamique de l'ordre de modulation pour un trafic DPBPP non-uniforme.

Chapitre 8 : Conclusion

Dans ce travail de thèse qui a été effectué dans le cadre du projet WiNoCoD, nous avons démontré la faisabilité de l'utilisation OFDMA comme mécanisme de modulation sur une interconnexion RF filaire sur-puce pour une architecture massive manycore. Nous avons expliqué et exploité les avantages que peuvent offrir l'OFDM et l'OFDMA, notamment en termes de reconfiguration et de capacité de diffusion. Nous avons développé et analysé les performances de plusieurs mécanismes et algorithmes efficaces d'allocation de la bande passante, en tenant compte des contraintes et des exigences de l'environnement sur puce très spécifique. Nous avons également etudié des algorithmes de sélection de commande de modulation dynamiques qui offrent un compromis optimal entre la latence et la consommation d'énergie. Les contributions de cette thèse sont :

- Une évaluation de la faisabilité d'une interconnexion RF OFDMA sur puce au niveau de la couche réseau.
- Des algorithmes innovants d'allocation de bande pour cette interconnexion RF, adaptées aux exigences tres spécifiques des puces ayant des milliers de coeurs.
- Un nouvel algorithme d'allocation de bande optimisé pour des longueurs bimodales de paquets est proposé pour les réseaux sur puce.
- Deux algorithmes différents de sélection d'ordre de modulation sont proposés pour un compromis entre la latence et la puissance consommée.

- La puissance d'emission minimum sur l'interconnexion RF est evaluée par une approche basée sur la théorie de l'information (non présenté dans ce résumé en français).

Chapter 1

Introduction

Accelerating computation is a continous challenge that has been driving semiconductor technology during the digital era. On the one hand, the previous tendency of shrinking the transistor sizes continuously has been disrupted and faced the wall of thermal issues and lithographic capabilities. On the other hand, the idea of utilizing multiple processing elements (PEs) in parallel on the same chip to execute applications flourished and has proven its viability. Today, it is common to see chips with multiple cores everywhere from our smart phones to superscalar computers for running servers. Actually, we have seen the constant increase of the number of cores from 2 to several tens throughout the last 10 years. And this trend is going to continue even with an accelerated pace. Number of cores on a single chip die is estimated to reach thousands in near future [4]. These systems are referred generally as *Chip Multiprocessors* (CMPs).

Traditional, relatively simple communication mediums for connecting on-chip processing elements were buses or crossbars. However, with increasing number of cores it has become impractical to implant dedicated point-to-point wires and congestion problem has arised with the buses. Researchers had to introduce a new framework known as *Network-on-Chip* (NoC), where communication layer is detached from the data generated by on-chip nodes, and packetized transmission is performed via buffered routers [5].

Even though, the proposed NoC models (mesh network for instance) has provided good performance in terms of latency and bandwidth, they are limited up to several tens of cores. Firstly, multihop routers approach does not scale well as the required number of routers to traverse increases with number of cores, causing unacceptable end-to-end latencies and also congestion in the system. Second, the links between routers are still built by traditional copper wires, where a digital bit is transmitted by Resistive-Capacitive (RC) charging-discharging. This, not only increases transmission latency, but

also causes highly elevated energy consumption, becoming one of the major responsible of NoC based chip power budget [6].

Recently, optical and RF on-chip interconnects have been proposed as interconnects to provide the necessary breakthrough for this bottleneck. These interconnects use electromagnetic waves to transmit signals at near speed-of-light, rather than charging electrically the copper wires [7][3]. Preliminary works demonstrated in the literature promise for remarkable performance gains in terms of latency, bandwidth and power consumption for thousands of cores. However, their implementability is questionable, as for some of the proposed architectures enabling technologies are not mature yet. In addition, both of these optical and RF architectures are plagued by the fact that they need to implant dedicated circuits in nodes' transceivers for each communication channel. One can see that, this scheme is not scalable as core number will continue to increase. Furthermore, due to static nature of these approaches, dynamic allocation of the communication channels to different on-chip nodes, depending on their instantaneous bandwidth demands is not possible. However, the drastically fluctuating traffic intensities of cores is a well understood phenomenon [8]. Hence, these architectures dimension the static bandwidth allocation to on-chip nodes based on its traffic peak, e.g. worst case, which means a significant waste of resources.

In order to alleviate these challenges, WiNoCoD Project (Wired RF Network-on-Chip Reconfigurable-on-Demand) has been initiated by the funding of French National Research Agency (ANR). Project partners are ETIS ENSEA laboratories, UPMC LIP6 laboratories, NXP Semiconductors and IETR/CentraleSupelec-Rennes. This thesis work is contributing to WiNoCoD Project. Project's most distinguished contribution for on-chip community is its OFDMA (Orthogonal Frequency Division Multiple Access) based RF interconnect. Orthogonal Frequency Division Multiplexing (OFDM) has literally revolutionized digital communication in previous decades, becoming the de facto modulation for various wired and wireless standards from DSL to satellite communications, from 4G LTE to optical data interconnects, from WiFi to digital TV broadcast. WiNoCoD is the first attempt to bring OFDM to the on-chip medium, thanks to enabling state-of-the-art technologies provided by NXP Semiconductors. However, the work of this thesis is not restricted to the WiNoCoD project's specific processor architecture and pretends to open a new field for next generation manycore processors in terms of on-chip communications.

Unlike the existing optical or RF on-chip interconnects, OFDM does not rely on prohibitively high amount of analog devices to generate orthogonal channels. Encoding of data on multiple orthogonal frequency channels is a purely digital procedure for OFDM, thus once the required transceiver exists, arbitrating bandwidth among on-chip nodes

based on their changing demands is trivial. In addition, its intrinsic broadcast nature is the key to sustain special on-chip traffic characteristics.

My thesis work in WiNoCoD concerns the allocation of OFDM subcarriers (spectral units-communication channels) to different tilesets based on their instantaneous demands. For this purpose, a basic transceiver infrastructure is also presented alongside these bandwidth allocation algorithms. In addition, the possibility of using intelligent mechanisms to increase data rate with increasing demand concerning energy expenditure is evaluated. This study revisits principles of queing theory, information theory, optimal resource allocation algorithms and stochastic processes.

In the scope of this project several other theses connected with my work have been prepared. One of these theses by Mohamad Hamieh (ETIS) concerns the design of transmission line and access to transmission line from on-chip nodes. Two other thesis of Lounis Zerioul (ETIS) and Frederic Drillet (ETIS) are focused on the feasiblity and modeling of the intended OFDMA analog RF transceiver circuitry. In addition, a thesis is entirely focused on the digital implementation of the RF bandwidth allocation and testing of our multiprocessor by Alexandre Brière (LIP6).

The contributions of this thesis can be listed as :

- A new queuing problem formulation requiring to transform existing queuing strategies concerning very specific requirements and constraints of on-chip environment.
- Several effective distributed and centralized bandwidth allocation algorithms are developed as solutions of the introduced queuing problem.
- An innovative subcarrier arbitration protocol and necessary infrastructure for bimodal packet lengths of on-chip traffic, which requires no extra signaling overhead is introduced. To the best of our knowledge, this is the first kind of attempt in the literature.
- Information theoretical limits and required transmission powers are studied.
- Utilization of higher modulation orders at the extend of higher power consumption is evaluated. Several near-optimal algorithms are proposed inspired from the literature.
- An RF controller structure which is flexible and programmable to support all proposed algorithms (and any other later) is proposed for WiNoCoD's OFDMA based RF interconnect.

- The introduced algorithms are not restricted to the limits of WiNoCoD, but also other possible future OFDMA based RF interconnects for massive multiprocessors or even generalized OFDMA based high speed networks in any kind of environment.

This thesis is organized as follows : In Chapter 2, we introduce the concept of Chip Multiprocessors (CMPs) and Network-on-Chip (NoC). The memory organization and programming principles of multicore systems are explained briefly. Concept of caches and cache coherency is abstracted, which is the major responsible of circulating traffic in on-chip networks. Several proposed protocols for cache coherency for the existing architectures are revisited. Then, NoC paradigm is abstracted and the need for the optical or RF interconnects are stated. Several proposed optical and RF architectures for massive manycore generic CMPs in the literature are examined. Finally in the chapter, the stochastic characteristics of on-chip traffic generated by cache coherency messages stemming from memory hierarchy is evaluated. Certain accurate synthetic traffic models from the literature are presented, emulating the self-similar nature of on-chip messages.

In Chapter 3, we present the details of project WiNoCoD. Our 2048-core generic CMP is introduced along with its hierarchical 3-level interconnects, each with its dedicated communication infrastructure. Following this, distributed hybrid cache coherency protocol designed for scaling 2048 cores is explained in detail, which produces packets that circulate through the interconnection mechanism. Next, basic principles of OFDM and OFDMA are reminded, which is the utilized modulation in our interconnect. Then, physical and electrical properties of our novel wired RF interconnects are examined.

Chapter 4 introduces the problem of bandwidth allocation for an OFDMA system based on instantaneous traffic demands, based on queuing theory principles. Preliminary notions necessary for bandwidth allocation in WiNoCoD architecture are briefly explained. These provide the basics to understand the proposed bandwidth allocation algorithms in next chapter.

In Chapter 5, our proposed bandwidth allocation algorithms are explained both with a centralized or decentralized coordination mechanism. Their performances are evaluated by several realistic and stressing stochastic traffic. The necessary constraints shaping the design of these algorithms are discussed.

Then, Chapter 6 introduces a novel and innovative algorithm called *Payload Channel Algorithm* is introduced which allocates extra bandwidth to incoming cache-line carrying packets without using any signaling overhead. To the best of our knowledge, this algorithm is the first of its kind.

In Chapter 7, option of using higher constellation orders for OFDM is evaluated at the expense of significantly higher power consumption. The trade-off between delays of packets and energy expenditure is discussed by setting a bound on maximum latency and average latency. In addition to this work, we have also studied the required minimum transmission powers for acheiving information theory capacity for different topologies.

Finally, Chapter 8 concludes the thesis work. The gained insights from the information exchange among WiNoCoD project partners are provided. Advantages and disadvantages of the proposed scheme are discussed briefly.

Chapter 2

1000-core Era and On-Chip Challenge

Silicon industry's persistent progress on increasing transistor count by shrinking gate size on a single processor surmounted the digital technology and transfigured our world through the last half of the 20^{th} century. In his seminal paper, Gordon Moore has predicted this phenomenon with an astonishing accuracy back in 1965, that number of transistors would double every 18 months [9], as seen in Fig 2.1. Similarly, clock rates have been multiplied thousands fold since the begining, reaching several gigahertz. However, this incline which has fueled the higher computation powers has been disrupted, as we are reaching towards the physical and thermal limits of nanometer design [10]. In order to utilize silicon resources more efficiently and scale the processing power for the market's current and future demands, the processing design has focused on multicore processors, or with a synonymous term Chip Multiprocessors (CMPs).

2.1 Chip Multiprocessors

CMPs overcome the limitations of uniprocessors by using relatively more simpler cores rather than a prohibitively powerful single core, which also increases the performance by exploiting thread level parallelism (TLP). By the year 2015, most powerful commercially available CMPs include Rapport inc.'s Kilocore processor which is composed of a single powerful core operating in harmony with 1024 8-bit 125 MHz cores [12], Tilera's 72 core 64-bit multiprocessor [13] and PicoChip's 300 cores 16-bit DSP processors [14].

A *core* is a processing unit which performs arithmetic, logical and control operations, that is typically composed of two separate entities : arithmetic and logic unit (ALU)

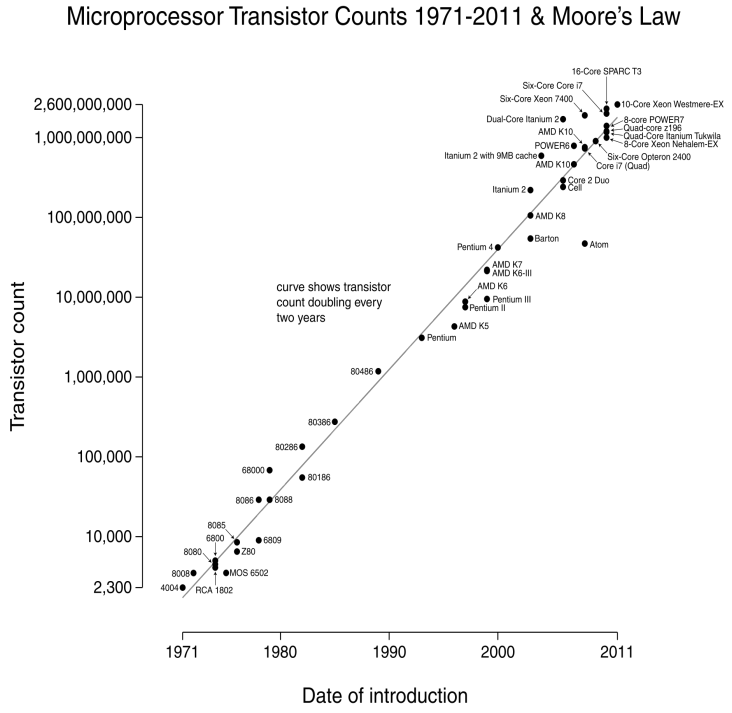


FIGURE 2.1: Number of transistors in a microprocessor over 40 years, validating Moore's prediction. (Image taken from [11].)

and a control unit (CU). A Floating Point Unit (FPU) can be implanted to perform floating point operations.

2.1.1 Caches and Memory

In a multicore processor, each core on the die has a dedicated first level data and instruction *cache*, which is denoted as L1 (Level 1) cache. A cache is a relatively low volume temporary register that stores the copies of main memory, but providing a faster access for the core. Depending on the architecture, there may be higher levels of caches such as L2 and L3 [15]. Based on the well known Von Neumann architecture, the workload of a core (or a processing unit) can be abstracted as the sequences of arithmetical or logical operations made on data copy in a L1 cache read from a memory address according to instructions taken from its instruction cache, and writing the results on an associated memory address. The generic simplified architecture of a 2-core multiprocessor is illustrated in Fig 2.2. In this specific example, each core has a dedicated L1 cache and they share a L2 cache. Monolithic external memory can be regarded as a L3 cache, which is accessed only in case the intended data is not found in L2 cache. However, not all architectures share this hierarchy. For instance, the CMP considered in this thesis work,

has only L1 caches for cores and a physically distributed shared RAM, which can be regarded as a L2 cache.

When a core needs to access a certain data associated with a location tag, it first checks whether it has a copy in its cache. Of course, this query starts from the closest L1 cache, and if there isn't any, the core demands from the higher level caches step by step until reaching the original data in main memory. There are two distinct approaches for CMP's main memory : first is to have a monolithic, single memory which is preferred for low number of cores and second, to distribute the main memory among chip elements which is referred as *Distributed Memory Architecture* [16]. For both of the cases, cores of CMP use a shared address space, hence they can access, read, modify or copy to its cache any part of the main memory. This scheme provides software developer a high level and holistic view of the processor and the memory, so the programming can be performed, as it is being done for a single-core case [17].

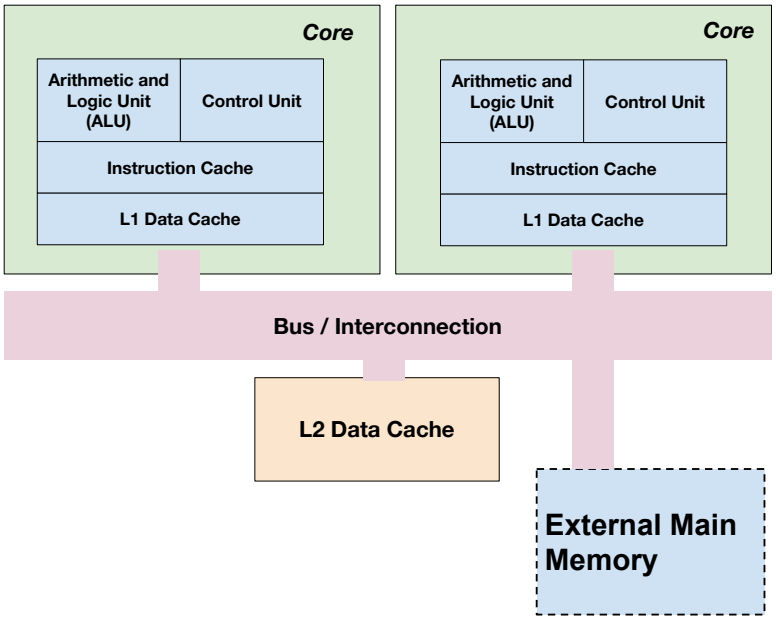


FIGURE 2.2: Simplified illustration of a 2-core CMP with 2 levels of cache and an external memory

2.1.2 Cache Coherency

We have seen that, cores in a multiprocessor can access any location in the shared memory. However, when a core modifies the data in an address location, it may have already copies at caches of other cores. Therefore, when the new data is written, the sharer cores will still have an erroneous copy for that address line. Hence, these cores

should be notified about the content change. This inconsistency problem is known as *Cache Coherency*.

There exists various protocols to resolve cache coherency problem, but most widely known two are *snooping* and *directory based*. In snooping, whenever a core fails to read data in its cache, it broadcasts a read request on the medium connecting all cores, caches and main memory (such as bus, crossbar or any type of network) protocol. As each cache controller in the system listens these broadcasts, they invalidate the copies in their caches with the tag in this request. The second approach is the directory based protocol, where *directories* or memory controllers are responsible for updating and keeping the states and the owners of memory blocks. Any core, who wants to fetch a copy of an address in main memory, must refer to the directory first. The snooping is faster but not scalable for large number of cores, as it requires broadcasting for each invalidation [18]. For any operation done in caches, the atomic unit of memory is a *cache line*. Cache memory is divided into cache lines, where a data is written/read to/from a cache or system memory as a single cache line. The size of a cache line depends on the architecture, but generally it varies between 32 and 256 bytes.

In order to orchestrate the execution of applications and provide cache coherency, the cores and the memory elements have to communicate with each other. Above mentioned cache coherency messages constitute the basis of the on-chip communication and execution of CMPs. For instance, a cache directory may send an invalidation message to an another cache controller, or a core may send a write request for an address space to the directory controller. The length and content of these messages depend on the architecture, used cache coherence protocol etc.

Today, experiencing the exponentially rapid development of semi-conductor technology, with a corollary drawn from Moore's Law, it is expected that the number of cores will double every 18 months, reaching thousands in less than a decade [19]. Thus, this introduces the coherence and communication of 1000-core era CMPs, as the main theater of challenge for designers. Table 2.1 gives 3 examples of massive generic multicore designs with their memory architectures and cache coherency mechanisms compared to WiNoCoD.

2.2 Network-on-Chip

The interconnection of on-chip elements is an essential topic for the processor industry, as it is the backbone for the harmony of parallelism. As it is stated in previous section, the cores and memory elements have to exchange cache coherence messages to operate shared

TABLE 2.1: Processor, memory and cache coherence details of 3 commercially available or proposed CMP architectures compared to WiNoCoD.

	Tile GX-72 [13]	**Corona [20]**	**ATAC [3]**	**WiNoCoD**
Commercially Available	Yes.	No.	No.	No.
Core Count	72	256	1024	2048
Core Type	64-bit RISC @1 GHz	RISC	32-bit RISC @1 GHz	RISC
Caches and Memories	32 KByte Instruction and Data Cache per core 256 KByte private L2 Cache per core L2 caches have extra lines for shared 18 Mbyte total L3 cache 1 TByte external 4 DDR3 RAM Blocks	32 KByte Instruction and Data Cache per core 256 Kbye L2 cache shared by each 4 processors (a tile) External off-chip DRAM block	32 KByte Instruction and Data Cache per core 32 Kbye L2 cache shared by each 4 processors (a tile) External off-chip DRAM block	32 KByte Instruction and Data Cache per core 1 TByte shared and distributed RAM to each 4 cores (a tile)
Cache Coherence	Distributed Directory Based Coherence	Distributed Directory Based Coherence	Distributed Directory Based Coherence	Distributed Directory Based Coherence

memory mechanism consistently, which provides a simple, transcendental programming interface for the software layer.

2.2.1 From buses to NoC

First multicore designs were including just a few number of cores, where all of them share a simple *bus*, basically a bundle of copper wires. This design also supports the aforementioned shared memory principle with snooping cache coherence due to broadcasting. However, with the increasing number of cores, memory elements and I/O units, the bus mechanism has faced certain electronic and bandwidth limitations.

As buses get longer to reach all the elements, the RC latency also increases with wire lengths. With so many elements connected to a bus, the capacitive load has increased which is an extra cause of delay. In these conventional deep sub-micron architectures, the digital bits are transmitted between nodes through copper wires by increasing or decreasing voltage (voltage level determines a 0 or 1). However, the latency of this transition is determined primarily by resistance, capacitence and length of the wire (even inductance to some extend) [21]. Because, in order to increase the DC voltage at the other end of the wire, first the capacitance of the wire shall be charged which is determined by the RC time constant (e.g. multiplication of resistance and capacitance). It can be approximated, transmission delay increases with the square of the wire length [22].

In addition, as number of units accessing to the bus increases, bandwidth per unit decreases, resulting in congestion [23]. As a solution, *crossbars* are proposed to connect 4 to 8 core processors. A crossbar is a simple switch, where multiple nodes are connected in a matrix manner, thus enabling concurrent communications, originating from different nodes, in contrast with the bus. Using tri-state buffer crossbar switches, multiple inputs can transmit information to different output nodes. This scheme increases the bandwidth compared to bus, however generally for more than 8 cores, it also fails to be sufficient [24]. Fig. 2.3 depicts the on-chip nodes connected by a simple bus and a crossbar. The chip architectures with more than 6-8 cores mandates the requirement for much more efficient interconnects than crossbars in terms of bandwidth, latency and power. Even, the design of on-chip interconnects have become the most challenging research interest for silicon industry recently, due to the fact that they are now the main bottleneck of the architectures with their delay, area and power expenditure. This phenomenon is known as the *computation to communication paradigm shift*.

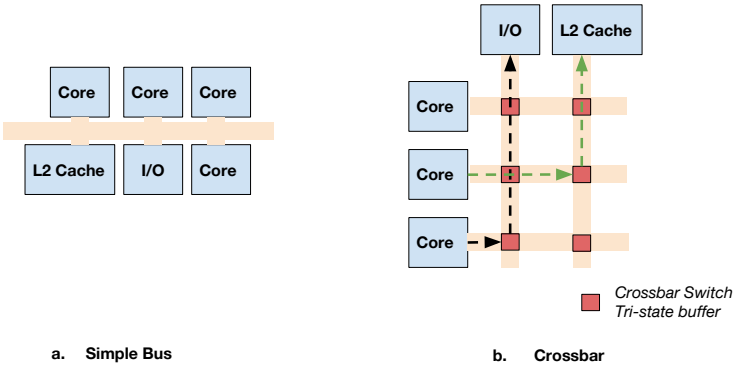


FIGURE 2.3: On-chip elements connected via a simple bus (a) and a crossbar (b). Bus enables only a single transmission at a time, however with a crossbar multiple transmissions between 2 nodes can be done, simultaneously (only to a single output at a time)

In order to scale CMPs with more and more cores, a layered, packetized, router based communication framework in a similar manner with the traditional data networks such as internet or computer networks has been introduced, known as *network-on-chip (NoC)*. Another formal definition in literature exists : *"An on-chip interconnect with decoupled transaction layer, transport layer and physical layer"* [24]. In a NoC, the higher layer packets (messages from cores, memory units etc.) are packetized with overhead, which is called a *flow-unit (flit)* in NoC terminology, and transmitted to destination node via routers, hop-by-hop. This modular approach does not only increase bandwidth, but also enables a wider spectrum of choice for designers, as various topologies, routing and arbitration algorithms can be applied. Today, NoC is the de facto accepted approach to design manycore systems. It has changed the approach of on-chip community to the

problem, where brand new research interests have emerged such as optimization and dimensioning these router based architectures.

2.2.2 NoC topologies

Since the initial research on NoC, various topologies have been proposed such as a 2D mesh, 2D-torus, octagon and many more [5]. Fig 2.4 illustrates the different NoC topologies. It has already became the de facto communication fabric for many core designs. Despite its tremendous advantages, as we are approaching to an era with hundreds, even thousands of cores, these conventional NoCs are also plagued with the scalability problem. For instance, one can see that the number of routers to traverse between two most distant cores in a 2D mesh network is $2\sqrt{N}$, where N is number of cores. Taking into account the several cycles of delays at each router due to packet processing, this implies the importance of decreasing the hop count. To further extend these types of architectures, even 3D structures are provisioned, where a new spatial dimension for routing is added [25]. However, one can see that this transitionary solution scales the number of nodes which can be interconnected effeciently only a few more times. To sustain the quadruple increase of cores, a breakthrough is needed. Not only the long distance, but other problems such as *deadlock routing* (a case where flits are switched between same routers in a vicious cycle due to routing and arbitration policy), not being able to support broadcast and multicast etc. force researchers to find innovative solutions.

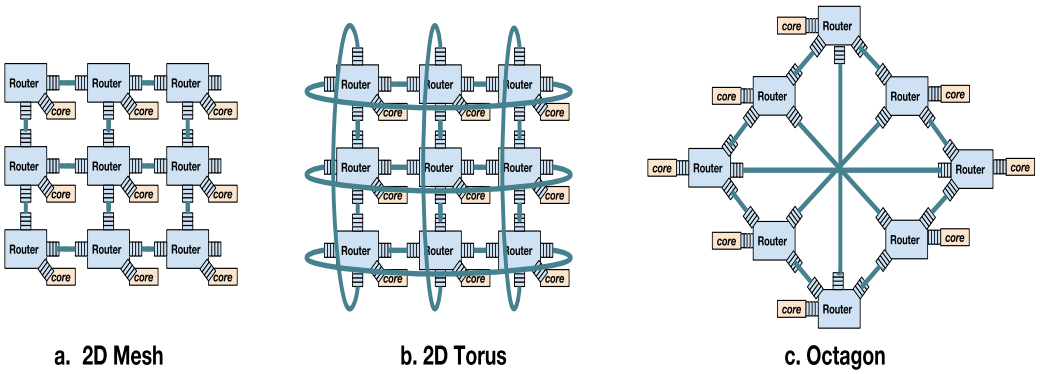


FIGURE 2.4: 3 of most preferred 2D NoC topologies, where processing elements are interconnected via buffered routers.

2.2.3 Tera-Scale Multi-core Processor Architecture (TSAR)

In this part of the manuscript, it is important to mention about the *Tera-scale Multi-core Processor Architecture (TSAR)* project [1]. Among its project partners, there are

UPMC-LIP6 Laboratories and NXP Semiconductors, which currently contribute the WiNoCoD Project. In a sense, this project can be seen as a base of WiNoCoD, as its core and memory architectures share many common features.

Like in WiNoCoD, TSAR also implements a distributed (physically separated but logically shared) memory architecture, that 1 TByte total RAM is distributed uniformly to 4-core tiles. There is no private L2 caches in TSAR and this distributed memory can be seen as a L2 cache. Distributed Hybrid Directory Based Cache Coherence Protocol (DHCCP) is utilized just as in WiNoCoD. There is no exclusive ownership for address lines and a simple, yet scalable write-through policy is adopted, which is also the same for WiNoCoD. Eventhough, the first prototype has been modeled for 128-cores, project has opted for up to 4096 cores.

TSAR has been designed with a conventional 2D mesh network for communication between tiles, whose scalability is limited. Fig. 2.5 illustrates the tiled architecture of TSAR, where each tile contains 4 cores (actually depends on the choice), a local RAM slice of the main memory, and a memory directory/controller. They are interconnected by a local interconnect (a crossbar). A similar approach is adopted in WiNoCoD due to its scalability, which we will present in next chapters in detail. These tiles in TSAR are interconnected by a specialized electrical network called *DSPIN*, which is the fundamental difference from the WiNoCoD project. WiNoCoD adds a higher level of interconnection based on wired RF transmission.

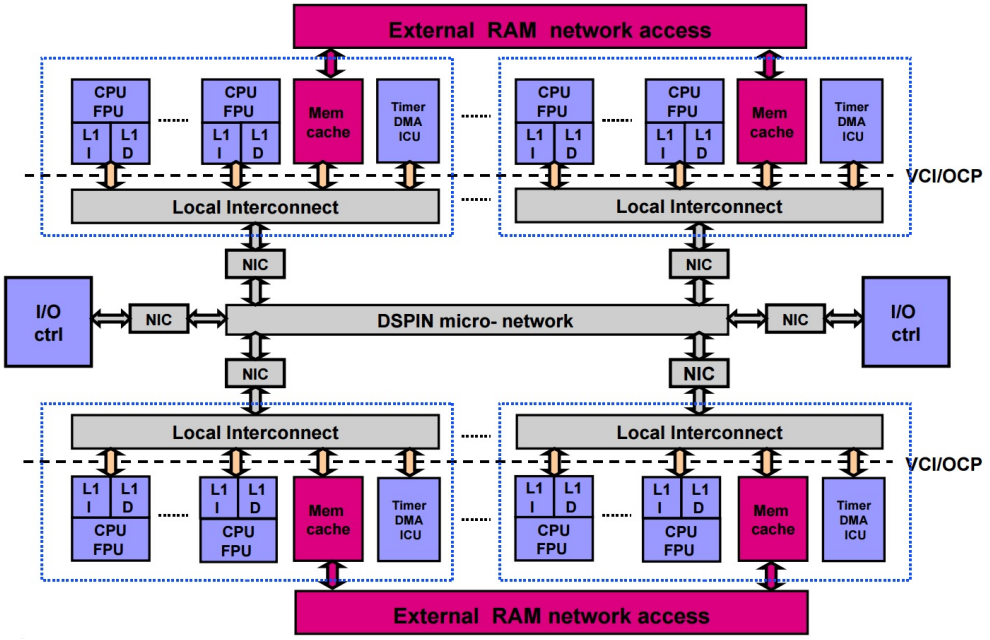


FIGURE 2.5: TSAR's tiled architecture and distributed memory

2.3 RF and Optical Interconnects

With thousands of cores on the horizon, silicon industry has figured out that the conventional wired NoCs are far away to furnish expected requirements in terms of latency, bandwidth or power. This provokes designers to explore uncustomary interconnect designs to meet the demands. First of all, as bit transmission through copper wires is performed via RC charging-discharging, with increased lengths this type of signaling caused unbearable latencies. In addition, these copper wires starts to constitute up to 70% of total system capacitance, being responsible a significant part of power dissipation. So, idea of communicating via electromagnetic waves at near speed of light sparked [26]. This innovative idea has paved a new avenue of research on interconnects, enabling numerous design options for 1000-core architectures. International Technology Roadmap for Semiconductors (ITRS) has declared, optical and RF (Radio Frequency) interconnects are the only viable approach to satisfy long term requirements of bandwidth, latency, power and reliance [27]. In addition, these types of interconnects may support broadcasting (we have emphasize the importance of broadcasting for scalable cache coherency in Section 2.1.2) due to their signaling mechanism and allow concurrent transmission from multiple nodes by creating orthogonal channels. Four of the proposed new approaches are illustrated in Fig. 2.6. In fact, they can be seen as a come back from mesh to bus, adding parallelism in frequency on the bus, then enabling concurrent communications.

2.3.1 Optical Interconnects

Recent development in nanophotonics made implantation of optical elements such as dense waveguides, filters, modulators etc. on a single silicon die available. With these enabling technologies, on-chip optical interconnects have attracted a lot of attention from both academia and industry as a potential to scale 1000-core CMPs, due to their capability to provide up to Terabits per second bandwidth by using simple ON/OFF modulators, minimizing cross-talk effect, and their relatively simple optical routing/signaling [28]. Besides, their energy consumption is affordable. Even though, free space intra-chip optical interconnects are investigated by various researchers [29],[30] the general direction in designs is to connect cores (or group of multiple cores and memory elements which is called as a *tile*) with an optical waveguide, serving as a fast information highway. Nodes transmit their messages on the waveguide by converting electrical signals to optical ones. At the reception these signals are reconverted.

Even though optical on-chip interconnects require some additional time to be practically mature enough, recently [2] has demonstrated a single chip using optical communication.

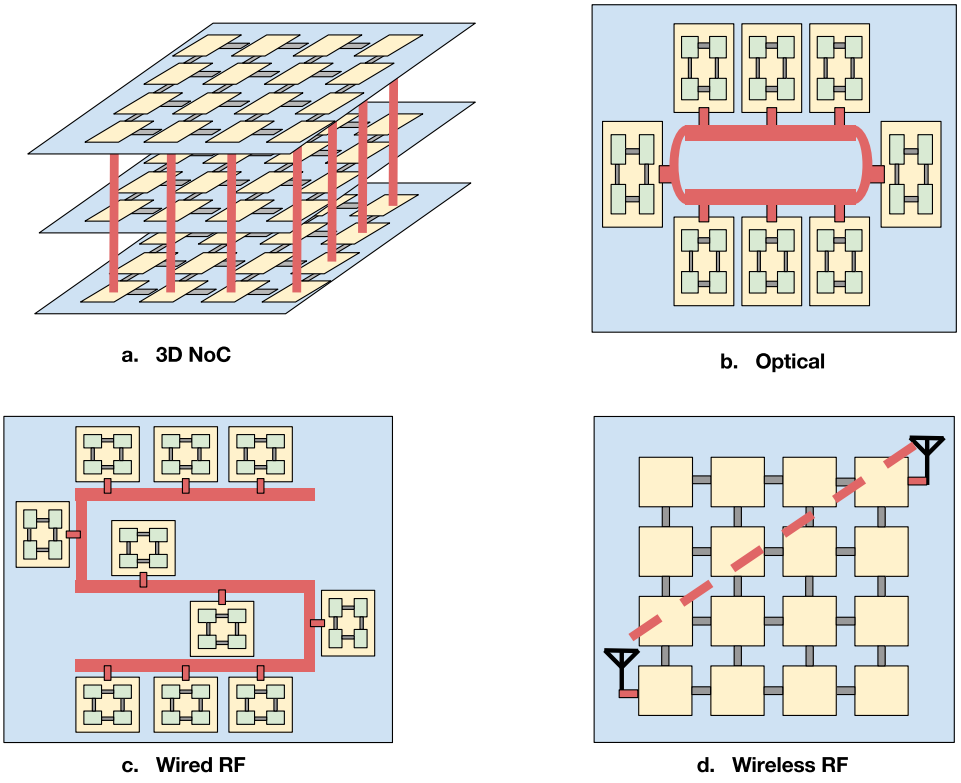


FIGURE 2.6: 4 of the recently proposed innovative on-chip interconnect options

Despite being a small scale architecture composed of 2 cores, this can be evaluated as a significant achievement for paving the road for future optical chips.

To enable simultaneous transfer of multiple signals on the same waveguide, orthogonal channels i.e. *wavelengths* are generated [31]. We speak about Wavelength Division Multiplexing (WDM) in this specific case. An on-chip or off-chip laser source generates and circulates photonic energy on a dedicated waveguide, where it contains all the wavelengths utilized in the system. Microring Resonators (MRRs) are circular nanophotonic devices which are able to refract the light with the desired wavelength. Using this property, they can be utilized as modulators for on-chip, where based on the incoming electrical signal, nodes can refract the light with desired wavelength from the optical source waveguide and drive it to the data waveguide as 1 or 0. Similarly, for reception, they can be used as filters to detect data transmission on desired wavelength [6]. The operation of MRRs as modulators at transmission and filters at reception for on-chip interconnects is depicted in Fig. 2.7. Each of the MRRs is manufactured for a specific single wavelength statically, where it is determined based on the thermal procedure, injection of different amount of electrical charge, or varying the radius during the production [32]. One can see that for transmitting each single bit orthogonally, a different MRR on a seperate data waveguide is needed. And similarly for the reception, detecting

each bit on a different wavelength requires another MRR as a filter. As number of on-chip nodes increase, the required bandwidth increases as well, meaning for each bit in reception and transmission, a new MRR is needed. As in [3] or [20], tens of waveguides are bundled to implement required optical datawidth, and hundreds of MRRs are implanted to generate orthogonal channels. In addition, they are static, thus dynamically allocating bandwidth to different nodes, according to changing demands is not feasible. Also, the requirement for the constant laser source is another drawback.

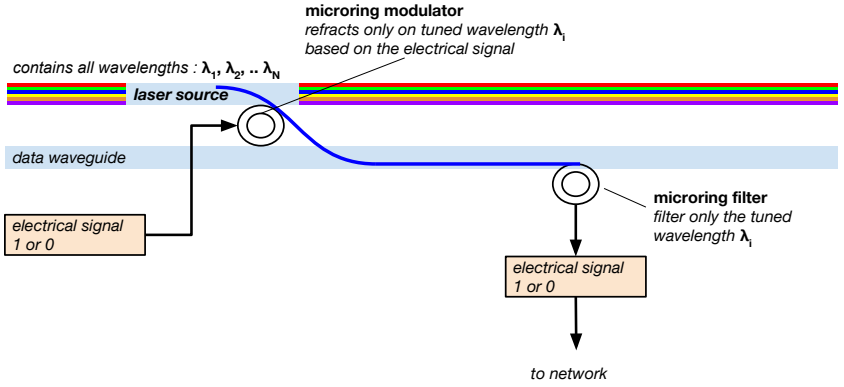


FIGURE 2.7: Microring Resonators (MRRs) which are able to refract light on the pre-tuned wavelength are utilized both for transmission and reception. [32]

At this point, we wish to highlight an important concept in orthogonal channel implementation. In literature there are two complementary approaches, which are called Single Write-Multiple Read (SWMR) and Multiple Read-Single Write (MWSR). As illustrated in Fig. 2.8, in SWMR, each node transmits their data on specific wavelength, whereas at reception each node are tuned in to each channel. Thus, they can receive others' transmission. In MWSR, each node is allocated a specific channel, called home channel, to receive information. At transmission side, each can modulate on every channel. Both schemes have specific advantages and drawbacks. Note that, SWMR supports broadcast naturally, which is desirable for on-chip traffic. On the other hand MWSR does not support broadcast, besides it has another disadvantage, that in order to avoid a collision caused by concurrent transmissions of multiple sources to the same destination node, an arbitration mechanism is needed. Despite these drawbacks, usually a MWSR mechanism is preferred to SWMR, due to fact that its power consumption is less by only activating a filter for a single channel at RX [33].

For this purpose, different paths can be followed such as using tunable ring resonators or laser modulators at different frequencies. However, implanting these optical devices on each node is required. Considering the fact that number of these devices shall increase exponentially with increasing number of cores and channels, the scalability question

arises with today's technology. Besides its CMOS incompatibility raises concern on feasibility [28].

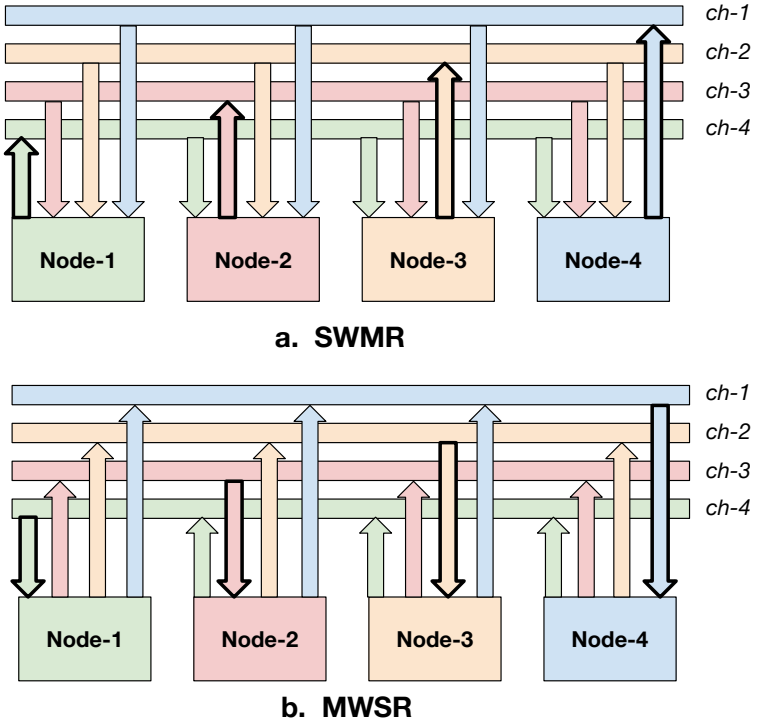


FIGURE 2.8: Two different approaches of managing orthogonal channels in an on-chip interconnect (a) Single Write Multi Read (SWMR) : each node transmits on a dedicated channel, where each other node can receive. (b) Multi Write Single Read : each node is allocated a specific channel called *home channel*, but they can transmit on any channel. Those who want to transmit data to a node, transmits on its home channel.

There exists a vast literature on optical on-chip interconnects, however we present only the two most relevant architectures, one representing a SWMR example and other a MWSR one. In the following sections of this thesis, we will revisit these two architectures to draw analogies with our system in order to explain it better.

2.3.1.1 ATAC

ATAC is a 1024-core CMP, with an SWMR optical interconnect, developed by MIT [3]. It has a tiled structure, that 16 tiles (a core, a directory-memory controller and a local cache) form a cluster, which are interconnected via an optical bus. They have an innovative but costly design for intra-cluster communication : for outgoing messages a regular 2D electrical mesh network is used, however for the incoming messages from optical waveguide, a special electrical, tree-like *broadcast network* is built, which reaches each tile directly. The monolithic memory is also connected to the waveguide.

Researchers use a *hybrid directory based cache coherency* protocol, where if the number of cores sharing an address block exceeds a certain threshold, directory only keeps the number of sharers rather than keeping their IDs exclusively. Directory broadcasts any cache coherency message regarding this address block to all 1024 cores in the system. For responses coming from sharing cores following the previously transmitted cache coherency message, the initiator directory just counts the number of ACK messages and compare it to number of sharers for the related memory block. That's why authors refer this scheme as *ACKwise cache coherence protocol*. The strong broadcast infrastructure of ATAC enables this protocol's implementability. Authors mark that this type of cache coherence protocol provides an efficient ease of programmability for 1000-core CMPs.

Even though, ATAC provides a terabit bandwidth with strong support for broadcast, it has certain drawbacks. First of all, this SWMR bus requires significant optical circuitry in each 64 clusters. In the optical receiver of clusters, an optical ring resonator filter for each wavelength is needed. This means 64 ring resonators per cluster. In addition, each cluster is dedicated one wavelength (64 bit channel) for transmission, statically. In case of bursty traffic, these channels cannot be redistributed among clusters, thus the system has to be dimensioned for the worst case. This significantly lowers this architecture's scalability, especially if the traffic load is not homogeneous. Indeed, there could be some channels overloaded while others are unused. Then the global throughput can be significantly lower than the theoretical total capacity of the optical NoC.

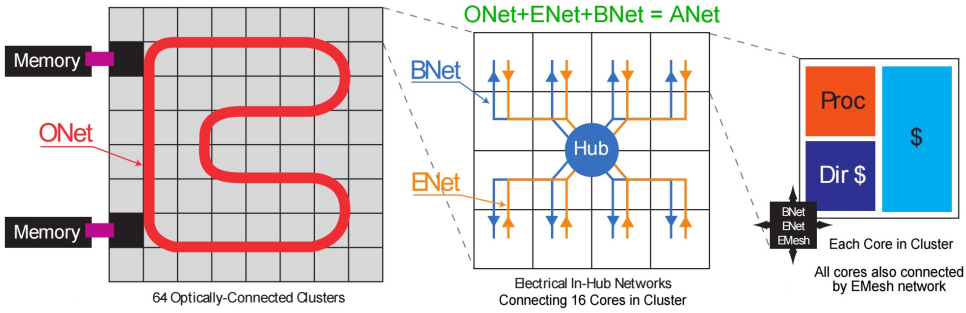


FIGURE 2.9: ATAC's 2 level clustered architecture, where clusters are connected by bundle of optical waveguides using WDMA. For reception, each cluster incorporates a special electrical broadcast network. (Image taken from [3])

Fig. 2.9 shows the 2 level clustered architecture of ATAC. 16 generic cores are interconnected by two separate conventional electrical networks for transmission and reception.

2.3.1.2 Corona

Corona is a CMP with 256-cores using a MWSR optical interconnect, developed at University of Wisconsin-Madison [20]. As previously mentioned in this section, MWSR mechanisms require an arbitrator to resolve a potential conflict, caused by multiple sources node trying to send messages to the same destination node at the same time, by encoding data on its home channel. Corona, uses optical tokens to provide arbitration, that is based on the optical diversion technology.

The system prefers a *non-uniform memory architecture (NUMA)*, so that the memory is distributed physically but logically shared as in ATAC. In other words, it is a distributed memory architecture. In Corona, tiles are composed of four cores and a common L2 cache, a directory and a slice of main RAM. Each of 64 tiles are directly connected to a serpentine, ring optical waveguide. The light travels unidirectional through the waveguide, that emission sources from one end of the waveguide and sinks at the other end. Each optical channel is composed of bundles of independent,thus orthogonal 64 waveguides using the same wavelength, at the end gives a 64-bits wide physical channel. Each tile has the necessary optoelectronic converters and optical circuitry : ring resonator detectors (receiver) tuned to home channel, modulators (transmitter) tuned to every channel (expect its home channel) and optical token diverters.

As Corona is an MWSR system, for multicasting and broadcasting,-*which are required for memory block invalidation messages intending a large pool of sharers*, redundant copies of the message has to be encoded on every receiver's home channel. To avoid congestion, the researchers employ an additional separate waveguide called broadcast bus. It is a single end 2 times folded spiral like optical broadcast bus, which passes every cluster 2 times. At first cycle on the broadcast bus, nodes modulates for invalidation messages. On the second cycle invalidate messages becomes active and thanks to their optical detectors, only concerned nodes snoop their caches.

To resolve the contention of MWSR optical bus, a single additional waveguide *optical token* arbitration channel is augmented to the system. Optical token is simply an ON/OFF optical signal (or 1-bit) transmitted on all 64 wavelengths, representing the *right to transmission* on the 64-bit channel with the associated wavelength. It can be seen as an optical flag. When a node is not receiving any signal on its home channel, it constantly modulates an optical signal on its token channel. If another node would like to send a packet to this node, it has to acquire this token. To do so, with its optical diverter circuit, it diverts and sinks the optical signal. Thus, any further node in the system, may not send on this channel, as its token is already acquired by this node, which resolves the contention problem.

Authors claim their architecture is scalable for massive multicore systems and provide a significantly low power dissipation thanks to its nanophotonics infrastructure. However, the system requires prohibitive amount of optical circuitry and MWSR bus still poses important drawbacks.

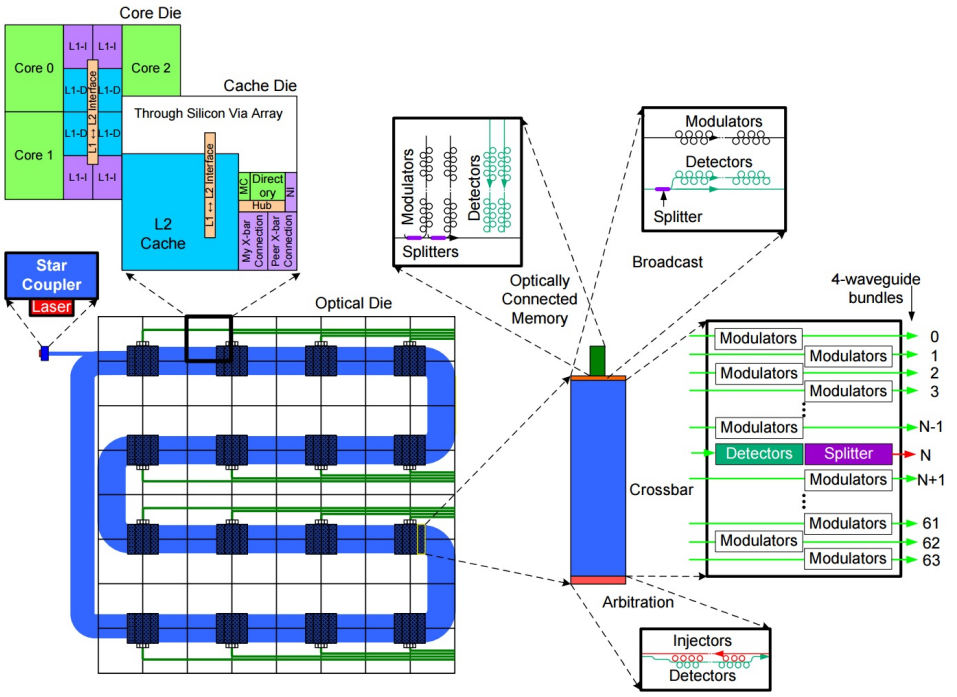


FIGURE 2.10: Corona has a similar serpentine optical interconnect similar to ATAC, except the choice of MWSR rather than SWMR. Tiles, each are composed of 4 cores are also illustrated (Image taken from [20]).

Fig. 2.10 illustrates Corona's bundle of optical waveguides in a serpentine form similar to ATAC and the tiled core architecture.

2.3.2 RF Interconnects

Photonic interconnects are considered as an effective technology to reduce the latency, decreasing hop count between cores compared to traditional planar metallic NoCs, with a significantly high bandwidth, while providing low power consumption. However, their practicability is doubted, at least for near future, due to noise from waveguide coupling and relative sizes of optical components. The fabrication of on-chip photonic components faces challenges such as low quantum efficiency [34]. In addition, there exists no optical storage elements, thus system depends on an electrical infrastructure [35]. As mentioned previously they require constant on-chip or off-chip laser sources with a dedicated separate waveguide. There is also a fundamental incompatibility with CMOS

technology which is the most common and cheapest chip technology. Integrating optics in silicon chips is still cumbersome.

Concerning drawbacks of optical interconnects, academy has oriented towards Radio Frequency (RF) interconnects, which still use Electromagnetic (EM) waves at near speed-of-light. Present on-chip high frequency silicon technology permits implementation of cut-off frequencies up to hundreds of GHz, thus a sufficient data bandwidth for CMPs. The evident advantage of this approach is its CMOS compatibility, which is a much more mature and viable technology compared to other alternatives [36].

[37] states that the switching frequency of CMOS transistors is still in the trend of increase exponentially with each generation, therefore enabling maximum frequencies up to 1 THz. This presents the RF CMOS devices as scalable candidates for high rate on-chip data transmitters, even in the near future considering the increasing bandwidth demand. In addition, it is demonstrated in this article that energy consumption of RF interconnects is significantly lower than optical communications up to distances of 30 cm, which falls to the interest of on-chip network [37].

There are two distinct proposals for RF interconnects : free space (wireless) communication or waveguided communication (wired RF).

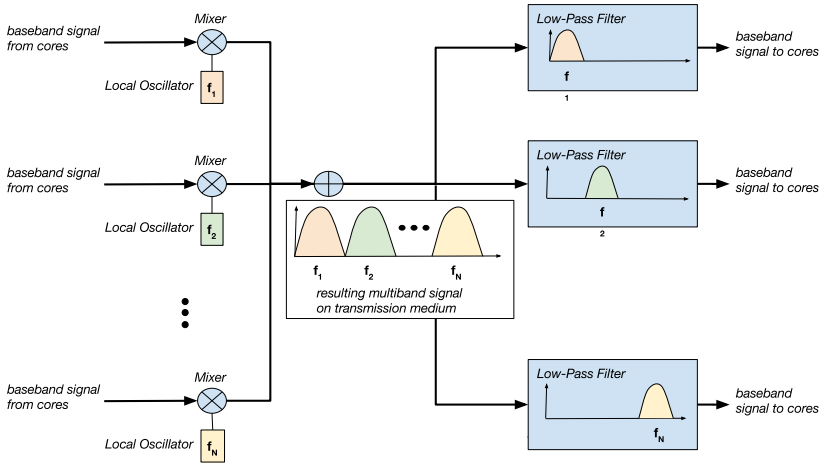


FIGURE 2.11: RF interconnects require a mixer circuit driven by a local oscillator to modulate baseband signal to a specific frequency. Similarly at reception, for acquiring data on a specific frequency, a Low Pass Filter which is centered to the desired frequency by a local oscillator is needed.

2.3.2.1 Wireless RF Interconnects

Implanting miniature antennas on-chip and providing free space propagating communication links is referred as *on-chip Wireless RF interconnects*. The idea is to place

high throughput wireless links between distant cores, thus reducing latency and congestion, without the need of an extra propagation medium such as a waveguide. The major design challenge in this paradigm is manufacturing such small antennas with desired electromagnetic propagation characteristics [7]. Different central frequencies and bandwidths are proposed in the literature, using completely divergent transmission and reception antenna designs. [38] proposes Gaussian Monocycle short impulses based Ultra Wideband (UWB) communications provided by Meander type dipole antennas. Due to frequency and antenna characteristics this scheme can allow a transmission range of 1 mm, thus a multihop *wireless mesh* transmission infrastructure is proposed.

Another suggestion is to utilize zigzag shape miniature metal antennas to propagate milimeter length waves [39]. A transceiver design of 16 Gbps using milimeter waves is shown in [40]. Besides these, an additional proposal is to employ *Carbon Nanotubes* (CNTs) as on-chip antennas [41]. Their special electrical and propagation characteristics make them suitable for Terahertz (THz) wide on-chip communication. One advantage of these wireless on-chip RF interconnections, is to exploit *small world phenomenon*, decreasing congestion and latency significantly, by few additional links between distant cores.

Even though wireless RF *intra-chip* interconnects need fundamental breakthroughs such as CNT antennas etc., less complicated wireless *inter-chip* interconnects are already being developed. For instance, [42] demonstrates a 3D stacked system-in-package (SiP) interconnected by wireless signals.

2.3.2.2 Wired RF Interconnects

Viability of the aforementioned wireless on-chip antennas have not been demonstrated yet and innovative proposals such as Carbon Nanotube antennas are still not mature technologies. Thus, RF propagation via guided transmission line (wired RF) has received more attention throughout research community compared to its wireless counterpart. As communication distance is low, effective capacitive coupling method can be used to realize transmission [43]. In addition, the waveguide allows for a recoverable attenuation. Chang et al. have implemented a multiband wired RF CMP architecture, where small distance communications is performed by conventional electrical routers and for long distance destinations high speed RF line is used via local RF routers (switches) [44]. This scheme uses Frequency Division Multiple Access (FDMA), that 6 orthogonal 10 GHz bands are utilized to encode 6 different signals concurrently on the transmission line. To do so, each 6 RF router uses mixers and local oscillators in transmission, and filters at reception. Note that, this scheme can be classified as SWMR, and allows broadcasting

up to a degree. The authors also mention about static or dynamic re-allocation of bands for changing traffic demands. However, as RF routers has to include all local oscillators, mixers and filters for each band, the reconfigurability and bandwidth division granularity is limited. Wired RF transmission lines are considered as a suitable candidate for high speed EM propagation based on-chip interconnects with current CMOS technology [45].

2.4 Characteristics of On-Chip Traffic

Network on Chip (NoC) is a relatively new research area. As a first step, researchers relied on the primitive synthetic traffic patterns to evaluate their designs [46]. These models are too naive to ensure validity of the simulations. In addition, certain benchmarks including limited number of multicore applications such as PARSEC [47] or Splash-2 [48] exist. However considering the fact that future CMPs will run applications from a much more vast spectrum, a universal, realistic statistical model for imitating NoC traffic is essential. The need for the synthetic models for testing future CMPs, rather than relying on few limited benchmarks is emphasized also by [49].

On-chip traffic has similar properties with previous shared memory and inter-chip systems [50]. However, for single-chip multicore systems, with decreasing distance between cores and smaller caches, traffic is much more frequent and shows different features. In 2002, Varatkar and Marculescu were the pioneers to identify self-similarity of intra-chip traffic on a multicore system with NoC which is running an MPEG-2 application [51].

Following, Soteriou et al. have presented an empirically derived statistical model for any kind and any scale of CMP NoCs, that captures the spatio-temporal characteristics of on-chip traffic with only 5% of average deviation from real benchmarks [8]. It uses 3-tuples (H, σ, p) representing temporal burstiness, spatial load distribution and hop-count to destination, in order.

The self-similarity of on-chip traffic (in other words scale invariant burstiness [52]) has been observed by academia extensively [53][54]. This roots from the cache hierarchy in shared memory systems. In contrast with the Poisson model approach, self-similar traffic shows temporal long range dependent features, and its characteristics are scale invariant as in Fig 2.12. In different time intervals total number of packets injected by a node in CMP shows statistical similarity.

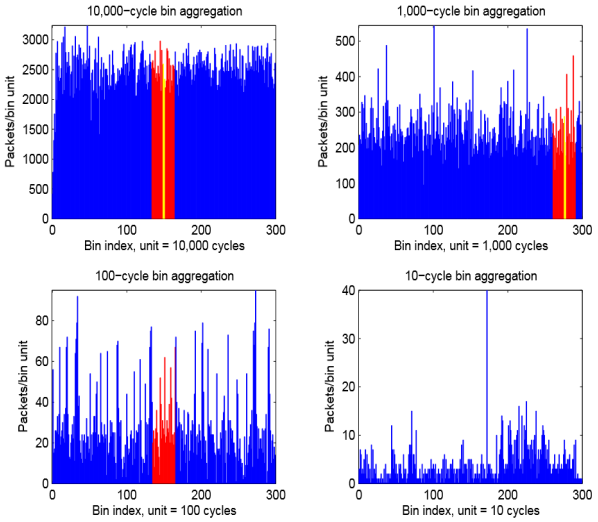


FIGURE 2.12: Self-similar traffic is scale invariant as seen for this example for the number of packets generated by a node in different sizes of time bins. (Image taken from [8])

2.4.1 Temporal Burstiness : Hurst Parameter

Self-similar or long-range dependent (LRD) traffic can be formalized mathematically, as a stoachastic process with an auto-correlation function that is decaying hyperbolically slower than an exponentical function (representing *memorylessness* of Markovian processes) : $r(k) \sim a_0 k^{-\beta}$, where a_0 is a finite scalar and $0 < \beta < 1$. In literature $H = 1 - \frac{\beta}{2}$ is known as *Hurst parameter* which reflects higher burtiness as it approaches to 1. Among 30 different applications, authors found that the number of packets generated by a core throughout whole execution time is well fitted to Hurst parameter based representation [8]. There are certain methods to generate Hurst parametric self-similar traffic, though we explain two of them here.

The method explained in [55], aggregates multiple independent Pareto distributed ON-OFF processes to produce self-similar traffic. Let us say that number of aggregated independent processes are K, and the desired injection rate (average number of packets generated per time unit) is λ. Average injection rate is the ratio of ON process's mean duration to the sum of ON and OFF processes' mean duration, which are Pareto distributed : $\lambda = \frac{T_{ON}}{T_{ON}+T_{OFF}}$. So the location parameter for Pareto distribution of the ON duration, b_{ON} is determined at first by the user, and the b_{OFF} is set by the desired injection rate as : $b_{OFF} = b_{ON}(1 - \frac{\lambda}{K})^{-1}$, where K is a scaling parameter.

Following this both of ON and OFF duration's Pareto distribution shape parameter β is determined according to desired Hurst parameter as : $3 - 2H$.

The second method is called Discrete Pareto Burst Poisson Process (DPBPP) [56] [57]. In this approach, each node in the system at every instance, generates flows whose number is determined by a Poisson distribution. Each flow generates packets or bits at a constant rate when active. And the duration of each flow is Pareto distributed with the desired Hurst parameters, as in the previous method. At each instance, the aggregated throughput of the currently active flows gives the desired self-similar traffic generated by a node.

2.4.2 Spatial Burstiness : Standard Deviation

Next tuple is the spatial injection rate distribution indicator, σ. In [8], authors have discovered a remarkable phenomenon, that the total number of packets generated by a certain node in the system can be well modeled by a Gaussian distribution. Thus, they use standard deviation of normal distributions to represent an application's spatial traffic characteristics. For instance, Fig 2.13 shows the well fitting Gaussian distribution to model spatial burstiness, for 3 different CMP architectures with different memory hierarchy and cache coherency protocols running 3 different applications. One of the CMPs have 25 cores and the rest have 16 cores.

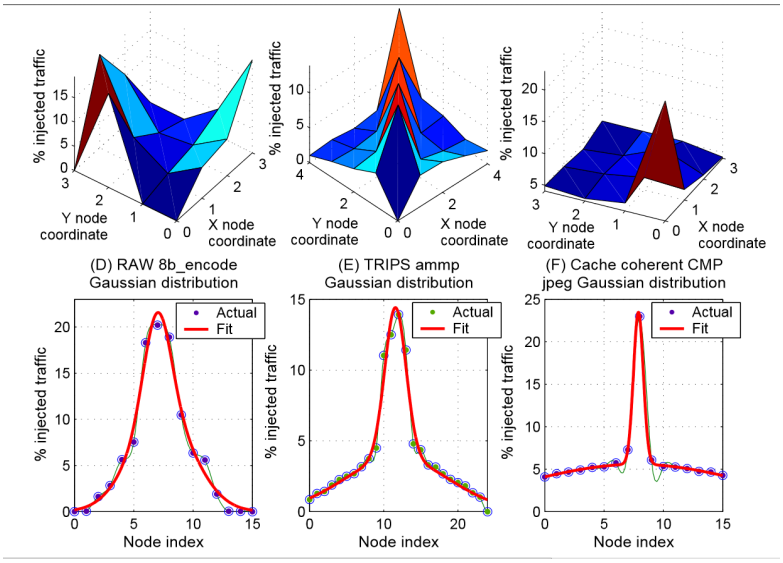


FIGURE 2.13: Gaussian normal distribution fits well for 3 different applications under different cache coherency protocols and CMP architectures. (Image taken from [8])

2.4.3 Statistical Distance Property of on-chip Transactions

Last tuple, p represents the source-destination distance (number of hops-*routers* to traverse) of any generated packet. It was well observed previously that in CMPs and shared

memory systems, the destination of packets tends to show a certain locality pattern, referred as *Rentian locality* [58]. This phenomenon roots from the memory hierarchy of CMPs. As cores store frequently accessed data in nearer caches, they tend to reach exponentially less to far memory locations [59]. In parallel with this observance, authors derived the probability of a generated packet to have a probability P, of having a destination equal or smaller than d hops : $P = (1-p)^{s(d)}$, where $s(d) = \sum_{i=1}^{n} n_i$ and n_i is the total number of nodes that has i distance from the source.

Even though, workload benchmarks such as PARSEC [47] and SPLASH [48] is still widely used by on-chip research community, their coverage for the traffic characteristics of vast spectrum of possible applications, as mentioned previously. Generally, the considered applications in these benchmarks are highly specific to a limited group of interest, such as scientific or industrial applications. Hence, their performance for evaluating interconnects is questionable for generic purpose massive CMPs. One can see that, an accurate abstracted (decoupling traffic from application layer) stochastic model which is easily parametraziable would be essential. Furthermore, they require exceedingly prohibitive computation power and time to test them. Hence, researchers have started to seek new, realistic stochastic models, which compromise between accuracy of basic synthetic traffic models and complexity of benchmarks [60][8].

2.4.4 Bimodal on-chip Packets

Another important feature of CMP intra-chip communication is the bimodal packet structure. In other words there are 2 different lengths : short and long packets. This comes from the fact that on-chip traffic is composed of cache coherency messages. Short packets are control packets just containing information such as source address, sharers of memory block, memory address, destination address etc. For instance, a read request, invalidation, write reply are short packets. The length of these packets depend on the architecture, number of cores and used cache coherence protocol, however, it can be stated, generally they are between 32 bits and 96 bits [1][24]. Long packets are simply packets that carry cache lines with signaling overhead. As mentioned previously, cache lines are the atomic unit of memory transactions of shared memory systems. Write request, read reply are such examples. Size of a cache line depends on the architecture, but generally they are between 32 bytes and 96 bytes (256 bits - 768 bits). In [61] authors have examined through execution of diverse applications that the percentage of long cache-line carrying packets constitute between 20%-30% of all generated packets. Fig. 2.14 shows the percentages of short and long packets obtained by the research in [61] for various PARSEC benchmark applications, where in their architecture short packets are 8 bytes (64 bits) long and long packets are 72 bytes (576 bits) long.

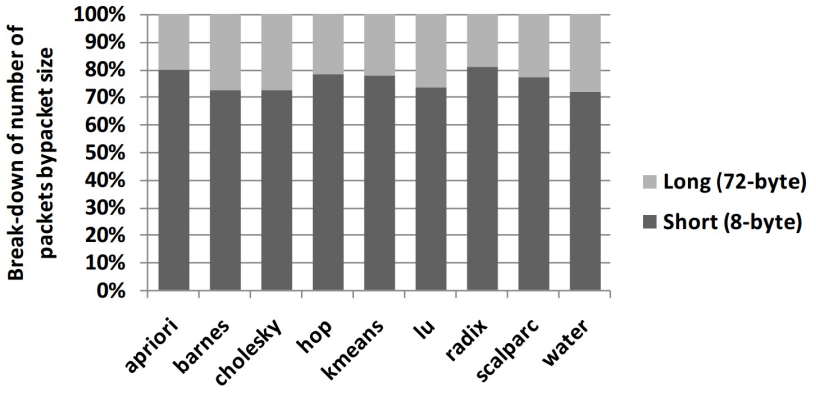


FIGURE 2.14: Percentages of short and long packets for various PARSEC benchmark applications (Image taken from [61]).

Hence, we have chosen to face our proposed channel access policy to a realistic generic model, which is composed of 25% of signaling packets and 75% of cache-line carrying long packets, which will be explained further in Section 4.3.4.2.

2.5 Conclusion

In this chapter, we have presented a general context of the multicore processors. First, the on-going trend for computational parallelism via integrating higher number of cores has been discussed and the provisioned boom in the core count in next few decades has been emphasized. Then, the architectural principals of shared memory CMPs has been introduced, which is important to understand due to its effect on the NoC traffic. We have explained the notion of general cache coherency briefly and referred to scalable cache coherency protocols for massive CMPs. The paradigm shift of the on-chip research community towards packetized NoCs and the primary electrical NoC topologies previously proposed have been presented. Next, we have introduced the advantages and disadvantages of several electromagnatic based solutions such as wired or wireless RF and optical to scale interconnects for 1000-core massive multiprocessors. The feasibility and scalability of wired RF technology was emphasized due to its mature and promised CMOS manufacturability. And lastly, we have presented a brief literature survey on the fundamental charachteristics of on-chip traffic rooted from the aforementioned cache coherency of shared memory. The requirement for accurate stochastic models to stress proposed NoC architectures has been pronounced.

We have then identified the following notions which will play a fundamental dimensioning role for our project proposal :

- Choice of RF-NoC for CMOS technology compliance.
- Preference for SWMR.
- Cache-coherency dominated traffic composed of bimodal packets.
- Choice for bimodal traffic model constituted of 75% small control packets and 25% cache-line carrying long packets.
- TSAR-based tiled multi-core architecture and distributed memory, which is orchestrated by DHCPP.

Next chapter details the WiNoCoD CMP architecture and proposed radio access technology.

Chapter 3

WiNoCoD Project and Wired OFDMA Based RF Interconnect

In Section 2.3, we review the state-of-the-art propositions for 1000-core interconnects. Wired RF distinguishes itself with its viability with today's technology due to CMOS compatibility [44]. Both of the proposed RF and optical proposals seem to provide necessary high bandwidth and scalable power budget, however there exists a significant problem. These architectures rely on utilization of dedicated *hardware circuitry* to create orthogonal channels, i.e. as many dedicated set of hardware circuits as number of channels. Optical interconnects require ring resonators, filters, modulators for composing channels on different wavelengths. Whereas, in RF frequencies, electrical CMOS devices such as local oscillators, filters and mixers are used. There are two important drawbacks with this approach. First of all, intra-chip bandwidth demand keeps increasing in parallel with increasing core count, which results in the quadratic increase for these circuits to implant on-chip. Obviously, this tendency is not scalable in terms of area, power and budget. Next, we see that this type of FDMA (or WDMA) implementation is not reconfigurable or encloses a very limited reconfigurability. In other words, bandwidth can not be distributed among nodes, according to instantaneous traffic demands. We presented significant spatio-temporal heterogeneity of on-chip traffic in Section 2.4. ATAC presented in Section 2.3.1.1, uses a SWMR scheme which is not reconfigurable due to fact that modulators for each channel cannot be implanted in every transmitter [3]. Even for the architectures with limited reconfigurability such as [44], the bandwidth granularity is really low and channel allocation is done in analog level by switching circuitry which is not rapid and effective. Due to these constraints, state-of-the-art RF and optical architectures are dimensioned for the worst-case traffic, i.e. an on-chip node is guaranteed to have bandwidth which can sustain its peak traffic. However, we know that for most of the time traffic intensity for on-chip nodes are far lower than this peak, thus

making this scheme non-efficient due to redundant bandwidth allocation [8]. Besides, the only logical scheme supporting broadcast -*which is the most vital requirement for scalable cache coherence protocols*- SWMR is prohibitive in energy and not preferred.

