Evaluation de délais dans les systèmes de communication temps-réel en utilisant des files d'attente virtuelles

Boussad Addad, Saïd Amari

▶ **To cite this version:**

Boussad Addad, Saïd Amari. Evaluation de délais dans les systèmes de communication temps-réel en utilisant des files d'attente virtuelles. Modelisation des systèmes reactifs MSR 2009, Nov 2009, Nantes, France. pp.985-999. hal-00434314

HAL Id: hal-00434314
https://hal.science/hal-00434314

Submitted on 22 Nov 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Evaluation de délais dans les systèmes de communication temps-réel en utilisant des files d'attente virtuelles

Boussad Addad — Saïd Amari

Laboratoire Universitaire de Recherche en Production Automatisée (LURPA), ENS de Cachan
61 avenue du Président Wilson, F-94235 Cachan cedex
{prenom.nom}@lurpa.ens-cachan.fr

RÉSUMÉ. *Dans cet article, une méthode à base de files d'attente virtuelles (MFAV) pour l'évaluation de délais dans les réseaux de communication temps-réel, est développée. Une méthodologie générale (modélisation et évaluation) est décrite pour l'estimation du délai de bout en bout qu'un message subit depuis sa source jusqu'à sa destination. Le pire délai étant particulièrement important pour prédire l'acheminement d'un message, nous montrons comment l'évaluer formellement avec une précision bien déterminée. Par la suite, nous nous intéressons au cas des réseaux Ethernet commutés en considérant différentes architectures de commutation. Sur un cas d'étude, nous évaluons le pire délai de bout en bout et comparons les résultats de MFAV à des mesures réelles et du calcul réseau.*

ABSTRACT. *In this paper, a virtual queuing based method to evaluate the end to end delays in real time communication systems is presented. A general methodology is described to achieve an assessment of the delay of a message delivery from its source to its destination. Since the worst end-to-end delay is particularly important for the guarantee of the timing delivery, we show how to use the proposed method to assess it formally with a well known accuracy. Thereafter, we apply it to the evaluation of the worst end to end delays in switched Ethernet networked systems. On a case study, we evaluate the end to end delays and compare the results of the proposed method to real measures and the results obtained using the widely accepted network calculus.*

MOTS-CLÉS : *Files d'attente virtuelles (FAV), Evaluation de performances, Délai de bout en bout, réseaux de communication temps réel.*

KEYWORDS: *Virtual queues, performance evaluation, End to end delay, real time communication networks.*

1. Introduction

Les réseaux sont de plus en plus utilisés pour les échanges de données dans les communications industrielles et techniques. Ils présentent en effet beaucoup de d'avantages en souplesse, efficacité et coûts. Toutefois, un message traversant un réseau est confronté au problème de partage de ressources et donc de délai d'attente. La problématique qui en découle est bien sûr la maîtrise de ces délais et la capacité à prédire si un message peut être délivré ou non avant un certain délai. Actuellement, de nombreuses méthodes existent pour l'évaluation de ces délais. Parmi celles-ci, on trouve les méthodes probabilistes comme la théorie des files d'attentes stochastiques (John *et al.*, 2005), chaînes de Markov (Gunter *et al.*, 2006), calcul réseau stochastique (Scharbarg *et al.*, 2009)... Ces méthodes sont cependant mal adaptées aux systèmes temps réel où une évaluation stricte et non probabiliste doit être assurée. Le calcul réseau déterministe (NC) est probablement la méthode la plus répandue pour assurer cet objectif (Cruz, 1991a, 1991b), (Le Boudec, 2004), (Georges *et al.*, 2005). Néanmoins, cette méthode est aussi connue pour son pessimisme dû à la surestimation des flux en utilisant le modèle taux-rafale (ρ, b) et ceci n'est pas sans conséquences. En effet, l'utilisation d'estimations pessimistes induit une sous exploitation des ressources de certains réseaux (Fan *et al.*, 2008) vu que le mécanisme de contrôle d'admission est basé sur les délais estimés. Dans le domaine de commande digitale en réseau, il a été montré dans (Addad *et al.*, 2008a, 2008b) que le pessimisme dans l'évaluation induit aussi des effets néfastes sur la qualité de commande (instabilité et dégradation de la dynamique de commande).

Contrairement aux systèmes ouverts comme Internet, dans les systèmes temps-réel dans un environnement industriel, les flux sont majoritairement périodiques avec seulement quelques échanges asynchrones (e.g. signaux d'alarmes). Cependant, cette caractéristique n'est jusqu'à lors pas assez exploitée et ces systèmes sont souvent étudiés comme dans le cas général. Parmi les rares travaux qui ont pris cela en compte, nous trouvons le travail très récent présenté dans (Fan *et al.*, 2008). Une méthode de pire cas y est en effet développée pour évaluer ces systèmes particuliers. Elle procure des résultats moins pessimistes que ceux du NC dans certain cas et mitigés dans d'autres. L'objectif de ce papier est de développer une nouvelle approche qui soit formelle (évaluation stricte) et moins pessimiste dans le cas général. Cet article est organisé comme suit ; dans la section 2, nous proposons des éléments simples pour modéliser les systèmes de communication. Dans la section 3, nous décrivons le modèle des flux considérés et dans la section 4, expliquons le principe de la méthode d'évaluation à base de files d'attente virtuelles (MFAV). Ensuite dans la section 5, nous adaptons MFAV pour l'étude des réseaux Ethernet commutés. La section 6 est dédiée à l'étude d'un système réel pour la vérification de la validité de MFAV. Une confrontation des résultats de MFAV à ceux du NC ainsi qu'à des mesures réelles y est faite. Enfin, une conclusion et quelques perspectives sont données dans la section 7.

