UNIVERSITE D'ANTANANARIVO

ECOLE SUPERIEURE POLYTECHNIQUE D'ANTANANARIVO

DEPARTEMENT ELECTRONIQUE

MEMOIRE DE FIN D'ETUDES EN VUE DE L'OBTENTION DU DIPLÔME D'INGENIEUR

Spécialité ELECTRONIQUE

Option :

INFORMATIQUE APPLIQUEE

Intitulé :

UTILISATION DU RESEAU DE PETRI POUR L'ETUDE D'UN NŒUD DE COMMUTATION DANS UN RESEAU *WAN*

Présenté par :

RAJAONARISON Tolotriniaina Mirado

Soutenu le 27 Avril 2010

N° : 04 /EN/IA/2009

Année Universitaire 2008-2009

MEMOIRE DE FIN D'ETUDES EN VUE DE L'OBTENTION DU DIPLÔME D'INGENIEUR

Spécialité ELECTRONIQUE

Option :

INFORMATIQUE APPLIQUEE

Intitulé :

UTILISATION DU RESEAU DE PETRI POUR L'ETUDE D'UN NŒUD DE COMMUTATION DANS UN RESEAU *WAN*

Présenté par :

RAJAONARISON Tolotriniaina Mirado

Membres du Jury :

Monsieur RAKOTOMIRAHO Soloniaina	Président
Monsieur RAKOTONDRASOA Justin	Examinateur
Monsieur RATSIMBAZAFY Guy Prédon Claude	Examinateur
Monsieur RANDRIAMAROSON Rivo Mahandrisoa	Examinateur

Rapporteur :

Madame RABEHERIMANANA Lyliane

Soutenu le 27 Avril 2010

N° : 04 /EN/IA/2009

Année Universitaire 2008-2009

REMERCIEMENTS

Je tiens à remercier, en premier lieu, Dieu Tout-puissant de m'avoir accordé du temps et de la patience pour la réalisation de ce mémoire.

J'adresse mes vifs remerciements:

A notre Chef de Département Electronique Monsieur RATSIMBA Mamy Nirina, pour toutes les aides qu'il a investies pour chaque étudiant du Département Electronique.

A Monsieur RAKOTOMIRAHO Soloniaina, qui a bien voulu présider la soutenance de ce mémoire de fin d'études.

Que soient remerciés ici,

Monsieur RAKOTONDRASOA Justin

Monsieur RATSIMBAZAFY Guy Prédon Claude

Monsieur RANDRIAMAROSON Rivo Mahandrisoa

qui ont accepté de juger ce travail.

Ma profonde gratitude s'adresse à mon encadreur, Madame RABEHERIMANANA Lyliane, qui, malgré ses maintes occupations, a partagé ses connaissances et ses précieux conseils pour l'élaboration à temps et à terme de ce travail.

A tous les enseignants du Département Electronique de l'Ecole Supérieure Polytechnique,

A mes parents,

A tous mes collègues,

Mirado

RESUME

Une issue cruciale pour la conception de composants logiciels dans un réseau étendu est l'aptitude à supporter des types de flux de réseau variés et à garantir les besoins de Qualité de Service (QdS) pour les utilisateurs. En particulier, cela nécessite des techniques rigoureuses mais faciles à utiliser pour prédire et analyser la performance des réseaux informatiques. Pour ce besoin, des techniques de modélisation formelle peuvent être utilisées pour l'étude des nœuds de réseau. Le Réseau de Petri (RdP) est une technique puissante à cet effet, grâce à sa capacité analytique et à sa possibilité d'évolution dans le temps. Dans ce mémoire, nous présentons des modèles de Réseaux de Petri (incluant des modèles de base, RdP colorés, RdP temporels et des RdP stochastiques) pour la modélisation des caractéristiques d'un nœud de commutation dans un réseau *WAN (Wide Area Network).* En utilisant des RdP dans cette étude de cas, nous avons démontré la possibilité d'évaluer la performance des autres nœuds de réseau comme ceux d'ATM et des réseaux IP tels qu'Internet.

Mots-clés : Réseaux de Petri, nœud de commutation, WAN, gestion et contrôle de trafic, QdS, évaluation de performance.

TABLE DES MATIERES

LISTE DES ABREVIATIONS

ATM : Asynchronous Transfer Mode

BSC : Binary Synchronous Communication

DARPA : Defense Advanced Research Projects Agency

DSCP : Differentiated Service Control Point

FIFO : First In First Out

HDLC : High level Data Link Control Protocole

IP : Internet Protocol

ISO : International Standardization Organization

LAN : Local Area Network

LIFO : Last In First Out

MAN : Metropolitan Area Network

MDI : Multiple Document Interface

OSI : Open Systems Interconnection

QdS : Qualité de Service

QoS : Quality of Service

RdP : Réseau de Petri

RED : Random Early Detection

RFC : Request For Comments

RSVP : Resource reSerVation Protocol

RTC : Réseau Téléphonique Commuté

SED : Système à Evènement Discret

SPQ : Strict Priority Queuing

TCP : Transport Control Protocol

ToS : Type of Service

UML : Unified Modeling Language

VoIP : Voice over IP

WAN : Wide Area Network

WPQ : Weighted Fair Queuing

LISTE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION

Un réseau téléinformatique est le résultat de la connexion de plusieurs machines entre elles, afin que les utilisateurs et les applications qui fonctionnent sur ces dernières puissent échanger des informations. La plupart des domaines de recherche sont actuellement confrontés aux réseaux de grande échelle. Les réseaux actuels se chiffrent effectivement en milliers de nœuds pour les réseaux de télécommunication, voire milliard de nœuds pour l'*Internet* [1]. Outre cet enjeu de taille, le réseau téléinformatique doit gérer des flux de natures aussi diverses que la voix, les données ou la vidéo. Ceci peut influencer les paramètres d'un réseau tel que le débit, la qualité de service qu'il offre aux clients. Ainsi, la phase de modélisation est devenue primordiale pour la conception d'un système informatique ou téléinformatique complexe. La modélisation et l'évaluation de performance des systèmes et réseaux informatiques sont des disciplines de l'informatique qui, depuis ses débuts dans les années 1970, étudie et développe des méthodes mathématiques permettant de comprendre, prédire et optimiser les performances de ces systèmes, incluant à la fois le matériel et le logiciel. Selon le but à atteindre, plusieurs techniques sont utilisées, telles que les blocs fonctionnels pour comprendre un protocole, la théorie des graphes pour optimiser la topologie, la recherche opérationnelle pour définir la performance d'une table de routage, l'étude analytique pour prédire un trafic [2] et les réseaux de Petri pour valider la dynamique du comportement d'un système informatique.

La présente étude est axée sur la dernière technique mentionnée ci-dessus. Elle s'intitule **« UTILISATION DU RESEAU DE PETRI POUR L'ETUDE D'UN NOEUD DE COMMUTATION DANS UN RESEAU *WAN* ».**

Pour ce faire, ce rapport de mémoire est décomposé en trois chapitres. D'abord, les généralités sur le réseau WAN (ou *Wide Area Network)* seront présentées au premier chapitre. Le second chapitre traitera la modélisation du nœud de commutation dans un réseau WAN par Réseau de Petri. Le dernier chapitre est consacré à la présentation de la simulation du simulateur *PetriSim* développé sous Visual Studio 2005 ainsi que les résultats obtenus.

Chapitre 1

GENERALITES SUR LE RESEAU *WAN* ET LE RESEAU DE PETRI

Grâce aux résultats de recherche sur des systèmes informatiques de plus en plus performants, on assiste à une croissance importante d'équipements réseau tant dans le domaine de télécommunication que de l'informatique. Cependant, dès que le nombre de ces équipements dépasse quelques stations, il devient nécessaire de gérer les échanges sur un réseau. Une analyse au préalable est donc primordiale. Dans ce premier chapitre une description du système à modéliser, qui est le réseau WAN, sera présentée, les outils de modélisation seront ensuite examinés.

1.1 TYPES DE RESEAUX

a) Réseau informatique

Les *réseaux informatiques* étaient initialement conçus pour relier des terminaux distants à un site central. Ils intègrent ensuite des ordinateurs interconnectés entre eux, puis des stations de travail ou serveurs. Les concepteurs de réseaux informatiques essayaient, dans un premier temps, d'assurer tout simplement l'intégrité des données véhiculées dans les réseaux [3].

b) Réseau de télécommunication

Contrairement aux informaticiens, les opérateurs de télécommunication se souciaient plus sur la synchronisation et sur le temps de traversée des données, qui sont les contraintes de base des *réseaux de télécommunication*, que sur l'intégrité de ces données.

Vu cette différence de point de vue, l'intégration de la parole téléphonique et de la vidéo dans un réseau informatique et l'utilisation des supports de télécommunication par des données informatiques ne se font pas sans difficulté.

c) Réseau téléinformatique

Ainsi est née la *téléinformatique* qui se définit comme étant l'ensemble de services de nature informatique pouvant être fournis à travers un réseau de télécommunication [4].

Un *réseau téléinformatique* est donc un système de transmissions numériques intégrant entre autres des terminaux, des ordinateurs, des imprimantes, etc. reliés entre eux par des liaisons de communication (Fig 1.1).

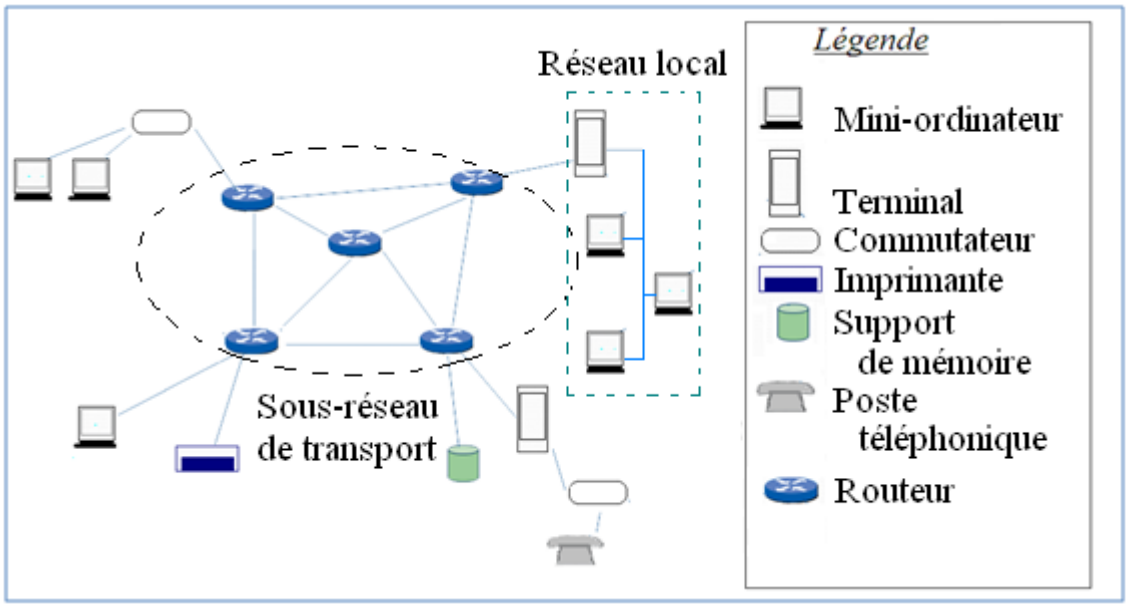

Figure 1.1 : **Réseau téléinformatique intégrant un sous-réseau de transport, des réseaux locaux et des ordinateurs de différentes tailles.**

1.2 DENOMINATIONS DE RESEAU

Plusieurs dénominations de réseau existent. Elles dépendent de la distance maximale séparant les points les plus éloignés du réseau (Fig 1.2) [5].

a) Réseau LAN (Local Area Network)

Le réseau LAN désigne un réseau s'étendant sur quelques mètres jusqu'à plusieurs kilomètres. C'est un réseau à la taille d'une entreprise (un étage, un bâtiment, voire un site).e taille (réseau personnel interconnectant sur quelques mètres des équipements dans une localité). Mais il peut s'étendre sur plusieurs centaines de mètres.

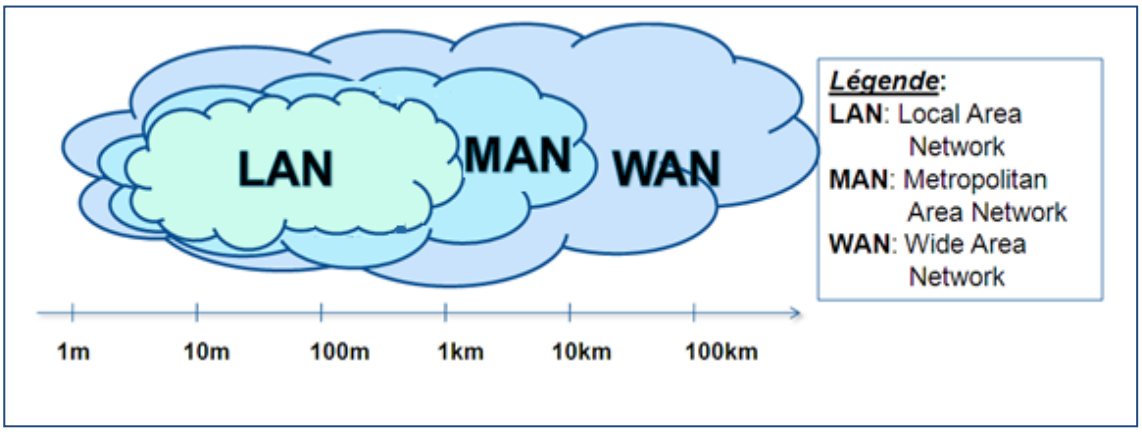

Figure 1.2 : **Catégories de réseaux**

Son débit varie aujourd'hui de quelques mégabits à plusieurs centaines de mégabits par seconde (Mbps).

b) Réseau MAN (Metropolitan Area Network)

Le réseau MAN traduit par réseau métropolitain, intègre les réseaux d'interconnexion des entreprises ou des réseaux particuliers à l'échelle d'une ville ou d'une région. En général, il permet de véhiculer les données entre les réseaux locaux d'entreprise.

c) Réseau WAN (Wide Area Network)

Appelé aussi réseau étendu, le réseau WAN englobe des sites géographiquement éloignés les uns des autres et est destiné à transporter les informations sur des distances à l'échelle d'un pays. Il sert surtout pour désigner tout réseau dépassant l'étendue d'un seul établissement physique et constitué par l'interconnexion de plusieurs réseaux élémentaires.

Dans toute la suite, nous nous intéresserons à cette dernière catégorie de réseau, en particulier, au réseau *Internet* qui peut se définir comme étant le réseau d'interconnexion des réseaux *WAN* à l'échelle mondiale. Le réseau *Internet* transporte des paquets IP (*Internet Protocol*) qui sont véhiculés d'une machine terminale à une autre et dont le principe sera présenté dans la suite de l'ouvrage. Par abus de langage, *Internet* désigne à la fois un réseau et un protocole, nous préférons garder le terme de réseau IP pour désigner une telle catégorie de réseau et IP pour le protocole.

1.3 ARCHITECTURE RESEAU

Une architecture réseau est un ensemble hiérarchisé de protocoles de communication permettant à divers équipements informatiques d'échanger des informations par le biais d'un réseau. Ce paragraphe présentera le modèle OSI *(Open Systems Interconnection)* qui est l'architecture de référence au niveau international et l'architecture TCP/IP (*Transfer Control Protocol/Internet Protocol*) qui est le modèle standard de fait, à cause de l'évolution du réseau *Internet* qui l'utilise.

a) Modèle de référence OSI

L'architecture proposée par l'ISO (*International Standardization Organization*) comporte sept couches spécifiées par la norme ISO 7498-1. La figure 1.3 illustre ces sept couches dont les détails sont présentés dans l'annexe 1. Ceci a pour objectif de permettre la constitution de réseaux téléinformatiques normalisés au sein desquels peuvent varier des équipements hétérogènes.

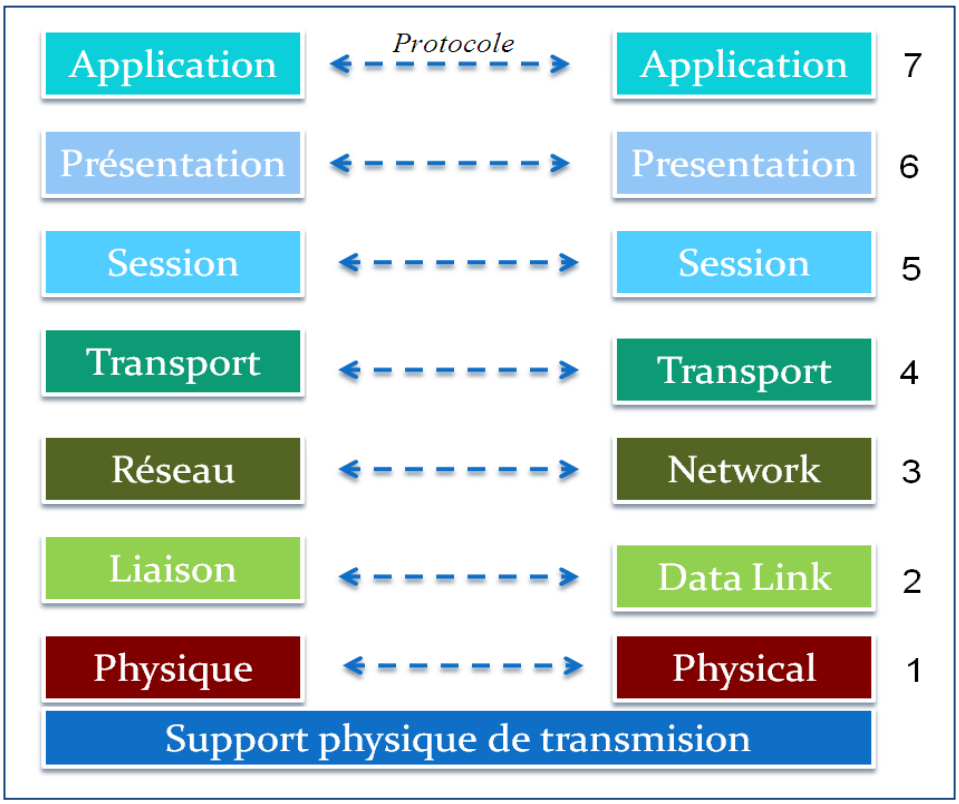

Figure 1.3 : **Modèle de référence OSI**

b) Architecture TCP/IP

Le modèle TCP/IP est né à la suite d'un projet expérimental de l'agence américaine DARPA (*Defense Advanced Research Projects Agency*) en 1969. A partir de 1983, le modèle TCP/IP s'est progressivement imposé comme modèle de référence [6].

i. Comparaison entre OSI et TCP/IP

Le modèle TCP/IP ne suit pas tout à fait l'architecture du modèle OSI. La figure 1.4 montre que TCP/IP ne possède que quatre couches. Mais cette architecture peut varier d'une implémentation à une autre, les deux couches correspondant aux protocoles de transport (TCP) et réseau (IP) restent, cependant, les piliers de l'architecture.

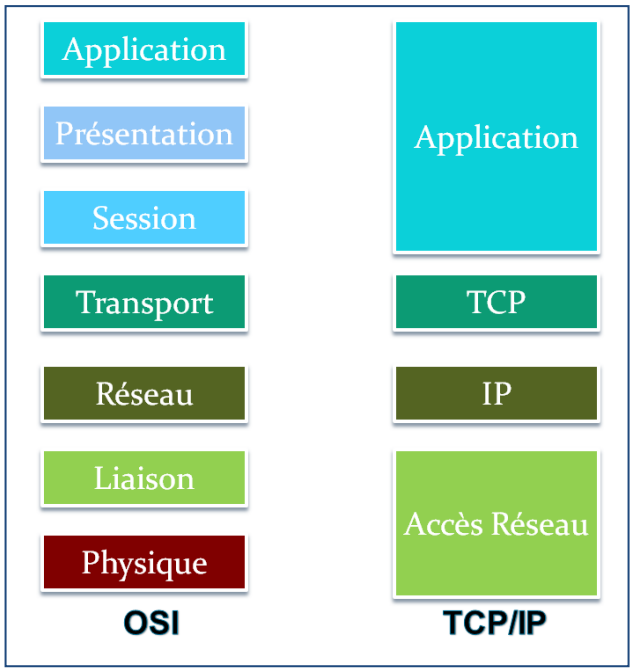

Figure 1.4 : **Comparaison OSI-TCP/IP**

ii. Encapsulation de paquets IP

Les messages des utilisateurs sont découpés en paquets (paquets IP), qui ont des longueurs maximales variables de l'ordre de 1000 ou 2000 bits. Cette étape, appelée fragmentation, permet de réduire le temps de transit de l'information dans le réseau. Etant donné que ces paquets vont traverser des réseaux de télécommunication, ils doivent être de nouveaux fragmentés et encapsulés dans une autre structure (Fig 1.5), telle que le paquet ATM (*Asynchronous Transfer Mode*) et reconstitués à la destination.

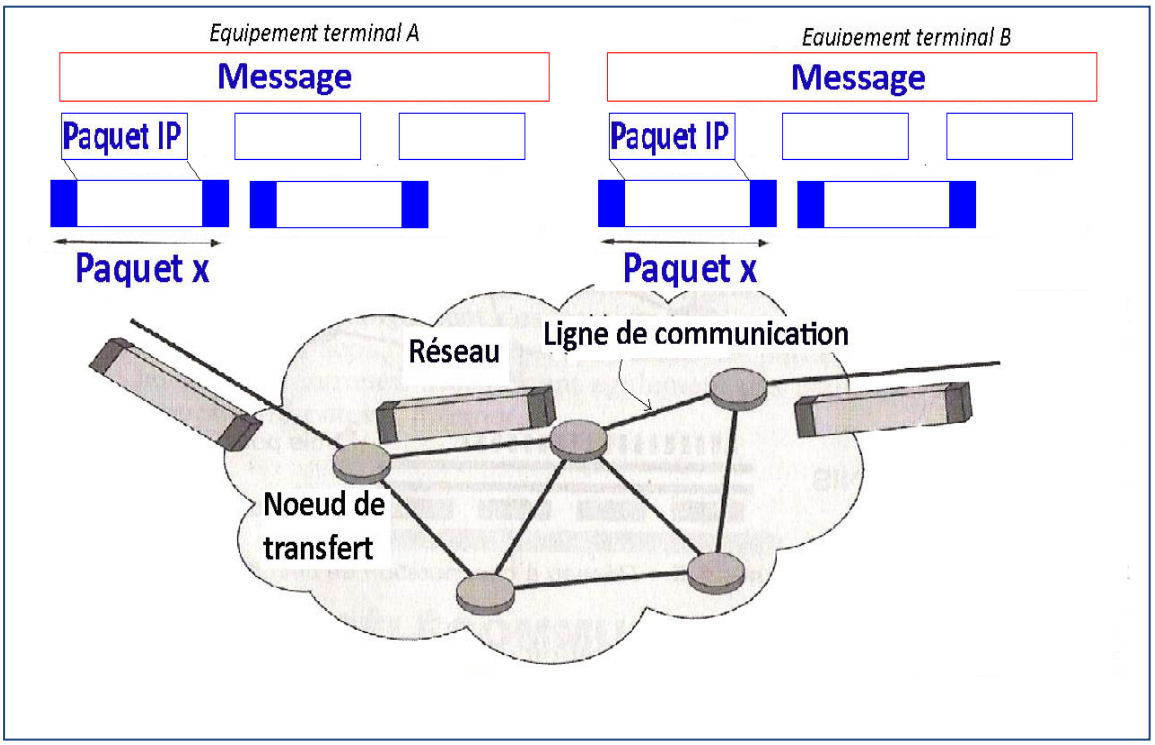

Figure 1.5 : **Encapsulation d'un paquet IP**

1.4 EQUIPEMENTS RESEAU

Les équipements réseau sont responsables de l'acheminement des informations depuis l'émission jusqu'à la réception. Le tableau 1 représente les rôles des principaux équipements réseau suivant leurs interventions dans les sept couches du modèle OSI. La suite de notre étude concerne essentiellement les commutateurs et les routeurs.

a) Rôles des principaux équipements réseau

i. Commutateur

Un *commutateur* est un équipement à N entrées et N sorties. Il route le paquet arrivant sur ses entrées vers sa destination de sortie. Dans le cas d'un commutateur ATM, il assure à la fois la traduction de l'en-tête de la cellule, son routage et le multiplexage sur la sortie correspondante. Il est parfois nécessaire de stocker temporairement une cellule quand la ressource requise est utilisée par une autre cellule. Ainsi, le commutateur est équipé de mémoires tampons ou *buffers* qui forment une file d'attente, dont les différents placements possibles sont illustrés par Fig. 1.6.