To overcome all of these on-chip drawbacks and provide the necessary breakthrough with bandwidth reconfigurability, in 2012, WiNoCoD project (Wired RF Based Network on Chip Reconfigurable on Demand) is initiated by the partnership of ANR, ETIS-ENSEA, LIP6 laboratories, Supelec-IETR, NXP Semiconductors. At the heart of the project lies the revolutionary Orthogonal Frequency Division Multiple Access (OFDMA) based RF Interconnect for a NoC, which provides *digital level*, high granularity, rapid and effortless bandwidth allocation.

FIGURE 3.1: WiNoCoD is a project funded by French National Research Agency (ANR) with the aim of implementing the first OFDMA RF interconnect backed massive multiprocessor, with the partnership of ETIS-ENSEA laboratories, IETR-CentraleSupelec, UPMC-LIP6 laboratories and NXP Semiconductors

3.1 WiNoCoD On-Chip Architecture

WiNoCoD provisions a 2048 core generic massive CMP. A shared memory principle is adopted, so that the address space is accessible by all processing elements such as in TSAR architecture developed by LIP6 laboratory partner (Section 2.2.3). As we have mentioned previously, in general in Section 2.1, placing a single RAM block with a dedicated single memory controller in one part of the multiprocessor is not scalable for the future CMP systems with hundreds, thousands of cores [62]. In order to alleviate this effect, researchers have developed the *Non-Uniform Memory Architecture* (NUMA) principle [3]. WiNoCoD utilizes this type of a memory architecture, where physical

RAM of the system is partitioned over the chip, but any core can reach any part of the memory, through the sharing of a general address space. WiNoCoD architecture is composed of 512 tiles, including 4 processors and 2 Gbytes of RAM on each tile. Project plans to employ 1 TBytes of memory in total, which is divided in to uniform 512 pieces, 2 Gbytes of RAM slices. Our CMP is arranged in 3 different hierarchical levels due to sustain scalability and modularity, with each of its level containing a dedicated different NoC infrastructure (Fig. 3.2). These choices, as well as the RF based topology will be explained later. This approach provides a full modular scalability, so that new tiles can be added if needed in future.

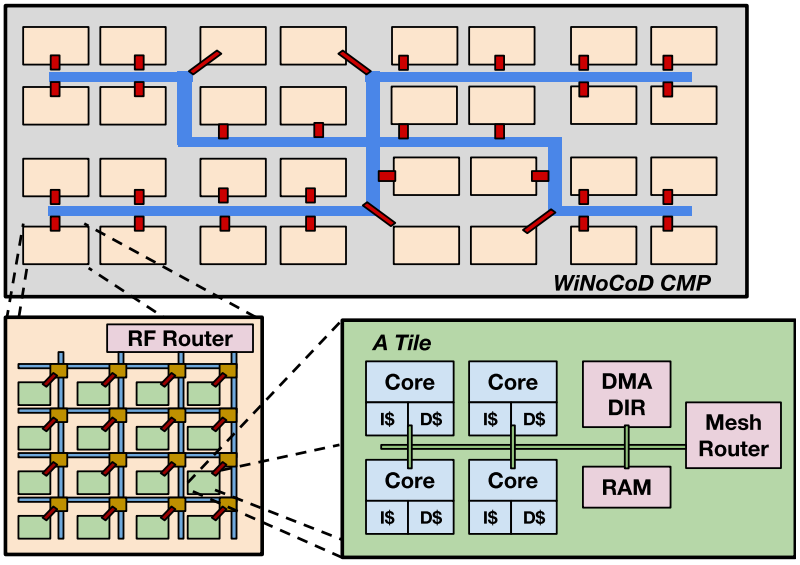


FIGURE 3.2: 3 level hierarchy of WiNoCoD architecture incorporating 2048 cores. Each level has a different dedicated NoC infrastructure

3.1.1 Need for independent communication layers

We have mentioned previously that a standard electrical NoC lacks the scalability to sustain the interconnection of hunderds of cores, especially due to high number of routers to traverse. Particularly taking our architecture as an example, if 2048 cores would be connected by a standard 2D mesh, we could have envisaged a topology configured as 64x32 mesh. Therefore, for the communication between the two farthest cores, a packet would have to traverse 96 routers. Assuming few cycles of processing latency in each router, one can see that this will lead to prohibitively large communication delays, even under low traffic load. And additionally, even under a low traffic intensity, the mesh network shall be congested easily, causing the saturation of the interconnection. However, using RF to interconnect every one or few cores is not scalable also. As we will highlight the details on the RF interconnection in next sections, the necessary transceivers lead to

significant amount of power consumption and surface area. For instance, the estimated surface area of a core is $0.35\,mm^2$ and it has been estimated by other WiNoCoD's other partners that the necessary RF transceiver is $1.17\,mm^2$. Moreover, the estimated power consumption of a core is $39\,mW$ and the power consumption of an RF transceiver is $309.5\,mW$. We can understand the ineffectiveness of employing an RF transceiver for each few number of cores, from these figures. In WiNoCoD, 2048 cores are provisioned, so that we make a hypothetical comparison for different number of RF transceivers, to understand the need of a hierarchical structure. Table 3.1 shows the percentage of required surface area and power consumption for all required RF transceivers and their percentages compared to total core and memory area and power consumption; assuming different number of cores are grouped together to be served by the RF interconnect. For instance, if 2 cores are grouped together, we need 1024 transceivers and if 1024 cores grouped together we need only 2 transceivers. However note that, this assumption does not include the area of the waveguide, which spans a considerable surface area, increasing substantially by increasing number of RF accesses, as we will mention in incoming sections.

From Table 3.1, we see that if each core has a transceiver, the total required surface area for RF transceivers consitute 75.53% of total area including surface area of cores and memory, and consume 88.09% of total power, which is quite unacceptable. Even if each 16 cores are grouped to use RF interconnect, the transceivers span 16.36% of total area and consume 31.62% of total power. In WiNoCoD, 64 cores are grouped at the highest layer to communicate with RF interconnect, where it corresponds to 4.84% of total surface area and 10.36% of total power consumption. These figures justify this choice. Considering all of these, WiNoCoD employs a 3-level hierarchical architecture, where each layer has a special type of interconnection.

3.1.2 3-level hierarchy

In this section, we present the 3 different modular hierarchical level of WiNoCoD, each with its dedicated type of NoC infrastructure. To demonstrate the feasibility of the proposed design, the surface and power consumption estimation is done [63]. 22 nm technology is targeted to be utilized in the project, and based on the equations given in [64], we estimate the normalized surface and power by using the characteristics of technology used in state-of-the-art components. For instance, the surface and power of a target technology's gate length N can be estimated by using the values from reference technology's gate length Q, with equations below, respectively :

TABLE 3.1: Estimated total surface area and power consumption for different number of RF transceivers, compared to total area and power consumption of 2048 cores and 1 TByte RAM.

Number of aggregated cores to communicate with RF.	Total area of required RF transceivers. (mm^2)	Percentage of total area of required RF transceivers compared to total area of cores and RAM	Total power of required RF transceivers. (W)	Percentage of total power of required RF transceivers compared to total power of cores and RAM
1	2396.16	75.53%	633.855	88.09%
4	599.04	38.16%	158.464	64.91%
16	149.76	16.36%	39.616	31.62%
64	37.44	4.84%	9.904	10.36%
256	9.36	1.25%	2.476	2.80%
1024	2.34	0.31%	0.619	0.71%

$$S_N = S_Q \, (\frac{N}{Q})^2 \tag{3.1}$$

$$P_N = P_Q \, 0.65^{log_{0.7}(N/Q)} \tag{3.2}$$

For instance, as we are targeting a technology of 22 nm gate length for processors, we have estimated the power and surface by using the ARM Cortex-A5 specifiactions which is built with $Q = 32$ nm gate length [65]. FFT/IFFT blocks are based on [66] and ADC/DAC circuits are based on [67], both with a target technology of 180 nm gate length.

3.1.2.1 A Tile

A *tile* is our lowest hierarchical entity where each of them incorporates 4 processing elements (cores) (each with their dedicated L1 instruction and data caches), a uniform portion of the total RAM (2 GBytes) and a Direct Memory Access (DMA) controller. All of these components share the same address space and are connected by a standard local crossbar. Crossbars provide a low latency and simultaneous communication, but their performance degrades drastically after the number of connected nodes increase a few. Therefore number of elements accessing to crossbar is limited to 7 in our architecture including 4 cores, memory directory, RAM slice and the mesh NoC router. We assume a

22 nm technology and a standard 32-bit core. However, our architecture is independent of the processor type, that any existing standard processors such as MIPS32, PPC 405, SPARC V8 etc. can be used. Each core has an estimated surface area of $0.35\,mm^2$ and power consumption of $39\,mW$. 2 GBytes of RAM slices in each tile has an estimated surface area of $0.03\,mm^2$ and a power consumption of $11.29\,mW$. At the end total tile surfarce area is $0.91\,mm^2$ with all other elements including crossbar, routers, memory controller etc. [68]. Each tile has a mesh router which is connected to the crossbar at one end, where all elements in the tile can reach, that they can access the 2D mesh network. Note that, the tile structure is based on the previous TSAR project (Section 2.2.3).

3.1.2.2 A Tileset

Next hierarchical element in our architecture is a *tileset* which is composed of 16 tiles where they are interconnected with a conventional electrical 4x4 2D mesh network. The packets travel using a simple x-y routing with virtual channels. Each router (except edges) has 4 different interfaces to the adjacent tiles' routers each with a dedicated transmitting and receiving buffer. The nodes in tiles access this level, by a router and wrapper inside the tile. As at this level we switch to packetized NoC paradigm, the necessary fragmentation and defragmentation is done by these entities. We have mentioned the viability of a 2D electrical mesh network with tens of nodes in Section 2.2. Concerning our provisioned latency constraint, the number of nodes in a tileset is limited to 16. There are 32 tilesets in WiNoCoD CMP, each with an access to RF transmission line. With all its elements, a tileset has an estimated surface area of $14.62\,mm^2$ [68].

3.1.2.3 Inter-tileset Communication

The most distinctive feature of our CMP architecture is its OFDMA based RF interconnect. A core who wants to transmit a message to a memory controller in another tileset (and vice versa), sends the message via crossbar to the mesh NoC, and finally using the mesh to the RF transceiver. The message is propagated on the RF transmission line and reach the RF transceiver of the destined tileset. Following this, using the reverse path through the mesh network and crossbar, it reaches to the terminal position. We will describe the enabling technology and the details of our OFDMA based RF interconnect in following sections. Finally, with all its components including transmission line and RF transecivers, whole estimated surface area of our CMP is $476.83\,mm^2$ and a total power consumption of $95.27\,W$ [63].

TABLE 3.2: Estimated surface area and power consumption for certain elements in CMP

	Area (mm^2)	Power (mW)
Core	0.35	39
Tile (4 cores)	0.91	167.29
Tileset (16 tiles)	14.62	2977.25
RF Transceivers (x32)	1.17	309.5
Transmission Line	8.71	—
Total CMP (32 tilesets)	476.83	95272

Table 3.2 sums up the values for estimated surface area and power consumption for certain elements in our CMP.

3.1.3 Details of Cache Coherence Protocol

We have described the notion of cache coherence and its importance for multiprocessors in Section 2.1. NUMA type memory architecture explained previously, is the key to programmablity of massive CMPs with thousands of cores [62]. An important design goal for a cache coherence protocol and shared memory architecture is to introduce a high *usability*, which refers to the abstraction of hardware to the programmer [69].

Our project draws many parallelism with the Tera-Scale Architecture (TSAR) project, which is another 1024-core generic shared NUMA memory CMP design, but without an RF interconnect as we have mentioned previously in Section 2.2.3 [1]. We employ a *Directory Based Hybrid Cache Coherence Protocol* (DHCCP) in WiNoCoD, like in TSAR. A similar approach is adopted in 1024-core, optically interconnected ATAC [3].

Each 2 GBytes of shared memory element in a tile (RAM slice) is associated with a *directory* and access to this memory is regulated via a Direct Memory Access (DMA) unit. Thus, the cache coherence is hardware initiated. A *Write Through* approach is adopted, where in case of a core want to write a data to an address line, it transmits a write request to the directory responsible for the corresponding address line. We have mentioned previously in Section 2.1, that these type of write request messages contain *cache lines*, which is including the raw data for computation. The address space for 1 TBytes, where each line is represented with 40 bits, is evenly divided among 512 tilesets. A directory in a tileset is responsible for the address lines spanned in its interval. Therefore, for instance, if the intended address line is spanned by the directory in the same tileset, the message does not go outside tile, and only uses crossbar to reach DMA. Similarly, if the adress line is in a different tile in the same tileset, it only uses the mesh network to reach. If the address line is in a tile of a different tileset, it has to use RF interconnect. Hence, exploiting spatial locality in memory programming is essential to

decrease traffic load in higher hierarchies. After receiving the write request, firstly the directory checks whether other cores have a copy of this address line. If so, it transmits *invalidate* message to the sharers, so they stop using the copy of this address line in their L1 caches, as it will change by the modification of the core who wants to write a new content to this address line [1]. This procedure is illustrated in Fig. 3.3, where CPU-A sends a write request for ADDRESS-X; CPU-B and CPU-C have a copy of it. The directory in charge of this address line, DIR sends invalidation messages to them.

In contrast, for the *Write Back* policy, when the core updates its L1 cache, the main memory (directory) is updated only when the modified cache line is updated. Even though write through policy is more bandwidth intensive, write back policy is preferred due to its scalability.

Similarly for reading data, L1 caches of cores gather the copy of the main memory from the responsible directory and DMA of the associated address line. In a sense, we can state that the distributed main memory is a L2 cache for the cores.

A directory is a simple register, containing data associated to each cache line it is responsible. For each cache line, it contains the IDs of the cores who has a copy of it. And of course, in the associated line in RAM, it has the original raw data. When we have thousands of cores, one can see that it is not possible for a directory to keep the IDs of hundreds or thousands of sharer cores, for each cache line. Depending on the length of a cache line, entries in a directory changes, however for instance for the case of 2048-cores and 64 bytes of cache lines, each directory in a tile with 512 Gbytes of RAM is responsible of 8 billion cache lines. Assuming ID of each core is represented by 10 bits, each of the 8 billion cache line in a directory should have a 256 Kbyte of memory, which is practically impossible.

In order to alleviate this, distributed hybrid cache coherence protocol (DHCCP) is adopted in WiNoCoD, as in [1][3]. When the number of sharers exceeds a certain threshold (for instance 8), the directory keeps the *number of sharers*, rather than ID of each sharing core. The directory in a tileset is depicted in Fig. 3.4. If the number of sharers are lower than the threshold, in case of an invalidation, directory transmits invalidation messages to each of the sharing cores. However, if this threshold is exceeded, directory simply broadcasts this invalidation message to all 2048 cores. Then, it counts the number of acknowledgement messages from the intended cores and validates it by comparing to the number of sharers it holds in the register. This bandwidth intensive cache coherence protocol is a price to pay for the scalability of future massive CMPs. One can understand the quest of developing an effective reconfigurable and broadcast capable interconnect such as in WiNoCoD, due to this cache coherency protocol.

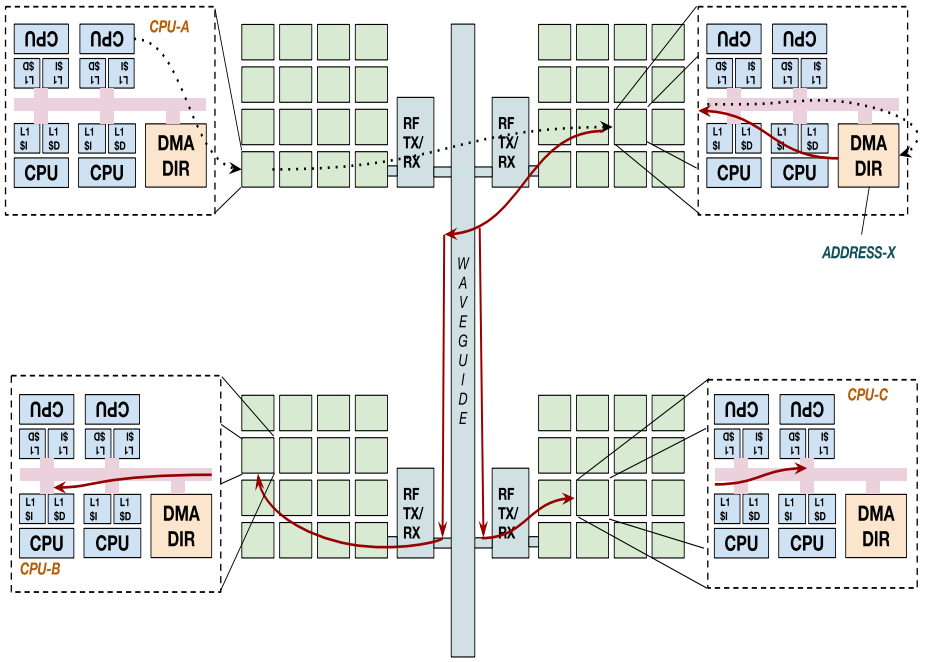


FIGURE 3.3: Illustration of a write request initiated by a core in WiNoCoD, where the intended address is currently shared by 2 other cores (CPU-B and CPU-C).

Memory Directory in a Tileset

Address Line	Flag	Sharer Cores			
#0000000	0	C-0013	C-0048	•••	C-1004
#000000F	1	27 *(Number of sharers)*			
⋮				⋮	
#0000100	0	C-0203		•••	

FIGURE 3.4: Implementation of memory directory in a tileset. With hybrid cache coherence protocol, for each cache line in the RAM, if the number of cores are lower than a threshold, sharer core IDs are explicitly registered, if not, only number of sharers are stored.

3.2 Basics of OFDMA

In this section, we explain the basics of OFDMA technology along with its possible benefits for on-chip interconnects. OFDMA is a medium access scheme based on Orthogonal Frequency Division Multiplexing (OFDM).

3.2.1 OFDM

Orthogonal Frequency Division Multiplexing (OFDM) is a modulation technique that transforms a large bandwidth signal into many, orthogonal narrow band channels and encodes digital information on frequency domain rather than time domain. Since its first standardization in 1995 for Digital Audio Broadcasting (DAB) [70], it has conquered and revolutionized all fields of digital telecommunications and became the popular physical layer solution due to its numerous advantages [71]. Main advantage is its capability to resist to multipath issue, thanks to a simplified equalization. However, most appealing advantage for on-chip application is the capability to multiplex several transmissions in the same signal with a high level of flexibility and dynamicity.

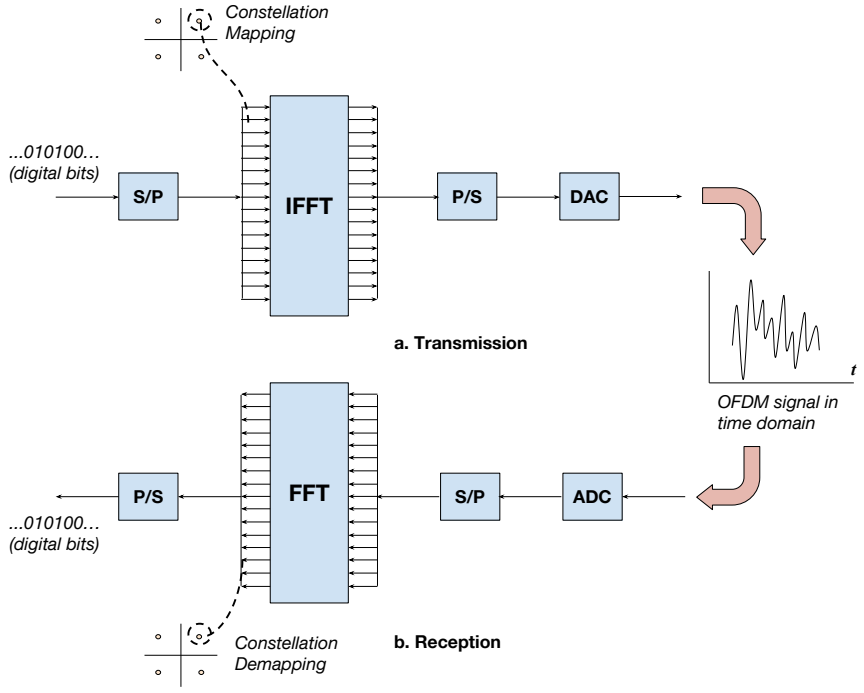


FIGURE 3.5: Transmission and reception chain of an OFDM modulator/demodulator

OFDM is remarkably different than other conventional FDM systems, due to its method of generating orthogonal channels. As stated previously, in OFDM, a large band signal

(high rate) is decomposed to many parallel narrow band (lower rate) symbols. To implement this, the digital data is first mapped to constellation symbols such as BPSK, QPSK, M-QAM etc. and each constellation symbol is associated with a *subcarrier*. A subcarrier is the atomic frequency unit in an OFDM signal, or in other words it is one of these parallel narrow band signals. Then an Inverse Discrete Frequency Transform (IDFT) is applied to the parallel vector of N subcarriers, where each of them is now a complex number associated with the encoded constellation symbol (each complex number represents a certain number of bits). In order to perform the frequency transformation rapid, a Fast Fourier Transform (FFT) or Inverse Fast Fourier Transform (IFFT) is applied [72]. The result of this transform gives again a vector of N complex numbers. Following, this vector of N points is serialized and converted to a time-domain signal with the appropriate data rate. This resulting signal is called as an *OFDM symbol*, and unless stated else we will refer it as a *symbol* throughout the rest of this thesis. Fig. 3.5 shows the transmission and reception chain of a typical OFDM modulator-demodulator. At reception the exact inverse of the aforementioned procedures are followed to decode the received OFDM symbol, whereas FFT is used, instead of IFFT at the TX. The time and frequency representation of an OFDM symbol can be illustrated as in Fig. 3.6. Loosely speaking, in the frequency axis each subcarrier (orthogonal channel) of OFDM symbol can be seen as a superposition of sinusoids with different amplitudes, phases and frequencies. It is important to remark that an OFDM symbol is the atomic decodable unit, in this system. The whole OFDM symbol should be decoded to extract the data, as data are encoded not in time, but frequency domain.

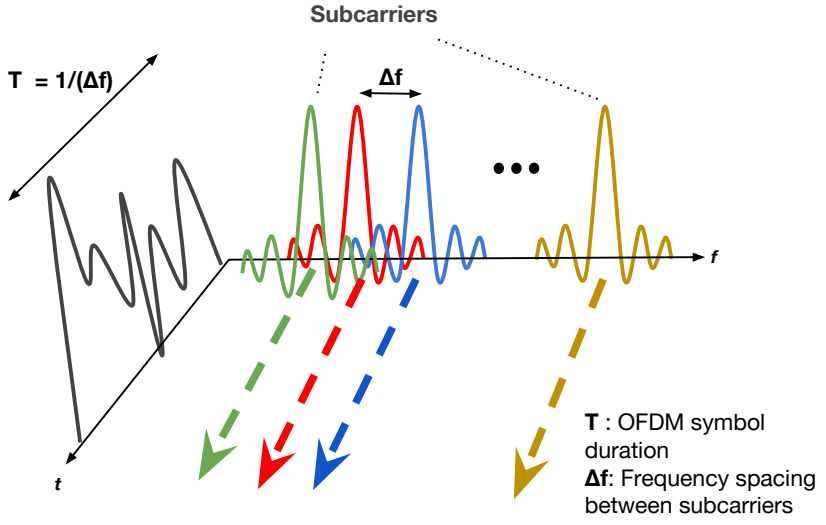


FIGURE 3.6: Representation of an OFDM symbol with duration T both on frequency and time domain.

At this point we shall make certain points clearer on mathematical basis of OFDM. As it can be seen from Fig 3.7, the frequency domain is spanned by subcarriers which are formulated by a *sinc* function [73]. This comes from the fact that the sampling and transformation to analog domain are done with rectangular pulses, and the frequency transform of this gives a $sinc(x) = \frac{sin(x)}{x}$ function. Another advantage of OFDM roots from these sinc functions, as they are still orthogonal mathematically at the center of frequency spacings, even they overlap. This allows for the maximum bandwidth efficiency [74]. Reciprocal of the frequency interval between subcarriers Δf gives an OFDM symbol's duration : $T = 1/\Delta f$.

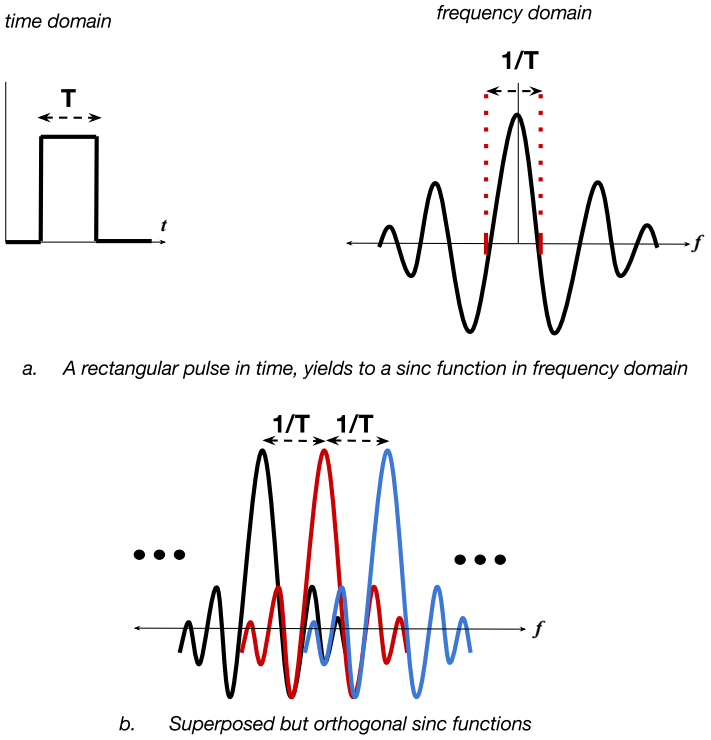


FIGURE 3.7: (a) Subcarriers in frequency domain are represented by *sinc* function, as it is the frequency transformation of a rectangular interpolator which is used to convert digital samples to analog OFDM signal. (b) Even though sinc functions are superposed in frequency domain, they are still mathematically orthogonal at the center of frequency spacings. This gives the maximum spectral efficiency.

Note that, the resulting OFDM time domain analog signal still incorporates a vector of digital data, however coded on frequency domain rather than time domain. This property has numerous advantages. The multi-path effect in transmission channels is the result of multiple constructive and destructive copies of the transmitted signal superposed at receiver, distorting the content of the information. Thus an equalization is needed. It results in different gains through the spectrum, which is also referred as *frequency selective channels*. For OFDM signals, this means different channel gains for each subcarrier, which can be mitigated by a simple multiplication with the inverse

channel gain, as mentioned previously. Apart from equalization, even just a portion of digital data on subcarriers with relatively better channel gains can be recovered.

3.2.2 OFDMA

Orthogonal Frequency Division Multiple Access (OFDMA), on the other hand is an OFDM based multiple access scheme, featuring all physical layer advantages of OFDM, along with an efficient bandwidth reconfigurability [75]. In parallel with OFDM, it gained popularity for the multiple user telecommunications standards. The key point of OFDMA is its powerful dynamic bandwidth reconfigurability, which allows assigning different subcarriers to different users, (thus data rate), on every symbol. A user only encodes information on its allocated subcarriers at transmission as explained previously, and nulls the rest (encodes *zero information* before IFFT) to allow other users in the system access the medium. The encoding of digital data on allocated subcarriers is illustrated in Fig. 3.8 via a simple example, where TX-2 is allocated the 32 subcarriers between No. 32-64.

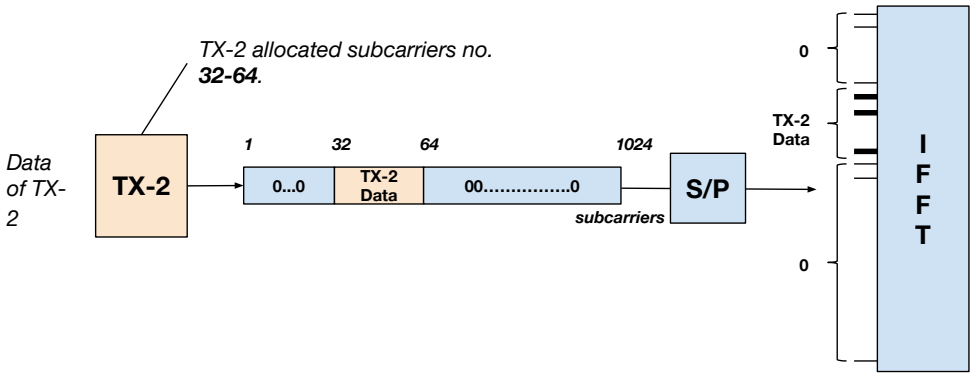


FIGURE 3.8: Encoding of digital data by TX-2 on its allocated subcarriers, where it pads 0s to remaining subcarriers.

This way, contention resolution in the system is greatly simplified. Note that, this procedure can be implemented in *digital domain*, i.e. by just manipulating the bit vector before the IFFT. This task can be performed by a simple microprocessor or a digital circuit. In addition, as all nodes in the system decode the same received OFDM symbol, they can receive the transmitted by all other nodes, thus OFDMA inherits an intrinsic broadcast capability. The logical transmission and reception between nodes in an OFDMA based medium (in case our RF interconnect) is illustrated in Fig. 3.9. For instance, TX-1 (transmitter of Tileset-1) encodes its data on its allocated subcarriers,

TX-2 encodes its data on its allocated subcarriers etc. and RX-1 (receiver of Tileset-2), RX-2, etc. receive all transmissions from each tileset.

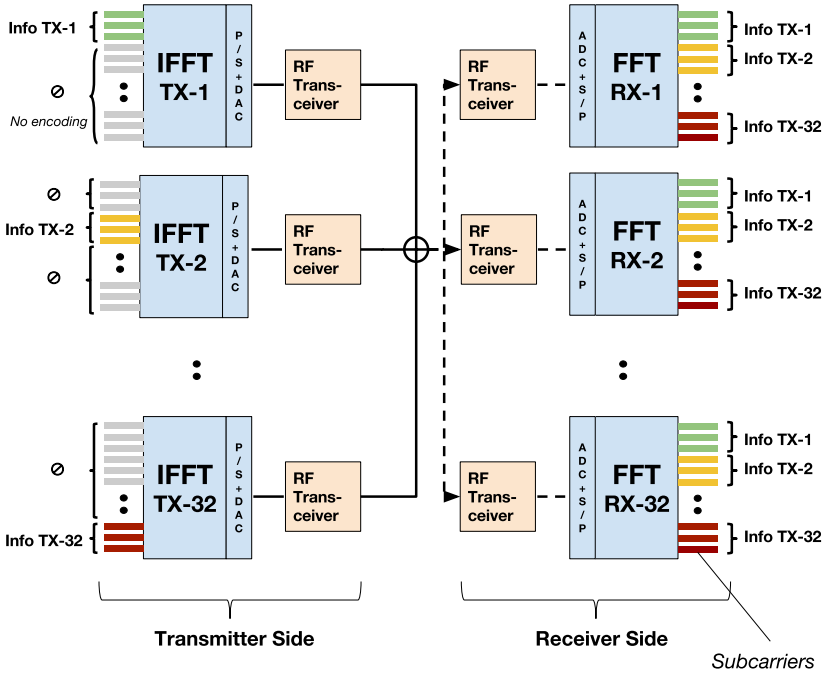


FIGURE 3.9: Communication between nodes in an OFDMA medium using different subcarriers. Note the intrinsic broadcast capability and nulled subcarriers.

In contrast to OFDMA, as explained in detail in Section 2.3 conventional FDMA (RF) and WDMA (optical) networks rely on static, pre-tuned CMOS or optical circuits to generate orthogonal channels. Only few of the proposals has a limited reconfigurability option to allocate these channels to nodes dynamically, which is unacceptable for the highly bursty on-chip traffic. In addition, as far as we know, a digital bandwidth allocation has not been introduced before to this field of RF-NoC, which allows designers to implement highly basic, adaptive, and rapid algorithms. Besides, the robustness and bandwidth efficiency of OFDM signals makes it a good candidate for interconnects. Considering the strong spatio-temporal heterogeneity of on-chip traffic and the vitality of multicast-broadcast cache coherence messages, we can better perceive the technological leap that OFDMA can bring to 1000-core CMPs. In a way, we can claim that OFDMA Interconnect implements a SWMR cross-bar which allows re-arbitrating the bandwidth for each transmitter on every symbol.

3.3 OFDMA Based RF Interconnect

OFDM and OFDMA's potential was already provisioned when it was first theorized in 1957 [76]. However at the time its implementation on field was limited to few military applications. This was due to the fact that the realization of the system was costly and not feasible with the technology of the era. Until 1980's, OFDM and OFDMA was not introduced to market, however it received attention from the academia and several articles and patents were issued. With exponentially increasing abundance of silicon resources, FFT chips have become a reality for OFDM and OFDMA to conquer the market and become the standard modulation of many recent communication standard. Examples of systems that use OFDM both on wire and wireless are ADSL, IEEE 802.11 (WLAN), 4G (LTE), digital TV and radio etc. With recent technology, even Ultra Wideband systems (UWB) [77], or intra-data center OFDMA based optical interconnects [78] are proposed.

Considering that the time for enabling technologies to implement OFDMA on-chip has come too, WiNoCoD project was initiated to pioneer this field. We expect a similar tendency with the above mentioned standards, that OFDM shall be the preferable modulation for on-chip communication in near future. As wired RF is a feasible approach due to CMOS compatibility (Section 2.3), it is preferred for WiNoCoD's interconnection.

As mentioned in Section 3.1, in WiNoCoD chip, 32 tilesets are interconnected via an serpentine, cross-shaped RF transmission line for the inter-tileset communication as in Fig 3.2. Each RF interface in tilesets, has an OFDM modulator and demodulator. The packets that are generated inside a tile in tileset, which are destined to a tile in another tileset, traverses the electrical mesh network and reaches to the RF access point. Provided converter technology by our partner NXP envisions a 20 GHz bandwidth for the system. Based on the design constraints and circuit simulations, most suitable spectrum is chosen between 20-40 GHz [79]. It is decided to have 1024 subcarriers, thus 1024-point FFT and IFFT blocks are required. Hence, as we have a 20 GHz bandwidth with 1024 subcarriers, we have subcarrier frequency spacing of $19.53 MHz$, where an OFDM symbol duration is $T = 1/19.53\,MHz = 51.2\ nanoseconds$. This value is approximated as 50 ns for ease of use throughout the manuscript. Table 3.2 summarizes these figures. One may question the choice for 1024 subcarriers in the system as there are only 32 tilesets are intended to use it. As number of subcarriers increase, the size of the FFT (and IFFT) increases, which at the end increases the complexity and computation time for the modules. More specifically, the complexity of FFT (IFFT) computation increases in $\mathcal{O}(log_2(N_{subcarriers}))$, with number of subcarriers [80]. Therefore, in a most basic sense, having FFT (IFFT) blocks of 32 would simplify the complexity 5 times compared to 1024-size FFT/IFFT modules. However, at the start of the project, 1024 subcarriers

TABLE 3.3: Characteristic parameters of WiNoCoD's OFDMA interconnect

Bandwidth	Number of subcarriers ($N_{subcarriers}$)	Sampling Frequency / Bandwidth Per Subcarrier (Δf)	Symbol Duration ($T = 1/(\Delta f)$)
20 GHz (20-40 GHz spectrum)	1024	19.53 MHz	51.2 ns

are dimensioned, which allow for a finer granularity of bandwidth. For instance, as we will investigate in Section 4.3.3.6, this resolution allows for the proper and effective signaling in our system. In addition, with further development the OFDMA based RF interconnect may need to sustain much more on-chip nodes, which require this fine granularity scheme for bandwidth partition.

With ever increasing demand for computational power in following decades, 100s of GHz of bandwidth shall be required for sustaining on-chip communication. In this thesis and WiNoCoD project, the operational bandwidth is restricted to 20 GHz, due to the currently feasible state-of-the-art technology provided by NXP. However, in parallel with developing silicon technology, one can expect for the on-chip OFDM bandwidth to increase, to keep up with the rising data rate demand. Actually, with its digital nature, reconfigurability and reliance on less amount of circuitry, we can designate the proposed on-chip OFDMA interconnect as a highly viable approach for the future trends.

3.3.1 RF Controller

A packet that is going to be transmitted by a tileset through RF is welcomed primarily by the RF controller of this tileset. These packets (composed of flits) are processed if necessary -*fragmented or defragmented, extracted or padded information such as source ID etc.* and inserted into the transmission queue. As mentioned previously, most important reason of using OFDMA on-chip is to reallocate bandwidth among different transmitters. There are varios approaches and algorithms to allocate subcarriers and modulation orders among transmitters, through using a central arbiter or decentralized synchronous decisions. These methods which constitute the backbone of this thesis work, are explained in following chapters. Thus, RF controller makes certain decisions according to information such as queue state coming from the transmission buffer (TX) and the traffic information coming from other tilesets or a central intelligent unit, and changes the configuration of subcarriers, where it will encode data. With the modulation order on each subcarrier, this determines the throughput of the tileset on current symbol. As it is reviewed in Section 3.2.2, the data in TX buffer is fetched and encoded on dedicated

subcarriers along with the chosen modulation order and the rest of the subcarriers are leaved idle. In addition, RF controller can be also made responsible of scheduling and sorting of packets, according to needs of architecture. Note that, this RF controller is completely digital thanks to OFDM, which allows a vast range of possibilities for physical and link layer actions. This intelligent module can be implemented as a simple single or multiple multiprocessors, or a basic digital circuit, based on requirements.

This thesis work focuses on the utilization of the frequency resources of the proposed OFDMA interconnect optimally as possible, taking into account the constraints of the on-chip environment. The main goal is to reduce the waiting times in the transmission side queues as much as possible, as it is directly related to the allocated data rate (i.e. subcarriers) to a tileset. The state of the receiver side queues, or the intra-tileset communication (i.e. how flits transmit inside a tileset) is out of scope of this thesis. We assume that the intra-tileset mesh network can serve the packets in the receiver queues with a consdirably higher rate compared to OFDMA transmission. In other words, the received packets can be sent to their destination tiles via the electrical mesh network with a constantly operating router. Without loss of generality, we assume that the packets do not have to wait for a long time to be transmitted. For instance, an OFDM symbol is approximately 50 ns and we can assume a router can serve one flit every ns. In addition, we do not also take into account the any other metric in the system other than transmission queue states to allocate the subcarriers, such as traffic in intra-tileset traffic etc..

3.3.2 RF Front-end

3.3.2.1 Transmitter Side

The number of flits to be transmitted by a tileset on waveguide, is fetched from the TX queue in the RF Controller, according to number of subcarriers (and modulation order) on that OFDM symbol. Then it is sent to RF Front-end which includes OFDM modulation. In Section 3.2, we have examined OFDM and OFDMA from an upper-layer perspective, abstracting over the underlying electronics. Now, we provide a more in-depth view. At first step, the bits to transmit are parallelized and mapped to constellation symbols with chosen modulation order. Constellation belongs to BPSK and M-QAM modulation. This is a basic encoding mechanism in digital communication : bits are represented by *constellation symbols*, which can be formalized as a complex value, $a + jb$. Each of 2 indexes of this value, are the amplitude levels belonging to *in phase* and *quadrature* components. Then these constellation symbols are mapped on subcarriers. Rest of the subcarriers remain idle. Then IFFT gives a N element vector

of complex values. These I/Q values are serialized and fed to two seperate Digital-to-Analog Converters (DAC).

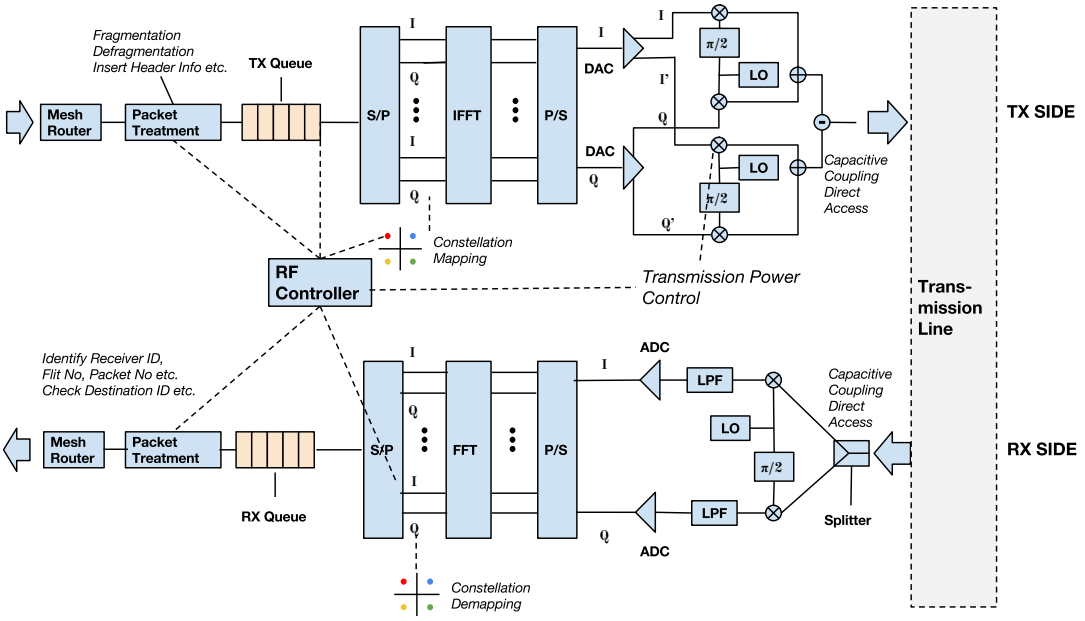


FIGURE 3.10: The detailed illustration of transmission and reception RF interface of a tileset.

The digital modelling and testing of the RF front-end is done by the thesis work which is also in collobaration with WiNoCoD project of A. Briere [63]. Two other PhD students working in the project, Frederic Drillet and Lounis Zerioul are responsible of the analog design, parametrisation and the simulation of the utilized CMOS components in the RF front-end.

The up-conversion mixers combine a baseband signal with a local oscillator signal. Mixing occurs in a MOSFET, whose gate and drain are respectively fed by the local oscillator and the baseband signal. The higher frequency output is recovered in the MOSFET source. A drawback of such a device is the weak LO/RF isolation. It leads to a high power LO carrier in the output spectrum. As the local oscillator frequency is 30 GHz, which is the middle of our 20 GHz bandwidth, it needs to be suppressed. Thanks to the differential outputs of the DAC, two IQ-Modulators can work together to do so. Besides avoiding interference caused by image frequencies they can reduce the LO level in the output. As we use the same local oscillator for both of IQ-modulators and opposite I-Q signals, the IQ-Modulators outputs are subtracted in a differential amplifier to perform this suppression. Then this signal is amplified by a Low-Noise Amplifier (LNA) and transmitted on waveguide. This procedure is illustrated in Fig 3.10.

Note that, this operation is done synchronously every T = 51.2 ns at every cluster's RF interface. At the end of this operation the resulting analog signal is the *OFDM symbol*. All the analog components in the transceiver is projected to be manufactured in 250 nm SiGe technology for demonstration purposes.

3.3.2.2 Receiver Side

The reception is done synchronously every T = 51.2 ns as transmission, too. The received signal from the transmission line is amplified and fed to a separator circuit, mixers and 30 GHz local oscillator to obtain in-phase and quadrature components. Low Pass Filters (LPF) are used for down-conversion. Then I and Q components are converted to digital domain by our Analog-to-Digital (ADC) components. After Serial to Parallel conversion this vector of I and Q values are converted to frequency domain by an FFT block. The resulting constellation symbols are demapped to bits, serialized and finally switched to the RF controller. The detailed reception chain of the RF Interface is shown in Fig. 3.10. Note that, the exact inverse of transmission operations is done to retrieve data.

Utilized FFT/IFFT processors are estimated to be manufactured with 22 nm CMOS technology. We estimate the area of each of these modules as 0.31 mm^2 and power consumption of 67.5 mW. FFT/IFFT computation duration can be omitted and treated as a *pipelined latency* through symbol-by-symbol communication. Each of ADCs and DACs are designed with 22 nm technology and have an estimated surface area of 0.12 mm^2 and power consumption of 81 mW [63].

3.3.2.3 Transmission Line and Access

M. Hamieh has developed and simulated transmission line and access for his thesis in collaboration with WiNoCoD Project [79]. He has designed a cross-shape transmission line, to minimize the distance between two farthest tilesets in CMP, as in Fig. 3.2. In this configuration maximum distance between two nodes is 80 mm. Cross-shape design also gives an additional advantage, by providing a near flat frequency response over 20 GHz bandwidth. Based on the simulations, through the spectrum of 20-40 GHz, attenuation is measured in range of -40 dB and -50 dB between two farthest nodes and relatively non-varying [79]. This simplifies the utilization of any subcarrier in the system without changing transmission power drastically and does not require equalization.

A state-of-the-art silicon Microstrip Transmission Line delivered by NXP Semiconductors is used. The height of the transmission line is 8.78 µm with a permittivity of 4.2 and loss tangent of 2.10^{-4}. In order to minimize the metal loss, characteristic impedance

is determined as 30 Ω. A new access mechanism to transmission line via transistors has been developed by M. Hamieh, preferred over existing capacitive coupling or direct access schemes. This new method reduces the frequency reflections phenomenon, frequency response fluctuations and attenuation compared to capacitive coupling and direct access [79].

The linear attenuation through the transmission line is approximately 0.2-0.3 dB/mm. For instance, assuming a bit error rate of 10^{-8}, the required minimum transmission power on a single subcarrier, between two farthest nodes is -39 dBm for BPSK, and -25 dBm for 64-QAM. Note that, the required power increases linearly with the number of used subcarriers [79]. Fig. 3.11 illustrates the proposed access method via transistors. In Fig. 3.12, the frequency response for the transmission between two farthest tilesets is shown for the newly proposed access mechanism via transistors, compared to two existing access schemes; direct access and capacitive coupling. As it can be seen from Fig. 3.12, the proposed access mechanism results in much lower attenuation, between -10 dB and -42 dB. As frequency increases the attenaution in dB increases linearly. We see that capacitive coupling provides a relatively flat frequency response, but with a much higher attenuation of approximately -60 dB.

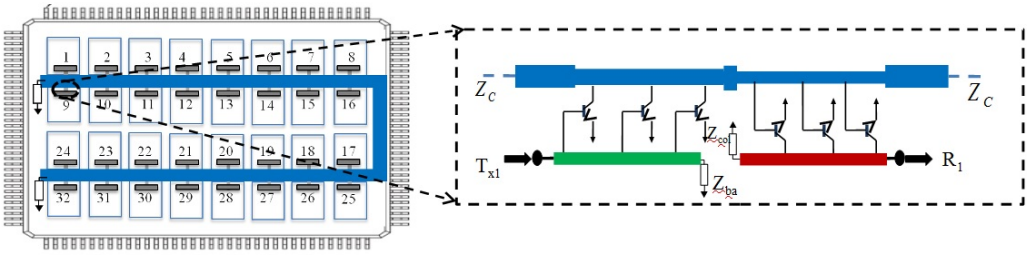


FIGURE 3.11: Proposed access via transistor mechanism for WiNoCoD (Image taken from [79]).

3.4 Conclusion

In this chapter, we have presented the WiNoCoD project along its contributing partners and specified the intended 2048-core CMP architecture with the multi-level NoC. We have defined the lowest hierarchical level composed of 4 cores with a system RAM slice and responsible memory directory. Tilesets composed of 16 tiles, which are interconnect with a conventional electrical 2D mesh network has been presented. These tilesets are the highest hierarchical elemenets in the NoC, that they are interconnected by the wired RF network, therefore they constitute the main interest of focus for this thesis work.

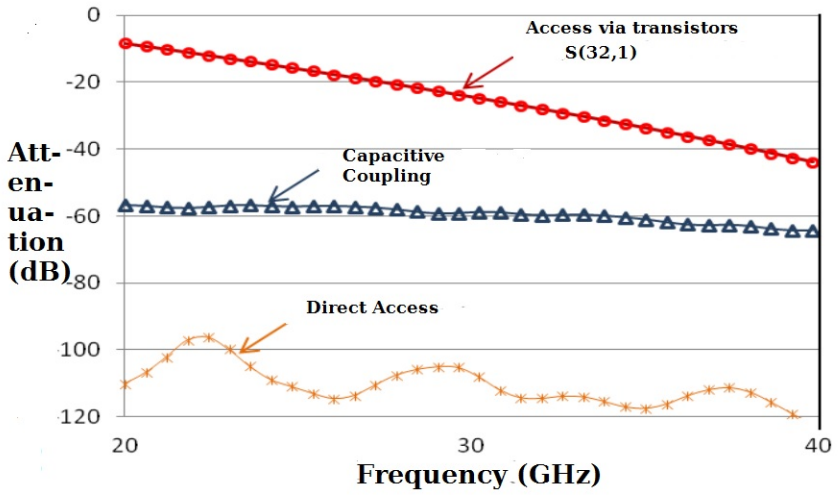


FIGURE 3.12: The frequency response between two farthest tileset on transmission line for the developed access mechanism via transistor compared to capacitive coupling and direct access (Image taken from [79]).

The necessary information is derived from the thesis work of A. Briere, which is also in collaboration with WiNoCoD project.

Next, we have explained the scalable hybrid cache coherency protocol to be used in WiNoCoD. This is important for the goals of this thesis work, as cache coherency packets constitute the traffic circulating in the NoC infrastructure, which also includes the RF level.

Before presenting our OFDMA based RF interconnect, we have introduced the notion and basics of OFDM. Its revolutionary advantages have been listed. It is important to highlight the preliminary information on OFDM and OFDMA, as this project is to first to consider OFDMA for an on-chip CMP interconnect, to the best of our knowledge.

Then, we have specified the details of the RF Front-ends and utilized transmission line. The necessary information is derived from the thesis works of M. Hamieh, F. Drillet and L. Zerouil which are also in collaboration with WiNoCoD project. The rest of this thesis work focuses on the OFDMA based allocation issue for the WiNoCoD architecture, so that it is essential to based on the parameters and details of the wired RF infrastructure, we have seen in this chapter.

Chapter 4

RF NoC Bandwidth Allocation Problem

In Chapter 3, we have presented the 2048-core architecture of WiNoCoD and features of its OFDMA based RF interconnect. As mentioned previously, using OFDMA for an RF interconnect is a pioneering attempt that would enable reconfiguring bandwidth effectively and rapidly among tilesets, based on their changing demands. The purpose of this chapter is to define the bandwidth demands of tilesets, in other words cache-coherency packets created by each tileset which are destined to locations in other tilesets to use this RF interconnect, more precisely from a queuing theory perspective. Each symbol, each 1024 subcarrier of the interconnect, can be regarded as an element that can serve these packets. Briefly, this thesis work aims to develop strong algorithms which allocate subcarriers dynamically to tilesets effectively, while respecting the nanosecond scale constraints of the architecture, thanks to the high bandwidth reconfigurability of OFDMA. Therefore in this chapter, firstly we formulate this bandwidth allocation problem we encounter, considering physical constraints of WiNoCoD and main metrics to improve in terms of on-chip requirements. Then, we review the vast literature on similar bandwidth allocation problems, belonging to diverse fields of cellular communications to processor scheduling. Lastly, the required concepts and notions which we use for the bandwidth scheduling in WiNoCoD, are explained in detail. In a nutshell, this chapter forms a theoretical preliminary basis for our bandwidth allocation algorithms, which we propose in following chapters.

4.1 WiNoCoD's OFDMA RF Interconnect

4.1.1 Motivation

WiNoCoD's innovative OFDMA RF interconnect aims to improve the performance of the network-on-chip laid on a 2048-core CMP architecture, by changing users of subcarriers dynamically. Recalling the specifications given in previous chapter, there are 32 tilesets, where each contain 64 cores. We have seen that each tileset has an access point to transmission line via its OFDMA modulator. Thanks to the intrinsic broadcast capability of OFDMA, any broadcast/multicast packet need not to be multiplied, as transmissions by a user are received by all others (SWMR). This shall increase system capacity drastically considering the requirement of large number of broadcast/multicast packets of a threshold based hybrid directory based cache coherency system. In previous section, we have seen that there exists a transmission buffer where packets coming from intra-tileset mesh network which are destined to other tilesets are stored, and served by OFDMA RF interconnect in a FIFO manner. Similarly a receiver buffer exists at the end of the OFDMA demodulator, receiving all transmissions from other tilesets. Wrappers process headers of these transmissions and store packets which are destined in a tile in its tileset.

A subcarrier on an OFDM symbol can be seen as an element, serving transmission of information of 1,2 etc. bits based on the utilized modulation order BPSK, QPSK etc. As mentioned previously, primary motivation behind using OFDMA on-chip is its high reconfigurability, which refers to the ability of changing owners of subcarriers over time. Therefore main goal shall be to exploit this reconfigurability as much as possible, concerning unique constraints of our OFDMA interconnect and unorthodox requirements of on-chip traffic. Hence, our problem can be formulated as allocating subcarriers in different time instants (i.e. one or several OFDM symbols) to different transmission queues (Fig. 4.1). Assuming that receiver side queues can be served with a constant high rate, (which means that received packets can be transmitted into the intra-mesh network with high rate), we can claim that dynamic allocation of subcarriers should target decrease of latencies in transmission side queues. Therefore, in this project, we seek subcarrier arbitration mechanisms which tries to minimize the transmission side delays and buffer capacities. The details of protocol specifications for on-chip packets, such as detailed definition of packet bits etc., are out of scope of this thesis work. A generic type of bandwidth allocation mechanism is envisaged, which can be utilized for any type of massive multicore architecture.

Based on changing parameters such as queue lengths of on different instants, the allocation algorithm may aim to minimize the average latency, a delay or buffer length

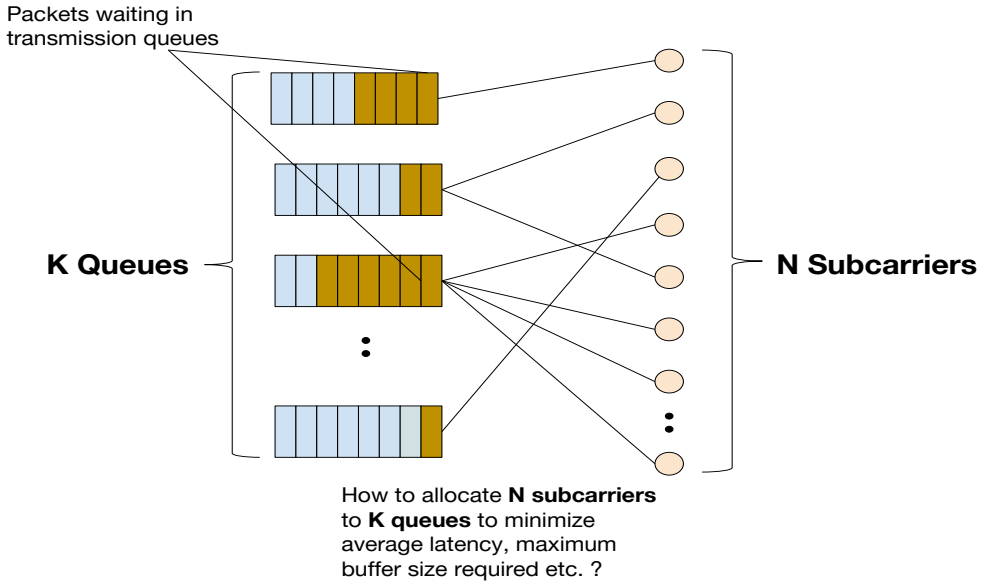


FIGURE 4.1: Bandwidth allocation problem in WiNoCoD can be formulated as arbitrating 1024 subcarriers to 32 transmission queues of tilesets on different symbols.

exceeding probability or a totally different cost function. Next, we revise a vast literature on the similar problems of bandwidth allocation to parallel queues which is encountered in very diverse fields such as space telecommunications, processor scheduling, optical communications, cellular communications etc.

4.1.2 Cognitive Radio

Cognitive radio [81] is all about making a radio device which is self-adaptive to the changes in its environment, in order to improve; or optimize if possible, its radio capabilities [82]. For this purpose, it requires 3 functionalities :

- Sensors : to capture changes in its radio environment, which can be the absence of transmission on certain part of spectrum (i.e. white space), detection of a certain type pre-coded signal (e.g. a beacon signal), detection of a change in energy levels or modulation of signals etc.
- Decision Making : The required intelligent and robust algorithms that perform these optimization processes based on environmental stimuli. This can be change of utilized spectrum, modulation type, transmission power etc.
- Adapt : Based on the taken decisions, cognitive radio equipments can change their RF characteristics. As a digitally implemented modulation, OFDM provides a great flexibility for this adaptation, such as changing utilized bandwidth etc..

These 3 main steps of cognitive radio are continously implemented, such that after adapting its RF properties, the intelligent node continues to sensing its environment as illustrated in Fig. 4.2.

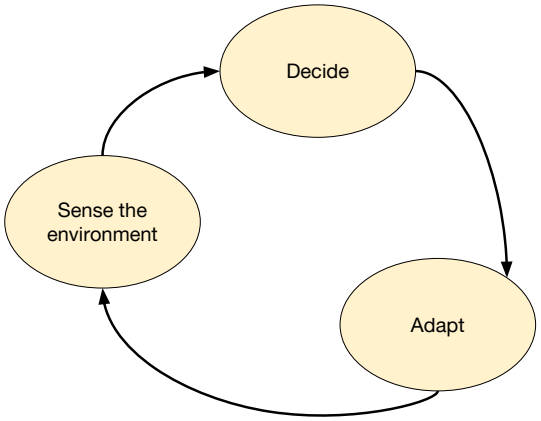


FIGURE 4.2: Cognitive radio cycle, illustrating the circular flow of 3 main steps [82].

In this work, from a certain perspective, we build an on-chip cognitive radio system, which takes into account the instantaneous transmission queue lengths of tileset RF transceivers (sensing by acquiring this information from other tilesets), run the algorithms proposed in explained in Chapter 5, 6 and 7 to choose the required bandwidth (number of subcarriers) and modulation order (decision step) and apply these to its OFDMA based transceiver (adaption step).

4.1.3 LTE

As OFDMA is an ubiquitous modulation for wired and wireless standards, one may profit by examining the existing solutions developed for these systems. Especially LTE 4G is the recent cellular standard which utilizes OFDMA for its medium access scheme [83]. Existing literature for partition of bandwidth in spatial and temporal dimension among mobile nodes, rate allocation mechanisms, coordination between the base station and mobile nodes, carrier aggregation etc. can form a basis for current and future conception of WiNoCoD and similar projects. Even though cellular systems have a much lower bandwidth and less strict timing requirements, certain ideas from this domain can be migrated to our problem. For instance, Fig. 4.3 shows the carrier aggregation concept, where non-contiguous bands of spectrum can be aggregated by a node to increase data rate. In a sense, this concept is also applied in WiNoCoD thanks to the OFDMA based RF infrastructure.

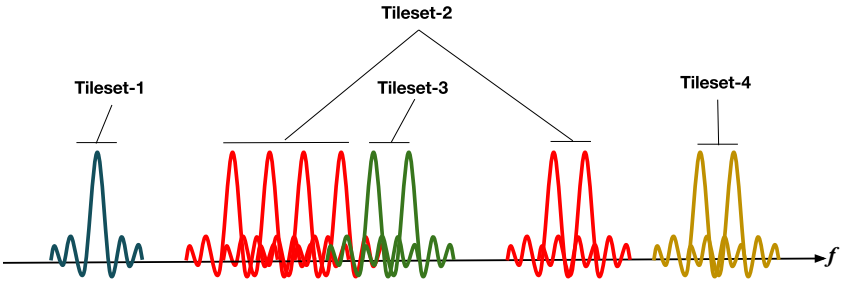


FIGURE 4.3: Carrier aggregation concept which is also used in LTE, where non-contiguous and different widths of bands can be aggregated to be used by a node.

4.2 Dynamic Bandwidth Scheduling for Parallel Queues

Aforementioned problem is referred as "Multi-user Multi-server" or "Parallel Queues" Scheduling in the literature. Countless number of varieties of this problem are encountered in different areas such as; multiprocessor scheduling (computational scheduling) [84], satellite communications [85], optical networks [86] etc.. When formulating the optimal scheduling policy, generally the average delay (or identically average queue length from Little's Law [87]) is chosen as the function to minimize [88], which is also the main metric of interest for on-chip networks.

Research on scheduling policies for multi-user multi-server case incorporates fields of Queuing Theory, Network Theory, Stochastic Optimization and Markov Decision Processes. Considering the nature of our OFDMA interconnect, we will investigate the most effective algorithms which are referred as dynamic scheduling policies. These type of policies reassign servers to the queues every single or multiple time slot(s), based on the instantaneous state of the system. Even though certain state metrics such as recently measured mean response time or mean queue lengths, or utilization of the servers can be used to determine the new server allocation, generally algorithms who are using instantaneous queue length state information (QSI) are best (queue aware policies) [88].

Even though this problem might appear as straight-forward, optimal solution generally requires complex stochastic operations and only valid for very limited circumstances with specific unrealistic assumptions. In this section, we present some of the approaches for dynamic bandwidth scheduling, which are most convenient for WiNoCoD case.

4.2.1 Longest Queue First

For the case where there is a single server, which has to be assigned to one of **K** queues every time slot (i.e. multi-user single-server case). It was proven that Longest Queue First (LQF) policy provides the lowest average latency, given that arrival processes to queues are independently and identically distributed (i.i.d.). This policy simply assigns the server to the queue who has most number of packets (i.e. longest queue). Although it is straight-forward, this policy has received a wide attention from research community for its stochastic analysis under certain circumstances and variables. [89] compares the buffer flow exceeding probabilities of LQF with any other non-queue aware policy, and also proves that it is also the optimal policy in terms of buffer exponents given that arrivals are i.i.d..

The optimality of this algorithm for multiple server case is still valid under i.i.d. arrivals assumption. [90] investigates the multi-queue multi-server problem in the context of wireless communications, where connections between users (queues) and certain frequency channels (servers) are not available for certain instances. This stochastic assumption for frequency selective channels in wireless communications is common. They prove that LQF among the connected servers is the optimal solution under i.i.d. arrivals assumption. Explaining in detail, the suggested algorithm (Longest Connected Queue First) iterates through all servers, and assigns each server to the longest queue among users who are connected to it on that time slot. It is trivial to see that this optimality is valid for the special case of full connectivity, where all queues are connected to all servers always. Note that, this is the case for WiNoCoD's OFDMA RF interconnect, where each tileset can use any of the 1024 subcarriers. However, the literature generally approached this queuing problem from the wireless communications point of view, where channel conditions are considered and a joint power/rate scheduling is performed. Their algorithms generally use complex maximization via utility functions [91][92]. Due to static nature of channel and transmission dynamics, our problem in WiNoCoD is purely a queuing optimization.

As stated previously, this algorithm is optimal in the sense of minimizing average latency for the special case of symmetric queues, where stochastic arrivals to queues are i.i.d. distributed. However, for most of the time this assumption is far to be accurate. Especially for the on-chip traffic, as we have revised in Section 2.4. In addition, this algorithm may cause a severe starvation, as certain nodes with high queue lengths may starve all the resources (servers), and packets in small length queues may not acquire any resource for a long period. Another drawback of this algorithm is its computational complexity. It iterates through each N servers, and each iterated server iterates all K queues. And after the assignment, the queue lengths are updated. Therefore it has a

complexity order of $\mathcal{O}(KN)$, which increases with the number of queues and servers. Execution of this algorithm may be feasible for certain scenarios such as cellular communications where bandwidth is reallocated every few hundred miliseconds, however time required for computation in an on-chip context should be limited to nanoseconds.

4.2.2 Queue Length Proportional Scheduling

Another solution to multi-queue multi-server allocation problem is to divide the total bandwidth proportional to instantaneous queue lengths. This approach is known as "Queue Proportional Scheduling"(QPS) or "Buffer Length Proportional Rate" in the literature [93]. Even though its delay optimality has not been proven, it provides a very good compromise between fairness and low average queuing delay. It was shown that this algorithm is capable of providing a proportional average delay differentiation under high traffic load [93]. This means; the average delay of two different queues can be guaranteed to have a desired arbitrary proportion as $\frac{d_i}{d_j} = \frac{c_i}{c_j}$, where c_i and c_j are two adjusted scalars, which are multiplied by instantaneous queue lengths, while proportionally dividing total bandwidth. This may be a strong tool for the scenarios where a Quality-of-Service (QoS) differentiation is required. In our case, all tileset queues have the same priority, i.e. $c_i = 1$ for all tilesets. Hence, it can be deduced that under high input traffic close to the system capacity, this scheduling would yield to an equal average latency at all tileset transmission queues.

4.2.3 Square Root of Queue Length Proportional Scheduling

Complexity of stochastic optimization without subtle assumptions, forces researchers to approach the multi-queue multi-server problem via different methods. Especially, without well defined arrival process distributions, queuing theory equations are hard to derive. For instance, [94] approached this problem from a different perspective. The authors intend to design a rate scheduler for classical ATM networks. The total service rate (can be visualized bandwidth) of R should be divided among K ATM nodes, i.e. queues as r_i. They formulate this problem as dividing the total R to K queues such that, it minimizes the time to drain all the backlog in K queues. In this case, they assume no further packets arrive to the queues, so the optimal arbitration clears out all the packets in the system as quick as possible. Writing the problem formally :

$$argmin \left\{ \frac{Q_1}{r_1} + ... + \frac{Q_K}{r_K} \right\} \; s.t., \sum_{i=1}^{K} r_i = R \tag{4.1}$$

They also study the case for proportional service differentiation as we have mentioned previously, where the total service acquired by two arbitrary nodes are proportional to a constant. However, we investigate the case for no service differentiation, where all queues have the same priority. After using dynamic programming (DP) techniques, they reach an elegant analytic solution, where the optimal rate allocation is proportional to square roots of the queues. The optimal rate should be allocated to two arbitrary nodes has a relation to their current queue lengths as follows :

$$\frac{r_i}{r_j} = \frac{\sqrt{Q_i}}{\sqrt{Q_j}} \tag{4.2}$$

However, recall that this assumption optimizes the queue draining times and assume no further exogenous packet arrivals. Thus, this approach may be unsuitable for certain cases.

4.2.4 Oldest Packet First Scheduling

Apart from scheduling algorithms using instantaneous queue length information of the queues, we would like to introduce another approach. This scheduling is omniscient in sense of knowing current delay of each packet in each queue, which is a non-realistic assumption for most of the scenarios. At every scheduling instance, the proposed algorithm allocates N resources to oldest N packets. If scheduling is performed every time slot, it is straight-forward that this algorithm yields to the minimum average and maximum delay. It is generally referred as "Oldest Packet First"(OPF) algorithm in the literature [95]. This algorithm not only needs to know each delay of each packet in the system, but also requires high number of iterations to choose the oldest head-of-line (HoL) packet in each queue for each of the resource to allocate. Even though this scheduling may seem unrealistic to apply in our case, we will use this algorithm as an optimal reference to compare our proposed schedulers.

4.3 Preliminaries of Bandwidth Allocation in WiNoCoD

4.3.1 Partitioning Bandwidth Statically

A basic instinctual approach for allocation of subcarriers for transmission would be to divide them equally among tilesets. ATAC [3], as mentioned in Chapter 2, is a thousand-core hybrid cache coherent CMP architecture comparable to WiNoCoD but utilizes an

optical interconnect. Similarly to WiNoCoD, for dimensionality purposes cores are clustered and there exists 16 clusters, each containing 64 identical cores which can access the optical waveguide. As explained in detail in Chapter 2, each cluster can transmit their information on 128-bit wide channels, which is composed of a different wavelengths (WDMA) on different separated parts of the waveguide. In reception part, all clusters are tuned to whole dedicated bandwidth, that they can receive all packets transmitted by others. This is similar to intrinsic broadcast capability of WiNoCoD's OFDMA interconnect, where main motivation behind this SWMR approach is to support large amount of broadcast cache-coherence packets of distributed hybrid memory. Recalling Chapter 2, this bandwidth intense cache coherence protocol and enabling SWMR interconnect, helps to isolate programmer from the hardware, where scalable, easy programmation can be done in thousand-core architectures. Due to static nature of these optical channels where they are generated by non-tunable circuitry, the researchers were obliged to allocate a bandwidth portion statically to tilesets at TX. They have chosen to equally partition the optical bandwidth to all 64 tilesets as in Fig. 4.4(a).

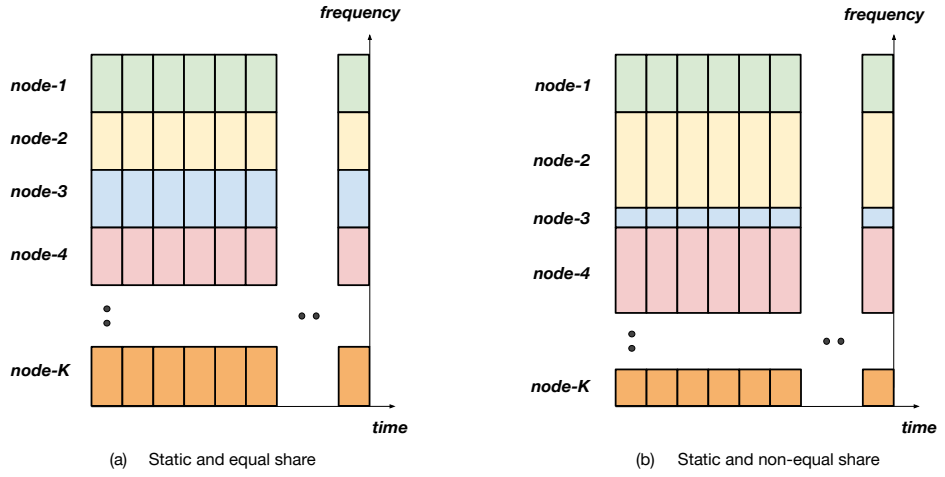


FIGURE 4.4: Static and equal or non-equal allocation of frequency resources among multiple nodes.