2. Blocs de base pour modéliser les systèmes de communication

Un système de communication peut être modélisé en utilisant les types d'éléments de base suivants: nœuds source, nœuds destination, files d'attente et ressources partagées (figure 1). Un nœud source est l'entrée par laquelle des messages entrent dans le système (e.g. un capteur dans un system d'automatisation). Un nœud destination est la sortie par laquelle les messages sortent du système (e.g. un actionneur). Une file d'attente est un buffer (il est supposé de taille assez large pour éviter tout débordement) dans lequel des messages attendent la fin de traitement d'autres messages pour être traités à leur tour. Une ressource partagée est un élément de service qui peut traiter q messages simultanément (de capacité q) et provenant d'une ou plusieurs files d'attente. La politique PAPS est considérée dans toutes les ressources i.e. les premiers arrivés sont les premiers servis.

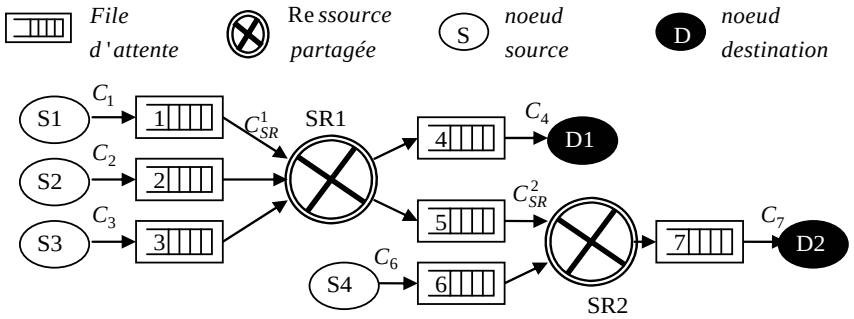

Figure 1. *Exemple de modèle utilisant les blocs de base*

Comme on peut constater sur la figure 1, tous les éléments de chaque type de bloc sont numérotés différemment (complètement identifiés par leur numéro). La $k^{ème}$ file d'attente liée à un nœud source reçoit des données à la vitesse C_k (capacité de transmission en bit/s). La $k^{ème}$ file d'attente liée à un nœud destination expédie des données à la vitesse C_k. Quand un buffer est lié à la $k^{ème}$ ressource, il lui sert ou en reçoit des données à la vitesse C_{SR}^k. Evidemment, tout message généré suit un certain trajet pour rejoindre sa destination (sous l'hypothèse d'absence de pertes). Ce trajet est déterminé à l'aide d'une fonction de routage statique notée $PATH$ qui à chaque couple *(source* S_i, *destination* D_j*)* associe un ensemble d'éléments tel que : $PATH(S_i, D_j) = \{B_{k1}, B_{k2} \cdots B_{kn}\}$ *où* B_{ki} est le $ki^{ème}$ buffer traversé par le message généré depuis la source S_i et allant vers la destination D_j. Nous ajoutons la fonction $LINK$ pour déterminer aussi les ressources partagées traversées tel que : $LINK(B_{ki}B_{kj}) = SR_l$ *où* SR_l est la ressource liant les deux buffer B_{ki} et B_{kj}. Ainsi, tous les blocs traversés sont totalement définis avec ces deux fonctions. Par exemple, sur la figure 1 nous avons : $PATH(S_3, D_2) = \{B_3, B_5, B_7\}$, $LINK(B_3, B_5) = SR_1$ et $LINK(B_5, B_7) = SR_2$. Au final, un message depuis S_3 vers D_2 traverse les éléments : B_3, B_5, B_7, SR_1 et SR_2.

3. Modélisation et analyse du trafic généré dans le réseau

Comme déclaré en introduction, nous focalisons notre étude sur les systèmes avec des générateurs de messages périodiques. Tout nœud source génère une rafale de messages avec une vitesse C_k et une période T_k. Chaque rafale est un ensemble de messages (paquets de bits) de longueur quelconque générés les uns après les autres comme montré sur la figure 2 :

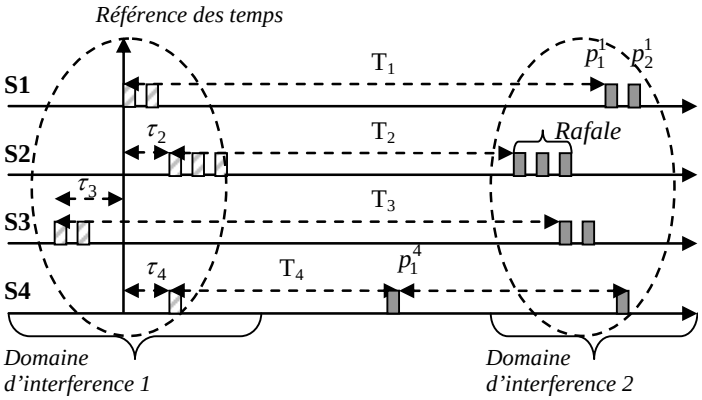

Figure 2. *Génération périodique des rafales*

Un scénario de génération de rafales dans le système de l'exemple de la figure 1, correspondant à la figure 2 est :

– S_1 envoie une rafale de deux paquets notés p_1^1 et p_2^1 avec une période T_1 pour respectivement les destinations D_1 et D_2.

– S_2 envoie p_1^2, p_2^2 et p_3^2 avec une période T_2 pour D_1, D_2 et D_1.

– S_3 envoie p_1^3 et p_2^3 avec une période T_3 pour respectivement D_2 et D_1.

– S_4 envoie un seul paquet p_1^4 avec une période T_4 pour D_2.