Tableau 1 : ***Principaux équipements réseau et leurs rôles***

Equipements	Niveau OSI	Rôles
Concentrateur (Hub)	Physique	prendre les données binaires parvenant d'un port et les diffuser sur l'ensemble des ports.
Commutateur (Switch)	Liaison	analyser les trames reçues pour les diriger vers la machine de destination (adresse physique)
Routeur	Réseau	choisir le chemin qu'un message va emprunter (adresse IP dans le cas du protocole IP)
Répéteur	Physique	régénérer le signal entre deux nœuds du réseau
Pont	Liaison	relier des réseaux travaillant avec le même protocole
Passerelle	transport ou supérieur	relier deux réseaux, servant d'interfaces entre deux protocoles différents

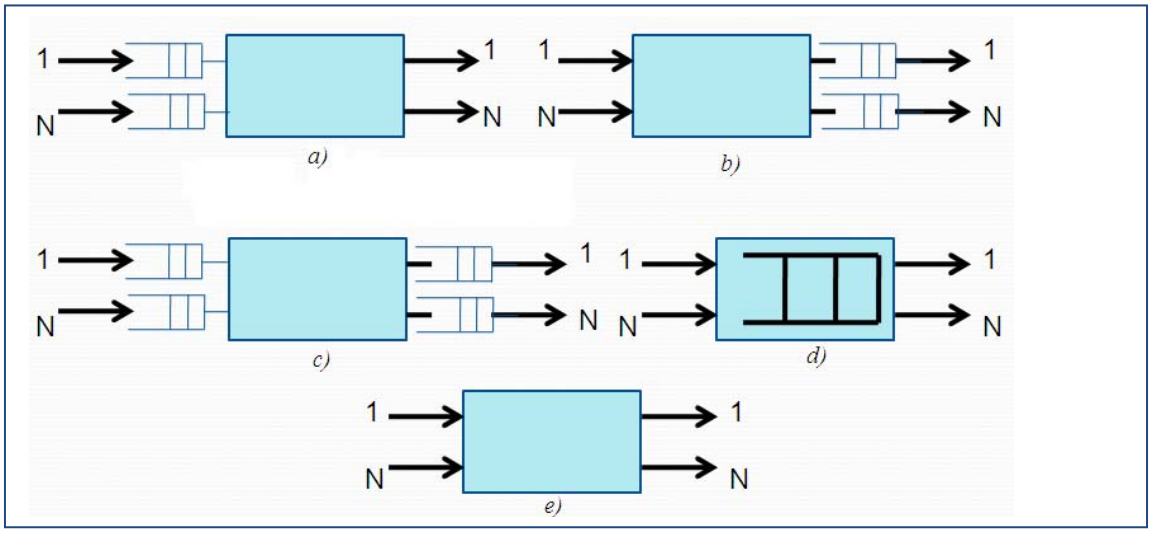

Figure 1.6 : **Dispositions possibles des mémoires tampons d'un commutateur :**

a) Tampon à l'entrée b) Tampon à la sortie c) Tampon à l'entrée et à la sortie d) Tampon partagé e) Aucun tampon

ii. Routeur

Le *routeur* se différencie d'un commutateur par le traitement de l'adresse du destinataire (adresse IP) indiqué dans le paquet IP. Un processeur lit l'en-tête du paquet et décide de l'envoyer sur son port de sortie selon le protocole de routage avec lequel il a été implémenté. C'est au niveau des routeurs (plus précisément routeurs IP) que les algorithmes de contrôle de flux et de qualité de service sont mis en place au niveau IP.

La figure 1.7 montre qu'un routeur agit au niveau paquet IP de TCP/IP ou au niveau 3 du modèle de référence OSI.

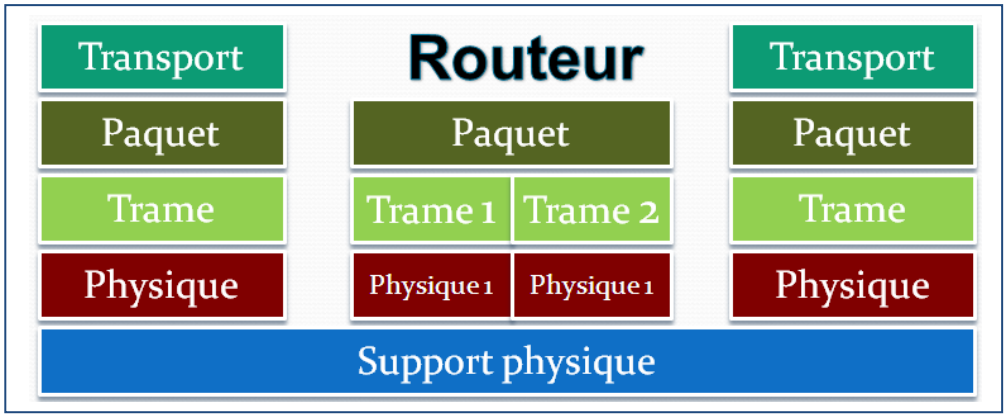

Figure 1.7 : **Architecture d'un routeur**

b) Caractéristique d'une liaison dans un réseau

Ainsi, du point de vue externe, le schéma d'un réseau peut être simplifié par un graphe (Fig 1.8) dont les nœuds représentent les équipements réseau et les arcs les liaisons. Une liaison est caractérisée par le *débit binaire maximal* (ou sa *capacité* en bit/s) avec lequel une quantité d'information peut être acheminée.

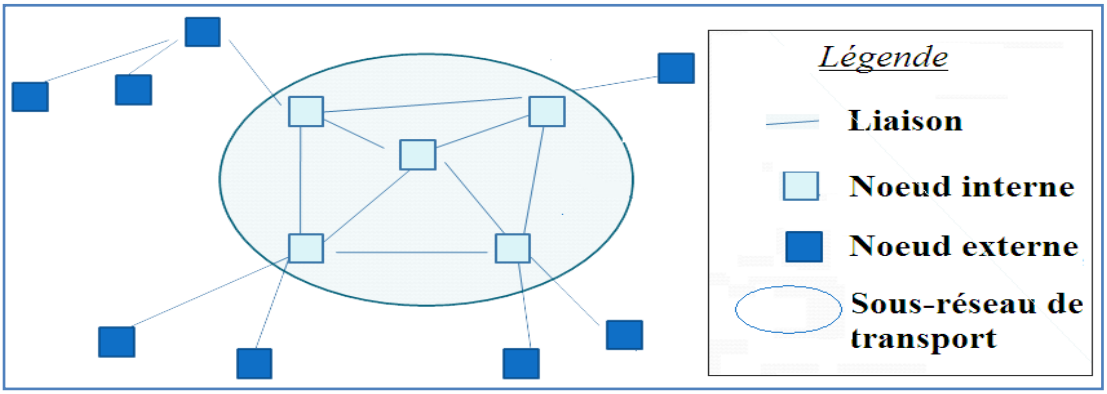

Figure 1.8 : **Modélisation d'un réseau par un graphe**

1.5 CARACTERISTIQUES ANALYTIQUES D'UN RESEAU

a) Trafic dans un réseau

i. Commutation et concentration de trafic

Considérons une application de type client/serveur (échange entre un terminal et un ordinateur par exemple). Si on examine le déroulement d'une session, limitée à un seul échange (Fig 1.9), on constate que :

– la durée de la session est bornée dans le temps, le support libéré est alors utilisable par un autre utilisateur : c'est la **commutation** ;

– pendant la durée de la session, le support n'est pas en permanence utilisé pour la transmission de données. Durant les instants de silence, le support est disponible pour un autre utilisateur ou une session différente : c'est la **concentration de trafic** [7].

ii. Intensité de trafic et taux d'activité

Dans un réseau, l'**intensité du trafic** mesure la durée de la session**,** par contre, le **taux d'activité** (ou *taux du trafic*) mesure l'utilisation effective du support c'est-à-dire le temps durant lequel le système traite au moins une quantité d'information (paquet ou message). Pour une unité de temps, ces deux grandeurs sont représentées par Fig 1.9 et exprimées par les équations (1.1) et (1.2).

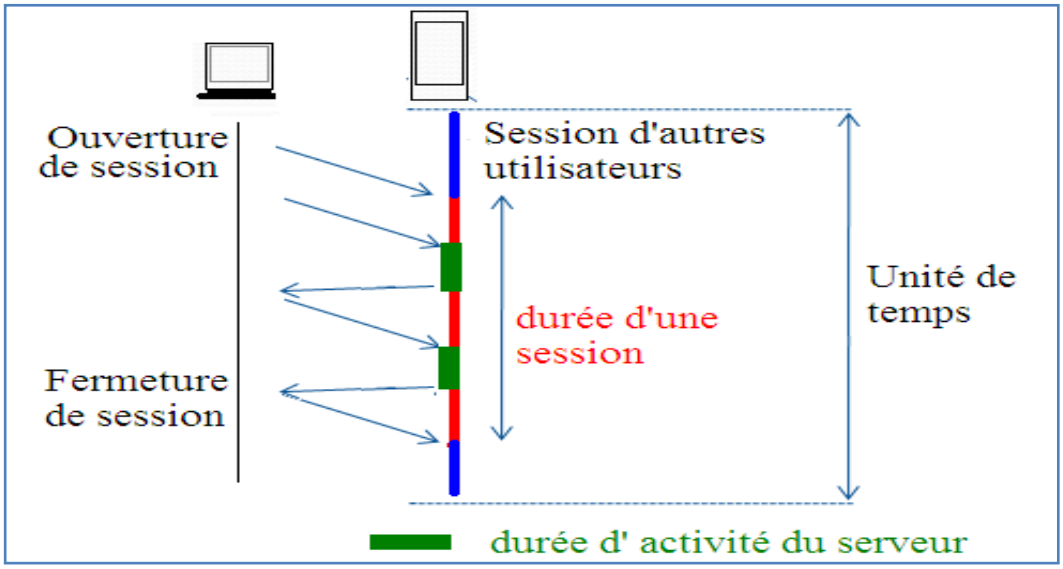

Figure 1.9 : **Intensité de trafic et taux d'activité d'un nœud de réseau**

Intensité de trafic $\rho = \frac{Durée\ de\ session}{Unité\ de\ temps}$ (1.1)

Taux d'activité $\theta = \frac{Durée\ d'activité}{Durée\ de\ session}$ (1.2)

iii. Temps de traversée T_R d'un paquet dans un nœud

Le *temps de réponse* T_r d'un système est la durée comprise entre l'envoi d'une commande et l'exécution de cette commande par ce système. Dans un réseau (Fig 1.10), c'est la durée de transmission d'un paquet (ou message) du nœud source vers le nœud destinataire comportant la *durée de traitement* du message obtenu (remontant les 7 couches *OSI*), le *temps de transit* dans le sous-réseau niveau 3 jouant le rôle d'un relai (somme des temps de traversée dans les nœuds intermédiaires) et des *délais de propagation* dans les supports de transmission.

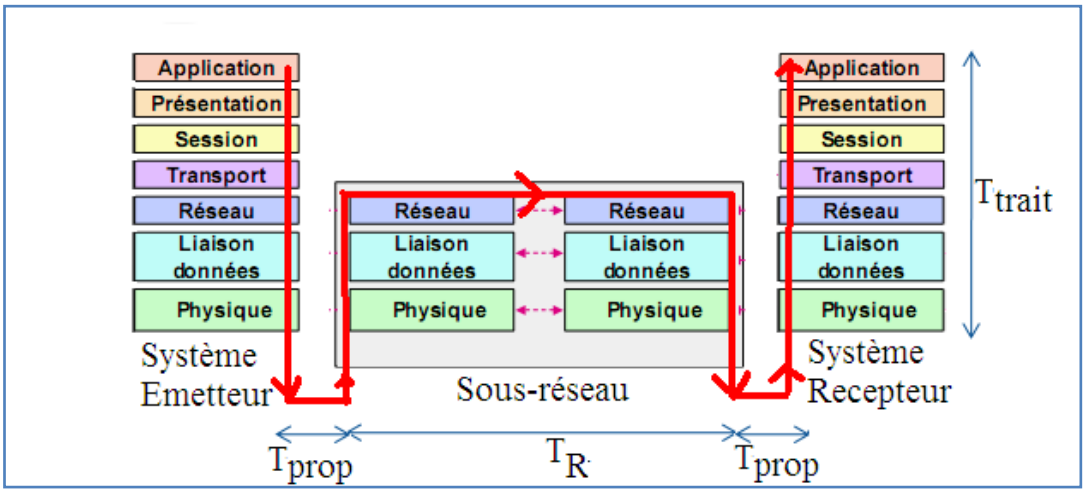

Figure 1.10 : **Temps de réponse d'un système émetteur-récepteur**

$$Temps\ de\ réponse\ T_r = T_{prop} + T_R + T_{trait} \quad (1.3)$$

Quand le trafic devient important, le temps de transmission d'une quantité d'information est essentiellement composé par l'ensemble des retards engendrés par le **temps de traversée T_R** de ces informations dans les nœuds intermédiaires, si bien que le temps de propagation T_{prop} et le temps de traitement T_{trait} dans les liaisons sont négligeables.

iv. Dimensionnement d'un système

Le **dimensionnement** d'un système consiste à attribuer à une de ses propriétés une valeur optimale de façon à ce que le système ne soit pas saturé. Il prend en compte tout équipement qui présente la propriété d'être libre ou occupé. La figure 1.11 représente par exemple la fluctuation du nombre instantané de paquets arrivant dans un nœud de réseau. Tout paquet arrivant au-delà d'une certaine limite sera perdu ou redirigé vers un autre nœud.

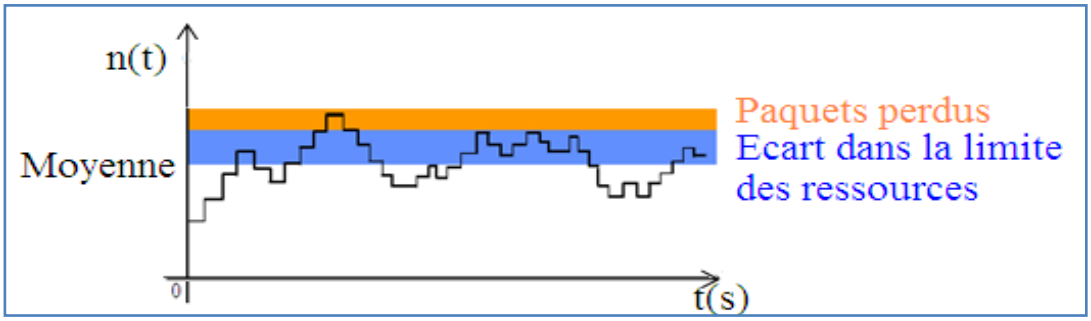

Figure 1.11 : **Fluctuation du nombre instantané de paquets arrivant à un nœud**

b) File d'attente dans un nœud de commutation

i. Présentation de la théorie des files d'attente

La théorie des files d'attente s'attache à modéliser et à analyser de nombreuses situations en apparence très diverses, mais qui relèvent néanmoins le schéma descriptif général suivant : des clients arrivent à intervalles aléatoires dans un système comportant un ou plusieurs serveurs auxquels ils vont adresser une requête. La durée du service auprès de chaque serveur est elle-même aléatoire. Après avoir été servis (ce qui suppose un arrêt chez un ou plusieurs serveurs selon le cas), les clients quittent le système. [8]

Voici quelques exemples de files d'attente : file d'attente à la caisse de magasin, file d'attente à la banque, salle d'attente d'un hôpital, système téléphonique, nœud dans un réseau informatique. Nous portons bien-sûr notre intention sur les deux derniers exemples ci-dessus appartenant aux domaines de la téléinformatique.

ii. Caractéristiques d'une file d'attente

Le but de l'analyse est de caractériser le degré de performance du système en répondant à des questions du type suivant:

- en moyenne, combien de temps attend un paquet avant d'être servi?
- A tout instant, quel est le nombre moyen de paquets dans le système?
- quel est le taux d'utilisation moyen des serveurs?

Le schéma synoptique d'une file d'attente est illustré par Fig 1.12

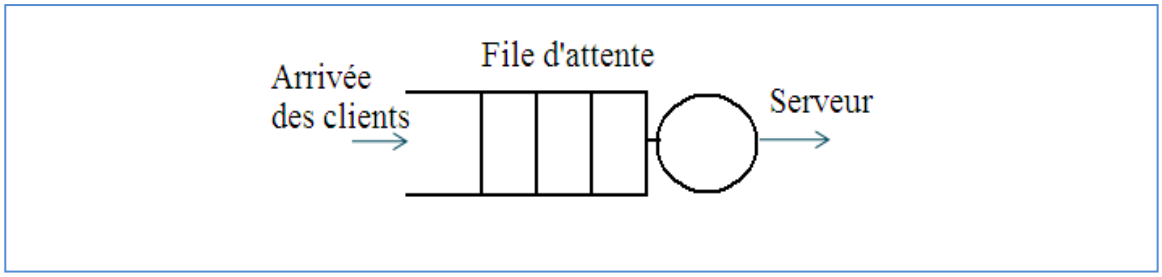

Figure 1.12 : **Schéma synoptique d'une file d'attente**

iii. Notation symbolique d'une file d'attente

Une file d'attente peut être classée selon ces caractéristiques suivant la notation *A/B/s/K* (notation de Kendhal) [8]:

- A : arrivée des clients. Elle est décrite à l'aide d'un processus de comptage
- B : temps de service.
- s : nombre de serveurs.
- K : capacité de la file (elle vaut infinie si omise). Tout client arrivant à l'entrée est rejeté si la file de capacité finie est pleine.

L'arrivée des clients est un processus de comptage. Elle est caractérisée par une fonction de densité de probabilité du temps séparant les arrivées. Le paramètre A vient de l'ensemble :

- M (Markov) si la distribution est exponentielle ou de Poisson.
- D (Déterministe) si la distribution est uniforme. Dans ce cas, les clients arrivent à intervalle régulier.
- G (général) pour les autres types.

Remarque :

Le même ensemble (M, D, G) est valable pour le paramètre de service B. Une notation plus étendue *A/B/s/K/m/Z* est souvent utilisée, où :

- m : la population des usagers (infinie si omise)
- Z : discipline de service. Elle peut être :
  - FIFO (*First In First Out*) ou premier arrivé, premier servi (par défaut)
  - LIFO (*Last In First Out*) ou dernier arrivé, premier servi
  - R (*Random*) pour aléatoire

iv. Formule de Little

Le temps de séjour de N paquets arrivant dans un système avec un taux d'arrivée λ est donné par la formule :

$$T_R = \mathrm{N}/\lambda \tag{1.4}$$

Cette formule présente un caractère général et reste valable pour des valeurs moyennes de T_R, λ et N.

1.6 RESEAU DE PETRI

Le Réseau de Petri (RdP) est un outil de modélisation graphique et mathématique qui s'applique à un grand nombre de domaines où les notions d'événements et d'évolutions simultanées sont importantes [10].

Il était surtout utilisé pour modéliser des systèmes à événements discrets, mais actuellement les modèles par RdP sont très divers. Parmi les applications, on peut citer : l'évaluation de performance des systèmes discrets, les protocoles de communication, les commandes des ateliers de fabrication, la conception de logiciels distribués et temps réels…

a) Concept de base

Un RdP est formé par un graphe dont les nœuds représentent une place ou une transition et les arcs (pondérés ou non) la liaison entre deux nœuds successifs. La figure

1.13 représente un exemple de RdP dont les composantes structurelles sont données dans Tab 2.

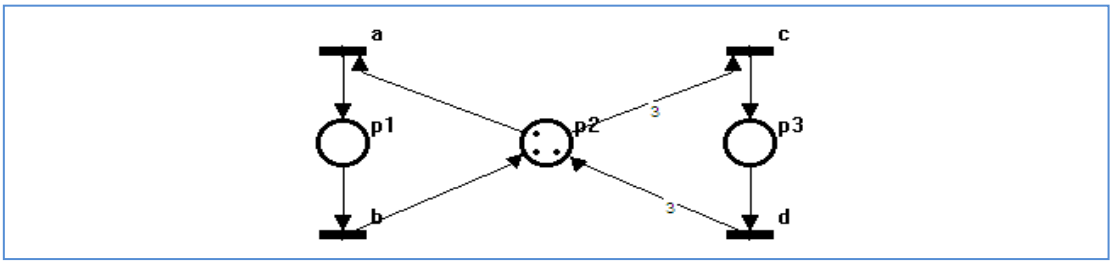

Figure 1.13 : **Exemple de Réseau de Petri**

Tableau 2 : **Composantes structurelles de Fig 1.13**

Type	Rôle	Représentation	Exemples
Nœuds	Place	Cercle	p1, p2, p3
	Transition	Trait	a,b,c,d
Arc	p → t	Flèche orientée	p2→a, p2→c, p1→b, p3→d
	t → p	Flèche orientée	a→p1, b→p2, c→p3, d→p2

Les matrices Pre (associée aux places précédentes) et Post (associée aux places suivantes) représentent respectivement les matrices obtenues en donnant le poids des arcs de chaque transition en fonction des places. Le tableau 3 donne le codage utilisé dans chaque cas.