The fundamental assumption behind this choice is that with non-uniform memory architecture, with broadcast intense hybrid distributed cache coherence protocol will lead to a uniform demand from tilesets on average. First of all, this assumption depends on the chosen application, where we know they may be so diverse in terms of spatial locality [8]. A more significant point is the instantaneous changing demands, that we know cores produce highly different amount of cache coherence packets both in small intervals of few cycles or much longer application phases [96]. Hence, designers of this optical interconnect were obliged to dimension their photonic bandwidth concerning the maximum load that a tileset generate. It is considerable to think this mechanism not

as scalable and bandwidth efficient, even when the average load of sharing nodes are equal, as generated number of packets by a tileset is a sporadic phenomenon and varies significantly over time.

The projection of this approach on WiNoCoD, would be to allocate equal number of subcarriers on each slot for each tileset. Considering we have 1024 subcarriers and 32 subcarriers, we could allocate 32 subcarriers per tileset per symbol for transmission. In case QPSK used, this corresponds to a configuration where each tileset can transmit 64 bits per 50 ns, thus a 1.25 Gbps rate, constantly and equally allocated for each tileset. Of course, this choice would require no reconfiguration mechanism or overhead but would be far away to exploit the highly efficient and rapid reconfigurability of OFDMA. Arbitrating bandwidth to nodes based on their instantaneous demands is a strong tool to achieve much lower delays and buffer sizes, but conventional optical or RF interconnects are limited by their static circuitry, where OFDMA alter this situation.

4.3.2 A Quasi-Static and Quasi-Dynamic Modification

Another perspective to partition the bandwidth is to be able to distribute it to nodes per used application. Certain applications may demand much less computational power and memory. Taking into account our 2048 cores and 1 TByte RAM, certain applications can be executed using much smaller parts of CMP. For instance, in case 4 out of 32 tilesets are guaranteed to be used by the application, whole RF bandwidth could be distributed for transmission of 4 tilesets only through the execution, which effectively increases available bandwidth by 8. In ATAC, where transmission channels are generated by non-tunable optical rings, we can claim even an application based quasi-static bandwidth arbitration cannot be effectuated.

In addition, even all applications are used, the total bandwidth demand of each tileset would not be equal. This is mainly due to fractal locality of computation and memory access patterns, which has received a wide attention from the research community. In [97], it was shown that this locality of on-chip traffic (that becomes much more apparent as core number reaches thousands) can be formulated accurately by *Rent's Law* which was actually developed to model the mathematical relation between number of output terminals and number of components in an integrated circuit. This theory, basically states that the traffic intensity between nearer cores is exponentially higher compared to farther cores. Also in [98] the communication probability of a core to an another core was tried to modeled by a negative exponential distribution. [99] attempts to characterize this locality especially for a 3D NoC interconnect. Authors in [100] and [101] proposes NoC architectures to exploit this locality. Finally, [8] analyzes various

number of diverse application run on a CMP and a mesh NoC, where they find out that total throughput generated by a core through whole execution time is distributed based on a Gaussian distribution. Briefly, this means for certain applications, even whole CMP cores and memory would be utilized, certain tilesets may use exponentially higher bandwidth compared to other ones.

Considering much more complex schemes, where multiple different characteristic applications are embedded on CMP, the bandwidth demand difference of tilesets shall be much more diverse. In a nutshell, if the demands of tilesets for certain applications and configurations through the execution time can be foreseen at compilation time, different number of subcarriers can be allocated to appropriate tilesets easily, thanks to basic digital reconfigurability of our OFDMA based interconnect as in Fig. 4.4(b). Even this configuration scheme does not exploit the temporal changes of traffic, this quasi-static modification with no overhead and effort at run-time, shall provide much higher performance compared to a fully static architecture like ATAC, where bandwidth allocated to tilesets are always same and equal. Undoubtedly, using rapid, trivial and efficient bandwidth arbitration mechanisms, our OFDMA interconnect can provide much more performance using temporal locality as an additional dimension.

Another possible option OFDMA provides us the offline optimization of the bandwidth allocation for execution of certain applications, especially non-real time ones. Actually, we can draw a straight analogy between subcarrier arbitration for OFDMA on-chip interconnect and multiple CPU resource allocation for applications. They both seek the optimal allocation of a resource (first one being data rate, and the latter one being computational resource) to reduce latencies. Especially, there exists a vast literature on the allocation of multiple cores to workload in thread or instruction level [102].

Like for the offline multi-core job scheduling analysis [103], optimal number of subcarriers and chosen modulation order for each tileset on each symbol can be determined. This shall boost the communication performance of the architecture, decreasing latencies to minimum. The previously presented Oldest Packet First (OPF) algorithm may be a candidate for this kind of optimization. However, in this thesis work we deal with the on-time dynamic allocation of resources, which is a more generic approach for all kind of applications.

4.3.3 Resource Blocks, Frames and QSI Signaling

Peculiar limitations of WiNoCoD's OFDMA RF interconnect require to develop new techniques to cope with. As mentioned previously, the relatively long symbol duration

with respect to inter-packet arrival times, short packet lengths, and extremely strict delay requirements make this environment really unique compared to any existing OFDMA based communication medium. Therefore, in this section we introduce certain preliminary notions rooted from these limitations, for bandwidth allocation in WiNoCoD. This context shape a new allocation problem concept, where we have to derive new solutions, as extensions of the ones presented here.

4.3.3.1 Resource Blocks

First notion we present is the *Resource Block* (RB), which defines a group of adjacent subcarriers on a single symbol. This term is borrowed from LTE jargon, where it is used for a group of adjacent subcarriers spanning multiple symbols in time [104]. Considering we have 1024 subcarriers, it is obvious that a very fine granularity for bandwidth allocation with one or few subcarriers is both computationally challenging and unnecessary. As symbol duration is relatively long (consider the case cores are operating at 1 GHz frequency, resulting in a symbol duration more than 50 cycles which is a remarkable time interval in terms of processing), we can state that a bandwidth reconfiguration which is performed every symbol would be an essential requirement. We choose to define a RB to serve exactly 1 short packet (1 flit - *64 bits*). This way, a short packet (which constitutes most of the packets circulating on RF interconnect as mentioned in Section 2.1) can be served in one symbol, regarding the unique long duration of OFDM symbols. Consider the case we define a RB of 16 subcarriers, where a flit is 64 bits. Assuming QPSK is used, 1 RB corresponds to 32 bits. Taking the example for a tileset which is allocated 5 RBs (80 bits) and it has 1 flit short packet of 64 bits in its queue. Therefore, it would utilize only 64 subcarriers and left idle the remaining 16 subcarriers, which is a waste of resources. For these reasons above, a RB is defined as 64 bits (integer multiple of a packet) of service in 1 symbol (32 subcarriers in case of QPSK, and 64 subcarriers in case of BPSK). Based on the progress of the project, flit size-packet fragmentation shall be regulated, however to test our algorithms, QPSK has been chosen as the default utilized modulation order and 64-bit flits, thus and a RB of 32 subcarriers is assumed, allowing each tileset to have a chance to send 1 flit per symbol. Symbol-wise on-chip packet scheduling with RBs is shown in Fig. 4.5.

4.3.3.2 Frames

After we partition our 20 GHz spectrum as RBs (in case of QPSK, 32 RBs per symbol), now we have to define the *structure of allocation*. As we have reviewed in Section 4.2, effective rate allocation algorithms require to utilize the recent queue length information

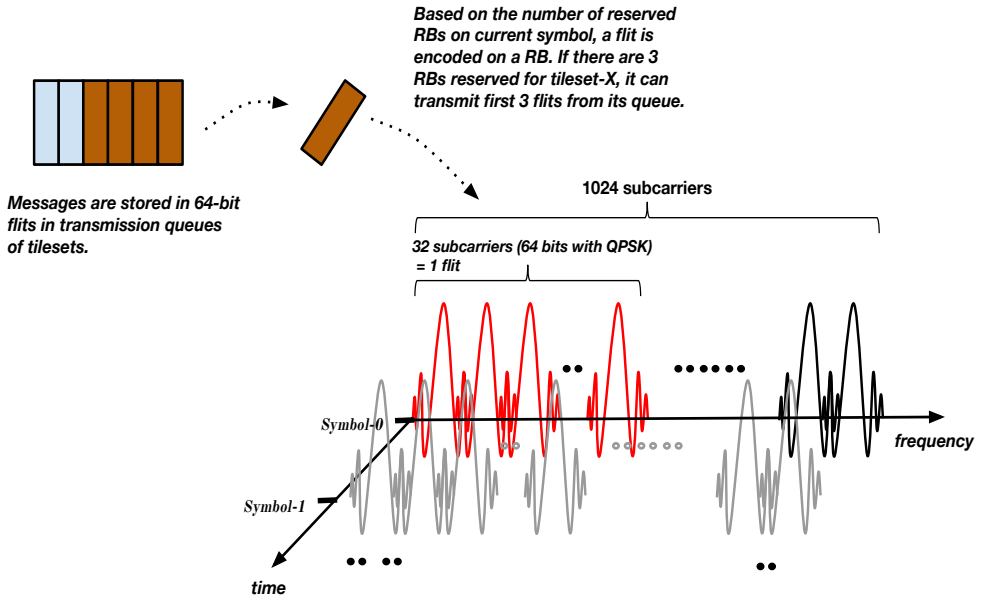


FIGURE 4.5: A Resource Block allows for the transmission of 1 flit during a 50 ns symbol in WiNoCoD

(QSI) of queues, in order to attain low delays and buffer overflow probability. However, taking into account the limited bandwidth per symbol, exchange of QSIs of each tileset every symbol seems not realistic due to overwhelming signaling. This cost of QSI signaling overhead will be examined in Section 4.3.3.6. In addition, most important requirement to perform re-arbitration of subcarriers is to give the microprocessors or RF controller modules in tilesets (or central intelligent unit) the time for computation. In conventional OFDMA networks, for instance LTE, computation time for allocation algorithms is not a bottleneck. 1 symbol takes around 66 microseconds in LTE, and due to reasons like channel coherence duration and propagation time (RBs are re-allocated every 13-14 symbols), computation for allocation is assumed to be performed instantly between symbols. However, in our specific case in WiNoCoD, where 1 symbol is 50 ns, the computation time appears as a significant actor, as the temporal limits of semiconductor technology is reached at the scale of nanoseconds. This strict and unorthodox temporal requirement makes the allocation problem which is examined in this thesis an original work compared to conventional networks. Not just the computation time, but all the cumulative delay incurred by subcarrier reconfiguration time, packet processing, synchronization, propagation time etc. can be treated in one frame. A generic microprocessor can be employed for bandwidth allocation, whereas a dedicated digital circuitry may be implemented in order to decrease computation time further. Also, we can expect the scale of 20 GHz OFDM bandwidth to hundreds of GHz in several decades, shortening symbol duration more and more, with rapidly developing semi-conductor technology. In

case, economic, low-scale generic processors will be continued to be employed for bandwidth reconfiguration, this bottleneck of computation time shall continue to tighten. As mentioned previously, IFFT/FFT computation with effective state-of-the-art modules and synchronization latency between tilesets due to propagation time are assumed to be zero (or included in symbol time), however in case these aspects are decided to be treated, this *computation time* can be assumed to include them, where queue length information becomes outdated. Hence, we envision a system, where QSIs (or other indicators of local traffic taking into account other possible algorithms) are broadcasted by tilesets every T *symbols* and the new allocation of subcarriers are activated after T symbols. Thanks to broadcast capable OFDMA interconnect, the QSI information sent by each tileset can be received by others. We call this T symbols as a *frame*. Thus, at the end, we have a pipelined allocation system, where allocation at the start of a frame is calculated according to QSI information sent T symbols before. Of course, we need certain number of subcarriers to be reserved on first symbol of a frame, in order to allow tilesets to encode their QSI. Fig. 4.6 illustrates the structures of frames and RBs in WiNoCoD. This time-frequency matrix composed by atomic temporal units of *symbols* and spectral units of *RBs* constitutes the backbone of the bandwidth allocation mechanism in WiNoCoD. We provision that tilesets have a certain digital circuitry and microprocessing system that implements this time-frequency (RB-symbol) matrix, and they modify it at the start of every frame, where each RB corresponds to a tileset ID. At every symbol, they use this virtual matrix to encode their flits before IFFT phase, on the subcarriers reserved to them (if there is any on that symbol). And for the receiver side, they do the necessary packet treatment and routing on received flits on each RB, with the ID associated with it. In other words, tilesets implement this imaginary matrix notion in their RF controllers, therefore they know on which symbol and on which RB, they shall transmit their flits and receive flits of other tilesets associated with their IDs. Note that, in case a central allocation paradigm is adopted, where a central intelligent unit (CIU) gathers local QSIs, deciding on the new configuration of this matrix must signal other tilesets about the bandwidth partition with a response message. Hence, additional subcarriers should be reserved at the end of the frame.

It is important to note that, the temporal delay mentioned here for allocation procedure, does not only includes the computation for arbitration but also includes all delays incurred by wave propagation, IFFT/FFT, serialization/parallelization, conversion and synchronization, both in transmitter and receiver sides. Therefore, in extremely rapid changing bandwidth demands of this specific configuration of WiNoCoD, with bimodal packets and different number of packets created every cycle by different tilesets, we have chosen to build this pipelined framework, where outdated QSI cannot be omitted as in conventional networks. Recall that one symbol duration is approximately 50 ns, which

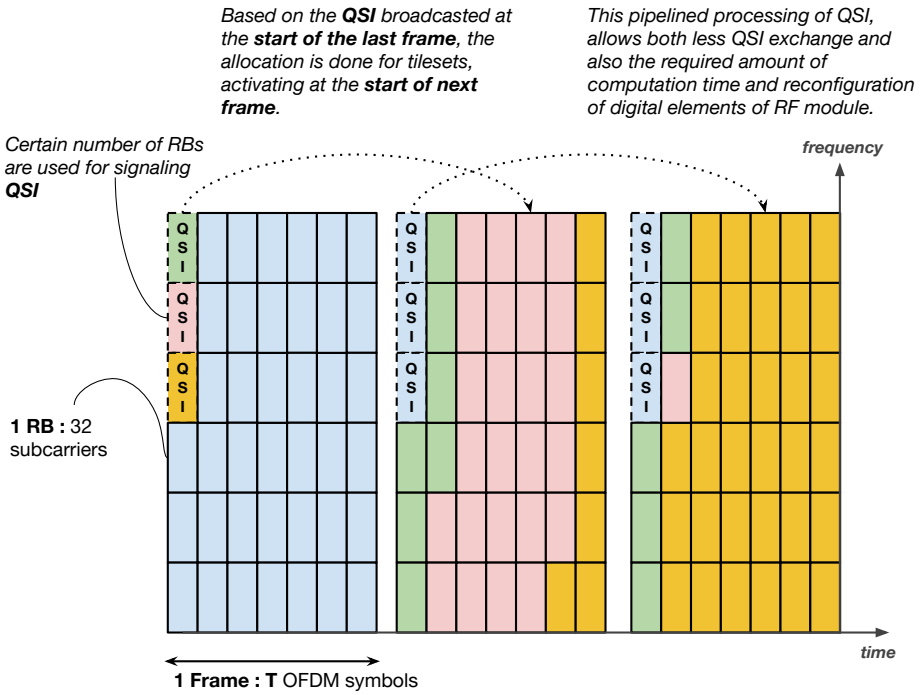


FIGURE 4.6: Frames and Resource Blocks (RBs) in WiNoCoD

corresponds to 50 clock cycles for processors running on 1 GHz. Rather than strictly defining these delays explicitly and individually, we have chosen to set a cumulative delay in terms of OFDM symbols, firstly because project is still in progress where these values are not certain yet, and also to offer a generic comprehension of the allocation algorithms, that their performances are evaluated under different frame lengths. Moreover, one can also determine different computation times based on chosen allocation algorithms or used microprocessor(s) with different speeds or ASIC implementations with different trade-offs between execution speed and power consumption, surface or budget. However, chosen frame lengths in experimental evaluation are all feasible and validated by the primal designs of the architecture.

4.3.3.3 Decentralized and Centralized Allocation

The subcarrier arbitration mechanism we have developed for WiNoCoD interconnect, can be classified into two cases, such as decentralized and centralized approaches. Both approaches have pros and cons. If a centralized allocation is preferred, a single central intelligent unit (CIU), which either can be a digital circuitry or simple microprocessor(s) installed inside a tileset's RF front-end (Before IFFT/FFT module). This way, it can use the OFDM modulator/demodulator of that tileset. It shall acquire the QSIs of each tileset, process the necessary allocation algorithm, determine the arbitration of RBs for

next frame and send to each tileset its allocation scheme. Thanks to this full broadcast capable structure, we do not need any beacon signal or similar channels like for most of the wireless cognitive radio or ad-hoc networks. To lower the transmission power requirement and error probability, CIU shall be installed in a tileset which is physically close to the center of transmission line. This is an obvious extra signaling overhead, which is not desirable for our limited on-chip bandwidth. Another disadvantage is the robustness. In case, a circuit failure happens, the bandwidth allocation infra-structure of the system collapses. In addition, with request-response nature of allocation, the response time of the system is increased. The main aim in central allocation is to minimize the intelligence at tileset front-ends, thus lowering the area or cost budget. The exact signaling of which tileset will use which RB is impossible due to extensive overhead, thus CIU shall broadcast only the *number of RBs allocated for each tileset*, and a simple circuitry in each tileset should map these to its *allocation matrix*, by sequentially placing them starting from the first RB. For instance, indicating a tileset to utilize RBs R_1, R_2.., on symbol t_1 and RBs R_3, R_4.. on symbol t_2 .. is overwhelmingly bandwidth consuming, which is not feasible in our case. Hence, central unit shall only indicate the number of RBs allocated for a tileset in next frame.

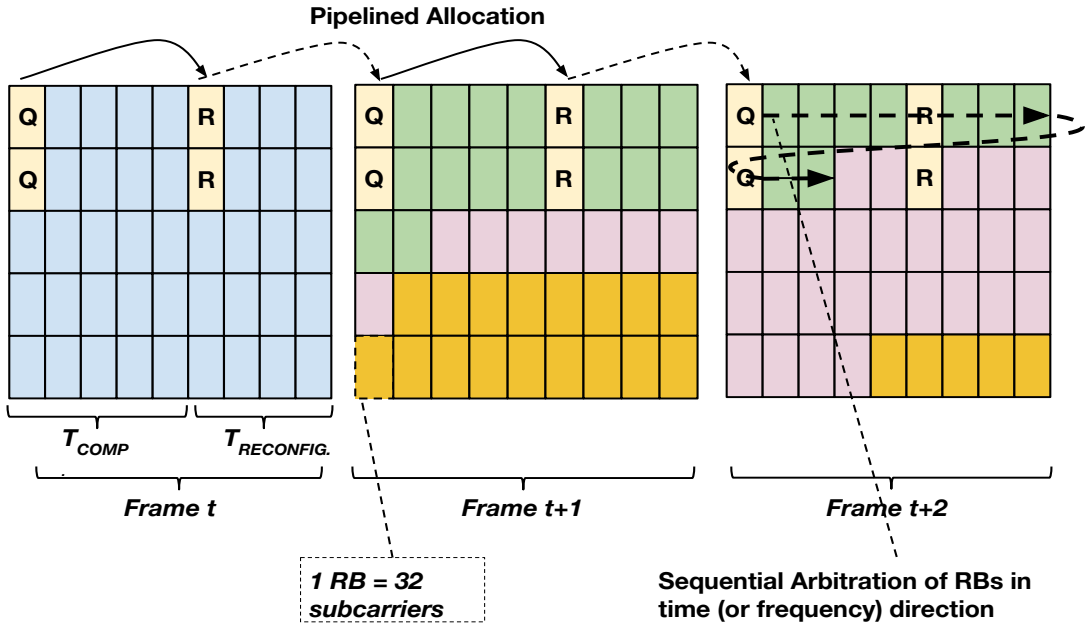


FIGURE 4.7: Allocation of RBs through a frame, in central bandwidth allocation

Also, note that we have positioned the response of CIU on reserved subcarriers on few symbols before the end of the frame as in Fig. 4.7. We spare $T_{reconfiguration}$ symbols of time for tilesets' processors to receive the number of RBs allocated for each tileset in next frame and compute the RB allocation pattern and reconfigure their transmissions.

4.3.3.4 Direction of Resource Block Allocation in a Frame

After marking the headlines of the frame structure and the pipelined fashion of RB allocation using recent QSI information, an important question arises : in which direction these RBs should be allocated? In other words, when we start to allocate RBs from the first RB in the frame, which one should be the next? This is a significant and thought-worthy aspect, both considering the network layer due to variation of delays and physical layer due to the fact that transmission power changes based on using adjacent subcarriers or not. We propose 2 different approaches as illustrated in Fig. 4.8 : Allocating RBs in *frequency direction* and *time direction*. We may claim that allocating RBs in frequency direction starting from the requirements of a tileset would allow it to clear out its queue in shortest duration as possible, however this would be unfair for the rest of the tilesets. Especially, in case of long frame lengths, allocating all RBs for a tileset on just first few symbols is disadvantageous, as this tileset would not have any RBs left in the rest of the frame taking into account new packets shall arrive during the current frame (Consider a relatively long frame, that new packets shall continue to arrive each symbol during the frame). These newly arriving packets would require bandwidth also, and may disrupt the latency performance dramatically. This situation may dramatically increase the variance of packet service duration, which increases the overall latency as we know from queuing theory. If we allocate RBs in explained frequency direction, a group of adjacent RBs may be concentrated along one or few symbols. In contrast, allocating RBs in time direction would increase the chance of a tileset to have at least one RB on each symbol of a frame). Thus, allocating RBs in time direction appears as another solution, which seems to allocate RBs more uniform in temporal manner. Therefore, as the performance of two proposed approaches cannot be estimated analytically, we both test them under various scenarios and configurations.

4.3.3.5 Idle Resource Blocks-Default Frame Configuration

In particular instants, sum of RB demands of tilesets (i.e. total QSI) may be much lower than the total number of RBs in a frame. For these cases, there may exist RBs left idle. Considering also the potential new packets which arrives during these *idle symbols*, where all or certain RBs are not used by any tileset-, sharing them equally among tilesets seems straightforward. In fact digital nature of OFDMA allocation makes this process quite trivial. We envision a virtual *default allocation matrix* where RBs are arbitrated in an FDMA fashion which allows for a RB to be utilized during each symbol for each tileset. In other words, in a default allocation matrix, each 32 tilesets have 1 RB each. Especially in case of time direction allocation, if we would keep the same FDMA configuration, this would impose an unfairness on the tilesets which utilize the RBs on the upper side of the

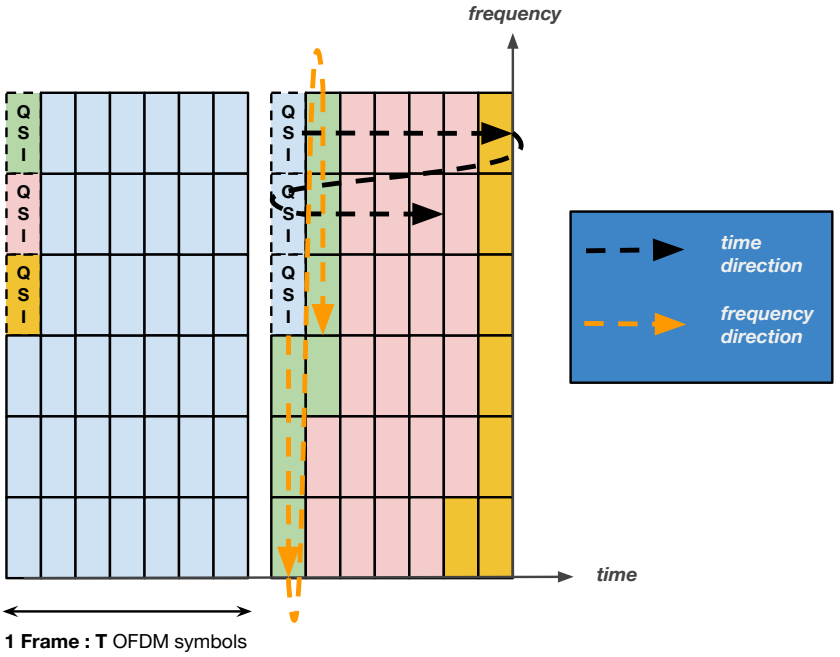


FIGURE 4.8: Allocation of RBs through different directions inside a frame

frame (where RB allocation starts). Therefore, we decided to employ a basic rotating mechanism for this default configuration as in Fig. 4.9.

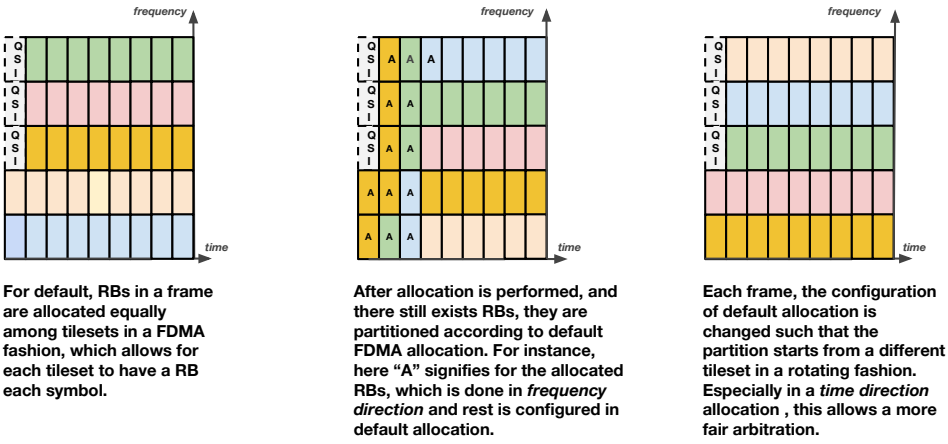


FIGURE 4.9: Default allocation matrix which is evenly distributed among tilesets in an FDMA fashion, is modified by the scheduler based on the reported QSIs. The rest of the RBs which are left idle, is partitioned according to this uniform distribution

4.3.3.6 QSI Encoding and Signaling

An important aspect is the encoding of QSIs of tilesets at the start of a frame. In case of QPSK constellation, we have 2048 bits available on a symbol. Based on our extensive

simulation campaign, 8 bits per tileset per frame to encode QSI has shown the best overall performance. With 8 bits, it is possible to encode 256 different levels. Of course, the empty queue (0 QSI) should also be indicated. By looking at simulation results, we have seen that the number of flits in a transmission queue very rarely exceeds 255 under evaluated scenarios. Therefore any QSI between 0-255 is directly encoded, and if the number of flits is larger than 255 it is encoded as 255. Of course, one can choose different approaches and granularity of QSI encoding, such as representing multiple flits by a single QSI level. There is an evident trade-off between QSI signaling overhead and accuracy of the allocation algorithm. However, from these many possibilities, by taking account the realistic interconnect configurations, proposed 8-bit encoding seems viable and effective.

On first symbol of each frame, tilesets have to signal their instantaneous QSIs after encoding the number of flits. Thanks to intrinsic broadcast capability of OFDMA, each other tileset or -*CIU in centralized case*-, can receive all of the QSIs. For this purpose, certain number of RBs on first subcarriers, on first symbol is reserved, as default. For instance, consider the example, 1 RB is composed of 32 subcarriers (QPSK-64 bits) and QSI for each tileset is encoded with 6 bits. Then, we need 32x6 = 192 bits (subcarriers), which corresponds to 3 RBs in this specific case. One can see that, total number of subcarriers for QSI encoding of 32 tilesets, should be an integer multiple of RBs, in order to avoid waste of resources and invalidation of the RB based allocation mechanism. In case, the QSI encoding would be done with 7 bits, with default QPSK mechanism, we would need to allocate either 3 or 4 RBs, which corresponds to 6 or 8 bits of encoding per tileset.

Another important aspect of QSI encoding and signaling is the robustness counter transmission errors. QSI sent by a tileset may be received erroneously by different tilesets, resulting in a discoordination of bandwidth allocation, causing collision at the end. We assume, these kind of errors are out of scope of this thesis, however one can take precautions to decrease probability of errors on QSI channels further, by employing BPSK, the lowest modulation order or a mechanism to allocate more power on these signaling subcarriers.

4.3.3.7 Taking Into Account the Outdated QSI

Very specific constraints of on-chip OFDMA interconnect enforces certain requirements. Considering the relatively long symbol duration compared to packet delay requirements, even a latency of few number of symbols is important. In this special context, just in few symbols, whole queue dynamics may change drastically. As explained above,

due to constraint of computation time and signaling overhead, the allocation is done in a pipelined manner through frames. In other words, the allocation is done with the T symbols outdated QSI. Hence, when the new reconfiguration of the bandwidth is activated, the QSI values may have changed significantly. For instance, consider the example where a tileset's demand is 8 RBs, and it has been already allocated 12 RBs during current frame and no new arrivals has occured. As the allocation is done for QSI of 8, it would be allocated unnecessary amount of resources, while it has no packets in its queue at the start of the new frame. Considering the highly temporally heterogeneous on-chip cache coherency traffic, we introduce the notion of *Definitive QSI* (DQSI). At the beginning of each frame, before encoding its QSI on the subcarriers, each tileset already knows the allocated number of RBs in the current frame. Therefore, rather than encoding QSI directly, it subtracts newly allocated number of RBs (S_i^t) from its QSI (Q_i^t). Of course, we eliminate the negative QSI, equating it to 0 for the negative values. :

$$Q_i^t = max(0, Q_i^t - S_i^t) \tag{4.3}$$

This operation is fairly simple, negligible in computation time T and processing power, but has a great potential to avoid unnecessary bandwidth allocation. Especially, when the QSIs are small, this value will tend to be zero, approaching to a equal allocation of RBs. In literature, various approaches like this to eliminate unnecessary bandwidth allocation due to outdated QSI can be found in certain protocols [105][106]. Once the *minimum number of flits* in the queue is determined, we may also take into account the number of flits that will arrive during allocation process (i.e. current frame). To do so, the tilesets has to estimate the number of flits that will arrive during a frame. Considering the self-similar temporal nature of on-chip traffic, these estimations shall be accurate enough. We refer this as the *Expected QSI*(EQSI), which is calculated as :

$$\hat{Q}_i^t = max(0, Q_i^{t-1} - S_i^{t-1}) + A_t \tag{4.4}$$

where A_t is the estimated number of flits that will arrive during a frame. For estimation an *Exponentially Weighted Moving Average* (EWMA) filter can be used, which calculates the moving average as follows :

$$A_t = \alpha A_{t-1} + (1 - \alpha) a_{t-1} \tag{4.5}$$

α is a scalar, which weighs the importance of recent observation and a_{t-1} is the number of flits arrived in last frame. The optimal value of α obviously depends on the traffic. Via extensive simulations under various scenarios, it shall be determined. Another approach is to take the number of arrivals in last frame as the estimation, which is the extreme case of EWMA with $\alpha = 0$. Note that, α depends on the nature of the traffic and may be determined online or offline intelligent algorithms, however this is out of scope of this work. Based on our simulations, we have determined and tested an α value of 0.95 in this thesis. The resulting bandwidth demand is labeled as *Expected QSI*(EQSI). Even though it is quite different, [107] also takes into account the average arrival rates for allocating bandwidth in epochs.

For decentralized and centralized approaches, calculation mechanisms for DQSI and EQSI differs. Firstly for the decentralized case, each tileset computes its own DQSI or EQSI, and signals this value for the allocation process to other tilesets. This eliminates the redundancy and greatly reduces the computational costs. Therefore, the QSI is encoded just after the calculation of the number of RBs in next frame and few cycles before the end of the current frame to allow the computation of DQSI or EQSI. Each tileset governs an EWMA for estimating its own EQSI. At each frame, tileset front-end is responsible of counting the number of flits that has arrived in last frame and computation of the moving average.

In contrary, for centralized approach, in order to minimize the computational burden at tileset front-ends, CIU is responsible for DQSI and EQSI calculation. Tilesets simply signal their QSIs and CIU uses this information to calculate DQSI or EQSI for each tileset before allocating RBs based on these values. For this purpose, CIU keeps the number of allocated RBs for each tileset in last frame. CIU governs 32 EWMA registers for each of the tilesets. As CIU cannot know the exact number of flits that has arrived in a frame to each tileset, it estimates this value for each tileset by using the current QSI value, last QSI value and allocated RBs in last frame :

$$A^i_{t-1} = Q^i_{t-1} - Q^i_t + S^i_{t-1} \tag{4.6}$$

The computational cost of EQSI or DQSI for CIU is roughly more than 32 times for the tilesets in decentralized mode. Note that these operations are parallel in nature, so they can be exploited by multiple processors employed in CIU.

4.3.4 Traffic Models and Evaluation Methods

Before we present our algorithms, it is essential to introduce the techniques and scenarios we will utilize to evaluate them. We use OMNeT++ [108], a discrete event simulator, to mimic the dynamics of 32 tileset transmission side queues, which is utilized widely among Network-on-Chip community. Simulations are run in a symbol-accurate manner, that the atomic unit of time is one OFDM symbol. This means that we approach the problem from the grounds of improving the latency performance of packets waiting to be served at RF interconnect. Therefore, any traffic/congestion issues and latencies incurred in intra-tileset network (2D electrical mesh, tile crossbars etc.) are omitted. Also, content of the packets and their priorities are ignored, where they are treated as subjects demanding bandwidth with certain quality of service. This approach is adopted in this thesis work, because isolating RF interconnect and optimizing its performance provides the possibility to employ a more generic massive CMP attached to it, opening a whole new avenue for chip designers to consider OFDMA based RF interconnect as a candidate for their specific designs. However, note that, a more holistic approach and possible extensions to this work can be developed, taking also intra-tileset traffic into account.

4.3.4.1 Metrics of Interest

First metric of interest, that we try to improve is the *average latency*. Through a deep literature review, the *average latency* incurred by Network-on-Chip under different traffic models and loads, appears as the most important metric of interest [109][110][111][112][101]. Delay between the time a cache-coherency packet is generated at a core/memory unit till the time it reaches its destination point through the Network-on-Chip is defined as its *latency*, where it includes all delays caused by waiting time at buffers of routers, switching, packet processing, fragmentation and propagation time. The main motivation behind aiming to minimize average latency as a priority rather than setting strict requirements for individual packet delays, is that this metric effects application total execution time directly. However, for various real time applications with strict QoS requirements, jitter of individual packet delays may be significant [113] also. Most of the time, efficiency of the NoC is tested by investigating the curve showing the increasing average latency under increasing average load (i.e. known as injection rate in literature -

average number of packets generated by processing elements at an instance). Generally, due to basics of network theory, the average latency starts to *blow up* after a certain threshold of injection rate, where the system starts to be insufficient compared to generated load, where packets build-up at a faster rate than their serving at queues, hence average latency increases exponentially at a massive rate. Generally most of the papers in literature seek to increase this system blow up limit as much as possible, by their new NoC designs. Therefore, as stated previously, in this thesis, we will investigate these kind of curves, and try to increase the system limit and also decrease the average latency under this limit, as much as possible with our novel bandwidth allocation algorithms.

We have mentioned the importance of individual packet delays in a NoC, especially for run time applications. Therefore, our RF interconnect should also be tested for this metric. For this purpose, for each simulation we evaluate the *packet delay exceeding probability* curve, which shows the probability of any packet generated throughout the simulation to exceed the given delay bound, D_0, : $P(D > D_0)$.

And third metric of interest, is the *buffer length exceeding probability* curve, which gives the probability for a tileset's transmission queue to exceed a certain value in terms of flits L_0, at any symbol throughout the simulation; $P(L > L_0)$. One can see that, this metric is significant for dimensioning buffer lengths of our interconnect.

We compare the latency performance of our algorithms to a pseudo-optimal scheduler called "Oldest Packet First"(OPF) which is explained previously in Section 4.2.4. At the start of each frame, this optimal algorithm allocates RBs one-by-one by choosing the tileset queue with the highest head-of-line (HOL) delay. As it name suggests, it always serves the oldest packet in the system first. If multiple oldest packets exists, a rotating priority mechanism is applied. In this approach, aforementioned pipelined mechanism is not used as this is a hypothetical reference algorithm. Only at each frame time a near optimal arbitration of RBs is calculated.

4.3.4.2 Employed Traffic Models

Currently utilized application benchmarks to test NoC performance is far to represent the load generated future embedded algorithms in 1000 core architectures [3][47][114]. There exists an intense attempt by the NoC research community to characterize the on-chip traffic and generate reliable synthetic traffic to span a wide spectrum of diverse applications to be used by future CMPs, as mentioned in Section 2.4. Therefore, to evaluate the validity of our OFDMA interconnect and proposed arbitration algorithms, we have chosen to use synthetic stochastic traffic models.

To test our algorithms first we start with homogeneous traffic, where we model the number of packets each tileset transmission queue generates every symbol as Poisson process. The most utilized traffic model for synthetic packet generation in NoC literature is the Bernoulli process where at any cycle, a tile or processor generates a packet with probability p. As the number of packets that a tileset generates in a symbol is the sum of generated packets by multiple tiles in multiple cycles, it can be modeled as Poisson distribution, because superposition of multiple independent Bernoulli Random Variables yields to a Poisson process.

Hence, firstly, we test our algorithms with *uniformly distributed Poisson Process* case, where total injection rate, *i.e. average number of total packets generated by each tileset on each symbol*, is equally divided among all tilesets (also denoted as "Uniform Poisson" throughout the thesis). Therefore, the average load of tilesets throughout the simulation is equal, but not the generated packets by them on each symbol as it is a Poisson Process. Of course, this model is far away to provide the necessary benchmark for performance evaluation of our algorithms, due to temporal and spatial heterogeneity of on-chip traffic, which is explained in detail in Section 2.4.

In order to stress our interconnect and proposed algorithms, as a second stochastic traffic model, we use the *non-uniformly distributed Poisson Process*, where we still use the Poisson process to generate packets by each tileset on each symbol, but total injection rate is distributed non-uniformly. In order to push the limits of performance evaluation, without loss of generality, we set the injection rates of each first 8 tilesets to 1/120 of total rate; each second 8 tilesets to 2/120 of total rate; each third 8 tilesets to 4/120 of total rate and each last 8 tilesets to 8/120 of total rate. Hence, we make sure certain tilesets generate geometrically larger amount of load on average compared to certain others. One can also loosely relate our chosen ratios to the observations of [8], performed with various applications, that the spatial distribution of total load to processing elements in a CMP follows a Gaussian distribution, with different standard variations depending on the application and architecture.

In Section 2.4, we have seen that nature of on-chip cache-coherency induces a highly spatial and temporal heterogeneous traffic, that average load generated by certain processing elements shall be significantly different than certain others in CMP. In addition to these, there exists the temporal *self-similar* nature of traffic, which means a tileset generates packets in a *bursty* sense. Of course, the degree of heterogeneity for both temporal and spatial dimensions depends on the type of application, cache sizes, used coherency protocols etc. However, this bursty self-similar temporal nature is a well observed phenomenon for all types of NoCs and CMPs. In Section 2.4, it was mentioned that a *Hurst Parameter* is used to measure the degree of temporal burstiness. Now, to

form a third and more realistic traffic model, we seek methods to generate traffic with desired Hurst parameter, H. Generation of traffic with accurate H is not trivial and there exists a vast literature for it. [8] uses Discrete Fast Fourier Transform (DTFT) to generate packets stochastically for a CMP with NoC with given H as a parameter. Similarly to this work, authors of [115] also propose to generate stochastic traffic for CMPs with given H, but by aggregating heavy tailed ON-OFF Pareto Processes. They also propose to employ a recursive mechanism to further tune the traffic's accuracy. This phenomenon, that the aggregation of multiple heavy tailed ON-OFF processes will yield to a approximate self-similar traffic with desired H, is well investigated by researchers [116][117][118]. From a different perspective, in [119], authors use linear approximation to form fractal Gaussian noise for this purpose. We use as the most feasible one for our case, the method used in [56], *Discrete Pareto Burst Poisson Process (DPPBP)*, where authors utilized to dimension buffer sizes in a NoC. In this method, each node generates certain number of flows according to a Poisson Process. Then, the length of each generated flow is determined stochastically according to Pareto distribution, which is a discrete number of slots (in our case a symbol). Each flow generates packets at a constant rate (1 packet/symbol in our case) and unlimited number of flows co-exist at any time. Parameters of Poisson and Pareto processes are defined according to desired average load and H. In our simulations, in order to stress our algorithms as much as possible, we generally used a H parameter of 0.9 which is much higher than most of applications' empirically derived H parameters in [8]. We use this realistic self-similar method with large H, to test our algorithms, rather than simulating with existing benchmarks, because current limited number of applications are far away to stress our interconnect, where WiNoCoD aims to serve a vast spectrum of diverse applications in future.

In addition, we also emulated the bimodal packet sizes of cache coherency. As previously mentioned, authors in [61] found that short control packets constitute between 70%-80% of all generated packets, where rest are the cache line carrying long packets. Therefore, we set the ratio of long packets to 25%. Also, in [3] this ratio for bimodal packets is used. Any generated packet is determined to be either a long or short packet based on this probability, independently. Using information from literature and consulting to memory architectures of WiNoCoD, we set short packet lengths to 64 bits and long packet lengths to 576 bits (where 64 bits for control header, and 512 bits for cache line payload), without loss of generality.

4.4 Using Generic Cores for Bandwidth and Modulation Order Allocation Algorithmss

In this chapter, we have demonstrated a dynamic bandwidth allocation principle, where a dedicated intelligent component composed of one or several processors, is responsible of computing bandwidth allocation algorithms. In case of decentralized allocation, these intelligent units are implanted inside RF front-ends of each tileset, before the IFFT block, executing the same algorithms. In case of centralized allocation, only a single unit is implanted inside one tileset's RF front-end.

Now, in this section, we introduce the possibility of using few of the generic 2048 cores of the CMP, for the bandwidth allocation and modulation order selection as a concept. As it can be seen from Fig. 4.9, one can utilize, for instance, the core (or several cores) for bandwidth arbitration purposes, inside the tile, which is closest to the RF front-end. This way a low latency communication between this tile and the RF front-end can be attained. The information coming from the local transmission and reception queues, information coming from other tilesets (QSI, utilization of subcarriers etc.) can be transferred to the responsible cores of this tileset, with predefined packet formats. After the necessary computation, the resulting allocation can be re-transferred to the RF front-end, for bandwidth reconfiguration. Modulation order selection algorithms can also be computed via this procedure.

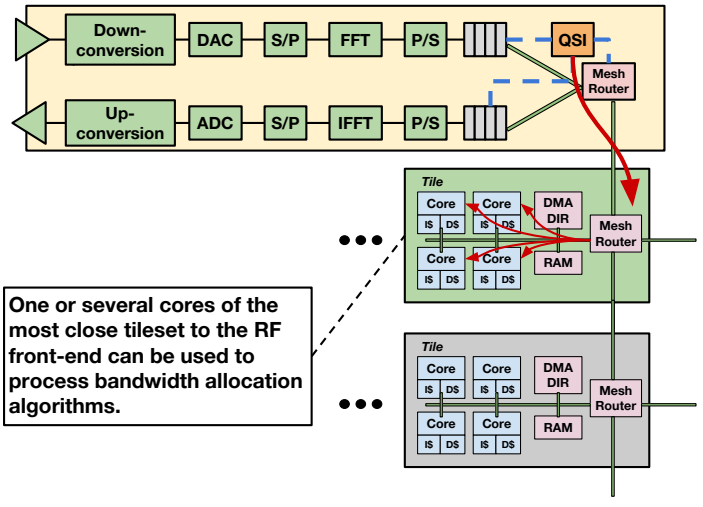


FIGURE 4.10: One or several cores inside the tile, which is closest to the RF Front-end can be utilized to compute banwidth and modulation order allocation algorithms. This implementation can be adopted both for decentralized or centralized approach.

With this approach both decentralized or centralized allocation can be performed. In decentralized case, a tile in each tileset shall be responsible for bandwidth allocation,

whereas for centralized case a tile inside just one tileset is necessary. Note that, applying a centralized case for this specific way of bandwidth allocation seems more scalable and efficient, as we only use just one tile. With this approach, we remove the extra area and cost overhead of implanting microprocessors or dedicated circuits inside RF front-ends, for computing allocation algorithms. However, note that, we have not studied the detailed implementation of this option in the scope of this thesis.

In order to decrease the communication latency further between the RF front-end and the responsible cores, their specific packets containing bandwidth allocation information can be assigned priority inside the mesh router over the other NoC packets coming from other tilesets to RF front-end.

With this approach, our previously presented pipelined and framed allocation paradigm gains more importance. One can see that, if this approach is adopted for bandwidth allocation, the reconfiguration and computation shall take more time. Therefore, in this case the necessity of a pipelined framework is more underlined. We have examined the proposed allocation algorithms for various frame lengths, hence this may consitute a guideline for this kind of implementation.

4.5 Conclusion

In this chapter, we have defined the goals of this thesis work precisely, by basing on the details of RF interconnect which was given in the previous chapter. It has been emphasized that the main interest of this thesis work for WiNoCoD is to allocate subcarriers to different tilesets to minimize the transmission latency as much as possible. We have justified our methodology for decoupling the effect of lower NoC layers and treat the resource allocation problem solely on the grounds of transmission side inter-tileset communication.

It has been remarked that the very specific constraints and features of the on-chip architecture presented in WiNoCoD, such as very short OFDM symbol duration or strict delay requirements, forces us to treat problem differently. We face an original problem of multiple queues on multiple symbols. We have introduced our frame based pipelined allocation scheme due to unorthodox, short OFDM symbol duration, which mitigates the latency of reconfiguration and other procedures.

Two different approaches to this defined bandwidth allocation problem have been explained : a centralized allocation, where a single microprocessor(s) or device is responsible for bandwidth arbitration computation or a decentralized allocation, where each tileset has a microprocessor(s) or device to compute the same bandwidth arbitration.

Pros and cons of these two complementary approaches have been listed. In addition, all necessary details on signaling and allocation procedures have been defined.

Finally, the metrics of interest for this thesis work and the stochastic traffic models for performance utilization have been presented. Note that, the information given in this chapter, will constitute the base of the to be presented resource allocation algorithms in next chapters.

Chapter 5

WiNoCoD Bandwidth Allocation Algorithms

In Chapter 4, the new problem of service allocation to multiple parallel queues, in multiple symbols is conceptualized, which is a fundamental projection of WiNoCoD subcarrier arbitration issue. Main metrics and goals of interest such as minimizing average latency, limiting delay-rate function or fairness have been discussed in detail also. Several approaches to this problem from literature are proposed in Chapter 4, along with their optimality under certain assumptions. Chapter 4 induces a broad sense of multi-user scheduling in general. Furthermore, preliminary concepts of subcarrier allocation for WiNoCoD due to unorthodox nature of on-chip architecture were presented in detail. Chapter 5, covers the proposed effective bandwidth allocation for the new problem of WiNoCoD's OFDMA RF interconnect. Under various configurations and input traffic models, pros-cons, feasibility and performance of proposed algorithms are evaluated extensively and classified as a tree diagram at the end of the chapter.

5.1 Serial QSI Allocation

5.1.1 Regular Serial QSI Allocation

First algorithm we propose is a relatively straight-forward one. After QSIs are broadcasted and acquired by each tileset, they allocate RBs in a frame (RB matrix) serially, looping through the QSI values of tilesets, i.e. serving requests of tilesets one-by-one. The algorithm terminates when there is no RBs left to allocate or if all requests of tilesets are served. In case, total QSI of all tilesets is fewer than the total number of RBs in

the frame, the remaining RBs are shared using default allocation matrix as in Section 4.3.3.5.

One can claim that, RB allocation can also be done one-by-one, looping each time through tilesets for each RB to allocate or with a completely different approach, however limiting the number of iterations is highly essential. Previously presented in Section 4.2.1, well known LQF requires high number of iterations like this. Considering our temporal budget of few hundreds of processor cycles for bandwidth arbitration computation, iterating through 32 tilesets only once is feasible.

Starting allocation from the same tileset in every allocation epoch will cause an unfairness, as first nodes will have priority for acquiring RBs. In these cases, if RBs are taken by these priority nodes, the rest will likely to starve. In order to avoid this, we propose a simple rotating priority scheme, that cyclically starts from the next tileset, every allocation epoch.

As serial allocation is a simple consecutive assignment of RBs to tilesets, no mathematical or computational demand exists. Hence, employing this algorithm in a decentralized allocation architecture is not too costly. However for the sake of coherency, we also perform our experimentation on serial allocation algorithm with centralized approach.

5.1.1.1 Decentralized Approach

Tilesets broadcast their QSIs on reserved subcarriers each T symbols (frame) as in Fig. 4.5. Then starting from the tileset with the instantaneous priority, the RBs equal to QSI demand of each tileset is allocated iteratively, based on the direction of allocation. Allocation terminates when there are no RBs left or the iteration is finished (i.e. all demands of tilesets are served). As allocation is performed on the default allocation matrix, if there are remaining RBs after iteration, they are arbitrated as explained in Section 4.3.3.5. Note that, in order to avoid inconsistency for bandwidth allocation and reulting congestion, each tileset shall perform the same algorithm without any error.

Average Latency

First, we evaluate the average latency performance of decentralized serial allocation algorithm with increasing injection rate for 4 different frame lengths ($T = 4, 8, 16$ and 32) under 3 stochastic traffic models explained in Section 4.3.4.2. As mentioned previously, different frame lengths may be imposed by the circuital constraints or be an intended choice. Longer frame lengths decrease the signaling overhead eventually, increasing the useful communication bandwidth. Framed and pipelined allocation is the result of the

TABLE 5.1: Total number of RBs for different lengths of frames and associated QSI signaling overhead percentage.

T (symbols)	Number of RBs	Overhead (%)
4	128	3.21
8	256	1.56
16	512	0.78
32	1024	0.39

short symbol durations, where we have to spare certain amount of time to microprocessors and components to reconfigure bandwidth occupation. Table 5.1 lists the total number of RBs for different lengths of frames and associated QSI signaling overhead percentage. As there may be different choices for the processing speeds of bandwidth reconfiguration units, we evaluate the proposed allocation algorithms for various frame lengths.

Fig. 5.1 shows the average latency curves under Uniform Poisson Traffic model, where the spatial or temporal variance is least. Before starting to evaluate the results, we would like to highlight a previously mentioned important aspect, which is known as the *network capacity* in queuing and network theory. When the average injection rate (input) surpasses the average service rate (i.e. utilization $\rho > 1$), the queue backlogs start to grow continuously, resulting in practically infinite average latency. Also, as average latency of a system grows quadrically with the utilization, even ρ values approaching 1 may cause very high latencies. In our WiNoCoD interconnect, as we have 1024 subcarriers encoded by QPSK, it can serve 2048 bits or 32 flits per symbol. As 25% of packets are 9 flits and 75% of packets are 1 flit long, the average service demand of a packet is 3 flits. Therefore, system capacity in our case is $32/3 = 10.66$ packets/symbol. In Fig. 5.1, we observe that near optimal OPF algorithm for each T values, is able to reach a point near to this theoretical capacity. However, for lower injection rates, OPF is performing remarkably worse compared to the allocation algorithm, because when the number of packets present in the queues is low, infrequent repartition (not every symbol, but every few symbols) of the bandwidth according to instantaneous latencies of the packets is not a valid approach. Therefore, we can state OPF does not constitute an efficient reference for low injection rates, but for the injection rates closer the system capacity.

Most remarkable results for uniform Poisson case with decentralized serial allocation is the relatively worse performance for $T = 4$ symbols, for injection rates larger than 5 packets/symbol. One would expect to see the shortest frame length to perform best as a result of more frequent, thus more accurate QSI information. As serial allocation, takes the broadcasted QSIs and allocates RBs serially to tilesets without any sophisticated treatment, when injection rate increases, tilesets' QSI values starts to increase also. Hence, when we have a short frame length, we have practically small number of RBs

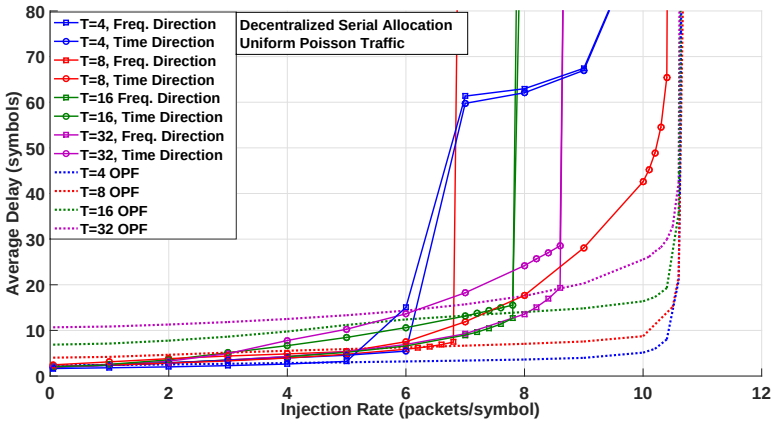


FIGURE 5.1: Average latency curves under uniform Poisson traffic with increasing injection rate for decentralized serial allocation algorithm.

to allocate. For $T = 4$ there are 124 RBs are available when QSI signaling subcarriers are excluded. In this case, when the QSI demands of tilesets are large, only one or few of the tilesets grab all the RBs in a frame. Even though, we have a rotating priority mechanism, this unfairness results in very large waiting times for tilesets to be served. We see from Fig. 5.1, this starvation effect deteriorates as frame length gets longer. In contrary, for small injection rates, shorter frame length, thus more frequent allocation shows better performance as expected.

Another interesting observation is on the direction of RB allocation. We observe that generally frequency direction allocation incurs a lower average latency compared to time direction. This roots from the default allocation matrix which reserves 1 RB for each tileset, when all QSI demands are served. In first few symbols of a frame, QSI demands of tilesets are served in chunks once at a time by using multiple RBs in a single symbol. When all allocation is terminated, the rest is partitioned uniformly.

As a next step, we evaluate the same curves under non-uniform Poisson traffic and non-uniform DPBPP traffic with H=0.9, in Fig. 5.2 and Fig. 5.3, respectively.

As it can be noticed from figures above, the most interesting result is the better performance of $T = 32$ than others under non-uniform DPBPP traffic in terms of approaching the system capacity. This is due to same starvation effect with lower frame lengths explained above. However, for smaller injection rates, still, the lower frame lengths show better performance as expected. Only, in uniform and non-uniform Poisson process, we see that $T = 8$ symbols of frame length is able to approach to system limit surprisingly, possibly due to being an effective compromise between frequent QSI signaling and low number of RBs to allocate once at a time.

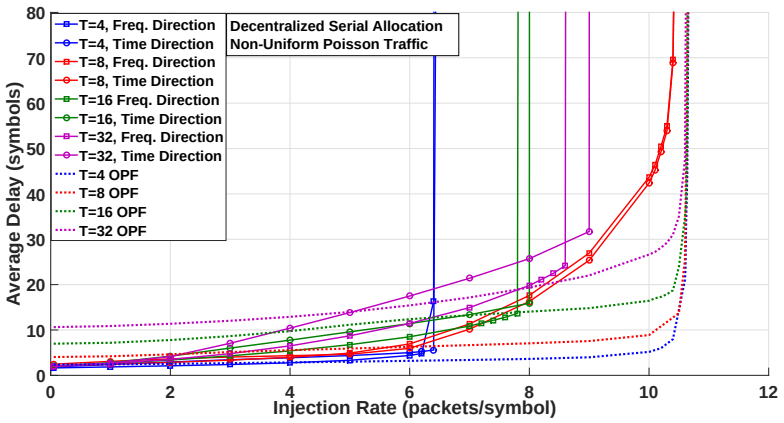


FIGURE 5.2: Average latency curves under non-uniform Poisson traffic with increasing injection rate for decentralized serial allocation algorithm.

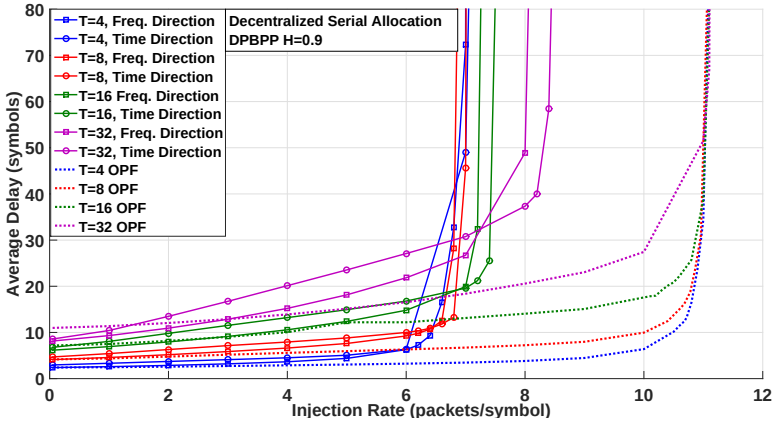
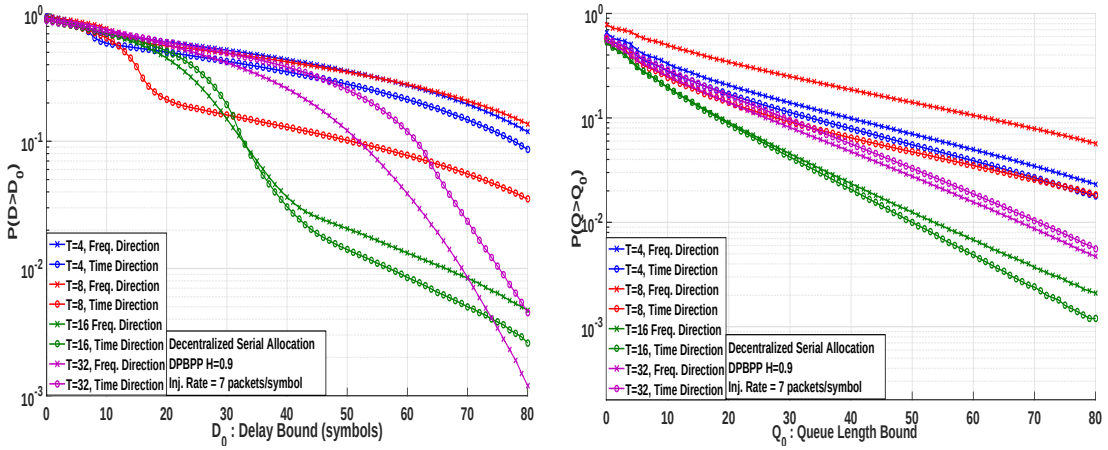


FIGURE 5.3: Average latency curves under non-uniform DPBPP traffic with increasing injection rate for decentralized serial allocation algorithm.

Packet Delay and Queue Length Bound Exceeding Probabilities

Next, we evaluate the delay and queue length exceeding probability curves (i.e. inverse cumulative probability distribution function) for different frame lengths for decentralized serial allocation algorithm. These curves give a good understanding on maximum delay a packet can experience or the necessary buffer capacity etc. as explained in Section 4.3.4.1. These curves are evaluated only for realistic and stressing non-uniform DPBPP traffic due to absence of space. By looking at Fig. 5.3, we see that for injection rates larger than 8 packets/symbol, all frame lengths fails (i.e. blows up), therefore we have selected an injection rate of 7 packets/symbol for this evaluation, where each frame length provides a reasonable average latency.

Fig. 5.4(a) and Fig. 5.4(b) show the probability graphs for a packet to exceed a certain delay or a transmission queue to exceed a threshold on any symbol, respectively.



(a) (b)

FIGURE 5.4: Packet Delay (a) and Queue Length (b) exceeding probability graphs for decentralized serial allocation algorithm under non-uniform DPBPP traffic (log-linear)

For instance, we observe in Fig. 5.4(a), that probability for a packet experiencing a delay between 30 and 70 symbols, is smallest for the frame length of 16 symbols with time direction allocation. However, for instance, probability of a packet experiencing a delay larger than 20 is minimum for $T = 4$ symbols and time direction allocation is around 0.5. Recalling an OFDM symbol is larger than 50 ns, one can claim that decentralized serial allocation does not perform well under an injection rate of 7 packets/symbol. For queue length exceeding graph, best performance for all thresholds is by T=16 symbols in time direction. For instance, under this configuration probability for a transmission queue of a tileset to have more than 50 flits on a symbol is around 0.01.

5.1.1.2 Centralized Approach

Whereas we have mentioned utilizing serial QSI allocation algorithm in centralized approach is not that much interesting due to no computation, however we perform the same experimentation procedure in previous section. As it was stated in Section 4.3.3.3, we give a certain amount of time for tilesets to receive allocation response from the CIU and reconfigure their transmissions. Throughout this thesis, when we are comparing centralized approach to decentralized approach, we always used a $T_{reconfig} = 2$ symbols. Hence, when comparing a decentralized frame length of 4 symbols (mainly due to computation time), we use a 6 symbols of frame length for the centralized approach. Recall that, in centralized approach we additionally have RBs reserved for transmission of response packets.

Fig. 5.5, Fig. 5.6 and Fig. 5.7 show the average latency with increasing injection rate under 3 different stochastic models for centralized serial allocation algorithm. One cannot

see any substantial difference between the decentralized case, except for the difference between the decentralized allocation with $T = 4$ symbols under uniform Poisson traffic. With centralized approach, we have a total frame length of 6 symbols, which eliminates the aforementioned starvation effect.

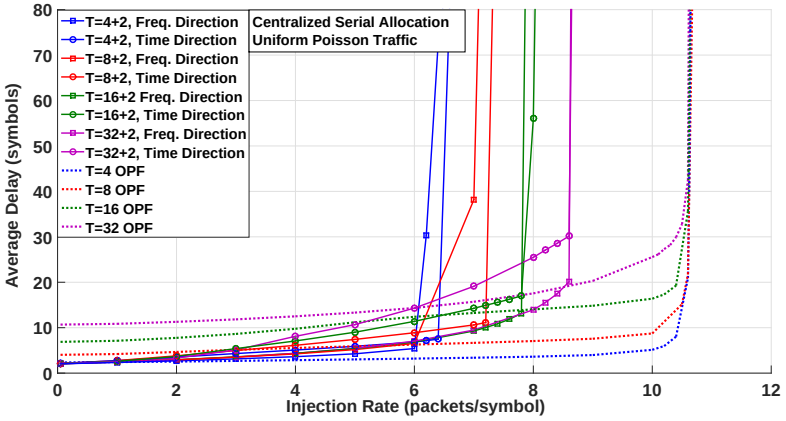


FIGURE 5.5: Average latency curves under uniform Poisson traffic with increasing injection rate for centralized serial allocation algorithm.

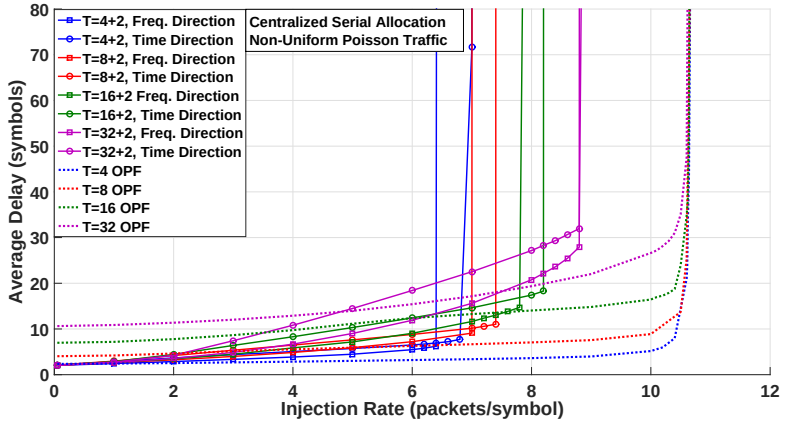


FIGURE 5.6: Average latency curves under non-uniform Poisson traffic with increasing injection rate for centralized serial allocation algorithm.

Packet delay and queue length exceeding probability graphs are also investigated, as for the decentralized case. Fig. 5.8 shows the similar performance of centralized case to decentralized case.

5.1.2 Serial QSI with 2-loop Allocation

Most unsatisfactory result from the regular serial QSI allocation algorithm was its significantly low performance in terms of approaching the network capacity. Most interestingly, frequent bandwidth allocation with very short frame length causes a significant increase

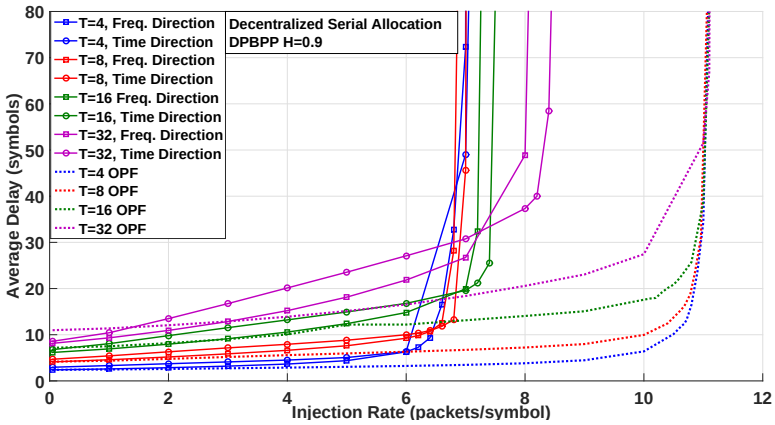
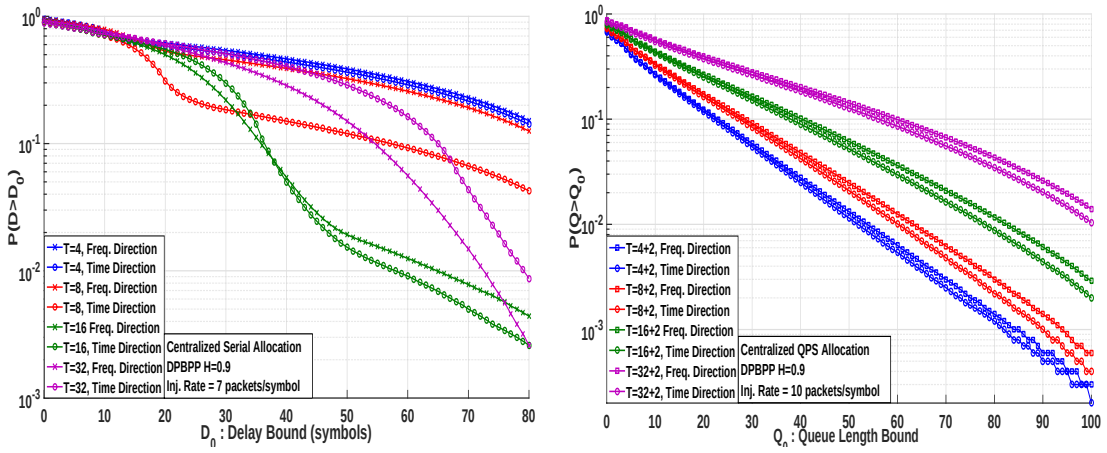


FIGURE 5.7: Average latency curves under non-uniform DPBPP traffic with increasing injection rate for centralized serial allocation algorithm.



(a) (b)

FIGURE 5.8: Packet Delay (a) and Queue Length (b) exceeding probability graphs for centralized serial allocation algorithm under non-uniform DPBPP traffic (log-linear)

in average latency. For instance, for a frame length of 4 symbols, regular QSI allocation causes to system to blow in terms of average latency, for injection rates larger than 6 packets/symbol. We have observed that this contradicting effect is due to low number of RBs to allocate under short frame lengths. By using neutral priority, certain tilesets exhaust RBs, starving tilesets who really need resources.

We have mentioned that the main motivation behind suggesting regular serial QSI allocation algorithm was its very low computational requirements, where only a single iteration of tilesets is performed. However, based our experiments, we have seen that this algorithm shall be improved through further modifications. In order to solve this fairness problem, and even increase the fairness among tileset transmission queues, we have established a modification of serial QSI allocation algorithm.

This time allocation of RBs is performed in 2 stages, as the name suggests. Firstly

after receiving QSIs of each tileset, they are summed and divided to number of tilesets to obtain the "average QSI" on current frame. This average QSI value is an integer, and found by upper rounding the result. At the first stage of the algorithm, 32 tilesets are iterated as in the previous simple serial allocation. However on this first loop, only tilesets who have a larger QSI than the current "average QSI" can grab RBs. After this first iteration, QSI values are updated by substracting the already allocated RBs in this frame. Followingly, at the second stage, the same iteration through tilesets are performed by their updated QSI values, this time as the default serial allocation algorithm. By this way, we make sure that tilesets who really need RBs get its share. This extension to regular serial QSI allocation algorithm can be seen as an attempt to compensate imbalances in queues. It is widely admitted in queuing theory that, optimal scheduling algorithms in terms of latency minimization, are the ones who balances the queues more effectively [90].

