Thèse

présentée par

Cyril Labbé

pour obtenir le titre de Docteur
de l'Institut National Polytechnique de Grenoble
(arrêté ministériel du 30 Mars 1992)

Spécialité : **Informatique**

Évaluation de performance et modélisation de réseaux à haut débit à l'aide d'une architecture reconfigurable

---

Date de soutenance : 1999

Composition du jury :

Président :	? ?	? ?	
Rapporteurs :	? ?	? ?	
	? ?	? ?	
Examinateurs :	Vincent	Olive	
	Brigitte	Plateau	(directeur)
	Frédéric	Reblewski	
	Jean-Marc	Vincent	(co-directeur)

Thèse préparée au sein du
Laboratoire de Modélisation et Calcul
(Institut de Mathématiques Appliquées de Grenoble)

Table des matières

Glossaire

AF : Arrival First
ATM : Asynchronous Transfert Mode
BLP : Bloc Logique Programmable
CAC : Call Admission Control
CMS : Commutateur Multi-Service
CNET : Centre National d'Étude des télécommunications
DC : Débit Crête
DF : Departure First
Dmg : Débit minimum garanti
DSP : Digital Sound Porcessor
FAST : FPGA-based ATM Simulation Testbed
FIFO : First In First Out
FPGA : Field Programmable Gates Array
FQ : Fair Queuing
GCRA : Generic Cell Rate Algorithm
GPA : Générateur Pseudo-Aléatoire
GPS : Generalized Processor Sharing
ITE : Instant Théorique d'Emmission
LB : Leaky-Bucket
MDC : Matrice De Connexion
MMBP : Markovian Modulated Bernouilli Process
MMPP : Markovian Modulated Poisson Process
MPEG : Moving Pictures Expert Group
NITE : Nouvel Instant Théorique d'Emission
PAM : Programmable Active Memories
PGPS : Paket-by-packet Generalised Processor Sharing
QdS : Qualité de Service
RM : Resource Management
SB : Seuil Bas

SEFF : Smallest Eligible virtual Finishing time First
SFF : Smallest virtual Finishing time First
SH : Seuil Haut
SSF : Smallest virtual Starting time First
TCP/IP : Transfert Controle Protocol / Internet Protocol
TTL : Transitor Transitor Logic
VCI : Virtual Circuit Identifier
VHDL : VHSIC Hardware Description Level
VLSI : Very Large System Integrated
VPI : Virtual Path Identifier
WFQ : Weighted Fair Queuing

Introduction

L'objectif de la thèse

L'objectif de la thèse est de valider une méthode de simulation pour les réseaux à haut débit. La démarche s'appuie sur la méthode classique de modélisation des réseaux à l'aide de files d'attente. Les files d'attente sont exprimées dans un langage habituellement utilisé pour la description des circuits intégrés. Elles sont donc représentées à l'aide des opérateurs logiques qui composent les circuits intégrés (portes logiques de base, bascules, mémoires...) que nous désignerons par la suite sous le termes de "ressources matérielles". Une machine à base de circuits reconfigurables (Sim-express[1]) permet ensuite l'émulation de ces files d'attente et la mesure d'indices de performance. L'utilisation d'une architecture reconfigurable rend possible l'étude du comportement d'un circuit sans pour autant avoir à le construire réellement sur silicium.

La démarche consiste donc à décrire un circuit, principalement composé de files d'attente et d'automates, qui modélise le comportement du réseau. L'étude, à l'aide de l'architecture reconfigurable, du comportement et des événements survenant dans le circuit apportera ainsi des renseignements sur le fonctionnement du réseau et sur les performances des protocoles utilisés.

Ce document comporte trois grandes parties. La première expose les raisons qui poussent à rechercher de nouvelles méthodes de simulation, la deuxième décrit en détail l'approche que nous proposons, enfin la nouvelle méthode est mise à l'épreuve de manière systématique dans la troisième partie.

Pourquoi une nouvelle méthode ?

Avec la popularité croissante des applications distribuées et multimédia, le rôle traditionnel des réseaux informatiques se transforme. Le même réseau est censé intégrer des services aussi différents que le transfert en temps réel (téléphonie, la visioconférence...), le transfert de données informatiques (e-mail, fax, fichiers...), ou la gestion de systèmes répartis (bases de données...). Ainsi, les réseaux à intégration de services deviennent une méthode de communication standard, en remplaçant l'utilisation d'anciens schémas basés sur la commutation de circuit de type téléphonique. De plus, les débits ne cessent d'augmenter et le nombre d'utilisateurs croît de manière exponentielle. Tous ces facteurs font de la mise au point des réseaux à intégration de services une tâche particulièrement importante.

[1] Machine commercialisée par la société Mentorg-Graphics.

Le développement de nouveaux protocoles réseaux

Le développement des réseaux à intégration de services passe non seulement par l'amélioration et l'adaptation des protocoles de communication existants, mais aussi par la mise au point de nouvelles techniques permettant de répondre à de nouveaux impératifs. En effet, un transfert de données informatiques doit pouvoir laisser "passer" une connexion ayant des exigences temporelles (téléphonie). A l'inverse, la perte d'information est grave pour des données informatiques, ce qui n'est pas forcement le cas pour l'image d'une visioconférence.

Ainsi, pour éviter que les différents types de services se nuisent mutuellement, le partage des ressources fournies par le réseau doit se faire de manière intelligente. Plus précisément, ce partage doit être assez dynamique pour permettre une gestion cohérente et efficace du trafic. Il est aussi nécessaire d'utiliser des méthodes performantes afin de prévenir et éventuellement résorber les congestions principales sources d'inefficacité. Bien entendu, l'utilisation des ressources est voulue optimale pour rentabiliser au mieux les investissements. La recherche de solutions à ces problèmes est indispensable pour assurer les qualités de services requises.

Le mode de transfert asynchrone (ATM) a été une des premières techniques de commutation à mettre en évidence les problèmes qui naissent de l'intégration des services sur les réseaux à haut débit. Cependant tous les types de réseaux qui cherchent à garantir des qualités de services sont confrontés à des problèmes de nature similaire.

Modélisation et analyse de performance : une étape obligatoire

La conception de protocoles et de techniques qui permettent une meilleure intégration des services passe immanquablement par une phase d'évaluation et de mesure de performances utilisant des techniques de modélisation. Cette phase de modélisation doit permettre une évaluation précise des techniques utilisées, et d'identifier les différentes parties du système qui limitent son efficacité. Le but de cette étape est d'éviter le déploiement de techniques et de structures non adaptés et ainsi, minimiser le risque de redéploiements coûteux.

Les difficultés rencontrées lors de cette phase de validation sont nombreuses. En effet, les méthodes de résolution traditionnelle sont peu ou pas adaptées à ce type de contextes. Cette relative inadaptation provient principalement du nombre d'évènements qu'il faut traiter pour pouvoir modéliser finement ce type de réseaux. Typiquement, la mise en évidence d'évènements rares est particulièrement difficile avec les outils traditionnels. De plus, il est parfois délicat de déterminer l'impact d'une décision prise à l'échelle de temps des connexions sur les phénomènes qui apparaissent à un niveau plus fin. En effet, les outils traditionnels ne permettent pas de comprendre ces phénomènes complexes qui nécessitent la génération d'un trop grand nombre d'évènements. Toutes ces considérations seront exposées dans la permière partie de la thèse.

Les architectures reconfigurables...

... pour la conception de circuits ...

Les circuits reconfigurables sont des outils relativement récents. Leur utilisation est devenue de plus en plus courante ces dernières années. En effet, leur utilité dans le domaine de la conception de circuits s'est très vite imposée. Ainsi, les architectures reconfigurables sont aujourd'hui pleinement intégrées dans la chaîne de conception des circuits intégrés.

Le principe de base de ces machines est de donner à l'utilisateur la possibilité d'utiliser des ressources matérielles pour émuler un circuit intégré. Cette technique a pour objectif de tester les fonctionnalités d'un circuit sans faire appel à de trop coûteuses simulations. Les architectures reconfigurables sont aussi très appréciées dans le cadre du co-design. Elles participent ainsi activement à raccourcir le du cycle de production des circuits intégrés.

... et pour d'autres utilisations.

Cependant, les machines à base d'architectures reconfigurables ont de nombreuses autres applications et leur utilisation a été testée dans un grand nombre de domaines. Ainsi, d'autres débouchés commerciaux ont été ouverts, notamment en biologie.

De manière générale, une machine à architecture reconfigurable est construite comme un assemblage de circuits reprogrammables qui en constituent les éléments de base. L'utilisation d'une telle machines peut se faire suivant plusieurs approches.

La plus courante consiste à créer une machine dédiée à une utilisation particulière. L'assemblage de circuits reprogrammables est alors optimisé pour l'application visée.

Une autre approche possible consiste à utiliser une machine dédiée à l'émulation de circuits. Une telle machine est alors considérée comme une architecture reconfigurable "généraliste" capable d'être utilisée pour traiter n'importe quel problème. C'est cette dernière approche qui a été retenue dans le cadre de la thèse. La machine Sim-express du CNET, initialement dédiée à l'émulation de circuits, a été utilisée pour l'étude des protocoles des réseaux à haut débit.

Validation de la méthode

L'approche proposée a été mise à l'épreuve à l'aide de trois réalisations différentes. Ces trois réalisations ont toutes pour objectif l'étude de protocoles réseaux. Elles ont toutes des objectifs différents et l'ensemble montre que la simulation à l'aide d'architecture reconfigurable peut être utilisée pour mettre en lumière des phénomènes non observables avec des techniques usuelles.

Pour commencer, nous nous sommes intéressés à une modélisation des commutateurs ATM sous la forme de réseaux multi-étages. L'étude de ces modèles, a permis de montrer que l'évaluation d'évènements rares est possible grâce à l'utilisation de ressources matérielles. Des taux de perte réalistes ont ainsi pu être évalués de manière précise. De plus, les composants réalisés permettent aussi d'étudier les propriétés du trafic qui traverse ces commutateurs.

Nous avons aussi testé l'utilisation d'une politique de service équitable (Fair Queuing). Ceci permet de montrer qu'il est possible de réaliser des simulations de service complexe à l'aide de ressources matérielles. De plus, les simulation réalisées ont permis d'illustrer l'importance de la méthode de choix en cas d'égalité des marques.

Enfin, une modélisation très détaillée d'un commutateur a été réalisée. Cette modélisation a pour objectif de permettre le test et la mise au point d'un régulateur de trafic. Dans cette application, la machine Sim-express est couplée à une application logicielle (le régulateur) et elle est utilisée comme un accélérateur de simulation. Ainsi, les ressources matérielles ont servi à simuler un commutateur à l'échelle du temps cellule. En les couplant à un simulateur logiciel de régulation de trafic, l'étude des interactions entre les différents niveaux d'échelle

a été rendue possible. L'utilisation de ressources matérielles permet donc d'étudier l'impact de décisions prise à l'échelle de temps des connexions, sur l'échelle de temps des paquets.

Comme on le voit, le mode de transfert asynchrone est au cœur de toute la thèse, cependant il est important de garder à l'esprit que toute autre technique de commutation est susceptible d'être confrontée aux problèmes qui sont exposés. En effet, ces problèmes proviennent de l'intégration des services, de la commutation de paquets et de l'augmentation des débits. Tous les types de commutation par paquets sont donc susceptibles d'être confrontés au même type de problèmes.

Structure de la thèse

Première partie de la thèse

La première partie du document introduit la problématique qui entoure l'intégration des services sur des réseaux à haut débit de type ATM.

Elle commence donc par présenter la technique de commutation du mode de transfert asynchrone. Ceci permet de montrer les problèmes liés à l'intégration des services et à la volonté de garantir une qualité sur les différents services proposés (Chapitre I). Cette partie montre aussi l'importance de la modélisation dans le développement et la validation des nouveaux protocoles réseaux. Enfin, la méthode de modélisation des réseaux à base de files d'attente est exposée ainsi que les techniques de résolution les plus couramment utilisées (Chapitre II).

Cette partie permet ainsi d'aborder les problèmes qui se posent dans le cadre de l'étude des réseaux à haut débit et de mettre en évidence les points où l'utilisation d'une architecture reconfigurable peut s'avérer utile.