Tableau 3 : **Codage des arcs dans les matrices Pre et Post**

Nom de la matrice	Arc entrant	Arc sortant	Sans contact
Pre	1	0	0
Post	0	1	0

Remarque : pour un arc pondéré, le poids est pris à la place du code.

Formellement, un Réseau de Petri est un quadruplet

$$R = < P, T, Pre, Post > \quad (1.5)$$

où P est l'ensemble de places, T l'ensemble de transitions.

Dans le cas de Fig 1.13, voici les résultats :

$$\text{Pre} = \begin{matrix}[a & b & c & d]\end{matrix} \begin{bmatrix}0 & 1 & 0 & 0\\ 1 & 0 & 3 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \begin{bmatrix}p1\\ p2\\ p3\end{bmatrix}$$

$$\text{Post} = \begin{matrix}[a & b & c & d]\end{matrix} \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 3\\ 0 & 0 & 1 & 0\end{bmatrix} \begin{bmatrix}p1\\ p2\\ p3\end{bmatrix}$$

Contrairement à la notation matricielle standard, Pre(p,t) (resp. Post(p,t)) désigne l'élément p-ème ligne et t-ème colonne de la matrice Pre (resp. de Post).

b) Types de RdP

i. RdP marqué

Un **RdP marqué** est défini par le couple

$$R_M = < R, M > \quad (1.6)$$

où R désigne le RdP et M le marquage initial du réseau c'est-à-dire le vecteur représentant le nombre de jetons dans chaque place.

Dans le cas de Fig 1.13,

$$M = \begin{bmatrix}0\\ 3\\ 0\end{bmatrix} \begin{bmatrix}p1\\ p2\\ p3\end{bmatrix}$$

Soulignons que la **transition t** est dite **franchissable** si et seulement si :

$$\forall p \in P\ M(p) \geq Pre(p,t) \quad (1.7)$$

et l'on note

$$M \geq Pre(.,t) ou\ M(t>) \quad (1.8, 1.9)$$

Graphiquement, t est franchissable si pour tout arc en entrée de cette transition, le nombre de jetons contenus dans la place précédente correspondante est au moins égal au poids de cet arc.

Dans le cas de Fig 1.13:

$M = \begin{bmatrix}0\\ 3\\ 0\end{bmatrix}$, Pre(. , a) = $\begin{bmatrix}0\\ 1\\ 0\end{bmatrix}$ donc $M \geq Pre(.,a)$ c'est-à-dire que a est franchissable, il en est de même pour c.

Si une transition t est franchissable pour un marquage M, le franchissement de t donne le nouveau marquage M' tel que

$$\forall p \in P\ M'(p) = M(p) - Pre(p,t) + Post(p,t) \quad (1.10)$$

On utilise également les notations : $M' = M - Pre(.,t) + Post(.,t)$ (1.11)

$$M(t > M') \ ou\ M \xrightarrow{t} M' \quad (1.12,\ 1.13)$$

Dans le cas de Fig 1.13, le franchissement de la transition a est illustré par Fig 1.14

$$M' = \begin{bmatrix}0\\3\\0\end{bmatrix} - \begin{bmatrix}0\\1\\0\end{bmatrix} + \begin{bmatrix}1\\0\\0\end{bmatrix} = \begin{bmatrix}1\\2\\0\end{bmatrix}$$

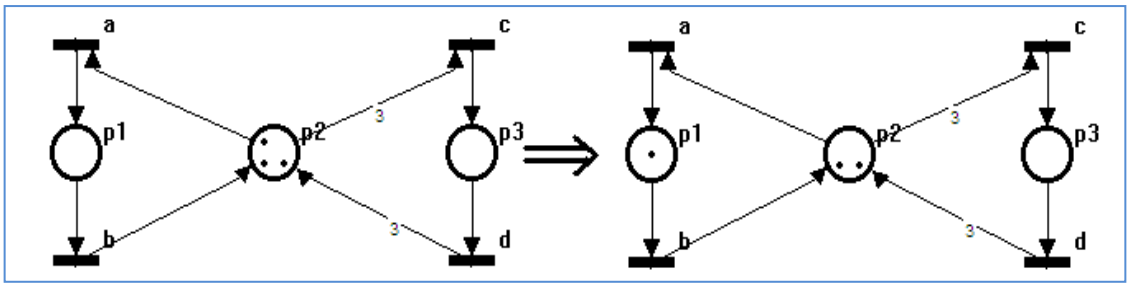

Figure 1.14 : **Franchissement d'une transition**

Le franchissement d'une transition consiste donc à enlever une quantité de jetons égale au poids de l'arc entrant dans la place source et à ajouter une quantité de jetons égale au poids de l'arc sortant dans la place destination [11].

ii. RdP avec arc inhibiteur

Un arc entrant valide la transition correspondante si son poids est supérieur ou égal au marquage de la place en amont. Un **arc inhibiteur** (muni d'un petit rond), par contre, valide une transition si son poids est inférieur au marquage de la place en amont. La figure 1.15 présente un exemple de validation d'un arc inhibiteur.

Le marquage d'une place associée à un arc inhibiteur n'intervient pas au franchissement de la transition, et un arc inhibiteur n'est utilisé que pour une entrée. Il est surtout utilisé pour des tests de validation de marquage (place vide dans la plupart des cas).

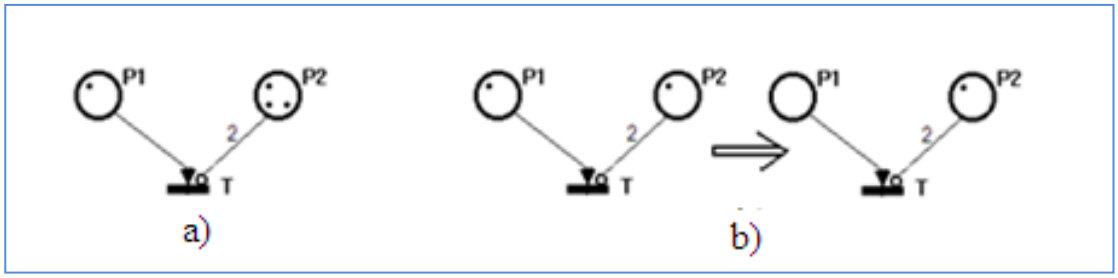

Figure 1.15 : **RdP avec arc inhibiteur :**
a) transition non-franchissable
b) franchissement d'une transition

Les RdP qui viennent d'être définis constituent les RdP ordinaires. D'autres classes de RdP interviennent dans la modélisation de notre étude, à savoir le RdP coloré et les RdP qui font intervenir le temps (temporisé et stochastique).

iii. RdP coloré

Les RdP que nous avons vus jusqu'ici ne permettent pas de distinguer les jetons qu'ils mettent en jeu. Dans le but de différencier les jetons, on les associe à des couleurs (entiers ou ensemble d'étiquettes), de même que pour les places qui les contiennent et les transitions mises en jeu. Ainsi, le franchissement de la transition sera considéré pour chacun des types de couleurs associées à cette transition.

Un RdP coloré est défini par le 6-uplet

$N_c = < P,T,C_{oul},C_{sec},Pre,Post, M_0 >$ où (1.14)

- P et T sont respectivement les ensembles de places et de transitions,
- C_{oul} l'ensemble de couleurs
- $C_{sec} : P \cup T \rightarrow P(C_{oul})$: fonction sous-ensemble de couleurs (couleurs acceptées par une place ou une transition donnée)
- Pre : $C_{sec}(t) \times C_{sec}(p) \rightarrow N$
- Post : $C_{sec}(t) \times C_{sec}(p) \rightarrow N$

- **Exemple de RdP coloré**

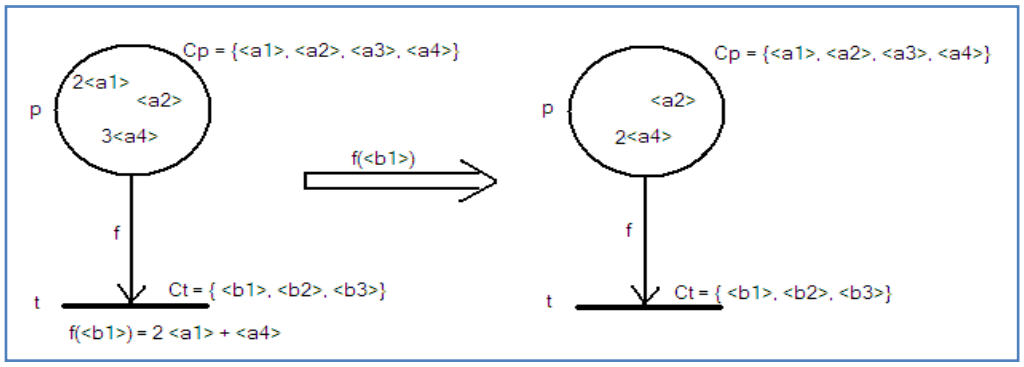

Figure 1.16 : **Franchissement dans un RdP coloré**

Considérons le réseau de Fig 1.16. Pour dire que le franchissement de t pour le type de couleur <b1> nécessite au moins 2 jetons de type <a1> et 1 jeton de type <a4>, on écrit :

f(<b1>) = 2<a1> + <a4>

= 2<a1> + 0<a2> + 0<a3> + <a4>

qui est un vecteur écrit sous forme de somme formelle. On définit de même f pour toutes les autres couleurs de Ct. Par exemple

f(<b2>) = <a2> +<a3>

La transition t est effectivement franchissable pour <b1> mais non pas pour <b2>. Ainsi, la condition de franchissement reste valable mais en considérant M(p) comme un vecteur de dimension égale au nombre de type de couleurs associées acceptées par la place p.

iv. RdP temporisé

Les RdP ordinaires décrivent une relation de causalité entre des événements modélisés par le franchissement d'une transition : un événement a est la cause de b, a précède b et a et b sont ordonnés dans le temps. Mais le temps peut être directement associé au RdP (aux places pour les RdP p-temporisés et aux transitions pour les RdP t-temporisés).

Un **RdP p-temporisé** est une paire

$$N_t = < N, \theta_f > \qquad (1.15)$$

- N est un RdP défini par (1.6)
- θ_f : fonction durée de franchissement

$\theta_f : T \rightarrow Q$ qui à chaque place fait correspondre un nombre rationnel positif décrivant la durée d'indisponibilité d'un jeton.

La sémantique est que les marques doivent rester dans la place p_i au moins le temps d_i associé à cette place. Pendant d_i la marque est indisponible ; elle ne participe pas à la validation des transitions. d_i représente donc la durée d'indisponibilité de la marque pour la validation des transitions ou encore le temps minimum de séjour d'une marque dans une place.

Le *RdP t-temporisé* peut être décrit de la même manière. d_i représente dans ce cas la durée de franchissement d'une transition t_i, c'est-à-dire la durée minimale de séjour d'une marque dans une place en amont de la transition t_i [12]. Dans la suite de cet ouvrage, ce type de RdP est utilisé pour créer un générateur périodique de jetons, tel représenté par Fig 1.17.

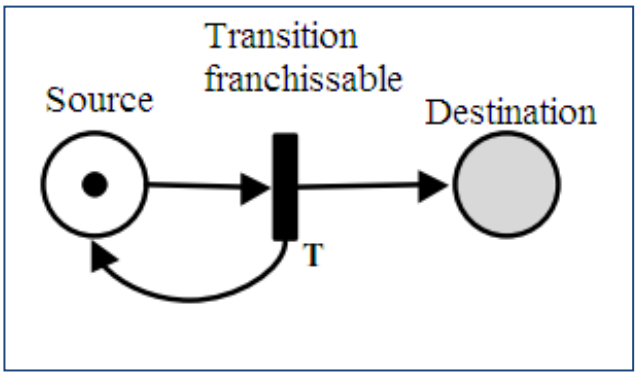

Figure 1.17 : **RdP t-temporisé générant périodiquement un jeton**

v. RdP stochastique

La littérature définit le *RdP stochastique* (RdPS) qui est obtenu à partir d'un RdP classique en associant des durées de franchissement aléatoires aux transitions pour l'évolution du marquage. Ce type de RdP est bien adapté pour la modélisation des phénomènes aléatoires où le temps s'écoulant entre deux événements n'est pas fixe. En outre, le RdPS est une sorte de RdP t-temporisé où les durées de franchissement associées aux transitions sont définies par des distributions exponentielles ou géométriques permettant de construire un processus Markovien (c'est-à-dire sans mémoire) équivalent [13].

Un RdPS est une paire

$N_{ts} = < N, \Lambda >$ (1.16)

- N est un RdP défini par (1.7)
- $\Lambda : t \rightarrow \lambda_t = \Lambda(t)$: taux de transition

Le RdPS qui sera utilisé dans cet ouvrage modélise une source de jetons suivant une loi de Poisson de paramètre λ telle représentée par Fig 1.18 .

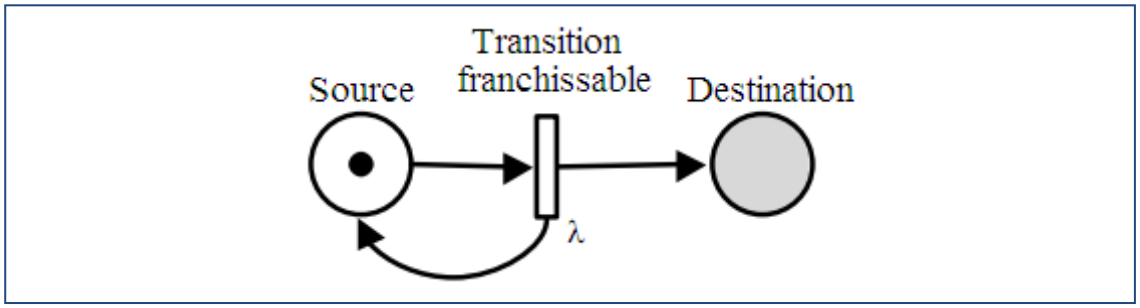

Figure 1.18 : **Génération aléatoire de jetons suivant la loi de Poisson de paramètre λ**

Conclusion

Ce chapitre est une brève introduction du système à étudier qu'est le nœud de commutation. Les notions rencontrées dans ce chapitre vont permettre de comprendre les mécanismes de base qui interviennent dans le traitement d'un paquet traversant ce nœud. Bien qu'aucune implémentation spécifique d'équipement réseau ne soit référencée, la suite de cet ouvrage essaie de se situer dans les cas réels rencontrés fréquemment dans la commutation par paquets surtout dans l'environnement IP. Le formalisme de RdP sera alors utilisé à cet effet.

Chapitre 2

MODELISATION D'UN NŒUD DE COMMUTATION PAR RESEAU DE PETRI

L'augmentation du nombre d'applications distribuées sur le réseau Internet telles que la vidéoconférence, la vidéo en ligne, la téléphonie sur IP (VoIP ou Voice over IP) exige une garantie de performance du service attendu par les utilisateurs du réseau. Dans le contexte de ce chapitre, il s'agit d'identifier ce mécanisme de fourniture de qualité de service (QoS ou Quality Of Service) en utilisant un modèle formel d'un nœud de commutation basé sur les RdP.

2.1 CADRE D'ETUDE DU NŒUD DE COMMUTATION

La modélisation des mécanismes du nœud de commutation dont il est question ici se situe dans le cadre du contrôle de flux dans un nœud de commutation de paquets. Bien qu'elle ne corresponde à aucune architecture de service particulière du réseau IP, les étapes de ces mécanismes se retrouvent dans la plupart des implémentations actuelles des routeurs.

a) Contrôle de flux dans les réseaux à commutation

i. Définition

Le contrôle de flux est une fonction essentielle effectuée au niveau d'un équipement de transfert (de paquets ou de trames). Il consiste à s'assurer que les paquets arrivent au récepteur dans le délai le plus court tout en évitant les pertes par débordement des mémoires tampons, en cas de surcharge du trafic dans les nœuds intermédiaires. Le paragraphe qui suit détaille quelques-unes de ces techniques de contrôle de flux.

ii. QoS

Le contrôle de flux s'évalue par la QoS que le nœud fournit à l'entité utilisateur, et qui est défini par un certain nombre de paramètres de performance. Par exemple, dans le cas des paquets IP, la qualité de service est spécifiée entre autres par : le taux d'erreur par paquet (*Packet Error Ratio*), le taux de perte de paquets (*Packet Loss Ratio*), le délai de transfert par paquet (*Packet Transfer Delay*), la variation du délai de transfert par paquet (*Packet Delay Variation*),… [14]

b) Approche modulaire

Considérons un flux de trafic particulier f qui entre dans le nœud de réseau à commutation par paquet représentant une file d'attente (Entrée – Tampon– Serveur), telle représentée par Fig 2.1.

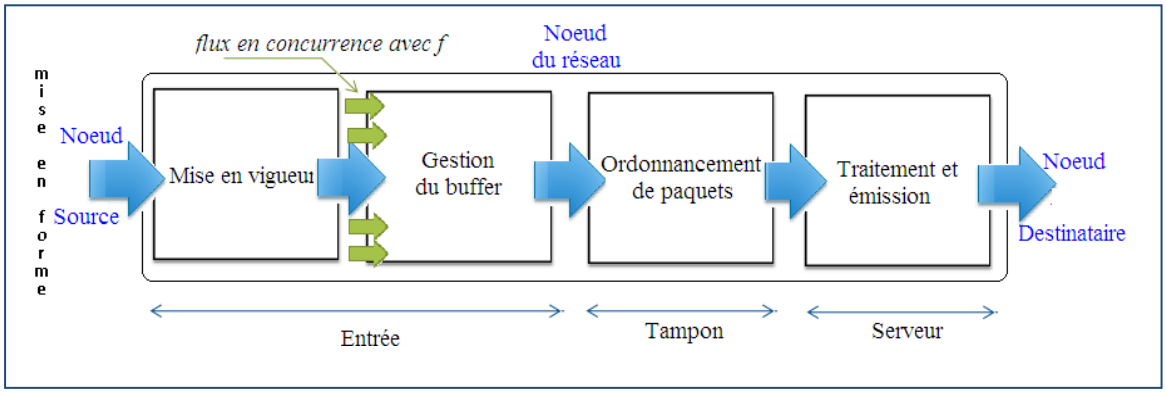

Figure 2.1 : **Modules de gestion de trafic entre nœud source et nœud destinataire.**

i. Sous-modules de mise en forme et de mise en vigueur

Ce sont des algorithmes qui détectent les paquets qui ne sont pas conformes à des contrats prédéfinis. Bien que ces deux sous-modules interviennent dans deux nœuds différents, la dépendance entre leurs fonctionnements internes a permis de les rassembler dans un même sous-module. Le mécanisme de *mise en forme* de trafic (*traffic shaping*) permet de régler le débit du trafic traversant le réseau, par exemple en affectant des retards au paquet. Le mécanisme de *mise en vigueur* (*traffic policing*) gère le trafic entrant dans un nœud en rejetant ou modifiant les flux de paquets qui ne sont

pas conformes à un certain critère. Un exemple pour chacun des deux mécanismes sera présenté dans la suite du paragraphe et illustré dans le cadre de notre étude [15].

- **Mise en forme par régulation de trafic**

Elle permet de lisser un flux erratique afin de limiter les risques de congestion ou encore d'adapter le flux entre deux réseaux à débits différents. Cette technique convient bien pour les bandes-passantes importantes si elle est bien paramétrée. La figure 2.2 illustre une méthode de régulation de trafic par le principe du régulateur à « seau à jeton » ou régulateur (σ, ρ). Il consiste à asservir le débit de sortie des paquets.

- **Mise en vigueur par écrêtage des pics de trafic**

L'écrêtage des pics de trafic sert à adapter le flux entre des réseaux à débits différents. Il est utile pour les nœuds traitant plusieurs types de trafics.

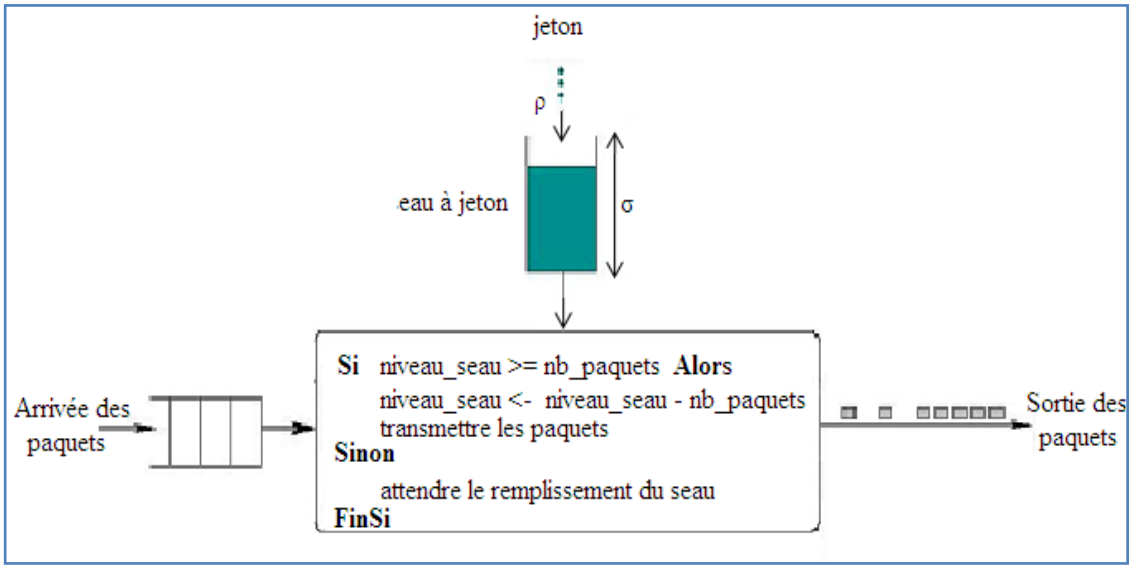

Figure 2.2 : **Principe d'un régulateur (σ, ρ)**

ii. Sous-module de gestion de buffer

- **Principe du *Drop tail***

Le mécanisme de gestion de paquets le plus simple consiste à rejeter tout paquet arrivant au nœud quand la file est pleine (*Drop tail*). Dans d'autres cas, le paquet sera redirigé, par défaut, vers un autre nœud ou sera stocké dans un autre tampon secondaire.

- **Algorithme de RED (*ou Random Early Detection*)**

Dans ce cas, les paquets sont aléatoirement rejetés lorsqu'on s'approche de la congestion. En effet, le protocole de transport, tel que TCP, réduit le volume du trafic (fenêtre d'émission) en cas de perte de paquets. Le principe est donc d'anticiper cette perte pour réduire le débit sans attendre la perte par saturation. Mais contrairement à l'algorithme du *Drop tail*, celui de RED utilise deux seuils. En dessous du premier seuil, le paquet est accepté. Quand ce premier seuil est franchi, le nœud de commutation peut commencer à perdre quelques paquets sélectionnés pour réduire le trafic de certaines sources. Au-dessus du second seuil, tous les flots passant par le nœud seront concernés.

iii. Sous-module d'ordonnancement de paquets

Cette étape est assurée par l'ordonnanceur de paquets. C'est à ce niveau que la classe de service est appliquée suivant laquelle l'ordonnanceur décide du traitement du paquet. Les mécanismes les plus utilisés sont les suivants.