However, note that this algorithm may introduce another dimension of unfairness. If certain high load tilesets keep being over the average QSI, and exhaust all the RBs in first stage; other active tilesets with low QSI values may not grab resources for very long time. This starvation issue can be solved simply by by-passing first stage periodically every certain frames. However, this issue is not discussed in this thesis.

Note that, this modification does not only require an extra iteration, but also introduces mathematical operations such as 32 32-bit summations, a single division, 32 binary comparisons etc. Hence, it presents a certain degree of computational cost compared to regular QSI allocation.

5.1.2.1 Decentralized Approach

In decentralized approach, each tileset first calculates the "average QSI" by themselves, based on the broadcasted QSI values by all other tilesets. Then through the first iteration RBs are arbitrated in allocation matrix in time or frequency direction. The second iteration continues to allocate RBs from the last RB of the first iteration. This way, allocation or RBs are pipelined and faster.

Average Latency

We perform our usual average latency experiments for this method to evaluate whether proposed modification increases the performance.

Comparing Fig. 5.9, Fig. 5.10 and Fig. 5.11 to Fig. 5.1, Fig. 5.2 and Fig. 5.3, respectively; we see a substantial performance increase. Thanks to its added fairness, this algorithm is able to reach near the network capacity. Most remarkable result is for

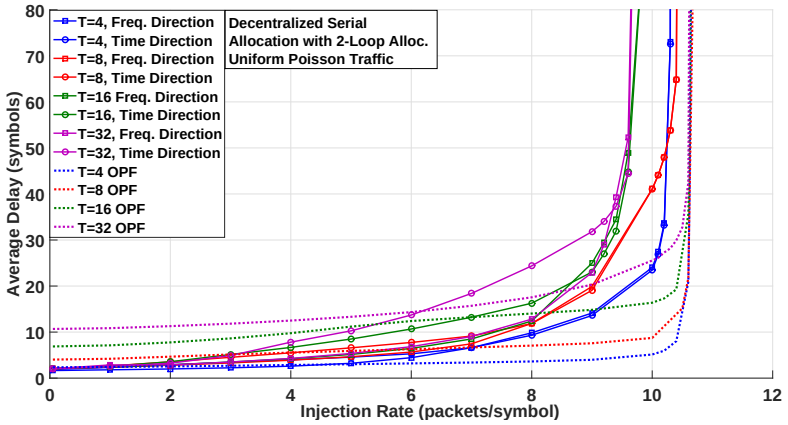


FIGURE 5.9: Average latency curves under uniform Poisson traffic with increasing injection rate for decentralized serial 2-loop allocation algorithm.

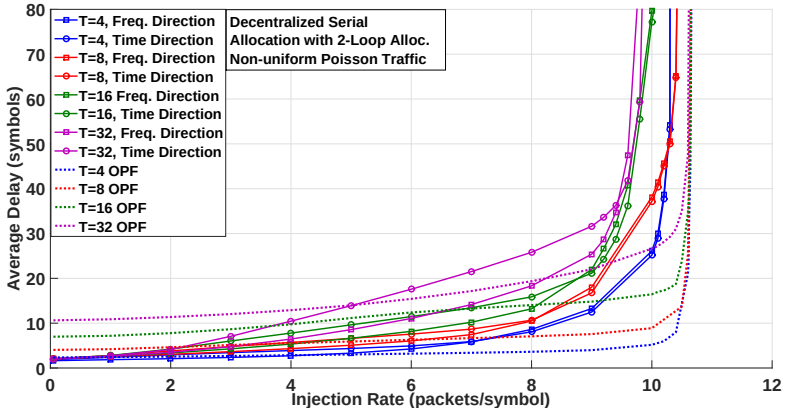


FIGURE 5.10: Average latency curves under non-uniform Poisson traffic with increasing injection rate for decentralized serial 2-loop allocation algorithm.

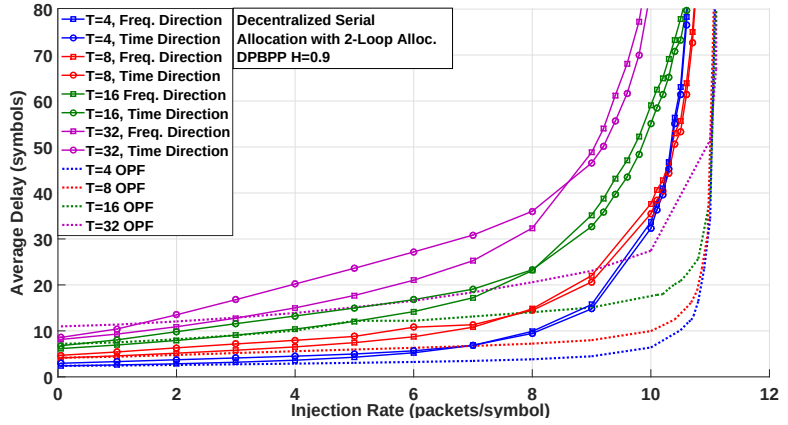


FIGURE 5.11: Average latency curves under non-uniform DPBPP traffic with increasing injection rate for decentralized serial 2-loop allocation algorithm.

the shortest frame length of $T = 4$ symbols. With 2-loop modification, now with a frame length of $T = 4$ symbol performs best compared to longer frame lengths as expected.

We have seen in previous section that, even though exchanging QSI more frequently should give lower average latencies, due to starvation phenomenon explained, it was degrading the performance. Now, by this simple modification with very limited additional complexity, we have shown that this effect can be mitigated.

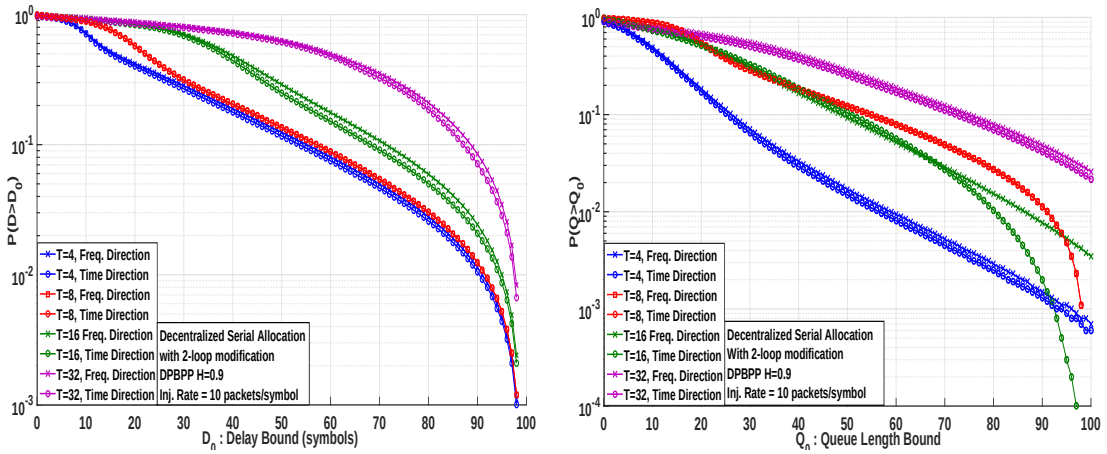
By inspecting, Fig. 5.9, Fig. 5.10 and Fig. 5.11, we can claim that proposed algorithm induces reasonable and scalable average latencies, especially for short frame lengths. Considering it has very limited computational requirements, we can state this algorithm can be implemented easily for $T = 4$ or 8 symbols, while we are taking all other delays incurred due to reconfiguration, propagation, serialization, packet treatment etc.

Another point to mention is on the direction of RB allocation. Different than the previous case, we see that now both direction of allocation in frequency or time, achieves nearly the same capacity. For lower injection rates, frequency direction provides lesser average latency. This is due to the fact that by allocating RBs in frequency direction, tilesets with large QSIs are able to be serve all packets in few symbols. Of course, as frame lengths longer, the gap between frequency and time direction allocation widens.

Packet Delay and Queue Length Bound Exceeding Probabilities

Next, in order to compare the improvement of this 2-loop algorithm we evaluate the packet delay and queue length exceeding probability graphs under realistic non-uniform DPBPP traffic. However, instead of 7 packets/symbol injection rate we choose to inspect the performance under higher traffic load. 7 packets/symbol of injection rate was a threshold, where all different frame lengths were under network capacity. Thanks to fairness with 2-loop algorithm, this threshold is further expanded. Hence, we choose an injection rate of 10 packets/symbol for the evaluation of maximum delay and buffer capacity.

Even under an injection rate of 10 packets/symbol with severe temporal and spatial burstiness, by evaluating Fig. 5.12, we see that serial allocation with 2-loop modification provides reasonable delays for packets and buffer capacities, despite its low computational complexity. However, considering an OFDM symbol is longer than 50 ns and strict requirements of on-chip packets, one can argue this performance gain is not sufficient, especially when the traffic load is high. For example, from Fig. 5.20(a), we see that probability for a packet in any queue in the system to experience a delay larger than 50 symbols is approximately 0.1, even under a frame length of 4 symbols.



(a) (b)

FIGURE 5.12: Packet Delay (a) and Queue Length (b) exceeding probability graphs for decentralized serial 2-loop allocation algorithm under non-uniform DPBPP traffic (log-linear) under 10 packets/symbol injection rate

5.1.2.2 Centralized Approach

In centralized version of the 2-loop serial allocation, CIU is responsible of acquiring QSIs, calculate the average QSI, than perform the allocation in 2 iterations. However, different than the decentralized approach, CIU first calculates the total number of RBs allocated to each tileset and broadcasts them in response. Then tilesets reconfigure their transmission matrix, simply by occupying RBs consecutively based on the direction of allocation. Even though, the computational burden is minimal for this type of an algorithm, we choose to implement the centralized version of it.

Average Latency

First, average latency performance with increasing injection rate under 3 different stochastic models are evaluated.

From Fig. 5.13, Fig. 5.14 and Fig. 5.15, we see that despite having an extra signaling overhead for response and 2 symbols of additional latency for reconfiguration, average latency performance is degraded slightly.

Packet Delay and Queue Length Bound Exceeding Probabilities

To check the delay fairness and maximum buffer occupancy, next we inspect the packet delay and queue length exceeding probability graphs under non-uniform DPBPP traffic with 10 packets/symbol injection rate.

As it can be seen in Fig. 5.16, centralized version of the 2-loop serial allocation does not provide drastically changed delay and queue length exceeding probability performance, due to 2 symbol extra latency or additional overhead and block allocation of RBs.

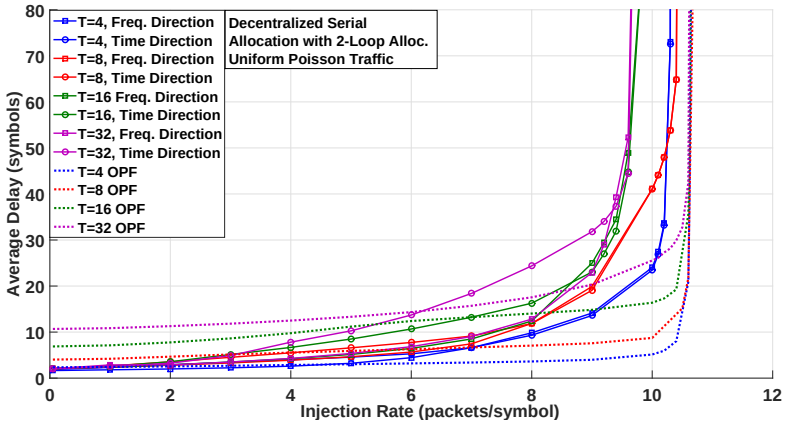


FIGURE 5.13: Average latency curves under uniform Poisson traffic with increasing injection rate for centralized serial 2-loop allocation algorithm.

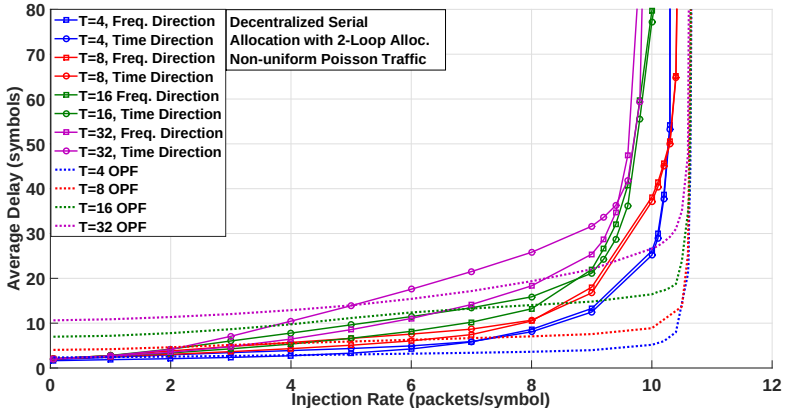


FIGURE 5.14: Average latency curves under non-uniform Poisson traffic with increasing injection rate for centralized serial 2-loop allocation algorithm.

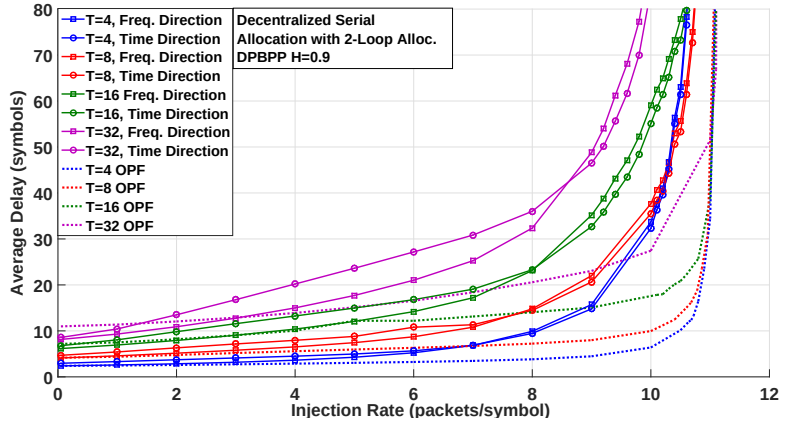
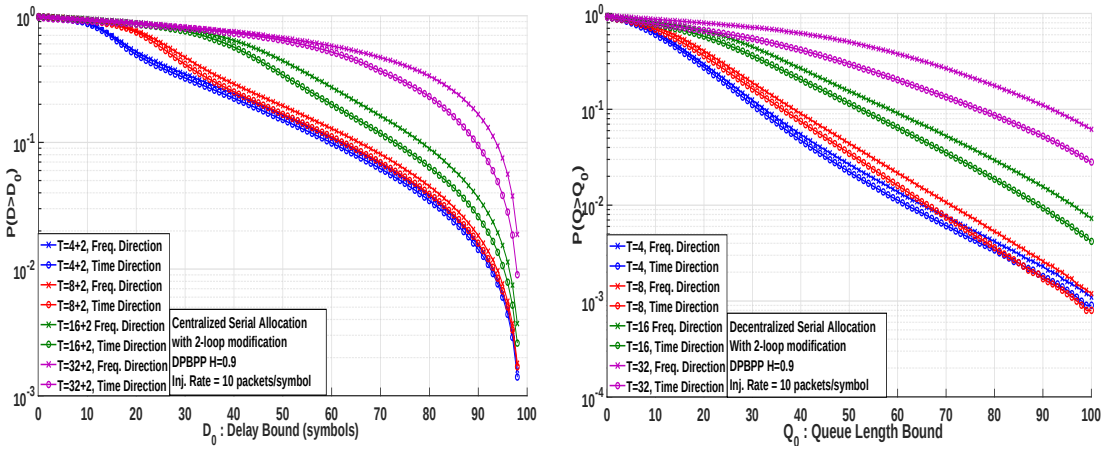


FIGURE 5.15: Average latency curves under non-uniform DPBPP traffic with increasing injection rate for centralized serial 2-loop allocation algorithm.



(a) (b)

FIGURE 5.16: Packet Delay (a) and Queue Length (b) exceeding probability graphs for centralized serial 2-loop allocation algorithm under non-uniform DPBPP traffic (log-linear) under 10 packets/symbol injection rate

5.1.3 Serial QSI Allocation with DQSI and EQSI

We have previously introduced the notion of Definitive Queue State Information (DQSI) and Expected Queue State Information (EQSI) in Section 4.3.3.7, to cope the outdated QSI due to pipelined allocation. We have seen the possible augmentation of capacity by employing a simple 2-loop modification. Now, we will investigate serial allocation algorithm under DQSI and EQSI modification. As α can take a vast amount of different values, due to absence of space, we only investigate the performance under $\alpha = 0.95$, which is generally provides the best performance in most of the scenarios.

5.1.3.1 Decentralized Approach

As mentioned in Section 4.3.3.7, in decentralized approach, each tileset computes its own DQSI or EQSI value by using currently allocated number of RBs and/or instantaneous moving average. Then, this value is broadcasted on the first symbol of the next frame. Then, as in the plain serial allocation in Section 5.1.(a), the RB allocation is performed in frequency or time direction, using these values.

Average Latency

Firstly, we investigate the average latency with increasing injection rate with DQSI and EQSI based serial allocation algorithm, for different frame lengths under 3 different stochastic models.

Evaluating Fig. 5.17, Fig. 5.18 and Fig. 5.19 and comparing them to simple regular serial allocation, the most remarkable outcome is to see the substantial average latency

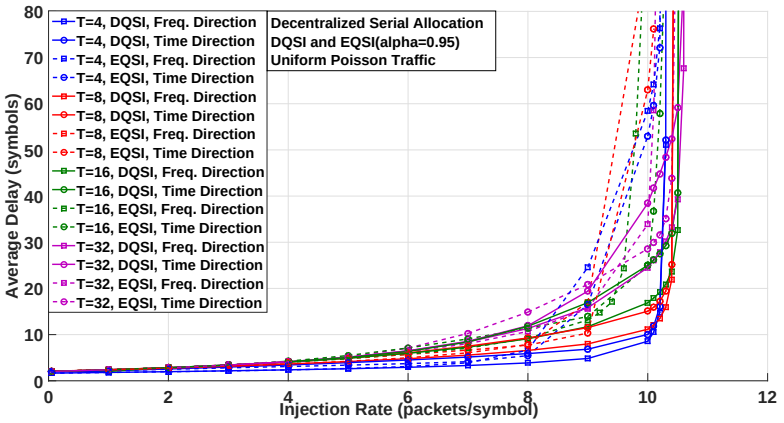


FIGURE 5.17: Average latency curves under uniform Poisson traffic with increasing injection rate for decentralized serial allocation with DQSI and EQSI algorithm.

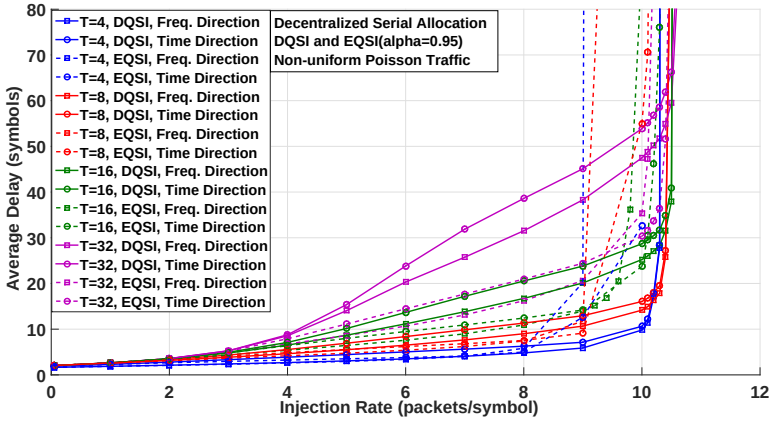


FIGURE 5.18: Average latency curves under non-uniform Poisson traffic with increasing injection rate for decentralized serial allocation with DQSI and EQSI algorithm.

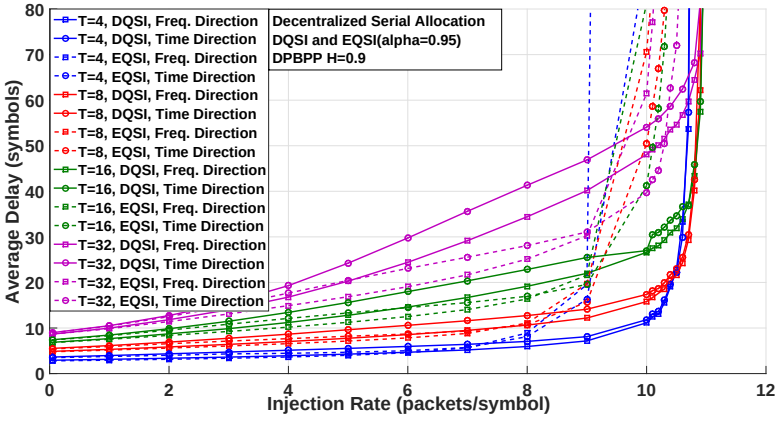


FIGURE 5.19: Average latency curves under non-uniform DPBPP traffic with increasing injection rate for decentralized serial allocation with DQSI and EQSI algorithm.

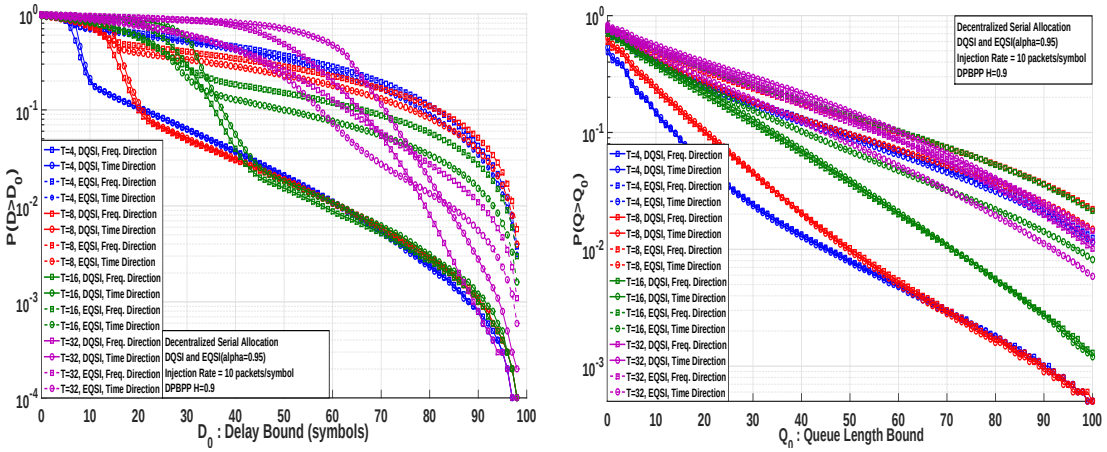
decrease by just utilizing DQSI. It is not only eliminating the unnecessary resource allocation, but also compensates the unfairness due to small number of RBs. For instance, comparing Fig. 5.8 and Fig. 5.22, we see that for plain serial allocation algorithm with a frame length of 4 symbols, queues blow up for injection rates higher than 7 packets/symbol, whereas DQSI based allocation can still provide average latency smaller than 10 symbols, for injection rates up to 10 packets/symbol. We see that for this case, EQSI provides a capacity around up to 9 packets/symbol.

The main motivation behind DQSI's strong improvement lies at its ability to avoid redundant service allocation, thus making sure only highly loaded queues gets their appropriate share from bandwidth. Apparently, it is even providing better fairness under these evaluated scenarios than 2-loop allocation mechanism. DQSI encoding requires only a single subtraction operation before broadcast of QSIs, which can be performed in a single cycle. In addition default allocation matrix mechanism, makes sure the non-utilized RBs are evenly arbitrated among all tilesets.

As expected, as frame gets longer, EQSI provides much better results compared to DQSI. However note that, DQSI just only require a single cycle subtraction operation which makes it feasible for short frame lengths such as 4 symbols.

Packet Delay and Queue Length Bound Exceeding Probabilities

Next, we evaluate the queue length and packet delay exceeding probability graphs for DQSI and EQSI under non-uniform DPBPP traffic.



(a) (b)

FIGURE 5.20: Packet Delay (a) and Queue Length (b) exceeding probability graphs for decentralized serial allocation with DQSI and EQSI algorithm under non-uniform DPBPP traffic (log-linear)

Evaluating Fig. 5.20(a), we see the strong performance of DQSI in terms of minimizing delay exceeding probability. For delay bounds larger than 40 symbols, the probability of

exceeding with DQSI for T=4, T=8, and T=16 follows practically the curve. Comparing DQSI to 2-loop algorithm (Fig. 5.12), we see the dramatic performance augmentation. For instance, with T=4 symbols of frame length, the probability of having a packet delay larger than 10 symbols for decentralized DQSI algorithm (both frequency and time direction) is around 0.1 while for decentralized 2-loop allocation it is around 0.8. We do not observe the same performance extension for queue length exceeding probabilities. Even though, for small queue length bounds probabilities are smaller for DQSI compared to 2-loop extension, for instance probability of a queue length to exceed 90 flits is practically the same.

Another result from Fig. 5.20(a) signifies that EQSI provides lesser delay exceeding probability (for delay bounds smaller than 80 symbols), compared to DQSI, with a frame length of 32 symbols. This is quite convenient as, with larger frames, QSI gets outdated more severly and the average input traffic has to be included in the allocation procedure. However, recall that, DQSI requires only a subtraction operation before QSI encoding, which makes it viable for short frame lengths.

5.1.3.2 Centralized Approach

In this section, centralized version of the DQSI and EQSI extension is evaluated with 2 symbols of additional reconfiguration delay and extra overhead for responses. As mentioned in Section 4.3.3.7, tilesets only broadcast their instantaneous QSI every frame, whereas CIU keeps the number of RBs of allocated in previous frame, QSIs reported in previous frame for each tileset, moving averages and computes the DQSI and EQSI.

Average Latency

The average delay with increasing injection rate and probability graphs for delay bound and queue length bound exceeding probabilities are examined to check whether a subtle degradation occurs due to centralization.

By comparing Fig. 5.17, Fig. 5.18 and Fig. 5.19 with Fig. 5.21, Fig. 5.22 and Fig. 5.23 respectively, we do not observe any intense increase in average latency (except few symbols) or any observable pattern change. Therefore, we can claim if DQSI or EQSI based serial allocation would be implemented, a centralized approach can be taken to decrease the computational complexity at the tileset front-ends. Similarly as for the decentralized case, EQSI provides much better performance especially for larger frame lengths.

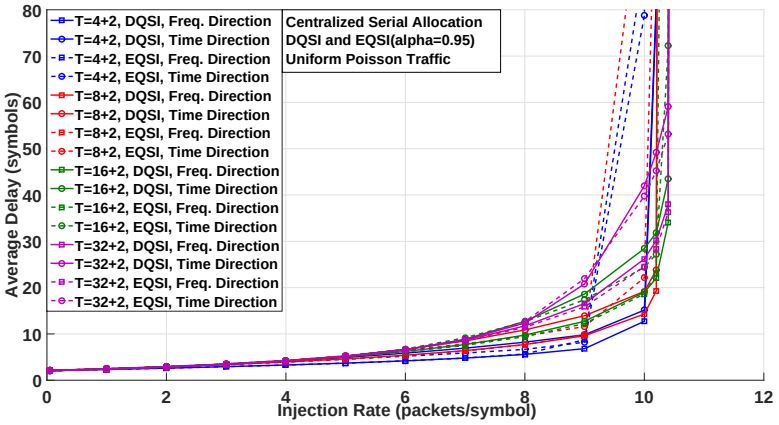


FIGURE 5.21: Average latency curves under uniform Poisson traffic with increasing injection rate for centralized serial allocation with DQSI and EQSI algorithm.

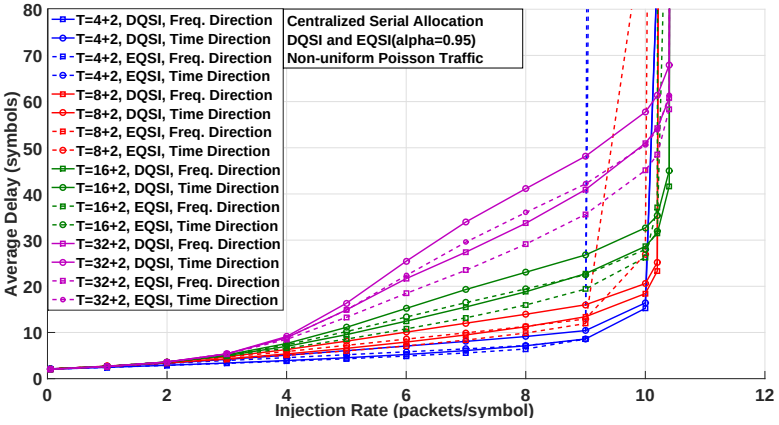


FIGURE 5.22: Average latency curves under non-uniform Poisson traffic with increasing injection rate for centralized serial allocation with DQSI and EQSI algorithm.

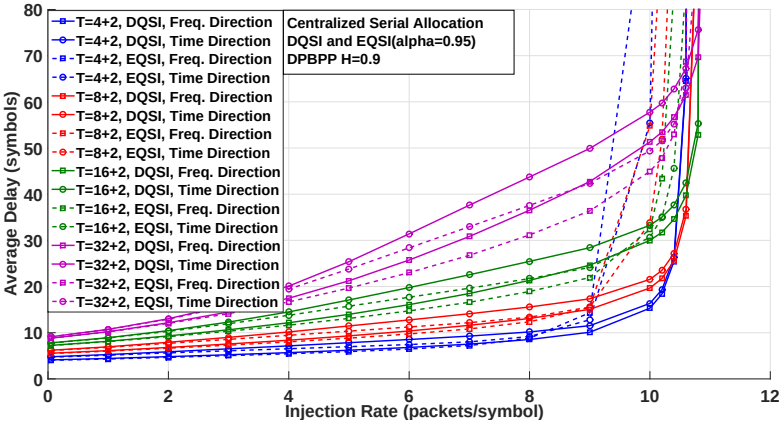
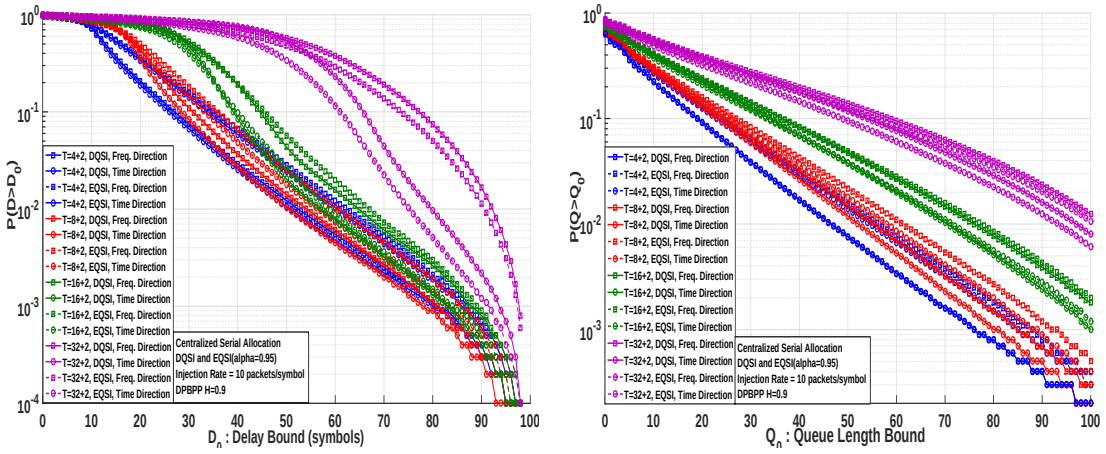


FIGURE 5.23: Average latency curves under non-uniform DPBPP traffic with increasing injection rate for centralized serial allocation with DQSI and EQSI algorithm.

Packet Delay and Queue Length Bound Exceeding Probabilities

Delay bound and queue length bound exceeding probability graphs are also evaluated for the centralized DQSI or EQSI based serial QSI allocation, under non-uniform DPBPP traffic with a total injection rate of 10 packets/symbol.



(a) (b)

FIGURE 5.24: Packet Delay (a) and Queue Length (b) exceeding probability graphs for centralized serial allocation algorithm with DQSI and EQSI under non-uniform DPBPP traffic (log-linear)

We see in Fig. 5.24, that probability of exceeding delay bounds (especially small delay bounds such as 10-20 symbols) is slightly larger for centralized approach both for DQSI and EQSI, especially for the small frame lengths (4-8 symbols), which is an expected outcome. Besides this small performance degradation, centralized and decentralized DQSI or EQSI shows close performance under these circumstances, which may be a good reason to choose centralized version of it to concentrate computational power in a single part of the CMP.

5.2 Queue Proportional Allocation

5.2.1 Regular Queue Proportional Allocation

In previous section (5.1), we have proposed and examined the feasibility of possibly the simplest option, serial QSI allocation algorithm, where tilesets broadcast their instantaneous QSI and RBs iniside a frame are arbitrated sequentially in a single loop. The major motivation behind this kind of a practice is to be able to effectuate allocation operation in few hundreds of nanoseconds including all other extra cumulative delays stemming from propagation, synchronization, processing etc., which is a strict constraint imposed by the 20 GHz OFDMA interconnect and on-chip environment. On-chip packets demand

delays as low as possible, tens of nanoseconds. Allocating RBs in a single loop can also be performed in small frame lengths, thus more and more up-to-date QSIs, which shall enhance the performance further. However, we have observed that this simple allocation pattern is not suitable to approach the capacity of the system, contradictory for smaller frame lengths, due to the starvation of all RBs in a frame by certain nodes. We were able to augment the performance of this algorithm by adding a second iteration. However, we have seen that most of the performance from this algorithm can be gained through using DQSI with small frame lengths such as 4 or 8 symbols.

In this section, we evaluate the viability of the Queue Proportional Scheduling (QPS) algorithm with additional limited complexity compared to serial allocation. The main idea is to allocate RBs in a frame proportionally to QSIs of the tilesets.

Formally, these operations can be written as :

$$S_i^{t+1} = \left\lceil N \frac{Q_i^t}{\sum_{j=1}^{K} Q_j^t} \right\rceil \tag{5.1}$$

where N is the number of RBs, K is the number of tilesets and S_i^{t+1} is the number of RBs allocated to tileset i in next frame. Note that, as the resulting ratio can be a fractional, it is always upper rounded. By this way, first we make sure that even a node with a QSI of 1 would be allocated a RB (thus preventing starvation) and second we avoid any RB left unassigned (therefore, there is no notion of default allocation matrix for as in previous serial allocation algorithm). However, one can see that, this would cause the calculated total number of RBs to exceed the total available number of RBs, N. This is simply resolved by the sequential RB allocation through a frame in time or frequency direction, just as in the previous section. This time, rather than using directly the emitted QSI values, recently calculated S_i^{t+1} values are taken from tilesets in a loop, and RBs are allocated as for the serial allocation algorithm. Hence, allocation terminates when there is no RBs left to assign. One can see that, the tilesets which are sequenced last may have a certain degree of disadvantage. Therefore, just as in the serial allocation algorithm a simple rotating priority mechanism is employed.

Another important point to highlight is on the computational complexity of the algorithm. First, QSIs of K (32 for our case) tilesets are summed. In case of 8 bits encoding is preferred as evaluated in this thesis work, for 32 tilesets, this operation yields to a 13-bit summation. Hence, 16 bit simple processors can be used, or in case of using 32 or

64 bit processor(s), summation can be performed in parallel, thus in lesser time. As can be seen in equation (5.1), the division of N and QSI summation is multiplied by the QSI of each tileset, therefore this value can be calculated once and kept in a register, and utilized repeatedly. This shall save a significant amount of computation time. At the end, there is one division operation (minimum 13-bits with 8-bit QSI), 32 summations and 32 multiplications. Multiplications and divisions can be performed in fixed point manner as numerical accuracy is not a problem for this specific allocation, which would ease further the task of the RB allocation processor(s).

Optimality Issue of Queue Proportional Scheduling

In Section 4.2.3, while reviewing the literature for rate scheduling, we have mentioned about the arbitration of bandwidth among multiple parallel queues proportional to square root of their QSIs. This was the solution to an optimization problem intending to solve the "minimum draining time", i.e. to minimize the time required to clear out a vector of K queues, by allocating the appropriate fraction of the whole rate (bandwidth) :

$$argmin \left\{ \frac{Q_1}{r_1} + \ ... \ + \frac{Q_K}{r_K} \right\} \ s.t., \ \sum_{i=1}^{K} r_i = R \tag{5.2}$$

This problem intends to minimize the total draining time of the queues, whereas we are more interested in minimizing the total delays experienced by packets in these parallel queues. Consider this illustrative simple example that a queue has P packets at $t = 0$ and it has a constant service rate of 1 packet/slot. The draining time for the queue is obviously P slots. In contrast, the individual delays for each packet is 1 slot for the first packet served, 2 slots for the second packet served, ... , and P slots for the last packet served, due to waiting time for service. Hence, at the end total (or average if divided by the total number of packets) delay experienced by all packets, D_{total} is :

$$D_{total} = \sum_{i=1}^{P} i = \frac{P(P+1)}{2} \tag{5.3}$$

In a nutshell, we observe a cumulative effect for total delay experienced by packets with the instantaneous number of packets in a queue. By looking at the above equation, we understand that total (average) delay for the packets in a queue is related with the square of the instantaneous queue state.

Returning back to the equation (5.2), for the minimization of the draining time, we can reformulate this equation loosely to minimize the "square of the QSIs", for minimizing average packet delay. As solution to the previous problem is to divide the total rate proportionally to the square root of QSIs, the solution to the new problem is simply to divide the total rate proportional to the instantaneous QSIs.

5.2.1.1 Decentralized Approach

First, decentralized version of the QPS algorithm is evaluated, where each tileset execute the same operation to arbitrate RBs in a frame, proportional to broadcasted QSI values. RBs are assigned both in frequency and time direction. As for the previously proposed serial QSI allocation algorithm, identical experiments and traffic models are evaluated.

Average Latency

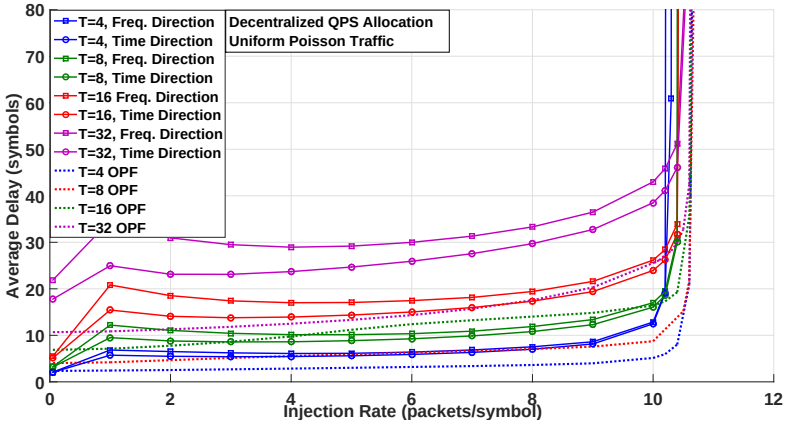


FIGURE 5.25: Average latency curves under uniform Poisson traffic with increasing injection rate for decentralized QPS allocation algorithm.

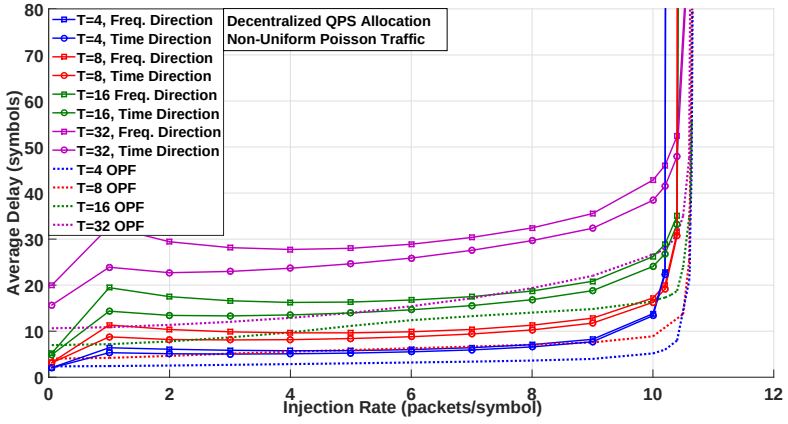


FIGURE 5.26: Average latency curves under non-uniform Poisson traffic with increasing injection rate for decentralized QPS allocation algorithm.

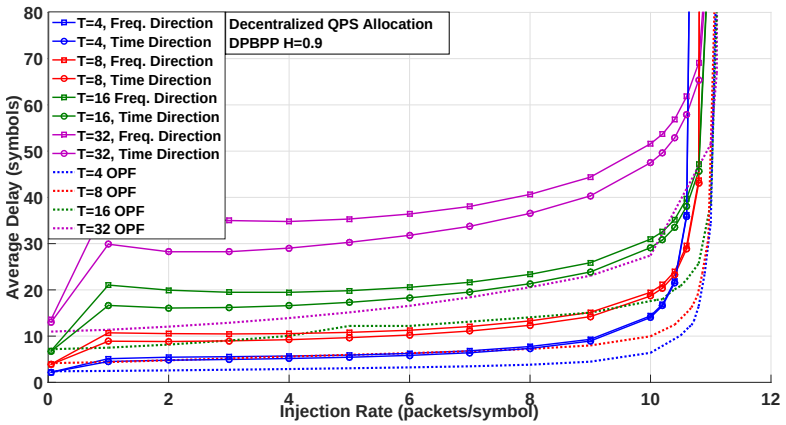


FIGURE 5.27: Average latency curves under non-uniform DPBPP traffic with increasing injection rate for decentralized QPS allocation algorithm.

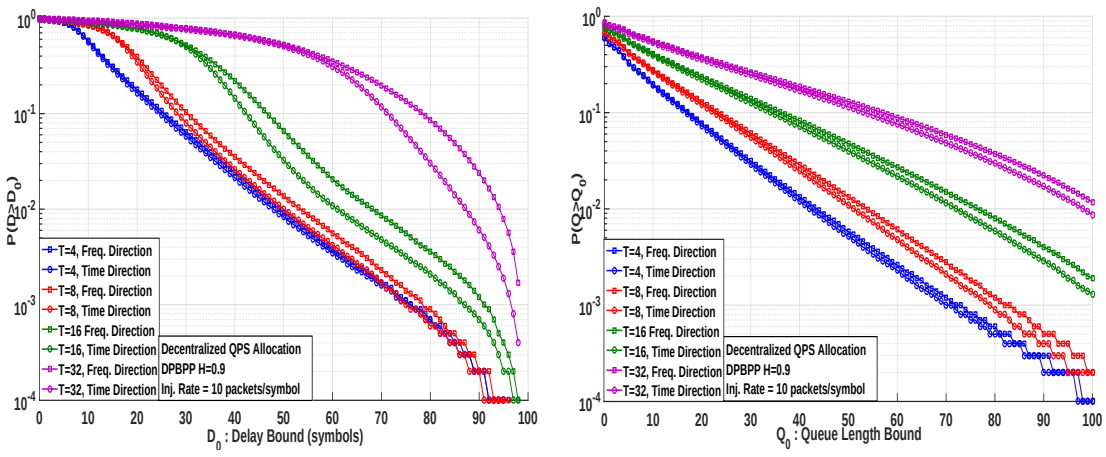
First thing we notice in these average latency curves is the remarkably high average latency under low injection rates, especially for longer frame lengths. Comparing these figures (Fig. 5.25, Fig. 5.26 and Fig. 5.27) to serial QSI allocation, we do not observe this high latency offset for low injection rates. This effect roots from the nature of QPS. If a tileset has zero QSI at the start of a frame, it will be allocated no RBs during the whole next frame, as allocation is done proportional to QSIs. Due to the pipelined allocation, if a packet arrives to an idle queue, it has to wait until the start of next frame to get a share of the total bandwidth. However, in regular serial QSI allocation, as we have mentioned, there exists the notion of "default frame allocation", which arbitrates unassigned RBs evenly among all 32 tilesets. Thus, even a tileset has zero QSI, it will be allocated certain amount of RBs, which avoids this phenomenon. Note that, obviously this effect gets more evident under lower injection rates.

In contrast, QPS provides a much better performance for larger injection rates, especially near system capacity compared to serial allocation (without any modification). Because, naturally allocating RBs proportionally to QSIs, eliminates the probability that certain nodes to exhaust all the RBs in a frame and starve other nodes.

From these figures, we observe that, QPS algorithm can approach to near the system limit even without any modification. Especially for small frame lengths, we observe that QPS provides reasonably low average latencies.

Packet Delay and Queue Length Bound Exceeding Probabilities

As a next step, packet delay bound and queue length exceeding probability graphs are evaluated under non-uniform DPBPP traffic with an injection rate of 10 packets/symbol.



(a) (b)

FIGURE 5.28: Packet Delay (a) and Queue Length (b) exceeding probability graphs for decentralized QPS allocation algorithm under non-uniform DPBPP traffic (log-linear)

5.2.1.2 Centralized Approach

In centralized version of the QPS algorithm, calculation of RBs proportional to QSIs are performed by the CIU and based on the response messages, bandwidth is reconfigured with 2 symbol latency as for the serial allocation algorithm.

Average Latency

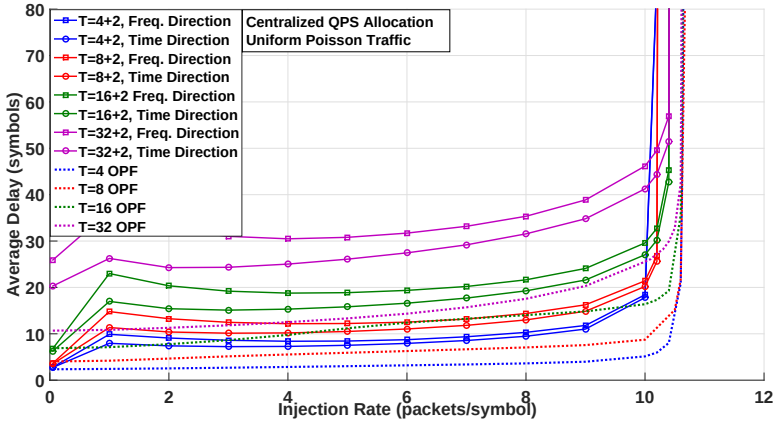


FIGURE 5.29: Average latency curves under uniform Poisson traffic with increasing injection rate for centralized QPS allocation algorithm.

Fig. 5.29, Fig. 5.30 and Fig. 5.31 show the slight increase in average latencies with centralizing the algorithm due to extra latency and signaling overhead, however performance patterns are similar with the decentralized case.

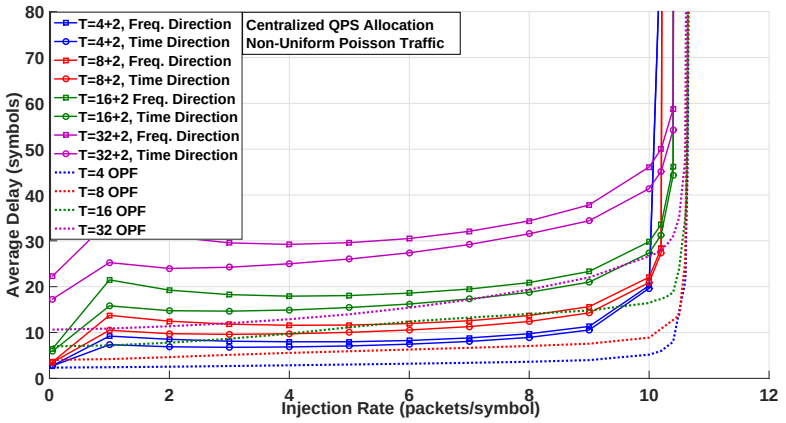


FIGURE 5.30: Average latency curves under non-uniform Poisson traffic with increasing injection rate for centralized QPS allocation algorithm.

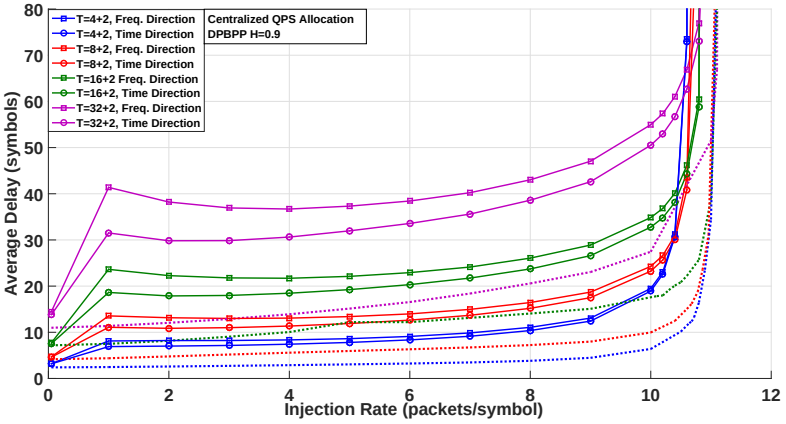


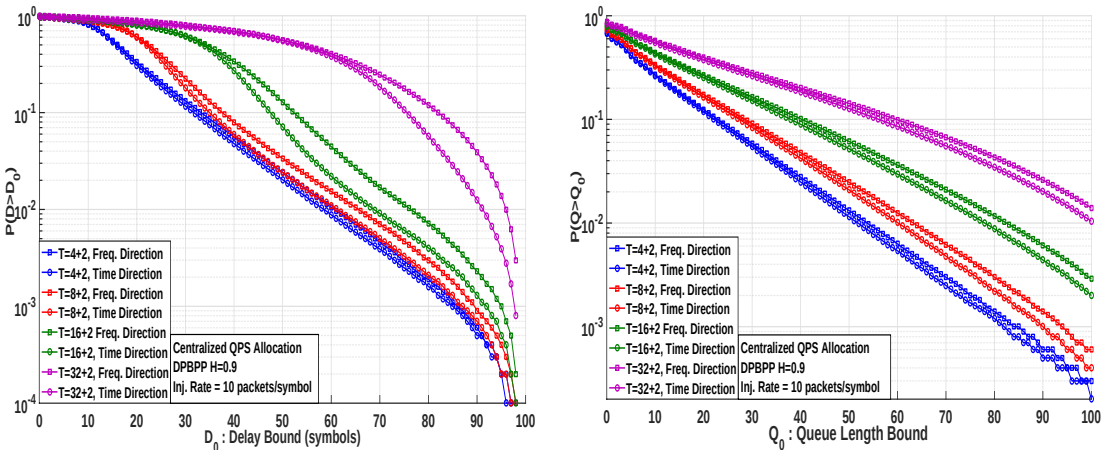
FIGURE 5.31: Average latency curves under non-uniform DPBPP traffic with increasing injection rate for centralized QPS allocation algorithm.

Packet Delay and Queue Length Bound Exceeding Probabilities

Next, we evaluate the packet delay and queue length exceeding probability graphs for centralized QPS algorithm. The slight performance degradation due to centralization can be observed in Fig. 5.32.

5.2.2 QPS Allocation with DQSI and EQSI

We seek to increase the performance of QPS algorithm by using DQSI or EQSI encoding especially for small injection rates. As for the previous cases, decentralized and centralized version of DQSI and EQSI differs.



(a) (b)

FIGURE 5.32: Packet Delay (a) and Queue Length (b) exceeding probability graphs for centralized QPS allocation algorithm under non-uniform DPBPP traffic (log-linear)

5.2.2.1 Decentralized Approach

Average Latency

First, we investigate the average latency with increasing injection rate for decentralized QPS with DQSI and EQSI under different traffic models.

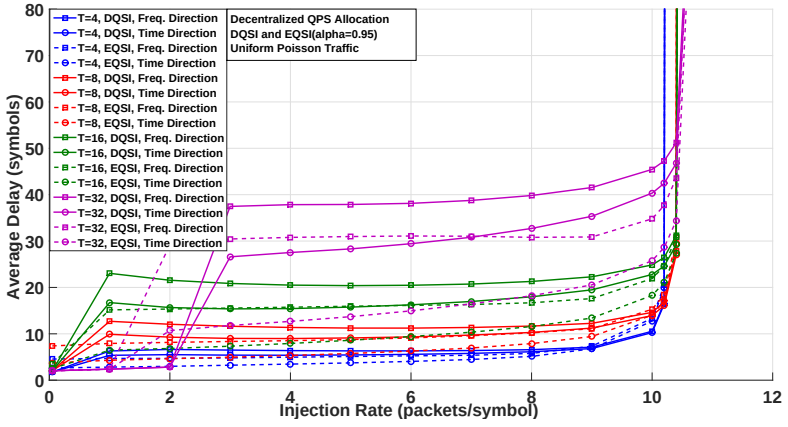


FIGURE 5.33: Average latency curves under uniform Poisson traffic with increasing injection rate for decentralized QPS allocation with DQSI and EQSI algorithm.

In Fig. 5.33, Fig. 5.34 and Fig. 5.35, we see that average latencies are significantly lowered with EQSI or DQSI for small injection rates. Arbitrating bandwidth proportionally to expected QSIs allow the allocation of resources for nodes who are likely to have packets in their queues on next frame.

Another important remark for QPS algorithm is on the direction of allocation. We observe that for most of the cases, assigning RBs in time direction is much better than

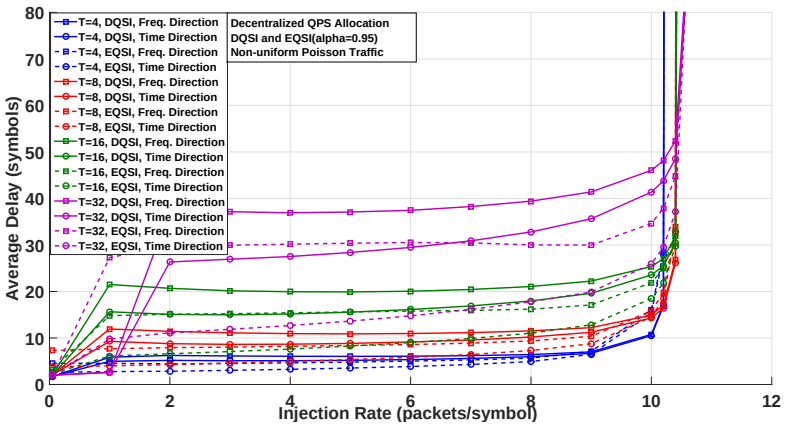


FIGURE 5.34: Average latency curves under non-uniform Poisson traffic with increasing injection rate for decentralized QPS allocation with DQSI and EQSI algorithm.

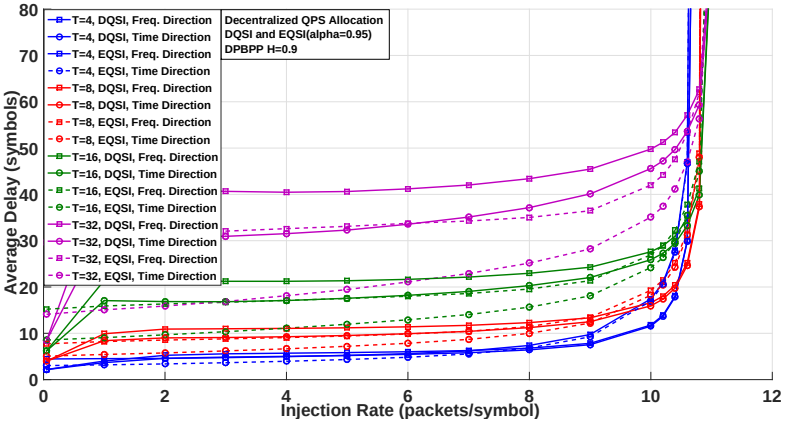
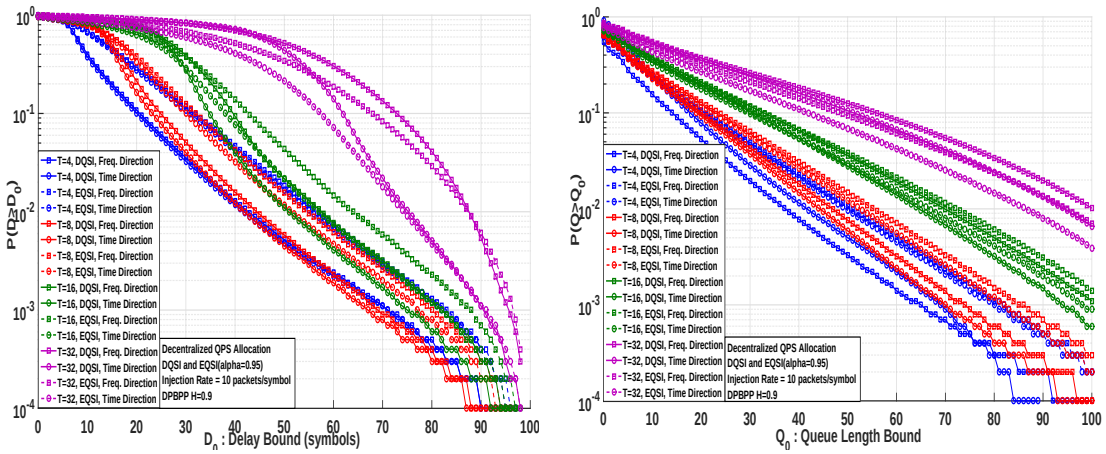


FIGURE 5.35: Average latency curves under non-uniform DPBPP traffic with increasing injection rate for decentralized QPS allocation with DQSI and EQSI algorithm.

frequency direction. In serial allocation, this was the reverse. This is due to the fact that, in serial allocation not all of the RBs are allocated but just the total demand of all tilesets. And the rest was divided evenly among tilesets. Therefore, serving all demands in few symbols by allocating them in frequency direction was more favorable. However, in QPS all of the RBs are always allocated proportional to QSIs of non-idle queues. Hence, allocation in time direction, which allows for a more temporally uniform pattern, is better for QPS algorithm.

Packet Delay and Queue Length Bound Exceeding Probabilities

Following this, we investigate the packet delay bound and queue length exceeding probability graphs for decentralized QPS algorithm with DQSI and EQSI encoding under DPBPP traffic, with 10 packets/symbol injection rate.



(a) (b)

FIGURE 5.36: Packet Delay (a) and Queue Length (b) exceeding probability graphs for decentralized QPS allocation algorithm under non-uniform DPBPP traffic (log-linear)

Among the serial QSI algorithms, best performance was shown by the DQSI encoded allocation with the lowest frame length of 4 symbols. Comparing Fig. 5.23 with Fig. 5.35, we see that QPS allocation with DQSI encoding with 4 symbols frame length provides nearly the same performance. However, for longer frame lenghts, EQSI encoded frame lengths perform better compared to serial allocation algorithm. One may ask, what is the purpose of introducing more computational complexity by employing QPS algorithm, rather than serial allocation, if it provides the same performance for the small frame lengths. However, we have observed that QPS algorithms is also capable to provide better performance. In addition, it provides better performance than serial allocation for delay bound and queue length exceeding probabilities under certain circumstances.

Let us compare serial allocation with DQSI to the secondly proposed QPS algorithm. Under non-uniform DPBPP traffic, for the average latencies we see that best one is DQSI encoded serial allocation with T=4 (both in frequency and time direction) (Fig. 5.19) which shows actually the same performance with DQSI encoded QPS with T=4 symbols. However, we observe difference up to a degree for delay and queue length exceeding probabilities for two different algorithms. For instance, probability of a packet exceeding a delay of 60 symbols is approximately 10^{-3} for DQSI encoded QPS with T=4 (Fig. 5.36(a)), whereas for DQSI encoded serial allocation with T=4 is approximately 10^{-2} (Fig. 5.20(a)). Similarly, exceeding an instantaneous queue length of 90 is approximately 10^{-4} for DQSI encoded QPS with T=4 (Fig. 5.36(b)), whereas for DQSI encoded serial allocation with T=4 is approximately 10^{-3} (Fig. 5.20(b)). We can claim that even though average latency performances are equal, delay and queue length exceeding performance of QPS may be remarkably higher than serial allocation. Besides, apart from scenarios covered in this thesis, QPS algorithm shall provide a more reliable

performance in most of the cases compared to serial allocation, as it proportionally divides the bandwidth. Despite its additional computational complexity, a designer may choose QPS over DQSI encoded serial allocation, where both options may be optimal depending on the situation.

5.2.2.2 Centralized Approach

Average Latency

Following this, centralized version of DQSI and EQSI encoding enhanced QPS algorithm is evaluated. Fig. 5.37, Fig. 5.38 and Fig. 5.39 shows the average latency with increasing injection rate under 3 different traffic models. We observe a very slight but not evident performance degradation for centralization.

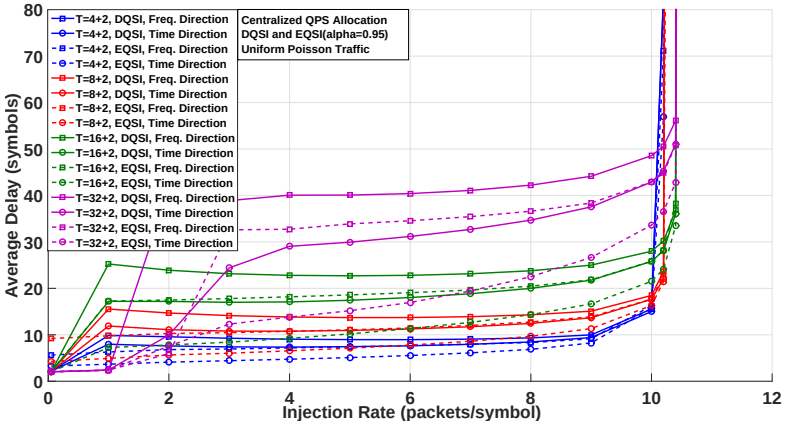


FIGURE 5.37: Average latency curves under uniform Poisson traffic with increasing injection rate for centralized QPS allocation with DQSI and EQSI algorithm.

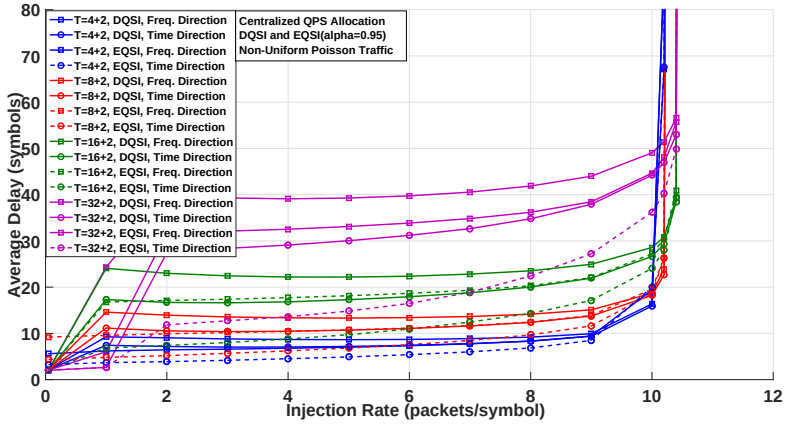


FIGURE 5.38: Average latency curves under non-uniform Poisson traffic with increasing injection rate for centralized QPS allocation with DQSI and EQSI algorithm.

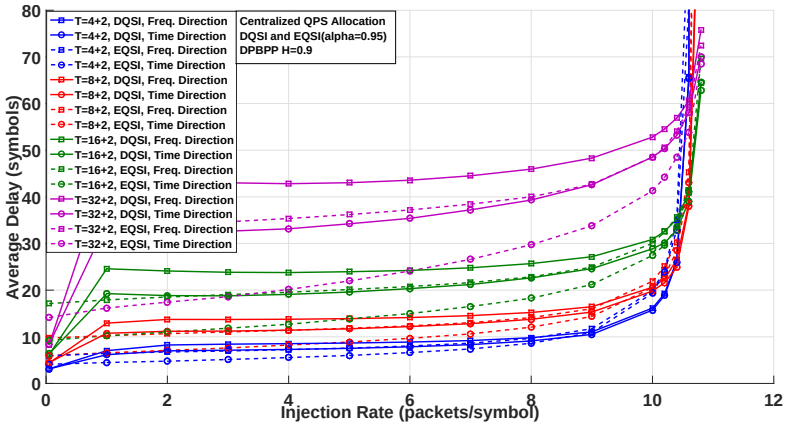
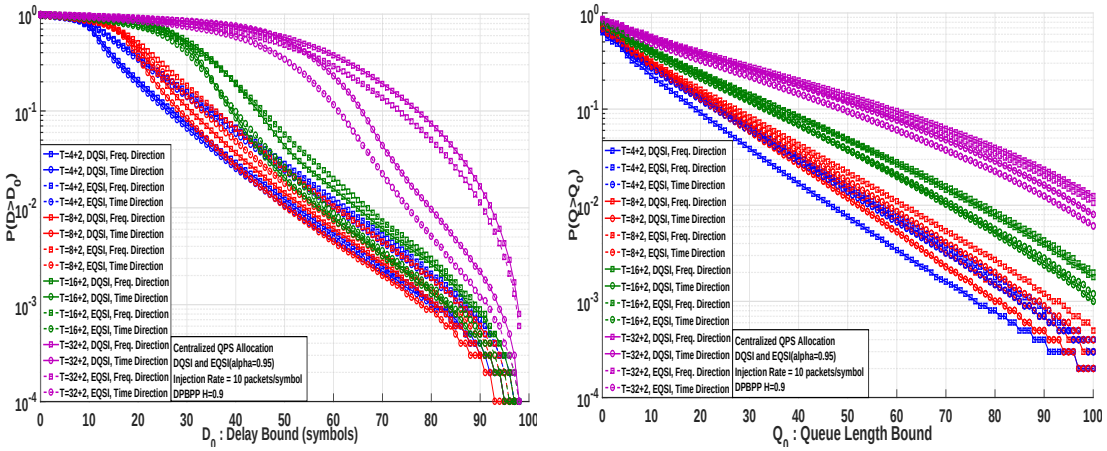


FIGURE 5.39: Average latency curves under non-uniform DPBPP traffic with increasing injection rate for centralized QPS allocation with DQSI and EQSI algorithm.

Packet Delay and Queue Length Bound Exceeding Probabilities

And lastly, packet delay bound and queue length bound exceeding probabilities are shown in following figures.



(a) (b)

FIGURE 5.40: Packet Delay (a) and Queue Length (b) exceeding probability graphs for centralized QPS allocation algorithm with DQSI and EQSI under non-uniform DPBPP traffic (log-linear)

5.3 Implementation of Algorithms

As we have designed RF controller in software executed on a dedicated tile of each tileset, all allocation algorithms can be implemented in a software way. We propose here the algorithm view to be programmed in order to implement our proposals.

Fig. 5.41 depicts the decentralized allocation scheme from a view of a single tileset, for serial or QPS allocation, with regular, definitive or expected QSI encoding. On the first symbol of each frame, a tileset acquires the QSI of other tilesets on the pre-defined subcarriers. If it is a serial allocation, these acquired QSI values are not treated further, and the allocation algorithm starts. However, if QPS is preferred, first the summation of all QSI values is computed. Then the value of the division of total number of RBs, N is divided with this total QSI value and stored temporarily. Using this, proportionally allocated number of RBs for each tileset are calculated, by upper rounding the results. Next, starting from the rotating priority ID tileset, and using the default allocation matrix, RBs are allocated to these calculated values of tileset in a loop, until all the RBs are exhausted, or all tilesets are served. At the end of the frame, tileset reconfigure their transmissions according to this computation, and broadcast their QSIs for next frame. If definitive QSI encoding is chosen; they broadcast the difference between current number of flits in transmission queue and the currently allocated number of RBs (minimal value can be 0). If expected QSI encoding is chosen; they also calculate the moving average value of the mean arrivals by using the number of flits has arrived in last frame; and broadcast the expected QSI values.

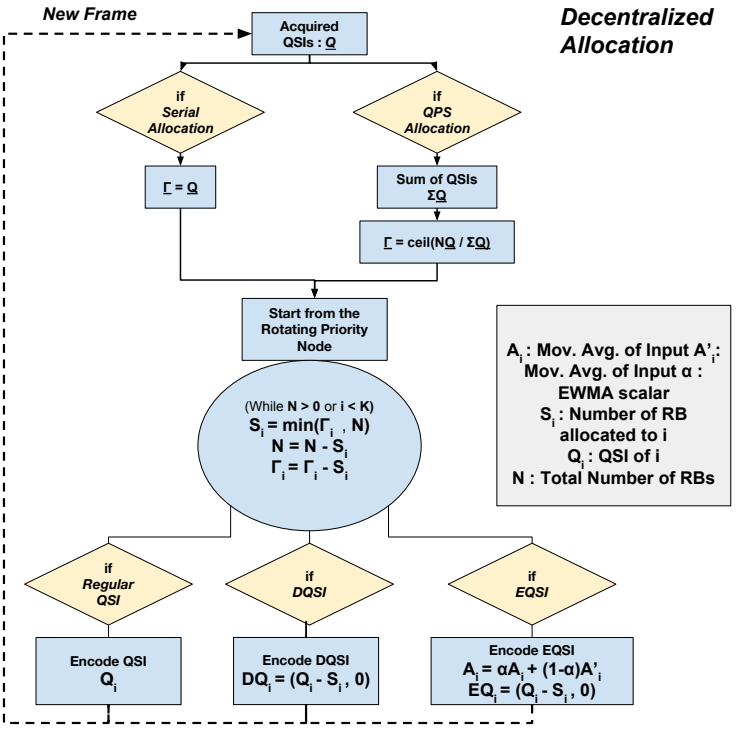


FIGURE 5.41: Flow-chart of decentralized subcarrier arbitration algorithms of serial or QPS with regular, definitive or expected QSI encoding.