Comme nous pouvons noter sur la figure 2, les nœuds source ne commencent pas l'envoi de leurs rafales en même temps. Ils sont en effet pas synchronisés et génèrent leurs messages indépendamment les uns des autres. Si nous prenons comme référence des temps la date de début d'envoi à partir du nœud S_1, le $i^{ème}$ $(i \neq 1)$ nœud source commence l'envoi avec un décalage noté τ_i. Trivialement, différents décalages vont induire différentes interférences entre les messages des rafales et en conséquence des délais de bout en bout aussi différents. Le délai qu'un certain message subit est entièrement dépendant de ces déphasages. Le problème est de trouver la situation critique conduisant au pire délai et la combinaison des décalages entre les sources correspondante. La question à se poser est alors ; comment trouver cette situation critique ? La réponse est discutée durant

l'explication du principe de la méthode MFAV dans la section 4.2. Toutefois, une autre question se pose au préalable ; combien de rafales consécutives de chaque source devons nous utiliser dans l'analyse des interférences pour trouver le pire cas ? Deux cas peuvent être discutés pour répondre à cette question :

– Cas 1 : *les délais de bout en bout sont clairement inférieurs à la période minimale des générateur de rafales*. C'est souvent le cas en pratique et cela réduit remarquablement les calculs. En effet, dans ce cas nous pouvons déterminer des domaines d'interférence disjoints très limités comme sur la figure 2. Un message appartenant à un domaine donné n'interfère pas avec les messages des autres domaines. Ceci peut être expliqué intuitivement par le fait que si la $k^{ème}$ rafale notée b_i^k générée par la source S_i interfère avec la $l^{ème}$ rafale b_j^l de la source S_j, elle ne peut pas interférer avec b_j^{l+1} et les rafales suivantes. En effet, b_i^k atteint sa destination et b_j^{l+1} n'est même pas générée encore (le délai de bout en bout étant très inférieur à la période). Une rafale d'une source donnée peut interférer au plus avec une seule rafale d'une autre source. Finalement, l'analyse des interférences (via MFAV) est faite en prenant en compte une seule rafale de chaque source.

– Cas 2 : Si au contraire la condition précédente n'est pas clairement vérifiée, nous prenons deux ou plusieurs rafales successives (séparées par la période de la source) et nous les utilisons dans l'analyse. Comme nous verrons par la suite, le nombre de rafales prises en compte n'influe pas sur le principe de MFAV mais seulement sur le nombre de calculs à accomplir (plus en détail dans la section 4.2).

4. Méthode d'évaluation des délais à base de files d'attente virtuelles (MFAV)

4.1. Définitions

Avant d'entamer l'explication du principe de MFAV, nous allons donner quelques définitions de certains concepts impliqués dans cette méthode.

– *Buffer FIFO Virtuel (BFV)* : Nous associons à chaque file d'attente (réelle) un buffer FIFO virtuel (BFV) dont l'identifiant est le numéro de celle-ci. Un buffer virtuel (BV) est un ensemble ordonné de couple (p, θ) où $p = \{S_i, D_j, L\}$ est un paquet de longueur L généré par la source S_i et dont la destination est D_j. θ est la date d'entrée du premier bit du paquet dans la file d'attente réelle.

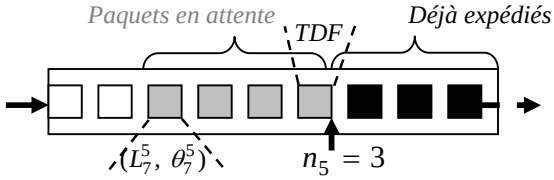

Figure 3. *Buffer FIFO virtuel n°5 (file d'attente n° 5)*

Nous associons aussi un compteur à chaque BV pour définir le nombre de paquets qu'il a déjà expédiés. Nous notons (p_l^k, θ_l^k) le $l^{ème}$ paquet reçu par le $k^{éme}$ BV dont le compteur est noté n_k (illustré sur la figure 3). Le BV est dit FIFO ou BFV si $\theta_{(l+1)}^k > \theta_l^k$, $\forall k, l \geq 1$ i.e. les paquets sont ordonnés selon leur ordre d'arrivée. Le premier arrivé est le premier expédié. Un BFV est rempli comme suit ; des que la date d'arrivée d'un paquet dans la file d'attente réelle est connue, le BFV correspondant est chargé avec ce paquet. A un moment donné, le BFV contient non seulement les paquets déjà reçus (réellement) à ce moment mais aussi les paquets futurs (pourvu que leurs dates d'arrivée soient connues). Quand un paquet est expédié, le compteur du BFV est incrémenté pour pointer le paquet suivant, pour une éventuelle expédition après. Les têtes de files (TDF) des paquets en attente (en gris sur la figure 3) sont connues à tout moment grâce à ces compteurs.

– *Date d'accès possible :* Dans les systèmes étudiés, les ressources partagées traitent les paquets suivant le mode PAPS sans aucune différentiation préalable. Le premier arrivé est le premier servi et chaque paquet doit attendre son tour avant d'accéder à une ressource. Par conséquent, des délais importants dus à l'attente sont causés et doivent être évalués. Pour cela, nous introduisons la notion de date d'accès possible (DAP). DAP est la date où l'accès est possible à une ressource. Comme on peut avoir plus d'une ressource dans tout le système, à chaque ressource SR_k nous associons une date d'accès possible DAP_k. L'exemple simple ci-après (très analogue à un commutateur) est donné pour expliquer cette notion.

Exemple 1 : un système usuel d'atelier avec un chariot en translation est utilisé (figure 4). Le chariot (ressource partagée SR de capacité $q = 1$) reçoit des paquets (sens large) de trois buffers d'entrée et les expédie vers un port de sortie commun.