- **Discipline FIFO** (First In First Out)

Les paquets suivent une queue (premier arrivé, premier servi). Cette discipline était implémentée par défaut dans le protocole IP où le nœud agit par le service dit « au mieux » (*Best effort*). Deux cas sont considérés selon que le tampon soit partagé par tous les flux de chaque trafic ou non.

- **Priorité fixe ou stricte** (SPQ ou *Strict Priority Queuing*)

Dans ce cas, tant qu'il y a de paquets prioritaires dans la file d'attente, ils seront traités avant de passer aux autres moins prioritaires. Par exemple, la classe de service sensible au délai est toujours prioritaire par rapport à la classe de service sensible à la perte. Différentes analyses ont montré que cette priorité fixe a tendance à augmenté le taux d'utilisation des ressources du nœud élevé.

- **Priorité relative** (WPQ ou *Weighted Fair Queuing*)

En cas de congestion, le nœud donne une priorité à certains trafics par rapport à d'autres. Pour ce faire, chaque file d'attente est affectée dynamiquement d'un poids. L'ordonnanceur laisse passer le trafic d'une file pendant une proportion de temps

correspondant à son poids. Cette technique est utilisée dans l'architecture d'un nœud DiffServ (*Differentiated Service*) présenté en annexe (Annexe 2).

2.2 MODELE DU SYSTEME PAR DES DIAGRAMMES UML

UML (**Unified Modeling Language**) est un outil de communication incontournable, utilisé sur des centaines de projets de par le monde. En effet, c'est un langage graphique de modélisation de données et de traitements [16]. UML possède neuf types de diagrammes de modélisation (cf. Annexe 4) mais notre étude n'en comportera que les diagrammes d'états-transitions.

Le diagramme d'états-transitions a pour objectif de représenter des traitements permettant de gérer le domaine étudié et est utilisé pour représenter tous les états possibles, ainsi que les événements provoquant un changement d'état du système à modéliser.

a) Point de vue « buffer »

Le buffer ou tampon mémoire peut être soit vide, soit plein, soit dans un état intermédiaire (que l'on va appeler « non-vide »). Ces états sont représentés par Fig 2.3, ils déterminent la politique de gestion du *buffer*.

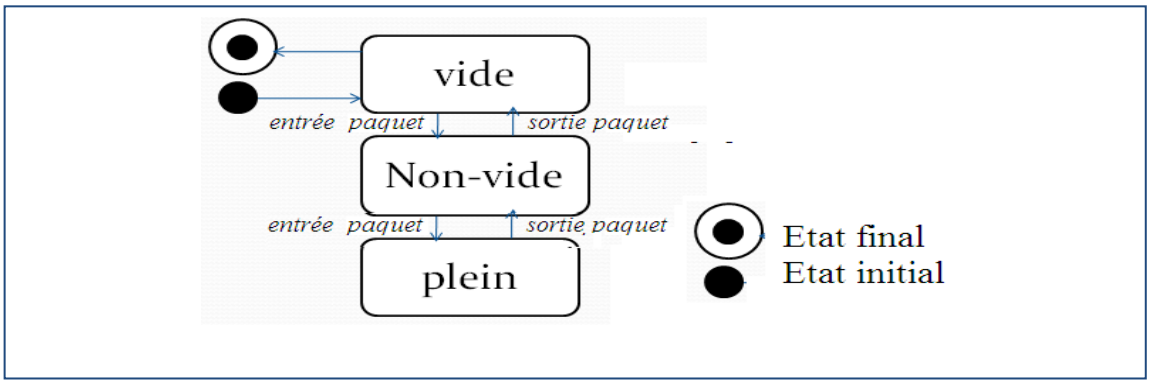

Figure 2.3 **: Diagramme états-transitions du modèle de buffer**

b) Point de vue « ligne de sortie »

La ligne de sortie est soit libre, soit occupée suivant qu'un paquet soit en cours de transmission ou non (Fig 2.4).

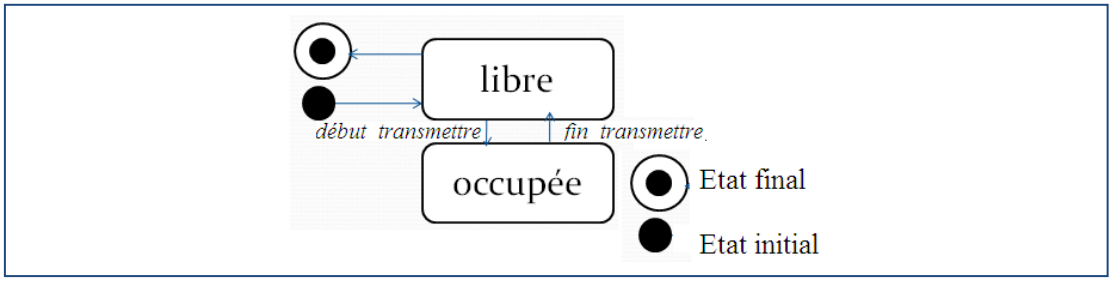

Figure 2.4 **: Diagramme d'activités du modèle de ligne de sortie**

2.3 MODELE DE RESEAUX DE PETRI DU NŒUD DE COMMUTATION

L'approche de modélisation que nous avons adoptée consistait d'abord à décomposer le système en des sous-modules fonctionnels. Pour migrer vers les Réseaux de Petri, les « états-transitions » des diagrammes UML seront utilisés pour élaborer les « places-transitions » du modèle. Enfin, le modèle du système final s'obtient par fusion des places des RdP.

Ainsi, pour un trafic donné, les paquets sont modélisés par des jetons et les actions des mécanismes par des transitions.

a) Sous module de source de trafic

i. Source de trafic Poissonien

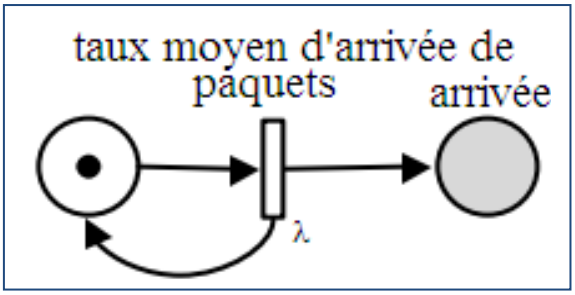

Figure 2.5 **: Source de trafic Poissonien**

La transition t est franchie aléatoirement suivant la loi de Poisson de paramètre λ. La loi de Poisson correspond assez bien à la réalité dans un système informatique comportant un grand nombre d'usagers. Les arrivées des paquets (jetons dans Pa « paquets arrivés ») sont indépendantes les unes des autres. Cette hypothèse est d'autant plus vérifiée qu'ils proviennent d'un grand nombre de sources aléatoires différentes comme c'est le cas dans les routeurs du réseau *Internet*.

ii. Source de trafic périodique

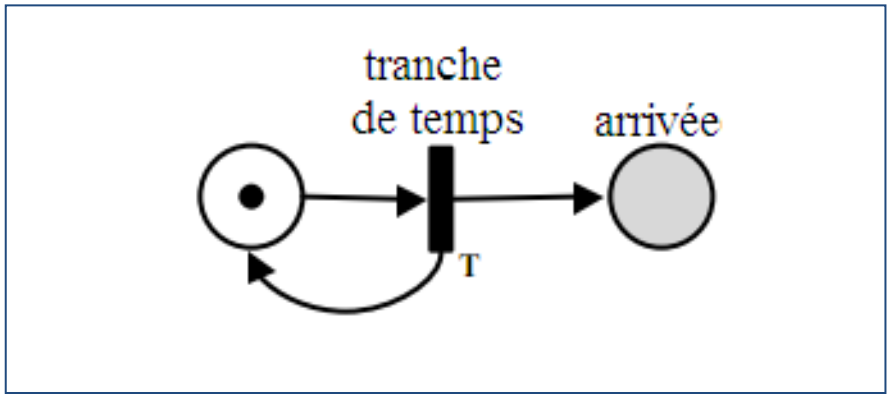

Figure 2.6 **: Source de trafic périodique**

La transition d'émission t (« tranche de temps ») est franchie périodiquement toutes les T secondes. Ce RdP modélise l'émission des paquets de taille fixe.

iii. Source de trafic de Bernouilli

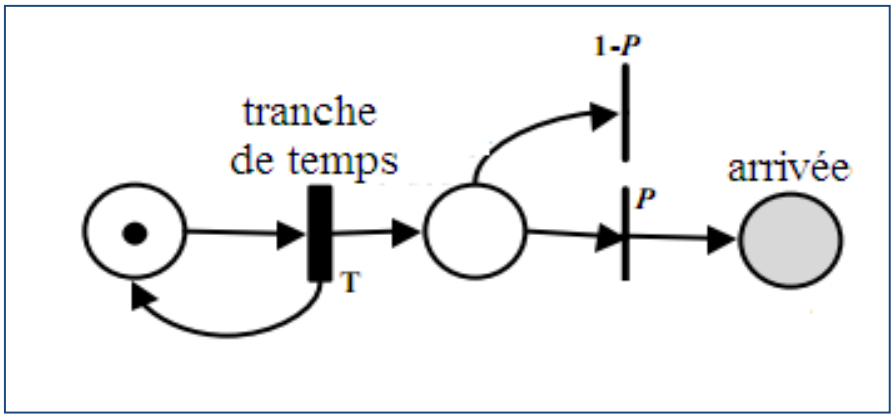

Figure 2.7 **: Source de trafic de Bernouilli**

La transition t est franchie périodiquement (toutes les T secondes). Mais une fois qu'un paquet est arrivé dans le tampon d'émission, la transition d'émission a une

probabilité p d'être franchie et 1-p d'être bloquée. Ce RdP modélise un système de transfert (par exemple une ligne de transmission) possédant une probabilité de perte p.

b) Sous-modules de mise en vigueur et de mise en forme

i. Régulation (σ, ρ)

La *régulation (σ, ρ)* (Fig 2.8) autorise l'émission de paquets tant que la place 'seau à jeton' est non-vide *(traffic shaping)*. Lorsque cette place est vide, tout paquet arrivant au nœud sera perdu *(traffic policing)*. Et le remplissage du 'seau' se fait périodiquement avec un débit ρ jetons par seconde.

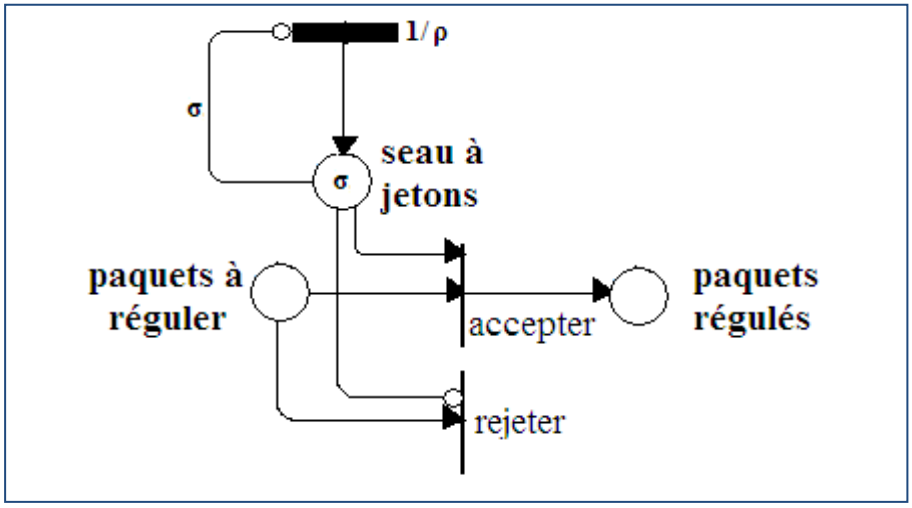

Figure 2.8 : **Sous-modules de mise en vigueur et de mise en forme : cas d'un régulateur (σ,ρ)**

Cette régulation autorise un trafic présentant un pic momentané (faux signe de congestion), puisque celui-ci n'est pas détecté tant que les ressources requises sont disponibles. Il n'y a donc aucune limite sur le débit de sortie des paquets.

ii. Régulation de débit crête

La *régulation de débit crête* (fig. 2.9) laisse passer un à un les paquets à chaque intervalle de temps T_{min}. Les autres paquets sont rejetés. Contrairement au régulateur (σ,ρ), un flux erratique (débit crête) momentané est tout de suite rejeté de sorte que le débit maximal du flux sortant est limité à 1/ T_{min} paquets par secondes.

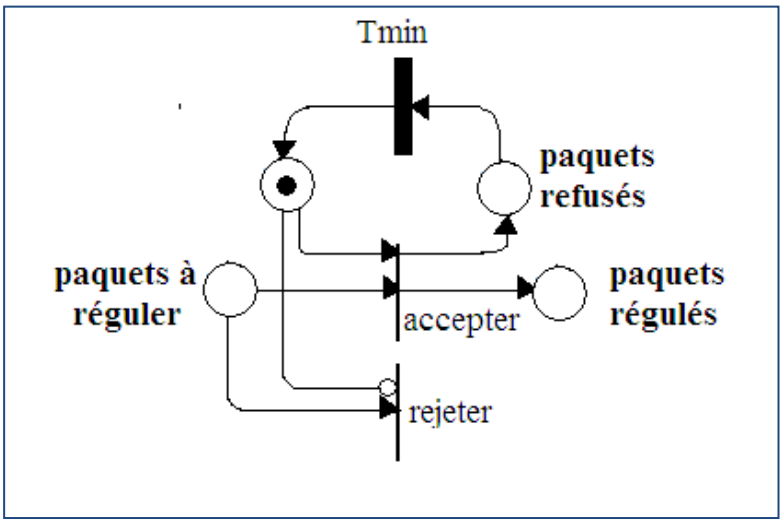

Figure 2.9 :**Sous-modules de mise en vigueur et de mise en forme : cas d'un régulateur de débit crête**

c) Sous-module de gestion du buffer

i. cas du « Drop tail »

Tout paquet arrivant au nœud est perdu si le buffer est occupé (Fig 2.10). Pour modéliser une file d'attente à capacité finie, il faut que la somme des jetons contenus dans les places '*buffer disponible*' et '*paquets mémorisés*' soit conservée.

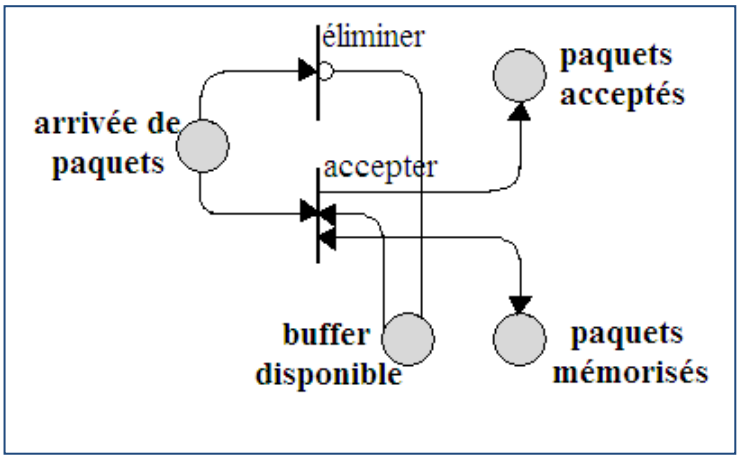

Figure 2.10 : **Sous-module de gestion du buffer par *Drop tail***

ii. Gestion du buffer par l'algorithme RED

La figure 2.11 montre que dans ce cas il y a deux types de rejets de paquets et deux types de passages de paquets:

- Si on est loin de la congestion (places *buffer>B* et *buffer>H* vides)

Alors tous les paquets sont acceptés (franchissement de la transition *buffer* ≤ *seuil_B*). Le système reste dans cet état tant que la place disponible dans le buffer est supérieure à *taille_max-seuil_B* (franchissement de la transition *seuil_B_franchi*).

- Si on s'approche de la congestion (places *buffer>B* non-vide et *buffer>H* vide)

Alors tous les (N+1) ème paquets sont perdus (place *compteur* contenant N jetons), les autres passent normalement. Dans le cas réel, un paquet a une probabilité 1/N d'être perdu. Le système reste dans cet état tant que la place *paquets mémorisés* contient plus de *seuil_B* paquets (transition *en-dessous de seuil_B*).

- Si on est dans la congestion (places *buffer>B* et *buffer>H* non-vides)

Alors tous paquets sont perdus sans exception. Le système reste dans cet état tant que la place *paquets mémorisés* contient plus de *seuil_H* paquets (transition *en-dessous de seuil_H*).

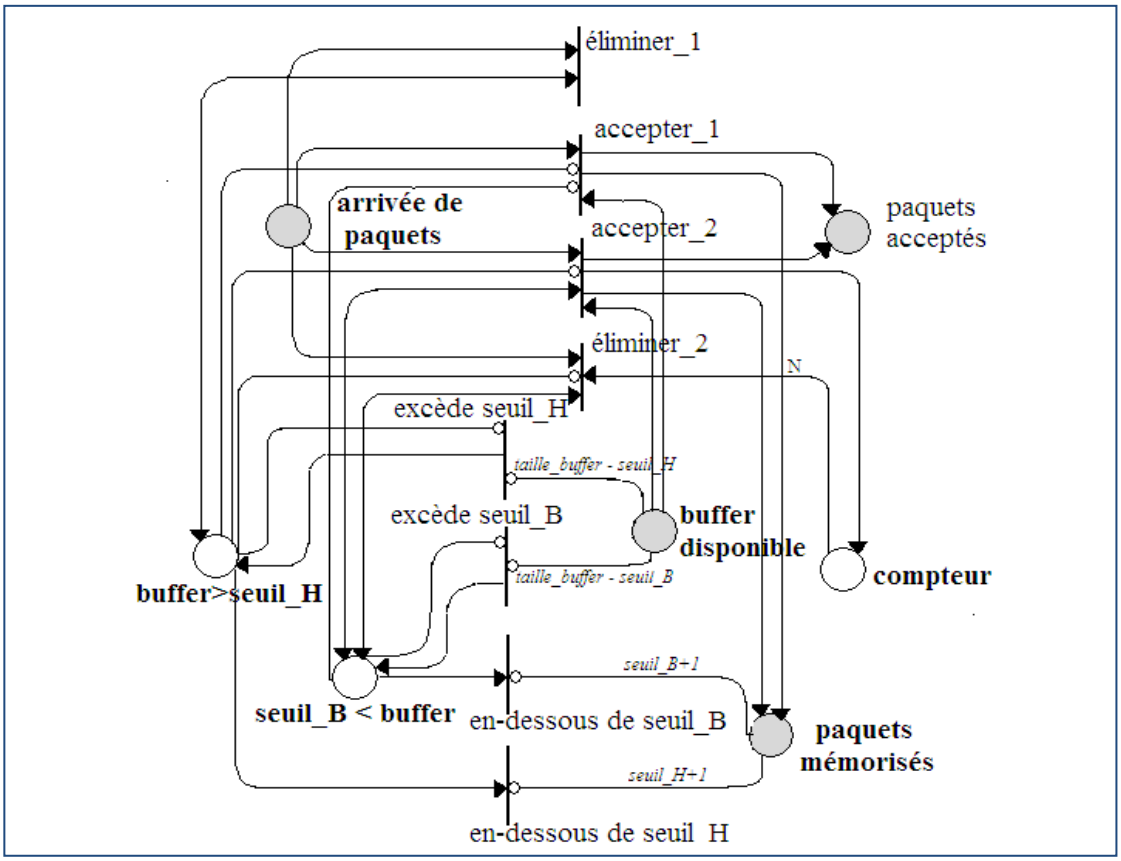

Figure 2.11 : **Sous-module de gestion du buffer par *RED***

d) Sous-module d'ordonnancement de paquets

i. Ordonnancement à priorité stricte

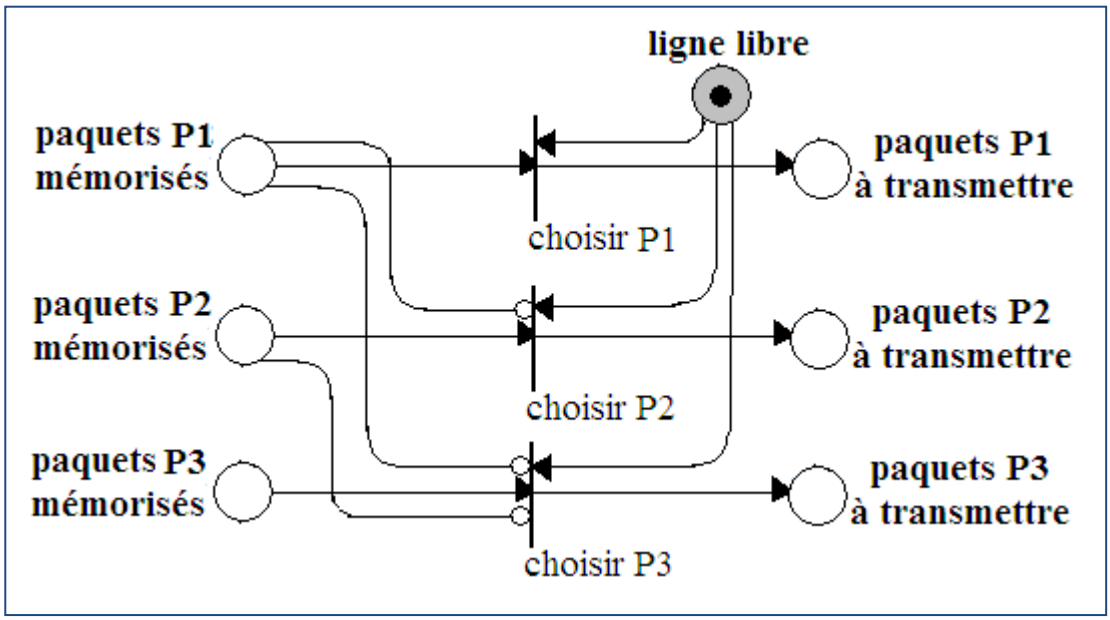

Figure 2.12 : **Sous-module d'ordonnancement à priorité stricte**

D'après Fig 2.12, aucune transition ne peut être franchie si la place '*ligne libre*' est vide. Cette place ne doit contenir qu'un jeton au plus pour modéliser une ligne transportant un seul trafic à la fois. Le franchissement de la transition '*choisir P1*' est le plus prioritaire par rapport aux deux autres à cause de la présence des arcs inhibiteurs. Il en est de même, '*choisir P3*' est inhibée tant qu'il y a de jetons dans le second trafic. ('*paquets P2 mémorisés*').

ii. Discipline de service FIFO

- **Nœud de commutation à tampon partagé**

Nous considérons d'abord le cas d'un tampon partagé par des types de trafics différents (Fig 2.13). Désignons par <p1>, … <p_n> les occupations disponibles dans le tampon. La transition '*choisir paquet* p_1' ne peut être franchie que par des jetons colorés occupant la place <p1>. Mais une fois que ce franchissement est effectué, les occupations de la place '*occupations disponibles*' sont mises à jour en faisant avancer le numéro d'occupation des paquets dans la file d'attente.