Deuxième partie de la thèse

La deuxième partie présente en détail la démarche proposée ainsi que les outils qui peuvent être utilisés pour sa mise en œuvre. Ainsi, on montre comment on a choisi de décrire les files d'attente en temps discret à l'aide de ressources matérielles. Ce sont ces descriptions qui seront utilisées pour modéliser des réseaux ATM. La description de la méthode a une certaine généralité, en effet, différentes architectures reconfigurables peuvent être utilisées pour émuler ces circuits. (Chapitre III).

Différentes architectures reconfigurables et leurs utilisations les plus courantes sont ensuite présentées. En particulier, on décrira l'architecture matérielle de la machine Simexpress ainsi que la chaîne logicielle qui permet de configurer cette machine. Cette partie permet ainsi de présenter les outils utilisés tout en montrant les "configurations" de base qui ont été réalisées pour la simulation des files d'attente (Chapitre IV).

Troisième partie de la thèse

L'objectif de la troisième partie de la thèse est de valider notre approche par les trois mises en œuvre de la méthode (Chapitre V, VI et VII). Elle présente donc les résultats des simulations qui ont été réalisées pour les différentes applications : calcul de taux de pertes, perturbation du trafic et enfin couplage de ressources matérielles et logicielles.

Bibliographie et Annexe

Chaque chapitre comporte sa propre bibliographie en fin de chapitre. Cependant, toutes celles-ci ont été regroupée dans une bibliographie générale à la fin de l'ouvrage.

D'autre part, on trouvera en annexe une comparaison entre notre méthode et des simulateurs logiciels classiques (NS et QNAP2).

Première partie

Réseaux haut débit et modélisation

Cette première partie a pour ambition de présenter la problématique qui entoure l'intégration des services sur des réseaux à haut débit de type ATM (Asynchronous Transfert Mode).

Ainsi, l'objectif du chapitre (I) est de présenter le mode de transfert asynchrone. Cette présentation a été construite pour mettre en évidence les besoins en modélisation que demandent le développement et la mise au point des protocoles nécessaires pour atteindre les principaux objectifs d'ATM. Ce chapitre permet ainsi d'aborder les problèmes qui se posent lors du choix entre les différentes techniques mises au point pour réaliser l'intégration des services. Il montre ainsi l'importance de la modélisation dans le développement et la validation des nouveaux protocoles réseaux.

Le chapitre (II) propose ensuite une présentation de la méthode de modélisation des réseaux à base de files d'attente. Les techniques de résolution les plus couramment utilisées sont présentées et c'est l'occasion de souligner les problèmes posés par ces méthodes lorsqu'elles sont utilisées pour la modélisation des réseaux ATM. Ainsi, ce chapitre permet de mettre en lumière les domaines où l'utilisation d'une architecture reconfigurable peut s'avérer utile.

Chapitre I

Le Mode de Transfert Asynchrone (ATM)

Ce chapitre expose les fondements et les objectifs du mode de transfert asynchrone (ATM). Le but n'est pas ici de présenter la technique de commutation ATM dans tous les détails, mais uniquement d'en exposer les grandes lignes. Le lecteur souhaitant plus de précisions peut se reporter à des ouvrages généraux sur les réseaux [Tan97, Rol95, BDM92, Héb85, GRI81, WV96], ou plus spécifiques aux techniques de commutation ATM [Cou91, AC91]. D'autre part, la mise en œuvre au niveau matériel d'un commutateur ATM peut être vue dans des ouvrages tels que [AM95, Puj92, Wei98].

Après avoir exposé les principes de base et les objectifs de l'ATM (section I.1), on verra que ces objectifs demandent une quantification et une qualification précise du trafic (section I.2). On exposera ensuite quelques uns des protocoles mis en œuvre pour délivrer les Qualités de Services (QdS) demandées par les utilisateurs des réseaux (section I.3). Ce dernier point nous permettra de voir que le choix, la mise au point et la validation de ces protocoles sont souvent difficiles et nécessitent un passage par la modélisation ou l'expérimentation (chapitre II).

I.1 Les principes de base

Après avoir passé en revue les raisons et les objectifs d'ATM, on présentera les principes de base de ce type de réseau : les cellules et les connexions.

I.1.a Objectifs d'ATM

Commutation de circuits

Dans la commutation de circuits on crée pour chaque communication un canal affecté à cette connexion, de telle sorte que toute l'information transite sur le même chemin (circuit) [GRI81]. Ce type de commutation est particulièrement adapté au transfert de données temps réel et donc au transfert de la voix. D'autre part, les techniques de multiplexage temporel permettent d'optimiser l'utilisation du canal en multiplexant plusieurs connexions. L'exemple type de ce genre de commutation est la commutation téléphonique.

Commutation de paquets

Dans la commutation de paquets, on traite des informations se présentant comme un bloc de données muni d'une adresse de destination. Les commutateurs stockent les messages, interprètent l'adresse, en déduisent la direction sortante et retransmettent le message dès que les circonstances le permettent. Dans chaque paquet, l'adresse est complète et permet l'acheminement de "bout en bout". Ce type de commutation est particulièrement adapté au transfert de données informatiques n'ayant pas de contrainte temps réel. L'exemple type de ce genre de commutation est la commutation utilisée pour le réseau internet (TCP/IP).

Intégration des services

L'objectif des réseaux ATM est l'intégration des services, c'est-à-dire l'utilisation d'un même réseau physique et donc d'une même technique de commutation pour transporter des trafics de nature différente. Ainsi, un tel réseau doit posséder les caractéristiques et les avantages de la commutation de paquets et ceux de la commutation de circuits (cf figure I.1), ceci dans l'objectif de faire coexister, sur le même réseau, des trafics ayant des exigences de Qualité de Service (QdS) très différentes. Il est probable que, au moins pendant quelques années, la plupart des artères ATM auront des débits de 150 Mbits/s. Ici c'est la technologie de commutation qui, par sa complexité, limite le débit.

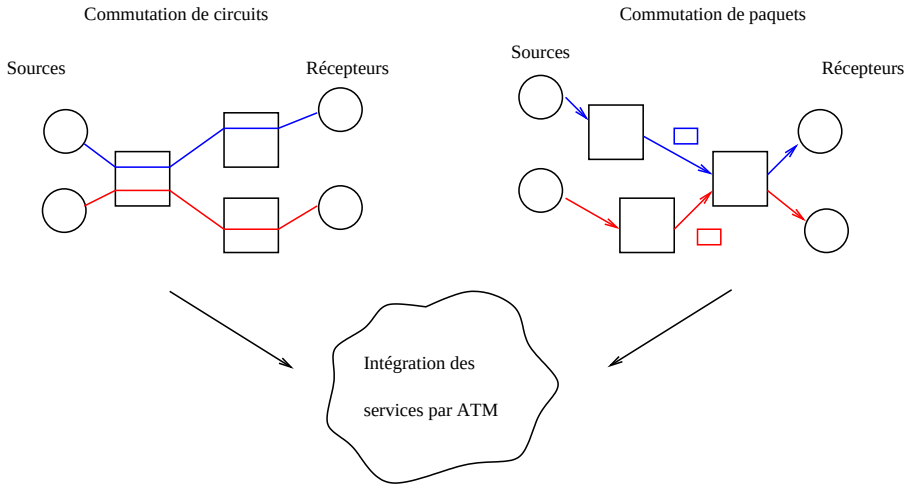

FIG. I.1: ATM est une technique de commutation qui tente d'intégrer les avantages de deux types de commutation différentes. La commutation de circuits d'une part et la commutation de paquets d'autre part.

I.1.b Les cellules : les paquets de l'ATM

La taille

Pour conserver les avantages de la commutation de paquets, il a été choisi de transporter l'information en la découpant en "cellules" (qui sont les paquets des réseaux ATM). La taille

de ces cellules a été très discutée. En effet, les services de téléphonie nécessitent des petits paquets pour éviter les phénomènes d'écho et de délai de transmission. D'un autre côté une grande taille est souhaitable pour éviter une surcharge trop importante. La taille des cellules a donc été normalisée à 53 octets. Quarante-huit octets sont utilisés pour coder les données, c'est la "charge utile". Les 5 octets restant sont réservés pour l'en-tête (cause de surcharge).

Le multiplexage

De la même manière que dans une méthode de connexion par paquets, les commutateurs stockent les cellules, interprètent l'adresse et retransmettent les cellules. Les informations nécessaires pour le routage sont codées dans les 5 octets d'en-tête. D'autre part, un même lien physique va être partagé entre plusieurs connexions (multiplexage). Cette technique permet d'utiliser au mieux les ressources du réseau. Par contre, comme dans la commutation de paquets, des conflits apparaissent du fait que plusieurs paquets risquent de demander en même temps l'accès au multiplexe sortant (cf figure I.2).

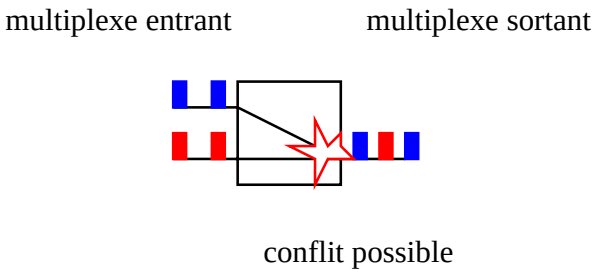

FIG. I.2: Les cellules sont multiplexées, elles peuvent rentrer en conflit. La mémoire du commutateur permet de mettre les cellules en attentes. C'est la taille de cette mémoire qui détermine les pertes.

I.1.c La connexion négociée

Mode connecté

Pour conserver les avantages de la commutation de circuit, on établit un circuit virtuel entre l'émetteur et le récepteur : c'est le mode connecté. Chaque communication est identifiée par un couple (VPI,VCI). Le VPI est le Virtual Path Identifier et le VCI le Virtual Circuit Identifier (souvent notés VP et VC). Ces identifiants transportés dans les en-têtes des cellules forment un numéro de voie logique. Vu la taille de l'en-tête, ces seuls numéros ne pourraient servir à établir une connexion de bout en bout comme dans la commutation par paquet. Ils n'ont donc de sens qu'entre deux nœuds de communication. Les commutateurs établissent ainsi une correspondance entre les numéros de voie logique sur les multiplex entrant et les multiplex sortant. Cet ensemble de correspondances marque un chemin fixe, mais dont le débit n'est pas réservé. La commutation ATM est donc une commutation de circuits.

La négociation

La mise en place de cette connexion se fait par l'intermédiaire d'une procédure de négociation entre la source et le réseau. Ce dernier accepte ou non d'établir la connexion en fonction de l'état du trafic sur le passage de la nouvelle connexion. Cette procédure d'établisement de la communication se nomme procédure de CAC (Call Admission Control). Une fois la connexion virtuelle établie, le réseau s'engage à respecter les QdS promises. Par exemple, un taux de perte très faible sans garantie de délai pour un transfert de fichier, ou une garantie forte de délai pour une transmission temps réel. D'un autre côté, la source s'engage elle aussi à respecter sa partie du contrat : ne pas émettre plus de cellules que prévues et ceci à la fréquence convenue.

I.1.d Conclusion

On remarque ici que garantir les QdS nécessite une quantification et une qualification précise du trafic pour permettre de définir une contractualisation non ambigüe. Ainsi, la négociation entre le réseau et la source se fait sur un certain nombre d'indices de QdS qui doivent être précis.

I.2 Quantification et Qualification du trafic

L'objectif de cette section est de présenter les principaux indices de QdS. Cet objectif nous impose de présenter au préalable les méthodes de police et de lissage du trafic (i.e : les méthodes pour contrôler le trafic).

En effet, les difficultés de caractériser correctement un trafic, et donc de définir des indices de QdS à négocier entre le réseau et l'utilisateur, ont débouchées sur des solutions qui forcent le trafic à adopter une structure particulière[Rol95, H98, For96, I.396].

I.2.a Caractérisation d'un trafic

Difficulté de caractérisation

La caractérisation du trafic est un problème difficile. En effet dans un système de communication, il se produit des phénomènes de rafales ou d'avalanches. Ils correspondent, par exemple, au transfert d'un fichier ou à une période de pointe du trafic. On appelle le débit crête, le débit engendré pendant la durée d'une rafale. Ce débit est donc supérieur au débit moyen observé. De plus ces rafales peuvent être corrélées. L'exemple type est le trafic généré par un algorithme de compression MPEG. La taille des blocs soumis dépend du cycle de l'algorithme de compression et de la nature des images compressées. Ce type de phénomène est difficile à caractériser par des indices simples.