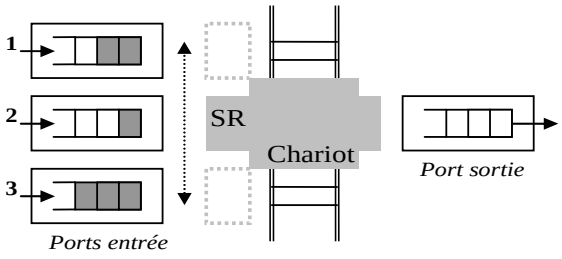

Figure 4. *Exemple de système avec ressource partagée*

SR se déplace de sorte à recevoir un paquet d'une seule entrée parmi les trois suivant l'ordre d'arrivée des paquets en amont. Cet ordre est donné dans la colonne 5 de Tableau 1 pour faciliter l'explication. Sur la figure 4, deux paquets entrent dans le buffer n°1, un seul dans le buffer n°2 et trois dans le buffer n°3. La date d'arrivée d'un paquet p_{ij} est notée $\theta(p_{ij})$ et le temps nécessaire pour son expédition est $T(p_{ij})$. Les différentes dates d'arrivée des paquets depuis tous les buffers sont

connues et les buffers FIFO virtuels correspondants sont conformément chargés. A $t = 0$, le chariot est supposé libre.

	Paquets	Dates d'arrivée (s)	Temps d'expédition (s)	Ordre d'arrivée
Buffer n°1	p_{11}	5.4	3	2
	p_{12}	14.2	4.5	4
Buffer n°2	p_{21}	3.6	6	1
Buffer n°3	p_{31}	12.9	2.4	3
	p_{32}	15.3	7	5
	p_{33}	24.1	2	6

Tableau 1. *Caractéristiques de l'exemple*

A $t = 0$, la date d'accès possible dans SR notée DAP_{SR} est égale à zéro puisqu'à cette date l'accès dans SR est possible. Le premier paquet arrivé dans le système est p_{21} (voir colonne 5) puisqu'il entre le plut tôt à la date $\theta(p_{21}) = 3.6s$. Dès que ce paquet arrive, il est expédié par le chariot durant un temps $T(p_{21}) = 6s$. L'accès est possible pour le deuxième paquet seulement à la date (3.6+6)s. Donc, la nouvelle date d'accès possible est $DAP_{SR} = 9.6s$. Le second paquet p_{11} qui arrive à $t = 5.4s$ est expédié seulement à cette date (le maximum entre $t = 5.4s$ et $t = 9.6s$ i.e. à t=max(9.6, 5.4)). Cela signifie un temps d'attente pour p_{11} égal à (9.6-5.4)s. L'expédition de p_{11} est accomplie à la date $9.6 + T(p_{11}) = 12.6s$ et la date d'accès possible devient $DAP_{SR} = 12.6s$. Le troisième paquet qui arrive à la date $\theta(p_{31}) = 12.9s$ est expédié sans attente puisque le chariot est libre à cette date. Nous pouvons noter que la date d'accès est aussi exprimée comme étant t=max(12.9, 12.6). Ainsi, une formule générale de date d'expédition peut être déduite et les délais d'attentes calculés en répétant le mécanisme précèdent. Le même principe peut être appliqué pour un système avec un nombre quelconque de ressources.

REMARQUE. — la tête de file (TDF) d'un buffer est en réalité une ressource partagée. En effet, un paquet ne peut quitter une file d'attente que si la TDF est libre i.e. tous les paquets en tête de file sont déjà expédiés. Nous considérons alors qu'il y a toujours une ressource partagée en aval d'une file d'attente. Comme seulement les buffers alimentant les nœuds destinations n'en ont pas, nous considérons que chaque nœud destination D_j est précédée par une ressource implicite notée $SR\,\mathrm{Im}_j$. La date d'accès possible à cette ressource est notée $DAP\,\mathrm{Im}_j$.

4.2. Principe de MFAV

L'objet de cette section est la description du principe de l'approche MFAV pour déterminer l'évolution du système, les décalages τ_i entre les différentes sources étant supposés connus. L'évaluation en est déduite suivant les étapes ci dessous :

– Etape 1 : les dates de début de génération des rafales sont connues et les dates d'arrivée de leurs paquets, dans les files d'attentes en aval des nœuds source, sont déduites. De la, nous chargeons tous les BFVs correspondants avec ces paquets. Les compteurs de tous les BFVs sont mis à zéro ($n_k = 0, \forall k$) puisque pour l'instant aucun paquet n'est encore expédié.

Figure 5. *Initialisation des BFVs de l'exemple de la figure 1*

Si nous reprenons l'exemple de la figure 1 avec le scénario de génération de rafales montré sur la figure 2, nous pouvons initialiser les BFVs comme montré sur la figure 5. Le BFV n°1 par exemple, est chargé avec deux paquets $(p_1^1,\ \theta_1^1)$ et $(p_2^1,\ \theta_2^1)$ où $p_1^1 = \left\{S_1, D_1, L_1^1\right\}$ et $p_2^1 = \left\{S_1, D_2, L_2^1\right\}$. La même chose est faite pour les autres buffers concernés (figure 5). Si par exemple deux rafales consécutives depuis S_1 sont considérées pour l'analyse, nous chargeons le BFV n°1 avec deux rafales (soit quatre paquets dont les dates d'arrivée sont connues puisque la période de la source l'est) sans pour autant changer le principe de MFAV.