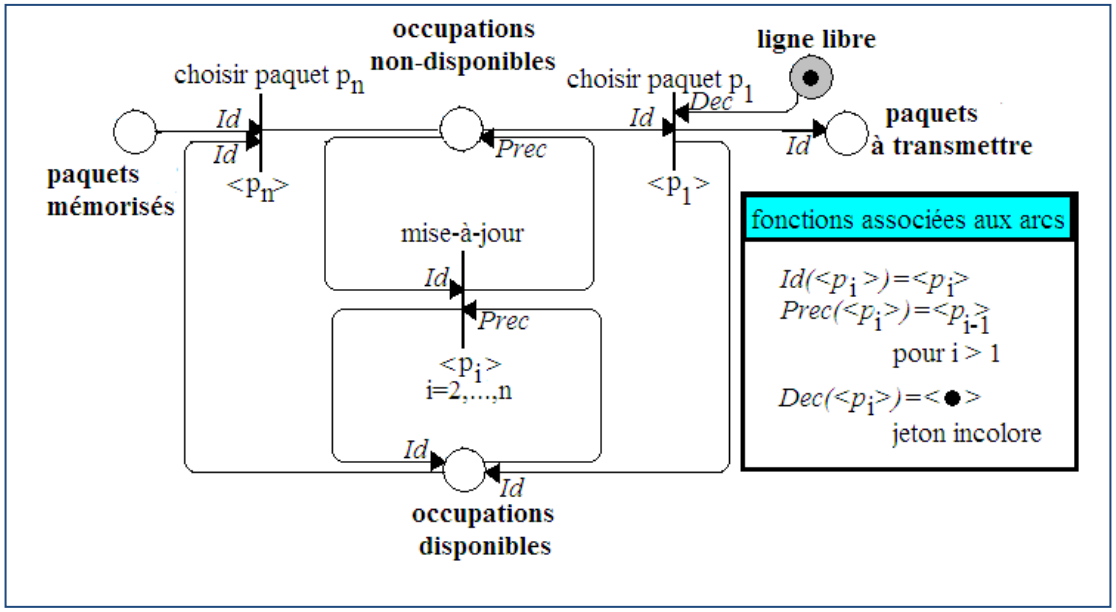

Figure 2.13 : **Sous--module d'ordonnancement à discipline FIFO : cas d'un tampon partagé**

- **Nœud de commutation à tampon séparé**

Dans ce cas, le premier jeton arrivé dans les places '*paquets mémorisés*' sera choisi dès que la ligne est libre.

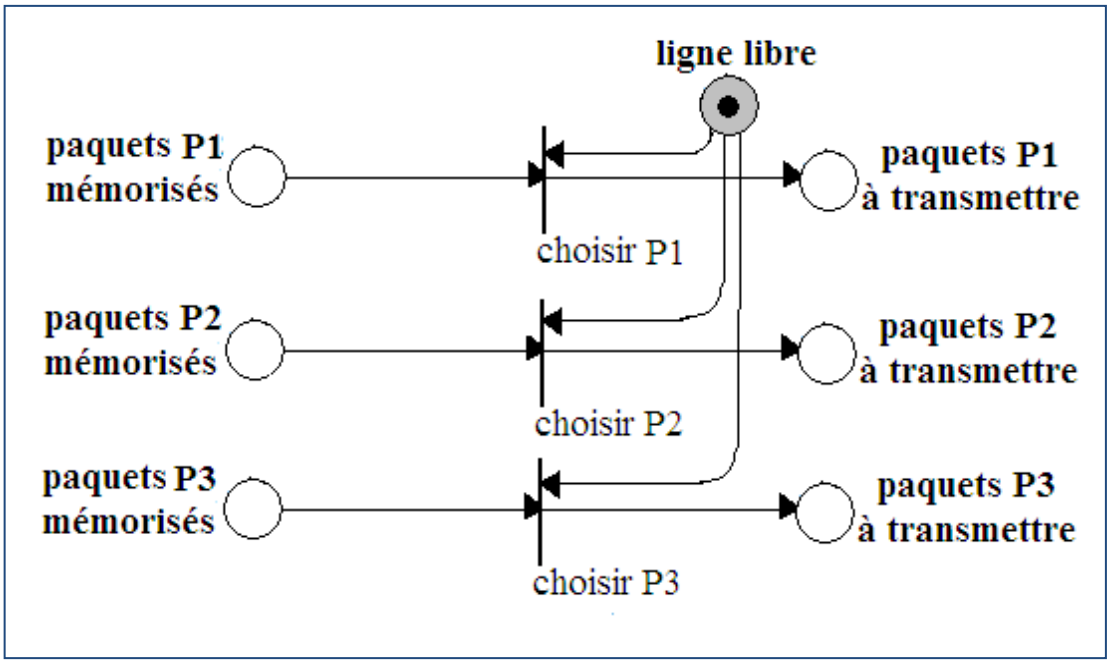

Figure 2.14 : **Sous-module d'ordonnancement à discipline FIFO : cas d'un tampon séparé**

e) Etats de la ligne de sortie et du buffer

La ligne de sortie est libre après la transmission d'un paquet (Fig 2.10). Un espace mémoire est disponible (ajout d'un jeton dans place '*buffer disponible'*) pour la réception d'un nouveau paquet.

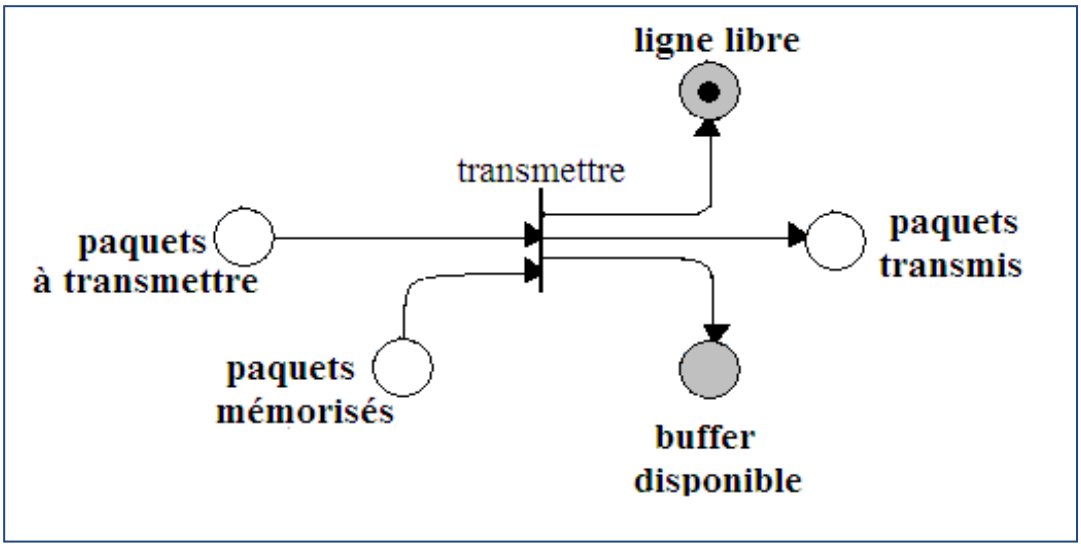

Figure 2.15 : **Ligne de transmission et du buffer**

Conclusion

Le monde IP s'est beaucoup préoccupé du contrôle de flux. Son intégration dans le nœud de commutation dans le réseau tel qu'*Internet* est délicate pour contrôler de façon fine des millions de flots de trafic afin que les utilisateurs soient satisfaits du comportement du réseau. Le modèle formel du nœud de commutation par RdP qui était présenté dans ce chapitre, montre qu'il est plus facile d'étudier l'architecture de ce mécanisme avec une vue d'abstraction plus élevée. Ainsi les résultats de la simulation du RdP vont permettre d'interpréter des cas possibles d'actions qu'un trafic donné peut subir.

Chapitre 3

SIMULATIONS LOGICIELLES

Les simulations présentées dans ce chapitre concernent la modélisation du nœud de commutation du chapitre précédent. Mais le développement du logiciel de simulation de RdP fait aussi partie de la ligne du projet fin d'études. Ainsi, ce dernier chapitre vise, dans un premier, à expliquer le fonctionnement du logiciel, ensuite à modéliser les exemples de l'étude. Enfin, c'est dans cette fin éducative qu'un autre exemple sur la modélisation de protocole de communication est proposé pour comprendre le formalisme par RdP.

3.1 IMPLEMENTATION DE *PETRISIM*

a) Présentation du logiciel

i. Exécution du programme

PetriSim est un logiciel développé sous la plate-forme MFC du Visual Studio® 2005. Initialement, il a été prévu interfacer avec un autre programme qui utilise ses résultats de simulation. En l'occurrence, un programme de simulation de file d'attente a été réalisé *sous Matlab® 7.0*. C'est pour ce besoin que le langage C++ a été choisi pour sa réalisation. Mais vu que des extensions du logiciel n'ont pas été achevées, cet interfaçage avec un logiciel tiers n'a pas été possible.

Pour cette première version, le logiciel est composé d'un simple fichier exécutable qui occupe moins de 4Mo dans le disque dur et ne demande pas un processeur très performant (nous l'avons essayé avec Pentium® III sous Windows XP®). Pour lancer le programme, il suffit de double-cliquer sur l'icône correspondant (Fig 3.1)

Figure 3.1 : **Icône du fichier exécutable de *PetriSim***

ii. Fonctionnalités du logiciel

PetriSim est un simulateur de RdP supportant les RdP Petri simples, les RdP avec arcs inhibiteurs et les RdP colorés. Actuellement, les RdP ne font pas encore l'objet d'une norme internationale, mais nous adoptons les notations utilisées par l'ouvrage référencé par [12] qui sont adoptées par la plupart des littératures traitant ce sujet. Les fonctionnalités de base du logiciel se résument comme suit.

D'abord, la simulation d'un RdP peut se faire de deux manières : soit en mode *pas-à-pas* dans lequel l'utilisateur active une à une le franchissement de chaque transition, soit en mode *rafale* dans lequel le logiciel active automatiquement les transitions franchissables tant qu'il n'y a aucun conflit structurel (c'est-à-dire que deux transitions ne peuvent pas être simultanément franchissables).

Avant toute simulation, un outil de vérification permet d'afficher les éventuelles erreurs lors de la construction du schéma du RdP, à savoir : la présence d'un nœud non-connecté et d'un composant non-paramétré.

Ensuite, les résultats statistiques de la simulation peuvent être recueillis dans un fichier texte ou dans un fichier *html*. Ils comprennent, par exemple, le nombre moyen de jetons contenus dans chaque place pour tous franchissements de transitions, le nombre moyen de franchissement de chaque transition, etc.

Enfin, les RdP colorés supportent les notions de couleurs complexes de la forme $< a_i, b_j, \ldots >$. En général, il n'y a pas de restriction quant au nombre de couleurs, l'allocation de ses jetons colorés se font dynamiquement mais un trop grand nombre ralentira le logiciel.

b) Développement du logiciel

i. Architecture de développement

La figure 3.2 représente l'architecture de développement du logiciel sous-forme d'un diagramme de classes d'entités. Nous remarquons que les classes de base représentant les composants du RdP héritent de la classe-mère *CObject* pour pouvoir utiliser la méthode de sauvegarde générique (ou *serialization*) de MFC.

Chacun de ces objets a donc été modélisé par une classe différente (classes *Place, Transition, Arcor et Arcinh*). Néanmoins, les places et les transitions ont des composants similaires : un nom, mais surtout un tableau comportant des pointeurs vers

leurs prédécesseurs et vers leurs successeurs (c'est à dire des arcs). Elles possèdent en outre des méthodes d'affichage et de détection similaires dans la vue *CPetriView* (Fig 3.3).

Afin de modéliser ce comportement semblable, nous avons créé une classe de base (classe *Base*) dont les places et les transitions héritent. De plus, cet héritage était indispensable, puisqu'un arc possède obligatoirement un prédécesseur et un successeur, mais selon le cas, il peut s'agir d'une place ou d'une transition. Il était donc indispensable de pouvoir utiliser un membre vers un objet commun, donc vers un *Base.*

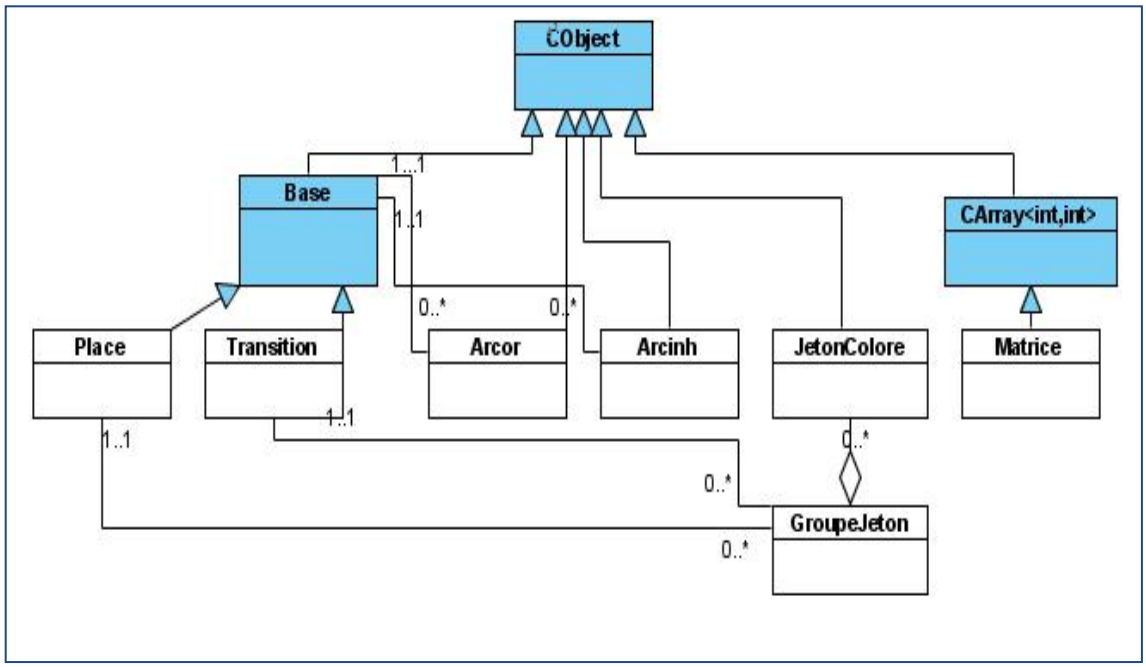

Figure 3.2 : **Diagramme de classes d'entités des RdP**

ii. Outils de développement

L'architecture MFC est organisée autour de 5 classes par défaut (générées par *Class Wizard*, résumées dans Tab 2) qui ont pour rôle l'interaction de *Windows* avec les composants principaux de la fenêtre MFC : le *document* et la *vue*. Elle est décrite par Fig 3.3.

Tableau 4 : **5 classes de base générées automatiquement par MFC**

Classes	Rôles
CPetriApp	C'est à partir de cette classe que l'application est réellement crée. C'est donc dans cette classe qu'il faut faire les initialisations globales : fichier INI, nouvelles vues…
CMainFrame	C'est elle qui gère par défaut les barres d'outils et les menus. Dans l'application, elle gère essentiellement la création de nouvelles fenêtres.
CChildFrame	Cette classe présente uniquement dans le cas d'applications *MDI* (*Multiple Document Interface*): elle représente une fenêtre fille et fait le lien entre la vue, le document et la *MainFrame*.
CPetriView	C'est elle qui fait le travail le plus important. Elle gère entre autres le dessin du réseau et l'impression. Nous lui associons également les différents évènements (clics souris, frappe touche…) et une partie des actions liées aux menus et à la barre d'outils.
CPetriDoc	C'est cette classe qui représente le document de travail et qui comporte donc toutes les informations à sauvegarder concernant le Réseau de Petri.

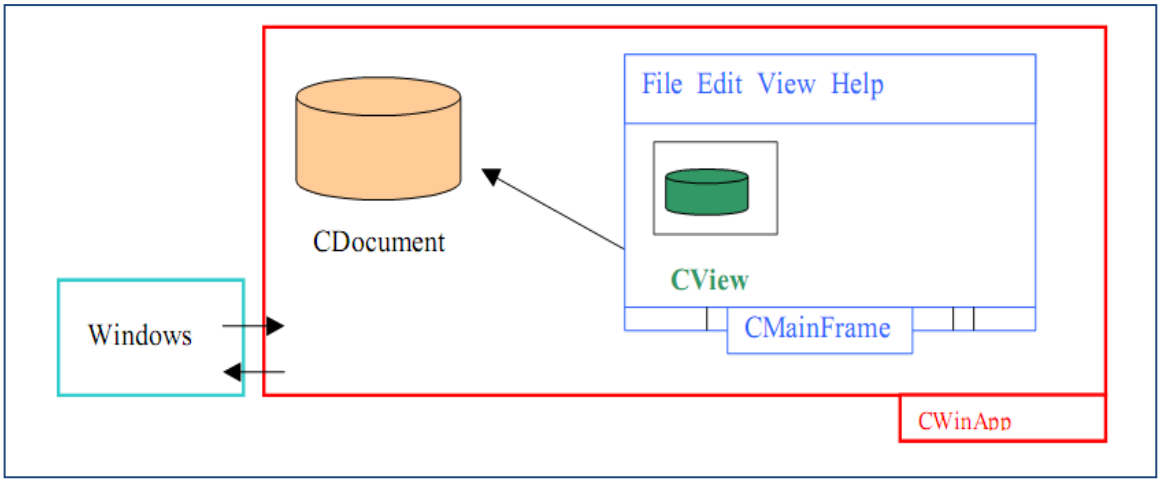

Figure 3.3 : **Vue globale de l'architecture MFC**

c) Interface logicielle de *PetriSim*

i. Fenêtre principale du logiciel

La figure 3.4 représente une partie de la fenêtre principale du logiciel. Quelques boutons de la barre d'outils sont mis à disposition de l'utilisateur pour dessiner chaque élément du graphe : places, transitions, arcs, arcs inhibiteurs et gomme (suppression d'objets). Une fois une place et une transition dessinées, le bouton "arc" relie les deux éléments par simple clic souris sur ceux-ci.

On définit les caractéristiques de chaque élément du graphe par double clic sur sa représentation graphique. Les différents mouvements de la souris servent à gérer le dessin d'un réseau, le déplacement d'objet, le paramétrage d'objet et le menu contextuel du programme.

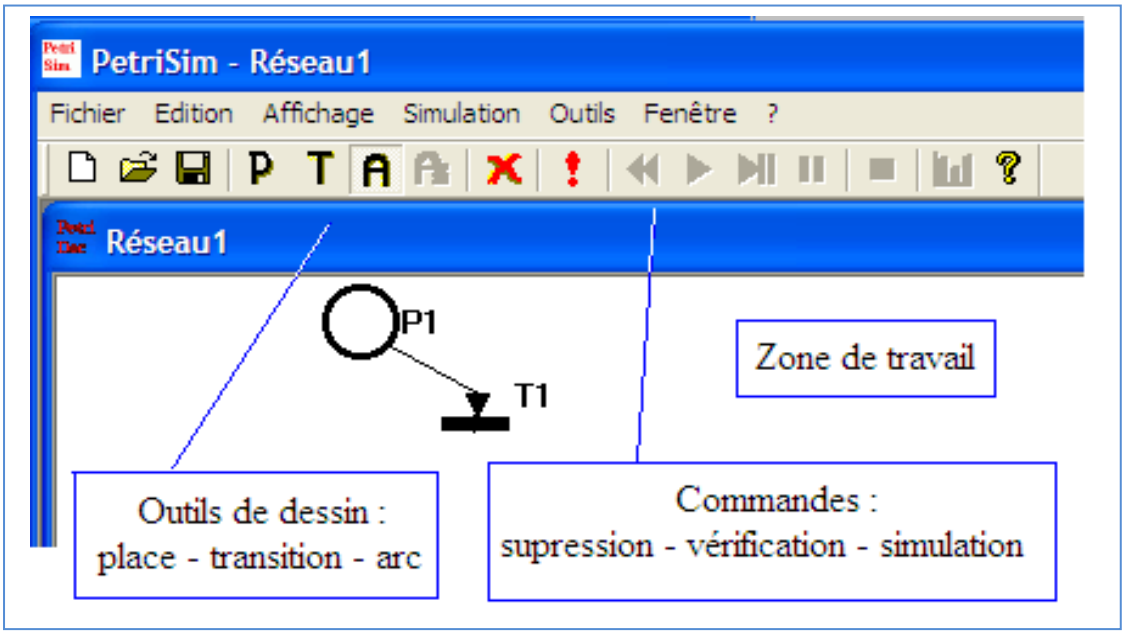

Figure 3.4 : **Barre d'outils et barre de menu du logiciel**

ii. Cas d'un RdP simple

- **Paramétrage d'une place**

La boîte de dialogue représentée par Fig 3.5 s'affiche par double clic sur une place. *Marquage* désigne le nombre de jetons contenus dans la place et *Capacité* le nombre maximal de ces jetons. Dans la définition d'un RdP, une place n'est pas limitée

à un nombre fini de jetons, cette option sert à éviter un débordement de la mémoire allouée dynamiquement lors de la simulation.

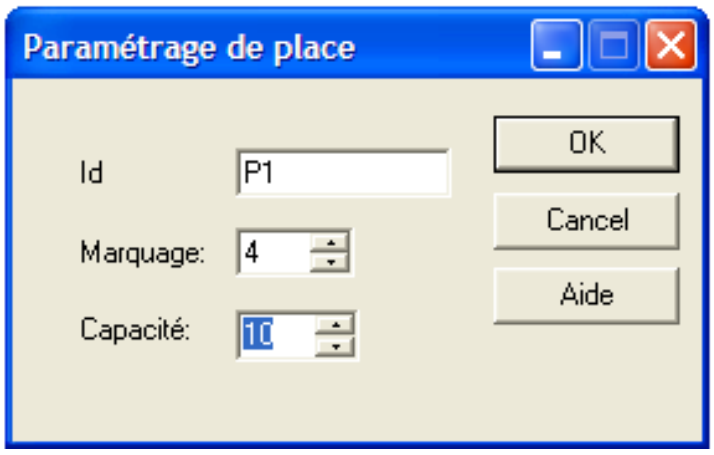

Figure 3.5 : **Paramétrage d'une place dans un RdP simple**

- **Paramétrage d'une transition**

Le paramétrage d'une transition consiste tout simplement à nommer la transition.