Fig. 5.42 depicts a similar flow-chart for the centralized approach from the view of the Central Intelligent Unit. On the first symbol of each frame, the CIU acquires QSIs from

tilesets on the reserved subcarriers. However, this time CIU is responsible of calculating DQSI or EQSI, in contrast the local computation in decentralized approach. Then the allocation is performed and the responses are broadcasted to tilesets.

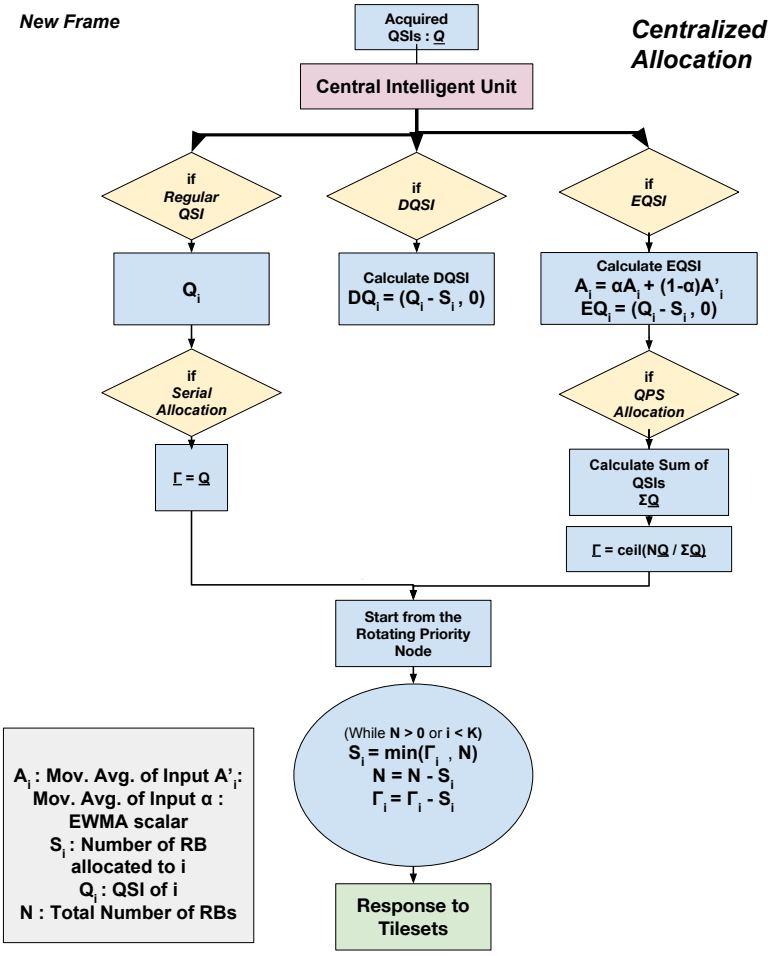


FIGURE 5.42: Flow-chart of centralized subcarrier arbitration algorithms of serial or QPS with regular, definitive or expected QSI encoding.

5.4 Classification of Bandwidth Allocation Algorithms

Fig. 5.43 depicts the classification of mentioned radio resource allocation in WiNoCoD, with a tree diagram. It gives a global view of the proposed algorithms in the right side, and also the possible static approaches as mentioned in Section 4.3 on the left side. Note that, Chapter 5 only dealt with a single transmission queue where both long and short packets are treated in common flits. In Chapter 6, we will present a new dual queue approach in order to improve the previous algorithms, where a seperate queue buffers payloads of long packets. This type of algorithms are shown in gray in Fig. 5.43.

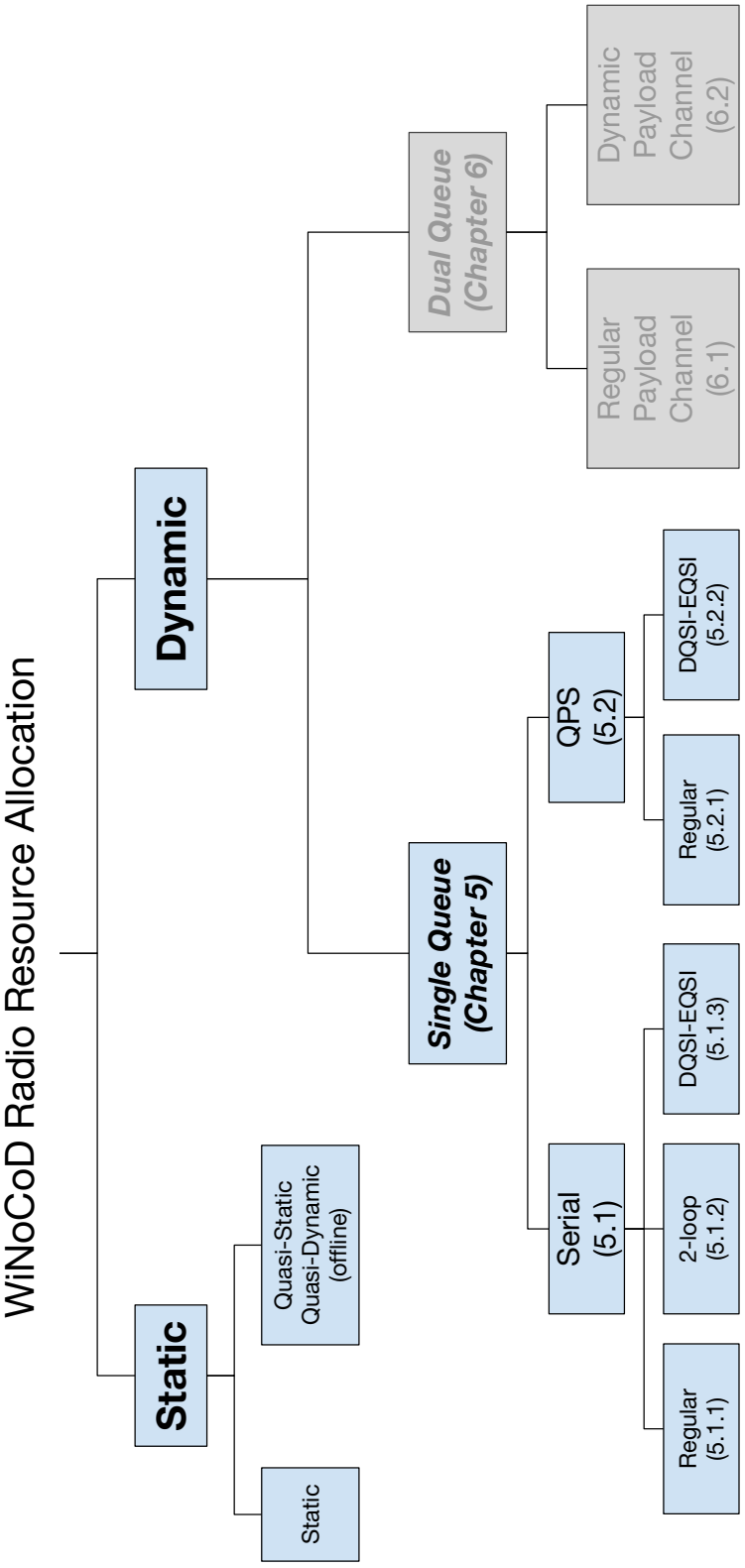


FIGURE 5.43: Classification of the proposed algorithms for dynamic bandwidth allocation for OFDMA RF NoC.

5.5 Conclusion

The primary criterion on the development of bandwidth allocation algorithms for WiNoCoD RF interconnect is low computational complexity. This obligation stems from the very short OFDM symbol duration of 50 ns, where bandwidth arbitration shall be effectuated in few symbol durations. Considering the processing speeds of current semiconductor technology, this time constraint imposes that the proposed allocation algorithm should be executed in few hundreds of ns. Therefore, proposed solutions should avoid large number of iterations and complex operations, but should be implementable with basic mathematical processor operators or Look-up Tables (LUTs). Considering this algorithm execution time and also other delays regarding reconfiguration, a pipelined allocation scheme is adopted for WiNoCoD. However, the latency performance of the algorithms are at the paramount of importance, even though OFDM symbols are short, ironically they are long for the on-chip packets. Hence, saving average delay by few symbols may increase the on-chip performance drastically.

For this purpose, first proposed algorithm was serial QSI allocation algorithm, which practically needs any mathematical operations and lowest number of iterations. However, we have discussed the starvation issue due to certain nodes acquiring all RBs, thus causing an early system fail, especially for short frame lengths. We have tried to mitigate this effect by employing a 2-loop queue balancing mechanism, but the best results for serial allocation algorithm is taken with DQSI encoding. This basic operation avoids the redundant allocation of RBs, and makes sure for a fairer arbitration of the bandwidth.

Next, we introduce the QPS algorithm, which divides bandwidth among tilesets proportional to their instantaneous QSIs. First thing we observe for the performance of this algorithm is the relatively high average latency for small injection rates. We discuss that this was due to the fact that, when a QSI is 0 for a tileset, it is allocated 0 RBs on that frame, and any newly arrived packet has to wait for the next frame. By employing EQSI and DQSI mechanisms, we are able to mitigate this high average latency under low injection rate.

DQSI encoded serial QSI allocation provides really good performance, and needs practically the lowest computational complexity. This mechanism is feasible to be implemented with short 4-8 symbols frame lengths. Apart from scenarios covered in this thesis, QPS algorithm shall provide a more reliable performance in most of the cases compared to serial allocation, as it proportionally divides the bandwidth. Despite its additional computational complexity, a designer may choose QPS over DQSI encoded serial allocation, where both options may be optimal depending on the situation.

Chapter 6

Payload Channel Algorithm

In previous chapter, we have proposed a framework for dynamic bandwidth allocation among tilesets, concerning on-chip time constraints and investigated certain algorithms for arbitrating bandwidth for minimizing transmission latency. However, these algorithms do not take into account the lengths of packets and just consider the instantaneous total number of flits in the queue. We have noticed that bimodal packet lengths of on-chip cache coherency packets can be exploited in order to further decrease transmission latencies in certain circumstances. Especially, when traffic load is low, so that most of the nodes do not transmit on their allocated bandwidth and cache-lines carrying packets are considerably long, these packets can be transmitted instantly by using under-utilized channels. Based on this principals, we build a new context for exploitation of bimodal packet lengths, taking into account the coordination among tilesets. These new type of algorithms can perform better compared to algorithms proposed in Chapter 5, and also they can be combined together to increase dynamicity further.

6.1 Context

On-chip cache coherency traffic has an unorthodox nature compared to conventional networks due to its very specific features. As mentioned previously, in Section 2.4, one of these features is *bimodal packet lengths*. We are using a *Distributed Hybrid Directory Based Cache Coherency*, where its detailed description is given in Section 3.1.3. This viable coherence mechanism for 1000-core shared memory architectures imposes high number of broadcast messages, therefore a traffic burden on the wired RF interconnect, which is a ransom to pay for scalability. There are two types of cache coherency packets circulating in a network-on-chip, which are short control packets (i.e. request to read an

address line, acknowledgement etc.) and cache-line carrying long packets (i.e. request to write to an address line, response for reading an address line etc.).

In the scope of the project, a short packet length of 64 bits is chosen as feasible. Long packets are composed of a control header, which contain the necessary information such as destination/source ID, packet type etc. and the data in cache line. Cache line size is determined as 64 bytes (512 bits), therefore making long packets 576 bits long. Considering the default QPSK modulation, in case of simple static and uniform allocation of subcarriers (where each tileset acquires 1 RB- i.e. 64 bits), a long packet would take 9 symbols to transmit even under zero queuing delay. Considering 1 OFDM symbol is longer than 50 ns (50 clock cycles if a 1 GHz core frequency is assumed), one can understand the bottleneck long packets create for the on-chip interconnection in terms of transmission latency. Even the basic mathematical and logical operations take several cycles. Considering also into account the OFDM signal reception, decoding, reconfiguration etc., one can see that bandwidth allocation needs multiple symbols to be processed.

Cache line (block) size has a significant impact on the performance of the shared memory system. Not just large cache size but larger cache lines decrease the cache miss rate generally (the case, a core cannot find referenced copy of required datum in its own L1 cache.) [120]. One of the reasons behind is the spatial locality of the information distribution, so that a fetched larger cache line upon a miss will include more data close to the referenced address, which are likely to be referenced soon. Obviously, there exists a trade-off between the miss rate and the bandwidth overhead, as longer cache line means larger data to traverse through network-on-chip in case of cache miss. Another important point about the performance of chosen cache line size is its mutual dependency on cache size also. For instance, inside a relatively smaller cache, very long cache lines even increase the miss rate [120]. The reason behind this phenomenon is the limited size of data structure of the program. However, this is not the case for every CMPs, where they can exploit spatial locality more effectively. In [47], authors had examined the effect of larger cache lines on miss rates for a 8-core CMP and different application from PARSEC benchmark. Certain applications had shown much lower miss rates compared to others with 256 bytes of cache lines, as they are more relying on spatial locality of the data. We can predict that a better performance can be gained via larger cache lines for future shared memory 1000-core architectures and optimized application running on them. Also in case of very large caches are utilized, larger cache lines would be essential. In addition, a reconfigurable, effective low latency interconnect would amortize the penalty of carrying large cache lines, increasing system performance further. In [121], it was shown that the benefit of long cache lines as much as 256 bytes is more apparent on

shared memory massive multicore architectures, especially for applications with higher locality.

Taking into account all of these aspects mentioned above, we have developed a novel bandwidth allocation algorithm for WiNoCoD's OFDMA based RF interconnect, which decreases the latency of long packets substantially by transmitting payloads (cache line) in a single symbol. It exploits the intrinsic broadcast capability and elevated reconfigurability of OFDMA. This algorithm is designed especially for large cache lines which may be preferred by the programmer based on the characteristics of the application(s). Different than the used 64 bytes (512 bits) long cache lines throughout the project and the thesis, this algorithm is optimized especially for 256 bytes (2048 bits) long cache lines. In this case, the static algorithm would have to transmit long packets in 33 symbols even under zero queuing (more than 1650 ns) which is highly prohibitive. We also fuse the bandwidth allocation algorithms mentioned in Chapter 5 with this packet size aware policy, in order to increase its performance further under heterogeneous traffic.

6.2 Regular Payload Channel Algorithm

6.2.1 Description of Regular Payload Channel Algorithm

The proposed algorithm relies on the basic idea of transmitting 2048-bits payloads of long packets in a single OFDM symbol by using all 1024 subcarriers modulated with QPSK. The 64-bit single flit control packets (which corresponds to 70-80% of all packets) and 64-bit header flit of long packets contain the necessary source/destination ID, corresponding address line etc. information, but also the flag bits marking the type of packet (whether it is a short or long packet). Our novel Payload Channel Algorithm exploits these flag bits to allow transmission of long packet payloads in a single OFDM symbol without using any extra signaling overhead.

In this proposal, initially, each tileset is allocated 32 subcarriers, where we refer them as *home channels*, with a terminology used in literature as in static case of Section 4.3.1. As QPSK modulation is utilized, these home channels can serve 1 flit per each symbol, either a short control packet or a long packet header.

Different than the previously presented architecture, for Payload Channel Algorithm, each tileset has two different transmission queues at their RF interfaces. Primary transmission queue buffers short packets or headers of long packets only in a FIFO manner. Whenever a new long packet arrives to RF interface from the interior mesh network, it is segmented into its header (which is buffered in the primary short queue) and its payload

(which is buffered in the secondary payload queue in a FIFO manner). This 2-queue architecture is depicted in Fig. 6.1. Motivation behind employing two different queues for headers/short packets and payloads is to avoid the inconsistency between transmitted headers and their payloads and also to allow transmission of a new packet before transmitting a payload of a previous packet. This notion will be explained further in detail within next paragraphs. The idea is to give priority to long packets' payloads, so that they can be transmitted fast, by using other tilesets' allocated channels, especially if they are under-occupied.

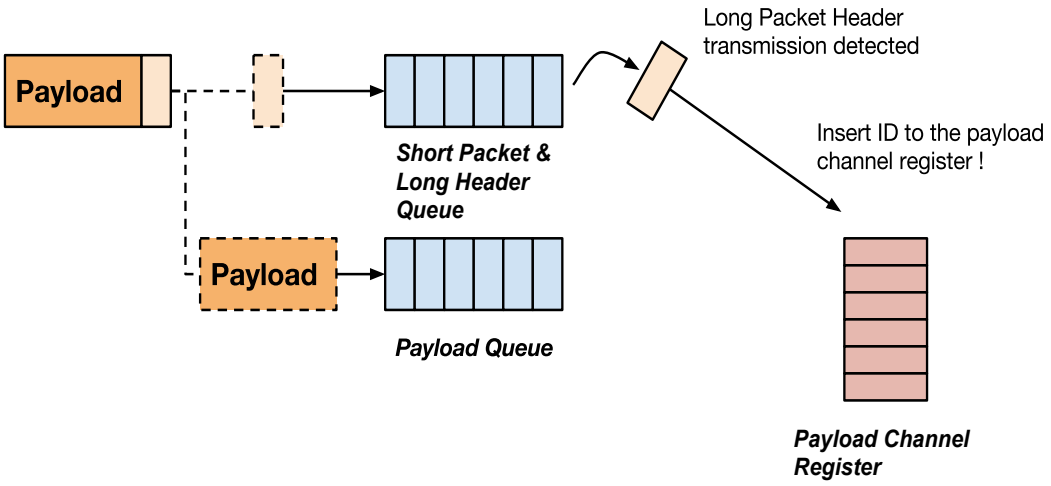


FIGURE 6.1: FIFO "Short Queue" for 1-flit short packets and long packet headers and FIFO "Long Queue" for payloads at the front-end of each tileset. FIFO short queue is sent on the RF-NoC channel until a long packet header is detected from any tileset including itself. Then ID of the tileset is inserted to Payload Channel Register. Note that, each tileset keeps the same payload channel register content.

When a tileset transmits a long packet header in its home channel, each tileset can acquire and decode this thanks to intrinsic broadcast capability of the OFDMA interconnect. Each tileset can use simple packet processing units to check the flag bits to understand whether this is a long packet or not. Upon tilesets detect a long packet header, they record the corresponding tileset-ID by inserting the ID in the *payload register*. Payload register is a simple FIFO queue at RF interface of each tileset, which buffers IDs of tilesets to transmit payloads. Note that, content of the payload registers is identical in all tilesets, in order to avoid incoherence.

However, acquiring the header flit, processing it and reconfiguring the bandwidth takes certain amount of time due to propagation, synchronization and computation delays. Therefore, we have determined a 1 symbol latency (50 ns) for all delays and activation of algorithm. Also note that, multiple long packet headers can be detected in a symbol from

different tilesets, that their IDs are enregistered to payload registers by their Tileset-No order in the same symbol.

At the start of each symbol, tilesets control their payload registers. If the payload register is empty, this means there is no tileset currently wanting to transmit a payload. Therefore, home channel configuration is applied and each tileset can use it 64-bit home channel. However, if payload register is not empty, first ID in the register transmits its payload. If it not the tileset's own ID, it does not use its home channel in this symbol to allow the transmission of payload. System returns to home channel configuration only if there is no tileset remains in the payload register. Fig. 6.2 and Fig. 6.3 illustrate the flow-charts of the payload channel algorithm from the view of a transmitter and receiver side of a tileset, respectively.

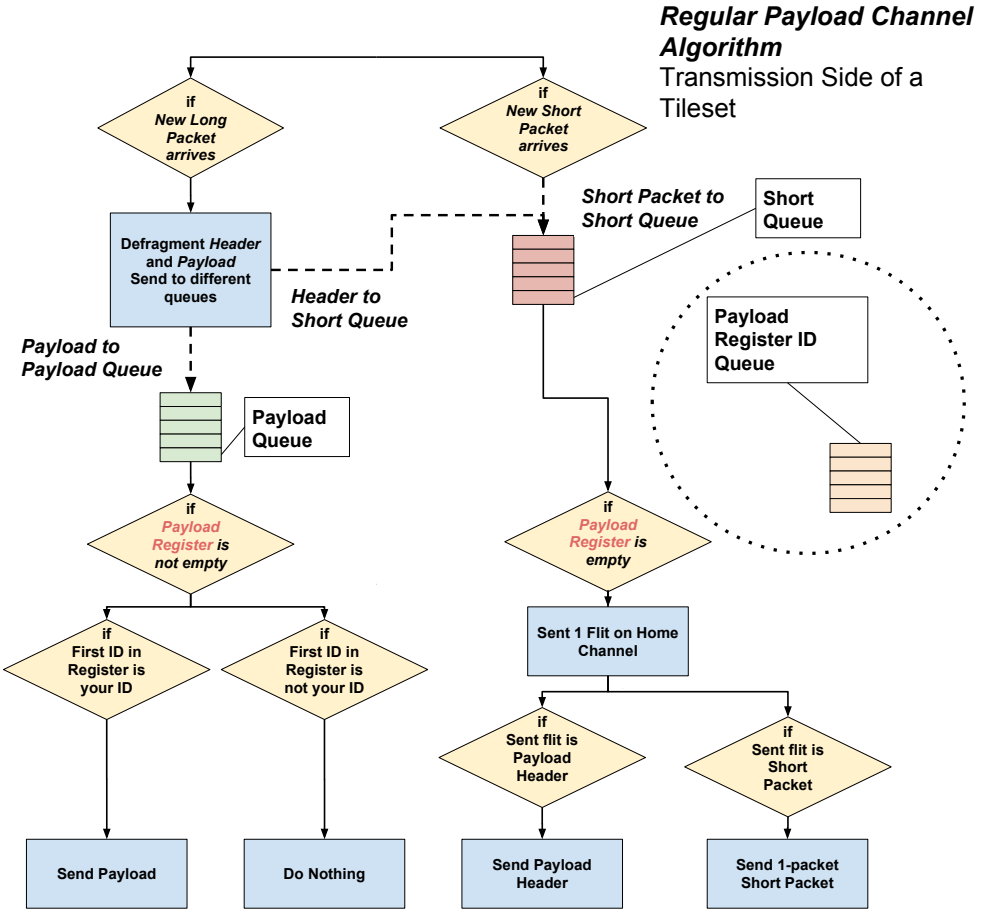


FIGURE 6.2: Flow-chart of regular payload channel algorithm from the view of the transmission side of a tileset.

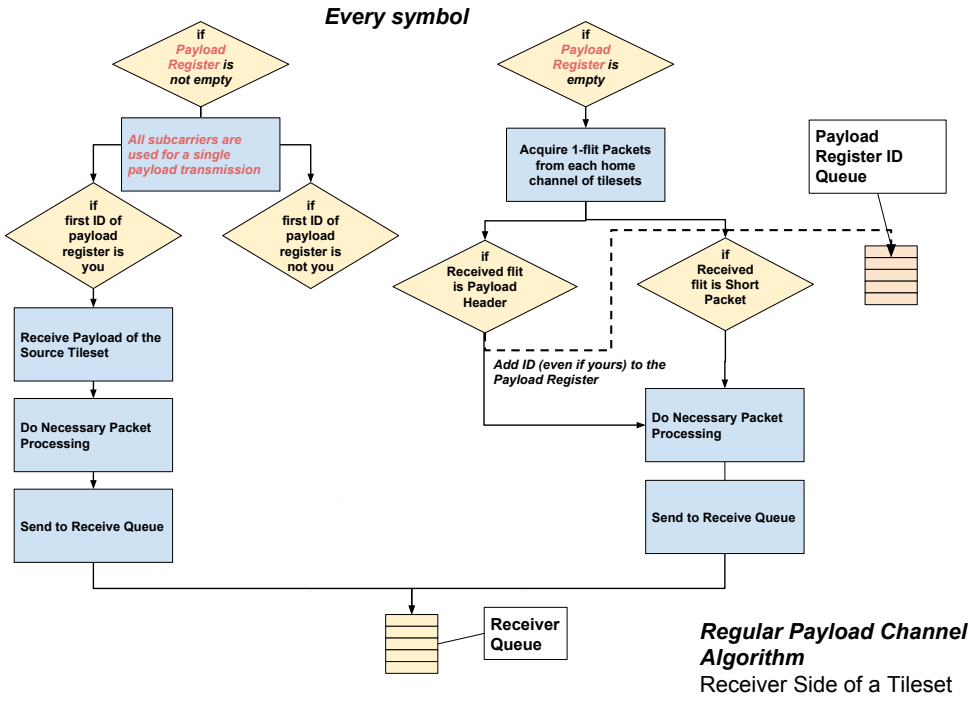


FIGURE 6.3: Flow-chart of regular payload channel algorithm from the view of the reception side of a tileset.

6.2.2 An Illustrative Scenario

These procedures are shown in Fig. 6.4 with a scenario. In order to explain illustratively, in this example only Tileset-1,2 and 3 are active (rest of the nodes do not transmit any packets). On symbol t=0, Tileset-2 and Tileset-3 transmits a long packet header and Tileset-1 completes the transmission of a short packet. As it takes 1 symbol long latency to receive and process long packet headers, they are registered on symbol t=2 (Note that, this 1 symbol latency includes all delays concerning reception, propagation, synchronization, processing but also enregistering to payload register and reconfiguration of transmission just before the next symbol. In this graphic, for simplicity purposes, register contents are illustrated just at the same time with current symbol).

As Tileset-2 and Tileset-3 had sent long packet headers, they are enregistered in the payload register with respect to their Tileset-No on the same symbol. Note that, tilesets are still in home channel configuration on symbol t=1, as long packet headers are not processed and Payload Channel Algorithm is not activated, yet. Even though, Tileset-2 has sent a long packet header in previous symbol (and not yet sent the rest of it, i.e. the payload), it can transmit a short packet on symbol t=1. This is possible, thanks to the separate queues for short packets/headers and payloads. On symbol t=2, it can just

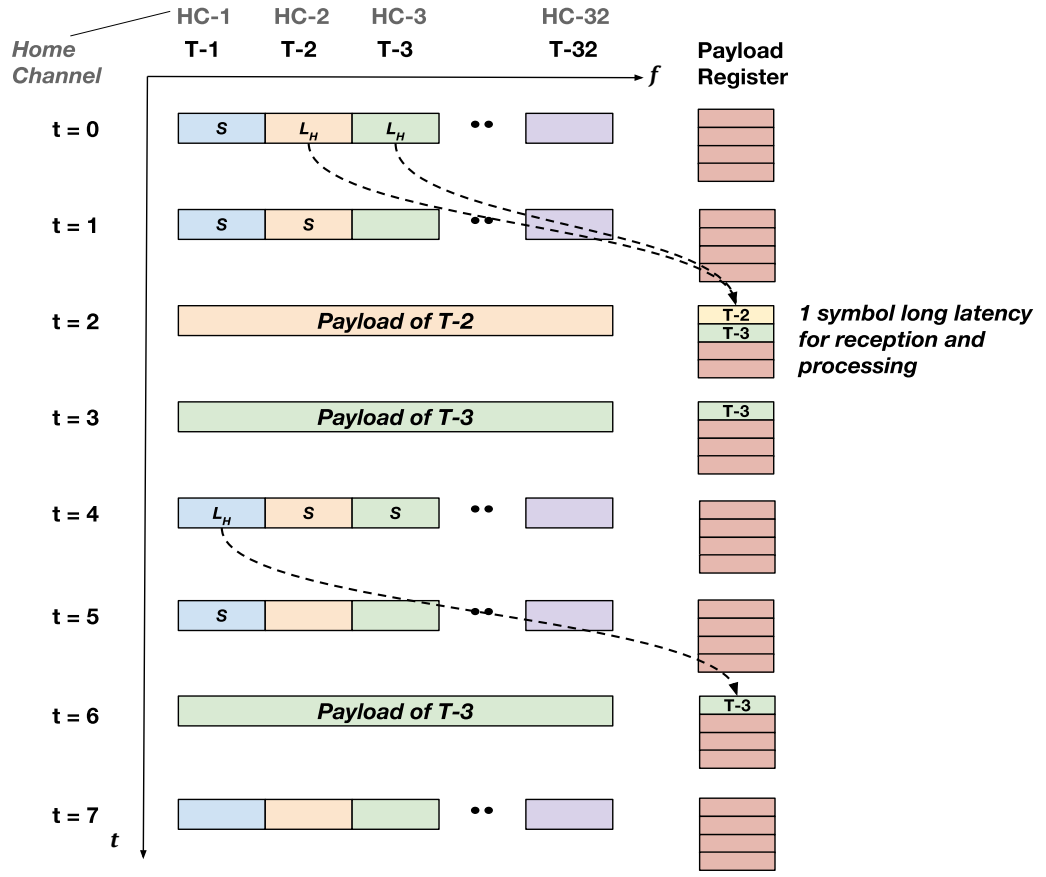


FIGURE 6.4: Illustration of our Payload Channel Algorithm both in frequency and time domain through a simple scenario, where only 3 tilesets are active.

transmit the payload of the previous packet and at reception, other tilesets can process and perform the necessary desegmentation without inconsistency.

After Tileset-2's payload is transmitted on symbol t=2, it is deleted from the payload register. On next symbol t=3, Tileset-3 is transmitted. As there is no ID pending left at the payload register, on symbol t=4, system returns to home channel configuration, where tileset can again transmit headers and short control packets. Remark that, they had to stall their transmissions until the transmission of payloads is terminated.

This algorithm is decentralized in its nature and only reconfigurable in terms of payload transmission. One can argue that this algorithm does not provide enough home channels to tilesets with fluctuating bandwidth demands due to heterogeneous on-chip traffic. We will present a "more dynamic" version of our algorithm in next section, however first, we present experimental results for Regular Payload Channel Algorithm under spatially uniform traffic scenarios (where injection rate of tilesets are equal).

6.2.3 Analytic Approximation for Average Latency Calculation

Before we present the simulation results for average latency induced by our algorithm, we investigate the possibility of deriving a closed form equation for the average latency under specific scenarios. We revisit certain aspects of *Queuing Theory* and develop reasonable approximations.

First of all, we build a queuing theoretical model of our system as in Fig. 6.5. Assuming a uniform Poisson traffic, any incoming 1-flit packet (long packet header or short packet) to short queue of a tileset obeys a Poisson process. The service time of a short queue is 1 symbol when the payload channel register is empty (home channel configuration). However, for other times service time for this queue is equal to number of tilesets waiting in the payload channel plus 1 symbols, as its transmission is stalled until all current waiting payloads are served. Hence, we can model the short queues at tilesets' front-end as M/G/1 queues based on Kendall's notation (one can refer to [122] for the basics of queuing theory and Kendall's notation based queue models).

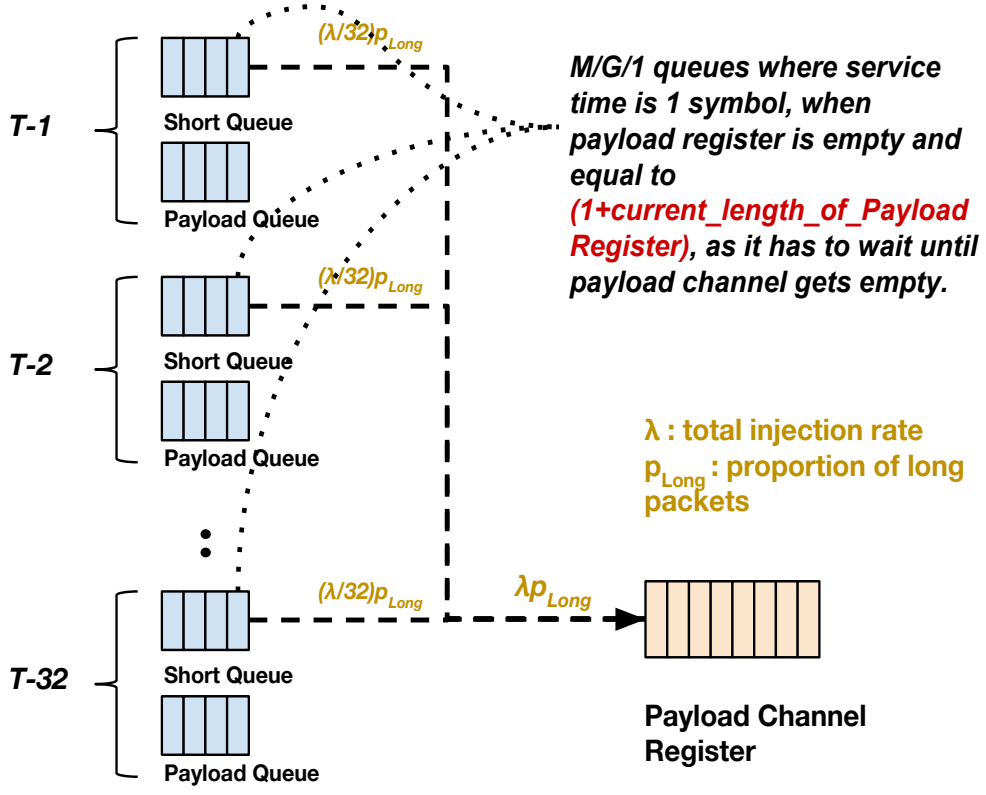


FIGURE 6.5: Modeling of Payload Channel Algorithm as a 33 M/G/1 queuing network

Next, we try to derive a queuing model for the payload channel register. We can visualize a Payload Channel Register is serving 1 payload per symbol by using all of the subcarriers, whenever there is a tileset ID in the register. Therefore, service time in payload channel register is "deterministic" and equal to 1 symbol. Modeling the arrival process to payload register is more complicated. It acquires the long packet headers departing from 32 tileset short queues (M/G/1 queues). Deriving a reasonable departure process model from an M/G/1 queue is not straightforward and requires very complex Markov Process analysis. We do know that departure process from an M/M/1 queue (inter-arrival and service times are exponentially distributed) is a Poisson process [123]. An interesting article, [124] investigates the characteristics of departure process from an M/G/1 queue with heavy tailed service time distribution and finds that departure process is heavy tailed with the same Hurst parameter.

As we wish to model the all arrivals to the payload channel register (as an aggregation of 32 tilesets' outputs), rather than individual departure processes, we derive an approximation for it. We know that Poisson distribution is derived as an infinite approximation of binomial for very large number of elements with very small Bernoulli probability [125]. In order to explain in detail, when we have N random values, each independently and identically distributed (i.i.d.) and can be 1 with probability p or 0 otherwise. When N is very large (going to infinite) and p is very small (going to zero), the probability distribution of the number of 1's at any time is a Poisson distribution. Our case is similar to this scenario for long packet header arrivals to payload channel, as we have 32 random variables, each can be '1' (i.e. long packet header is transmitted) with a certain probability. Although either N is not very large (32) or p is very small (probability of having a long packet header), we think modeling the arrivals to the payload channel register as a Poisson Process seems highly reasonable. Therefore, at the end, we have modeled the Payload Channel Register as an M/D/1 queue. We may guess that this approximation would be much valid when injection rate is low, lowering p.

Firstly, we derive the service time probability distribution for the 1-flit packets (either long packet headers or short control packets) in the short queues at tilesets' front-ends, S_{short}. As noted previously, service time for short packets is equal to $1+L_{payload}$ (current number of elements waiting in payload register). We have mentioned that payload register is modeled as an M/D/1 queue and we have to have its Queue Length distribution in order to derive service time distribution of short packets. This distribution was derived for the special case where service time is equal to 1 unit, which is exactly the same case for the payload register [126]. By adding 1 symbol to these equations in [126], as service time in a short queue is 1 plus the number of elements waiting in the register, we can rewrite the distribution for the service time of short packets as :

$$p(S_{short}) = \begin{cases} 1 - \lambda_{payload} & \text{if } S_{short} = 1 \\ (1 - \lambda_{payload})(e^{\lambda_{payload}} - 1) & \text{if } S_{short} = 2 \\ \dots & \dots \\ (1 - \lambda_{payload})\Bigg(e^{(n-1)\lambda_{payload}} + \sum_{k=1}^{n-2} e^{k\lambda_{payload}}(-1)^{(n-k-1)} & \\ \left[\frac{(k\lambda)^{(n-k-1)}}{(n-k-1)!} + \frac{(k\lambda)^{(n-k-2)}}{(n-k-2)!}\right]\Bigg) & \text{if } S_{short} = n \end{cases}$$

where $\lambda_{payload}$ is the mean arrival rate (i.e. injection rate) to the payload register. Note that, $\lambda_{payload} = \lambda p_{long}$, i.e. total injection rate multiplied by the proportion of long packets in the system. Here, for instance, for a 1-flit packet to be served in the short queue in 1 symbol (either a header of a long packet or a 1-flit short packet), the payload queue has to be empty when the packet has arrived. In other words there should not be any long packet transmission, so that the home channels can be used and a short packet can be served in 1 symbol. This probability equals to $1 - \lambda_{payload}$, as this is also equal to the probability of payload queue to be empty. As service time for a short packet, S_{short} increases, the associated probability, $p(S_{short}) =$, increases with exponentially (and oscillatory due to Taylor's series expansion) with service time and average input to the payload queue.

We can derive the mean of service time of short packets from this analytical distribution as :

$$\mu_{short} = \sum_{i=1}^{\infty} ip(S_{short} = i) \tag{6.1}$$

and using (6.1) and taken M/D/1 distribution, we can also derive the variance as :

$$Var(S_{short}) = \sum_{i=1}^{\infty} (i - \mu_{short})^2 p(S_{short} = i) \tag{6.2}$$

Having the probability of service time distribution of short packets (either long packet headers or 1-flit control packets), we can continue to derive the average queuing plus service time in M/G/1 modeled short queues in tilesets' interface using Pollaczek-Khinchine Formula [127] :

$$W_{short} = \frac{\rho_{short} + \lambda_{short}\mu_{short}Var(S_{short})}{2(\mu_{short} - \lambda_{short})} + \frac{1}{\mu_{short}} \tag{6.3}$$

where μ_{short} (mean service rate) is the reciprocal of mean service time of short packets which can be derived by taking the expectation over the service time distribution given previously. Similarly variance of the service time can also be calculated from this distribution. ρ_{short} signifies the utilization in these short queues which is equal to $\lambda_{short}/\mu_{short}$. λ_{short} is simply the average injection rate to each of the short queues which is total injection rate divided by the number of tilesets, as both long packet headers and short control packets arrive to these queues.

In parallel, as long packets are composed of a header firstly served at short queue and a payload served by the payload register queue model; average queuing plus service time of a long packet can be calculated as the sum of the average queuing plus service time of a short packet and payload : $W_{long} = W_{short} + W_{payload}$.

We have stated previously that payload register queue is simply an M/D/1 model where D=1. Thus, its average queuing plus service time (average delay) can easily be calculated from the Pollaczek-Khinchine Formula :

$$W_{payload} = \frac{2 - \lambda_{payload}}{2(1 - \lambda_{payload})} \tag{6.4}$$

At the end, we can write the approximated average delay in the system as the proportional sum of long and short packets as a function of total injection :

$$W = (1 - p_{long})W_{short} + p_{long}W_{long} \tag{6.5}$$

Due to the Taylor series expansion based derivation of the stationary distribution of M/D/1 queue, we cannot write down the resulting average latency formula in closed from here, as it requires too much space. The analytic approximation is calculated by Matlab using loops for series. Note that, in definition these series expansions go to infinity in alternating directions. Using too large expansions may cause instabilities, therefore we have calculated these values up to several hundreds. In addition, probability of having too large values for the service of short packets, such as larger than 100 symbols, is too small.

6.2.4 Experimental Results for Payload Channel Algorithm

We test the performance of our payload channel algorithm by comparing it to the scenario where bandwidth is allocated statically and equal among tilesets. Understandably, this version of the payload channel algorithm is only tested for the uniform Poisson and uniform burst traffic models.

Average Latency

Fig. 6.6 shows the average latency with increasing total injection rate under uniform Poisson traffic, along with the our theoretically derived approximation. Note that, thanks to its effective reconfiguration to transmit payloads in single symbols it can provide up to 10 times lesser average latency, compared to static case. Also note that, our queuing theory approximation well fits to resulting average delay, except for large injection rates close to system limit. This shall root from the success of Poisson distribution with low injection rate, as previously explained. Results provide a slightly lower injection rate limit compared to theoretical one, which we can speculate that it is due to higher variance of service and arrival rates compared to Poisson distribution (higher variance increases average latency [127]). Also, as we are considering 256 bytes long payloads for long packets now, the average length of a packet is 8.75 flits, which explains the new system limit for static allocation Fig. 6.6, injection rate approximately equal to $32/8.75 = 3.6$ packets/symbol.

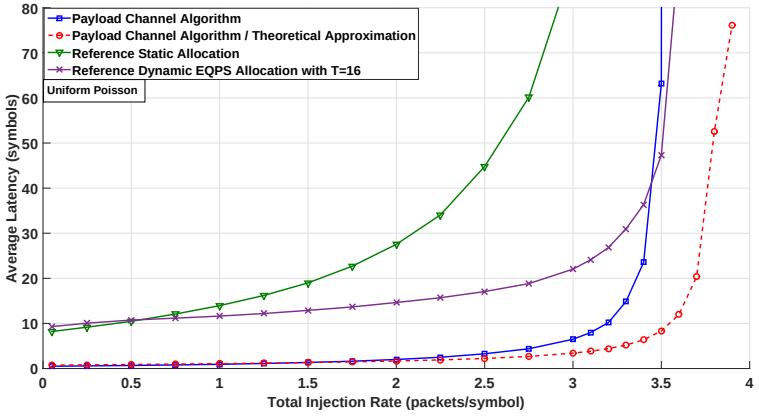


FIGURE 6.6: Average latency under uniform Poisson traffic with increasing injection rate for Payload Channel Algorithm, analytic approximation and the reference static case.

Fig. 6.7 shows the same performance measure for our algorithm and reference static and dynamic QPS allocation under uniform DPBPP traffic with H=0.9 presenting a much higher temporal heterogeneity. Note that, even under this much of temporal burstiness, our algorithm is still able to induce up to x10 less average latency, thanks to its separate

payload queues and novel subcarrier reconfiguration without requiring any signaling overhead.

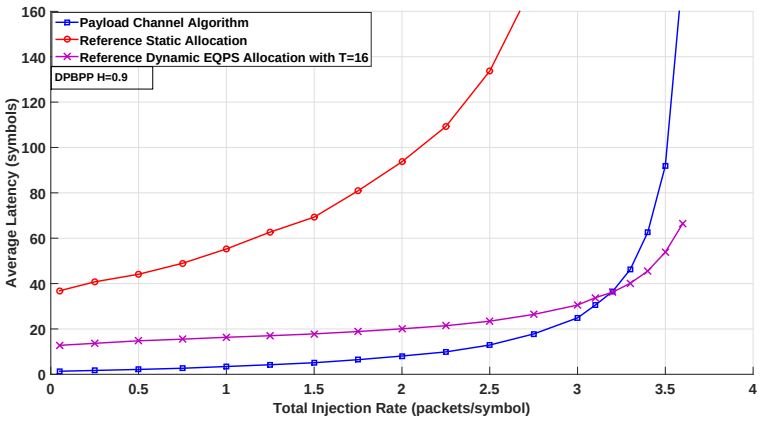


FIGURE 6.7: Average latency under uniform DPBPP traffic (H=0.9) with increasing injection rate for Payload Channel Algorithm and the reference static case.

Packet Delay and Queue Length Bound Exceeding Probabilities

Next, we investigate the probability of exceeding given delays for packets for our payload channel algorithm and the reference static algorithm.

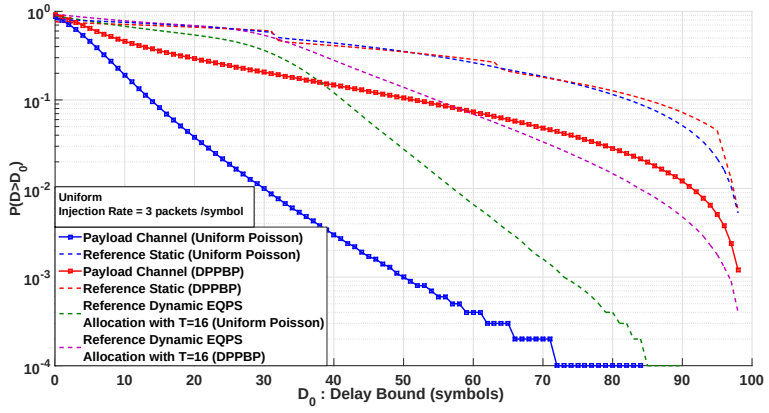


FIGURE 6.8: Delay exceeding probability graphs for our payload channel algorithm under uniform Poisson and DPPBP (H=0.9) traffic

Fig. 6.8 shows the delay exceeding probabilities under 2 uniform policies, highlighting the absolute performance gain of our payload channel algorithm. For instance, under uniform Poisson traffic, the probability of a packet to have a delay larger than 30 symbols is x100 lesser compared to static allocation. Under uniform DPBPP traffic, with introduced burstiness our algorithm's performance degrades significantly, but it still provides approximately x5 times lesser probability to exceed a delay of 30 symbols.

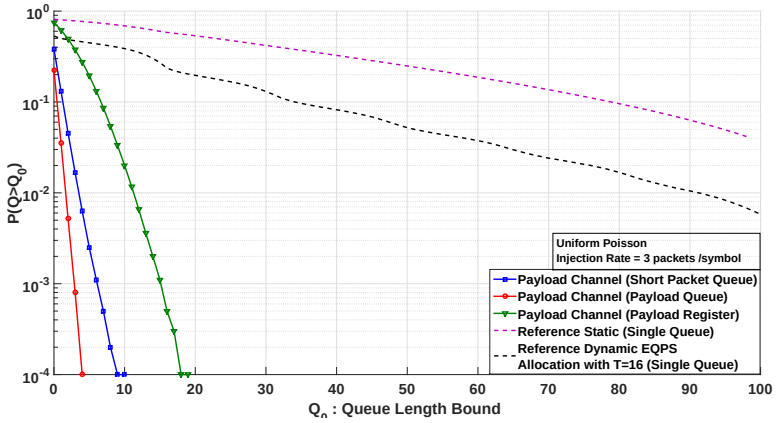


FIGURE 6.9: Queue Length exceeding probability graphs for our payload channel algorithm under uniform Poisson traffic

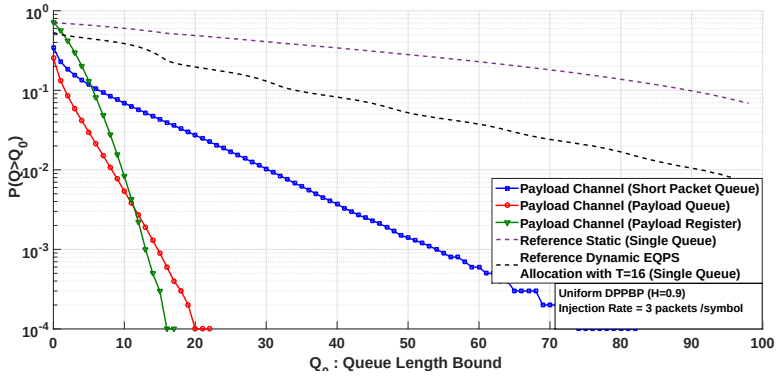


FIGURE 6.10: Queue Length exceeding probability graphs for our payload channel algorithm under uniform DPBPP traffic (H=0.9)

And finally the queue exceeding probabilities are compared. Payload channel algorithm requires 3 separate queues to be install at each tileset's front-end : a short queue for control packets and long packet headers, a payload queue for 32-flit long payloads and a payload register to keep the ID of the tilesets dynamically, demanding to use payload channel (which stores the copy of the same information all time at each tileset). And conventional static allocation requires only a single queue. Note that, given queue lengths are in terms of units, that for payload queue it is the number of payloads (means one has to multiply it by 32 in order to calculate the capacity in terms of flits), for payload register it is the current number of tileset IDs (as there are 32 tilesets we need 5 bits storage to represent each).

Fig. 6.9 and Fig. 6.10 shows the queue length exceeding probabilities under uniform Poisson and DPBPP traffic, respectively. Let us remark that, in Fig. 6.9 and Fig. 6.10, the x-axis does not represent the number of flits, particularly. For payload queue, x-axis represents the number of payloads of 32 flits long (cache line); for short packet queue, it

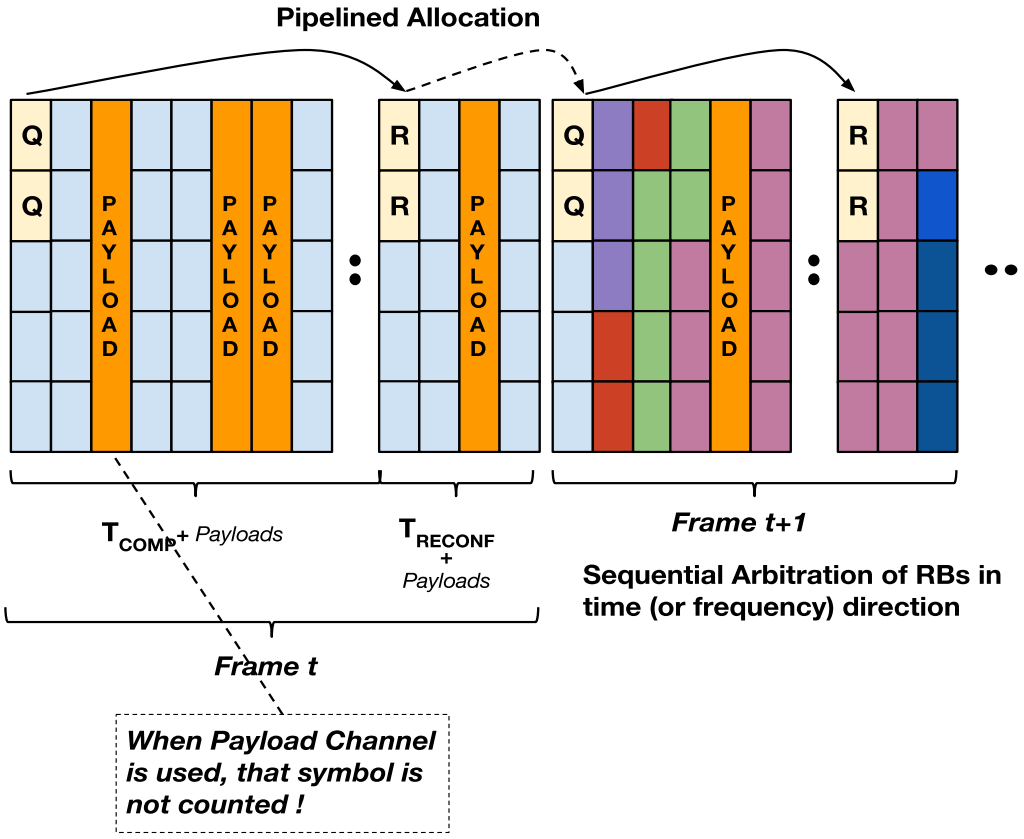


FIGURE 6.11: Frame structure and centralized bandwidth allocation mechanism of the proposed dynamic payload channel algorithm

represents the number of 1-flit long short packets; for payload register, it represents the number of IDs in a payload register at a time (for 32 tilesets, it is 5-bit long number) and for reference algorithm, it represents the number of flits in a transmission queue as in Chapter 5. We have combined these different indicators in a single graph in order to provide a conceptual coherence. One can dimension and compare the required memory for each of these 3 different queues for payload channel algorithm by looking at these probabilities of exceeding. For instance, in Fig. 6.10, with a probability of 10^{-2}, for the payload channel algorithm, the number of payloads (where each is 32 flits long) in the payload queue exceeds 8 (256 flits), for the short packet queue (where each is 1 flit), the number of packets exceeds 30 (30 flits) and finally the instantaneous number of tileset-IDs waiting (where each is 5 bits for 32 tilesets) exceeds 10 with this probability. For the same probability of 10^{-2}, for instance reference dynamic allocation algorithm with a single queue, the instantaneous number of flits can only exceed 90, where for payload channel algorithm total required capacity is 286 flits. As you can see even though, the proposed payload channel algorithm can provide less average latency compared to reference, it may require higher buffer capacity, as we need to store 32 flits long payload

for relatively longer durations.

One can see the reasonably high performance of our algorithm in terms of requiring small buffer lengths, even under the spikes of bursty traffic. Most importantly, payload queue, where each element is composed of 2048 bits, does not require prohibitively large capacity, thanks to effectiveness of our algorithm.

6.3 Dynamic Payload Channel Algorithm

6.3.1 Description of Dynamic Payload Channel Algorithm

Spatial imbalance of cache coherence traffic was discussed intensively through the thesis manuscript. With additional temporal burstiness, our payload channel algorithm needs a modification in order to cope with this situation. For this purpose, we develop Dynamic Payload Channel Algorithm by merging previously mentioned payload channel algorithm with centralized queue length proportional (QPS) algorithm.

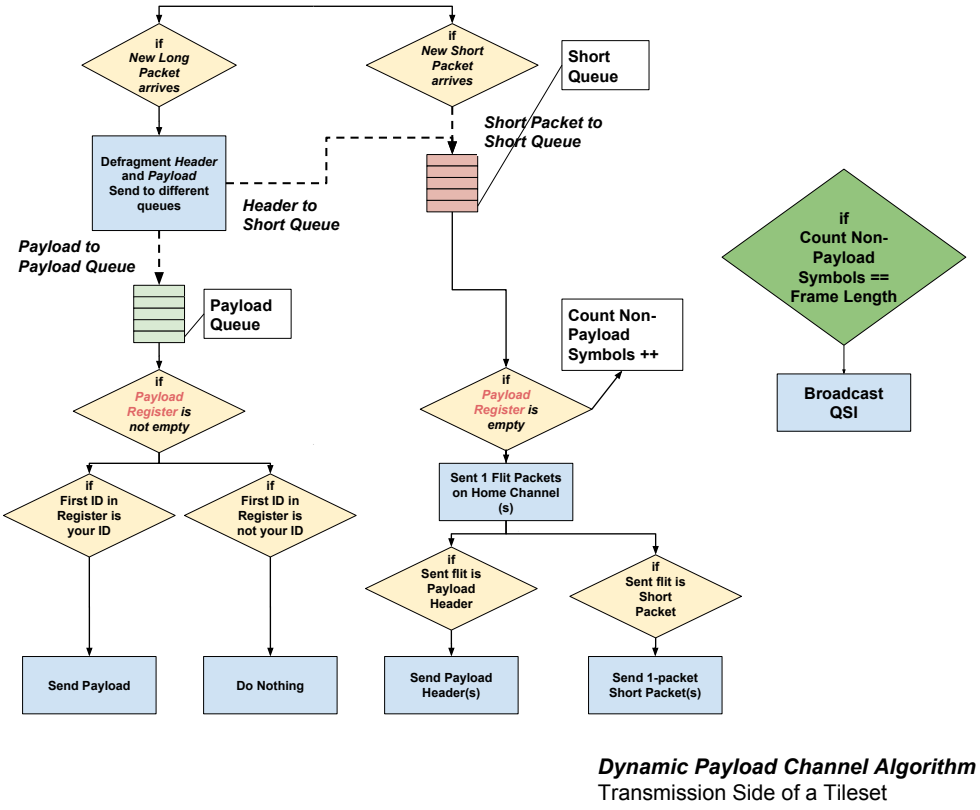


FIGURE 6.12: Flow-chart of dynamic payload channel algorithm from the view of the transmission side of a tileset.

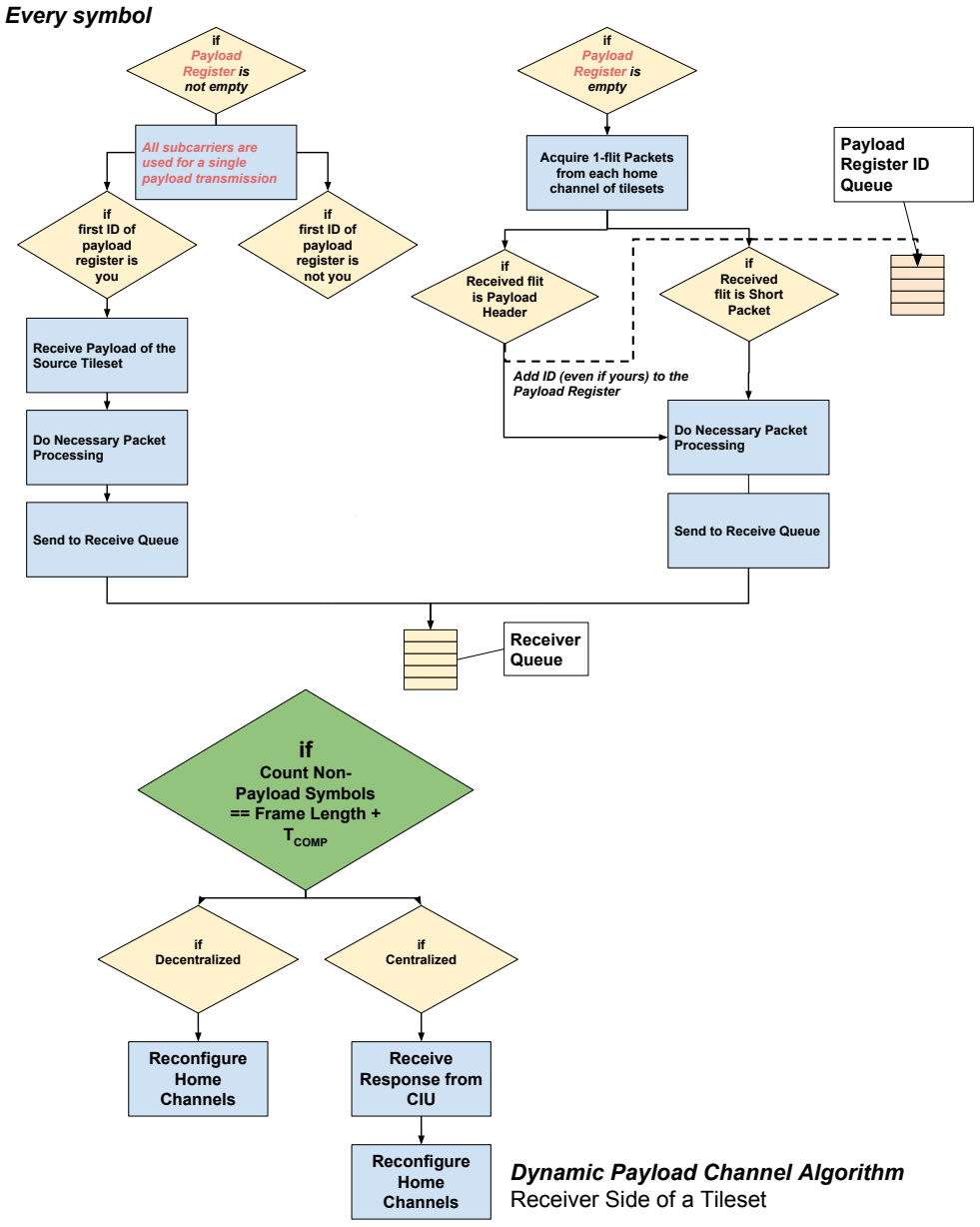


FIGURE 6.13: Flow-chart of dynamic payload channel algorithm from the view of the receiver side of a tileset.

The idea is to change the number of home channels of each tileset has within a frame, by the help of a central intelligent unit (CIU) as in previous dynamic algorithm mechanisms. Similarly, on the first symbol of the each frame, each tileset encodes its QSI. CIU, calculating expected QSIs (EQSI) of tilesets, it allocates the RBs (i.e. home channels in this case) to tilesets according to QPS algorithm. Different than the conventional QPS algorithm, the symbols where payloads are being transmitted is not counted, while determining the length of a frame. Therefore, length of a frame is indefinitive. However, its minimum value, where payload channel is never used, is defined by the required computation and reconfiguration time as in QPS algorithm. Based on the response messages sent by CIU on pre-determined symbol and subcarriers, tilesets reconfigure their arbitration for home channels in next frame. This procedure is illustrated in detail in Fig. 6.11.

Fig. 6.12 and Fig. 6.13 illustrate the flow-charts of the dynamic payload channel algorithm from the view of a transmitter and receiver side of a tileset, respectively.

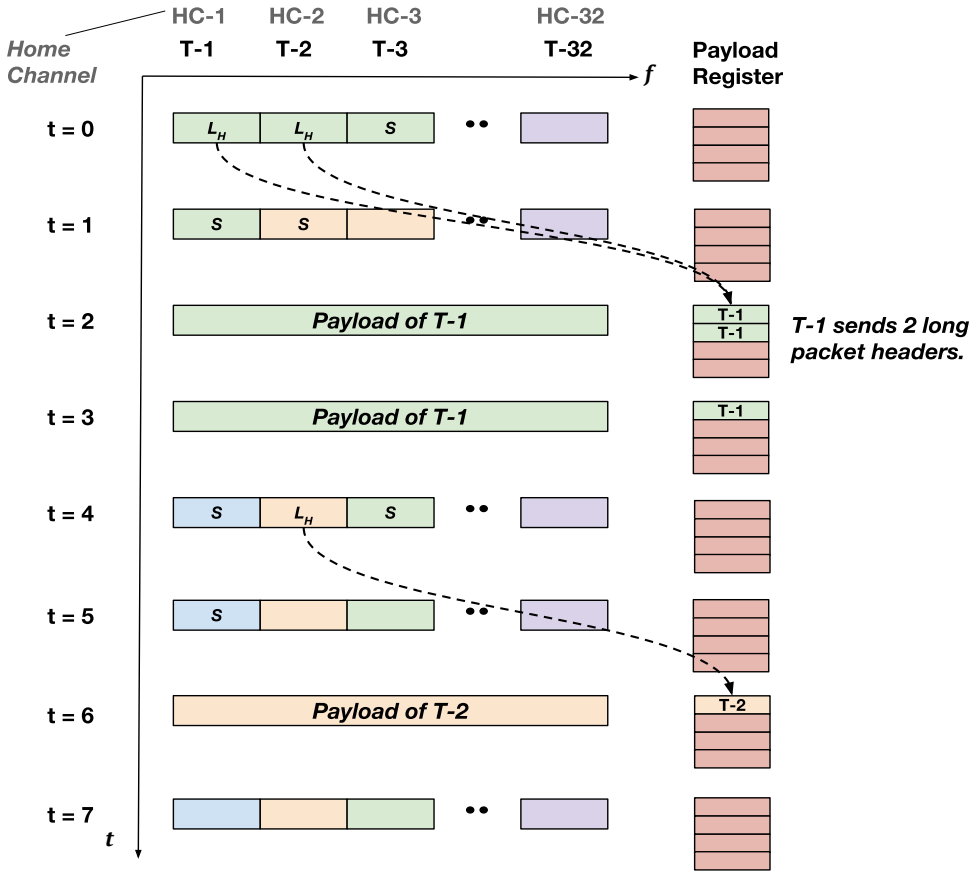


FIGURE 6.14: Illustration of our Dynamic Payload Channel Algorithm both in frequency and time domain through a simple scenario, where only 3 tilesets are active.

Note that, encoded QSIs in this case do not represent the real number of flits waiting in the front-end, but the number of packets (short and long), in other words the elements in the short queue. This is due to fact that, each time payload channel is being used it serves one payload and by allocating home channels proportionally to the number of all packets. We make sure that bandwidth demand is shared proportionally, as each home channel serves a short packet or give access to the payload channel for a long packet.

6.3.1.1 Illustrative Scenario

Fig. 6.14 shows a simple illustrative scenario for dynamic payload channel algorithm where only 3 tilesets are active. For instance, on symbol t=0, Tileset-1 has 3 home channels, where it has transmitted 2 long packet headers and a short packet. As mentioned previously, thanks to separate payload channel queues, inconsistency between long packet headers and payloads is resolved. Therefore, a tileset may also send multiple long packet header on the same symbol and stall the transmission of payloads in payload channel queue, while continuing to use next home channels it acquired. Due to the system latency, both of these payload channel requests are activated on symbol t=2. As Tileset-1 has sent 2 payload channel requests, in the payload channel register its ID is inserted twice. Then, on symbol t=2 and t=3 it transmits the payloads of these 2 packets. And on symbol t=4, the configuration of tilesets' home channels are changed due to distribution in the frame.

6.3.2 Experimental Results for Dynamic Payload Channel Algorithm

We test the performance of our proposed dynamic payload channel algorithm for non-uniform Poisson and DPBPP traffic. In order to ensure a reliable competence, we compare the results to the basic centralized QPS algorithm mention in 5.2, where we also base our algorithm on.

Average Latency

Fig. 6.15 and Fig. 6.16 shows the average latency with increasing total injection rate under non-uniform Poisson and DPBPP (H=0.9) traffic, respectively. The most interesting observation from these points is the better performance of conventional QPS algorithm compared to Dynamic Payload Channel Algorithm after a certain injection rate (For non-uniform Poisson around 2 packets/symbol and for non-uniform DPBPP around 2.5 packets/symbol). This shall root from the requirement of more rapid response to the fluctuations of QSIs under higher injection rates. In higher traffic load, dynamic payload channel algorithm induces more and more payload channel utilization, which disrupts

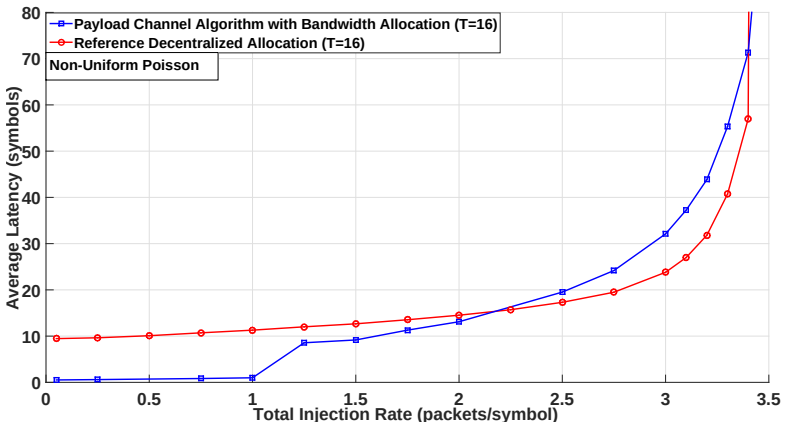


FIGURE 6.15: Average latency under non-uniform Poisson traffic with increasing injection rate for Payload Channel Algorithm and the reference basic QPS allocation.

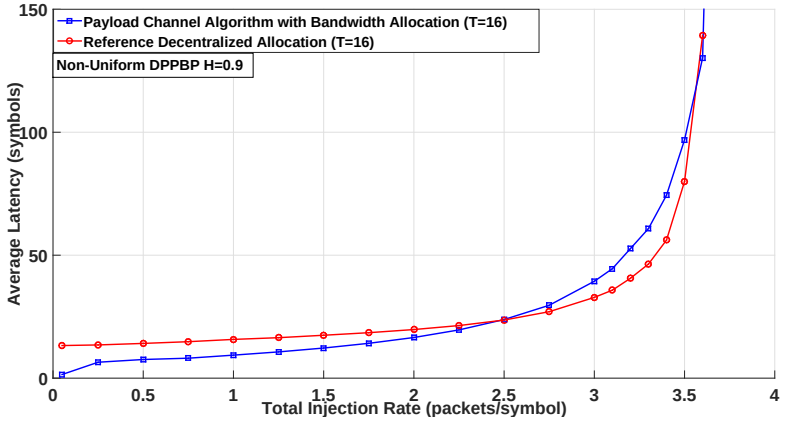


FIGURE 6.16: Average latency under non-uniform DPBPP traffic (H=0.9) with increasing injection rate for Payload Channel Algorithm and the reference basic QPS allocation.

the default home channel configuration and also makes frame lengths much longer (as payload channel symbols are not counted in frame length). This causes to reconfiguration of subcarriers much less frequently and with outdated QSI. However, for low injection rates, the performance of the payload channel algorithm is evident, especially for Poisson traffic.

Packet Delay and Queue Length Bound Exceeding Probabilities

Next, we evaluate the delay and queue length exceeding probability graphs, as usual. Fig. 6.17 shows the delay exceeding probability for proposed dynamic payload channel algorithm and the normal QPS algorithm, under both non-uniform Poisson and DPBPP (H=0.9) traffic with a total injection rate of 3 packets/symbol. Interestingly, we see some better performance (lower probability of exceeding) of conventional QPS compared to dynamic payload channel algorithm, especially for packet delays larger than 30 symbols.

Thus, we can deduce dynamic payload channel algorithm may induce larger delays under relatively high load of traffic. However, we do have shown the better performance of the proposed payload channel algorithm for lower traffic rates.