– Etape 2 : Tous les buffers en concurrence pour accéder aux ressources sont connus. Parmi ces BFVs (1-2-3-6 dans l'exemple à l'initialisation), nous sélectionnons le BFV dont la TDF a la date d'arrivée minimale i.e. le paquet entré le plus tôt dans le système. Comme mentionné précédemment, dans le $k^{ème}$ buffer dont le compteur à un moment donné est $n_k = n_t$, la TDF est le $(n_t + 1)^{ème}$ paquet entré dans ce buffer. Si, parmi les BFVs en concurrence, le $K^{ème}$ BFV est sélectionné, il doit vérifier la condition : $\theta^{K}_{(n_K+1)} = \min_k(\theta^k_{(n_k+1)})$. L'élément récepteur est déterminé en utilisant les fonctions *PATH* et *LINK* , la source et la destination étant connues (elles sont des composantes du paquet sélectionné). A ce stade, deux cas sont à étudier suivant le type de récepteur : une ressource partagée ou un nœud destination :

a) Ressource partagée : supposons que c'est la ressource numéro l notée SR_l. Le paquet accède à cette ressource une fois qu'il est totalement reçu dans le $K^{ème}$ buffer et à condition que la ressource SR_l soit disponible. La

date de début d'expédition notée φ du paquet sélectionné est :
$\varphi = \max(\theta^{\mathrm{K}}_{(n_{\mathrm{K}}+1)} + L^{\mathrm{K}}_{(n_{\mathrm{K}}+1)} / C, DAP^I_l)$ où C est la vitesse de réception dans la file en en cours (la $\mathrm{K}^{ème}$). Si la ressource SR_l est de capacité q_l, le paquet accède à la première sous-ressource à devenir libre i.e. à la date $DAP^I_l = \min_{i \le q}(DAP^i_l)$ où DAP^i_l est la date d'accès possible à la $i^{ème}$ sous-ressource de SR_l. La date de début d'expédition φ est déterminée et la $I^{ème}$ sous-ressource de SR_l est occupée jusqu'à la fin de cette tache à la date $\xi = \varphi + L^{\mathrm{K}}_{n_{\mathrm{K}}+1} / C^l_{SR}$ où $L^{\mathrm{K}}_{n_{\mathrm{K}}+1}$ est la longueur du paquet en question. La nouvelle date d'accès possible doit être mise à jour : $DAP^I_l = \xi$. De là, nous pouvons calculer le délai mis dans la file d'attente K qui est donné par : $D_{queue}(\mathrm{K}, n_{\mathrm{K}}+1) = \xi - \theta^{\mathrm{K}}_{n_{\mathrm{K}}+1}$. En fait, le paquet sélectionné a été expédié par la ressource partagée vers une file d'attente en aval que nous supposons de numéro s (doit être déterminé en utilisant $PATH$). Ainsi, le $s^{ème}$ buffer reçoit un nouveau paquet et par conséquent son BFV doit être chargé avec ce paquet (un paquet a été enlevé du $\mathrm{K}^{ème}$ BFV et ajouté dans le $s^{ème}$ BFV). Aller dans Etape2.

b) Nœud destination D_j : de manière similaire que précédemment, le paquet peut quitter la file s'il est complètement reçu et la tête de file est libre (pour rappel, la TDF est une ressource partagée implicite). Le départ de la file est seulement possible à la date : $\psi = \max(\theta^{\mathrm{K}}_{(n_{\mathrm{K}}+1)} + L^{\mathrm{K}}_{(n_{\mathrm{K}}+1)} / C, DAP\,\mathrm{Im}_j)$. Le paquet est entièrement expédié à la date $\varsigma = \psi + L^{\mathrm{K}}_{(n_{\mathrm{K}}+1)} / C_j$ où C_j est la vitesse de réception du nœud D_j. La date d'accès possible de la TDF est mise à jour : $DAP\,\mathrm{Im}_j = \varsigma$. Finalement, le délai dû à l'attente dans le buffer peut simplement être calculé : $D_{queue}(\mathrm{K}, n_{\mathrm{K}}+1) = \varsigma - \theta^{\mathrm{K}}_{n_{\mathrm{K}}+1}$. Cette fois aucun buffer n'est alimenté et on fait seulement un test si tous les paquets présents dans le système sont expédiés. Si c'est le cas, l'algorithme est fini. Sinon, aller dans *Etape2*. Finalement, le délai de bout en bout peut être déterminé par sommation de tous les délais D_{queue} mis dans toutes les files d'attente traversées par un paquet depuis sa génération d'une source jusqu'à son arrivée à une destination. L'algorithme précédent est représenté étape par étape sur le schéma de la figure 6 et permet la détermination de l'évolution de tout système, les dates de génération des rafales étant fixées à l'avance.

4.3. Evaluation du pire délai de bout en bout en utilisant MFAV

Dans la section précédente, nous avons décrit MFAV pour un ensemble donné de dates (ou de décalages) de début d'émission des rafales fixées à l'avance. Comme les délais de bout en bout sont entièrement dépendants des décalages entre les

sources, la question suivante se pose ; comment trouver la combinaison de ces décalages qui mène vers le pire délai de bout en bout qu'un paquet p peut subir?

– *Lemme :* Soit θ la date d'entrée du paquet p dans une file d'attente où il subit un délai Δ. Soit p^* un autre paquet (d'une autre file en concurrence avec p pour accéder à une ressource partagée) qui entre immédiatement avant p à la date θ^*. Si le délai Δ est maximal, alors la date d'arrivée de p est nécessairement $\theta = \theta^* + \varepsilon^*$ avec $\varepsilon^* = 0^+$ (p et p^* entrent quasi simultanément).