Cette difficulté est mise en évidence par la multiplicité des modèles de source développés pour tenter de modéliser de manière réaliste des sources de trafic. Souvent des indices simples comme le débit ne suffisent pas pour caractériser correctement un trafic.

Ainsi, une multiplicité de modèles sont utilisés : processus geométrique, MMBP... Des modélisations plus complexes ont aussi été utilisées comme des processus browniens, ou des trafics de nature fractale.

Propriété d'auto-similarité

Les observations de trame ont mis en évidence des caractéristiques d'auto-similarités au cours du temps. Ces phénomènes sont particulièrement difficiles à reproduire. Ainsi on peut distinguer au moins trois "niveaux d'échelle" de temps dans une connexion ATM : le niveau appel ou connexion, le niveau rafale, et le niveau cellule. Le trafic observé à ces différents niveaux possède des propriétés propres à un instant donné, mais présente aussi des comportements similaires au cours du temps. Chacun à leurs niveaux d'échelle, les connexions, les rafales et les cellules arrivent par groupes puis subissent une baisse d'activité. De tels phénomènes sont particulièrement difficiles à reproduire à l'aide de modèles.

Pourtant il semble que l'on se dirige vers une modélisation plus réaliste de ce genre de processus en utilisant des processus de Markov ayant des propriétés d'auto-similarités [RLB96]. Ces modèles ont l'avantage d'utiliser des outils qui permettent une étude et une caractérisation des phénomènes mis en jeu.

I.2.b Contrôle et lissage du trafic

Police (à l'interface Utilisateur/Réseau)

Pour maîtriser ce problème de grande variabilité du trafic, les concepteurs d'ATM ont décidé d'introduire un contrôle strict des émissions des sources (police). Ainsi, on limite les rafales émises par l'utilisateur et on peut associer l'émission à une source pseudo-périodique.

Ce contrôle est effectué par l'algorithme du GCRA (Generic Cell Rate Algorithm)(figure I.3 et [I.396]). Le principe de cet algorithme est le suivant. A chaque connexion deux paramètres sont attribués. Une période d'émission T exprimée en temps-cellule et une tolérance de gigue τ (en temps-cellule) par rapport à cette période. Ceci permet d'établir l'heure théorique d'arrivée dans le réseau ($H_{théo}$) de chaque cellule. Si la cellule arrive (date t) avec un retard supérieur à la tolérance ($t > H_{théo} + \tau$), elle est alors considérée comme non conforme au contrat de trafic. La sanction prise par le réseau peut aller jusqu'à l'élimination de la cellule contrevenante ou, plus simplement, à un estampillage qui permettra de l'éliminer en cas de congestion.

Cet algorithme a la propriété de borner la longueur L des rafales (nombre de cellules collées) que la source émet dans le réseau :

$$L \leq 1 + \frac{\tau}{T-1}$$

Ainsi, la quantité $A(n)$ de cellules émises en n slots par une telle source est :

$$A(n) \leq min(n, L + \frac{n}{T}) \tag{I.1}$$

Dans un cadre plus général, on dit qu'une source de fluide émet sous la contrainte "Leaky Bucket" (LB) si la quantité d'information A(s,t) émise pendant la période]s,t] vérifie :

$$A(s,t) \leq \min((t-s)C, \sigma + \rho(t-s)), \forall t \geq s \geq 0$$

où σ est la sporadicité ($\sigma = L$ dans I.1), ρ le taux moyen d'émission ($\rho = \frac{1}{T}$) dans I.1) et C le taux crête d'émission (1 dans I.1).

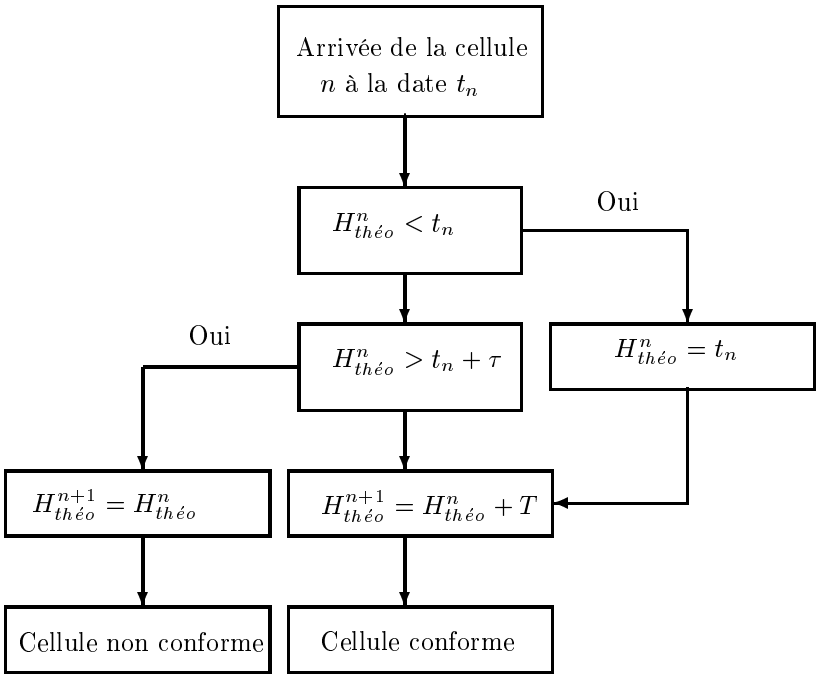

FIG. I.3: L'algorithme GCRA est un algorithme de contrôle, il permet de vérifier la conformité d'un trafic à l'entrée du réseau. Il a pour effet de limiter la taille des rafales émises dans le réseau, on obtient ainsi un trafic pseudo-périodique.

Lissage ou espacement (entre les commutateurs du réseaux)

D'autre part, le multiplexage des sources et le passage dans les commutateurs introduisent des perturbations du trafic. En effet, une cellule peut rester un certain temps en attente dans un commutateur avant d'être servie. Les commutateurs sont ainsi susceptibles de générer des rafales de cellules. Ce sont ces rafales qui risquent de remplir les mémoires des commutateurs situés en aval. Il faut donc éviter cette génération de rafales. De ce fait, des politiques de lissage ont été indroduites après le passage dans un commutateur.

Le lissage consiste simplement à supprimer les grumeaux de cellules en réémettant les cellules à la période T associée à chaque connexion.

Une fois ces conventions décidées, il devient plus facile de déterminer des indices précis sur lesquels une négociation saine peut s'établir. Les QdS sont alors clairement définies.

I.2.c Les indices négociés

Taux de pertes

Lors du transfert des données des pertes sont possibles. Ceci est dû au multiplexage et à la taille finie des tampons dans les commutateurs. Le taux de pertes est un des indices les plus

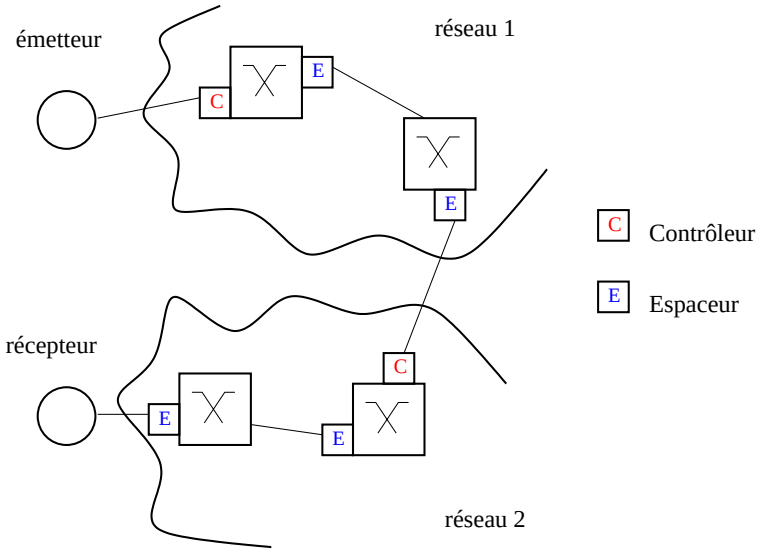

FIG. I.4: Les techniques de contrôle/espacement tentent de limiter les rafales de cellules qui sont les principales causes de saturation des mémoires et donc de pertes et de perturbation du trafic.

importants pour le transfert des fichiers informatiques. En effet, la perte d'une cellule peut rendre un fichier inutilisable. Dans ce cas, une demande de réémission doit être générée. Or, s'il y a eu perte, c'est que le réseau est déjà fortement sollicité. Réémettre un fichier risque de congestionner encore plus le réseau et entrainer des pertes sur d'autres communications qui, à leur tour, généreront des demandes de réemission. Le risque de saturation du réseau est alors important.

Débit

Le débit est aussi un "indice" important, il peut même être déterminant pour un certain nombre d'applications comme les applications temps réel. Le problème est de savoir si l'on doit parler de débit moyen, de débit crête ou de débit minimum garanti. La police présentée dans (I.2.b) permet de mettre l'utilisateur et le réseau d'accord sur une définition du débit. Le débit négocié est alors celui qui correspond à la période T du GCRA.

Gigue

Particulièrement importante pour les communications temps réel, la gigue peut être définie de manière intuitive comme étant la différence entre le flux soumis par l'utilisateur et le flux réel. L'apparition de la gigue peut être due à plusieurs facteurs. Ainsi, on distingue la gigue introduite par les couches protocolaires et la fragmentation (mise en forme de paquet de l'information) qui est plutôt d'origine logicielle et la gigue d'insertion dans le trafic et de multiplexage qui est d'origine plutôt "matérielle". En général, on parlera ici de la gigue de multiplexage.

Délai de transmission

Il s'agit du temps que va mettre une cellule pour parcourir tout le chemin entre l'émetteur et le récepteur. Pour certaines applications, ce délai n'est pas du tout critique. Ainsi, une transmission ne demandant pas de QdS temps réel (mail, transfert de fichier...) n'aura pas d'exigence sur le délai de transmission. Par contre, pour une application temps réel un délai trop important peut être strictement équivalent à une perte d'information.

I.2.d Conclusion

Encore une fois, on voit que le principal problème est de faire coexister sur un même réseau des exigences de QdS contradictoires. Pour utiliser au mieux les capacités du réseau tout en garantissant les QdS, on met en place des politiques de services "sélectives". Ces politiques doivent permettre au commutateur de choisir les cellules en attente de réémission en fonction des caractéristiques et des exigences des connexions auxquelles appartiennent ces cellules.

I.3 Protocoles garantissant les QdS

L'objectif est ici de présenter des politiques de services mises en place pour garantir les QdS. On présentera trois idées : la séparation des flux, la répartition équitable de la bande passante, et l'utilisation des notifications de congestion.

I.3.a File d'attente par connexion

Le partage des ressources d'un réseau par plusieurs clients peut être effectué de différentes manières. La plus simple est de stocker dans le même tampon les cellules appartenant aux différentes sessions et qui arrivent au même nœud du réseau, puis de les commuter dans un ordre FIFO (First In First Out). Cette solution simpliste a beaucoup d'inconvénients vis-à-vis des garanties sur les QdS. En effet, elle souffre de l'absence totale de séparation entre les connexions et le commutateur ne distingue pas les paquets pressés ou importants. Depuis le milieu des années 80, beaucoup de travaux ont pour but de trouver des disciplines de service plus sélectives et plus "intelligentes".

L'exemple de la figure (I.5) illustre le propos en prenant l'exemple de plusieurs connexions. C'est un cas typique où la politique de service FIFO est catastrophique pour l'une des connexions. On met ici en concurence trois connexions ayant un comportement périodique avec des exigences de QdS différentes. L'une de faible débit avec des contraintes temps réel (C) et les deux autres de fort débit (A, B) avec des exigences faibles sur les temps de transfert. Si la connexion faible débit a la malchance à chaque commutateur de passer après tout le monde, le résultat est un allongement important du délai de transmission.

L'idée la moins coûteuse et la plus simple pour résoudre ce problème consiste à séparer les différentes connexions en les stockant dans des tampons différents. Ainsi, les politiques de type "Round Robin" ("Round Robin", "Weighted Round Robin", "Hierarchical Round Robin",...) assurent la séparation physique des flux qui sont servis à tour de rôle selon un ordre prédéfini. Ces disciplines souffrent d'un manque de différenciation de traitement entre les flux et ne permettent pas de garantir un débit à l'un d'eux.