- **Paramétrage d'un arc**

Quant à l'arc (orienté ou inhibiteur), il faut paramétrer le poids, c'est à dire le nombre de jetons nécessaires au franchissement de celui-ci. Par défaut, le poids de l'arc est de 1. Une boîte de dialogue représentée par Fig 3.6 s'affiche alors pour ce paramétrage.

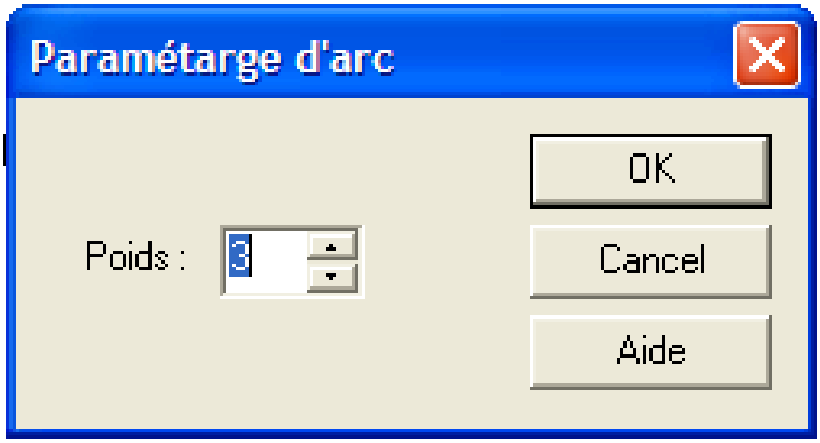

Figure 3.6 : **Paramétrage d'un arc dans un RdP simple**

iii. Cas d'un RdP coloré

- **Création de jetons colorés**

En mode coloré, la création des jetons peut se faire de deux manières différentes, soit directement par l'utilisateur, soit dynamiquement en cours de simulation. La création utilisateur peut se faire dans le menu *Outil / Couleurs et Jetons* ou bien directement dans la boîte de dialogue d'une place. La boîte de dialogue de Fig 3.7 suivante s'affiche.

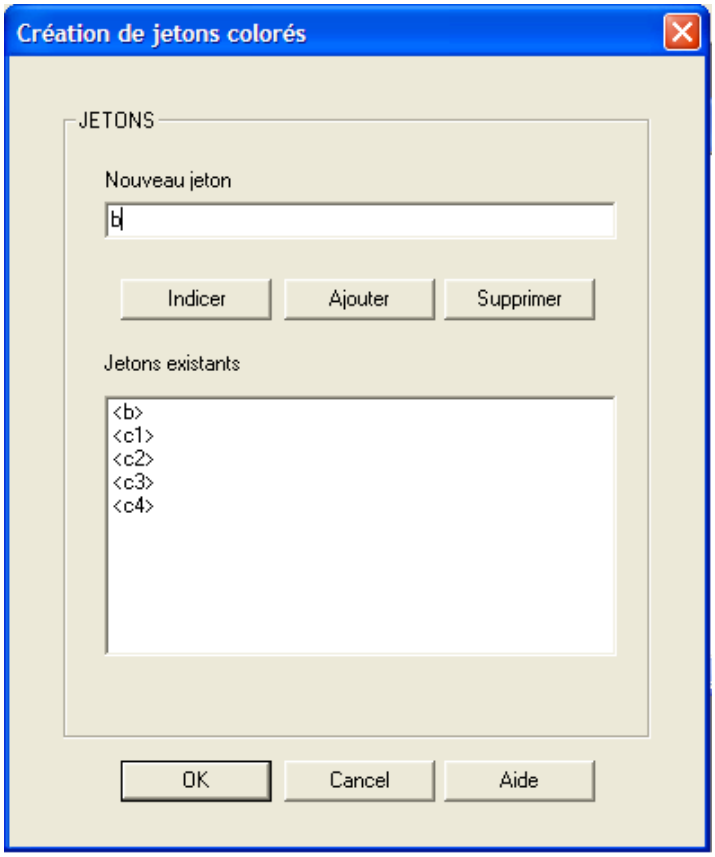

Figure 3.7 : **Boîte de dialogue de création de jetons colorés**

- **Paramétrage d'une place**

Pour paramétrer une place dans un RdP coloré, on retrouve les mêmes caractéristiques que pour un réseau simple, sachant que chaque jeton existant dans le réseau est affiché à l'ouverture de la boîte de dialogue (Fig 3.8), avec un marquage par défaut de 0. C'est donc à l'utilisateur d'éditer la quantité souhaitée pour chaque jeton. Il est possible de réinitialiser la valeur de tous les jetons par appui sur le bouton "*Reset*".

A l'aide du menu contextuel (clic bouton droit sur une place), il est possible de faire apparaître une fenêtre représentée par Fig 3.9 contenant le marquage complet de la place correspondante, ce qui permet de visualiser le marquage d'une place pendant la simulation sans surcharger le dessin.

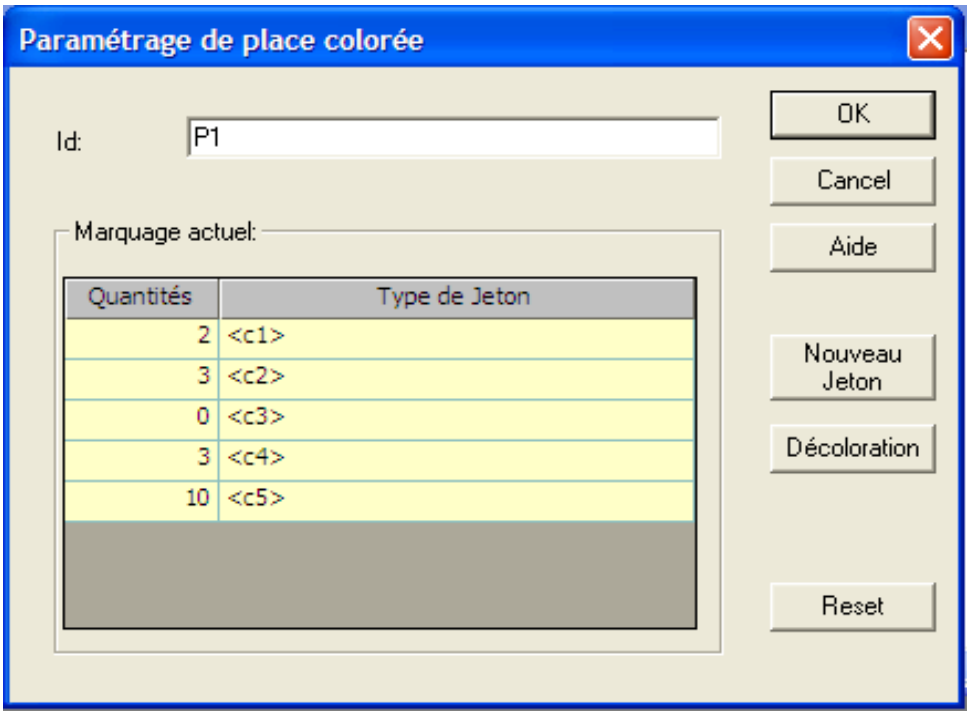

Figure 3.8 : **Paramétrage d'une place dans un RdP coloré**

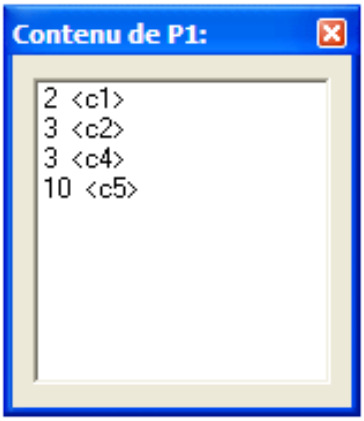

Figure 3.9 : **Marquage d'une place dans un RdP coloré**

- **Paramétrage d'une transition**

Le paramétrage d'une transition se fait par une boîte de dialogue du menu contextuel de cette transition, représentée par Fig 3.10. La présence de jetons différents nécessite de paramétrer un ensemble de franchissement, qui déterminera quels jetons

pourront éventuellement franchir la transition en cours de simulation. Plusieurs paramètres sont à prendre en compte.

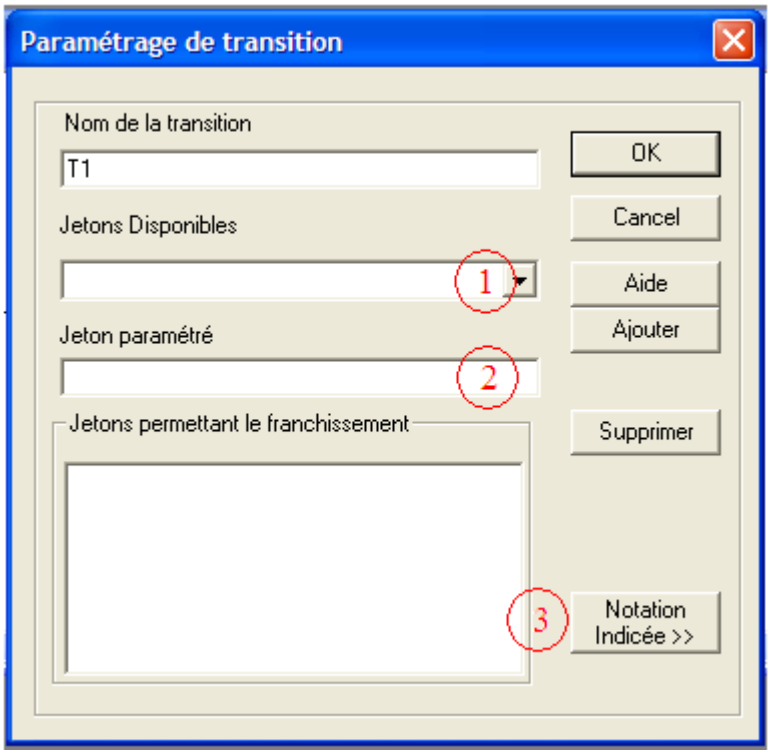

Figure 3.10 : **Paramétrage d'une transition dans un RdP coloré**

Pour cela, trois possibilités sont offertes à l'utilisateur:

- (1) Choix dans une liste déroulante proposant tous les jetons définis dans le menu Outil/Couleurs.
- (2) Paramétrage d'un jeton possédant des composantes indéfinies représentées par le caractère "?".

Par exemple, dans la boîte ci-dessus, on désire définir un ensemble de franchissement incluant tous les jetons possédant une première couleur "a" et une seconde couleur quelconque. Pour ces deux modes d'édition, un clic sur le bouton "ajouter" est nécessaire pour valider le jeton choisi et le faire figurer dans la liste en dessous.

- (3) Création d'un ensemble de jetons indicés

Dans ce dernier cas, une extension de la boîte de dialogue (Fig 3.11) apparaît. On entre alors le type de jeton que l'on désire paramétrer. Dans l'exemple qui suit (Fig 3.12), on désire définir l'ensemble $< a,b_i >$ avec i variant de 1 à 3. On définit d'abord les couleurs génériques (a,b) du jeton, puis un clic sur le bouton "paramétrage" fait

apparaître une boîte permettant de définir l'intervalle correspondant à chaque couleur. Il reste alors à cocher ou non la case "indice" selon qu'on désire ou pas définir un intervalle pour cette couleur, et à paramétrer les valeurs minimales et maximales de celui-ci.

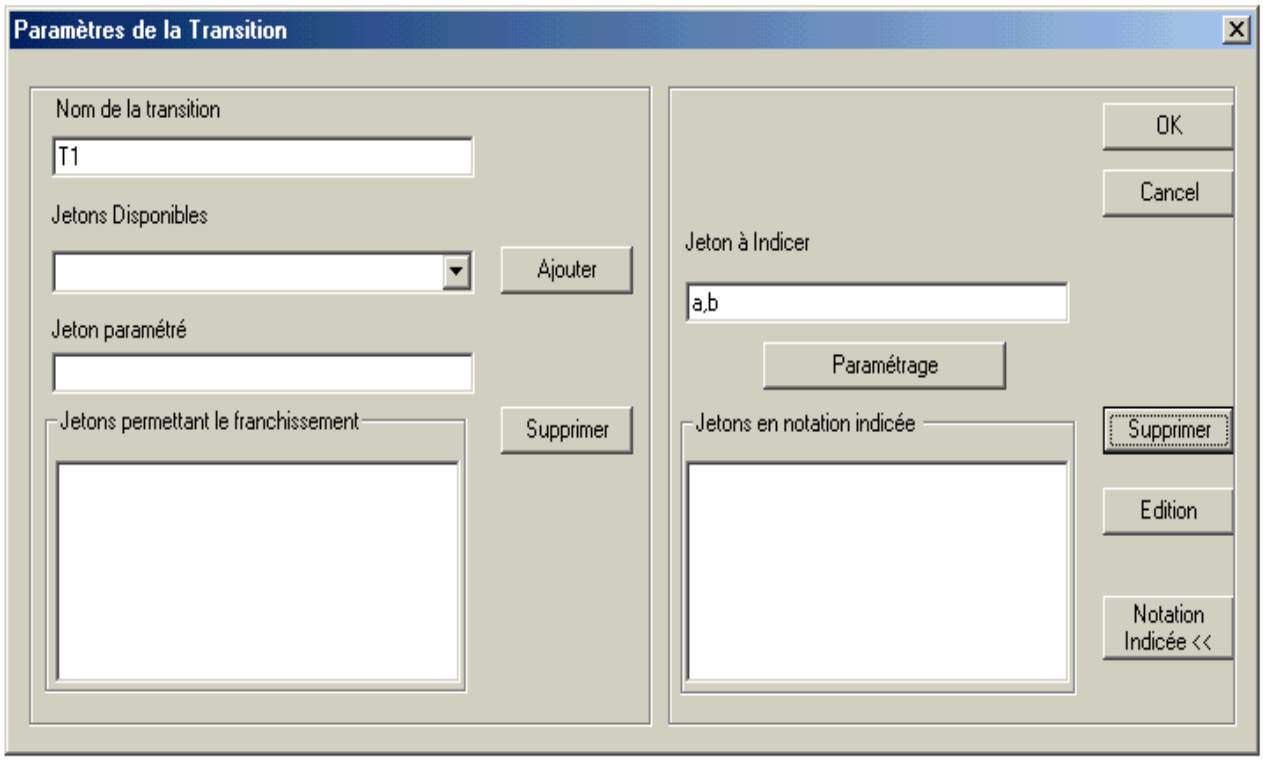

Figure 3.11 : **Création d'un ensemble de jetons indicés**

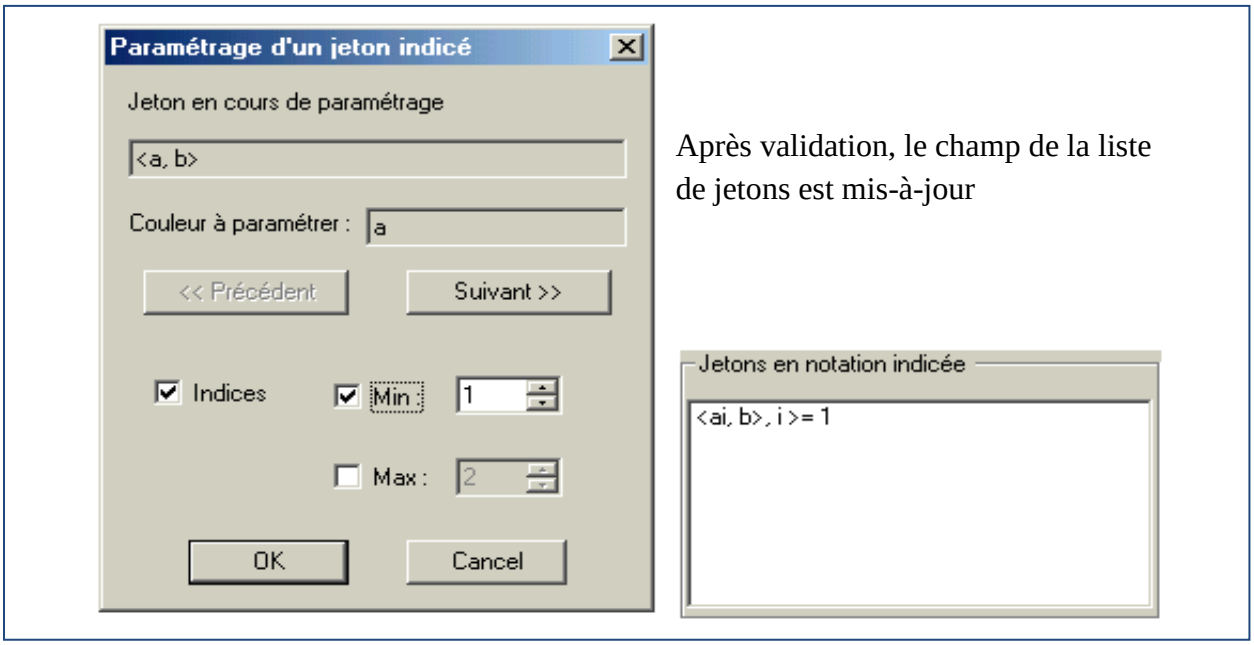

Figure 3.12 : **Paramétrage d'un jeton indicé**

3.2 SIMULATION DU MODELE DE NŒUD DE COMMUTATION

a) Modèle de nœud de commutation

Le but est de faire une simulation en mode *"Pas à Pas"* des sous-modules de la gestion de trafic définis au chapitre précédent. Le RdP de Fig 3.13 définit notre modèle de référence. Ce graphe modélise un nœud de commutation ayant les caractéristiques résumées dans le tableau 3. Afin d'illustrer l'étude, simulons le processus suivant :

4 paquets p1, p2, p3 et p4 sont émis,

- p1 est transmis avec succès.
- p2 est transmis mais avait du attendre que la ligne se libère après p1.
- p3 a été éliminé car le *buffer* est plein juste avant que p2 soit transmis.
- p4 a été rejeté car le trafic a été écrêté à cause d'un flux erratique. (p3 est donc le dernier à être régulé).

Le marquage initial serait donc : M(*Seau)* = 3, M(*Paquets émis)* = 0, M(*Paquets régulés)* = 0, M(*Buffer disponible)* = 2, M(*Paquets acceptés)* = 2, M(*Ligne libre)* = 1, M(*Paquets à transmettre)* = 0, M(*Paquets transmis)* = 0 ;

Tableau 5 : **Caractéristiques du nœud de commutation modélisé par Fig 3.13**

Caractéristiques	Eléments du graphe de Fig 3.12
Source de trafic	délivrant un paquet par franchissement de la transition *Emettre*
Régulateur de débit	*Seau à jetons* avec une capacité maximale de 4 paquets et un débit ro paquets par franchissement de la transition '*1/ro*'
Buffer	de taille 4 paquets
Discipline de service	FIFO (un seul trafic)
Ligne de sortie	Place contenant au maximum 1 jeton

b) Simulation du modèle

Une transition sensibilisée est en surbrillance rouge. Par exemple, la transition '*Emettre'* est toujours sensibilisée. La taille du buffer est de 4 paquets. On remarque donc que la relation *:*

M(*Buffer disponible)*+ M(*Paquets acceptés)*+ M(*Paquets à transmettre)*+ M(*Ligne libre) = 4* est toujours vérifiée lors de la simulation.

La séquence de franchissement est donnée par :

-pour p1 : Emettre – Réguler – Accepter – Choisir trafic

-pour p2 : Emettre – Réguler – Accepter

-pour p3 : Emettre – Réguler – Eliminer

-pour p4 : Emettre – Rejeter

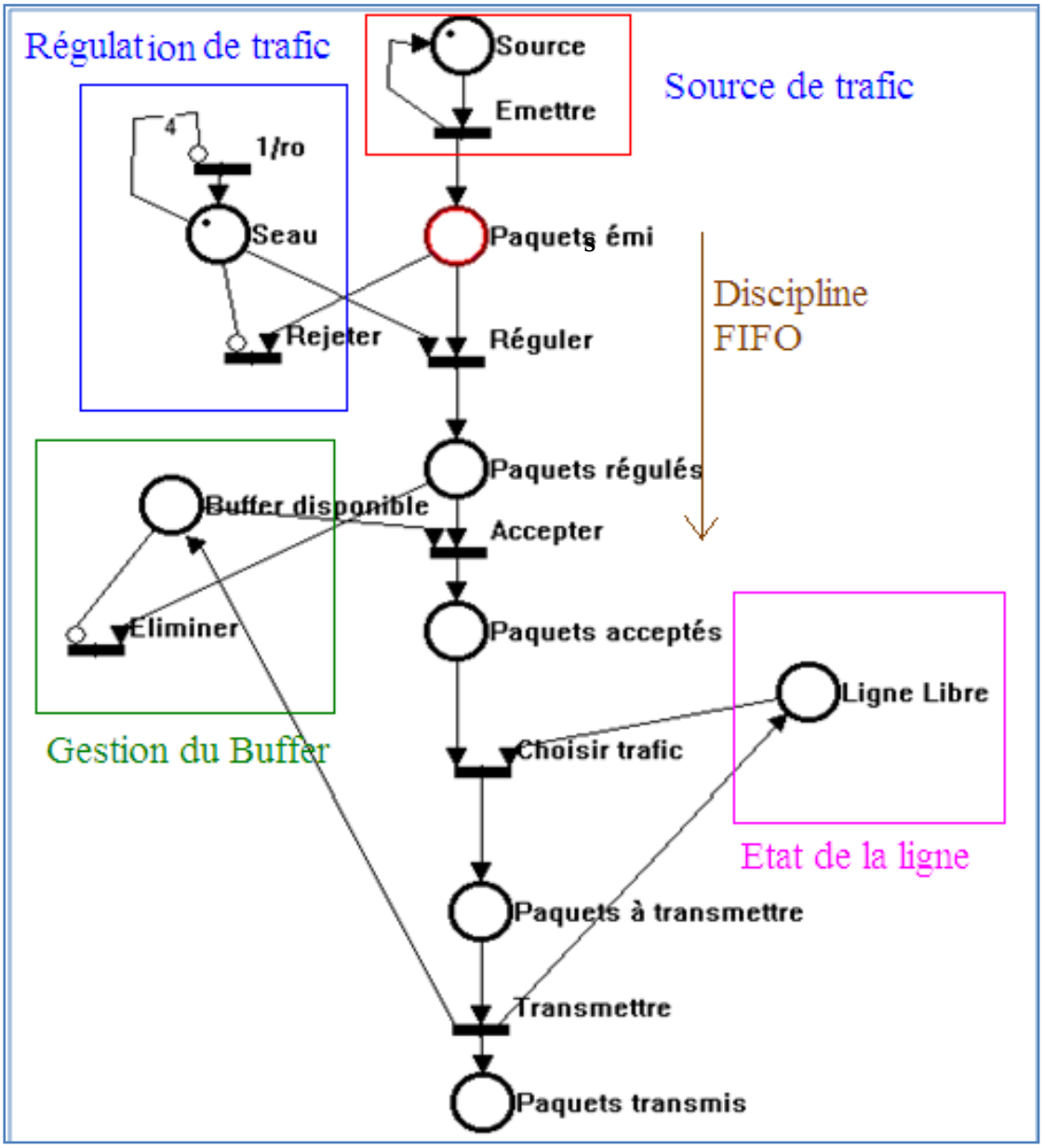

Figure 3.13: **Modèle du nœud de commutation à simuler**

c) Interprétation et évolutions possibles du logiciel

Les résultats statistiques de cette simulation sont résumés dans un fichier texte représenté par Fig 3.14. Bien que l'exemple ci-dessus paraisse simple, cette simulation nous fait comprendre l'enchaînement des modules de contrôle de flux dans un nœud de commutation par paquets et nous donne des idées sur des améliorations possibles de ces mécanismes.

```
Nombre de passages par place:

Source  4
Paquets émis    4
Seau    0
Paquets régulés 3
Buffer disponible       0
Paquets acceptés        2
Ligne Libre     0
Paquets à transmettre   1
Paquets transmis        0


        ==> Nombre moyen de passage par place : 1.55556

Nombre de franchissements par transitions:

Emettre 4
Réguler 3
1/ro    0
Rejeter 1
Eliminer        1
Accepter        2
Choisir trafic  1
Transmettre     0


        ==> Nombre moyen de franchissements par transition: 1.5
```

Figure 3.14: **Résultats statistiques de la simulation**

Les modèles qui ont été étudiés dans le chapitre précédent faisaient intervenir des RdP étendus au domaine temporel. Une évolution ultérieure du logiciel devrait supporter ces extensions. Dans ce cas, des paramètres de la qualité de service fourni par le nœud pourraient être déduits comme suit :

- Temps de traversée moyen = $\frac{\bar{M}(Paquets\ acceptés)}{\bar{N}(Accepter)}$ (Formule de Little)
- Probabilité de perte par débordement = $\frac{\bar{N}(Eliminer)}{\bar{N}(Accepter)+\bar{N}(Eliminer)}$
- Taux de rejection par écrêtage = $\frac{\bar{N}(Rejeter)}{\bar{N}(Réguler)+\bar{N}(Rejeter)}$

$\bar{M}(place) et\ \bar{N}(transition)$ désignent respectivement les valeurs moyennes temporelles du marquage de *place* et du nombre de franchissement de *transition*.