$$\Delta \text{ est maximal} \Rightarrow \theta = \theta^* + \varepsilon^*$$

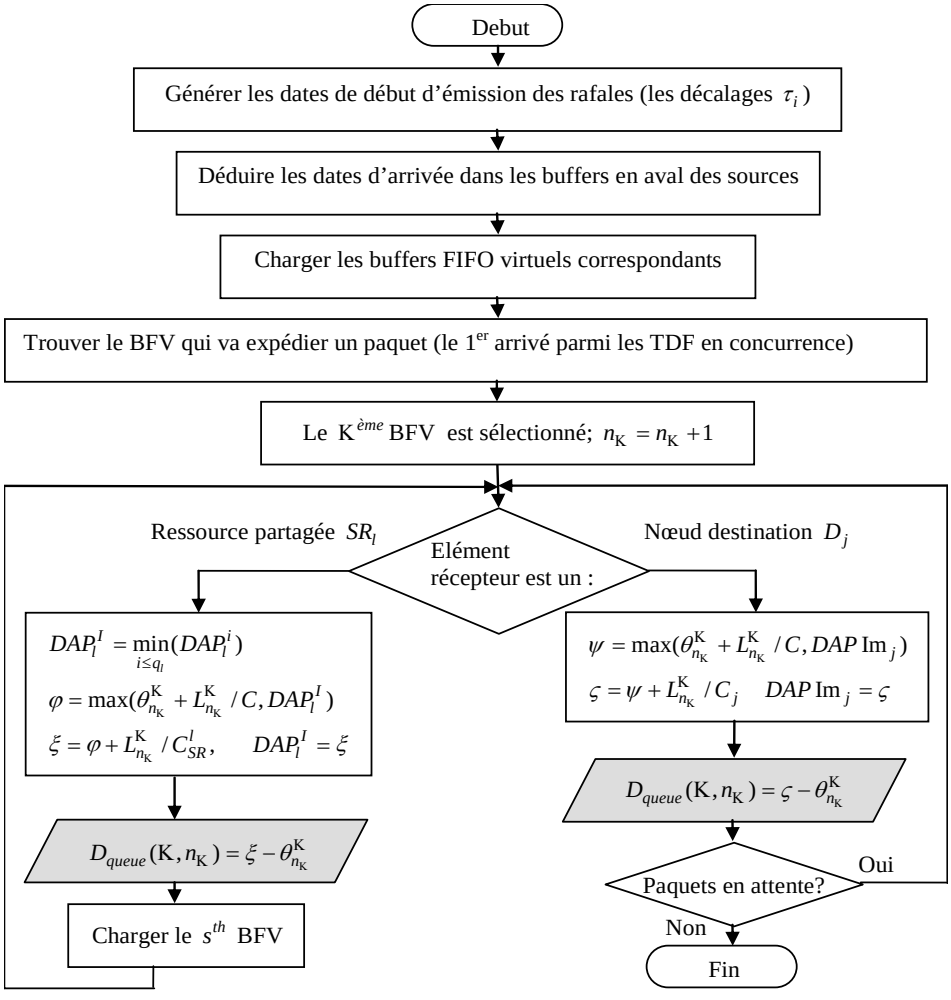

Figure 6. *Algorithme de MFAV*

Démonstration (par l'absurde): soit $\theta = \theta^* + \varepsilon$ la date d'arrivée du paquet p avec $\varepsilon > \varepsilon^*$ (pas de quasi simultanéité). Pour rester sous la condition : p^* est le paquet entrant immédiatement avant p, nous posons $\varepsilon < \delta$ où δ est la période minimale d'inter-arrivée des paquets d'une rafale. Supposons que $\varepsilon > \varepsilon^*$ mène vers le délai maximal noté Δ^*. Si maintenant le paquet p entre à la date $\theta' = \theta^* + \varepsilon'$ où $\varepsilon > \varepsilon' > \varepsilon^*$. Donc, p entre plus tôt que dans le cas précèdent mais toujours après le paquet p^* et par conséquent sans influence sur les paquets qui le précédent, notamment p^*. Ceci implique que le paquet est expédié à la même date que précédemment. Dans ce cas, le délai subi est égal à $\Delta' = \Delta^* + (\varepsilon - \varepsilon')$ i.e. $\Delta' > \Delta^*$ puisque $\varepsilon > \varepsilon'$. Ceci est absurde puisque depuis le début Δ^* est le délai maximal. □

– *Théorème:* Pour trouver le pire délai, nous recherchons les valeurs des décalages τ_i dans un domaine $[-T, +T]$. Si un pas $\delta_e < \delta$ de variation des décalages τ_i est utilisé pour cela, une recherche exhaustive est accomplie et la précision du pire délai trouvé est de δ_e. En d'autres termes, l'erreur d'estimation du pire délai de bout en bout est au plus égale à δ_e.

Démonstration: d'après le lemme précédent, le pire délai pour un paquet p est atteint si celui-ci entre juste après un autre paquet p^* (quasi simultanément). Comme le pas de recherche est inférieur à la période minimale d'inter arrivée, nous sommes sûr de balayer ce cas où p entre juste après le paquet p^* (même si ce n'est pas nécessairement quasi simultanément). Ainsi, la sous-estimation du délai est égale exactement à la déviation par rapport au cas de quasi simultanéité. La pire déviation est atteinte quand p entre δ_e après p^* i.e. la précision est égale à δ_e. □

Comme une surestimation du délai est recherchée, δ_e doit être ajouté à la valeur trouvée à la fin. Si la précision n'est pas satisfaisante, nous pouvons réduire le pas δ_e et le domaine de recherche après avoir cerné le pire cas avec la procédure précédente (serrer le pas et le domaine progressivement).

NOTE. — Avec le protocole de communication Ethernet par exemple ; $\delta = \min_k((72 + IFG)/C_{in}^k)$, *IFG* étant le temps d'inter trames (96bits) et 72octets est la longueur minimale d'une trame (préambule inclus).

REMARQUES. — 1) Le trafic apériodique comme des signaux d'alarmes (e.g. détection d'une fuite de fluide) peut être considéré avec MFAV. Si la période minimale d'inter-arrivée des alarmes est clairement supérieure aux délais de bout en bout (comme la condition du Cas 1 de la section 3), aucune différence dans la procédure par rapport au trafic périodique n'est à noter puisque dans ce cas, une seule alarme est considérée dans l'analyse. Si en revanche la condition n'est pas vérifiée, une source d'alarmes peut être considérée comme un groupe de sources indépendantes générant des alarmes avec des périodes d'inter-arrivée inconnues (supérieures à une borne connue tout de même). Ces périodes sont équivalentes à des décalages τ_i qu'il faut calculer en recherchant le pire cas en procédant de la même manière que précédemment. Par ailleurs, MFAV peut être utilisée avec un

trafic encore plus général mais le nombre de calculs pourrait devenir assez important (Malgré le temps virtuel de MFAV).