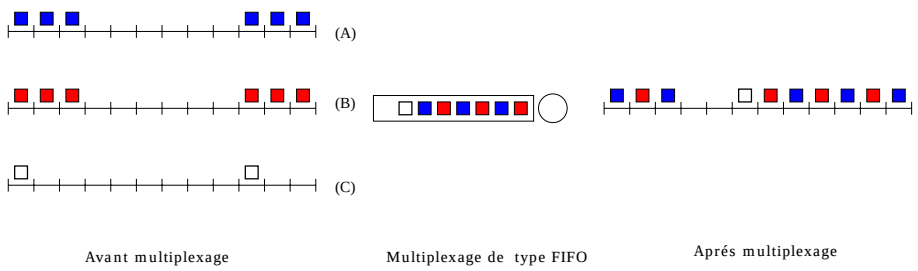

FIG. I.5: Avec une politique purement FIFO, des cellules "pressées" peuvent se retrouver servies après les autres.

I.3.b Les disciplines équitables (Fair Queuing)

C'est pour tenter de résoudre ces problèmes que la famille des politiques FQ (Fair Queuing) a été dévelopée. Ces politiques permettent de fournir des garanties sur les taux de service. En effet, l'ordre de service n'est pas prédéfini et la session à servir est choisie selon des critères qui tentent de tenir compte des QdS que l'on souhaite offrir à chacun des flux. Ces critères varient selon la version de la discipline.

De manière générale, un serveur FQ est un serveur qui est partagé entre plusieurs sessions dont chacune a réservé un taux de service négocié à l'avance. L'idée de base de ces disciplines est de tenter de se rapprocher au maximum d'un modèle fluide qui serait idéal dans le sens où le serveur est partagé exactement comme prévu entre chaque flux. Les disciplines FQ tentent donc à chaque instant de servir le paquet qui "l'éloigne" le moins de ce modèle idéal.

La méthode utilisée est la suivante : le serveur gère une fonction de temps virtuel (qui modélise le système fluide). Il affecte à chaque cellule une marque de début et une marque de fin de service virtuel. Les diciplines FQ se différencient entre elles par la définition du temps virtuel et par la politique de sélection du paquet à servir. En effet on peut choisir un paquet en fonction de sa date de fin de service ou en fonction de sa date de début. Ces disciplines sont décrites en détail dans le chapitre (VI).

Le problème est ensuite de pouvoir mesurer les différences entre ces méthodes et de savoir comment ces politiques influent sur les propriétés du trafic (cf. chapitre VI). La figure (I.3.b) montre l'évolution des politiques de service, celles-ci tendent à se complexifier pour mieux remplir leurs rôles.

Ces politiques de service visent à mieux gérer les flux et tentent ainsi de garantir une QdS sur les débits. Une autre méthode pour tenter de garantir ces QdS est d'essayer de prévoir et de résoudre les problèmes de congestion.

I.3.c Contrôle de congestion

Pour résoudre les problèmes de congestion on utilise des cellules spéciales, les cellules RM (Resource Managment) qui vont transmettre des informations sur l'état du réseau. Ces cellules sont aussi utilisées pour d'autres tâches comme par exemple l'établissement d'une connexion.

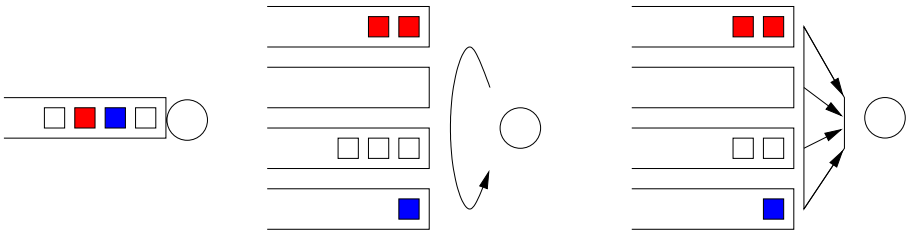

FIG. I.6: Pour garantir la bande passante à chaque connexion, on est passé de la discipline la plus simple (FiFo : First in First out), à des disciplines de plus en plus complexes. Ces disciplines séparent les flux pour mieux les servir.

La congestion se produit principalement aux nœuds du réseau, c'est-à-dire par débordement de files d'attentes dans les commutateurs. Trois causes de congestion sont généralement identifiées :

1. sur-allocation des ressources,
2. trafic non conforme par effet de multiplexage,
3. trafic non conforme par émission incontrôlée d'une source (ignorance ou malveillance).

Ces trois causes de congestion sont gérées par le réseau à diverses échelles de temps et par des méthodes différentes mais complémentaires :

- A l'échelle de temps du nouvel appel ($> 1s$), les ressources sont gérées de façon globale par le réseau. Cette gestion du réseau met en œuvre des fonctions de détection de pannes, de routage et de signalisation, et lors de l'établissement de chaque appel, effectue les procédures de contrôle d'admission (CAC). Ces procédures sont exécutées à l'accès au réseau. Leur rôle est d'accepter ou de refuser les connexions et de négocier les contrats de trafic. Ces méthodes visent à limiter les congestions ayant pour origine la cause 1.
- A l'échelle de temps d'aller-retour dans le réseau (typiquement $10ms$), deux types de mécanismes de contrôle de congestion sont proposés : des mécanismes de type binaire qui utilisent l'indication explicite de congestion et des mécanismes avec indication explicite de débit qui utilisent des cellules RM et permettent l'allocation dynamique de débit au cours d'une connexion. Ces méthodes visent à réagir au plus vite en cas de congestion.
- A l'échelle de temps de la cellule ATM ($2,8\mu s$ à $155Mbit/s$), les algorithmes de contrôle de trafic (filtrage à l'entrée des cellules non-conformes au contrat), et les algorithmes de lissage de trafic espacent les agrégats de cellules. Ces méthodes visent à limiter les causes 2 et 3 de congestion.

L'intéraction de ces différents niveaux de décission sont souvent durs à évaluer. Ainsi, l'impact d'un décission prise au niveau connexion sur le niveau celllule n'est pas évident.

I.3.d Conclusion

Le choix, la mise au point de ces protocoles nécessitent une évaluation préalable qui se fait grâce à la modélisation ou à l'expérimentation. Cette dernière, bien que coûteuse, est indispensable et constitue toujours la dernière étape avant l'implantation réelle. La modélisation est tout aussi indispensable, elle permet un test préalable pour sélectionner les protocoles les plus intéressants. Néanmoins, la rapidité des réseaux et la petitesse des cellules font que l'on se retrouve confronté a un très grand nombre d'évènements. Ce grand nombre d'évènements devient très vite un facteur limitant pour les méthodes classiques de résolution. Ces considérations font l'objet du chapitre (II).

Bibliographie

[AC91] P. Adam and J-P. Coudreuse. Atm et réseau : domaines d'application. *L'écho des recherches*, 145 :33–44, 1991.

[AM95] R.Y. Awdeh and H.T. Mouftah. Survey of ATM switch architectures. *Lecture Notes in Computer Science*, 27 :1567–1613, 1995.

[BDM92] Marc Boisseau, Michel Demange, and Jean-Marie Munier. *Réseaux haut débit.* Eyrolles, 1992.

[Cou91] J-P. Coudreuse. Atm : principes généraux. *L'écho des recherches*, 144 :7–19, 1991.

[For96] ATM Forum. *Traffic management specification.* Version4.0, Avril 1996.

[GRI81] GRINSEC. *La commutation électronique.* EYROLLES, 1981.

[H98] G. Hébuterne. Quality of service and related issues in broadband networks. In *Proceedings of ICT'98*, 1998.

[Héb85] Gérard Hébuterne. *Écoulement du trafic dans les autocommutateurs.* Masson, Paris, 1985.

[I.396] ITU-T I.371. *Traffic control and congestion control in B-ISDN.* Genève, octobre 1996.

[Puj92] G. Pujolle. Commutateurs ATM : Classification et architecture. *Technique et Science Informatique*, 11, 1 :11–29, 1992.

[RLB96] S. Robert and J.-Y. Le Boudec. Can self-similar traffic be modeled by markovian processes ? *Lecture Notes in Computer Science*, 1044, 1996.

[Rol95] Pierre Rolin. *Réseaux haut débit.* réseaux et télécommunications. Hermes, Paris, 1995.

[Tan97] A. Tanenbaum. *Réseaux.* Prentice Hall Inter-Editions, 1997.

[Wei98] D. Weil. *Architectures de Circuit pour la Commutation et la Gestion de Ressources en ATM.* PhD thesis, Institut National Polytechnique de Grenoble, 1998.

[WV96] Jean Walrand and Pravin Varaiya. *High-Performance Communication Networks.* Morgan Kaufmann Publishers, Inc., San Francisco, 1 edition, 1996.

Chapitre II

Méthodes classiques de modélisation de réseaux ATM

L'objectif de ce chapitre est de présenter la méthode classique de modélisation de réseaux ATM par files d'attente. L'approche présentée ici permet essentiellement de modéliser des commutateurs pour étudier des phénomènes au niveau de la cellule ATM. Seule la modélisation par files d'attente est abordée ici, malgré l'existence d'autres modèles comme les réseaux de Petri ou les réseaux d'automates. Toutes ces méthodes sont introduites dans un grand nombre d'ouvrages tels que [Héb85, PBL97, Mis87, Ros91]. Des modélisations de réseaux à un plus haut niveau, réalisées à base de graphes peuvent être trouvées dans [DA93] par exemple, mais elles ne seront pas traitées ici.

Dans un premier temps (section II.1), on présentera la théorie des files d'attente en temps discret (l'outil de base de la modélisation), puis les méthodes de résolution par simulation et par calcul (section II.2). Enfin, on verra les limites de ces méthodes et les techniques qui ont été développées pour contourner ces problèmes (section II.3).

II.1 Les files d'attente en temps discret

La théorie des files d'attentes est l'un des principaux outils utilisés pour la modélisation et l'analyse de performances des réseaux ATM. En effet, de manière générale, le chemin suivi par l'information peut se réduire au passage dans une série de files d'attente qui modélisent les buffers des commutateurs. D'autre part, les réseaux ATM se prêtent particulièrement bien à une modélisation par files d'attente en temps discret. En effet, toutes les cellules d'ATM sont de même taille. L'unité de temps sera donc la durée nécessaire au traitement d'une cellule. En général, cette unité de temps est désignée sous le nom de "slot". L'objectif de cette section est de présenter le formalisme des files d'attente. D'autres éléments importants sur les files d'attente sont disponibles (entre autres) dans [GP87, Woo94].

II.1.a Notations et définitions

Les files d'attente peuvent être considérées comme des objets mathématiques. Ainsi, une file d'attente est formellement caractérisée par 4 composantes :

- Le processus d'arrivée : c'est un processus stochastique qui décrit comment les clients arrivent dans le système. Les clients peuvent être des messages ou des paquets dans le cas des réseaux.

- Le processus de service : de manière générale, c'est un processus stochastique qui décrit le temps que passe le serveur à s'occuper d'un client (utilisation de ressources physiques). Dans le cas des réseaux de paquets il s'agit du temps nécessaire pour qu'un paquet soit transmis au commutateur suivant. Dans le cas des réseaux ATM, ce temps est donc déterministe de durée un slot.
- Le nombre de serveurs (nombre de ressources) : certains systèmes peuvent comporter une "salle d'attente" et plusieurs serveurs. Pour les réseaux ATM, on a plutôt un serveur pour plusieurs zones d'attente (section I.3).
- La taille de la salle d'attente : la limite maximum du nombre de clients qui peuvent rester en attente. En réseau il s'agit de la taille des tampons d'un commutateur, c'est-à-dire du nombre de paquets qui peuvent être mis en attente.

Ainsi on définit complètement une file avec ces quatre composants. Par exemple : $A/B/c/k$ définit une file où A décrit le processus d'arrivée, B le processus de service, c donne le nombre de serveurs et k le nombre maximum de clients pouvant être mis en attente. Clairement c et k ne peuvent prendre que des valeurs positives et éventuellement infinies. Lorsque k est infini, il est souvent omis.