3.3 VALIDATION D'UN PROTOCOLE DE COMMUNICATION

L'utilisation des RdP n'est pas limitée à la modélisation d'un système physique à événements discrets. Grâce à son formalisme d'événement synchrone, un RdP peut décrire le mécanisme d'un protocole de communication [17]. Ce paragraphe illustre une démarche de validation d'un protocole de communication en utilisant *PetriSim*.

a) Protocole de type «Stop and Wait»

Considérons deux machines distantes qui veulent échanger des informations. La fiabilité d'acheminement des données consiste à réémettre les paquets perdus dans le réseau. Pour ce faire, un mécanisme d'acquittement des paquets envoyés est utilisé. C'est le cas par exemple du protocole de transport TCP où l'émetteur ne peut poursuivre son envoi qu'après réception des paquets qui acquittent les paquets précédemment envoyés.

b) Fonctionnement normal du protocole

La figure 3.15 illustre le fonctionnement normal du protocole. Pour pouvoir le modéliser avec un RdP, il convient d'abord de bien définir les systèmes mis en jeu : émetteur, récepteur et canal de transmission.

Figure 3.15: **Fonctionnement normal du protocole**

Ensuite, les notions d'événements qui sont susceptibles de changer les états de ses systèmes conduisent à la disposition des places et transitons du RdP (Fig 3.16). Une fois le schéma élaboré, il est nécessaire de vérifier certaines propriétés du réseau, à savoir les conflits entre franchissement, la synchronisation, le parallélisme et l'exclusion mutuelle. Pour simuler le RdP de Fig 3.16, fixons la durée de franchissement en mode automatique à 10, c'est-à-dire10 dixième de seconde soit une seconde. (Menu *Outils*, puis *Préférences*) et lançons la simulation en mode *rafale*.

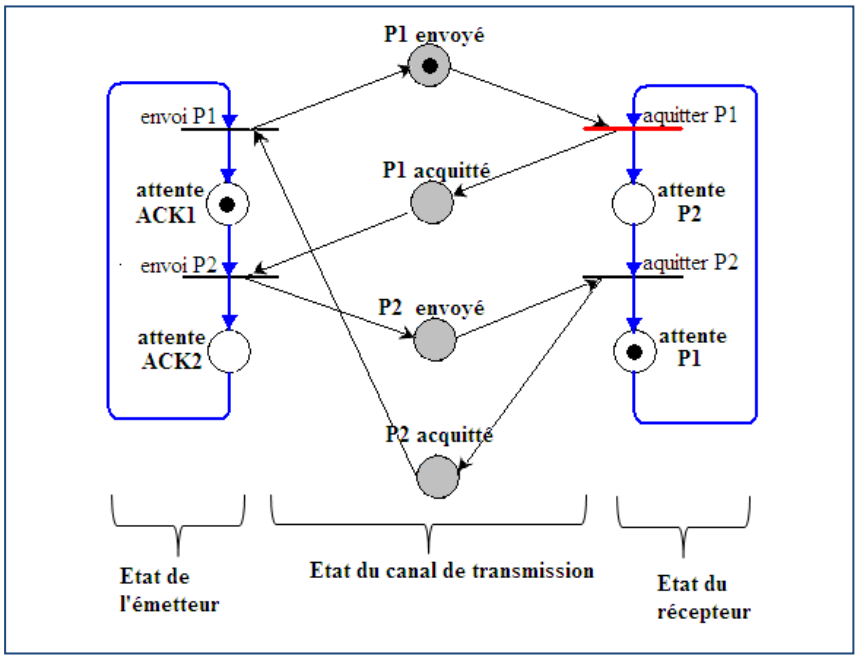

Figure 3.16: **RdP associé à Fig 3.15**

c) Fonctionnement avec perte de paquets émis

En cas de perte de paquets, un temporisateur, qui est réarmé à chaque émission se déclenche au bout d'un certain temps si aucun acquittement n'est reçu (Fig 3.17), L'émetteur réémet alors le paquet perdu dans le canal de transmission. On introduit donc une nouvelle transition permettant de simuler cette perte et une autre pour réémettre le jeton dans le canal de transmission (Fig 3.18).

Deux remarques méritent d'être mentionnées. D'abord, la réémission du paquet perdu ne doit pas changer l'état de l'émetteur. Il est toujours en attente du premier acquittement. Ce qui justifie le test de présence de jeton par l'arc à double sens reliant la place a*ttente ACK1* et la transition *Timer*.

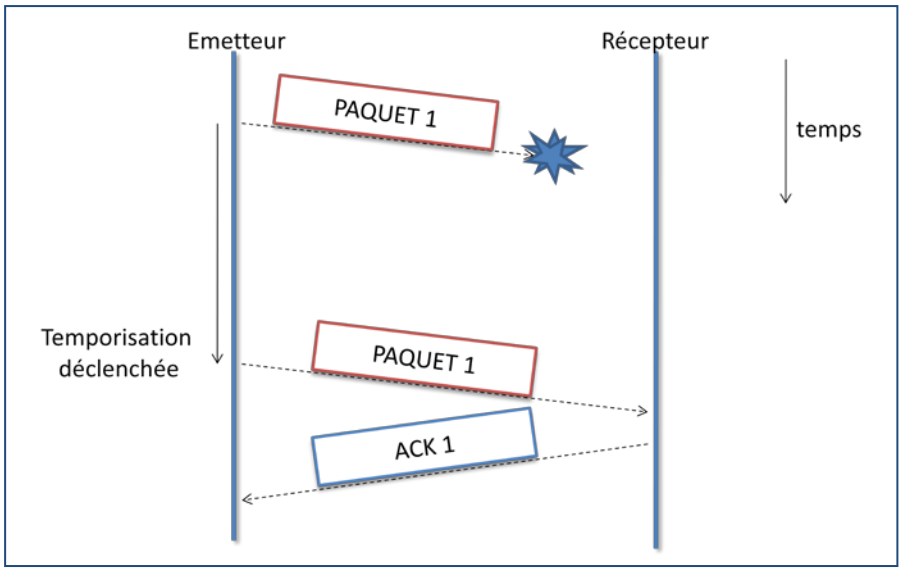

Figure 3.17: **Perte d'un paquet émis**

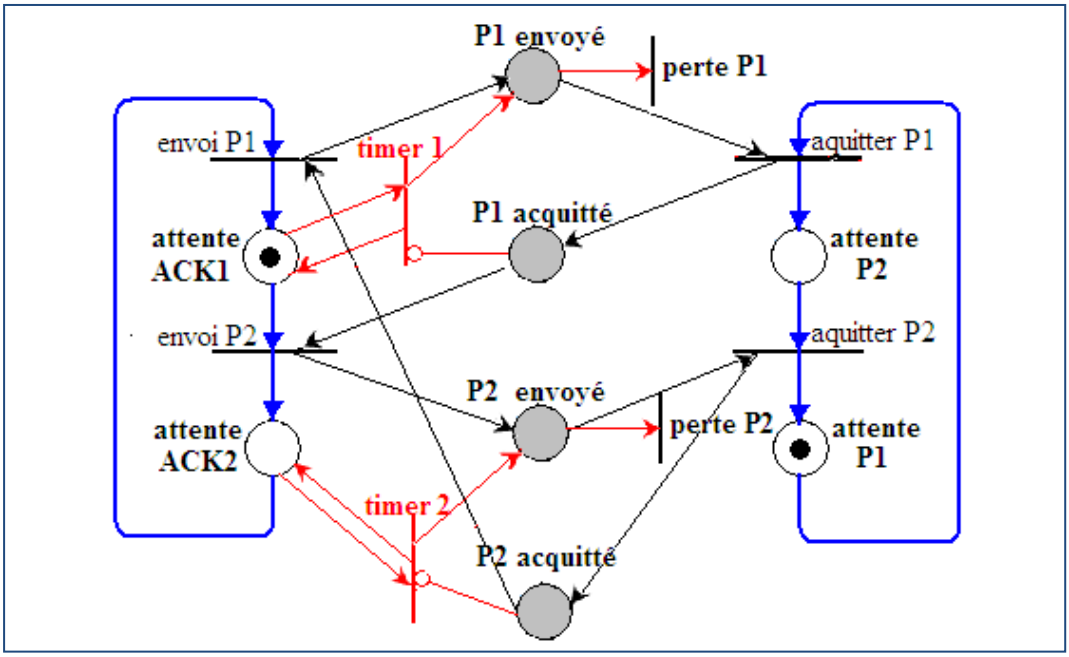

Figure 3.18: **RdP associé à Fig 3.17**

Ensuite, l'activation de la transition *Timer* doit considérer l'absence d'acquittement de P1 (*P1 acquitté* vide) mais ne tient pas compte si P1 a été réellement perdu ou tout simplement retardé. La simulation de ce dernier cas implique une augmentation de jetons dans la place *P1 envoyé*. Dans ce cas, si le récepteur recevant des paquets dupliqués ne devra transmettre qu'un seul acquittement. Une transition

identique à *Perte P1* devra être introduite du côté récepteur pour éviter des acquittements multiples identiques.

d) Fonctionnement avec perte de paquets d'acquittement

Lors d'une perte d'acquittement, la situation est plus délicate car le récepteur ne possède aucun temporisateur pour déclencher la réémission d'un acquittement. De l'autre côté, l'émetteur ne peut distinguer si c'est une perte de paquets émis ou d'acquittement. Il se contente tout simplement de retransmettre le paquet non-acquitté. (franchissement de la transition *Timer*). Cet état est illustré par Fig 3.19. Une transition *renvoi ACK1* (Fig 3.20) est donc nécessaire pour modéliser cette retransmission d'acquittement, mais qui n'est pas franchissable que si l'émetteur a effectivement retransmis le paquet supposé perdu. Les deux remarques précédentes sont valables pour la perte de paquets d'acquittement. Le même mécanisme de retransmission d'acquittement s'effectue pour le deuxième paquet. Seule la première retransmission est présentée par Fig 3.20 pour ne pas surcharger la figure.

Pour simuler le RdP de Fig 3.20 dans *PetriSim*, il est préférable d'utiliser le mode *pas-à-pas*, afin de suivre les évolutions possibles du marquage du Réseau et de ne pas se perdre dans le déroulement du protocole. Il est toutefois possible de revenir en arrière (bouton *retour arrière* de la barre d'outils) selon le nombre de retour arrière autorisé (Menu *Outils*, puis *Préférences).*

Figure 3.19: **Perte d'un paquet d'acquittement**

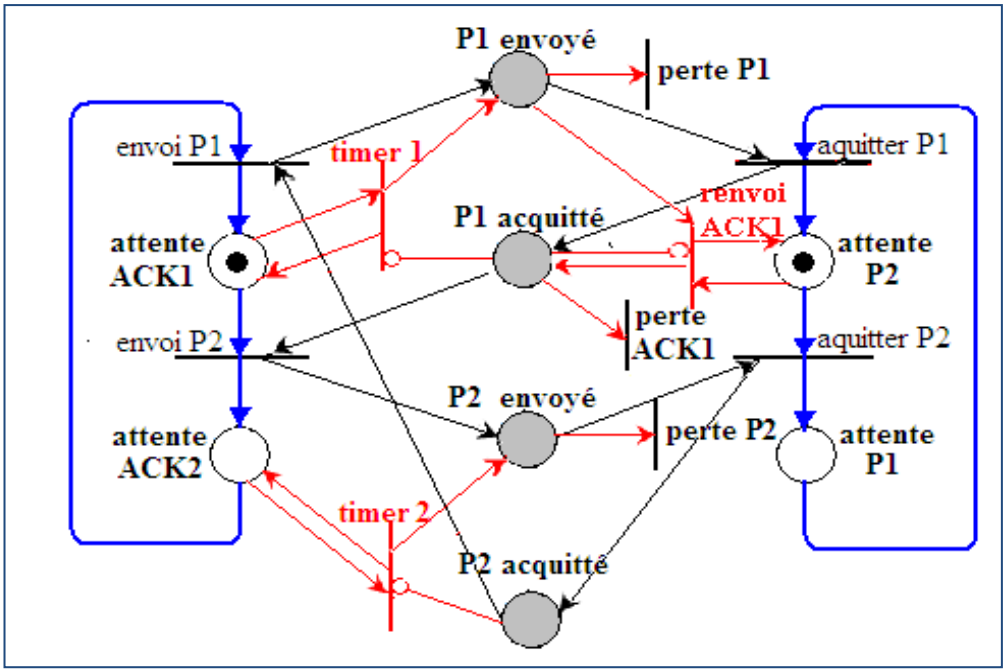

Figure 3.20: **RdP associé à Fig 3.19**

Conclusion

Ce logiciel a apporté deux contributions majeures pour cette étude. La première contribution de *PetriSim* concerne la compréhension des bases de l'outil de modélisation : RdP. Etant donné qu'il n'y a pas encore de logiciels « standards » pour la simulation d'un RdP, c'est à travers la conception de ce simulateur que nous avons pu assimiler les démarches d'étude avec des RdP.

Bien que le logiciel n'atteigne pas encore sa version finale, *PetriSim* a aussi contribué à la simulation de la plupart des modèles qui faisaient l'objet de ce mémoire de fin d'études. En effet, grâce à l'aspect dynamique de sa programmation, la taille des éléments du modèle (nombre de nœuds, jetons) n'est pas limitée, ceci permet de simuler un grand nombre de figures de RdP dans un même schéma.

CONCLUSION

L'étude d'un nœud de commutation dans un réseau *WAN* fait partie d'une des grandes lignes de l'ingénierie de réseaux actuels. C'est un sujet délicat puisque la fourniture de qualité de service au niveau d'un tel réseau dépend à la fois de la partie logicielle du nœud mais aussi de la nature du trafic du réseau. Ainsi, cette étude nécessite des outils souples et dynamiques tels que les RdP.

Le RdP présente une interprétation graphique facile pour comprendre la dynamique du comportement d'un système. Grâce à son niveau d'abstraction élevé, la modélisation des mécanismes de synchronisation, de parallélisme et de partage de ressources est utilisée pour décrire le comportement du nœud.

La méthode d'étude proposée a pour objectif de décomposer les étapes du traitement d'un flux traversant un nœud du réseau en plusieurs modules indépendants et de les simuler ensuite. Bien que muni d'un outil mathématique de base, le modèle de Petri peut être étendu à des modèles complexes faisant intervenir le temps. Il permet ainsi une approche qualitative d'étude du nœud de commutation.

Ce document s'adresse à toute personne ou compagnie désirant entamer une étude des mécanismes de contrôle de flux dans un nœud de commutation qui s'adaptent aux réseaux étendus. Plus concrètement, ce présent mémoire pourrait contribuer à l'amélioration de la fourniture de qualité de service dans les réseaux à commutation tels qu'*ATM*. Il pourrait aussi servir d'optimiser le trafic Internet dans les routeurs par l'intermédiaire des architectures de services telles que *DiffServ*.

Une étude plus fine du comportement d'un réseau serait d'associer à cette approche par RdP, une analyse qualitative sur la nature aléatoire du trafic d'un réseau. Mais lorsque, dans un trafic donné, le nombre d'entités du système à modéliser est important, la taille du RdP devient rapidement énorme, ce qui implique une croissance du coût de simulation (en termes de temps de calcul et de ressources mémoires). Pour pallier ce problème, d'autres approches telles que la formalisation de modèles multi-agents ou des méthodes heuristiques de l'intelligence artificielle sont envisageables et constituent des extensions possibles du présent travail.

ANNEXES

Annexe 1

MODELE DE REFERENCE OSI

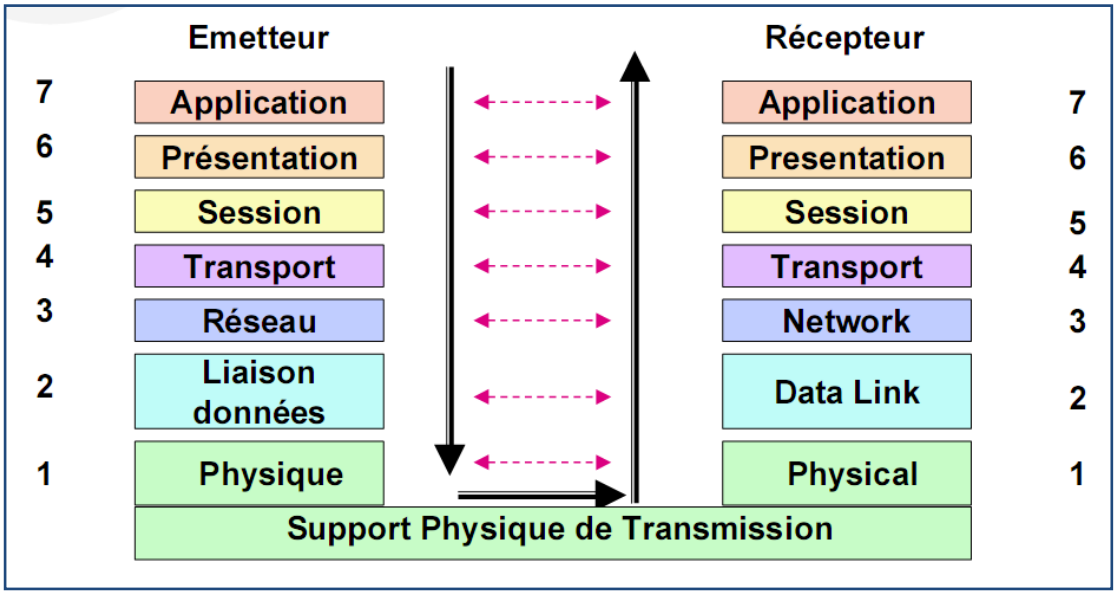

Figure A1: **Les 7 couches du modèle *OSI***

Couche Physique

Elle joue un double rôle, elle est tout d'abord chargée de l'interconnexion entre le système et le support physique. Elle est également charger d'assurer le relais des éléments binaires transmis. Elle réalise la fonction d'interconnexion entre les circuits de données. Pour ce faire, elle assure la transmission de données; la modulation ainsi que le multiplexage y sont pratiqués. Un grand nombre de technique de transmission contrôlé par des protocoles y sont utilisés.

Ex : V24, X21

Couche Liaison

La fonction de base de cette couche est de gérer les trames ainsi que d'effectuer le cas échéant, la détection et la correction d'erreur entre systèmes ouverts adjacents. Elle peut également intervenir pour coordonner le partage des connexions physiques multipoint: invitation à émettre, recevoir.

Exemple de protocole: HDLC (*High level Data Link Control Protocol*), BSC (*Binary Synchronous Communication*)

Couche Réseau

La commande de l'interconnexion c'est-à-dire le routage est réalisée par cette couche. La fonction essentielle de cette couche consiste à effectuer le relais de paquets ainsi que des circuits de données. Elle offre également un contrôle de congestion qui permet d'éviter les pertes de paquets par engorgement de chemin.

Ex : X25-3

Couche Transport

La fonction essentielle de cette couche est d'effectuer le contrôle de bout-en-bout et l'optimisation du transport de données entre système. Elle s'assure également que les messages des utilisateurs connectés parviennent correctement à leurs destinataires. La fragmentation des messages et le réassemblage des paquets font parties des services qu'elle offre. Elle offre des services supplémentaires de protection de données et contrôle de flux.

Couche Session

Elle réalise les fonctions qui sont nécessaire au support de dialogue entre processus telle l'initialisation, la synchronisation et la terminaison. La facturation des utilisations, la reprise de la connexion font aussi parties des services qu'elle offre.

Couche Présentation

Elle prend en charge les problèmes associés à la représentation des informations que les applications désirent échanger ou manipuler. Elle s'occupe aussi de la syntaxe des données échangées permettant ainsi aux entités d'application de ne se préoccuper que des aspects sémantiques des informations.

A ce titre, elle réalise la compréhension, l'organisation et l'encryptage des données à échanger. Autrement dit, cette couche assure une compréhension syntaxique en gérant le format de données et en effectuant les transformations nécessaires sur les

structures de données pour les rendre accessibles, compréhensibles pour des équipements hétérogènes.

Couche application

Elle recouvre les programmes d'application avec leurs conventions d'échange et de coopération c'est-à-dire la compréhension et l'exécution des commandes liées aux applications. De plus, elle établit les règles d'échange entre opérateur entre périphérique - programme tout cela assorti de contrôle d'accès.

Dans la pratique, toutes les couches ne sont pas nécessairement utilisées. Par exemple, une simple connexion entre deux ordinateurs n'a pas besoin de la couche Réseau. Par contre, un réseau téléinformatique hétérogène à commutation par paquet distribué géographiquement nécessite toutes les 7 couches.