2) MFAV peut être aussi utilisée pour l'évaluation de la moyenne des délais de bout en bout en considérant une série de rafales successives générées par chaque source et en faisant évoluer le système sur un long horizon.

5. MFAV et les réseaux Ethernet commutés

Pour pouvoir appliquer MFAV, il suffit de représenter un système en utilisant uniquement les blocs de base décrits dans la section 2. Dans cette section, nous procédons de la sorte pour étudier les réseaux Ethernet commutés en considérant les principales architectures de commutation (suivant le mode de bufferisation utilisé). Ethernet est en effet le protocole LAN le plus utilisé dans les bureaux et de plus en plus comme réseau de terrain dans les systèmes d'automatisation.

5.1. Commutateur avec bufferisation en entrée

Un commutateur est dit avec bufferisation en entrée si les congestions des trames sont localisées au niveau des entrées. Cette architecture est caractérisée par des buffers très larges aux entrées et très limités aux sorties (figure 7). Ce type de commutateur souffre du problème de blocage de tête de ligne. Celui-ci apparaît quand une trame de tête de ligne ne peut pas être expédiée car le buffer de sortie est déjà saturé et bloque toutes les trames suivantes même si ces dernières ont des ports de sorties libres. Cette caractéristique doit être prise en compte dans le modèle.

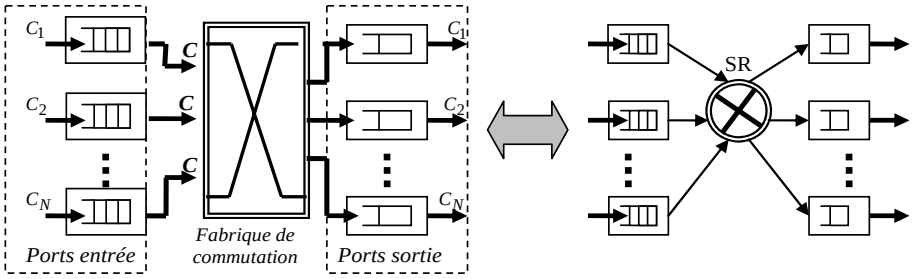

Figure 7. *Modèle de commutateur avec bufferisation en entrée*

Sur la figure 7, un modèle de ces commutateurs est représenté en utilisant uniquement les blocs de base présentés précédemment. Cependant, les ports de sortie sont de largeur limitée (nous supposons qu'une sortie peut contenir au maximum une trame) et par conséquent quelques changements doivent être apportés. Pour cause, une trame f peut être expédiée vers une sortie s si deux conditions sont satisfaites à la fois: la fabrique de commutation (notée SR_k) est

libre et le port de sortie est vide. Au lieu de représenter le port de sortie s par un buffer, il est plus judicieux de le considérer comme une ressource partagée implicite avec une date d'accès possible $FAD\,\mathrm{Im}_s$. Une trame peut être expédiée si elle est complètement reçue et les deux ressources SR_k et $SR\,\mathrm{Im}_s$ sont libres i.e. à la date : $\varphi = \max(\theta_l^K + L_l^K / C, DAP_k, DAP_s)$. Finalement, l'algorithme général de MFAV décrit précédemment reste le même avec seulement de légers changements.

5.2. Commutateur avec bufferisation en sortie

On parle de commutateur avec bufferisation en sortie quand les trames sont mises dans des files d'attente après leur commutation (figure 8). Le problème de blocage a été éliminé mais des buffers en sortie très larges et une vitesse de commutation relativement élevée sont requis. Une autre approche aussi souvent adoptée est d'ajouter une mémoire partagée entre les entrées et les sorties pour y transférer le problème de blocage. Ceci permet d'éviter l'implémentation de très larges buffers dans toutes les entrées qui ne serviraient pas pleinement en pratique puisqu'un blocage de toutes les entrées à la fois n'est pas très probable. Une combinaison des deux approches précédentes est fréquemment utilisée aussi (figure 8).

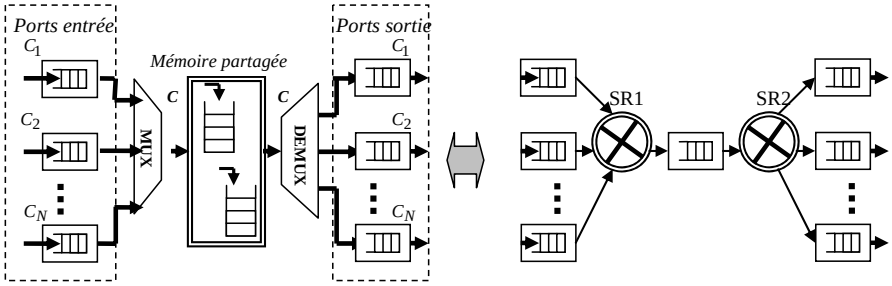

Figure 8. *Commutateur avec bufferisation en sortie et mémoire partagée*

Sur la figure 8, un modèle de commutateur avec buffers en sortie et mémoire partagée est représenté à l'aide des blocs de base. MFAV peut être appliquée directement.