Les valeurs les plus courantes pour A (resp B) sont :

- D pour Déterministe, c'est-à-dire que les temps d'inter-arrivées (resp. les temps de services) sont constants.
- M pour Memoryless, c'est-à-dire sans mémoire. Les temps d'inter-arrivées (resp. de services) sont de distribution exponentielle. Dans ce cas, il s'agit de temps continu.
- G pour Général, ce qui signifie que l'on ne précise pas les processus.
- Geo pour Géométrique, c'est la version discrète de M. Il s'agit d'un processus sans mémoire avec une distribution géométrique entre chaque événement.

Nous utiliserons une extension commune [Woo94] de cette notation. Cette extension permet de pouvoir traiter le cas des arrivées et des départs multiples dans un slot. Ainsi, nous utiliserons a_n (resp. d_n) pour noter le nombre d'arrivées (resp. de départ) durant l'unité de temps n.

Exemples :

- $M/M/1/k$: décrit un système où les arrivées sont exponentielles, de même que les départs, avec un serveur et une file de taille (capacité) k.
- $Geo/Geo/1/k$: c'est la version discrète de la file $M/M/1/k$. Dans ce cas, les arrivées et les départs suivent un processus geométrique.
- $Geo^{a_n}/Geo/1/k$: à chaque slot, un nombre aléatoire a_n de clients entre dans le système.

II.1.b Le temps discret

Pour les systèmes discrets, il faut choisir entre deux modèles de traitement de la simultanéité. Contrairement au temps continu - où un intervalle de temps, si faible soit-il, sépare toujours deux événements distincts - en temps discret, les arrivées et les départs peuvent avoir lieu aux mêmes instants. Dans le cas discret, il est important de considérer la façon dont on gère la file d'attente. Les deux modèles possibles de traitement de la simultanéité sont :

- soit, on fait sortir les cellules dont l'émission est terminée avant de laisser entrer un nouveau groupe de cellules (option "DF", Departure First).

– soit, les cellules entrantes sont admises dans le tampon avant le départ de la cellule émise (option "AF", Arrival First).

Bien entendu, le choix de l'option a une influence sur les résultats observés. En effet si l'on applique l'option "DF", les cellules entrantes disposent (éventuellement) d'une place supplémentaire dans le tampon par rapport à l'option "AF". Les indices de performance observés peuvent donc être différents [GH92].

Couramment le modèle AF est le plus utilisé pour ses bonnes propriétés mathématiques. Dans la suite, c'est ce modèle qui sera utilisé sauf indication particulière.

Notons a_n le nombre d'arrivées durant le slot n, et d_n le nombre de départs dans le slot n, y_n la longueur de la file au début du slot n avec y_0 arbitraire (voir figure II.1). On a alors :

$$y_{n+1} = y_n + a_n - d_{n+1}$$

La longueur de la file juste après les arrivées, noté x_n est donnée par :

$$x_n = y_n + a_n$$

D'où l'on obtient :

$$x_{n+1} = x_n + a_{n+1} - d_{n+1}$$

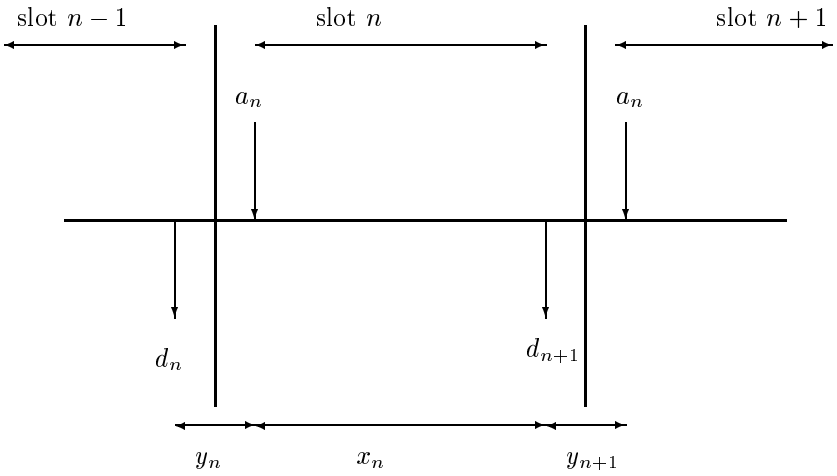

FIG. II.1: l'option "les arrivées d'abord" (AF).

II.2 Méthodes de traitement : simulations et approches analytiques

On se propose ici de montrer comment le formalisme des files d'attente peut être utilisé pour l'étude d'indices de QdS. Un certain nombre d'outils analytiques et de méthodes de

simulation ont été développés pour permettre l'étude des réseaux de files d'attente. Typiquement on cherche à calculer le temps moyen qu'un client passe dans le système. Pour les réseaux cet indice permet d'estimer le retard que prend une cellule en passant dans un commutateur. Si k (nombre maximum de client en attente) est fini la probabilité qu'un client soit rejeté par le système est un indice important. Pour les réseaux ceci permet d'évaluer le taux de perte dans chaque commutateur. On peu aussi s'intéresser au taux d'utilisation du serveur. Ceci pour savoir si le serveur est bien dimensionné.

II.2.a Stationnarité et ergodicité

Les différentes méthodes de traitement ont pour objectif l'étude du régime stationnaire du système. Le régime stationnaire est un régime d'équilibre dans lequel les valeurs observées lors d'une simulation se situent dans un voisinage de leur valeur limite. Ainsi, aprés une période de lancement (régime transitoire) le système entre dans sa phase stationnaire. La notion de stationnarité affirme qu'on peut observer un processus stochastique n'importe quand dans le temps. L'autre notion importante est la notion d'ergodicité qui affirme, elle, que plus la période d'observation est longue plus la connaissance que l'on a du processus est fiable.

Les meilleures hypothèses pour une simulation sont que le processus étudié est à la fois stationnaire et ergodique. Dans ce cas, il est possible d'étudier le processus au travers de l'étude d'une seule trajectoire de ce système.

Ces notions de stationnarité et d'ergodicité présentées ici de manière très intuitive se définissent en fait de manière mathématique [J.L80, Kel79, Ste94].

II.2.b Simulation par événements discrets : Qnap, SES Workbench

On considère généralement qu'il existe deux types de simulateurs à évènements discrets. Des simulateurs "orientés événements" et des simulateurs "orientés processus". Pour plus de renseignements sur la simulation par événements discrets, on peut se reporter à [Ros91, ED96, J.L80].

Simulateurs orientés événements

Les systèmes orientés évènements constituent l'approche classique dans laquelle le système simulé est modélisé événement par évènement. On décrit un évènement par son heure d'activation, c'est-à-dire l'instant où l'évènement est censé se produire et par une procédure d'activation qui comprend les actions qui doivent être effectuées à ce moment. Le système possède une horloge interne qui donne en tout temps l'heure du système de simulation. C'est en principe l'heure d'activation de l'évènement courant. Le système gère aussi un échéancier, réalisé comme une liste structurée, où les évènements sont rangés en fonction de leur heure d'activation.

Ainsi le moteur central du simulateur extrait, à chaque pas, l'évènement situé en tête de l'échéancier et provoque la liste des actions qui lui sont attachées. Ces exécutions peuvent provoquer la création, la révocation ou le report d'évènements, et donc une mise à jour de l'échéancier. L'horloge du système n'est pas modifiée pendant l'exécution de l'évènement courant, mais seulement avec le traitement d'un nouvel évènement.

Simulateurs orientés processus

Chaque activité de la simulation est vue à un niveau logique comme l'activité d'un processus. Le modèle simulé est alors décrit comme un ensemble de processus progressant parallèlement dans le temps. Chaque processus représente une entité qui peut évoluer dans le temps au contraire des autres entités de la simulation qui demeurent passives. Le noyau de synchronisation fournit alors des ordres d'activation de processus. L'échéancier est constitué des processus et de la date à laquelle ils doivent être activés. A chaque pas de simulation, le processus en tête de l'échéancier est activé et l'horloge centrale du système est avancée à l'heure d'activation de ce processus.

Simulation dirigée par une horloge

On trouve parfois [J.L80] un troisième type dit "simulation dirigée par une horloge". Dans ce modèle, on définit une entité de temps appropriée au problème et on dispose d'une horloge centrale qui progresse par pas d'unité de temps. A chaque incrémentation de l'horloge, on explore la liste des événements pour voir si l'un d'eux apparaît à cette date. Cette exploration pouvant s'avérer infructueuse, il est particulièrement important de bien choisir l'unité de temps. Ce type de simulateur est très peu utilisé, en effet, les approches par évènement ou par processus permettent de ne pas simuler les intervalles de temps où il ne se passe rien. C'est ce type de simulation qui sera implanté à l'aide de ressources matérielles (voir section III.3)

Exemple

Il existe de nombreux simulateurs de files d'attente (Qnap, Network Simulator...). Parmis eux, SES/Workbench est un exemple d'outil de simulation graphique à évènements discrets permettant d'étudier les systèmes modélisés sous forme de réseaux de files d'attente. Les principaux champs d'application de ce genre d'outil sont la simulation des systèmes informatiques et des réseaux de communication.

Ce type d'outil met à disposition de l'utilisateur un certain nombre de "nœuds" de base tels que par exemple (liste non exhaustive) :

- les sources qui créent les clients injectés dans le réseau de file d'attente ;
- les puits qui absorbent les clients ;
- les nœuds de service constitués de file et de serveur.

Les caractéristiques de chaque nœud sont ensuite décrites en remplissant des cases dans des "formulaires de spécification". Si cela ne suffit pas, le comportement du nœud peut être précisé au moyen d'une "méthode utilisateur" constituée d'une séquence d'instructions en langage C.

Ce genre d'outil permet de définir un fichier de sortie et les informations que l'on souhaite y voir figurer : statistique standard portant sur les temps de réponse et l'occupation des files d'attente ou résultats calculés par l'utilisateur.

II.2.c Approche analytique : les chaînes de Markov

Les chaînes de Markov sont un des outils les plus utiles pour obtenir des résultats de manière analytique sur les files d'attente. De manière générale, un réseau de file d'attente

peut être modélisé par un automate en temps discret. Ces automates peuvent se traduire par une chaîne de Markov. La matrice de transition permet de trouver la loi stationnaire de l'automate par des calculs numériques [Ste94].

Ce paragraphe montre au travers d'un exemple trés simple (la file $Geo/Geo/1$) la manière dont ces techniques peuvent être utilisées. C'est aussi cette file qui est utilisée comme exemple dans la section (IV.3) pour montrer comment utiliser des ressources matérielles. Dans la file $Geo/Geo/1$, les arrivées sont indépendantes et distribuées selon un processus de Bernouilli, avec $a_n \in \{1, 0\}$ et une capacité infinie.

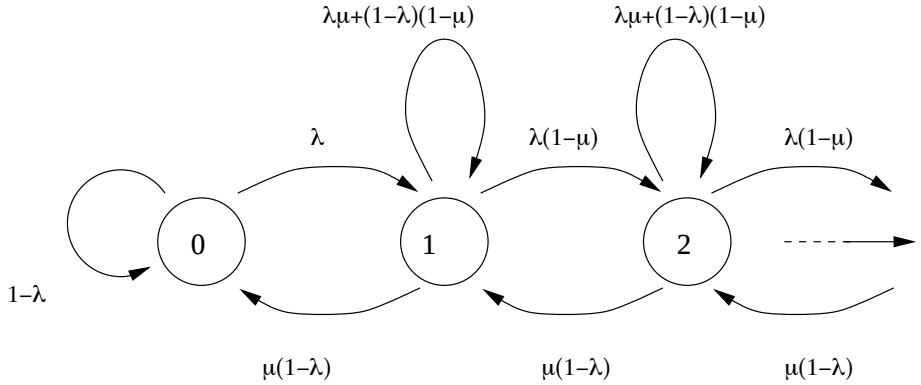

FIG. II.2: Le diagramme d'état pour une file $Geo/Geo/1$.

Notons λ la probabilité d'une arrivée dans un slot et μ la probabilité d'un départ. Le diagramme des états et des transitions est donné par la figure (II.2.c). Le processus de l'occupation de la file $\{x_n, n = 0, 1, 2, ...\}$ est une chaîne de Markov. Notons π_n la probabilité que le système se trouve dans l'état n. Il est intuitif que si $\lambda > \mu$ le nombre de clients présents dans la file va croître jusqu'à l'infini. Le cas où $\lambda = \mu$ est un cas pathologique de par son instabilité, on ne l'étudiera donc pas ici.