Les couches 1-2-3 sont appelées « couches basses » ou « de bas niveau » et sont orientées vers l'acheminement des données. Elles fournissent le service de transport. Les couches supérieures 5-6-7 ou « couches de haut niveau » définissent les caractéristiques particulières des moyens de communication et de connexion. Les couches hautes fournissent les services d'accès.

Annexe 2

ELEMENTS DE LA THEORIE DE LA FILE D'ATTENTE

La théorie des files d'attente est un grand domaine d'application des processus aléatoires. Depuis quelques décennies, elle est très utilisée dans le domaine de la télécommunication pour évaluer le nombre de communicants par unité de temps dans un réseau. C'est aussi un outil mathématique qui décrit bien le processus d'arrivée et de service dans un nœud de commutation de paquets.

A2.1 Loi de Poisson

La *loi de Poisson* correspond assez bien à la réalité dans un système informatique comportant un grand nombre d'usagers. En effet, cette loi est basée sur les trois hypothèses suivantes:

Hypothèse 1 : Les arrivées des paquets sont indépendantes les unes des autres. Cette hypothèse est d'autant plus vérifiée qu'ils proviennent d'un grand nombre de sources aléatoires différentes.

Hypothèse 2 : le processus est stationnaire, c'est-à-dire que ses caractéristiques ne varient pas en fonction de l'origine du temps d'observation.

Hypothèse 3 : pour un intervalle de temps très faible Δt, la probabilité $\lambda \Delta t$ d'arrivée d'un seul paquet est plus grande que la probabilité d'arrivée de plusieurs paquets.

La loi de Poisson de paramètre λ définit la probabilité $P_n(T)$ d'observer n arrivées de clients pendant l'intervalle de temps T. Elle est exprimée par :

$$P_n(T) = \frac{(\lambda T)^n e^{-\lambda T}}{n!} \qquad \text{(A2 .1)}$$

A partir de cette expression, on peut déduire le *nombre moyen d'arrivées de clients pendant la durée T* :

$$E_T(n) = \sum_{n=0}^{+\infty} nP_n(T) = \lambda T \qquad \text{(A2.2)}$$

A2.2 Loi exponentielle

Une autre façon de voir le processus d'arrivée est de considérer non pas le nombre de clients n mais l'intervalle de temps τ séparant deux arrivées successives. Cette variable aléatoire (d'interarrivée) suit la *loi exponentielle f* définie par:

$$f(\tau) = \lambda e^{-\lambda\tau} \qquad \text{(A2.3)}$$

La *durée moyenne entre deux arrivées successives* est :

$$E(\tau) = \int_0^{+\infty} \tau\, f(\tau)\, d\tau = \frac{1}{\lambda} \qquad \text{(A2.4)}$$

A2.3 Files d'attente M/M/1 et M/M/1/K

Les tableaux A2.1 et A2.2 suivants résument les caractéristiques des files d'attente M/M/1 et M/M/1/K.

Tableau A2.1: **Comparaison des caractéristiques des files d'attente M/M/1 et M/M/1/K**

Caractéristiques	M/M/1	M/M/1/K
Probabilité d'état stationnaire	$p_n = p_0 \left(\frac{\lambda}{\mu C}\right)^n$	
Taux d'activité du serveur $\theta = \sum_{n=1} p_n$	$\theta = \frac{\lambda}{\mu C} = \rho$	$\theta = \frac{1-\rho^K}{1-\rho^{K+1}}\rho$
Débit du système $D = \mu C \sum_{n=1} p_n$	λ	$\frac{1-\rho^K}{1-\rho^{K+1}}\lambda$

Tableau A2.2: **Comparaison des caractéristiques des files d'attente M/M/1 et M/M/1/K (suite)**

Caractéristiques	M/M/1	M/M/1/K
Nombre moyen de paquets $N = \sum_{n=0} {}^{n}p_n$	$\frac{\rho}{1-\rho}$	$\frac{\rho}{1-\rho}\frac{1-(k+1)\rho^k - k\,\rho^{k+1}}{1-\rho^{k+1}}$
Temps de traversée moyen	$T_R = \frac{N}{D}$ *(Formule de Little)*	
Taux de blocage p_b	0	$\frac{(1-\rho)\rho^K}{1-\rho^{K+1}}$

Avec :

λ : taux d'arrivée de paquets dans le nœud

K : capacité de lafile (en nombre de paquets)

μ : taux de service, ie. nombre de paquets que le nœud peut commuter par unité de temps

C : capacité de la ligne de sortie (en bit/s)

$\rho = \frac{\lambda}{\mu C}$: Intensité du trafic

Annexe 3

ARCHITECTURES DE SERVICES

La qualité de service (QoS, Quality of Service) constitue un axe de recherche majeur dans les réseaux. Deux approches ont été étudiées : la réservation de ressource (réseaux à état) et la prioritisation de flux (réseaux sans état). ATM et Frame Relay appartiennent à la première catégorie, tandis que dans les réseaux IP, aucun état n'est maintenu dans le réseau (réseau datagramme ou best effort). Dans ce contexte seule l'approche de prioritisation semble possible. Le champ TOS (Type Of Service) de IP a été redéfini par la RFC 1812 (Request For Comments). Deux approches de la QoS sont actuellement définies : Integrated Services (ou IntServ, RFC 1633) et Differentiated Services (ou DiffServ, RFC 2474).

A3.1 Architecture *IntServ*

Le modèle *IntServ* est un service orienté flot, c'est-à-dire que chaque flot peut faire sa demande spécifique de qualité de service. *IntServ* définit 3 types de services invoqués via le protocole RSVP (*Resource reSerVation Protocol*), représentés par Tab A3.1.

L'approche *IntServ* peut être considérée comme une adaptation dans le mode datagrammes des techniques orientées connexion. Chaque application formule une demande de réservation de ressources de bout en bout. Mais le modèle *IntServ* et *RSVP*, compte tenu de sa complexité, n'a connu aucun succès.

A3.2 Architecture *DiffServ*

Le rôle de *DiffServ* est d'agréger les flots en quelques classes offrant des services spécifiques. La QoS est assurée par des traitements effectués dans les routeurs spécifiés par un indicateur situé dans le paquet. Les routeurs sont commandés par un champ DSCP (*Differentiaied Service Control Point*) situé dans le paquet IP. L'approche *DiffServ* est donc plus conforme à l'approche IP. Outre le service *Best-Effort* d'*IntServ*, deux services, résumés dans Tab A3.2, ont été définis.

Tableau A3.1: **Types de services dans *IntServ***

Type de service	Rôle
Service garanti (*Guaranteed Services*)	• affecte une borne supérieure au délai d'acheminement. • deux parties : une spécification de la QoS (*FlowSpec)* et une spécification de paquets (*FilterSpec).* • certains paquets du flot peuvent avoir une QoS et pas les autres.
service *Best Effort*	• aucune garantie ni sur les pertes, nisur le temps de transport.
Un service contrôlé (*Controlled Load*)	• qualité à peu près égale à celle du même réseau mais avec des liaisons peu chargées. • Les temps de passage des flots CL similaires à ceux des clients d'une classe *Best-Effort* dans un réseau très peu chargé.

Tableau A3.2: **Types de services dans *DiffServ***

Type de service	Rôle
Assured Forwarding ou *Olympic Service* (RFC 2597)	• négociation d'un agrément correspondant à un profil déterminé par un taux moyen. • détruction d'un flux en priorité si un risque de congestion existe. • politique d'écartement (RED, *Random Early Detection*) en fonction de l'état du réseau (*Low Drop, Medium Drop et High Drop*). À chaque classe sont affectées une priorité et une garantie de bande passante.
service *Best Effort*	• identique *à IntServ.*
Expedited Forwarding	• trafic sensible au délai et à la gigue. • priorité forte mais doit être cependant être contrôlée.

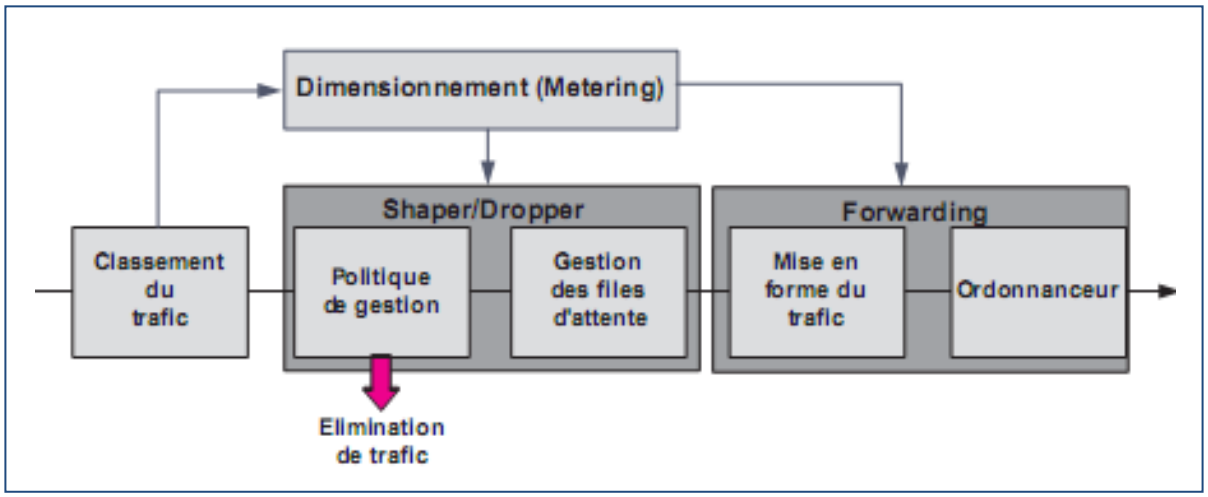

Figure A3: **Mécanismes de QoS sous DiffServ**

La classification et la vérification des flux sont effectuées à la périphérie du réseau (*Classifier*). Les propriétés des flux sont ensuite analysées (*Metering* ou dimensionnement) en fonction d'un contrat de service préétabli. Les datagrammes sont alors marqués (*Marker*) par positionnement du champ DS Field. Le trafic différentié ou colorisé (*Multiflow Classifier*) est analysé, certains paquets peuvent être retardés (mise en forme du trafic ou *shaper*) voire éliminés pour prévenir un éventuel état de congestion (*Dropper*). Les paquets sont ensuite affectés à une file d'attente spécifique avant d'être transmis sur le réseau (*Forwarding*). La figure A 2 illustre les mécanismes que nous venons de décrire.

L'architecture *DiffServ* est bien adaptée aux grands réseaux. En effet, les classes de services étant attribuées en périphérie du réseau *DiffServ*, elles ne génèrent ni trafic de gestion, ni surcharge CPU. Cependant, si *DiffServ* permet de hiérarchiser les flux, il ne dispose pas de mécanisme d'information d'état du réseau. De ce fait, les routeurs de bordures ne sont pas en mesure d'anticiper et ni de réagir à un état de précongestion ou de congestion.

Annexe 4

LES 9 DIAGRAMMES UML

Les diagrammes d'objets (ou d'instance)

Ils représentent les objets et leurs relations sans représenter les envois de messages (représentation statistique). Ils permettent ainsi la compréhension générale du système et facilitent la compréhension des structures complexes (récursives).

Les diagrammes de collaboration

Ils correspondent à une extension des diagrammes d'objets. Ce sont des représentations de la structure spatiale statique qui permet la mise en relation d'un groupe d'objets. Ils représentent aussi des interactions entre objets.

Les diagrammes de classes

Ils représentent la structure abstraite des liens potentiels d'un objet vers d'autres objets.

Les diagrammes de cas d'utilisation

Ils décrivent sous forme d'actions et de réactions le comportement d'un système du point de vue de l'utilisateur. Un utilisateur ou acteur est représenté par un petit personnage, un cas d'utilisation ou fonctionnalité de système par un ovale, les flèches issues d'un personnage pointent vers les cas d'utilisation.

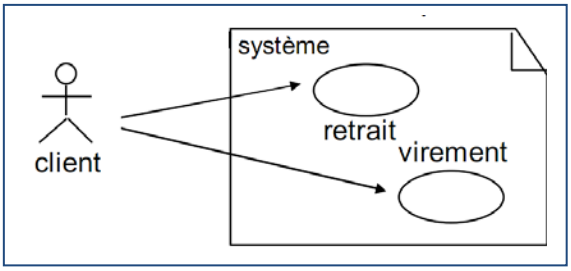

Figure A4.1: **Exemples d'acteur et de cas d'utilisation**

Il y a quatre catégories d'acteurs :

- *Acteurs principaux* : qui utilisent les fonctions principales du système
- *Acteurs secondaires* : qui font les tâches administratives ou de maintenance
- *Matériels externes* : qui sont des dispositifs matériels incontournables utilisés
- *Autres systèmes* : qui sont les systèmes avec lesquels le système doit interagir

Il y a trois types de relations entre acteurs et cas d'utilisation :

- *Relation de communication :* Déclenche = déclenchement d'un cas d'utilisation par un acteur
- *Relation d'utilisation :* Utilise = un cas d'utilisation comprend le comportement d'un autre cas d'utilisation
- *Relation d'extension :* Etend = le cas d'utilisation source étend et enrichi le comportement

Les diagrammes de séquences

Un diagramme de séquence montre des interactions entre objets selon un point de vue temporel. La représentation se concentre sur l'expression des interactions.

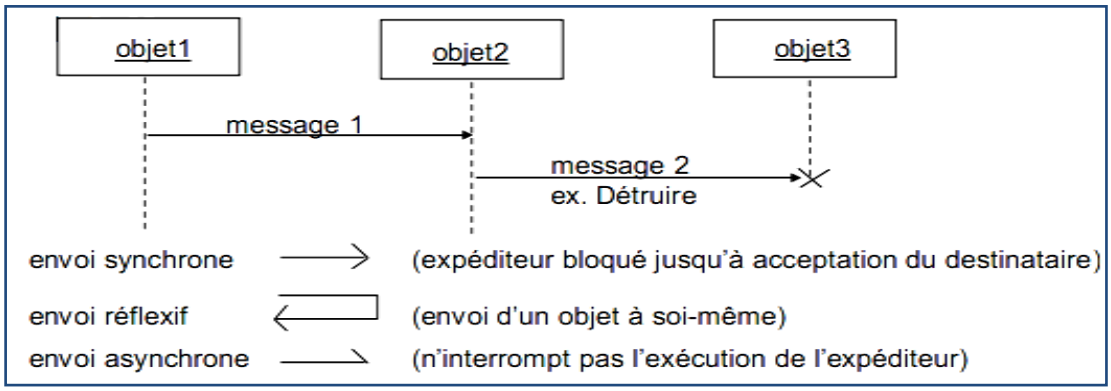

Figure A4.2: **Exemples d'interaction entre objet**

Les diagrammes d'états-transitions

Un diagramme d'états-transitions est un automate d'état fini déterministe associé à une classe. Pour l'abstraction des comportements possibles, chaque objet suit le comportement décrit dans l'automate associé à sa classe, son état caractérise ses conditions dynamiques. On associe un tel automate à toute classe qui présente un comportement réactif marqué.

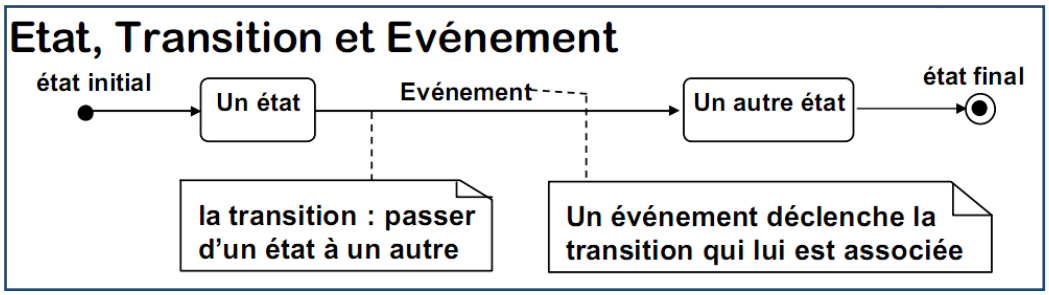

Figure A4.3: **Exemple d'états-transitions**

Les diagrammes d'activités

Ce sont des variantes des diagrammes d'états-transitions destinées à représenter le comportement interne d'une méthode. Ils représentent :

- Le déroulement d'étapes regroupées séquentiellement
- Les synchronisations entre flots de contrôles.

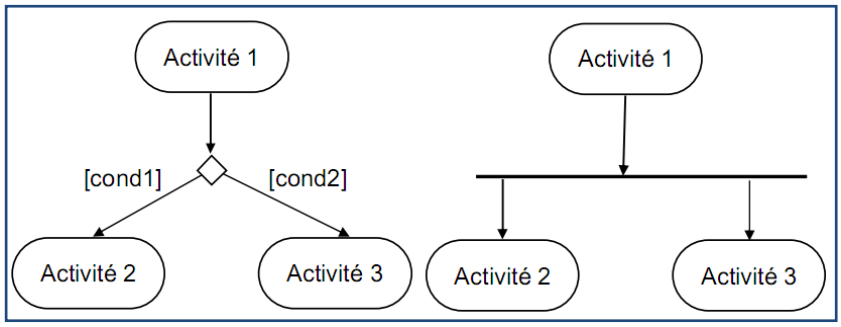

Figure A4.4: **Exemple de déroulement d'étapes d'activités**

Les diagrammes de composants

Ils décrivent des relations physiques et leurs relations dans l'environnement de réalisation. Un composant est un élément informatique qui entre dans la fabrication d'une application. On distingue trois types de composants : Spécification ou interface, corps, spécification générique.

Les diagrammes de déploiement

Ils montrent la disposition physique des matériels qui composent le système ainsi que la répartition des exécutables sur ces matériels. Un nœud est une ressource matérielle physique (dispositif, processeur, mémoire). Les liens entres nœuds sont des lignes qui symbolisent un support de communication à priori bidirectionnel.

REFERENCES

[1] « *Architecture d'une boîte à outils d'algorithmes d'ingénierie de trafic et application au réseau GÉANT* », *ftp://ftp.run.montefiore.ulg.ac.be/pub/RUN-PP05-02.pdf*

[2] Randriamampianina Sitraka Sedera Nandrasana, « *Etude de file d'attente dans un réseau WAN par la méthode analytique* », Mémoire de fin d'études-Département Electronique-EA, ESPA-Université d'Antananarivo-Promotion 2009

[3] G. Pujolle, « *Les Réseaux*», Edition Eyrolles 2003

[4] Cours « *E551 Téléinformatique* », Département Electronique-IA 5ème année, ESPA-Université d'Antananarivo 2008-2009

[5] Claude Servin, « *RESEAUX ET TELECOMS* », DUNOD 2003

[6] François Laissus, « *Cours d'introduction à TCP/IP* » *ftp//ftp.laissus.fr/pub/cours/cours.pdf*, Avril 2003

[7] Stéphane Lohier, « *Transmissions et Réseaux* », DUNOD 2003

[8] Henri Nussbaumer, « *Téléinformatique 2 : Conception des réseaux-Réseau-Transport* », Collection Informatique, Lausanne 1987

[9] *http://www.lohri.net/PDFTelecom/Simulation and Performance Analysis of Distributed Systems.pdf*

[10] C.A. Petri, « *Kommunikation mit Automaten* ». Schriften des Rheinisch-Westfälischen Institutes für Instrumentel le Mathematik an der Universität Bonn Nr. 2, 1962, extrait traduit en Français sur *http://www.rap.prd.fr/pdf/ CAPetriAndPetriNets.pdf*

[11] R. David and H. Alla, «*Du Grafcet aux Réseaux de Petri* », Hermes 1992

[12] Robert Valette, « *Les Réseaux de Petri* », LAAS-CNRS Toulouse, *http://www.laas.fr/~robert*, Septembre 2002

[13] G. Scorletti et G. Binet, « *Réseaux de Petri* ». Université de CAEN/BASSE-NORMANDE. *http://www.greyC.ensicaen.fr/EquipeAuto/Gerard S.*

[14] Guillame Lehman, *cours « Quality of Service »*, h*ttp://www..fr/Cours_QoS.ppt*, 2005

[15] Slim Abdellatif et Guy Juanole, « *A Petri Nets framework for QoS analysis in packet switched networks* », LAAS-CNRS
Toulouse, *http://www.control.auc.dk/~henrik/OnLineDocs/guyslim2001.pdf*

[16] Cours « *E550 Génie Logiciel II* », Département Electronique-IA 5ème année, ESPA-Université d'Antananarivo 2008-2009

Abstract : A crucial issue in the design of software components in a wide network is the ability to support various types of network flow and to guarantee the Quality of Service (QoS) requirements to the users. In particular, we need rigorous but easy-to-use techniques for predicting and analyzing the performance of computer networks. To this purpose, formal modeling techniques can be used for studying network nodes. Petri Net (PN) is a powerful technique to this aim, because of its analytical capability and its possible evolution in time. In this paper, we present PN models (including PN based-model, colored PN, timed PN and stochastic PN) for modeling the characteristics of a switching node in a Wide Area Network (*WAN*). By using PN in this case study, we demonstrated the possibility of evaluating the performance of any other network nodes as in ATM network or in IP network such as *Internet*.

Keywords: Petri Nets, switching node, WAN, traffic management and control, QoS, performance evaluation.

Auteur : RAJAONARISON Tolotriniaina Mirado

Titre : « **UTILISATION DU RESEAU DE PETRI POUR L'ETUDE D'UN NOEUD DE COMMUTATION DANS UN RESEAU *WAN*** »

Nombre de pages : 68

Nombre de figures : 59

Nombre de tableaux : 9

Résumé : Une issue cruciale pour la conception de composants logiciels dans un réseau étendu est l'aptitude à supporter des types de flux de réseau variés et à garantir les besoins de Qualité de Service (QdS) pour les utilisateurs. En particulier, cela nécessite des techniques rigoureuses mais faciles à utiliser pour prédire et analyser la performance des réseaux informatiques. Pour ce besoin, des techniques de modélisation formelle peuvent être utilisées pour l'étude des nœuds de réseau. Le Réseau de Petri (RdP) est une technique puissante à cet effet, grâce à sa capacité analytique et à sa possibilité d'évolution dans le temps. Dans ce mémoire, nous présentons des modèles de Réseaux de Petri (incluant des modèles de base, RdP colorés, RdP temporels et des RdP stochastiques) pour la modélisation des caractéristiques d'un nœud de commutation dans un réseau *WAN (Wide Area Network).* En utilisant des RdP dans cette étude de cas, nous avons démontré la possibilité d'évaluer la performance des autres nœuds de réseau comme ceux d'ATM et des réseaux IP tels qu'Internet.

Mots-clés : Réseaux de Petri, nœud de commutation, WAN, gestion et contrôle de trafic, QdS, évaluation de performance.

Rapporteur : Mrs. RABEHERIMANANA Lyliane Irène

Email : miradortm@yahoo.fr