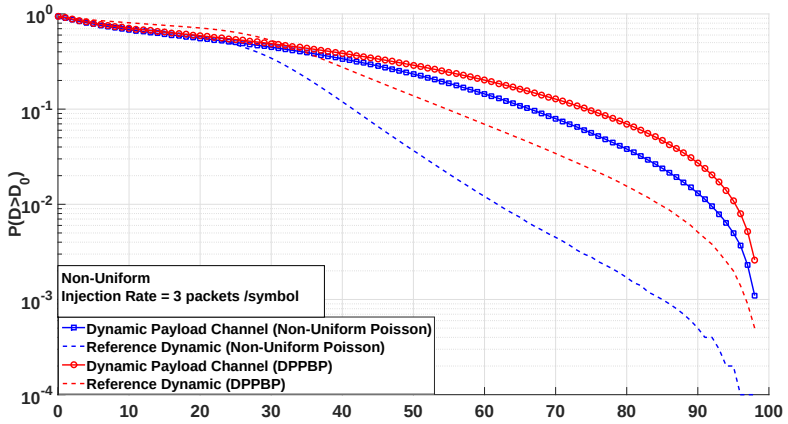


FIGURE 6.17: Delay exceeding probability graphs for our payload channel algorithm under non-uniform Poisson and DPBPP traffic (H=0.9) compared to reference QPS algorithm

Fig. 6.18 and 6.19 shows the queue length exceeding probability graphs for dynamic payload channel algorithm and conventional QPS under non-uniform Poisson and DPBPP (H=0.9) traffic, respectively. Let us remark that, in Fig. 6.18 and Fig. 6.19, the x-axis does not represent the number of flits, particularly. For payload queue, x-axis represents the number of payloads of 32 flits long (cache line); for short packet queue, it represents the number of 1-flit long short packets; for payload register, it represents the number of IDs in a payload register at a time (for 32 tilesets, it is 5-bit long number) and for reference algorithm, it represents the number of flits in a transmission queue as in Chapter 5. We have combined these different indicators in a single graph in order to provide a conceptual coherence. One can dimension and compare the required memory for each of these 3 different queues for payload channel algorithm by looking at these probabilities of exceeding.

We observe that, even for the relatively high bursty input traffic of 3 packets/symbol rate, 1-flit short queue exceeds the 80 flits capacity with a probability around 10^{-4}. Compared to reference conventional QPS, the real bottleneck is the payload queues, as each element is 32 flits long. However, even for a 3 packets/symbol DPBPP traffic, number of payloads being buffered in a tilesets payload queue exceeds 10 with only a probability of 10^{-4}, which shows the feasibility and robustness of the proposed algorithm.

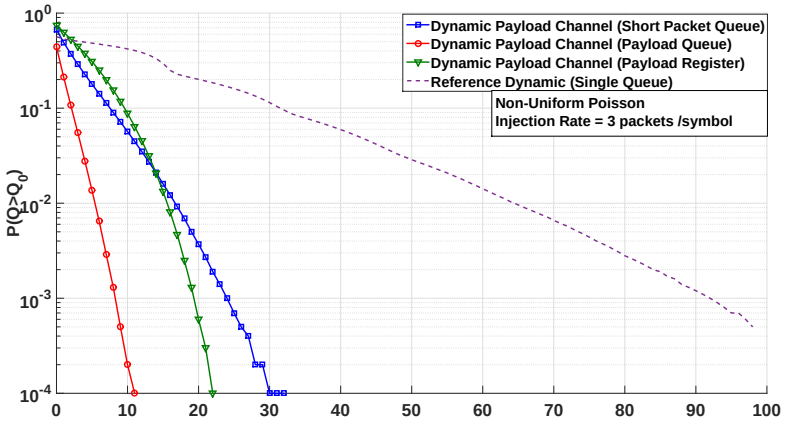


FIGURE 6.18: Queue length exceeding probability graphs for our payload channel algorithm under non-uniform Poisson traffic compared to reference QPS algorithm

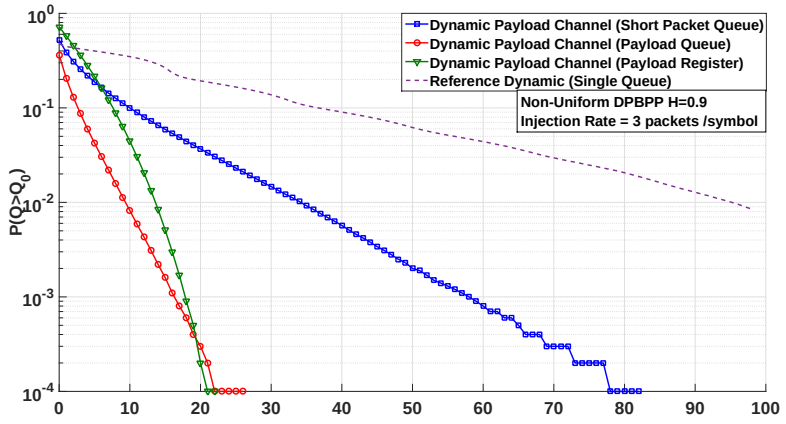


FIGURE 6.19: Queue length exceeding probability graphs for our payload channel algorithm under non-uniform DPBPP traffic (H=0.9) compared to reference QPS algorithm

6.4 Conclusion

We have presented an innovative bandwidth allocation mechanism called "Payload Channel Algorithm", which exploits the bimodal nature of on-chip coherence packets. To the best of our knowledge, this is the first approach of this kind. Using strong reconfigurability and broadcast support of OFDMA, proposed algorithm can decrease average latency up to 20 times compared to a static one, under certain specific scenarios. This novel algorithm requires no additional signaling overhead, and is implemented with significantly low complexity. The algorithm is specially intended for the on-chip architectures with longer cache lines. We have also presented a dynamic version of this algorithm, by merging a centralized bandwidth allocation proposed in Chapter 5; which adapts bandwidth allocation according to spatial traffic fluctuations. We believe proposed Payload Channel Algorithm would constitute an important pioneer work for the management of bimodal on-chip packets.

Chapter 7

Adaptive Modulation Order Selection

Until now, we have reviewed only two degrees of freedom for assigning transmission rates : time (different OFDM symbols) and frequency (different subcarriers). Now, we introduce modulation order as a third degree of freedom, giving the possibility of encode higher number of bits in order to decrease the latencies of packets, at the expense of higher power consumption. We provide in this chapter 2 dynamic modulation order selection algorithms : Maximum delay bounded scheduler and average delay bounded scheduler, both inspired from the literature for generic wireless communications [128][129]. They give a trade-off between latency and power consumption relatively to BPSK case, where we do not impose a real transmission power in Watts. Next in this chapter, an information theoretic analysis investigates required transmission powers to achieve certain capacities on different transmission line topologies.

7.1 Delay-Power Trade-off

A widely known rule of thumb of digital communications is the need for exponentially larger power transmission for higher rates. If we project this rule on a constellation diagram, we see that the Additive White Gaussian Noise (AWGN) drifts locations of received symbols, which causes decoding errors. Since his seminal paper in 1949 [130], Claude Shannon had founded a new paradigm in telecommunications, *Information Theory*, which sets a bound on the achievable rate for a communications channel with arbitrarily small error probability, given a specific Signal-to-Noise Power Ratio (SNR). Another way to approach this to problem is the other way around, i.e. given a desired

probability of error, what is the required minimum transmission power. The well known transmission capacity formula from Shannon is as :

$$C_0 = Blog_2(1 + \frac{P_R}{P_N}) \tag{7.1}$$

where C_0 is capacity in bits/sec, B is bandwidth in Hz, P_R is received signal power in Watts and P_N is the ambient white noise power in Watts. $C = \frac{C_0}{B}$ is spectral capacity density in bits/sec/Hz. P_N can be calculated as :

$$P_N = BN_0 \tag{7.2}$$

where N_0 is the noise spectral density in Watts/Hz (scalar), which we assume as -174 dBm/Hz. P_R/P_N is referred as received Signal-to-Noise Ratio (SNR). Note that even though N_0 is called noise power spectral density, we have chosen to denote the capacity spectral density as C, and capacity as C_0. This is because, we use capacity spectral density as a more important metric, which is associated to modulation orders.

After manuplating these equations, we can achieve the required minimum received signal power for achieving a transmission rate capacity (Note that we are using information theoretic rate capacity (C_0) and transmission rate (R) as interchangable terms) :

$$\frac{P_R}{P_N} = 2^{C\,B} - 1 \tag{7.3}$$

In case of quantized digital communications, such as OFDM, we can transform this equation to the relation, showing the ratio of minimum transmission power required with increasing modulation order (i.e. rate-*bits/subcarrier/symbol*) compared to minimum modulation order BPSK (1 bit/subcarrier/symbol), where M is the modulation order rate (e.g. if 16-QAM : 4 bits/subcarrier/symbol) :

$$\frac{P_R}{P_N} = 2^{M\,B} - 1 \tag{7.4}$$

We have seen that required power transmission is an exponentially increasing cost for higher rates. Taking into account how energy is a valuable resource, a voluminous literature on energy efficient optimal rate selection has been created by researchers, from cellular communications to satellite communications, from optical fiber connections to wireless sensor networks [131][132][133][134]. *Rate*, actually, is a tool, not the ultimate goal. In other words, a designer would like to control the metrics of Quality-of-Service (QoS) such as average latency, maximum delay, maximum buffer length required, probability of buffer overflow etc., which are results of the selected rate. It is interesting to see that, optimality of delay-power relation is rather an under-exploited subject. The main idea is simple : packets in queues can be delayed in order to send them in longer durations, with slower rates (lower modulation orders), thus to save power. Fig. 7.1 shows the typical Pareto like delay-power relation, which can be potentially exploited.

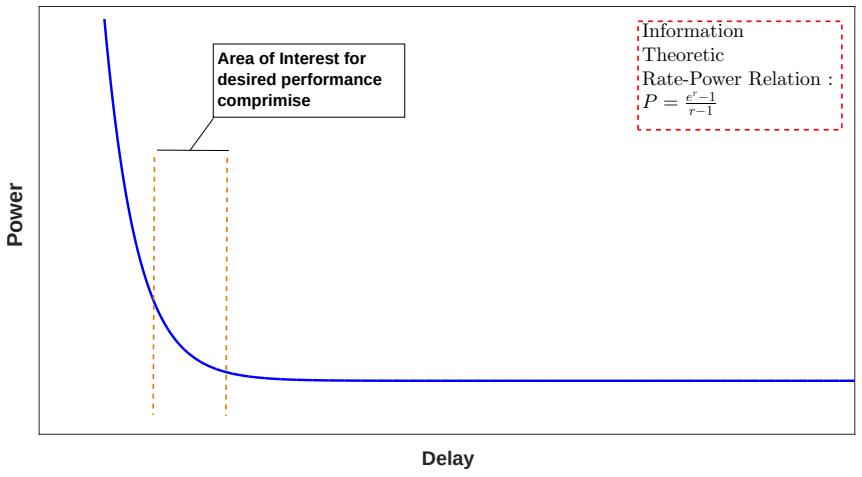


FIGURE 7.1: Typical Convex Delay-Power relation, which gives designer to exploit large energy savings by trading latency

Various researchers approached this issue from different perspectives. At first, parameters and metrics of interest are relatively defined; i.e. one may desire to minimize average delay while setting a bound for the transmission power budget, or vice-versa. One other may wish to set bounds on the maximum delay due to Quality-of-Service requirements or maximum transmission power due to circuit restrictions. In addition, this rate scheduler shall be designed based on different paradigms depending on the mode of transmission, environment, nature of traffic etc. In this chapter, we provide 2 dynamic modulation order selection algorithms for WiNoCoD, trying to minimize the average energy expenditure, while first one is setting a bound on maximum delay of the packets, second one average delay of the packets.

7.2 Adaptive Modulation Option in OFDMA and Delay

Another revolutionary advantage of using OFDMA in WiNoCoD is the option of using higher modulation orders dynamically on different symbols and subcarriers. In addition to frequency and time, this option provides a third dimension of flexibility. In design phase of WiNoCoD project, 4 modulation orders : BPSK, QPSK, 16-QAM and 64-QAM are selected to be implemented, which provides 1,2,4 and 6 bits per subcarrier respectively. It was shown in M. Hamieh's work that, for our transmission line (see Section 3.3.2.3), between two farthest tilesets, minimum required power for transmission on a single subcarrier with a bit error rate set to 10^{-8} using BPSK is -88 dBm and using 64-QAM is -74 dBm [79]. Noise Factor of Low Noise Amplifiers (LNAs) are set to be 3 dB over whole spectrum. Therefore, using 64-QAM needs 14 dB more power than using BPSK, in other words approximately more than 25 times in linear scale. However, in our simulations while testing our proposed solutions, we will stick to the equation 7.1, which gives the information theoretic rate-power relation. This would allow any kind of different channel coding techniques to be applied, without considering a specific bit or packet error rate. In addition, we have decided to employ modulation orders up to 256-QAM, including also 8-PSK, 32-QAM and 128-QAM, which may be available in further phases of the project. While testing our simulations, we considered the power consumption metric as the linear ratio to the power of lowest modulation order, BPSK. In this thesis work, adaptive modulation order selection is performed on the dynamic bandwidth allocation algorithms, that reconfiguration in all 3 dimensions are exploited.

7.3 Decentralized and Centralized Modulation Order Selection Policy

7.3.1 Decentralized Modulation Order Selection

In decentralized algorithms, just as in previous chapters, on the first symbol of each frame, tilesets broadcast their QSIs. After each tileset receives information on buffer lengths of each other tileset, all tilesets process the same algorithm on the same QSI data to reach the same decision on the arbitration of subcarriers. This process is assumed to take 1 frame length, thus new configuration of the bandwidth is activated for the next frame. We introduce here a set of decentralized dynamic modulation order selection policies on to this framework, where at each frame, each tileset decides on its modulation order for the next frame using its QSI and also newly calculated number of subcarriers to be used in next frame. As other tilesets must learn this chosen modulation order to

successfully decode this tileset's transmissions on its dedicated RBs, after deciding on the modulation order each tileset encodes its modulation order on the last symbol of the current frame. The possibility for computing and activating different modulation orders on each frame (coherent with the framed structure explained in previous chapters) in a decentralized setting, is illustrated in Fig. 7.2.

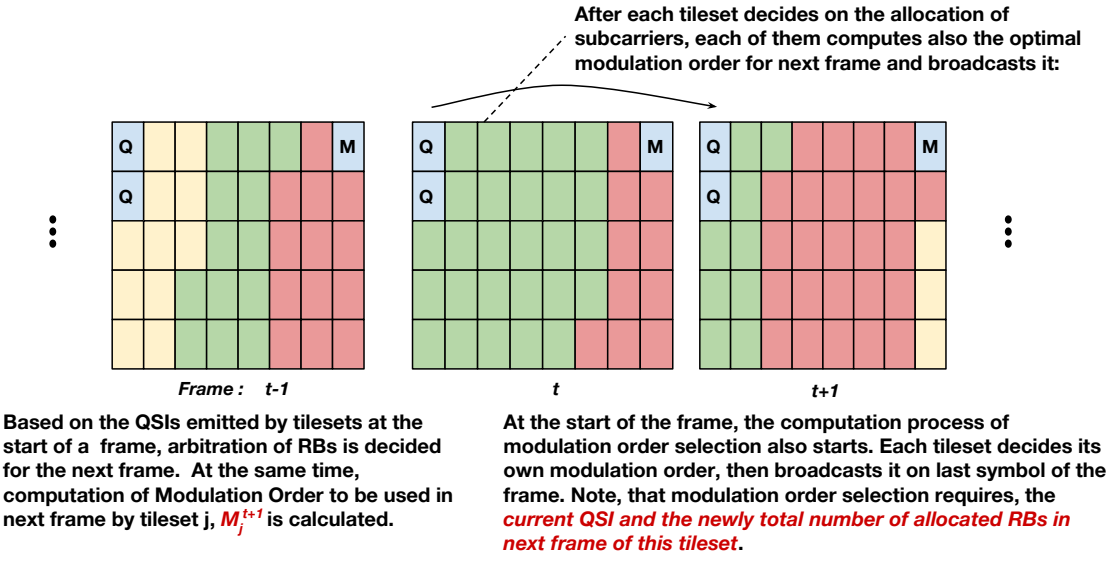


FIGURE 7.2: Framed structure of decentralized RB allocation and modulation order selection in WiNoCoD

Note that, a tileset needs the total number of RBs it will use during the next frame. OFDMA gives the possibility to encode different modulation orders on each subcarrier, every symbol. However, in an on-chip environment like WiNoCoD with bandwidth and computation restrictions, we constructed a paradigm, where a tileset decides on its modulation order to be used, through whole next frame. Therefore, if a tileset has S RBs (say which can serve S flits with the lowest modulation order, BPSK) in next frame, and if it decides to use a modulation order of 16-QAM, it can serve up to 4S flits. Hence, modulation order selection should be done carefully, provisioning power expenditure, based on the instantaneous QSI and traffic characteristics. Remind that, a tileset selects its modulation order based on the total number of RBs it will have in next frame, therefore through the current frame, it shall perform this operation after the computation of bandwidth allocation. After selecting its modulation order, a tileset broadcasts this modulation order on dedicated subcarriers, so that other tilesets which decode RBs of this tileset can use this moduation order for successful decoding. As you can see, the tileset receivers should activate this operation, *matching RBs and decoding modulation order* via a circuitry or processing unit, just in one symbol. In next symbol, the new frame starts with the new RB arbitration and used modulation orders. We also assume, tilesets are able to tune their transmission powers in one symbol duration.

Another important point is that each tileset, after deciding its modulation order, should tune their transmission power according to this new modulation order. Fig. 7.3 illustrates the process of power tuning and constellation mappings in a tileset's OFDMA transmission chain.

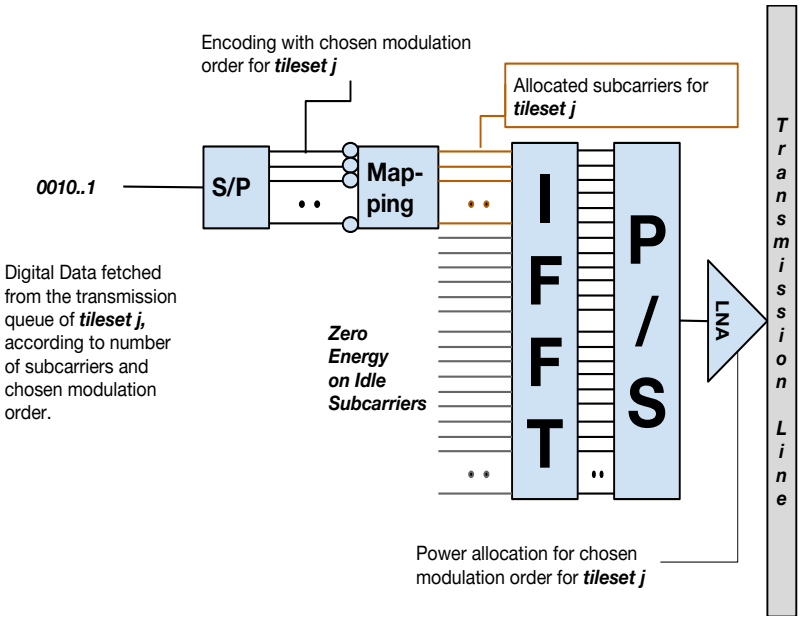


FIGURE 7.3: Transmission power tuning for the chosen constellation order

7.3.2 Centralized Modulation Order Selection

For the centralized modulation order selection, Central Intelligent Unit (CIU) is also responsible for the chosen modulation order of each tileset. After acquiring the QSIs of the tilesets, CIU determines the number of RBs allocated to each tileset for the next frame, as explained in previous chapters. However, this time, after calculating the number of RBs allocated to each tileset, it determines the selected modulation order for the next frame using this information and QSI. Following this, it broadcasts the selected modulation order along with the number of allocated RBs on the reserved subcarriers. As there are 8 possible modulation orders, this means an extra 3 bits are required for overhead. Just as in the centralized approach explained in Section 4.3.3, after receiving this information tilesets reconfigure their transmission for the next frame in $T_{reconfig}$ symbols. Additionally, tilesets also tune their transmission power for the next frame, and reconfigure their receiver for proper decoding subcarriers from each tileset with different modulation orders. This procedure is illustrated in Fig. 7.4.

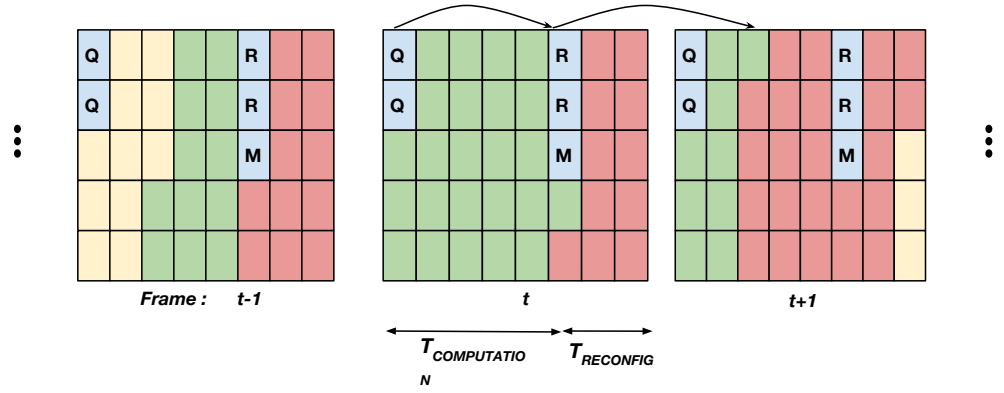


FIGURE 7.4: Framed structure of centralized RB allocation and modulation order selection in WiNoCoD

7.4 Maximum Delay Bounded Scheduler

We have mentioned the possibility of saving energy, by sending data at lower rates, hence more delayed. One shall formulate this compromise elegantly in order to exploit the trade-off efficiently. One method to approach this aspect is to setting a maximum delay bound on every generated packet in the system, guaranteeing them to depart the queues in less than D_{max} time, and choosing the transmission rate every instant, such that the average power is minimized. Authors in [128] has investigated this subject for single user and single Gaussian channel case with slotted transmission. They have shown that an optimal rate scheduler of this type is actually a *Low Pass Filter*. This is intuitive as main idea of energy saving is to lower the instantaneous output rate compared to instantaneous input rate, thus lowering transmission power at the expense of delay. The dual of this perspective is the smoothing of output rate by a time averager, a linear low-pass filter. Authors have assumed arbitrary distribution of input statistics, therefore bursty arrivals. In case of linear time-invariant scheduler, where coefficients of scheduler are constant, not depending on time, they have proved that filter has a length of D_{max} and all coefficients are equal to $1/D_{max}$, regardless of the statistics of input traffic. In other words, for each number of packets that arrived at time t, the optimal allocation of rate is to divide the rate equally in this slot and further $D_{max} - 1$ slots. This gives the minimum transmission power for time-invariant case due to convex relation between power and rate. For instance, when a message of 6 packets arrive and we set a delay bound of 6 slots, optimal rate allocation is to serve 1 packet on current and next 5 slots, because power is an exponential function of rate. The formulation of

this optimal scheduler, including also all inputs arrived up to $D_{max} - 1$ slots before, is as follows :

$$R_t = \frac{X_t + X_{t-1} + .. + X_{t-(D_{max}-1)}}{D_{max}} \tag{7.5}$$

An example of this scheduler is shown in Fig. 7.5, where theoretically it can save up to 25 times of energy, by delaying the transmission by 5 slots, in this specific case. Further in [135], authors extend the case to linear time-varying filters, proving that they may improve the performance. However, in limited computational constraints of on-chip environment, we evaluate the possibility of adapting this basic optimal time-invariant scheduler to WiNoCoD, where there is no single channel per tileset, but changing number of multiple channels per frame. We believe, this adaptation not only improves the energy efficiency of WiNoCoD, but also extends the delay bounded optimal scheduler for all generic communication systems with varying number of channels.

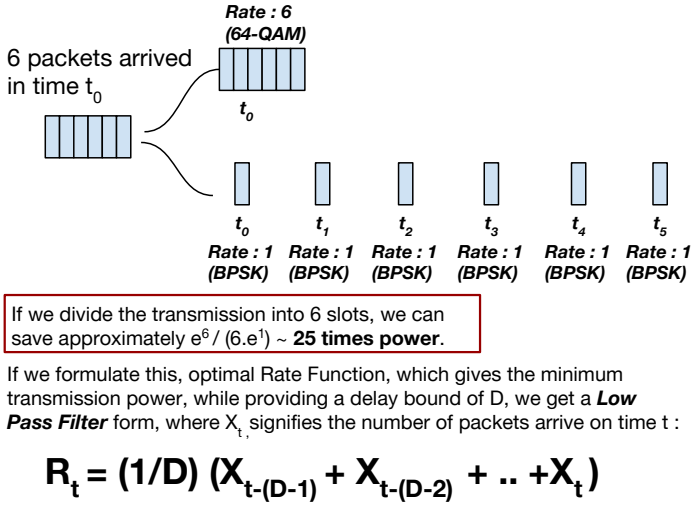


FIGURE 7.5: Maximum delay bounded rate scheduler can save substantial amount of power, as shown in this example

7.4.1 Extension of Delay Bounded Scheduler to Multiple Channels

In a scenario, where a transmitter can have more than one channel, -*say where each of them can serve one packet with the lowest modulation order*-, the aforementioned scheduler is not valid. This roots from the fact that, transmitter can serve more than the already scheduled rate for current symbol as it may have larger number of channels, thus optimal scheduling should be updated such that lowering rate in next slots, while

ensuring serving all data in D_{max} slots. Let us explain this in a basic example which is illustrated in Fig. 7.6. Say for an example that 8 packets arrive at a transmitter's queue on a slot t and maximum delay bound is 4 slots. Also, say that this transmitter has 1 channel on current symbol and 3,0,2 channels in next 3 slots, respectively. Note that each channel can serve 1 packet with the lowest modulation order. According to the maximum delay bounded optimal scheduling explained above, we should choose the lowest modulation order on every slot, such that allocating $8/4 = 2$ packets/slot rate on each slot. In this case, we should choose the second modulation order (i.e. QPSK, 2 packets per channel) on the first slot, as we have 1 channel. According to equation, we should choose the lowest modulation order on second slot, as we should schedule 2 packets/slot and we have 3 channels, we can serve 3 packets on this slot, even with the lowest modulation order. Note that, we have served more packets than the scheduled 2 packets/slot. On third slot, we have no channels, therefore we cannot transmit any packets. And finally, on last slot, we have 2 channels and we should choose the lowest modulation order to schedule 2 packets/slot. However, note that a packet left in the queue which violates the maximum delay bound.

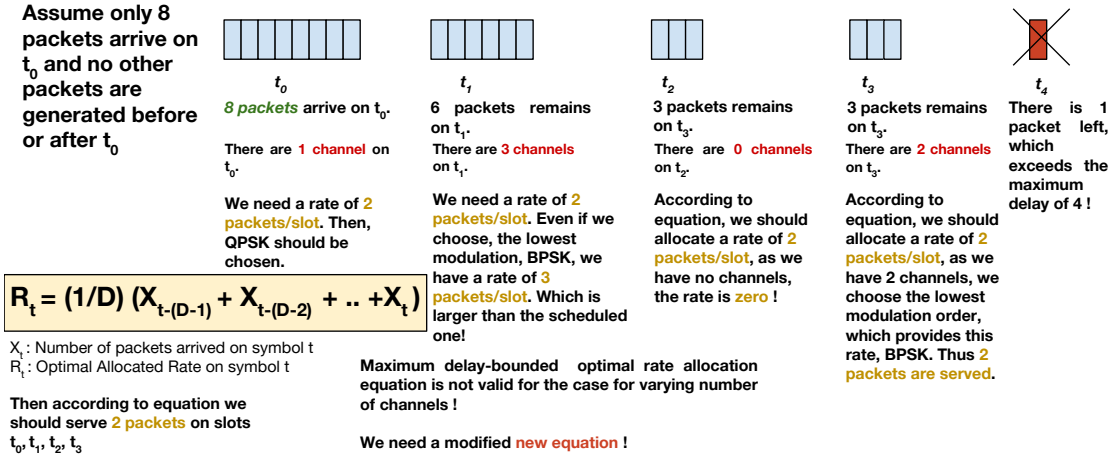


FIGURE 7.6: Standard low-pass filter equation is not valid for the multichannel case.

Hence, this encouraged us to re-formulate the scheduler for changing number of channels. Another approach to look to this problem is not scheduling according to arriving number of packets in previous D_{max} slots, but according to current number of packets with different deadlines (i.e. *remaining time for the packet till violating delay bound*). Therefore, system needs to profile all packets in the queue, according to their deadlines. As, main idea of the optimal time-invariant scheduler is to divide the total transmission uniformly in further D_{max} slots, we may think this as to allocate a total rate on current symbol, summing the different rates allocated to packets with different deadlines,

dividing the number of packets to their deadline. This is formulated as follows, where Q_τ symbolizes the number of packets with a deadline of τ on slot t :

$$R_t = Q_1 + \frac{Q_2}{2} + .. + \frac{Q_\tau}{\tau} + .. + \frac{Q_{D_{max}}}{D_{max}} \tag{7.6}$$

The duality of this new formula is illustrated in Fig. 7.7, with the previous example. Note how this basic modification is valid under multiple channel case.

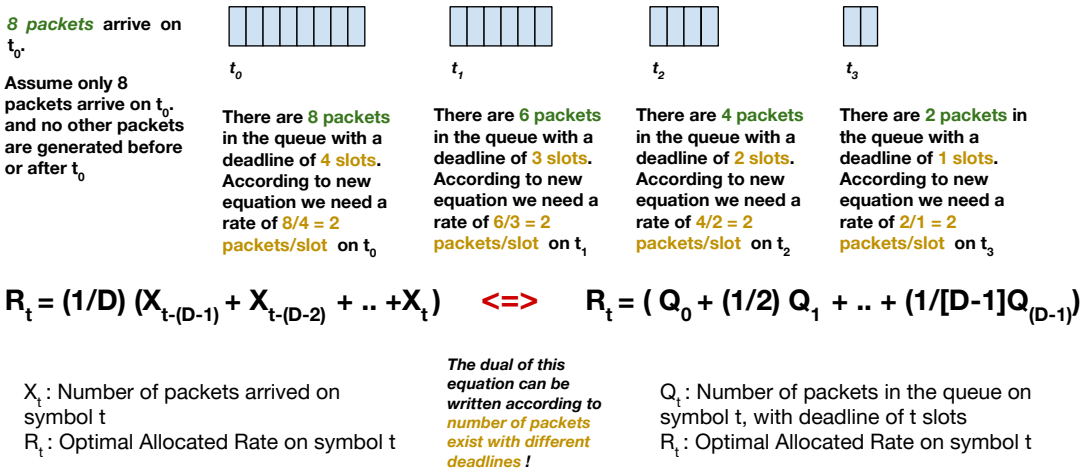


FIGURE 7.7: The proposed dual equation for maximum delay bounded scheduler using instantaneous queue length

7.4.2 Maximum Delay Bounded Scheduler for WiNoCoD

Using the principle proposed in the previous section, we design a maximum delay bounded scheduler for WiNoCoD, with a decentralized approach. Upon receiving QSI on the first symbol of a frame, tilesets run the conventional bandwidth allocation algorithm to determine their RBs to use in next frame, as previously explained pipelined fashion. Then, using its own QSI and newly allocated number of RBs, each tileset also determines its modulation order based on the maximum delay bounded rate scheduling principle and broadcasts it on the last symbol of the frame.

As scheduling is done in frames, setting delay bounds to packets can only be effectuated in terms of frames. In other words, a packet can be guaranteed to have a maximum delay as an integer multiple of symbols in a frame. Furthermore, as new modulation order and bandwidth allocation is activated in next frame and based on the QSI broadcasted on the first symbol of current frame, we can only guarantee a minimum value for the

maximum delay of 3 frames. In order to better explain this situation, we depict it in Fig. 7.8 with an example. The currently broadcasted QSI may include a packet arrived in any symbol in last frame. Therefore, the QSI may be contributed by a packet arrived T symbols before. And on the other hand, as the new rate is scheduled for the next frame and all the packets are guaranteed to be served by the end of this frame, we can only guarantee the service of a packet with a maximum bound of 3 frames length.

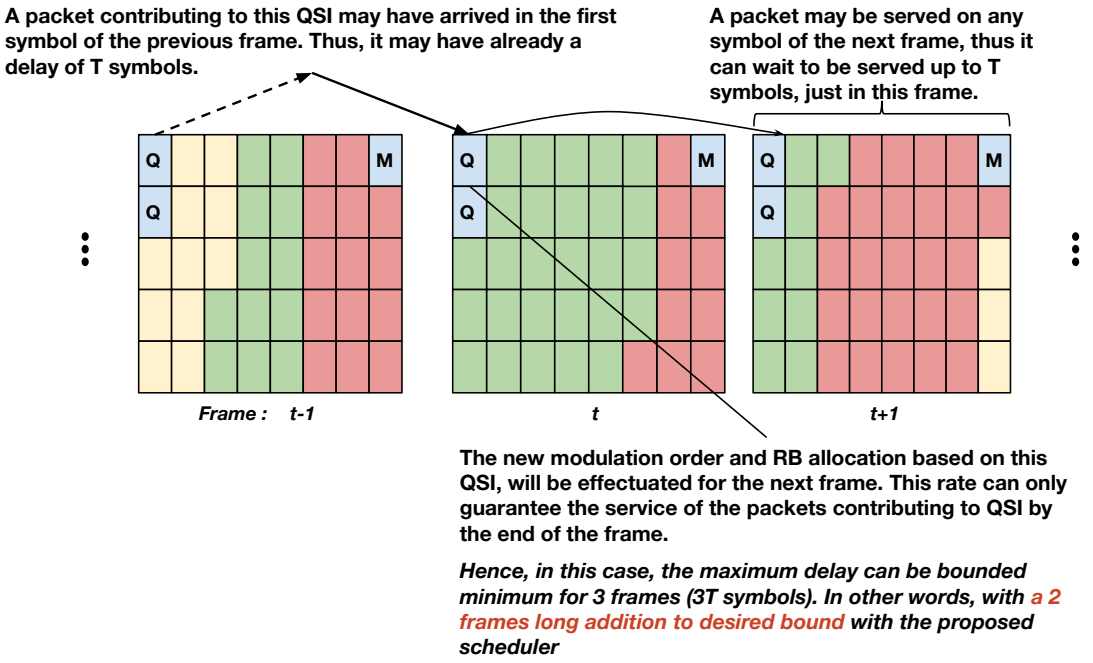


FIGURE 7.8: With the proposed scheduler, the delay of packets can only be guaranteed to have a desired maximum bound with a 2 frame length addition.

7.4.2.1 Experimental Evaluation

We have tested the proposed maximum delay bounded scheduler with the aforementioned decentralized context, by using Expected Queue Proportional Scheduling (EQPS) explained in 5.2.1 with allocation in time direction. The main reason behind that it allows the allocation of at least 1 RB to every tileset which is likely not to be idle. If a tileset is not allocated any frequency resource in a frame, setting the modulation order, therefore the necessary instantaneous rate is not possible. In addition, we have shown that EQPS provides remarkably good performance under high injection rates. 8 modulation orders from BPSK (1 bit/subcarrier) to 256-QAM (8 bits/subcarrier) are utilized, as mentioned previously. However, we remind that any other kind of subcarrier arbitration algorithm can be implemented on this decentralized maximum delay bounded modulation order selection framework. Due to limited space, only a few experiments are demonstrated. Note that, this time the lowest modulation order is BPSK. We have

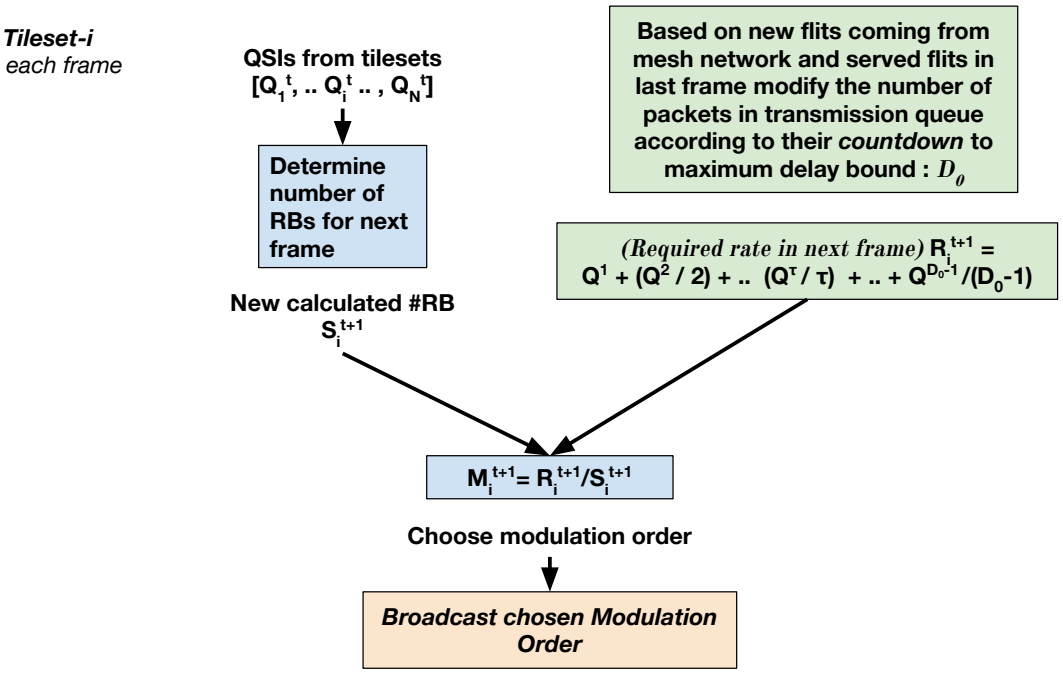
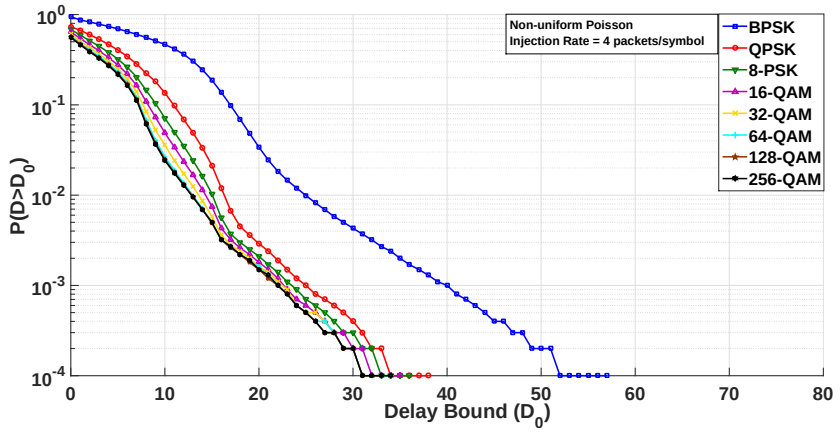


FIGURE 7.9: Flow chart of the maximum delay bounded scheduler with distributed subcarrier allocation.

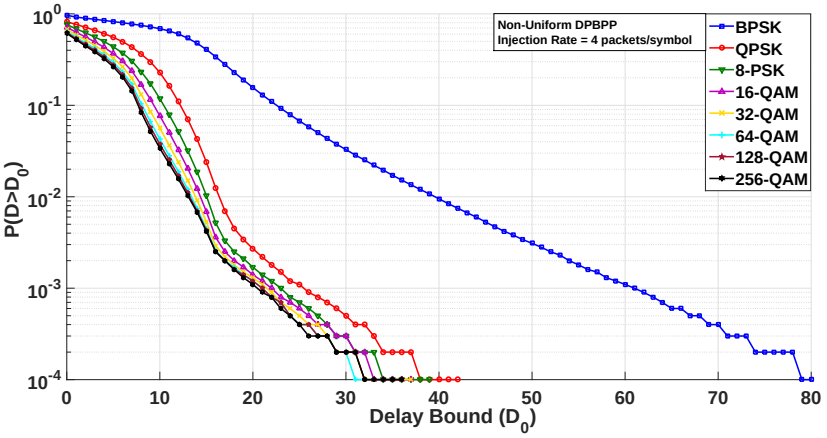
used QPSK as default in previous chapters, thus this means the capacity of the system is halved if only BPSK is utilized. In addition, QSI and modulation order signaling is always done in BPSK for minimizing probability of error. Therefore, the required number of RBs allocated for signaling is two times more in this case. For total injection rate values of 4, 6 and 8 packets/symbol, the simulation for a frame length of 8 symbols is executed under non-uniform Poisson and DPBPP traffic with H=0.9, for different maximum delay bounds.

Injection Rate = 4 packets/symbol

Before testing our algorithm for the injection rate of 4 packets/symbol non-uniform Poisson and DPBPP traffic, in Fig. 7.10(a) and in Fig. 7.10(b), we provide the delay exceeding probability graphs for the framed EQPS allocation with static modulation order under these scenarios. The reference static modulation order for modulation orders from BPSK to 256-QAM uses the same frame length and bandwidth allocation mechanism, and consitutes a reference for the proposed dynamic modulation order. Remind that, utilizing higher modulation orders constantly gives much more lower delay exceeding probabilities at the expense of exponentially higher modulation orders. The resulting average power in terms of power required for BPSK with constant modulation order utilization is shown in Fig. 7.11.



(a) Non-Uniform Poisson



(b) Non-Uniform DPBPP (H=0.9)

FIGURE 7.10: Packet delay exceeding probability graphs for non-uniform Poisson and DPBPP traffic with injection rate of 4 packets/symbol for a static modulation order system for different utilized modulation orders (EQPS($\alpha = 0.95$), time direction allocation, T=8 symbols)

Fig. 7.12(a) shows the probability of a packet exceeding a given delay for each tried maximum delay bound with the proposed scheduler under non-uniform Poisson traffic with an injection rate of 4 packets/symbol. As mentioned previously, the delay bounds can be quantified in our algorithm in terms of frame lengths and 2 frames length shall be added to this initially given bound due to the structure of the algorithm. For instance, in this case we have assumed a frame length of 8 symbols; therefore one can start to guarantee a minimum value for maximum delay bound as 3 frames (24 symbols). By inspecting Fig. 7.12(a), we see that except for the 1 (1+2) frame length (24 symbols), all the given delay bounds have been validated (For a delay bound of 24 symbols, probability of exceeding 24 symbols is around 0.0003). In addition, after 24 symbols bound, the algorithm operated with 2 (2+2) frame length (32 symbols) perform even better compared to 1 (1+2) frame length bound. This is because, when we impose a delay bound of 1 frame (where it is a single length filter), most of the time the algorithm can not perform coherently, as it can not sustain the necessary instantaneous rate with the highest

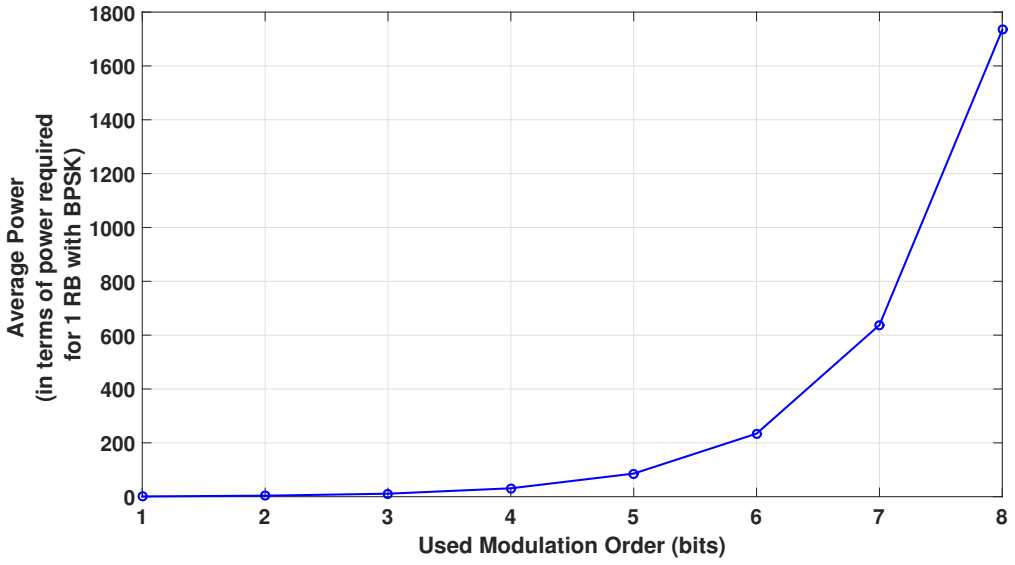
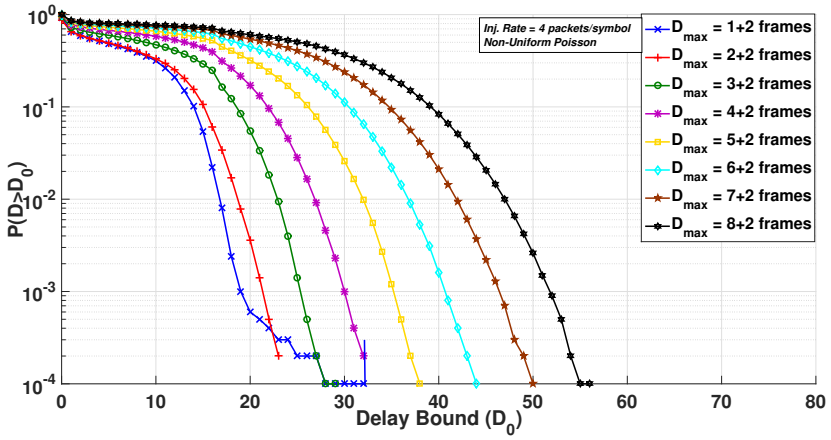


FIGURE 7.11: The average transmission power increases drastically, if higher modulation orders are used constantly.

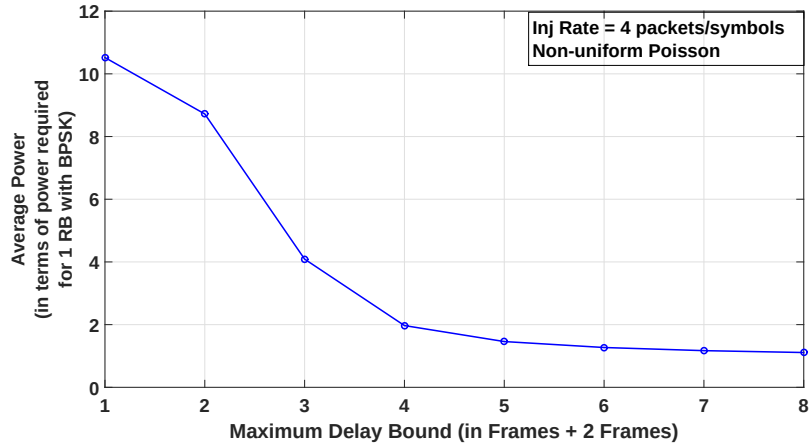
modulation order of 256-QAM. Also, frame structure causing outdated information on queue lengths, makes tight delay bounds almost impossible.

In Fig. 7.12(b), we see the resulting average power expenditure for 8 different delay bounds (from 24 symbols/1+2 frames to 80 symbols/8+2 frames). We can see that, as we are increasing the maximum delay bound, we can decrease the consumed avarage power. For a 1 (1+2) frame length (24 symbols) maximum delay bound, the resulting average power is slightly above 10 units (in terms of minimum transmission power required for a single RB (32 subcarriers) coded with BPSK); whereas for a delay bound of 4 (4+2) frame length (48 symbols), the resulting average power is around 2 power units. This means, by increasing average delay bound 2 times for this scenario, we can decrease the total energy expenditure by 5 fold. For higher delay bounds, we observe that there are no more substantial power gains. This scheme is predictable and consistent with the previously explained Pareto like delay-power relation.

Then, we evaluate our algorithm for the same injection rate of 4 packets/symbol, but for realistic non-uniform DPBPP traffic with H=0.9 as in Fig. 7.13. First of all, we see that intended delay bounds of 1 (1+2) frame (24 symbols), 2 (2+2) frames (32 symbols), 3 (3+2) frames (40 symbols) and 4 (4+2) frames (48 symbols) could not have been achieved, with a suitable probability of error. We observe that the probability of exceeding these delay bounds are 0.0413, 0.0067, 0.0025, 0.0012 respectively (See Table 7.1). We also observe that, delay bound of 1 frame is even performing worse than a delay bound of 2 frames. This inconsistency is similar to the previous non-uniform



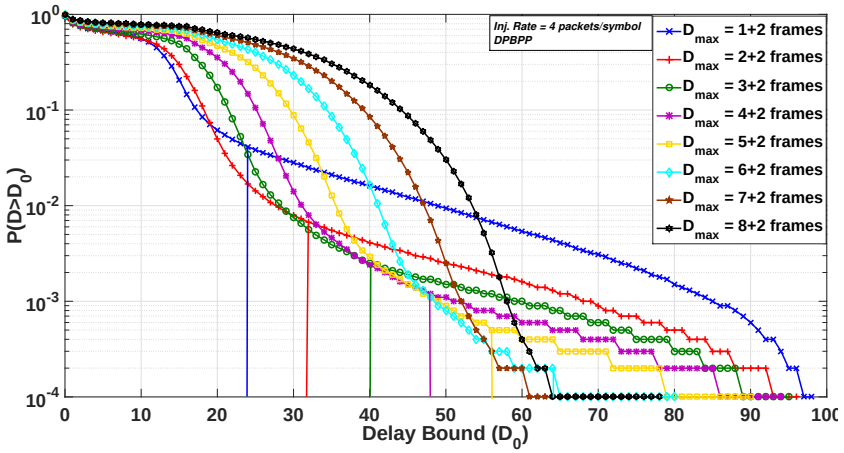
(a) Packet delay exceeding probability graph. For each tried maximum delay bound (in frames + 2 frame length addition).



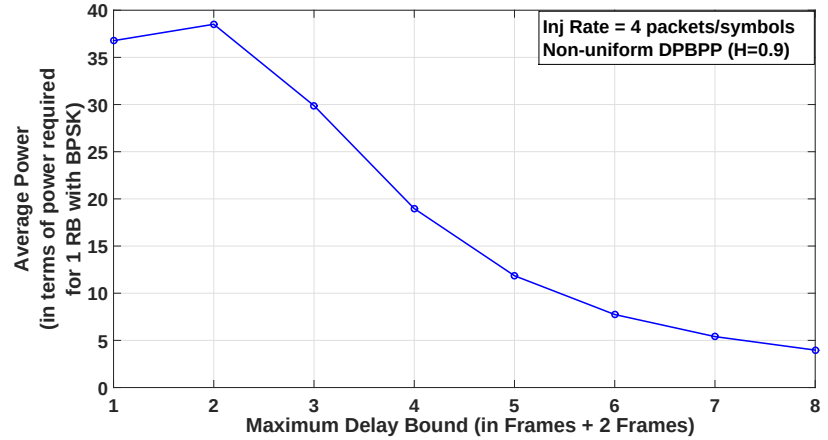
(b) The resulting average power in terms of transmission power required for a single RB with BPSK, for each tried maximum delay bound in frames. (Actual delay bound is 2 frames more)

FIGURE 7.12: Decentralized maximum delay bounded scheduler performance under non-uniform Poisson traffic with injection rate of 4 packets/symbol (EQPS($\alpha = 0.95$), time direction allocation, T=8 symbols)

Poisson case, but even worse. This is due to the more temporal burstiness of DPBPP traffic. Our algorithm can not provide the instantaneous peaks in traffic demands, as the highest modulation order is limited and pipelined/framed structure does not allow for efficient rate allocation, especially under rapidly fluctuating traffic. At this point, it is also important to note that for delay bounds larger than 60 symbols, all algorithms (except for delay bound with 1 frame), converge to nearly same exceeding probabilities. We observe that, even larger delay bounded algorithms provide lesser probabilities of exceeding delays larger than 60 symbols. This is also an unexpected and erroneous result. One of the reasons behind this is that even for delay bounds with several frames, the packets experiencing delays larger than 60 symbols is a relatively rare case. Due to highly bursty traffic, queue dynamics would not evolve consistently, with dynamic rate allocation. However, we can still claim that the proposed algorithm provides a maximum



(a) Packet delay exceeding probability graph. For each tried maximum delay bound (in frames + 2 frame length addition).

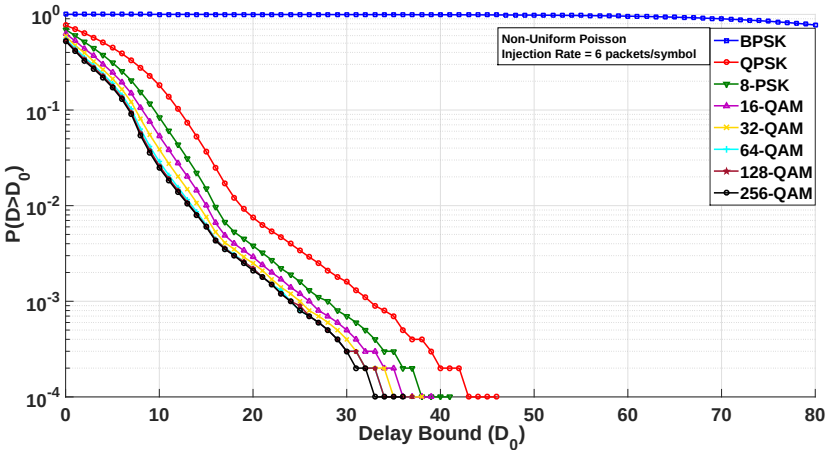


(b) The resulting average power in terms of transmission power required for a single RB with BPSK, for each tried maximum delay bound in frames. (Actual delay bound is 2 frames more)

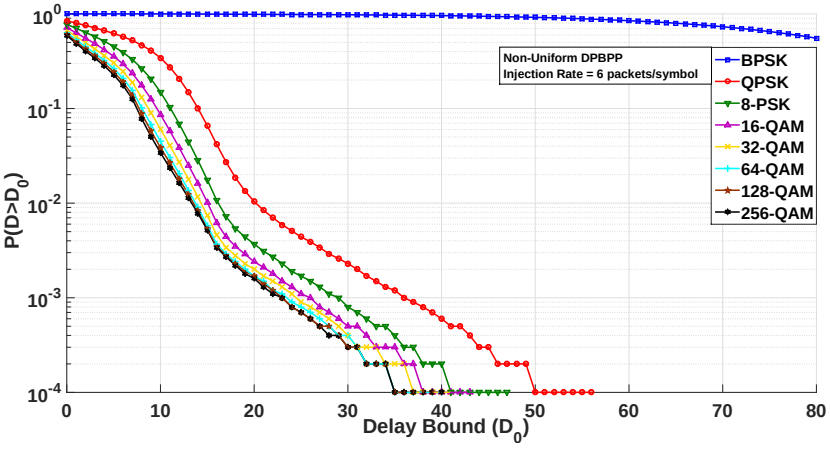
FIGURE 7.13: Decentralized maximum delay bounded scheduler performance non-uniform DPBPP traffic (H=0.9) with injection rate of 4 packets/symbol (EQPS(α = 0.95), time direction allocation, T=8 symbols)

delay bounded energy minimization with a reliability up to a degree.

In Fig. 7.13(b), we see the resulting average power expenditure for 8 different delay bounds (from 24 symbols/1+2 frames to 80 symbols/8+2 frames). First noteworthy result from this figure, is that 1 frame length maximum delay bound has even consumed less power, compared to 2 frame length maximum delay bound. This proves an invalidity for the performance of the algorithm with 1 frame length bound. This errenous result is also highly consistent with the observation we made for Fig. 7.13(a), for the probability of exceeding a packet delay; that it looks like the algorithm with 2 frame length bound has chosen even higher rates compared to 1 frame length bound. However, we observe that, as we are increasing the maximum delay bound above 2 frames, we can decrease the consumed average power substantially. For instance, by decreasing maximum delay



(a) Non-Uniform Poisson



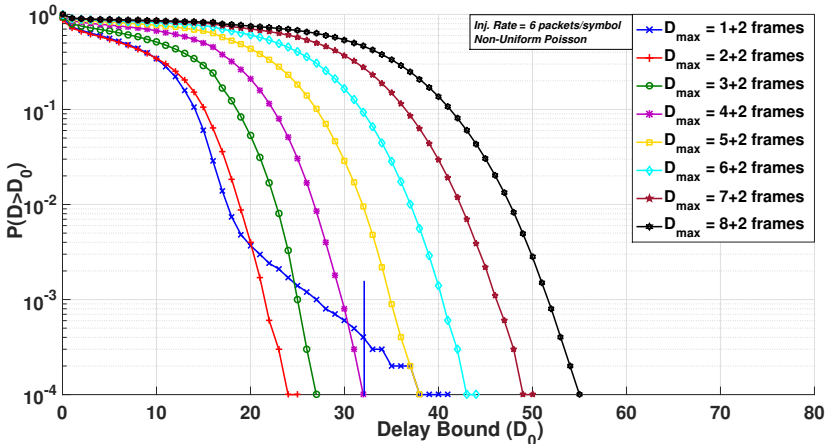
(b) Non-Uniform DPBPP (H=0.9)

FIGURE 7.14: Packet delay exceeding probability graphs for non-uniform Poisson and DPBPP traffic with injection rate of 6 packets/symbol for a static modulation order system for different utilized modulation orders (EQPS($\alpha = 0.95$), time direction allocation, T=8 symbols)

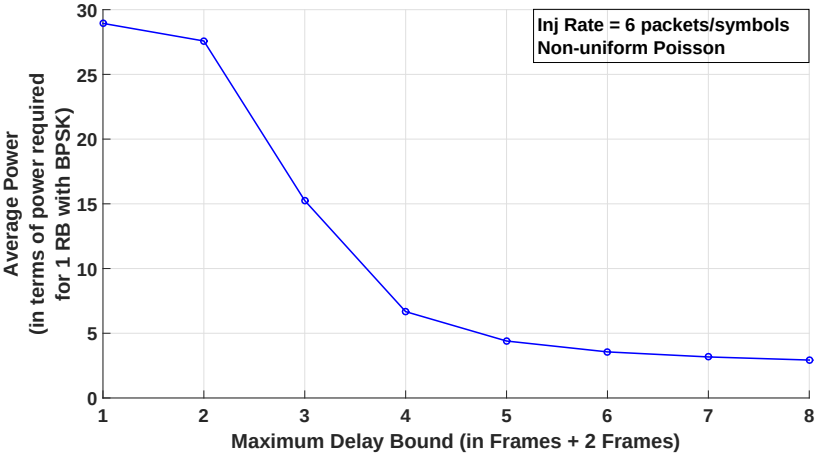
bound from 2 (2+2) frames to 8 (8+2) frames we can save up to 8 times of energy. As a last point, also note that with bursty traffic, the average power expenditure is significantly higher for all different delay bounds compared to Poisson traffic; as higher modulation orders are selected much more frequently under fluctuating traffic demands.

Injection Rate = 6 packets/symbol

We perform the same experiments by increasing total injection rate to 6 packets/symbol. Fig. 7.14 shows the packet delay exceeding probabilities for the static modulation EQPS order case for different modulation orders under if statically used. Fig 7.15(a) shows the propbability of delay exceeding for different delay bounds for non-uniform Poisson and DPBPP traffic, under 6 packets/symbol total injection rate. Similarly, only the bound of algorithm with delay bound of 1 frame is violated. We observe from Fig. 7.15(b) that, by decreasing maximum delay bound from 1 (1+2) frames to 4 (4+2) frames we



(a) Packet delay exceeding probability graph. For each tried maximum delay bound (in frames + 2 frame length addition).



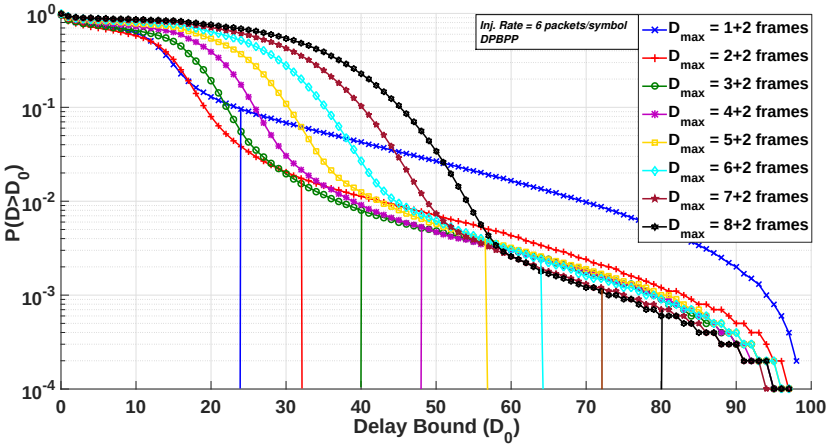
(b) The resulting average power in terms of transmission power required for a single RB with BPSK, for each tried maximum delay bound in frames. (Actual delay bound is 2 frames more)

FIGURE 7.15: Decentralized maximum delay bounded scheduler performance under non-uniform Poisson traffic with injection rate of 6 packets/symbol (EQPS($\alpha = 0.95$), time direction allocation, T=8 symbols)

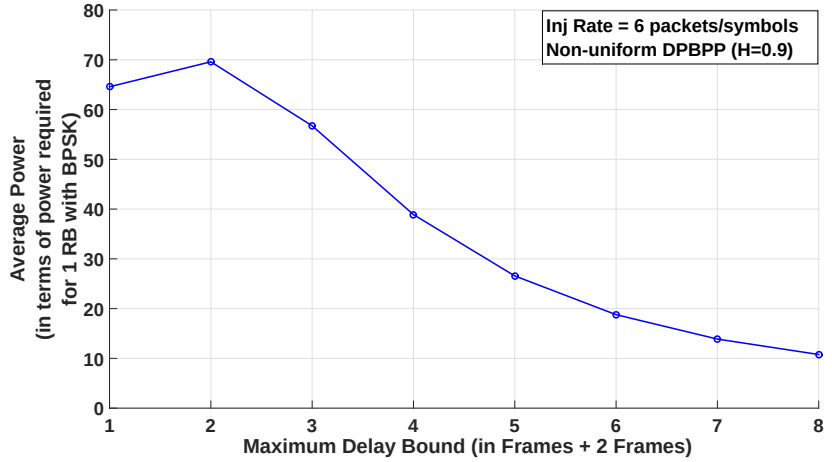
can save up to 5 times of energy. And next, in Fig. 7.15(a), exceeding probabilities with different delay bounds for non-uniform DPBPP traffic is evaluated. This time, all the tried delay bounds are violated and have probabilities of violation between 0.1 and 0.01. As expected, when the average traffic load and temporal burstiness increase, performance of the algorithm degrades. By increasing delay bound from 1 (1+2) frame to 8 (8+2) frames, up to 7 times of average power can be saved.

Injection Rate = 8 packets/symbol

And lastly, we evaluate the proposed maximum delay bounded modulation order selection and bandwidth allocation algorithm for a total injection rate of 8 packets/symbol. Fig. 7.17 shows the packet delay exceeding probabilities for the static modulation orders



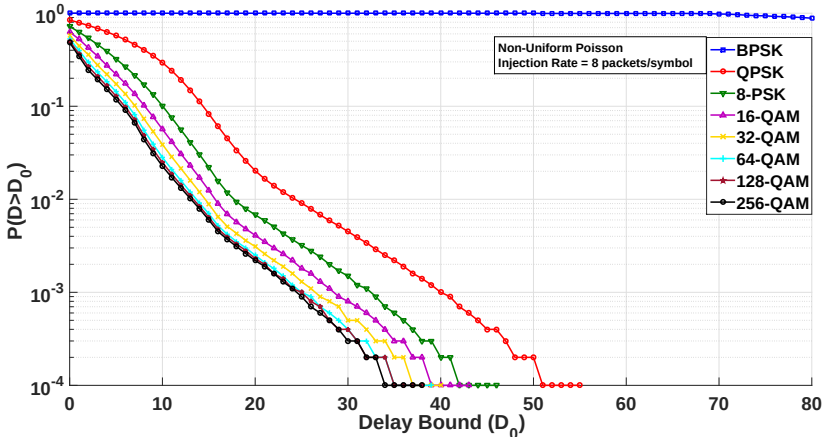
(a) Packet delay exceeding probability graph. For each tried maximum delay bound (in frames + 2 frame length addition).



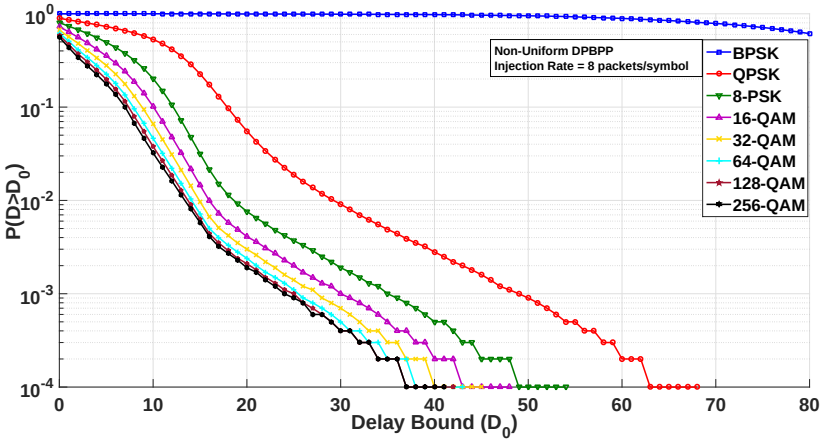
(b) The resulting average power in terms of transmission power required for a single RB with BPSK, for each tried maximum delay bound in frames. (Actual delay bound is 2 frames more)

FIGURE 7.16: Decentralized maximum delay bounded scheduler performance under under non-uniform DPBPP traffic (H=0.9) with injection rate of 6 packets/symbol (EQPS($\alpha = 0.95$), time direction allocation, T=8 symbols)

EQPS order case for different modulation orders under. Fig 7.18(a) shows the propbability of delay exceeding for different delay bounds for non-uniform Poisson and DPBPP traffic, under 8 packets/symbol total injection rate. First of all, we highlight this important point : theoretically, our interconnect should support a total injection rate up to 45 packets/symbol with highest modulation order of 256-QAM. However, we have seen that proposed maximum delay bounded algorithm fails for every given delay bound for total injection rates higher approximately 8 packets/symbol. This stems from the exact same reason behind the violations observed in previous experiments. Due to framed nature of the algorithm and limited rate, most of the time certain traffic peaks cannot be served, thus queue dynamics fail. However, we again note that, this algoirthm may provide an efficient scheduling for energy expenditure minimization, especially under lower traffic



(a) Non-Uniform Poisson



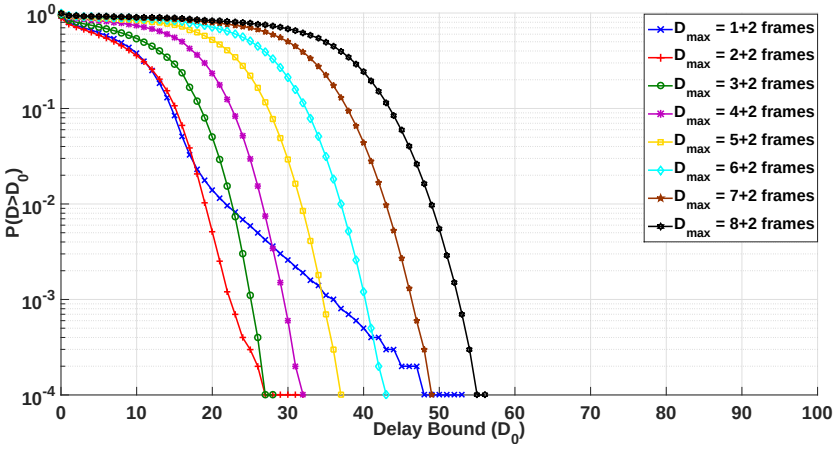
(b) Non-Uniform DPBPP (H=0.9)

FIGURE 7.17: Packet delay exceeding probability graphs for non-uniform Poisson and DPBPP traffic with injection rate of 8 packets/symbol for a static modulation order system for different utilized modulation orders (EQPS($\alpha = 0.95$), time direction allocation, T=8 symbols)

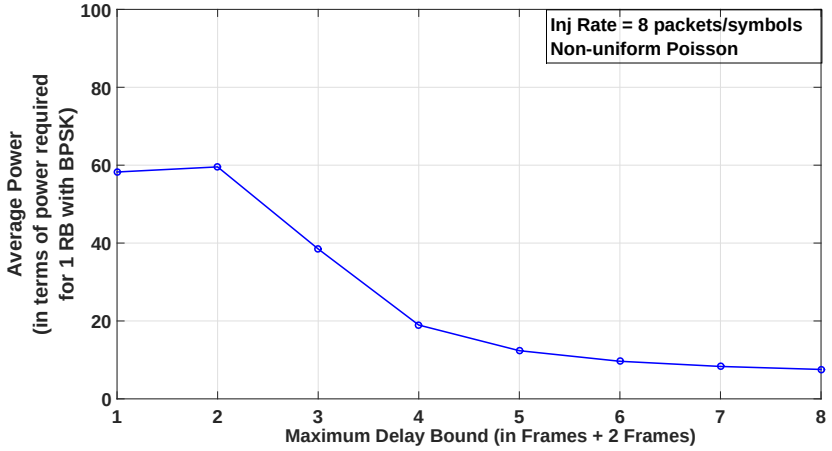
loads and heterogeneity.