Par contre sous l'hypothèse que $\lambda < \mu$ on peut exprimer les équations décrivant le système.

$$\pi_0 = \pi_0(1 - \lambda) + \pi_1\mu(1 - \lambda) \tag{II.1}$$

$$\pi_1 = \pi_0\lambda + \pi_1[\lambda\mu + (1 - \lambda)(1 - \mu)] + \pi_2\mu(1 - \lambda) \tag{II.2}$$

Ou, de manière plus générale :

$$\pi_k = \pi_{k-1}\lambda(1 - \mu) + \pi_k[\lambda\mu + (1 - \lambda)(1 - \mu)] + \pi_{k+1}\mu(1 - \lambda) \tag{II.3}$$

En notant :

$$\gamma = \frac{\lambda(1 - \mu)}{\mu(1 - \lambda)}$$

On obtient :

$$\pi_k = \frac{(1-\gamma)\gamma^k}{1-\mu(1-\gamma)} \tag{II.4}$$

Il faut noter que quand le slot devient très petit et que $\lambda \to 0$ et $\mu \to 0$, de telle manière que $\lambda/\mu \to \rho$, alors II.4 devient :

$$\pi_k = (1-\rho)\rho^k \tag{II.5}$$

Ceci est un résultat classique. C'est l'occupation de la file d'attente pour une file $M/M/1$. Cette formule permet de calculer le taux "d'occupation" du serveur (π_0).

On peut également remarquer que le processus est réversible [Kel79], avec la capacité infinie de la file cela assure que le processus de sortie est géométrique.

II.3 Les limitations des méthodes classiques

L'objectif est d'énoncer les principaux problèmes rencontrés par ces méthodes classiques dans leur utilisation pour la résolution de problèmes en ATM. On présentera aussi de manière rapide un certain nombre de méthodes qui ont été développées pour les résoudre.

II.3.a Les problèmes

Modélisation des sources

L'un des principaux problèmes est d'arriver à avoir une bonne modélisation d'un trafic difficilement caractérisable (cf I.2). En effet, les performances observées par tels ou tels protocoles vont fortement dépendre des caractéristiques du trafic.

Le nombre de manières de modéliser le trafic est infini. Nous en proposons ici quelques unes parmi les plus fréquemment utilisées. On se propose de se placer dans le cadre de la modélisation d'un commutateur ATM. Les modèles de trafic présentés intègrent aussi des choix sur les probabilités de routage.

- Trafic uniforme : dans ce modèle, chaque source est modélisée par un processus de Bernouilli de paramètre ρ. En d'autres termes, à chaque slot une cellule est émise avec une probabilité ρ et la source reste silencieuse avec la probabilité $1-\rho$. Chaque cellule arrivant peut choisir son chemin entre toutes les voies possibles de manière uniforme. Ainsi, s'il existe N voies de sortie pour la cellule, alors chacune de ces destinations peut être empruntée avec la probabilité $1/N$. Adopter ce modèle de trafic comme étant une bonne modélisation du trafic réel peut conduire à obtenir des résultats optimistes. En effet, le trafic généré par un tel modèle est un trafic plus lisse qu'un trafic non contraint. Par contre, si l'on compare ce trafic avec le trafic contraint par le GCRA, alors on obtiendra des résultats pessimistes, puisque le trafic contraint est plus lisse que le trafic uniforme. L'avantage de ce modèle est, entres autres, sa simplicité théorique qui permet d'obtenir un grand nombre de résutats analytiques. Ce trafic sera utilisé dans le chapitre (V) qui traite de la modélisation de réseaux multi-étages.
- Par rafales : de nombreux modèles tentent de décrire des sources qui génèrent des rafales. Ces sources fonctionnent à leur débit crête pendant un certain temps puis restent silencieuses. Une des méthodes les plus populaires consiste à utiliser le modèle

des sources On/Off. Ces sources alternent entre une période actives et une période inactive. La longueur des périodes actives (en slot) est distribuée de manière géométrique de paramètre a, et les périodes de silences de paramètre s. On note par r le taux d'émission lorsque la source est dans l'état actif. La charge ρ du trafic est alors donnée par $ra/(a+s)$.
La probabilité qu'une période active dure i slots est donnée par :

$$B(i) = a(1-a)^{i-1}, 1 \leq i.$$

Dans cette équation, on suppose que la période active dure au moins un slot. La longueur moyenne d'une rafale est donnée par :

$$A = \sum_{i=1}^{\infty} iB(i) = \frac{r}{a}$$

De la même manière, on calcule la distribution de la durée d'une période de silence :

$$I(i) = s(1-s)^{i}, 0 \leq i.$$

Et la longeur moyenne d'une période de silence :

$$B = \sum_{i=0}^{\infty} iI(i) = \frac{1-s}{s}$$

Dans ce cas, les cellules se présentant à un port sont toujours routées uniformément vers les différents ports de sortie. Ce type de modélisation permet d'obtenir un trafic plus proche de la réalité puisqu'il va générer des rafales.

– Hot-spot trafic : ceci se réfère à la situation où plusieurs ports d'entrée veulent communiquer avec un même port de sortie. Cette situation arrive classiquement quand de très nombreuses personnes veulent accéder à des sites très populaires. A un niveau plus local, ceci peut correspondre à des ordinateurs accédant a un serveur de fichier, de nom, ou toute autre information partagée. Dans ce cas, si l'on considère que le processus d'arrivée des cellules est le même que dans le cas du trafic uniforme, l'un des ports de sortie est plus populaire que les autres (hot spot). Notons h la proportion des cellules dirigées vers le hot-spot. Dans ce cas (avec ρ comme charge d'entrée) la charge sur le port de hot-spot notée ρ' est donnée par $\rho' = \rho h + \rho(1-h)$. Ainsi, ρh cellules sont dirigées vers le point hot-spot et $\rho(1-h)$ sont réparties uniformement sur toutes les sorties. Un trafic de ce type sera utilisé pour l'étude des réseaux multi-étages au chapitre (V).

Ces modèles de trafic sont simples, mais représentent mal la réalité et la nature d'un trafic réel. Plus on complexifie les sources, plus il est difficile d'obtenir des méthodes simples et efficaces, et ceci aussi bien d'un point de vue analytique que du point de vue de la simulation.

Les sources MMPP (Markov Modulated Poisson Process) [FMH93] et les sources auto-similaires [RLB96, RLB98] sont beaucoup plus réalistes. Cependant l'utilisation de telles sources compléxifie trés rapidement les modèles étudiés.

Le problème de la modélisation du trafic est un problème inérent à la modélisation. Pour contourner ce problème certains préfèrent travailler sur des traces de trafic réel. Cette approche risque cependant de conduire à une perte de généralité puisque l'on travaille alors sur des cas particuliers.

Politiques de service complexes

Un autre facteur de difficulté vient de la complexité croissante des politiques de service (section I.3). En effet, plus les politiques de service sont complexes plus les modèles mathématiques présentés en (II.1) s'éloignent de la réalité. Il devient illusoire, voire impossible de simuler une politique de Fair Queuing avec un processus aléatoire tel que ceux présentés dans la section (II.1).

D'un autre côté, au niveau de la simulation, les politiques de service complexes rendent les simulateurs logiciels très lourds. Ils deviennent alors de plus en plus gourmands en temps de simulation (cf A).

Recherche d'évènements rares

L'étude des réseaux ATM revient souvent à l'étude de ce que l'on nomme des évènements rares. Ainsi, on estime généralement qu'une estimation correcte d'un taux de perte fixé à 10^{-9} va nécessiter la simulation de 10^{12} évènements. Or, ces évènements qui sont rares à l'échelle de la cellule ne sont pas si exceptionnels à notre échelle. Ainsi, en prenant un lien à 155 Mbit/s et un taux de perte de l'ordre de 10^{-9} on observera une perte par heure. Il est donc important de pouvoir étudier ces évènements.

Ce grand nombre d'évènements à simuler est un gros problème pour les simulateurs logiciels. L'obtention d'une estimation correcte d'un taux de perte réaliste n'est pas envisageable avec de tels outils. Ainsi, Dimitrios Stiliadis dans [Sti96] estime à plusieurs jours le temps de simulation nécessaire pour estimer correctement un taux de pertes de l'ordre de 10^{-9}. Ce propos est largement illustré par les résultats d'expériences présentées dans l'annexe A.

Du point de vue analytique, le problème vient plus du fait que les systèmes à étudier comportent un très grand nombre d'états. Ainsi, si l'on note K la capacité des files d'attente, et q leur nombre, le nombre d'états du sytème est $(K+2)^q$. Ainsi, si l'on modélise un commutateur 16×16 (un petit commutateur) par 16 files d'attente de capacité 128 cellules, le nombre d'etats possibles du système est de l'ordre de 10^{33}. Calculer l'état stationnaire d'une telle chaîne de Markov sans la simplifier est illusoire.

Interraction entre les différentes échelles de temps

D'autre part, la section (I.3) a montré que différents niveaux de protocoles peuvent intéragir les uns avec les autres. Ainsi, il est difficile de mesurer l'impact d'une décision prise à l'échelle de temps des connexions sur le niveau des cellules.

Encore une fois, le grand nombre d'évènement à simuler et le nombre d'états des systèmes considérés rendent ce type d'étude trés complexes.

Passage à taille réelle

La plupart des méthodes classiques de simulation de réseaux ont aussi un gros problème de passage à l'échelle. En effet, un commutateur doit gérer des centaines de VC et il est intéressant de savoir comment les algorithmes élaborés vont se comporter dans cet environement. Des simulations comportant tant de VC sont innacessibles par des approches traditionnelles. De fait, la plupart des études sur les protocoles des réseaux ATM se teste sur un nombre très limité de connexions (typiquement moins de dix).

Ainsi, les problèmes posés par le passage à taille réelle sont ignorés, au risque de devoir réaliser des modifications coûteuses sur les commutateurs.

II.3.b Les autres méthodes et leurs limites

Pour résoudre ces problèmes un certain nombre de pistes ont été explorées :

- Ainsi les méthodes de résolutions par simulation peuvent être améliorer par un certain nombre de techniques comme par exemple la réduction de variance ou la P-simulation [Pel92, PBL97]. Il est aussi possible de réaliser des simulations sur des machines parallèles [PEF97, PBF98].
- Pour les méthodes de résolution numériques des techniques d'encadrement stockastique des chaînes de Markov [Tru95, Abu98] ou l'étude des "Poisson clumbing process" [Ald89, GRSS94] ont été testées.

Cepandant la plupart de ces méthodes permettent difficilement d'évaluer correctement des événements rares sur des systèmes complexes comportant un grand nombre d'états, ni d'éffectuer un passage à l'echelle avec la gestion de nombreuses connexions.

II.4 Conclusion

La modélisation de réseaux à haut débit pour l'étude de QdS pose donc un certain nombre de problèmes (événements rares, passage à l'échelle...). C'est pourquoi des techniques de simulation ou de calcul ad-hoc sont souvent utilisées.

L'une de ces possibilités, pour tenter de résoudre ces problèmes, consiste à utiliser des ressources matérielles pour effectuer des simulations de manière efficace. La deuxième partie a pour objectif la présentation de la manière dont on peut utiliser des ressources reconfigurables pour simuler des réseaux de files d'attente.

Bibliographie

[Abu98] O. Abuamsha. *Application des méthodes de la comparaison stochastique pour l'analyse des disciplines "Fair-Queueing"*. PhD thesis, PRiSM, Université de Versailles, 1998.

[Ald89] D. Aldous. *Probability Approximations via the Poisson Clumbing Heuristic*. Springer-Verlag, 1989.

[DA93] Alan Dolan and Joan Aldous. *Networks and Algorithms*. Johnn Wiley & Sons, 1993.

[ED96] Pierre-Jean Erard and Pontien Déguénon. *Simulation par événements discrets*. Presses polytechniques et universitaires romandes, 1996.

[FMH93] W. Fischer and K. Meier-Hellstern. The markov-modulated poisson process (MMPP) cookbook. *Performance Evaluation North-Holland*, 18, 2 :149–171, 1993.

[GH92] A. Gravey and G. Hébuterne. Simultaneity in discrete-time single server queues with Bernouilli inputs. *Performance Evaluation North-Holland*, 14 :123–131, 1992.