6. Application et validation de MFAV

Pour valider la méthode de modélisation et d'évaluation proposée, nous considérons une plateforme réelle (figure 9) déjà étudiée dans (Georges *et al.*, 2005). En effet, les trames y sont générées périodiquement et donc adéquat pour la vérification de notre approche. Le système est constitué de deux commutateurs (bufferisation en sortie et mémoire partagée) ainsi que trois PCs pour générer le trafic dans le réseau (tous les liens à 10 Mbps). Pour éviter des problèmes de mesure

dus à la non synchronisation des horloges de la source S2 et la destination D2, leurs interfaces Ethernet (*Eth1* et *Eth2*) partagent la même horloge (celle de PC2).

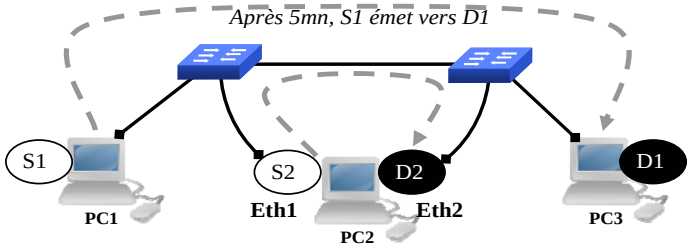

Figure 9. *Plateforme étudiée*

Durant 10 mn, S2 envoie toutes les secondes des trames de 72 octets vers D2. Après 5 mn du début, S1 commence à envoyer, en parallèle, des trames de 1026 octets vers D1 avec une période de 10 ms. Deux phases de génération de trafic sont étudiées. Il s'agit d'évaluer le pire délai de bout en bout qu'une trame envoyée depuis S2 vers D2 peut endurer en présence et en absence de trafic généré par S1. Nous appliquons MFAV pour ce système et comparons les résultats obtenus avec des mesures expérimentales et du calcul réseau déterministe (figure 10).

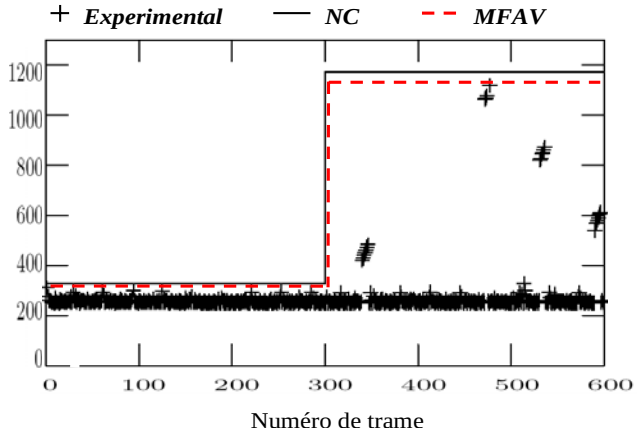

Figure 10. *Résultats d'évaluation des délais*

Durant la 1ère phase, toutes les mesures expérimentales restent en dessous de 312 μs et en dessous de 1119 μs durant la 2nde phase. Ce seront les références pour la comparaison des différents résultats. Georges *et al.* ont appliqué le calcul réseau et ont abouti à des majorants des délais de bout en bout de 328 μs pour la 1ère phase et 1173 μs pour la 2ème phase, soit une surestimation d'environ 5% dans les deux cas (ligne continue sur la figure 10).

Nous avons appliqué la méthode MFAV pour cette plateforme et nous avons abouti à des estimations de $315\ \mu s$ pour la 1ère phase et $1130\ \mu s$ pour la 2nde phase, soit environ 1% de surestimation du pire délai de bout en bout (ligne discontinue sur la figure 10). Ainsi, nous avons obtenu des résultats moins pessimistes que ceux du NC tout en surestimant les délais réels i.e. une évaluation stricte et précise à la fois.

7. Conclusion et perspectives

Dans ce papier, une nouvelle approche de modélisation et d'évaluation des systèmes de communication a été proposée. Une preuve formelle sur la capacité à garantir l'évaluation du pire délai de bout en bout a été donnée. Sur un cas d'étude pratique, nous avons appliqué la méthode et comparé ses résultats à des mesures réelles et les résultats du NC. Pour les travaux futurs, il serait intéressant de considérer des systèmes plus complexes avec prise en compte des pertes de paquets.

8. Bibliographie

Addad B., Amari S., « Delays evaluation and compensation in Ethernet networked control systems », *16th Annu. Conf. Real Time and Network Systems*, Rennes, 2008, p. 139-148.

Addad B., Amari S., « Modeling and response time evaluation of Ethernet based automation systems using Max-Plus algebra and timed events graphs », *4th Annu. IEEE Conf. Automation Science and Engineering*, Washington DC, 2008, p. 418-423.

Cruz R. L., « A calculus for network delay, Part I: network in isolation », *IEEE Trans. Information Theory*, Vol. 37, n° 1, 1991, p. 114-131.

Cruz R. L., « A calculus for network delay, Part II: Network analysis», *IEEE Trans. Information Theory*, Vol. 37, n° 1, 1991, p. 132-141.

Fan X., Jonsson M., Jonsson J., « Guarabteed real time communication in packet switched network with FCFS queuing », *Computer and Networks*, Vol. 53, n° , 2008, p. 400-417.

Georges J. P., Divoux T., Rondeau E., « Validation of the network calculus approach for the performance evaluation of switched Ethernet based industrial systems », *16th IFAC world Congress*, 2005.

Gunter B., Stefan G., Hermann de. M., Kishor S. T., *Queuing networks and Markov chains*, Editions John Wiley & Sons, 2006.

John N. D., *Queuing Theory with application to packet communication*, Boston, Editions Springer, 2005.

Le Boudec J. Y, Thiran P., « Network Calculus: a theory of deterministic queuing systems for Internet », Editions Springer Verlag, 2004.

Scharbarg J *et al*., A probilistic analysis of end to end delays on an AFDX avionics network, *IEEE Trans. Industrial Informatics Theory*, Vol. 5, n° 1, 2009, p. 38-49.