In Fig. 7.18(a), delay exceeding probabilities for algorithms with different maximum delay bounds are shown, for a total injection rate of 8 packets/symbol under non-uniform Poisson traffic. For another time, we observe that the algorithm with 1 (1+2) frame length bound is far to provide the desired performance. Fig. 7.18(b), shows that we can decrease the average power by 3 times, by increasing delay bound from 2 (2+2) frames to 4 (4+2) frames.

And finally, we evaluate our algorithm under non-uniform DPBPP traffic. Fig. 7.19(a) shows that all intended delay bounds are violated with a probability of between 0.1 and 0.01, which is quite undesired. However, we can still claim, even under highly bursty traffic and higher loads, this algorithm can save power while providing not a strict delay bound, but a reasonable expectation of maximum delay violation. Fig. 7.19(b) shows



(a) Packet delay exceeding probability graph. For each tried maximum delay bound (in frames + 2 frame length addition).



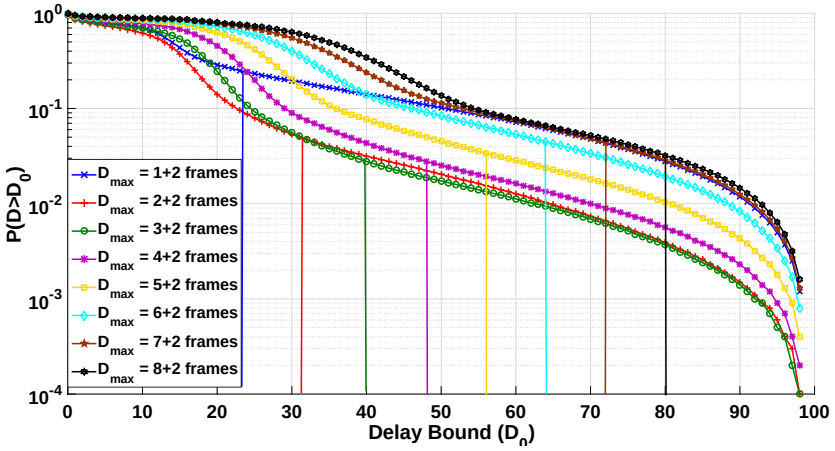
(b) The resulting average power in terms of transmission power required for a single RB with BPSK, for each tried maximum delay bound in frames. (Actual delay bound is 2 frames more)

FIGURE 7.18: Decentralized maximum delay bounded scheduler performance under non-uniform Poisson traffic with injection rate of 8 packets/symbol (EQPS($\alpha = 0.95$), time direction allocation, T=8 symbols)

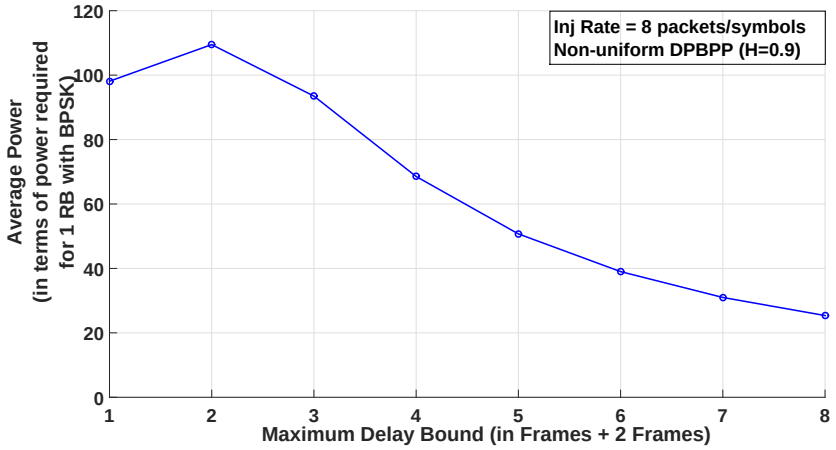
that, we can save approximately 3 times of power by reducing the maximum delay bound from 2 (2+2) frames to 7 (7+2) frames.

7.5 Average Delay Bounded Scheduler

We propose to develop another modulation order scheduler for WiNoCoD, which is based on [129]. This time the main motivation is to minimize transmission power, while setting a bound on *average delay* of the packets, which is a more valuable metric of interest, as mentioned previously. [129] presents a scheduler for a single user single AWGN channel under bursty arrivals, which gives the minimum transmission power, while setting a bound on average delay. Optimal solution includes complex processes and operations.



(a) Packet delay exceeding probability graph. For each tried maximum delay bound (in frames + 2 frame length addition).



(b) The resulting average power in terms of transmission power required for a single RB with BPSK, for each tried maximum delay bound in frames. (Actual delay bound is 2 frames more)

FIGURE 7.19: Decentralized maximum delay bounded scheduler performance under under non-uniform DPBPP traffic (H=0.9) with injection rate of 8 packets/symbol (EQPS($\alpha = 0.95$), time direction allocation, T=8 symbols)

In addition, in order to apply these stochastic optimization techniques, unrealistic assumptions on arrival traffic have to be made, such as no temporal correlation.

The authors make a remarkable observation on the optimal scheduler that rate (modulation order) is chosen approximately proportional to natural logarithm of instantaneous queue state. They explain this behavior with the tendency of scaling rate linearly proportional to consumed power (Note that, required transmission power is proportional to exponential of scheduled rate). Then, they propose a near optimal heuristics based scheduler called *log-linear scheduler*, which chooses the modulation order proportional to instantaneous queue state. To perform this, they multiply queue state before taking its natural logarithm, by a scalar κ, which is incremented by a small step size if the current estimated average delay is larger than the desired average delay bound. Similarly, if

TABLE 7.1: Probability of exceeding the intended delay bound of the algorithm for different traffic intensities and models.

	Given Delay Bound							
Probability of Exceding Delay Bound	**1+2 Frames (24 symbols)**	**2+2 Frames (32 symbols)**	**3+2 Frames (40 symbols)**	**4+2 Frames (48 symbols)**	**5+2 Frames (56 symbols)**	**6+2 Frames (64 symbols)**	**7+2 Frames (72 symbols)**	**8+2 Frames (80 symbols)**
Inj. Rate = 4 pkts/sym (Non-Uni. Poiss.)	0.0003	0	0	0	0	0	0	0
Inj. Rate = 4 pkts/sym (DPBPP)	0.0413	0.0067	0.0025	0.0012	0.0005	0.0002	0.0001	0
Inj. Rate = 6 pkts/sym (Non-Uni. Poiss.)	0.0017	0	0	0	0	0	0	0
Inj. Rate = 6 pkts/sym (DPBPP)	0.0958	0.0176	0.008	0.0053	0.0039	0.0024	0.0015	0.0007
Inj. Rate = 8 pkts/sym (Non-Uni. Poiss.)	0.0069	0.0001	0	0	0	0	0	0
Inj. Rate = 8 pkts/sym (DPBPP)	0.2409	0.0473	0.0278	0.0278	0.0339	0.0446	0.0441	0.0321

TABLE 7.2: Achieved average power with the intended delay bound in terms of power required for 1 RB with BPSK for different traffic intensities and models.

	Given Delay Bound							
Average Power	**1+2 Frames (24 symbols)**	**2+2 Frames (32 symbols)**	**3+2 Frames (40 symbols)**	**4+2 Frames (48 symbols)**	**5+2 Frames (56 symbols)**	**6+2 Frames (64 symbols)**	**7+2 Frames (72 symbols)**	**8+2 Frames (80 symbols)**
Inj. Rate = 4 pkts/sym (Non-Uni. Poiss.)	10.52	8.72	4.09	1.97	1.46	1.27	1.17	1.11
Inj. Rate = 4 pkts/sym (DPBPP)	36.77	38.51	29.88	18.97	11.85	7.74	5.4	3.97
Inj. Rate = 6 pkts/sym (Non-Uni. Poiss.)	28.94	27.58	15.25	6.67	4.4	3.56	3.17	2.93
Inj. Rate = 6 pkts/sym (DPBPP)	64.58	69.6	56.73	38.89	26.57	18.78	13.88	10.78
Inj. Rate = 8 pkts/sym (Non-Uni. Poiss.)	58.25	59.56	38.49	18.95	12.37	9.65	8.32	7.55
Inj. Rate = 8 pkts/sym (DPBPP)	98.11	109.48	93.49	68.53	50.72	39.01	30.99	25.37

the currently estimated average delay meets the desired bound, κ is decremented by a small step size. As κ gets larger, the resulting logarithm would be larger, thus choosing higher rates. This way, the minimum transmission power is tried to be achieved while respecting the average delay bound. This near-optimal scheduler hugely simplifies modulation order selection procedure, as it just requires few number of basic mathematical operations.

7.5.1 Average Delay Bounded Scheduling with Centralized Approach Based on EQPS

Using the strong efficiency of this computationally trivial tool, we would like apply it to WiNoCoD. But as previously stated, this scheduler is for the single-user, single-channel case, therefore it shall be adapted to multi-user, multi-channel nature of WiNoCoD. In

contrast with the previous section, average delay bounded modulation order selection algorithm is demonstrated with the centralized paradigm. The reason behind this is first to show the ability of using fully centralized intelligence including also modulation order selection and second is the nature of average delay bounded scheduler which enables centralized computation. The algorithm is executed as explained in 7.3.2, that the each tileset broadcasts its QSI and CIU calculates the number of RB allocated to each. Then using these information, CIU also calculates the optimal modulation order for each tileset, using average delay bounded scheduler. Finally, CIU broadcasts number of allocated RBs and selected modulation order for each tileset, 2 symbols before the end of the frame. Hence, tilesets can reconfigure their transmission patterns and powers at the start of next frame. Similarly for the maximum delay bounded algorithm, EQPS ($\alpha = 0.95$) in time direction is chosen to allocate frequency RBs.

The idea of the algorithm is straightforward; as in [129], modulation order (thus rate) is chosen proportional to natural logarithm of the QSI for each tileset. However, different from this, a tileset shall have different number of RBs (each can serve 32 bits flit with BPSK, 2 flits with QPSK, 3 flits with 16-QAM, .. , 8 flits with 256-QAM). Hence, if there are S RBs allocated to a tileset, and the chosen modulation rate can serve M flits (32 bits) per RB, the total service rate of this tileset will be MS through the whole frame. Therefore, we have modified the algorithm such that CIU choose the modulation order of tileset i, proportional to the natural logarithm of its QSI divided by the number of RBs allocated :

$$M_i^{t+1} = \left\lceil log\left(\frac{\kappa_i \hat{Q}_i^t}{S_i^{t+1}}\right)\right\rceil \tag{7.7}$$

Just as in [129], κ_i is a dynamic scalar in order to achieve the desired average delay bound D_{avg}. CIU keeps a κ_i value for each tileset and updates every frame. If currently calculated average delay $\hat{D}_i^t$ is under the delay bound it is decreased by a constant differential, $\Delta\kappa$ multiplied by the difference $(\hat{D}_i^t - D_{avg})$ to tune the algorithm. This way, the chosen modulation order, therefore the transmission power can be minimized as much as possible given that we are not violating the average delay bound. If calculated instantaneous average delay sample is exceeding the D_{avg}, κ_i is increased by $\Delta\kappa(\hat{D}_i^t - D_{avg})$. As you can see, increasing κ_i values tend to choose higher modulation orders.

So in order to update κ_i, CIU has to compute instantaneous average delay sample for each tileset. As in the reference paper, *Little's Law* [129] is used to estimate average waiting time. Little's Law states that for any kind of queuing system with any kind of input process, division of average queue state by average number of packets arrive gives

the average waiting time in the system. Note that CIU keeps already average arrival rates (estimated number of flits per frame) with a exponential moving averaging filter. In addition to it, another 32 Exponentially Weighted Moving Average (EWMA) filters are kept for averaging QSIs. Then division of this two values gives the moving average of the estimated average delay of the tileset :

$$\hat{D}_i^t = \frac{\hat{Q}_i^t}{\hat{A}_i^t} \tag{7.8}$$

Then for each 32 tilesets, κ_i values are updated as follows :

$$\kappa_i = \kappa_i + \Delta\kappa(\hat{D}_i^t - D_{avg}) \tag{7.9}$$

Log-linear algorithm uses just basic mathematical operators to give a solution to complex average delay bounded power minimization problem, except the logarithm. This can be performed by a simple look-up table which decreases its latency to few cycles. And as for EQPS, division can be done by a simple reciprocal multiplication. For each tileset, CIU has to perform 1 float division (1 time division to calculate the reciprocal of $\hat{Q}_i^t$), 3 float multiplications, 1 division-by-2 (bit shifting) operation, 1 natural logarithm by look-up table and finally an upper-rounding.

In addition to this, the computation for estimated average delay and κ_i updates shall be performed. At first, as for (7.2), the 32 EWMA calculations for QSIs had to be done, which includes 2 float multiplications and 1 addition. Then 1 division is done for (7.6). Next, for κ_i updates, 1 float multiplication, 1 subtraction and 1 addition is done. Note that, these computations for estimating average delay and κ_i updates can be done in the period where tilesets reconfigure their transmissions (as for EWMA calculations of average arrivals). We remind parallelism can be exploited for these operations for 32 tilesets by employing multiple simple cores at CIU, which decreases required computation time further and further.

Fig. 7.20 illustrates the flow chart of the proposed average delay bounded scheduler inside the CIU, including the estimation of arriving number of flits in each tileset, estimation of instantaneous average delays and update of the κ values.

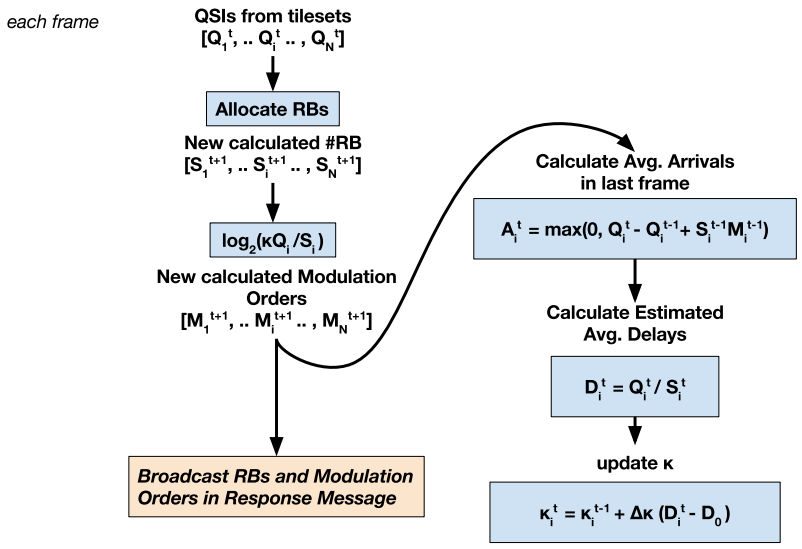


FIGURE 7.20: Flow chart of the average delay bounded scheduler algorithm executed at CIU every frame.

7.5.2 Experimental Evaluation

Now, we will test the performance of the proposed scheduler for various frame lengths and for each tried average delay bound from 1 to 20 symbols, and under different injection rates. Due to lack of space, only realistic and highly bursty non-uniform DPBPP traffic case is considered. Log-linear algorithm was shown to operate with a performance close to optimal scheduler for the single-user, single-channel case. One assumption that the authors had made, was the i.i.d. nature of the bursty arrival model. Therefore, with self-similarity we can accept to divert from optimality, however scaling transmission power linearly with increasing backlog shall still provide an effective delay-power trade-off.

Frame Length = 4+2 symbols

First, we evaluate results for T=4+2 symbols of frame length. Fig. 7.21, Fig. 7.22 and Fig. 7.23 shows the resulting actual average latency of the packets and resulting average power in terms of minimum transmission power required for 1 RB with BPSK encoding, with a 2-y plot, for an injection rate of 8, 12 and 16 packets/symbol respectively. As you can see from the figures, a diagonal line cuts the plot, which is placed in order to show if the resulting average delay is below the desired average delay bound.

Firstly, from the figures, we see that for all injection rates, one cannot guarantee an average delay bound lower than 4 symbols due to limited rate and nature of the traffic. Even the algorithm tries its best (with largest κ values) it can only provide the lowest

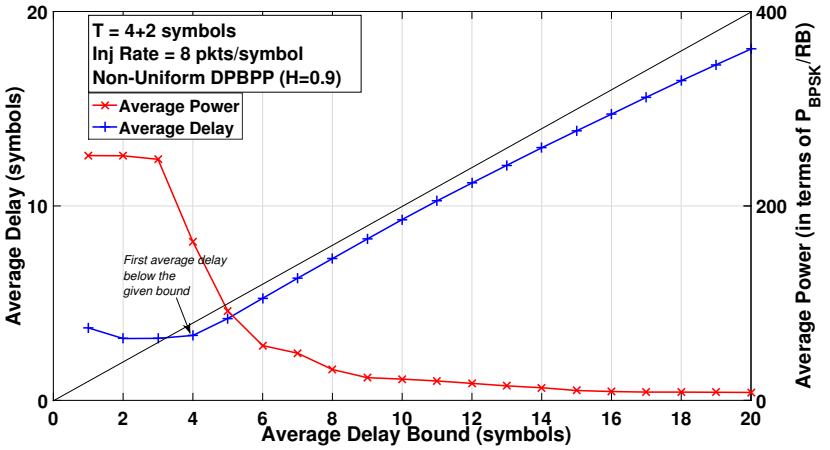


FIGURE 7.21: Resulting average delay and average power is shown with a 2-y plot, with T =4+2 symbols, under non-uniform DPBPP traffic under an injection rate of 8 packets/symbol.

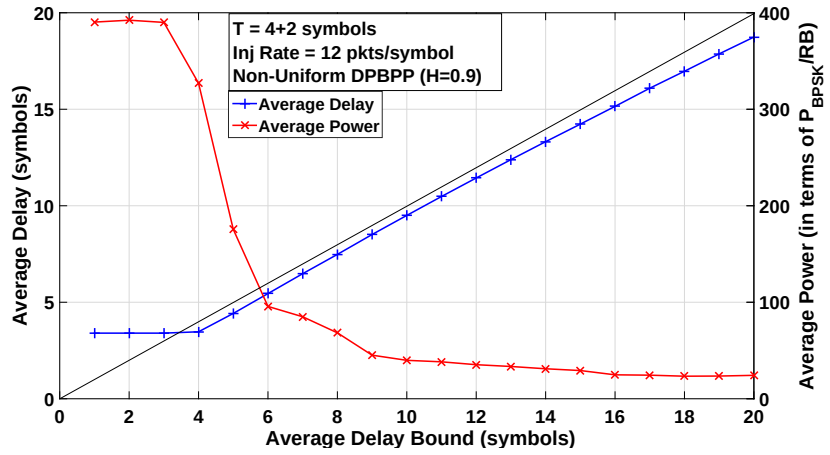


FIGURE 7.22: Resulting average delay and average power is shown with a 2-y plot, with T =4+2 symbols, under non-uniform DPBPP traffic under an injection rate of 12 packets/symbol.

possible average delay of approximately 3 symbols, with the highest transmission power, which is convenient. As we increase the average delay bound, we see that we are able to well provide the desired average latency just below the input bound. This shows the strong efficiency of the proposed algorithm. As we expected, increasing the average delay bound makes great power gains. Due to the exponential relation between power and rate, the Pareto like delay-power trade-off is evident in these cases as can be seen from the figures. Even though increasing the average delay provides substantial amount of energy, increasing further and further does not provide the same scale of return as mentioned previously. For instance, for injection rate of 16 packets/symbol, increasing average delay bound from 4 symbols to just 6 symbols, decreases the average power up to 4 times. As expected, with increasing injection rate required average power to provide

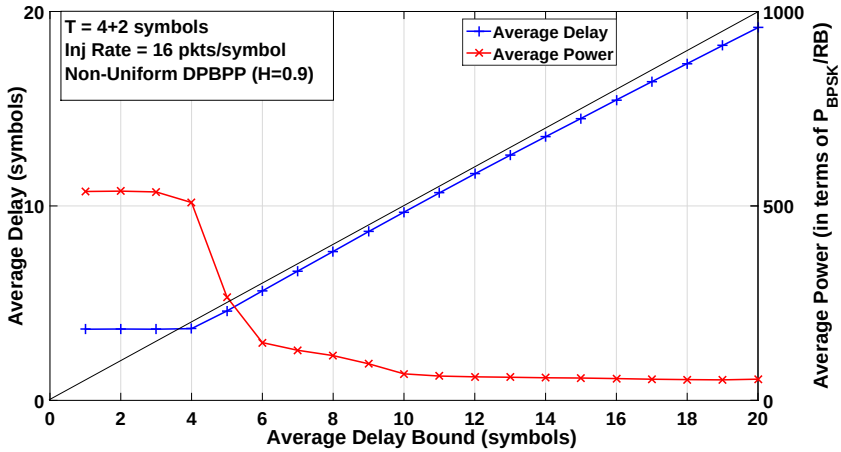


FIGURE 7.23: Resulting average delay and average power is shown with a 2-y plot, with T =4+2 symbols, under non-uniform DPBPP traffic under an injection rate of 16 packets/symbol.

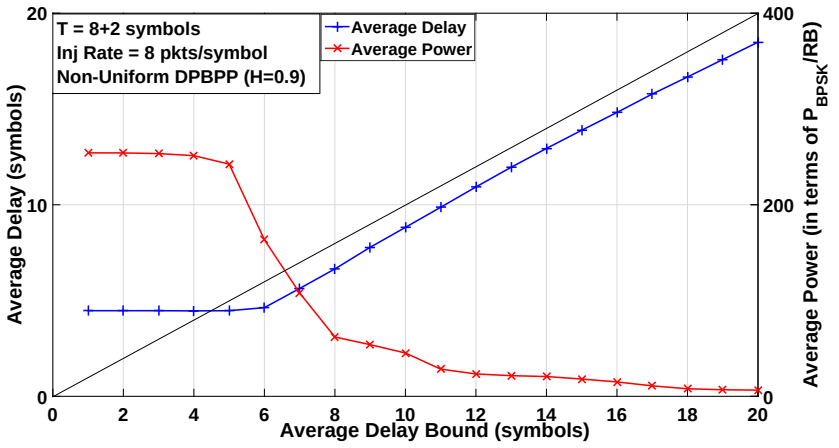


FIGURE 7.24: Resulting average delay and average power is shown with a 2-y plot, with T =8+2 symbols, under non-uniform DPBPP traffic under an injection rate of 8 packets/symbol.

the same average delay bound gets larger.

Frame Length = 8+2 symbols

Next, we continue our evaluation with longer frame lengths, in order to provide different options for computational power for WiNoCoD. Fig. 7.24, Fig. 7.25 and Fig. 7.26 shows the same performance graphs for T=8+2 symbols, under injection rates of 8, 12 and 16 packets/symbol respectively.

We see similar performance for T=8+2 symbols, but with a slightly degraded performance due to more outdated QSI and more infrequent rate re-allocation. Now, average delay bounds lower than 7 symbols cannot be guaranteed. Similarly, after this point increasing average delay bound a few symbols can provide dramatically lower energy

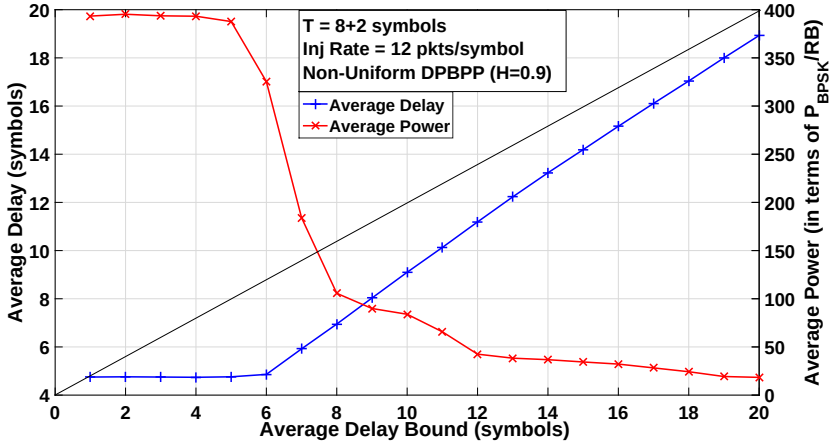


FIGURE 7.25: Resulting average delay and average power is shown with a 2-y plot, with T =8+2 symbols, under non-uniform DPBPP traffic under an injection rate of 12 packets/symbol.

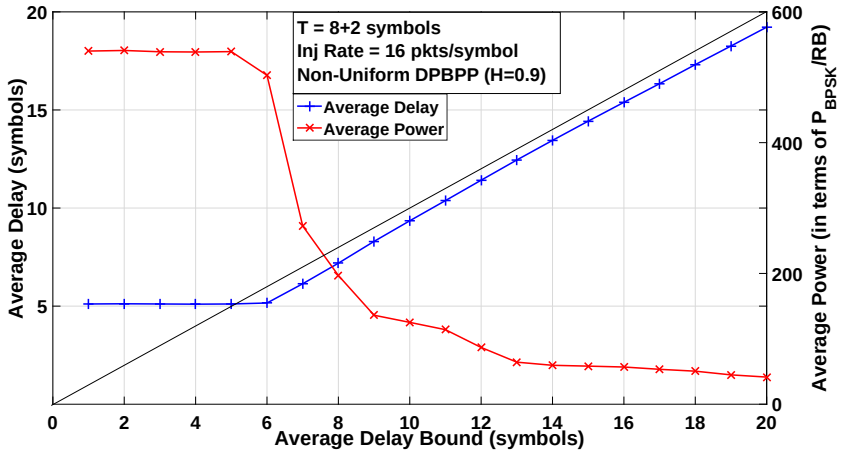


FIGURE 7.26: Resulting average delay and average power is shown with a 2-y plot, with T =8+2 symbols, under non-uniform DPBPP traffic under an injection rate of 16 packets/symbol.

expenditure. Also as expected, with longer frame length, to attain the same average delay bound, we have to spend more power for the same injection rate.

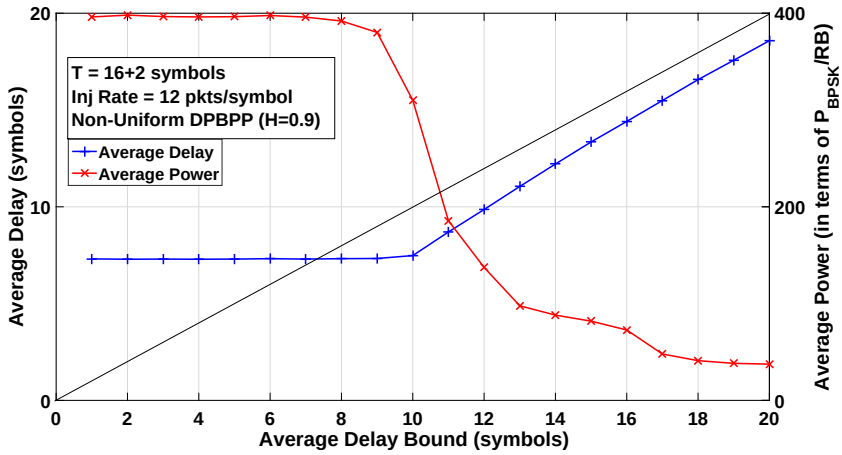


FIGURE 7.28: Resulting average delay and average power is shown with a 2-y plot, with T =16+2 symbols, under non-uniform DPBPP traffic under an injection rate of 12 packets/symbol.

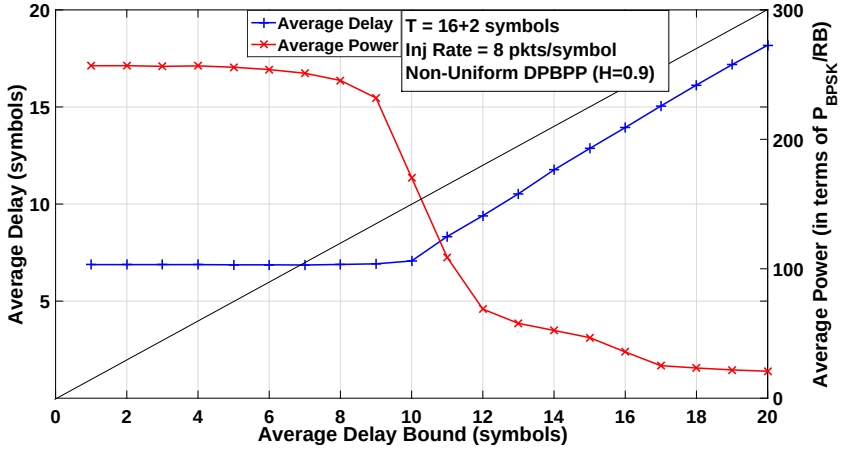


FIGURE 7.27: Resulting average delay and average power is shown with a 2-y plot, with T =16+2 symbols, under non-uniform DPBPP traffic under an injection rate of 8 packets/symbol.

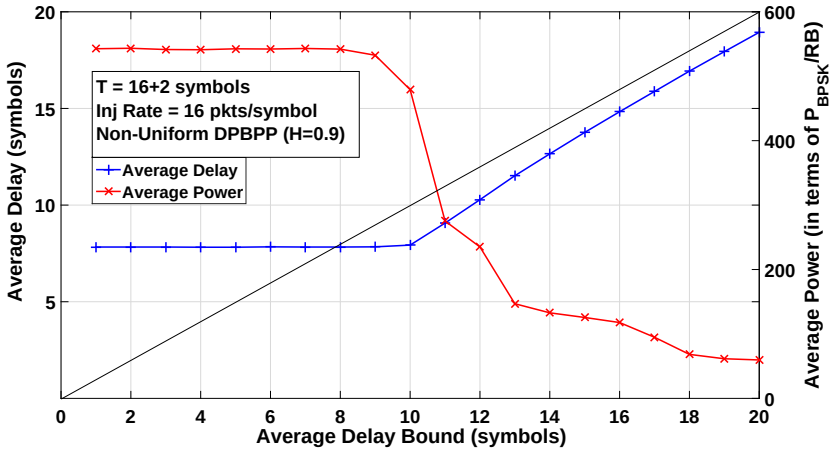


FIGURE 7.29: Resulting average delay and average power is shown with a 2-y plot, with T =16+2 symbols, under non-uniform DPBPP traffic under an injection rate of 16 packets/symbol.

Frame Length = 16+2 symbols

And finally, the simulations are executed for T=16+2 symbols. As frame lengths gets even longer, performance continues to degrade as it can be seen from Fig. 7.27, Fig. 7.28 and Fig. 7.29.

7.6 Information Theoretic Study of the WiNoCoD RF Interconnect

Previously in this chapter, we have proposed intelligent modulation order selection policies, which increases modulation orders only when necessary, to keep delays of packets bounded. The primary motivation behind this was the exponentially increasing energy expenditure with the modulation orders. We have calculated the required minimum transmission powers according to Shannon's capacity formula [130]. As mentioned previosuly, capacity defines the maximum trasmission rate on a channel, guaranteeing a probability of error approaching to 0, with any protective, reconstructive channel coding.

We have experimented our proposed algorithms by comparing the resulting average power to the case where the lowest modulation order (BPSK) would be utilized constantly. This has provided us an insight on how much energy can be saved while satisfying certain service demands.

In this section, we intend to determine minimum required transmission powers (in terms of dBm) for defined information theoretic capacities. For instance, defining transmission

powers for spectral capacity densities as 1 bits/sec/Hz, 2 bits/sec/Hz, etc. may provide a good perspective for the respective modulation orders BPSK, QPSK, etc.. For that we make the hypothesis that the transceiver access via transistors to the microstrip transmission line provides a relatively non-varying (non-frequency selective) attenuation over the 20-40 GHz spectrum, in Section 3.3.2.3. If this hypothesis is strong for the via transistor access proposed by M. Hamieh, Fig. 3.12 shows that it is valuable for capacitive coupling but with more attenuation.

In this section, firstly we examine the topology of a U-shaped transmission line, which was previously proposed in the first phase of the project and then we evaluate the cross-shaped transmission line, which was chosen finally due to its much better performance. We will try to dimension the required powers for different communication configurations between tilesets for having spectral capacity densities corresponding to modulation orders. These configurations are unicast communication occuring only between two arbitrary tilesets and the broadcast communication, where a tileset's transmission shall achieve the necessary capacity density even to the farthest node (Any destination tileset should receive packets sent from this tileset with a minimum defined capacity density). In addition to these two configurations, required power for a tileset to achieve an overall capacity density (overall of capacity capacity densities to all other 31 tilesets).

M. Hamieh has determined that attenuation in the transmission line varies between -0.20 and -0.30 dB/mm over 20-40 GHz spectrum, thus without loss of generality we have chosen to set an average attenuation of -0.25 dB/mm. Wired RF provides not only the advantage of CMOS compatibility, but also significantly less power attenuation compared to wireless interconnects.

The adjacent transceivers have 8 mm spacing (thus, the signal power attenautes 2 dB between two neighboring tilesets.) And we have assumed a 1 mm of distance for facing tilesets (such as tilesets 1-9, 2-10 etc. in Fig. 7.30). Before this, we revisit the well known information theoretic capacity formula of Shannon, defining the highest achievable transmission rate on a channel ensuring a lossless communication, C_0 in bits/s:

$$C_0 = Blog_2(1 + SNR) \tag{7.10}$$

where, B is bandwidth in Hz and SNR is Signal-to-Noise Power Ratio in linear. The power of the noise depends on the ambient temperature and bandwidth. For our calculations, we will assume the equally shared scheme where each tileset is allocated 32 subcarriers, thus 640 MHz, uniformly. In other words SNR is the ratio of received signal power P_R to the noise power $\frac{P_R}{P_N}$. It's assumed that noise power P_N has a spectral density of -174 dBm/Hz at room temperature [136], which we also accept for our calculations.

-174 dBm/Hz is equal approximately to $4\,10^{-21}$ W/Hz in linear scale. Therefore, we can rewrite the capacity formula as :

$$C_0 = Blog_2(1 + \frac{P_R}{B\,4\,10^{-21}}) \tag{7.11}$$

Assuming the only attenuation of transmitted signal is due to the losses with distance through the transmission line, which is -0.25 dB/mm, we can rewrite the capacity formula as a function of transmission power :

$$C_0 = Blog_2(1 + \frac{P_T}{10^{0.025d}\,B\,4\,10^{-21}}) \tag{7.12}$$

where, d is distance in mm.

Capacity density in bits/sec/Hz, $C = C_0/B$, is also used, which defines the achievable transmission rate per bandwidth.

By inverting these formulas, one can also find the required minimum transmission power for our tilesets for a desired capacity density :

$$P_T = 10^{0.025d}\,B\,4\,10^{-21}\,(2^C - 1) \tag{7.13}$$

Required Minimum Transmission Power for desired Bit Error Rate

As mentioned above, information theoretic capacity is a tool to dimension the commmunication on a channel, defining the upper bound of transmission rate, where probability of error approaches to 0. This is an analytical bound and does not imply any method how to achieve this rate. For instance, theoretically a channel correcting code can achive this rate, where probability of error at the end approaches to 0. As the channel frequency response is flat, we can make an AWGN hypothesis whatever the subcarrier is in 20-40 GHz band. Under this AWGN condition, we can derive the probability of bit error, or with a more common reference, Bit Error Rate (BER) with given transmission power, noise power and transmission rate on a *uncoded* channel. However, it is important to highlight this point : if we use the minimum transmission power for a desired capacity on these BER formulas, we would not get a probability of error of 0. This is because, capacity defines the maximum rate for a coded channel. However, defining required minimum transmission powers for various BER values is still important.

For BPSK and QPSK, the BER or probability of bit error (p_b) can be written as [137]:

$$p_b = Q(\sqrt{2\,SNR_b}) \tag{7.14}$$

where $Q(.)$ is the Gaussian tail function and SNR_b is the received SNR per bit. For instance if we are using QPSK (2 bits per constellation symbol) on a bandwidth of 640 MHz, SNR_b is calculated by dividing SNR value to 1320 Mbits/s.

For square M-QAM constellation symbols (such as 16-QAM, 64-QAM ..), p_b can be written approximately as [137]:

$$p_b = Q\left(\sqrt{\frac{3\,SNR_b\,b}{2^b - 1}}\right) \tag{7.15}$$

where b is the number of bits per constellation such as 4 bits for 16-QAM, 8 bits for 256-QAM etc.

Let us calculate the SNR per bit at first. As we did in capacity formula in (7.12), we can write the received SNR as the ratio of transmission power to ambient noise power and attenuation by distance. For BPSK and QPSK :

$$p_b = Q\left(\sqrt{\frac{2\,P_T}{10^{0.025d}\,B\,4\,10^{-21}}}\right) \tag{7.16}$$

and for M-QAM :

$$p_b = \frac{4}{b} Q\left(\sqrt{\frac{3\,b\,P_T}{10^{0.025d}\,B\,4\,10^{-21}\,(2^b - 1)}}\right) \tag{7.17}$$

By inverting these equations we can calculate the required minimum transmission powers with a given BER. The minimum required transmission power for BPSK and QPSK :

$$P_T = 0.5\,10^{0.025d}\,B\,4\,10^{-21}(Q^{-1}(p_b)^2) \tag{7.18}$$

and minimum required transmission power for M-QAM constellations :

$$P_T = \frac{1}{3}\,10^{0.025d}\,B\,4\,10^{-21}(Q^{-1}(0.25\,p_b\,b))^2\,(2^b - 1) \tag{7.19}$$

7.6.1 U-Shaped Transmission Line

The first topology proposed was a U-shaped transmission line as in Fig. 7.30. Not having a circular loop (close ends) avoids interfering reflections of the tranmitted signal.

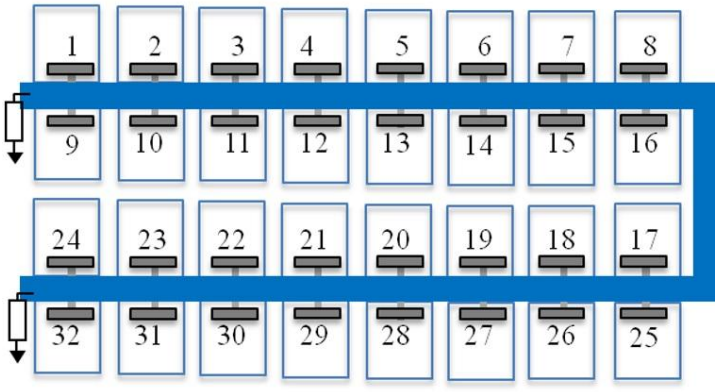


FIGURE 7.30: U-shaped transmission line

7.6.1.1 Unicast communication

Fig. 7.31 shows the distances from any tileset to any other tileset in U-shaped transmission line (32x31 combinations of unicast communications). As it can be seen from Fig. 7.31 that the largest distance (between tilesets at the opposite ends of the transmission line) is 120 mm, which causes an attenuation of -40 dB. Even though our OFDMA based RF interconnect is intended to provide a fully broadcast capable communication infrastructure, firstly we have analyzed the information theoretic limits for each of the 32x31 unicast communication combination between tilesets.

For these unicast communication combinations we can write the capacities in a matrix form for source-destination tileset pairs, C_0^{ij}, where N is the number of tilesets :

$$\begin{bmatrix} 0 & Blog_2(1+\frac{P_T}{10^{0.025d_{12}}B4\,10^{-21}}) & Blog_2(1+\frac{P_T}{10^{0.025d_{13}}B4\,10^{-21}}) & . & . & Blog_2(1+\frac{P_T}{10^{0.025d_{1N}}B4\,10^{-21}}) \\ Blog_2(1+\frac{P_T}{10^{0.025d_{21}}B4\,10^{-21}}) & 0 & Blog_2(1+\frac{P_T}{10^{0.025d_{23}}B4\,10^{-21}}) & . & . & Blog_2(1+\frac{P_T}{10^{0.025d_{2N}}B4\,10^{-21}}) \\ Blog_2(1+\frac{P_T}{10^{0.025d_{12}}B4\,10^{-31}}) & Blog_2(1+\frac{P_T}{10^{0.025d_{32}}B4\,10^{-21}}) & 0 & . & . & Blog_2(1+\frac{P_T}{10^{0.025d_{3N}}B4\,10^{-21}}) \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ & Blog_2(1+\frac{P_T}{10^{0.025d_{N1}}B4\,10^{-21}}) & Blog_2(1+\frac{P_T}{10^{0.025d_{N2}}B4\,10^{-21}}) & . & . & 0 \end{bmatrix}$$

Assuming a transmission power of -80 dBm per tileset the received capacities for unicast communication combinations are shown in Fig. 7.32. Note that, with this transmission power while neighboring nodes can achieve capacities up to 3 Gbps on their 640 MHz

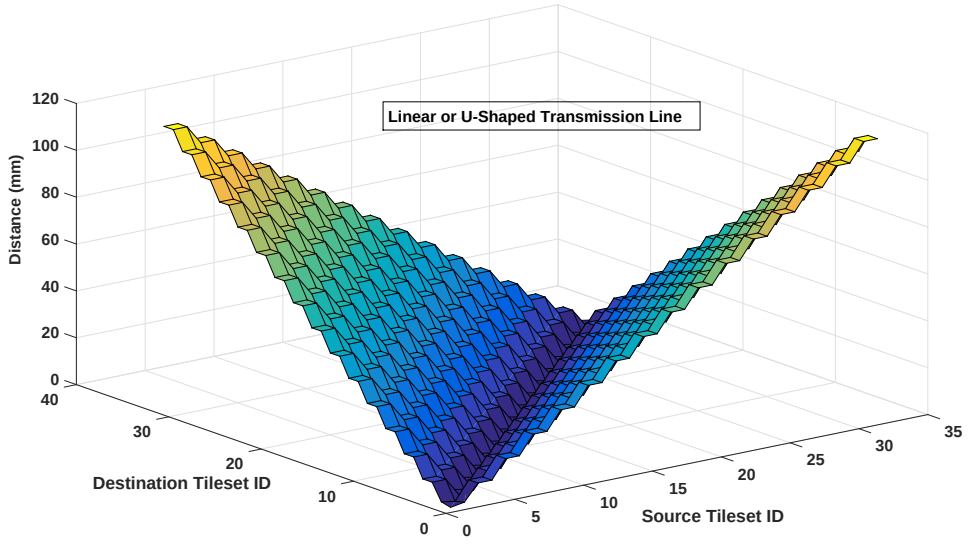


FIGURE 7.31: Distance between each 32x31 unicast communication in U-shaped transmission line

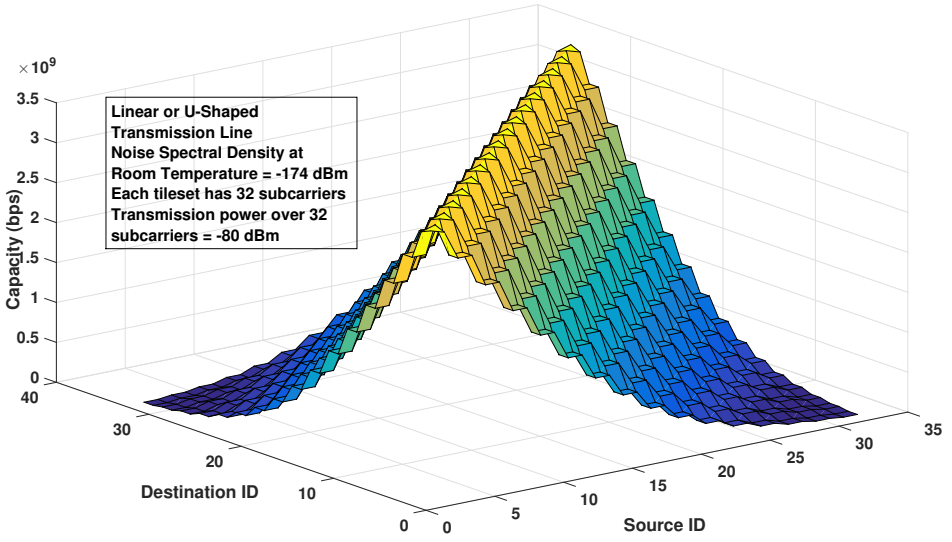


FIGURE 7.32: Capacity between each 32x31 unicast communication in U-shaped transmission line

bandwidth, the farthest nodes' capacity approaches to 0. From an information theoretic perspective, this means that with this transmission power these nodes cannot have a reliable communication.

Next, based on this distances, we calculate the required transmission power for each of 32x31 unicast communication combination between tilesets. We assume that, each tileset uses 32 subcarriers (thus, 640 MHz) for transmission and the noise spectral density at standard room temperature is assumed (-174 dBm/Hz). Note that, the required

transmission power can be calculated simply by adjusting these results linearly with increasing or decreasing bandwidth (number of subcarriers). The required transmission powers for unicast source-destination tileset pairs as a function of distance and channel capacity spectral density is as follows :

$$\begin{bmatrix} 0 & P_T = 10^{0.025d_{12}}\, B\, 4\, 10^{-21}\, (2^C - 1) & P_T = 10^{0.025d_{13}}\, B\, 4\, 10^{-21}\, (2^C - 1) & . & . & P_T = 10^{0.025d_{1N}}\, B\, 4\, 10^{-21}\, (2^C - 1) \\ P_T = 10^{0.025d_{21}}\, B\, 4\, 10^{-21}\, (2^C - 1) & 0 & P_T = 10^{0.025d_{23}}\, B\, 4\, 10^{-21}\, (2^C - 1) & . & . & P_T = 10^{0.025d_{2N}}\, B\, 4\, 10^{-21}\, (2^C - 1) \\ P_T = 10^{0.025d_{31}}\, B\, 4\, 10^{-21}\, (2^C - 1) & P_T = 10^{0.025d_{32}}\, B\, 4\, 10^{-21}\, (2^C - 1) & 0 & . & . & P_T = 10^{0.025d_{3N}}\, B\, 4\, 10^{-21}\, (2^C - 1) \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ & P_T = 10^{0.025d_{N1}}\, B\, 4\, 10^{-21}\, (2^C - 1) & P_T = 10^{0.025d_{N2}}\, B\, 4\, 10^{-21}\, (2^C - 1) & . & . & 0 \end{bmatrix}$$

Based on the attenuation values due to distances in Fig. 7.31, the required transmission power for each of these unicast communication combinations in dBm are calcuated and shown in Fig. 7.33, for different spectral channel densities between 1 bits/sec/Hz and 8 bits/sec/Hz, corresponding to different modulation orders.

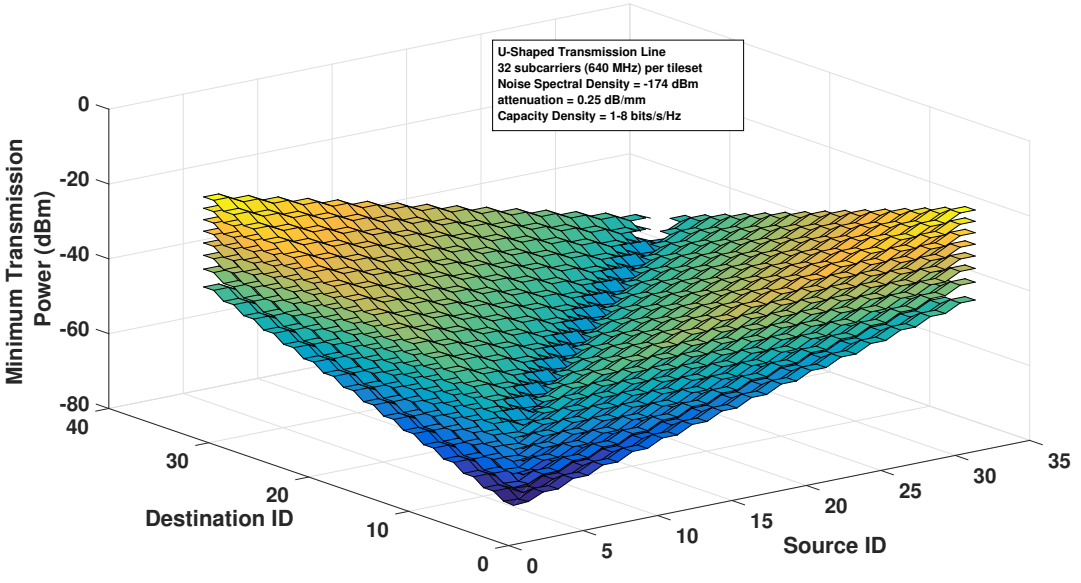


FIGURE 7.33: Required Transmission Power for each 32x31 unicast communication for the U-shaped shaped transmission line for capacity densities 1-8 bits/s/Hz

We have determined the required transmission powers for attaining the channel capacity densities for each possible 32x31 unicast communication. However, assuming that a tileset may communicate any other tileset throughout the execution of the application (not only one), providing an average value of the previously determined minimum transmission power values is utile. In other words, for each of 32 tilesets, we shall give an average of the unicast required transmission power values over each 31 distance tilesets as :

$$P_T^{avg}(i) = \frac{1}{31} \sum_{i \neq j, 1<i<32} P_T(d(i,j)) \tag{7.20}$$

Fig. 7.34 shows the average required transmission power for each tileset over its 31 destination tilesets, for the U-shaped topology, for capacity densities 1-8 bits/s/Hz. Note that, as tilesets position gets nearer to the center of transmission line (i.e. near tileset-16), required trasnmission power decreases and as its position gets nearer to the edges of transmission line (i.e. near tileset-1 or tileset-32), required transmission power increases. This is due to the fact that average distance to tilesets gets higher as we approach to the edges, and gets lower as we approach to the center. And as attenuation in dB increases linearly with distance, this significantly affects the transmission power. From this figure, we can understand that if a centralized mechanism is used, it is much more efficient to place the CIU inside the RF transceiver of Tileset-15, Tileset-16, Tileset-17 or Tileset-18, as they require much less transmission power due to their distances to other tilesets.

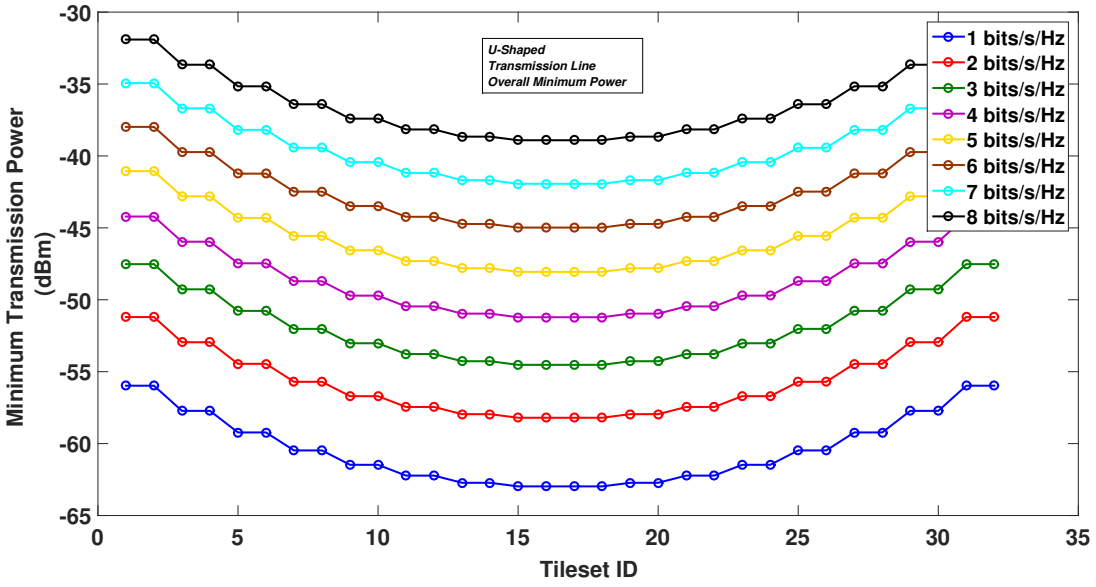


FIGURE 7.34: Average required transmission power (to 31 destinations) for each tileset for the U-shaped shaped transmission line for capacity densities 1-8 bits/s/Hz

Following this, we provide the average minimum transmission power of each tileset (average to 31 destinations), for BERs of $10^{-1}, 10^{-3}, 10^{-5}, 10^{-7}$ for BPSK and 256-QAM (the lowest and highest modulation orders). In order to have a reference, we also include the average required minimum transmission powers for information theoretic channel capacity densities of 1 bits/s/Hz and 8 bits/s/Hz, corresponding to BPSK and 256-QAM. Again, we remind that information theoretic capacity defines the maximum

rate on a coded channel where probability of error approaches to 0 (whereas, it does not define this hypothetical error correcting code explicitly). However, the average power values given here for different probabilities of error are for uncoded channels. Hence, it is possible that we might need larger minimum transmission powers for certain probabilities of error larger than 0, compared to channel capacities (where probability of error approaches to 0).

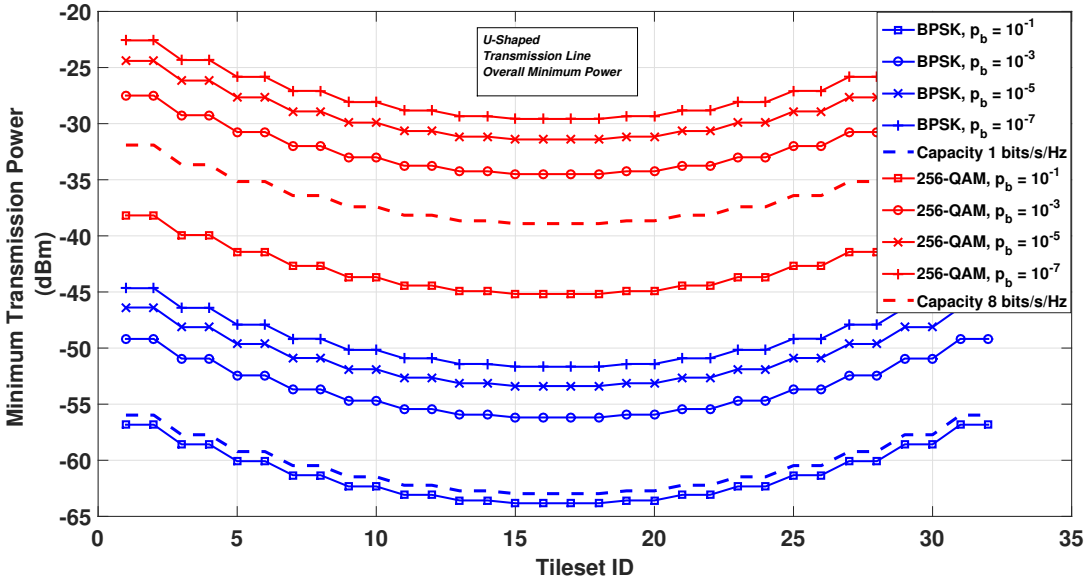


FIGURE 7.35: Average required transmission power (to 31 destinations) for each tileset for the U-shaped shaped transmission line for probabilities of error : 10^{-1}, 10^{-3}, 10^{-5}, 10^{-7} for BPSK and 256-QAM

Observing Fig. 7.35, we see that for BPSK, for a probability of error of 10^{-1}, tilesets require a little bit less average transmission power compared to power required for information theoretic channel capacity of 1 bits/s/Hz. For probabilities of error of 10^{-3}, 10^{-5} and 10^{-7}, we see that we need higher transmission powers. Even though these probability of errors are larger than 0, they require larger powers as these formulas are for uncoded transmission. Of course, in order to obtain results in accordance of intra-chip communication requirements, channel coding should be added. Hence, for a BER of 10^{-7} before decoding, an order of 10^{-14} can be achieved after usual realistic coding; but this is out of scope of this study.

7.6.1.2 Broadcast communication

As we have reviewed in this thesis previously, tilesets broadcast their packets and the other tilesets check the associated flags in header flits to understand whether this packet

is for it or not. Therefore, we must ensure a reliable communication for packets concerning every other possible destination. In addition to this, we have mentioned that the on-chip traffic possess high amount of broadcast packets. Hence, determining required minimum transmission power by concerning a capacity density even to the farthest destination tileset is essential. Fig. 7.36 shows the required transmission powers for each tileset to its farthest destination, for the U-shaped topology, for capacity densities 1-8 bits/s/Hz. As we have stated in previous section, as tilesets position gets more to the center of the transmission line, its distance to the farthest destination node also decreases, which is decreasing the required transmission power significantly.

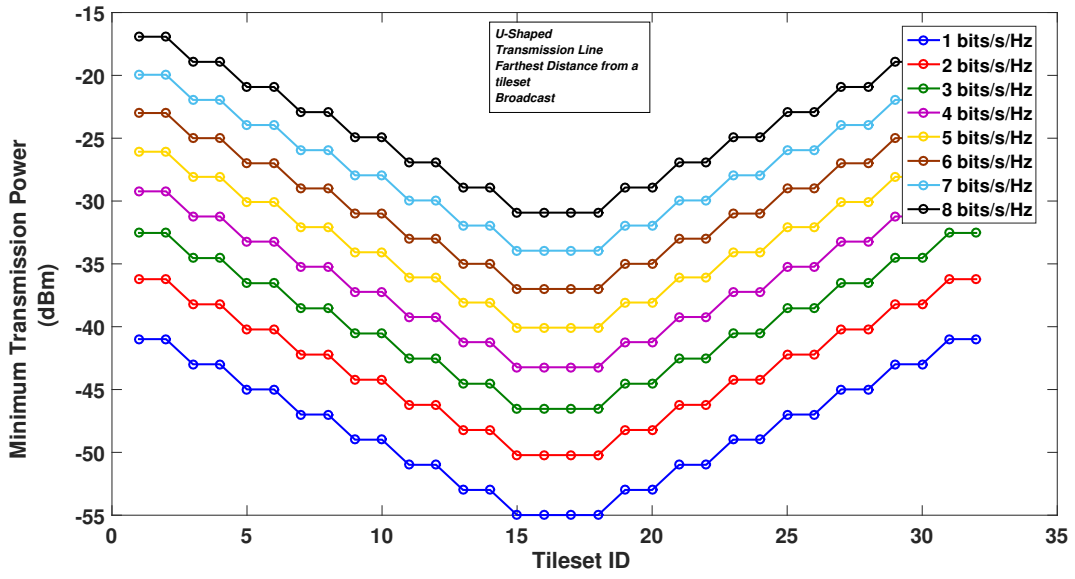


FIGURE 7.36: Minimum required transmission power for broadcasting (to maximum distance) for each tileset for the U-shaped shaped transmission line for capacity densities 1-8 bits/s/Hz

Fig. 7.37 shows the required minimum transmission powers for tilesets for broadcasting (to farthest destination node), for probabilities of error of 10^{-3}, 10^{-5} and 10^{-7}, under BPSK and 256-QAM, compared to powers required for corresponding channel capacities 1 bits/s/Hz and 8 bits/s/Hz. We see a similar pattern as we observed for average unicast transmission power values.

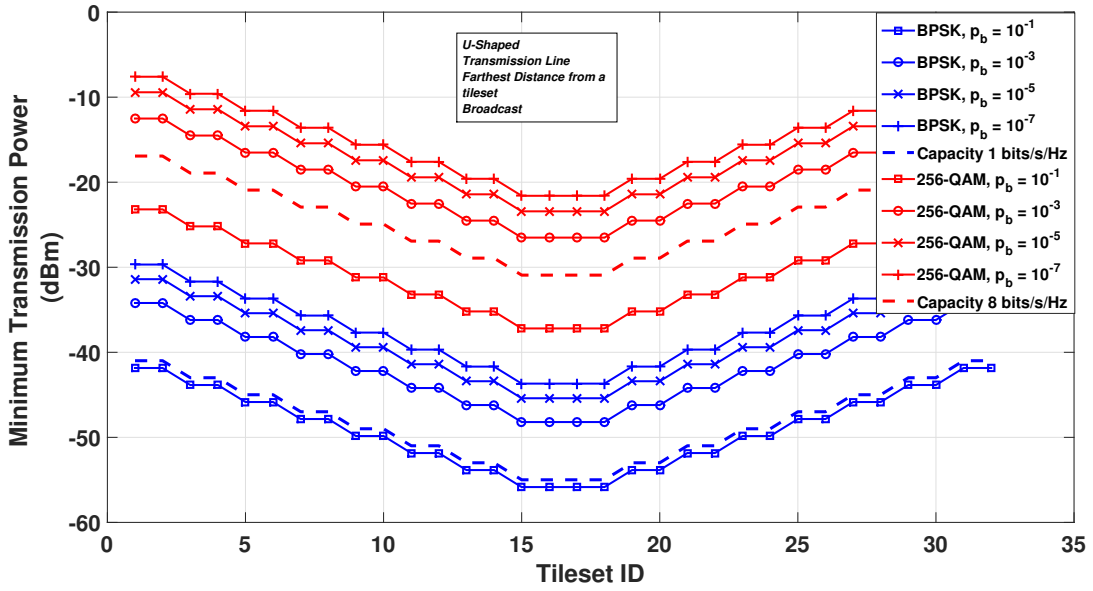


FIGURE 7.37: Required transmission power for broadcasting (farthest destination) for each tileset for the U-shaped shaped transmission line for probabilities of error : 10^{-1}, 10^{-3}, 10^{-5}, 10^{-7} for BPSK and 256-QAM

Table 7.3 gathers the total values (total of all 32 tilesets) for maximum transmission powers (i.e. required power for transmission to farthest node in broadcast case) for

TABLE 7.3: Total of maximum transmission powers (broadcast case) for various capacity densities and BER values with BPSK or 256-QAM with cross-shaped transmission line.

U-Shaped	Total of Maximum Transmission Powers (Broadcast case) - dBm
Capacity 1 bits/s/Hz	-30.75
Capacity 2 bits/s/Hz	-25.97
Capacity 3 bits/s/Hz	-22.29
Capacity 4 bits/s/Hz	-18.98
Capacity 5 bits/s/Hz	-15.83
Capacity 6 bits/s/Hz	-12.75
Capacity 7 bits/s/Hz	-9.71
Capacity 8 bits/s/Hz	-6.68
BPSK $p_b = 10^{-1}$	-31.6
BPSK $p_b = 10^{-1}$	-23.96
BPSK $p_b = 10^{-1}$	-21.16
BPSK $p_b = 10^{-1}$	-19.44
256-QAM $p_b = 10^{-1}$	-12.95
256-QAM $p_b = 10^{-1}$	-2.27
256-QAM $p_b = 10^{-1}$	0.81
256-QAM $p_b = 10^{-1}$	2.64

capacity densities between 1-8 bits/s/Hz and various BERs with BPSK or 256-QAM with a U-shaped transmission line. In other words, this table represents the total of values in Fig. 7.36 and Fig. 7.37. From this table, we can have an idea on the worst case, where highest transmission power budget is required. For instance, total required power for providing broadcasting with our CMP, for a capacity density of 8 bits/s/Hz is -6.68 dBm.

7.6.2 Cross-Shaped Transmission Line

We have mentioned previously in Section 3.3.2.3 that a cross-shaped transmission line topology was chosen later on the simple U-shaped topology. This cross-shaped topology decreases distances among tilesets and also diminishes the frequency selective channel attenuation. The chosen cross-shaped topolgy with positioned tileset-IDs is illustrated in Fig. 7.38.

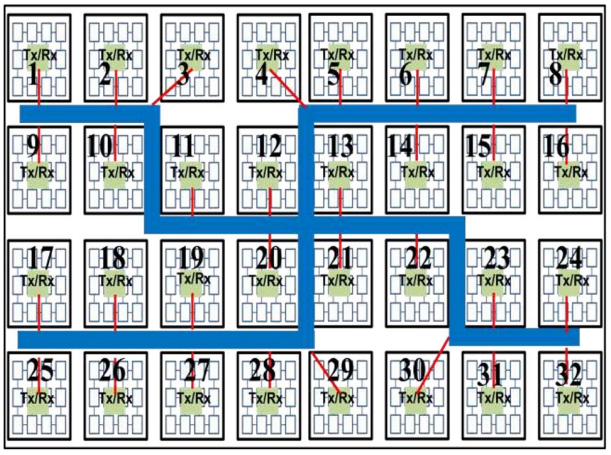


FIGURE 7.38: Cross-shaped transmission line

Next, we determine the required transmission powers for attaining desired capacity densities for our cross-shaped transmission line and compare them to the previous U-shaped transmission line.

7.6.2.1 Unicast communication

Finding distances among tilesets is not trivial as for the U-shaped transmission line due to irregularity of the topology. Firstly, we define a *symmetry* among tilesets, that who's distances to the 31 other tilesets would be same. For instance, from Fig. 7.35, we see that tileset-4 and tileset-29 have a positional symmetry. Tileset-4's distance to tileset-5

is same with tileset-29's distance to tileset-28 etc. Fig. 7.39 shows the distances in mm among 32x31 combinations of unicast communication on cross-shaped transmission line. Note that, the maximum distance in cross-shaped transmission line is 80 mm, while maximum distance in U-shaped transmission line is 120 mm. With an average attenaution of -0.25 dB/mm, this means that the largest attenaution is 10 dB more (10 times in scalar) for the U-shaped transmission line.

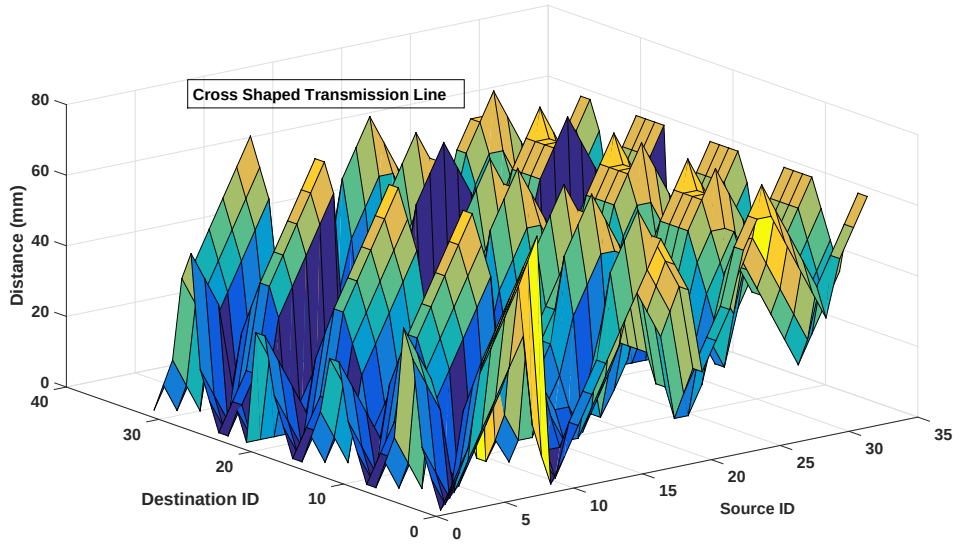


FIGURE 7.39: Distance between each 32x31 unicast communication in cross-shaped transmission line

Similarly we did for the U-shaped transmission line, we present the achievable capacities for transmitting-recepting tileset pairs with -80 dBm of transmission for power on the cross-shaped transmission line. By comparing Fig. 7.40 to 7.32, we see that on both of the topologies neighboring tilesets achieve a capacity of 3 Gbps as expected and farthest nodes' capacity still approaches to 0, however the achieved capacities for distances between these two extremum points are remarkably improved.

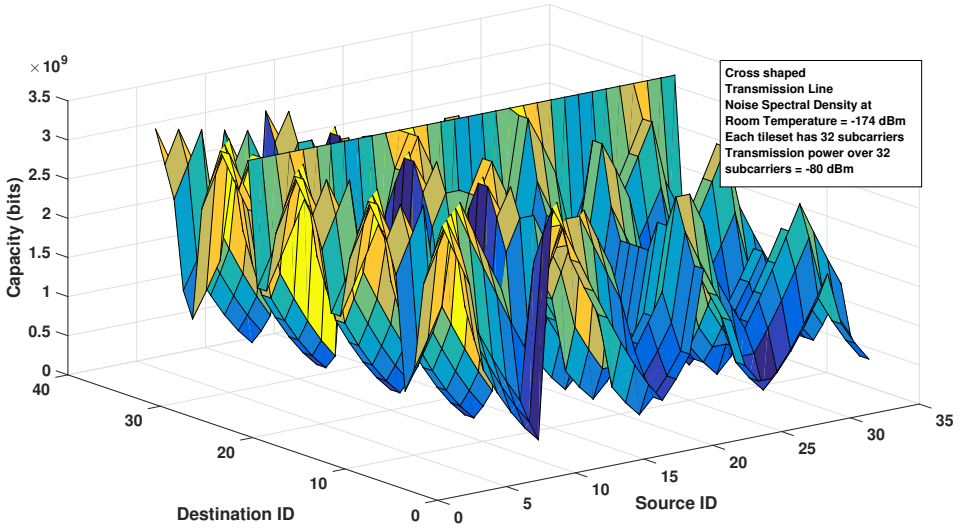


FIGURE 7.40: Capacity between each 32x31 unicast communication in cross-shaped transmission line

Fig. 7.41 shows the required transmission power in dBm for 32x31 unicast communication combination on the cross-shaped transmission line for capacity densities between 1-8 bits/s/Hz. We observe that required transmission powers are lowered by comparing it to U-shaped transmission line.