[GP87] E. Gelenbe and G. Pujolle. *Introduction to Queueing Networks*. John Wiley & Sons Limited, 1987.

[GRSS94] F. Guillemin, G. Rubino, B. Sericola, and A. Simonian. Transient characteristics of an $M/M/\infty$ system applied to statistical multiplexxing on an ATM link. Publication interne 874, IRISA, october 1994.

[Héb85] G. Hébuterne. *Écoulement du trafic dans les autocommutateurs*. Masson, Institut National de Télécommunications, 1985.

[J.L80] J.Leroudier. *La simulation à événement discrets*. Édition hommes et techniques., 1980.

[Kel79] F. P. Kelly. *Reversibility and Stochastic Networks*. John Wiley and Sons, Chichester, 1979.

[Mis87] Schwartz Mischa. *Telecomunication Networks. Protocols, Modeling and Analysis*. Addison-Wesley Publishing Company, November 1987.

[PBF98] C. D. Pham, H. Brunst, and S. Fdida. Conservative simulation of load-balanced routing in a large ATM network model. In *Proceedings of the 12th Workshop on Parallel and Distributed Simulation (PADS-98)*, pages 142–153, Los Alamitos, May 26–29 1998. IEEE Computer Society.

[PBL97] Jean Pellaumail, Pierre Boyer, and Patrice Leguesdron. *Autoroutes ATM*. HERMES, 1997.

[PEF97] C. D. Pham, J. Essmeyer, and S. Fdida. Simulation of a routing algorithm using distributed simulation techniques. *Lecture Notes in Computer Science*, 1300 :1001−? ?, 1997.

[Pel92] Jean Pellaumail. *Graphes, Simulation, L-matrices, application aux files d'attente.* Hermes, 1992.

[RLB96] S. Robert and J.-Y. Le Boudec. Can self-similar traffic be modeled by markovian processes ? *Lecture Notes in Computer Science*, 1044, 1996.

[RLB98] S. Robert and J.-Y. Le Boudec. Properties of a new class of models designed for self-similar traffic. In *Proceedings of the First Workshop on ATM Traffic Management*, Paris France, Descembre 1998. IFIP.

[Ros91] Sheldon M. Ross. *A Course in Simulation.* Mamillan Publishing Company, University of california, Berkeley, 1991.

[Ste94] William J. Steward. *Introduction to the Numerical Solution of Markov Chains.* Princeton University Press, 1994.

[Sti96] Dimitrios Stiliadis. *Traffic Scheduling in Packet-Switched Networks : Analysis, Design, and Implementation.* PhD thesis, University of California Santa Cruz, 1996.

[Tru95] L. Truffet. *Méthodes de Calcul de Bornes Stochastiques sur des Modèles de Systèmes et de Réseaux.* PhD thesis, Université Paris VI, 1995.

[Woo94] Michael E. Woodward. *Communication and Computer Networks, Modelling with discrete-time queues.* IEEE Computer Society Press, 1994.

Deuxième partie

Modéliser des protocoles à l'aide de matériel

La première partie a permis de mettre en évidence certaines carences des méthodes d'évaluation de performances utilisés pour travailler sur les réseaux à haut débit. l'objectif de cette deuxième partie est de proposer une nouvelle méthode de simulation des réseaux de files d'attente dans l'espoir de résoudre un certain nombre de ces problèmes. Il s'agit donc de décrire en détail la démarche proposée ainsi que les outils qui peuvent être utilisés pour sa mise en œuvre.

Le chapitre (III) tente à travers un exemple, de montrer comment un algorithme peut être effectué à l'aide de matériels (portes logiques, registres, mémoires). On tente de dégager une méthode générale de modélisation à l'aide de matériels. On montre aussi comment les modèles exposés dans la première partie peuvent se matérialiser. On montre comment on a choisi de décrire les files d'attente en temps discret à l'aide de ressources matérielles. On énnumère donc les briques de base qui ont été réalisées et qui vont permettre la modélisation de protocoles réseaux et être utilisées dans la troisième partie pour simuler ces protocoles.

Le chapitre (IV) dresse un petit panorama des architectures reconfigurables, l'usage qui en est fait usuellement et les utilisations émergentes de celles-ci. Ce sera l'occasion de présenter en détail l'architecture matérielle de la machine Sim-express. On décrira aussi la chaîne logicielle qui rend possible la configuration de cette machine et les outils qui permettent de la piloter.

On expose ainsi les outils utilisés tout en montrant les "configurations" de base qui ont été réalisées pour la simulation des files d'attente. Cette partie permettra de conclure sur la faisabilité et l'intérêt de tenter l'expérience.

Chapitre III

Utilisation de ressources matérielles pour la modélisation de systèmes en temps discret

Ce chapitre montre comment utiliser des ressources matérielles [Zak85] pour implémenter des algorithmes [CLR94, FLP98, KR90]. La discussion s'attachera particulièrement à décrire comment "matérialiser" les méthodes et les modèles utilisés pour la simulation des réseaux à haut débit qui ont été présentés au chapitre (II).

La section (III.1) propose une classification de ce que l'on nomme usuellement "matériel" (transistors, portes logiques, registres, mémoires). Cette classification permettra de mesurer la complexité d'une implantation utilisant des ressources matérielles. La section (III.2) montre, au travers d'un exemple, les différentes possibilités offertes par l'utilisation de matériels pour la réalisation d'un algorithme. Enfin, la section (III.3) détaillera les différentes solutions possibles pour l'implantation des files d'attente, en tant qu'outil mathématique, telles qu'elles ont été présentées dans le chapitre (II).

III.1 Ressources matérielles et mesure de complexité

III.1.a Classification des différentes ressources matérielles et complexité en espace

Tous les circuits intégrés sont construits à partir de transistors. Ces transistors permettent à leur tour de construire des portes logiques, avec lesquelles on peut réaliser des fonctions plus complexes. Ainsi, ces portes logiques sont regroupées et utilisées de différentes manières sur un même circuit, elles permettent de construire des opérateurs, des points mémoires...

En général, on différencie ces regroupements de portes logiques en trois grandes catégories. Ces catégories correspondent à des utilisations fonctionnellement différentes, mais aussi à des réalités physiques différentes.

On se propose donc de mesurer la complexité matérielle d'un algorithme en fonction de ces trois catégories de matériel : la logique, les registres, la mémoire. Le tout pouvant éventuellement, si on le juge intéressant, se ramener à un nombre de transistors.

Logique

On nommera "couche logique" ou "glu logique", un ensemble de portes logiques ayant un comportement uniquement combinatoire. Si l'une des entrées de la couche logique change, sa nouvelle valeur se propage au travers des portes logiques qui composent la couche. Les sorties sont ainsi mises à jour au bout d'un certain temps : le temps de stabilisation. Souvent ce temps de stabilisation est très petit et on considère qu'il est instantané. De manière plus générale, la période de l'horloge est choisie plus grande que les temps de stabilisation des différentes couches logiques du circuit.

Registre

On nommera "registre" un point mémoire isolé. En général, ce point mémoire sert à stocker le résultat d'un calcul d'une fonction logique sur un front d'horloge. L'association d'un registre et d'une couche logique permet de construire un automate. Le registre donne l'état de l'automate et l'état suivant est calculé par la couche logique. Sur le front montant de l'horloge, le registre prend la valeur calculée par la couche logique. Pour nous, un registre est donc un point mémoire isolé auquel on accède sans adressage.

Évidement, un registre est construit à partir de portes logiques elles mêmes constituées de transistors. Mais l'utilisation de ces portes logiques en tant que registres est complètement différente de leur utilisation dans une couche logique. Il est donc intéressant, lors d'une mesure de complexité, de comptabiliser les registres en tant que ressources différentes.

Mémoire

De nombreux points mémoires peuvent être regroupés autour d'un seul bus permettant d'accéder uniquement à l'un d'entre eux. On appellera ce regroupement une mémoire. Une "mémoire" est donc un ensemble de registres, chacun d'eux constitue un mot de la mémoire. On accède à un mot par son adresse qui le désigne de manière unique. Une mémoire est donc un banc de registres nécessitant un multiplexage (couche logique) des entrées pour la gestion de l'adressage.

Les mémoires les plus simples sont les mémoires simple-port. C'est-à-dire que l'on ne peut effectuer qu'une seule action sur les mots de la mémoire (lecture ou écriture) durant un top horloge. Il existe aussi des mémoires multi-port. Dans ce cas, plusieurs actions sont possibles en même temps (durant le même top horloge). Ces mémoires nécessitent un mécanisme pour gérer les conflits entre les différents ports.

Mesure de complexité en espace

Il existe des modèles de calcul de complexité pour les systèmes VLSI [Len90]. Ils permettent d'étudier l'aire d'un circuit, le temps de calcul et l'energie consommée qui est un critère particulièrement important. Dans le cadre de l'utilisation de matériel reconfigurable (cf chapitre IV) le concepteur dispose d'une certaine quantité de ressources limitée qu'il doit être à même d'utiliser au mieux. Dans ce cas, pour parler de la complexité matérielle d'un algorithme, il est plus judicieux de quantifier l'utilisation des trois ressources identifiés ci dessus : logique, registres, mémoires.

Sur un circuit, la différence entre ces trois ressources n'est pas toujours claire. Par exemple, un petit banc de registre sera plus facilement associé à de la glu logique qu'à une mémoire. En effet, l'utilisation des transistors dans la construction d'une mémoire est très fortement optimisé, le coût en surface d'une grosse mémoire sera donc bien différent du coût d'un registre ou d'une couche logique. La mesure de complexité la plus réaliste pour un circuit est plutôt sa taille en surface de silicium. Même une mesure en nombre de transistors n'est pas suffisante, en effet un transistor n'a pas le même coût matériel s'il s'agit d'un transistor d'une mémoire ou d'un transistor d'une porte logique.

Par contre, la différenciation de ces trois ressources paraît plus adaptée pour la mesure de complexité d'un algorithme. La figure (III.1) représente l'espace des ressources matérielles. Le paragraphe (III.2.b) montre comment on peut jouer sur ces différentes ressources pour implanter une fonction en utilisant l'une ou l'autre de ces ressources.

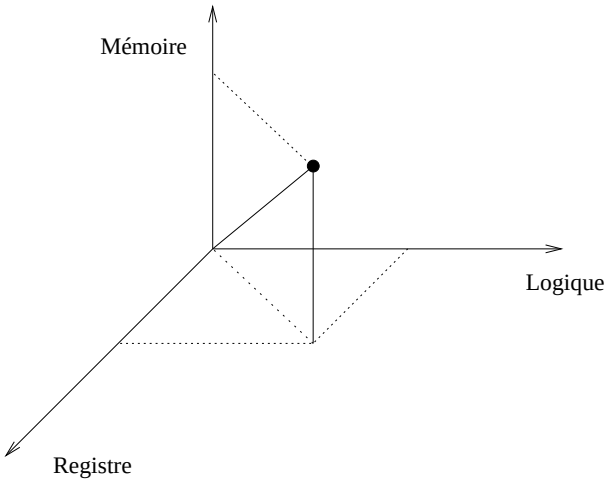

FIG. III.1: L'espace des ressources matérielles. Toute réalisation a un coût qui peut s'exprimer à l'aide de trois composantes. Ces trois composantes correspondent à trois ressources matérielles différentes. Une même fonction peut être implantée de plusieurs manières différentes avec un coût différent entre ces trois ressources.

III.1.b Mesure de complexité en temps

Ce paragraphe présente différentes manières de mesurer la complexité temporelle d'un algorithme implanté en utilisant du matériel. De manière très simple, le top horloge peut constituer une bonne unité de complexité. L'efficacité de deux algorithmes peut donc être comparée en comparant le nombre de tops horloge qu'ils utilisent pour fournir le résultat attendu.

La taille du top horloge

Le top horloge peut donc être choisi comme unité de comparaison. La vitesse de l'horloge se mesure en nombre de tops horloge par seconde, l'unité est le Hertz (Hz).

Cependant, la vitesse de l'horloge est principalement déterminée par la taille de la plus grosse couche logique. En effet, plus le temps de stabilisation d'une couche logique est importante, plus l'horloge sera lente.

Ainsi, d'un algorithme à l'autre, les fréquences d'horloge peuvent être très différentes.