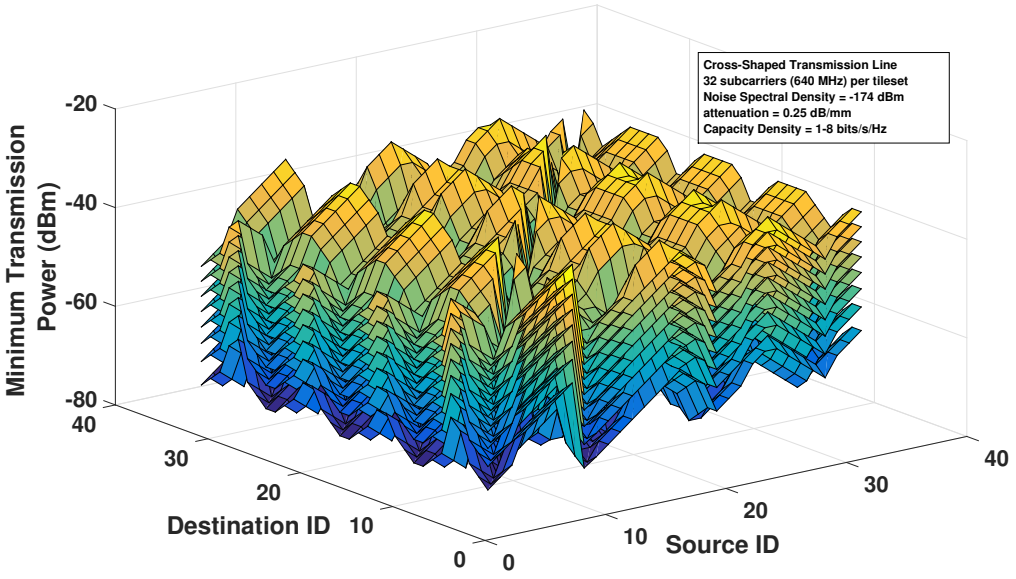


FIGURE 7.41: Required Transmission Power for each 32x31 unicast communication for the cross-shaped shaped transmission line for capacity densities 1-8 bits/s/Hz

Fig. 7.42 shows the average required transmission power for each tileset over its 31 destination tilesets, for the U-shaped topology, for capacity densities 1-8 bits/s/Hz.

One can see the reduction of the energy expenditure thanks to lower distances. From this figure, we can understand that if a centralized mechanism is used, it is much more efficient to place the CIU inside the RF transceiver of Tileset-3, Tileset-10, Tileset-20 or Tileset-29, as they require much less transmission power due to their distances to other tilesets.

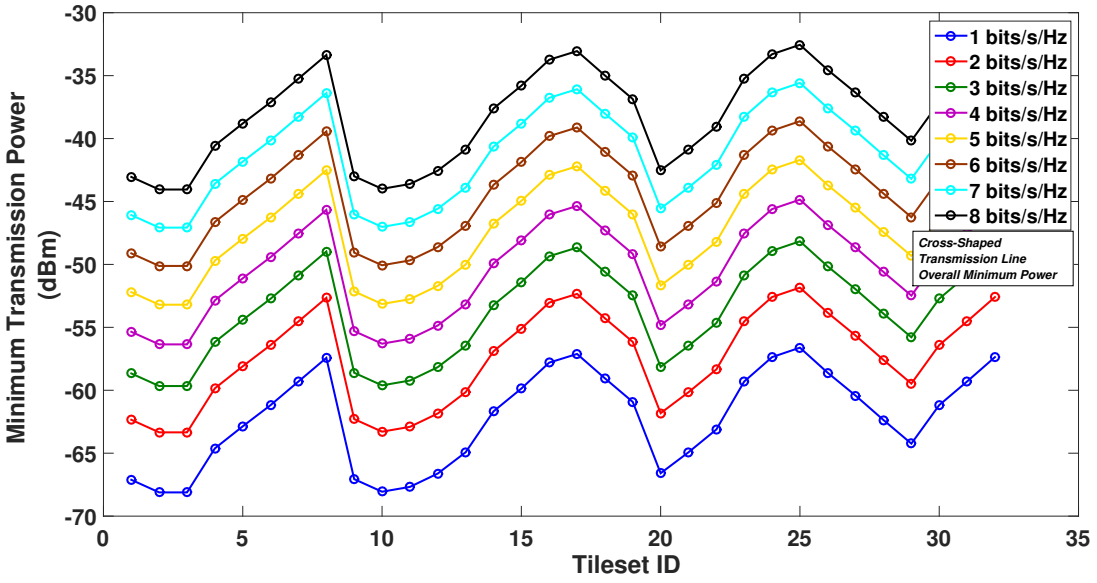


FIGURE 7.42: Average required transmission power (to 31 destinations) for each tileset for the cross-shaped shaped transmission line for capacity densities 1-8 bits/s/Hz

As we did for the U-shaped transmission line, we present the average minimum transmission power for each tileset (average of 31 destinations) for BPSK and 256-QAM, for probability of bit errors 10^{-1}, 10^{-3}, 10^{-5}, 10^{-7}. We observe similar patterns in Fig 7.43.

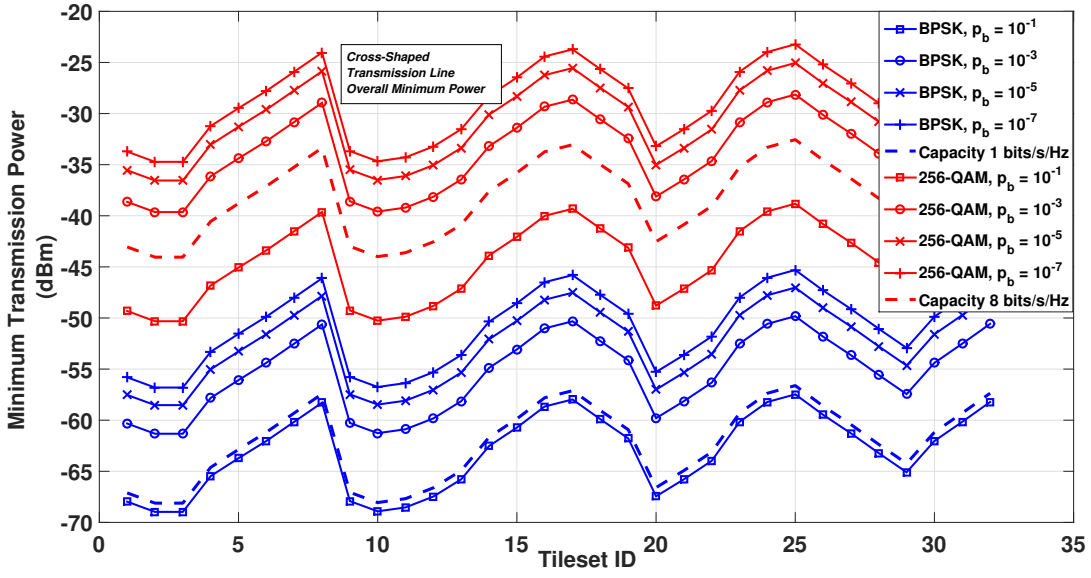


FIGURE 7.43: Average required transmission power (to 31 destinations) for each tileset for the cross-shaped shaped transmission line for probabilities of error : 10^{-1}, 10^{-3}, 10^{-5}, 10^{-7} for BPSK and 256-QAM

7.6.2.2 Broadcast communication

Fig. 7.44 shows the required transmission powers for each tileset to its farthest destination, for the cross-shaped topology, for capacity densities 1-8 bits/s/Hz. We see the improvements due to reduction of maximum distances. For instance, compared to U-shaped transmission line, required transmission powers for broadcast communications are reduced between 15 and 16 dBm.

And finally, Fig. 7.45, shows the minimum transmission power for various probabilities of error for BPSK and 256-QAM for broadcasting, concerning the farthest destination for each tileset on the cross-shaped transmission line.

Table 7.4 gathers the total values (total of all 32 tilesets) for maximum transmission powers (i.e. required power for transmission to farthest node in broadcast case) for capacity densities between 1-8 bits/s/Hz and various BERs with BPSK or 256-QAM with a cross-shaped transmission line. In other words, this table represents the total of values in Fig. 7.44 and Fig. 7.45. For instance, total required power for providing broadcasting with our CMP, for a capacity density of 8 bits/s/Hz is -18.14 dBm, which is approximately 12 dB lower compared to U-shaped topology.

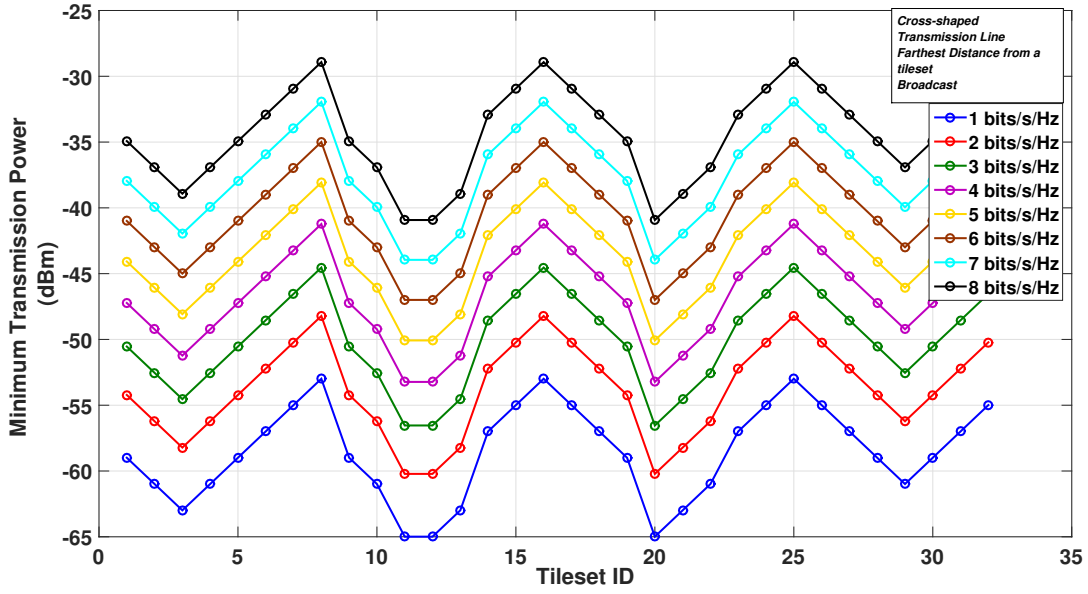


FIGURE 7.44: Minimum required transmission power for broadcasting (to maximum distance) for each tileset for the cross-shaped shaped transmission line for capacity densities 1-8 bits/s/Hz

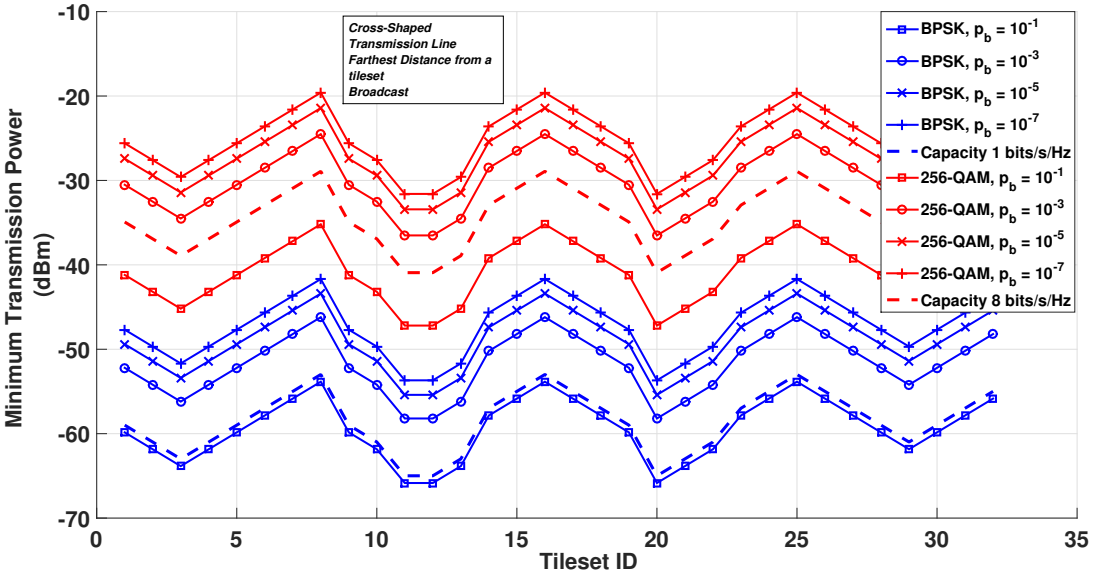


FIGURE 7.45: Required transmission power for broadcasting (farthest destination) for each tileset for the U-shaped shaped transmission line for probabilities of error : 10^{-1}, 10^{-3}, 10^{-5}, 10^{-7} for BPSK and 256-QAM

TABLE 7.4: Total of maximum transmission powers (broadcast case) for various capacity densities and BER values with BPSK or 256-QAM with cross-shaped transmission line.

Cross-Shaped	Total of Maximum Transmission Powers (Broadcast case) - dBm
Capacity 1 bits/s/Hz	-42.21
Capacity 2 bits/s/Hz	-37.44
Capacity 3 bits/s/Hz	-33.76
Capacity 4 bits/s/Hz	-30.45
Capacity 5 bits/s/Hz	-27.29
Capacity 6 bits/s/Hz	-24.21
Capacity 7 bits/s/Hz	-21.17
Capacity 8 bits/s/Hz	-18.14
BPSK $p_b = 10^{-1}$	-43.06
BPSK $p_b = 10^{-1}$	-35.42
BPSK $p_b = 10^{-1}$	-32.62
BPSK $p_b = 10^{-1}$	-30.9
256-QAM $p_b = 10^{-1}$	-24.41
256-QAM $p_b = 10^{-1}$	-13.73
256-QAM $p_b = 10^{-1}$	-10.64
256-QAM $p_b = 10^{-1}$	-8.81

7.7 Conclusion

OFDMA not only provides the oppurtunity of digitally and rapidly re-arbitrate the frequency resources, i.e. *subcarriers*, but also the chance of selecting different modulation orders, therefore different *rates* as a new flexible dimension of scalable rates for future on-chip interconnects with a trivial digital procedure.

Due to computational/circuitral difficulty and additional bandwidth overhead of signaling between tilesets, we have constructed a framework, where a single modulation order is selected for each tileset every frame. This modulation order selection procedure is completely based on the framed bandwidth allocation framework, explained in previous chapters. In this chapter, we have evaluated the relation between delay and power for the on-chip RF interconnect of WiNoCoD.

Based on our experiments, we can state firmly that average delay bounded algorithm is robust and provides much more reliable results compared to maximum delay bounded algorithm. We believe, setting bounds on averaged delays may be a more convenient way for the highly dynamic and fluctuating on-chip traffic. We have shown that, average power can be reduced up to 10 times, by increasing average delay bound by a few symbols. As a last remark, we should emphasize that this algorithm bases itself on the

equations from the reference paper, except we have added the number of channels to them and modified for the WiNoCoD's framed structure.

For these proposed intelligent modulation order selection policies, the required transmission power for different modulation orders were quantified as a multiple of the power required for BPSK. This omits the calculations of the real transmission powers that tilesets shall supply. Therefore, we have determined the destinations between tilesets for the previously proposed U-shaped topology and the improved latest cross-shaped topology. Based on these, by using the determined signal attenuation with respect to distance in our state-of-the-art transmission line, we have calculated the required minimum transmission powers concerning the information theoretic bounds. Required transmission powers for different channel capacity densities between 1-8 bits/sec/Hz are determined, which correspond to modulation orders between BPSK to 256-QAM.

Chapter 8

Conclusions and Perspectives

In this thesis work which has been performed in parallel with Project WiNoCoD, we have demonstrated the feasibility of utilizing OFDMA as a modulation mechanism on an on-chip wired RF interconnect for a massive manycore architecture. By using newly enabled necessary CMOS technology for the development of fast enough components, WiNoCoD Project aims to be the first attempt to bring OFDMA on-chip.

In Chapter 2, we have explained the emerging Chip Multiprocessor (CMP) paradigm and its basic computational structure. The need for cache coherency was discussed briefly, which is the contributor of the traffic between on-chip elements. However, electrically wired NoCs cannot sustain the traffic demand of the CMPs with hundreds of cores. The current estimation predicts that number of cores on a single CMP may reach several thousands before the end of next decade. Therefore, the state-of-the-art optical and RF interconnects are proposed for CMPs with hundreds and thousands of cores. However, these interconnects are static due to their reliance on analog circuits for generating orthogonal frequency channels. Hence, these interconnects shall be dimensioned according to peak traffic of a node. Further in the chapter, we discuss the statistical properties and burst nature of on-chip traffic and present certain synthetic traffic models to emulate cache coherency traffic.

Chapter 3 presents the Project WiNoCoD and its 2048-core CMP. First, we explain the scalable 3-level NoC hierarchy, along with each layer's specific communication infrastructure. We also mention the utilized scalable distributed hybrid cache coherency protocol (DHCCP) and its intensive demand of broadcast messages. Later in the chapter, before presenting the details of our wired RF interconnect, we briefly explain the basics of OFDM and OFDMA, and its potential advantages for the proposed on-chip architecture. OFDMA is broadcast capable intrinsically and orthogonal channel generation and bandwidth allocation in OFDMA is purely digital, rapid and efficient in contrast

with the previously mentioned optical and RF interconnects. Furthermore, circuitry and parameters of our wired RF interconnect are presented in detail.

Chapter 4 firstly formulates the main problem of this thesis from queuing and network theory perspective, which is to find the right algorithms to distribute frequency resources to the multiple transmission queues, such that certain metrics of interest like average latency is minimized. Certain algorithms from the literature approaching this issue from different grounds and their theoretical basis are discussed. Next, the infrastructure of WiNoCoD's OFDMA based RF interconnect and certain preliminary concepts are introduced. For instance, due to signaling and computational constraints, the subcarriers in the system are grouped to serve 1 flit on a single system. The unusually large OFDM bandwidth compared to most of the existing in our system, forces resource allocation algorithms to be done in few hundrerds of nanoseconds. Taking into account other additional factors increasing the cumulative latency for the effectuating bandwidth allocation, such as digital reconfiguration of the components, propagation time, computation time of OFDM components such as IFFT etc., a pipelined mechanism for subcarrier arbitration is adopted. The frame based bandwidth allocation based on this pipelined fashion is explained in detail. Two cardinally different methods for bandwidth allocation are considered in WiNoCoD : Decentralized mechanism, where each tileset broadcasts their QSIs and execute the same algorithm or centralized mechanism where a central intelligent unit (CIU) is responsible for this. The coordination between tilesets and other tilesets or between tilesets and the CIU for signaling is discussed for both of the cases. Pros and cons of two different methods are listed. In addition, utilized methods and realistic traffic models to stress algorithms are explained. Different granularity levels for signaling and resource partition was discussed and best option was chosen to implement bandwidth allocation algorithms.

In Chapter 5 we present and evaluate our proposed algorithms experimentally. First proposed method is called serial QSI allocation algorithm, where resource blocks (RBs) are sequentially allocated in blocks by iterating through QSI demands of tilesets. A major motivation behind this algorithm is to minimize the computational complexity and number of iterations as much as possible. Even though this algorithm provides very low average latencies and delay or queue exceeding probabilities, we have observed that it has a limited capacity due to its unfair nature. This phenomenon was examined and certain extension to enhance the algorithm with minimum additional complexity are presented. Next proposed algorithm was queue proportional scheduling (QPS), which allocates the available frequency resources proportional to queue lengths of tilesets. The capacity of the system was proven to be efficient, however we have observed that under lower traffic loads, the average latencies were unacceptably high. Using modified QSIs such as deterministic or expected QSI to combat outdated QSI and allow for a fairer

resource allocation are evaluated for both of these methods for both centralized and decentralized mechanisms. We conclude that there is no *ultimate best algorithm*, where they offer different good performances for different metrics of interest under different configurations and traffic statistics. However, using serial allocation algorithm with definitive QSI encoding was proven to provide the lowest average latency in general. This basic algorithm which does not require any computation complexity is feasible to be implemented with very low frame lengths such as few OFDM symbols. However, especially under higher traffic loads, we have observed QPS algorithms was able to outperform basic serial allocation in terms of remarkably lower packet delay and queue length exceeding probabilities.

In Chapter 6, one of the most innovative bandwidth allocation infrastructure we have developed for an on-chip interconnect is presented, where there exists no similar attempt in the literature, to the best of our knowledge. This novel Payload Channel Algorithm allocates all the subcarriers on a symbol to the payloads of cache line carrying long cache coherency packets. Thanks to intrinsic broadcast nature of OFDMA, this algorithm does not require any extra signaling overhead, as the type of the packet is encoded in header flits. Additionaly, two separate queues are used to avoid any inconsistency between headers and packets and also to be able to utilize resources while waiting for payload transmission. This algorithm was proven to decrease average latency up to 10 times under certain cases compared to a static counterpart. In addition, we derived an analytical expression for the average latency of this algorithm as a function of injection rate by using certain approximations and queuing theory. Furthermore, a more sophisticated dynamic version of this algorithm is developed, which allocates base resources to tilesets according to traffic fluctuations. We believe this algorithm may provide substantial performance increase for the future CMPs using OFDMA, especially with longer cache lines.

Chapter 7 is dedicated to option of using different modulation orders for WiNoCoD's OFDMA interconnect. Firstly, information theoretic relation between delay and power was revisited. Then, the necessary mechanism for using higher modulation order for extra transmission power in WiNoCoD was explained, briefly. Both inspired from the existing schedulers from the literature but designed for general wireless telecommunications, two novel modulation order selection algorithms are introduced. First one aims to minimize transmission power while setting a maximum delay of the packets. The original algorithm was designed for a single user-single channel case, which was not applicable to our case directly. Therefore, we have modified this scheduler for this special case, which we believe can also be applied to more generic multi-user multi-channel cases in wireless telecommunications. This algorithm is able to provide 3 times lower average power by increasing delay bound by just 16 symbols under certain scenarios. Moreover,

our scheduler is not able to provide the necessary delay bound especially under bursty traffic due to limited rate of the system, however it was shown that a probabilistic guarantee can be achieved. Second algorithm was also based on an existing scheduler from the literature, which we have modified to our case. Different than the previous one, this algorithm aims to minimize the average transmission power while setting a bound on average delay. The proposed scheduler was able to decrease the average energy expenditure 4 times by just increasing the average latency by few symbols. Algorithm was shown to be efficient in terms of tracing and obeying the average delay bound. We have also conducted an information theoretic analysis on two different transmission line topologies in this chapter. Proposed algorithms for intelligent modulation order choice seek to select lowest modulation order (BPSK) as possible as long as the service criteria on latency are met and give the resulting power consumption relative to BPSK. We have not given explicit transmission power values previously, however by this information theoretic analysis part we aim to determine the required minimum transmission power to achieve certain capacity and uncoded bit error rate.

Perspectives

WiNoCoD Project is an ambitious endeavor to break the reconfigurable bandwidth allocation bottleneck for massive manycore systems. We believe the proposed algorithms and discussed concepts shall be a pioneering guideline for the future OFDMA based on-chip interconnects.

In addition to this work conducted throughout the project, there exists a great potential for further contribution. With its reconfigurability, once OFDMA RF on-chip interconnects have become feasible for production, researchers may develop certain extensive algorithms and infrastructures. For instance, *carrier aggregation* paradigm from cellular communications may be investigated, that deals with the allocation of non-contiguous subcarriers to nodes, which can decrease peak-to-average power ratio problem and provide other certain benefits. Another perspective can be the *offline* optimization of subcarrier and modulation order allocation, by inspecting the traffic demand of nodes, after executing applications on the CMP.

Further work could try to evaluate the transmission power consumption both for static modulation order or dynamic modulation order case, for various traffic patterns. New energy aware intelligent modulation order selection algorithms can be developed.

By developing silicon technology in future, we can expect to have several hundreds of GHz of bandwidth which results in much shorter OFDM symbol durations. The proposed pipelined framework and bandwidth allocation mechanism can be extended to these extreme cases. In addition, we can expect to have more number of processing

elements, thus more elements accessing the RF transmission line. Also higher number of subcarriers can be implemented with future technology to sustain communication for this much number of RF nodes. The work conducted in this thesis can be extended to these cases as well.

Further work can be done on the physical layer aspects of this OFDMA based interconnect. The Peak-to-Average Power Ratio (PAPR) problem can be investigated and optimization on arbitration of subcarriers and modulation can be performed concerning this problem. In addition, channel coding and other error detection and correction techniques can be developed taking in to account the very specific constraints of this interconnect. Even though we have not taken into account the variation of attenuation between the 20-40 GHz spectrum, and assumed an average flat frequency response; further optimal subcarrier allocation algorithms can be developed, which also try to minimize the attenuation.

Appendix A

Personal Publications

Capacity Analysis of Radio Frequency Interconnect for Manycore Multiprocessor Chips (to be submitted)
Eren Unlu, Christophe Moy, Yves Louet, Jacques Palicot.
—

Bimodal Packet Aware Scheduling for an OFDMA Based On-Chip RF Interconnect (to be submitted)
Eren Unlu, Christophe Moy.
—

A Novel Bandwidth and Power Allocation Mechanism for an OFDMA Based On-Chip RF Interconnect (submitted)
Eren Unlu, Christophe Moy.
Elseiver Journal of Microprocessors and Microsystems

Reconfigurable Traffic-Aware Radio Interconnect for a 2048-core Chip Multiprocessor
Eren Unlu, Christophe Moy.
Digital System Design (DSD), 2015 18th Euromicro Conference on, Aug 2015, Funchal, Portugal. pp. 139-145.

A Dynamically Reconfigurable RF NoC for Many-Core
Alexandre Brière, textbfEren Unlu, Julien Denoulet, Andrea Pinna, Bertrand Granado, Francois Pêcheux, Yves Louët, Christophe Moy.
Great Lakes Symposium on VLSI, May 2015, Pittsburgh, United States. pp.139-144, Proceedings of the 25th edition on Great Lakes Symposium on VLSI

Flexible Radio Interface for NoC RF-Interconnect
Frédéric Drillet, Mohamad Hamieh, Lounis Zerioul, Alexandre Briere, **Eren Unlu**,

Myriam Ariaudo, Yves Louet, Emmanuelle Bourdel, Julien Denoulet, Andréa Pinna, Bertrand Granado, Patrick Garda, François Pêcheux, Cédric Duperrier, Sébastien Quintanel, Philippe Meunier, Christophe Moy, Olivier Romain.
Digital System Design (DSD), 2014 17th Euromicro Conference on, Aug 2014, Verona, Italy. pp.6.

An OFDMA Based RF Interconnect for Massive Multi-core Processors
Eren Unlu, Mohamad Hamieh, Christophe Moy, Myriam Ariaudo, Yves Louët, Frédéric Drillet, Alexandre Brière, Lounis Zerioul, Julien Denoulet, Andrea Pinna, Bertrand Granado, François Pécheux, Patrick Garda, Cédric Duperrier, Sébastien Quintanel, Olivier Romain.
NOCS 2014, Sep 2014, Ferrara, Italy. 2 p., 2014.

Bandwidth Allocation for Massive Multicore CMP RF Interconnect
Allocation des Fréquences pour les Interconnexions RF dans un Réseau sur Puce Multi-Coeurs Massivement Parallèle
Eren Unlu, Christophe Moy.
XXVe Colloque GRETSI 2015, Sep 2015, Lyon, France. Actes du XXV Colloque GRETSI, 4 p., 2015.

WiNoCoD : A Hierarchical RF Interconnection Network for MPSoCs
WiNoCoD : Un réseau d'interconnexion hiérarchique RF pour les MPSoC
Alexandre Brière, Julien Denoulet, Andrea Pinna, Bertrand Granado, François Pêcheux, Patrick Garda, Myriam Ariaudo, Frédéric Drillet, Cédric Duperrier, Mohamad Hamieh, Sébastien Quintanel, Olivier Romain, Lounis Zerioul, Yves Louët, Christophe Moy, **Eren Unlu**, E.Bourdel.
ComPAS'2014 : Conférence d'informatique en Parallélisme, Architecture et Système, Apr 2014, Neuchâtel, Switzerland. pp.track architecture.

Appendix B

Explanation of OMNeT++ Codes

OMNeT++ is a C++ based, *event driven* network simulator. It includes libraries, therefore C++ objects for simulating dynamics of queues, packet transmission and network related aspects. Event driven simulation means that any type of event (e.g. arrival of a packet, timeout of a downcounter etc.). Therefore, we have modeled an OFDM symbol based simulation (slotted communication paradigm/after finishing of an OFDM symbol, new symbol starts) as mentioned previously.

B.1 General Organization of the Main C++ Files and OMNeT++ Classes

Rather than compiling different C++ files for each tileset, we have chosen to implement a single long main C++ file composed of approximately 1500 lines. By exploiting certain pre-defined libraries of OMNeT++, with this single file we can accelarate the execution of our codes significantly.

As mentioned previously throughout the thesis, a transmitter queue at the RF front-end of each tileset is utilized as the main scope of our research, whereas other aspects are exluded for bandwidth allocation. And due to the temporal quantized nature of OFDM based communication, queue dynamics and bandwidth allocation related procedures are executed in a symbol accurate manner. We have implemented codes in 4 different main C++ files due to their structures for bandwidth allocation algorithms : (i) Regular Channel Allocation algorithms presented in Chapter 5, (ii) Payload Channel Allocation algorithms explained in Chapter 6, (iii) Maximum Delay Bounded Dynamic Modulation Order Allocation algorithm, (iv) Average Delay Bounded Dynamic Modulation Order Allocation algorithm.

At the first section in this general file, the user determines the duration of simulation in symbols, number of RBs per symbol, frame duration (for centralized allocation, reconfiguration time for tilesets and computation time for CIU can be determined separetely.), number of bits to encode QSI and the required number of RBs to reserve for signaling.

Each of tilesets' (where number can be determined according to user parameter) transmission queues are OMNeT++ queue objects (`cQueue`), and they are stored in a C++ vector.

As stated before, OMNeT++ is an event driven network simulator, where there exists *message* class (`cMessage`) objects, which imitate the behaviour of packets and these message objects can be transpassed between modules throughout the simulation. *Events* are also a subclass of this OMNeT++ message class. The structure and parameters of message objects are pre-defined in a special OMNeT++ file called `msg` file. For our simulations, we have defined one type of `msg` file, which emulates the *flits*.

B.1.1 Symbol Accurate Mechanism

A created OMNeT++ event object called `OFDM_symbol` triggers a start of a new symbol, which executes the necessary codes. Each time a new OFDM symbol starts, first certain amount of packets are generated stochastically for each of the tilesets and inserted to the transmission queues, with a dedicated function called `generate_pkts()`. After this, a second function called `serve_pkts()` determines the number of RBs each tileset has on this symbol, and serves the respective amount of flits from each of tileset's queues. Every new symbol, a counter is incremented and if this is a new frame, necessary bandwidth allocation algorithm is performed with function `allocateRBs()`. Finally, at the end of these operations, a new symbol event is called (switch to next symbol).

B.1.2 Regular Channel Allocation

Firstly proposed dynamic bandwidth allocation algorithms in Chapter 5 are all written in a single main C++ file. Firstly user chooses with a parameter whether a serial RB allocation (Section 5.1) or Queue Proportional Scheduling (Section 5.2) will be used. Next, a parameter determines whether a centralized or decentralized scheme is adopted. If a centralized scheme is adopted, certain subcarriers are reserved for CIU response and the necessary reconfiguration time for activating new subcarrier arbitration is taken account to frame structure as explained in the thesis. As mentioned in previous section, at the start of each symbol, if this is the first symbol of a frame, the function who is

responsible of bandwidth allocation, `allocateRBs()` is called. A matrix describes the identification of each RB for each symbol of a frame.

Then based on the chosen parameter, 2 different processes allocate RBs inside a frame to tilesets according to serial allocation or QPS allocation. For QPS allocation, the algorithm computes the total QSI and associated proportional allocation as explained in thesis. If serial allocation is chosen, a parameter defines whether a second loop will be applied in the arbitration procedure as mentioned in Section 5.1.1. By using necessary temporary variables the pipelined mechanism presented in the thesis is applied, that the new arbitration of RBs are activated in next frame.

B.1.3 Payload Channel Allocation

Different than the previous regular channel allocation mechanism, this time for payload channel allocation algorithm, we implement two different queues for each tileset, stored in seperate vectors. First vector stores the short 1-flit packets and header flits of the long packets, whereas the second one stores the payloads of the long packets. In addition to this, a single, global `cQueue` object is utilized for the representation of the *register*. The demands of tilesets for payload channel transmission are represented by generated `cMessage` objects and inserted into register queue. Similarly for the previous case, every symbol, `generate_pkts()` function is called to generate packets for each tilesets stochastically and `serve_pkts()` funtion is called to clear flits inside the short queue. If a tileset has the payload channel on a "specific symbol", , its payload message is served from its payload queue.

B.1.4 Dynamic Modulation Order Allocation

Different than the regular payload channel allocation algorithm, a counter is incremented on every symbol where payload channel is not utilized, and if this symbol is the start of a frame, `allocateRBs()` function is called to artbitrate home channels in next frame.

B.1.4.1 Maximum Delay Bounded Dynamic Modulation Order Allocation

A different main C++ file was created for each of the dynamic modulation order allocation algorithm as mentioned previously. Actually dynamic modulation order is just an extension to the bandwidth allocation algorithms presented in Chapter 5. Additionally, it has special functions and variables for modulation order purposes.

For Maximum Delay Bounded Dynamic Modulation Order Allocation, for each tileset there is a vector containing the number of flits with current time left to violation of delay bound, between 1 and $D_0 - 1$. Each vector for each tileset is also stored in a vector in a nested manner. A function named `updateDmaxVectors()` is called every symbol to update the new number of packets inside the transmission queues of tilesets with respective time left for delay bound violation, as explained in Section 7.4.2.

Every frame, bandwidth allocation is performed as in Chapter 5, however additionally each tileset determines its new modulation order to be used in next frame, with a function called `chooseModulationOrder_MaxDelayBound()`. The necessary signaling overhead for the modulation order broadcasting is also taken into account. Therefore with newly chosen modulation order and new number of RBs on each symbol, a frame can calculate how many flits it can serve. The `generate_pkts()` and `serve_pkts()` functions are utilized every symbol as for the bandwidth allocation algorithms.

B.1.4.2 Average Delay Bounded Dynamic Modulation Order Allocation

For Average Delay Bounded Dynamic Modulation Order Allocation Algorithm, in addition to structure for the dynamic bandwidth allocation algorithms in Chapter 5, the main file includes a `chooseModulationOrder_AvgDelayBound()` function, which is called at the start of every frame. Using recently obtained QSIs and computed number of RBs in a frame, it determines the modulation order for each tileset. The necessary signaling overhead for the modulation order broadcasting is also taken into account. Therefore with newly chosen modulation order and new number of RBs on each symbol, a frame can calculate how many flits it can serve. The `generate_pkts()` and `serve_pkts()` functions are utilized every symbol as for the bandwidth allocation algorithms.

B.2 Stochastic Packet Generation and Traffic Models

We are using stochastic and statistical models to emulate the number of packets coming to RF transmitters of tilesets. Without loss of generality, this procedure is modeled as an exogeneous packet generation in a symbol accurate manner based on the assumptions made in Section 4.3.4. At the start of each symbol the `generate_pkts()` function is called, which calculates the number of packets generated for packets and insert them into the transmission `cQueue` object of the associated tileset. For each generated packet a binary random value is drawn to determine the length of the packet (short or long).

Based on the length of the packet, that specific number of flits are generated (as a subclass of `cMessage` objects), and each of these flits are associated with a packet number. The last flit of a packet is marked with a flag, to determine the ending of a packet.

OMNeT++ provides a large class of statistical tools and C++ libraries for the generation of random values based on pre-defined or user-defined distributions. As mentioned previously in Section 4.3.4, we are using 2 main stochastic models : (i) Poisson and (ii) Discrete Pareto Burst Poisson Process (DPBPP). Before simulation user chooses whether a uniform or non-uniform spatial injection rate distribution is chosen (as mentioned in Section 4.3.4.2). Another parameter defines whether a Poisson or DPBPP model is preferred. If Poisson model is chosen, in `generate_pkts()` function, OMNeT++'s pre-defined `poisson()` function with the associated injection rate is used to determine the number of packets generated on each symbol for each tileset. However, implementing the DPBPP model taken from the literature is not possible with pre-defined functions, therefore we have written another function named `generate_packetsDPBPP()`. This function is called in `generate_pkts()` function if the DPBPP is chosen.

B.2.1 DPBPP packet generation

DPBPP traffic model was explained in Section 4.3.4.2. `generate_packetsDPBPP()` function is used for this purpose. First a Poisson random value drawn by pre-defined `poisson()` function determines the number of *flows*. Each flow is an entity with a duration, where it generates 1 packet per symbol throughout its execution. Multiple number of flows shall be active during an instance. For example, let us say that at time t_0 2 flows are generated with durations 2 and 3 symbols, respectively. And in next symbol , t_1, 1 flow is generated with a duration of 1 symbol. Therefore, with aggregation of these flows; 2 packets are generated on t_0, 3 packets are generated on t_1, 1 packet is generated on t_2, and no packets are generated on t_3 and after, as there is no active flows left. We have emulated this aggregation of active queues in OMNeT++ by using `cQueue` and `cMessage` objects.

The duration of a flow is also determined stochasitcally, which is *Pareto* distributed. There is no such a pre-defined Pareto distribution, as we have desired exactly, therefore we have created a function named `flowLengthGenDPPBP()`. For this purpose we have used a feature of OMNeT++, which allows the generation of discrete random values based on a user defined distribution. For creation of this pre-defined Pareto distribution, we have created a function named `initDPPBP()` which is called one time at the start of the simulation, if DPBPP traffic model is chosen. This Pareto distribution is defined inside this function according to mathematical definition with desired *Hurst (H)*

parameter. Based on these, injection rate is normalized, which is used for the Poisson random value to generate flows.

B.3 Collection of Metrics of Interest

OMNeT++ provides a large variety of metrics to be evaluated at the end of the simulation. For instance, each time we generate a packet (therefore flits) based on `cMessage` class, we can stamp the time of generation (in symbols). When these flits are being served from the transmission queue, we check whether this flit is the last flit of a packet by checking the associated flag of `cMessage` class. If this is an ending flit of a packet, we calculate the transmission delay by subtracting current simulation time from the stamped time of generation. For evaluation of scalar metrics, OMNeT++ has a `cSignal` class, which collects desired parameters (such as delay of packets) and provide the users information such average, maximum etc. of these metrics at the end of simulation. In addition to these, in order to obtain the distribution of packet delays or queue lengths, we have used `cHistogram` objects of OMNeT++, where we have converted to csv files at the end.

Bibliography

[1] Alain Greiner. Tsar: a scalable, shared memory, many-cores architecture with global cache coherence. In *9th International Forum on Embedded MPSoC and Multicore (MPSoC'09)*, 2009.

[2] Chen Sun, Mark T Wade, Yunsup Lee, Jason S Orcutt, Luca Alloatti, Michael S Georgas, Andrew S Waterman, Jeffrey M Shainline, Rimas R Avizienis, Sen Lin, et al. Single-chip microprocessor that communicates directly using light. *Nature*, 528(7583):534–538, 2015.

[3] George Kurian, Jason E Miller, James Psota, Jonathan Eastep, Jifeng Liu, Jurgen Michel, Lionel C Kimerling, and Anant Agarwal. Atac: a 1000-core cache-coherent processor with on-chip optical network. In *Proceedings of the 19th international conference on Parallel architectures and compilation techniques*, pages 477–488. ACM, 2010.

[4] Shekhar Borkar. Thousand core chips: a technology perspective. In *Proceedings of the 44th annual Design Automation Conference*, pages 746–749. ACM, 2007.

[5] Érika Cota, Alexandre de Morais Amory, and Marcelo Soares Lubaszewski. *Reliability, Availability and Serviceability of Networks-on-chip*. Springer, 2011.

[6] Cheng Li, Paul V Gratz, and Samuel Palermo. Nano-photonic networks-on-chip for future chip multiprocessors. In *More than Moore Technologies for Next Generation Computer Design*, pages 155–186. Springer, 2015.

[7] Sujay Deb, Amlan Ganguly, Partha Pratim Pande, Benjamin Belzer, and Deukhyoun Heo. Wireless noc as interconnection backbone for multicore chips: Promises and challenges. *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, 2(2):228–239, 2012.

[8] Vassos Soteriou, Hangsheng Wang, and Li-Shiuan Peh. A statistical traffic model for on-chip interconnection networks. In *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2006. MASCOTS 2006. 14th IEEE International Symposium on*, pages 104–116. IEEE, 2006.

[9] Gordon E Moore et al. Cramming more components onto integrated circuits, 1965.

[10] Kunle Olukotun, Basem A Nayfeh, Lance Hammond, Ken Wilson, and Kunyung Chang. The case for a single-chip multiprocessor. *ACM Sigplan Notices*, 31(9): 2–11, 1996.

[11] Edward J. Correia. Sorry, moore's law: Multicore is the new game in town, 2012. URL http://i.crn.com/misc/2012/moores_law_chart.jpg. [Online; accessed October 16, 2015].

[12] Mark L James, Andrew A Shapiro, Paul L Springer, and Hans P Zima. Adaptive fault tolerance for scalable cluster computing in space. *International Journal of High Performance Computing Applications*, 23(3):227–241, 2009.

[13] Tile-gx processor family. URL http://www.tilera.com/products/processors/TILE-Gx_Family.

[14] Lina J Karam, Ismail AlKamal, Alan Gatherer, Gene Frantz, David V Anderson, Brian L Evans, et al. Trends in multicore dsp platforms. *Signal Processing Magazine, IEEE*, 26(6):38–49, 2009.

[15] Kunle Olukotun, Lance Hammond, and James Laudon. Chip multiprocessor architecture: techniques to improve throughput and latency. *Synthesis Lectures on Computer Architecture*, 2(1):1–145, 2007.

[16] Bryan Schauer. Multicore processors–a necessity. *ProQuest discovery guides*, pages 1–14, 2008.

[17] András Vajda, Mats Brorsson, and Diarmuid Corcoran. *Programming many-core chips*. Springer, 2011.

[18] David J Lilja. Cache coherence in large-scale shared-memory multiprocessors: issues and comparisons. *ACM Computing Surveys (CSUR)*.

[19] Anant Agarwal and Markus Levy. The kill rule for multicore. In *Design Automation Conference, 2007. DAC'07. 44th ACM/IEEE*, pages 750–753. IEEE, 2007.

[20] Dana Vantrease, Robert Schreiber, Matteo Monchiero, Moray McLaren, Norman P Jouppi, Marco Fiorentino, Al Davis, Nathan Binkert, Raymond G Beausoleil, and Jung Ho Ahn. Corona: System implications of emerging nanophotonic technology. In *ACM SIGARCH Computer Architecture News*, volume 36, pages 153–164. IEEE Computer Society, 2008.

[21] Li-Rong Zheng and Hannu Tenhunen. Wires as interconnects. In *Interconnect-Centric Design for Advanced SoC and NoC*, pages 25–54. Springer, 2005.

[22] Jari Nurmi, Hannu Tenhunen, Jouni Isoaho, and Axel Jantsch. *Interconnect-centric design for advanced SoC and NoC*. Springer, 2004.

[23] Chrysostomos Nicopoulos, Vijaykrishnan Narayanan, and Chita R Das. *Network-on-Chip Architectures: A Holistic Design Exploration*, volume 45. Springer, 2009.

[24] SA Arteris. From" bus" and" crossbar" to" network-on-chip". *Arteris, Inc*, 1741, 2009.

[25] Abbas Sheibanyrad, Frédéric Pétrot, and Axel Jantsch. *3D integration for NoC-based SoC Architectures*. Springer, 2011.

[26] Sudeep Pasricha and Nikil Dutt. *On-chip communication architectures: system on chip interconnect*. Morgan Kaufmann, 2010.

[27] Wolfgang Arden. Future semiconductor material requirements and innovations as projected in the itrs 2005 roadmap. *Materials Science and Engineering: B*, 134 (2):104–108, 2006.

[28] Mikhail Haurylau, Guoqing Chen, Hui Chen, Jidong Zhang, Nicholas A Nelson, David H Albonesi, Eby G Friedman, and Philippe M Fauchet. On-chip optical interconnect roadmap: challenges and critical directions. *Selected Topics in Quantum Electronics, IEEE Journal of*, 12(6):1699–1705, 2006.

[29] Jing Xue, Alok Garg, Berkehan Ciftcioglu, Jianyun Hu, Shang Wang, Ioannis Savidis, Manish Jain, Rebecca Berman, Peng Liu, Michael Huang, et al. An intra-chip free-space optical interconnect. In *ACM SIGARCH Computer Architecture News*, volume 38, pages 94–105. ACM, 2010.

[30] Joseph W Goodman, Frederick J Leonberger, Sun-Yuan Kung, and Ravindra A Athale. Optical interconnections for vlsi systems. *Proceedings of the IEEE*, 72(7): 850–866, 1984.

[31] David AB Miller. Rationale and challenges for optical interconnects to electronic chips. *Proceedings of the IEEE*, 88(6):728–749, 2000.

[32] Vilson R Almeida, Carlos A Barrios, Roberto R Panepucci, and Michal Lipson. All-optical control of light on a silicon chip. *Nature*, 431(7012):1081–1084, 2004.

[33] Yan Pan, John Kim, and Gokhan Memik. Featherweight: low-cost optical arbitration with qos support. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 105–116. ACM, 2011.

[34] Robert H Havemann and James A Hutchby. High-performance interconnects: An integration overview. *Proceedings of the IEEE*, 89(5):586–601, 2001.

[35] Sujay Deb. *Millimeter-wave Wireless Network-on-Chip: A CMOS compatible interconnection infrastructure for future many-core processors.* PhD thesis, Washington State University, 2012.

[36] Xiaomeng Shi, Jian-Guo Ma, Kiat Seng Yeo, AV Do, and Erping Li. Equivalent circuit model of on-wafer cmos interconnects for rfics. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 13(9):1060–1071, 2005.

[37] Sai-Wang Tam, M Frank Chang, and Jongsun Kim. Wireline and wireless rf-interconnect for next generation soc systems. In *Circuits and Systems (MWSCAS), 2011 IEEE 54th International Midwest Symposium on*, pages 1–3. IEEE, 2011.

[38] Dan Zhao and Yi Wang. Sd-mac: Design and synthesis of a hardware-efficient collision-free qos-aware mac protocol for wireless network-on-chip. *Computers, IEEE Transactions on*, 57(9):1230–1245, 2008.

[39] Jau Lin Jr, Hsin-Ta Wu, Yu Su, Li Gao, Aravind Sugavanam, Joe E Brewer, et al. Communication using antennas fabricated in silicon integrated circuits. *Solid-State Circuits, IEEE Journal of*, 42(8):1678–1687, 2007.

[40] Xinmin Yu, Suman P Sah, Sujay Deb, Partha Pratim Pande, Benjamin Belzer, and Deukhyoun Heo. A wideband body-enabled millimeter-wave transceiver for wireless network-on-chip. In *Circuits and Systems (MWSCAS), 2011 IEEE 54th International Midwest Symposium on*, pages 1–4. IEEE, 2011.

[41] Amlan Ganguly, Kevin Chang, Sujay Deb, Partha Pratim Pande, Benjamin Belzer, and Christof Teuscher. Scalable hybrid wireless network-on-chip architectures for multicore systems. *Computers, IEEE Transactions on*, 60(10):1485–1502, 2011.

[42] Noriyuki Miura, Daisuke Mizoguchi, Mari Inoue, Takayasu Sakurai, and Tadahiro Kuroda. A 195-gb/s 1.2-w inductive inter-chip wireless superconnect with transmit power control scheme for 3-d-stacked system in a package. *Solid-State Circuits, IEEE Journal of*, 41(1):23–34, 2006.

[43] Xiaomeng Shi and Kiat Seng Yeo. On-chip interconnects of rfics.

[44] M Frank Chang, Jason Cong, Adam Kaplan, Mishali Naik, Glenn Reinman, Eran Socher, and S-W Tam. Cmp network-on-chip overlaid with multi-band rf-interconnect. In *High Performance Computer Architecture, 2008. HPCA 2008. IEEE 14th International Symposium on*, pages 191–202. Ieee, 2008.

[45] Sujay Deb, Kevin Chang, Amlan Ganguly, and Partha Pande. Comparative performance evaluation of wireless and optical noc architectures. In *SOC Conference (SOCC), 2010 IEEE International*, pages 487–492. IEEE, 2010.

[46] Kanishka Lahiri, Anand Raghunathan, and Sujit Dey. Evaluation of the traffic-performance characteristics of system-on-chip communication architectures. In *VLSI Design, 2001. Fourteenth International Conference on*, pages 29–35. IEEE, 2001.

[47] Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li. The parsec benchmark suite: Characterization and architectural implications. In *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, pages 72–81. ACM, 2008.

[48] Christian Bienia, Sanjeev Kumar, and Kai Li. Parsec vs. splash-2: A quantitative comparison of two multithreaded benchmark suites on chip-multiprocessors. In *Workload Characterization, 2008. IISWC 2008. IEEE International Symposium on*, pages 47–56. IEEE, 2008.

[49] Umit Y Ogras and Radu Marculescu. *Modeling, Analysis and Optimization of Network-on-Chip Communication Architectures*, volume 184. Springer Science & Business Media, 2013.

[50] Frederica Darema-Rogers, Gregory F Pfister, and Kimming So. *Memory access patterns of parallel scientific programs*, volume 15. ACM, 1987.

[51] Girish V Varatkar and Radu Marculescu. On-chip traffic modeling and synthesis for mpeg-2 video applications. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 12(1):108–119, 2004.

[52] Kihong Park and Walter Willinger. *Self-similar network traffic and performance evaluation*. Wiley Online Library, 2000.

[53] Paul Bogdan and Radu Marculescu. Statistical physics approaches for network-on-chip traffic characterization. In *Proceedings of the 7th IEEE/ACM international conference on Hardware/software codesign and system synthesis*, pages 461–470. ACM, 2009.

[54] Antoine Scherrer, Antoine Fraboulet, Tanguy Risset, et al. Analysis and synthesis of cycle-accurate on-chip traffic with long-range dependence. 2005.

[55] Walter Willinger, Murad S Taqqu, Robert Sherman, and Daniel V Wilson. Self-similarity through high-variability: statistical analysis of ethernet lan traffic at the source level. *Networking, IEEE/ACM Transactions on*, 5(1):71–86, 1997.

[56] Ahmad Khonsari, Mohammad R Aghajani, Arash Tavakkol, and Mohammad Sadegh Talebi. Mathematical analysis of buffer sizing for network-on-chips

under multimedia traffic. In *Computer Design, 2008. ICCD 2008. IEEE International Conference on*, pages 150–155. IEEE, 2008.

[57] Boris Tsybakov and Nicolas D Georganas. Overflow and losses in a network queue with a self-similar input. *Queueing systems*, 35(1-4):201–235, 2000.

[58] Daniel Greenfield, Arnab Banerjee, J-G Lee, and Simon Moore. Implications of rent's rule for noc design and its fault-tolerance. In *Networks-on-Chip, 2007. NOCS 2007. First International Symposium on*, pages 283–294. IEEE, 2007.

[59] Daniel L Greenfield. *Communication Locality in Computation: Software, Chip Multiprocessors and Brains*. Citeseer, 2010.

[60] Mario Badr and Natalie Enright Jerger. Synfull: synthetic traffic models capturing cache coherent behaviour. In *Computer Architecture (ISCA), 2014 ACM/IEEE 41st International Symposium on*, pages 109–120. IEEE, 2014.

[61] Yan Pan, John Kim, and Gokhan Memik. Tuning nanophotonic on-chip network designs for improving memory trafics. *PICA@ MICRO2009*.

[62] Sergey Blagodurov, Sergey Zhuravlev, Alexandra Fedorova, and Ali Kamali. A case for numa-aware contention management on multicore systems. In *Proceedings of the 19th international conference on Parallel architectures and compilation techniques*, pages 557–558. ACM, 2010.

[63] Alexandre Brière, Julien Denoulet, Andrea Pinna, Bertrand Granado, Francois Pêcheux, Eren Unlu, Yves Louët, and Christophe Moy. A dynamically reconfigurable rf noc for many-core. In *Proceedings of the 25th edition on Great Lakes Symposium on VLSI*, pages 139–144. ACM, 2015.

[64] Semiconductor Industry Association et al. Itrs: International technology roadmap for semiconductors, 2013.

[65] Korey Sewell, Ronald G Dreslinski, Thomas Manville, Sudhir Satpathy, Nathaniel Pinckney, Geoffrey Blake, Michael Cieslak, Reetuparna Das, Thomas F Wenisch, Dennis Sylvester, et al. Swizzle-switch networks for many-core systems. *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, 2(2):278–294, 2012.

[66] Tze-Yun Sung, Hsi-Chin Hsin, and Shengyong Chen. Reconfigurable vlsi architecture for fft processor. In *WSEAS International Conference. Proceedings. Mathematics and Computers in Science and Engineering*, number 9. World Scientific and Engineering Academy and Society, 2009.

[67] Chi-Hang Chan, Yan Zhu, Sai-Weng Sin, RP Martins, et al. A 3.8 mw 8b 1gs/s 2b/cycle interleaving sar adc with compact dac structure. In *VLSI Circuits (VLSIC), 2012 Symposium on*, pages 86–87. IEEE, 2012.

[68] Alexandre Briere, Julien Denoulet, Andrea Pinna, Bertrand Granado, François Pêcheux, Patrick Garda, Myriam Ariaudo, Frédéric Drillet, Cédric Duperrier, Mohamad Hamieh, et al. Winocod: Un réseau d'interconnexion hiérarchique rf pour les mpsoc. In *ComPAS'2014: Conférence d'informatique en Parallélisme, Architecture et Système*, pages track–architecture, 2014.

[69] Nakul Manchanda and Karan Anand. Non-uniform memory access (numa). *New York University*, 2010.

[70] Ronald Bruno, John E Miller, and Leonard Schuchman. Digital audio broadcasting system, February 1 1994. US Patent 5,283,780.

[71] Shengli Zhou and Zhaohui Wang. Ofdm basics. *OFDM for Underwater Acoustic Communications*, pages 23–38.

[72] W COOLEY JAMES, AW Lewis, and D WELCH. Historical notes on the fast fourier transform. *Proceedings of the IEEE*, 55(10):1675, 1967.

[73] Carlos E. Davilla. Fourier transform of common signals, November 2009. URL http://cnx.org/contents/7549bfa0-d833-4795-a522-60269735215a@8/Fourier_Transform_of_Common_Si.

[74] Mérouane Debbah. Short introduction to ofdm. *White Paper, Mobile Communications Group, Institut Eurecom*, 2004.

[75] Slawomir Pietrzyk. *OFDMA for Broadband Wireless Access (Artech House Mobile Communications)*. Artech House, Inc., 2006.

[76] RR Mosier and RG Clabaugh. Kineplex, a bandwidth-efficient binary transmission system. *American Institute of Electrical Engineers, Part I: Communication and Electronics, Transactions of the*, 76(6):723–728, 1958.

[77] Ebrahim Saberinia and AH Tewfik. Pulsed and non-pulsed ofdm ultra wideband wireless personal area networks. In *Ultra Wideband Systems and Technologies, 2003 IEEE Conference on*, pages 275–279. IEEE, 2003.

[78] Philip N Ji, Dayou Qian, Karthik Sethuraman, Junqiang Hu, Yoshiaki Aono, Tsutomu Tajima, William Blakney, and Ting Wang. First demonstration of real-time all-optical software-defined intra-data center star network using ofdm and burst switching. In *OptoElectronics and Communications Conference and Photonics in Switching*, page PD3_3. Optical Society of America, 2013.

[79] Mohamad Hamieh, Myriam Ariaudo, Sébastien Quintanel, and Yves Louët. Sizing of the physical layer of a rf intra-chip communications. In *Electronics, Circuits and Systems (ICECS), 2014 21st IEEE International Conference on*, pages 163–166. IEEE, 2014.

[80] Wang Hongwei. Fft basics and case study using multi-instrument. *Virtins Technology, Rev*, 1, 2009.

[81] Joseph Mitola. Cognitive radio—an integrated agent architecture for software defined radio. 2000.

[82] Jacques Palicot. *Radio engineering: From software radio to cognitive radio*. John Wiley & Sons, 2013.

[83] Jim Zyren and Wes McCoy. Overview of the 3gpp long term evolution physical layer. *Freescale Semiconductor, Inc., white paper*, 2007.

[84] Niranjan G Shivaratri, Phillip Krueger, and Mukesh Singhal. Load distributing for locally distributed systems. *Computer*, 25(12):33–44, 1992.

[85] Anand Ganti, Eytan Modiano, and John N Tsitsiklis. Tranmission scheduling for multi-channel satellite and wireless networks. In *Proceedings of the Annual Allerton Conference on Communication Control and Computing*, volume 40, pages 1319–1328. Citeseer, 2002.

[86] Philip Ji, Dayou Qian, Konstantinos Kanonakis, Christoforos Kachris, and Ioannis Tomkos. Design and evaluation of a flexible-bandwidth ofdm-based intra-data center interconnect. *Selected Topics in Quantum Electronics, IEEE Journal of*, 19 (2):3700310–3700310, 2013.

[87] John DC Little and Stephen C Graves. Little's law. In *Building Intuition*, pages 81–100. Springer, 2008.

[88] Thomas Kunz. The influence of different workload descriptions on a heuristic load balancing scheme. *Software Engineering, IEEE Transactions on*, 17(7):725–730, 1991.

[89] Krishna Jagannathan and Eytan Modiano. The impact of queue length information on buffer overflow in parallel queues. *Information Theory, IEEE Transactions on*, 59(10):6393–6404, 2013.

[90] Somsak Kittipiyakul and Tara Javidi. Delay-optimal server allocation in multiqueue multiserver systems with time-varying connectivities. *Information Theory, IEEE Transactions on*, 55(5):2319–2333, 2009.

[91] Beier Li, Paschalis Tsiaflakis, Marc Moonen, Jochen Maes, and Mamoun Guenach. Dynamic resource allocation based partial crosstalk cancellation in dsl networks. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1–5. IEEE, 2010.

[92] Atilla Eryilmaz and R Srikant. Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 1794–1803. IEEE, 2005.

[93] Constantinos Dovrolis, Dimitrios Stiliadis, and Parameswaran Ramanathan. Proportional differentiated services: Delay differentiation and packet scheduling. *ACM SIGCOMM Computer Communication Review*, 29(4):109–120, 1999.

[94] Hoon-Tong Ngin, Chen-Khong Tham, and Wee-Seng Soh. Generalised minimum queuing delay: an adaptive multi-rate service discipline for atm networks. In *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 398–404. IEEE, 1999.

[95] Peter Homan and Janez BESTER. Enhanced synchronous packet switching for ip packets. *IEICE transactions on communications*, 85(1):247–256, 2002.

[96] George P Nychis, Chris Fallin, Thomas Moscibroda, Onur Mutlu, and Srinivasan Seshan. On-chip networks from a networking perspective: Congestion and scalability in many-core interconnects. *ACM SIGCOMM computer communication review*, 42(4):407–418, 2012.

[97] DL Greenfield. *Rentian locality in chip multiprocessors*. PhD thesis, University of Cambridge, 2010.

[98] Amir-Mohammad Rahmani, Iman Kamali, Pejman Lotfi-Kamran, Ali Afzali-Kusha, and Saeed Safari. Negative exponential distribution traffic pattern for power/performance analysis of network on chips. In *VLSI Design, 2009 22nd International Conference on*, pages 157–162. IEEE, 2009.

[99] Poona Bahrebar and Dirk Stroobandt. Characterizing traffic locality in 3d noc-based cmps using a path-based partitioning method. In *High-Performance Interconnects (HOTI), 2014 IEEE 22nd Annual Symposium on*, pages 63–70. IEEE, 2014.

[100] Ran Manevich, Israel Cidon, and Avinoam Kolodny. Handling global traffic in future cmp nocs. In *Proceedings of the International Workshop on System Level Interconnect Prediction*, pages 40–47. ACM, 2012.

[101] Ankit More. *Network-on-Chip (NoC) Architectures for Exa-scale Chip-Multi-Processors (CMPs)*. PhD thesis, Drexel University, 2013.

[102] Eitan Frachtenberg and Uwe Schwiegelshohn. *Job scheduling strategies for parallel processing*. Springer, 2009.

[103] Mihai Pricopi and Tulika Mitra. Task scheduling on adaptive multi-core. *Computers, IEEE Transactions on*, 63(10):2590–2603, 2014.

[104] Stefania Sesia, Issam Toufik, and Matthew Baker. *LTE: the UMTS long term evolution*. Wiley Online Library, 2009.

[105] Xinen Zhu, Bo Yang, Ci Chen, Liang Xue, and Xiangyu Guan. Cross-layer scheduling for ofdma-based cognitive radio systems with delay and security constraints. 2014.

[106] Pierre Ansel, Qiang Ni, and Thierry Turletti. Fhcf: a simple and efficient scheduling scheme for ieee 802.11 e wireless lan. *Mobile Networks and Applications*, 11 (3):391–403, 2006.

[107] Amr Rizk and Markus Fidlerz. Queue-aware uplink scheduling: Analysis, implementation, and evaluation. In *IFIP Networking Conference (IFIP Networking), 2015*, pages 1–9. IEEE, 2015.

[108] András Varga et al. The omnet++ discrete event simulation system. In *Proceedings of the European simulation multiconference (ESM'2001)*, volume 9, page 65. sn, 2001.

[109] Hyung Gyu Lee, Naehyuck Chang, Umit Y Ogras, and Radu Marculescu. On-chip communication architecture exploration: A quantitative evaluation of point-to-point, bus, and network-on-chip approaches. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 12(3):23, 2007.

[110] Radu Marculescu, Umit Y Ogras, Li-Shiuan Peh, Natalie Enright Jerger, and Yatin Hoskote. Outstanding research problems in noc design: system, microarchitecture, and circuit perspectives. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 28(1):3–21, 2009.

[111] Santiago Gonzalez Pestana, Edwin Rijpkema, Andrei Rădulescu, Kees Goossens, and Om Prakash Gangwal. Cost-performance trade-offs in networks on chip: A simulation-based approach. In *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, volume 2, pages 764–769. IEEE, 2004.

[112] Paul Gratz, Changkyu Kim, Robert McDonald, Stephen W Keckler, and Doug Burger. Implementation and evaluation of on-chip network architectures. In *Computer Design, 2006. ICCD 2006. International Conference on*, pages 477–484. IEEE, 2006.

[113] Praveen Vellanki, Nilanjan Banerjee, and Karam S Chatha. Quality-of-service and error control techniques for network-on-chip architectures. In *Proceedings of the 14th ACM Great Lakes symposium on VLSI*, pages 45–50. ACM, 2004.

[114] Steven Cameron Woo, Moriyoshi Ohara, Evan Torrie, Jaswinder Pal Singh, and Anoop Gupta. The splash-2 programs: Characterization and methodological considerations. In *ACM SIGARCH Computer Architecture News*, volume 23, pages 24–36. ACM, 1995.

[115] Jun Ho Bahn and Nader Bagherzadeh. A generic traffic model for on-chip interconnection networks. *Network on Chip Architectures*, page 22, 2008.

[116] Parag Pruthi and Ashok Erramilli. Heavy-tailed on/off source behavior and self-similar traffic. In *Communications, 1995. ICC'95 Seattle,'Gateway to Globalization', 1995 IEEE International Conference on*, volume 1, pages 445–450. IEEE, 1995.

[117] Glen Kramer. On generating self-similar traffic using pseudo-pareto distribution. *Technical brief, Department of Computer Science, University of California, Davis.¡ http://wwwcsif. cs. ucdavis. edu/˜ kramer/papers/self_sim. pdf*, 2000.

[118] Ronald G Addie, Timothy D Neame, and Moshe Zukerman. Modeling superposition of many sources generating self similar traffic. In *In Proceedings of ICC'99*. Citeseer, 1999.

[119] Sergio Ledesma and Derong Liu. Synthesis of fractional gaussian noise using linear approximation for generating self-similar network traffic. *ACM SIGCOMM Computer Communication Review*, 30(2):4–17, 2000.

[120] N. Carter. *Schaum's Outline of Computer Architecture*. Schaum's Outline Series. McGraw-Hill Education, 2001. ISBN 9780071399623. URL https://books.google.fr/books?id=24V0OtD7HeAC.

[121] Takatsugu Ono, Koji Inoue, and Kazuaki Murakami. Adaptive cache-line size management on 3d integrated microprocessors. In *SoC Design Conference (ISOCC), 2009 International*, pages 472–475. IEEE, 2009.

[122] Donald Gross and Carl Harris. Fundamentals of queueing theory. 1998.

[123] Paul J Burke. The output of a queuing system. *Operations research*, 4(6):699–704, 1956.

[124] Ahmed Mokhtar. Characterization of the departure process of m/g/1 queue with heavy-tailed service time distribution. In *High Performance Switching and Routing, 2000. ATM 2000. Proceedings of the IEEE Conference on*, pages 373–376. IEEE, 2000.

[125] Frank A Haight and Frank A Haight. Handbook of the poisson distribution. 1967.

[126] Kenji Nakagawa. On the series expansion for the stationary probabilities of an m/d/1 queue. *Journal of the Operations Research Society of Japan-Keiei Kagaku*, 48(2):111–122, 2005.

[127] A Ya Khinchin. The mathematical theory of a stationary queue. Technical report, DTIC Document, 1967.

[128] Mohammad Khojastepour and Ashutosh Sabharwal. Power optimal scheduling with maximum delay constraints. 2003.

[129] Dinesh Rajan, Ashutosh Sabharwal, and Behnaam Aazhang. Delay-bounded packet scheduling of bursty traffic over wireless channels. *Information Theory, IEEE Transactions on*, 50(1):125–144, 2004.

[130] Claude E Shannon and Warren Weaver. The mathematical theory of information. 1949.

[131] Seema Bandyopadhyay and Edward J Coyle. An energy efficient hierarchical clustering algorithm for wireless sensor networks. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 1713–1723. IEEE, 2003.

[132] Alvin C Fu, Eytan Modiano, and John N Tsitsiklis. 'optimal energy allocation and admission control for communications satellites. *Networking, IEEE/ACM Transactions on*, 11(3):488–500, 2003.

[133] Yi Zhang, Pulak Chowdhury, Massimo Tornatore, and Biswanath Mukherjee. Energy efficiency in telecom optical networks. *Communications Surveys & Tutorials, IEEE*, 12(4):441–458, 2010.

[134] Fredrik Berggren and Seong-Lyun Kim. Energy-efficient control of rate and power in ds-cdma systems. *Wireless Communications, IEEE Transactions on*, 3(3):725–733, 2004.

[135] Mohammad Ali Khojastepour and Ashutosh Sabharwal. Delay-constrained scheduling: Power efficiency, filter design, and bounds. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1938–1949. IEEE, 2004.

[136] P Mohana Shankar. *Introduction to wireless systems*. Wiley New York, 2002.

[137] Andrea Goldsmith. *Wireless communications*. Cambridge university press, 2005.

List of Figures

List of Tables

Abbreviations

ADC	Analog-to-Digital Converter
BPSK	Binary Phase Shift Keying
CIU	Central Intelligent Unit
CMOS	Complementary Metal Oxide Semiconductor
CMP	Chip Multiprocessor
DAC	Digital-to-Analog Converter
DFT	Discrete Fourier Transform
DHCCP	Distributed Hybrid Cache Coherency Protocol
DPBPP	Discrete Pareto Burst Poisson Process
DQSI	Definitive Queue State Information
EQPS	Expected Queue Proportional Scheduling
EQSI	Expected Queue State Information
FFT	Fast Fourier Transform
FIFO	First-in First-out
IDFT	Inverse Discrete Fourier Transform
IFFT	Inverse Fast Fourier Transform
LO	Local Oscillator
LQF	Longest Queue First
MTL	Microstrip Transmission Line
MWSR	Multiple-Write Single-Read
NoC	Network-on-Chip
OFDM	Orthogonal Frequency Division Multiplexing
OFDMA	Orthogonal Frequency Division Multiple Access
OPF	Oldest Packet First
P/S	Parallel-to-Serial

QAM	Quadrature Amplitude Modulation
QPS	Queue Proportional Scheduling
QPSK	Quadrature Phase Shift Keying
QSI	Queue State Information
RB	Resource Block
RF	Radio Frequency
RX	Receiver
S/P	Serial-to-Parallel
SNR	Signal-to-Noise Ratio
SoC	System-on-Chip
SWMR	Single-Write Multiple-Read
TX	Transmitter