Une unité de temps arbitraire

Le top horloge n'est donc pas forcément un choix judicieux pour comparer le coût temporel de deux algorithmes.

Dans la plupart des cas, il est plus raisonnable de choisir comme unité de comparaison le temps d'une opération de base commune aux deux algorithmes.

III.2 Exemples d'utilisation de ressources matérielles

L'objectif de ce paragraphe est de montrer, à partir des trois types de ressources présentés dans (III.1), comment "transformer" des algorithmes en matériel. Les différents choix possibles sont présentés au travers de plusieurs exemples. Le premier, dans (III.2.a), montre comment réaliser des boucles. Le deuxième, dans (III.2.b), explique comment réaliser des fonctions plus complexes. Ces différents composants ont tous été réalisés et utilisés dans les chapitres (V), (VI), et (VII).

Les outils (VHDL, synopsys) permettant de réaliser ceci seront présentés dans le chapitre IV. Ici, on considère que l'on dispose d'un certain nombre de composants de base permettant de calculer des sommes, des soustractions, des minima...

III.2.a Utilisation du parallélisme à grain fin

Les nombreuses possibilités qu'offre l'utilisation de ressources matérielles pour la réalisation d'un algorithme vont être présentées au travers d'un exemple : Considérons la matrice $A = (a_{ij})_{(i,j=1..n)}$. On cherche à calculer le vecteur $m = (m_i)_{i=1..n}$ tel que :

$$m_i = \min_{j=1..n}(a_{ij}), \forall i \in [1..n],$$

où les a_{ij} sont des entiers positifs et n une constante.

Dans un premier temps, on présente les différentes possibilités envisageables pour le calcul de $\min_{j=1..n}(a_{ij})$. Puis l'on verra comment répéter ce calcul pour chaque ligne de la matrice.

Calcul du minimum de deux entiers positifs

Ce composant de base fournit le minimum de deux entiers positifs. Les entrées de ce composant sont deux entiers a et b. La sortie r est le minimum des deux : $r = min(a, b)$.

Ce composant doit être vu comme une boite qui donne de manière instantanée un nouveau résultat dès qu'elle reçoit un ordre d'évaluation. En général, cet ordre est donné par un front d'horloge (cf figure III.2). Par la suite, ce composant sera désigné par *Min2*. Il est composé d'une couche logique, pour le calcul du minimum, et d'un registre qui stocke le résultat.

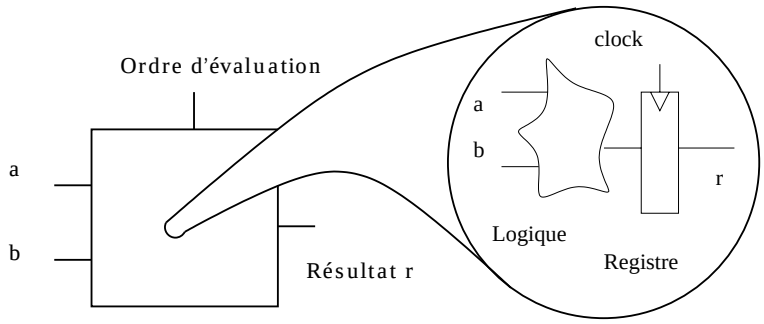

FIG. III.2: la "boite" *Min2* calcule le minimum de deux entiers positifs. Le résultat est réévalué à chaque top horloge. En général, le temps d'évaluation est très faible par rapport à la période de l'horloge, on pourra donc le considérer comme instantané.

Calcul d'une composante m_i

Trois méthodes de calcul différentes pour $m_i = \min_{j=1..n}(a_{ij})$ vont être présentées. Ces trois méthodes correspondent à trois manières différentes de construire un composant *Min_mi* pour calculer m_i. Il s'agit aussi de trois manières différentes de matérialiser une boucle de calcul. L'élément de base sera le composant *Min2* présenté au paragraphe précédent. Le calcul de m_i nécessite la comparaison des n nombres a_{ij} pour en trouver le minimum, la complexité de ce problème en nombre de comparaisons à effectuer est donc $n-1$.

- Méthode "séquentielle" :

  Cette méthode calcule m_i de manière purement séquentielle. Pour cela, elle utilise un seul composant $Min2$. En effet, il suffit, à chaque top horloge, d'en mettre à jour les entrées de manière correcte pour récupérer le résultat m_i au bout de $(n-1)$ tops horloge.

  Ainsi, au $n-1^{ième}$ top, les entrées sont r_{n-2} et a_n et le résultat m_i. Cette solution utilise donc $(n-1)$ tops horloge et un composant *Min2* pour réaliser le calcul (cf figure III.3).

- Méthode "pipelinable" :

  Ici, $(n-1)$ comparateurs *Min2* sont utilisés pour effectuer les comparaisons nécessaires au calcul. Le résultat du premier calcul est connecté à l'entrée du deuxième composant *Min2* et ainsi de suite. L'avantage n'est pas forcément visible tout de suite mais il le sera à l'étape suivante (lors de la construction du composant calculant toutes les composantes du vecteur m).

Cette solution utilise donc $n-1$ tops horloge et $n-1$ composants *Min2* pour réaliser le calcul (cf figure III.3).

– Méthode "parallèle" :

Cette solution profite au maximum du parallélisme intrinsèque du matériel et du calcul que l'on souhaite réaliser. En effet, pendant un même top horloge, $n/2$ comparaisons peuvent être réalisées en prenant les a_{ij} deux à deux. Puis, les résultats de ces comparaisons sont à nouveau comparés deux à deux. On obtient ainsi $\log_2(n)$ étages de comparateurs où chaque étage i comporte $n/(2^i)$ comparateurs.

Le coût de cette solution est donc de $\log_2(n)$ tops horloges et de $n-1$ composants *Min2* (cf figure III.3).

Pour la suite de l'exemple – le calcul de toutes les composantes du vecteur m – on considérera que l'on possède une boite noire nommée *Min_ mi* qui a pour entrées les entiers a_{ij} et qui fournit en résultat m_i. La composition interne de cette boite et l'une des trois méthodes proposées ci-dessus.

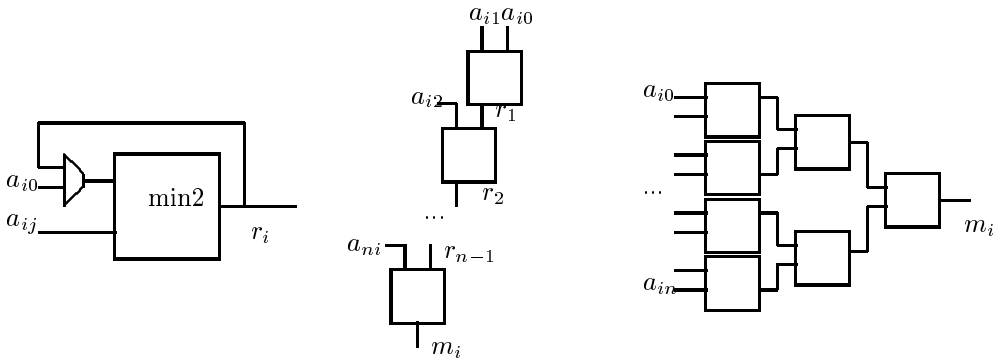

FIG. III.3: Trois méthodes différentes pour calculer m_i. Toutes sont différentes au niveau de l'utilisation du temps ou du nombre d'utilisations du composant *Min2*.

Calcul du vecteur m

Le calcul de m_i pour un i quelconque se fait donc grâce au composant *Min_ mi*. On veut maintenant construire un composant *Calcm* qui fournit toutes les composantes du vecteur m. De la même manière que le composant *Min_ mi* était construit à partir de *Min2* ce composant peut être construit de différentes façons à partir du composant *Min_ mi*. On propose ici trois manières différentes de réaliser *Calcm*.

– Méthode "séquentielle" :

La méthode purement séquentielle, comme pour la construction de *Min_ mi*, consiste à utiliser un seul composant de base et à modifier les entrées à chaque changement de l'horloge. Ainsi à chaque fois qu'un résultat est fourni par le composant *Min_ mi*, on change ses entrées pour obtenir le m_i suivant. Ainsi ce composant utilise un seul composant *Min_ mi* et les résultats sont obtenus après n "exécutions".

- Méthode "parallèle" :
  La méthode parallèle consiste à utiliser le fait que les calculs de m_i sont complètement indépendants les uns des autres. Ainsi ces calculs peuvent s'exécuter en parallèle. La construction du composant calculant le vecteur m passe donc par l'utilisation de n composants *Min_ mi*. Le temps nécessaire au calcul du vecteur m correspond au temps que mettent les composants *Min_ mi* pour fournir les m_i.
- Méthode "pipelinée" :
  La méthode pipelinée consiste en l'utilisation des composants *Min2* qui ne servent pas à un instant donné. En effet, dans les architecture dites "pipelinable" et parallèles du composant *Min_ mi*, quand un étage du composant travaille les autres restent inactifs. L'idée est donc de fournir au premier étage du composant de nouvelles données à chaque top horloge. Ainsi le premier résultat sort au bout d'un temps *nombre d'étages* et ensuite un nouveau résultat sort à chaque top... On utilise ainsi un seul composant *Min_ mi* et le temps de calcul dépend du nombre d'étages de ce composant ainsi que du nombre n de calculs à effectuer.

Différences

Les différences entre ces solutions sont principalement des différences de coût matériel ou de coût temporel. Ainsi la méthode purement séquentielle est la plus coûteuse en temps, mais la moins coûteuse en matériel. Par contre une méthode purement parallélisée est très coûteuse en matériel mais très économique au niveau temporel.

La complexité du problème, ici un nombre de comparaisons à effectuer, se retrouve forcément sous forme matérielle ou temporelle.

Le tableau (III.1) montre la grande diversité des solutions offertes par le traitement matériel d'un problème. Un autre exemple de réalisation d'un algorithme à l'aide de ressources matérielles sera présenté dans la section (IV.1).

TAB. III.1: Récapitulation des coûts des différentes méthodes de calcul du vecteur m. La complexité des différentes possibilités est donnée par le couple (M,T) où M est le coût matériel mesuré en nombre de composants $Min2$ et T le coût temporel en temps horloge.

Calcm Minmi	Séquentielle	Pipeline	Parallèle
Séquentielle	$(1, n(n-1))$	$(n-1, n(n-1))$	$(n-1, n\log_2(n))$
Pipeline	impossible	$(n-1, n+n-1)$	$(n-1, n+\log_2(n))$
Parallèle	$(n, n-1)$	$(n(n-1), n-1)$	$(n(n-1), \log_2(n))$

III.2.b Fonctions "complexes"

Le paragraphe (III.2.a) montre comment mettre bout à bout des fonctions simples, ici l'objectif est de montrer comment certaines fonctions apparemment plus complexes peuvent aussi être implantées en utilisant des ressources matérielles.

Générateur pseudo-aléatoire

La génération de nombres pseudo-aléatoires est un problème important pour toutes les activités de modélisation [SLC75, Dev97, Ent98, Mat98, MN98, Tez95]. Le problème consiste à trouver un algorithme déterministe qui fournisse l'impression du hasard.

L'un des principes souvent employé consiste à utiliser des congruences linéaires [Ent98]. La méthode retenue ici est celle qui présente le plus de facilité pour être réalisée à l'aide de matériel. Il s'agit de la méthode du registre à décalage bouclé.

Le choix de cette méthode peut se justifier par des considérations algébriques. Considérons $\mathcal{K} = \{0, 1\}$ le corps à 2 éléments muni de l'addition binaire (XOR) et de la multiplication binaire (AND). L'état du registre peut être considéré comme le reste de divisions successives par le polynôme

$$\mathcal{P}(X) = 1 + \sum_{i=1}^{k} a_i X^i$$

dans le corps de Galois à 2^n éléments. On peut montrer que, quand le polynôme $\mathcal{P}$ est primitif, alors le registre prend les $2^n - 1$ valeurs possibles.

On obtient un algorithme efficace [Tez95] avec un polynôme caractéristique de la forme $1+X^s+X^r$, avec $s < r/2$. Des exemples de valeurs pour r et s sont donnés dans la table III.2.

TAB. III.2: Exemple de valeurs pour r et s, tiré de [Tez95] page 88.

r	s
2	1
7	1, 3
31	3, 6, 7, 13
127	1, 7, 15, 30, 63
607	105, 147, 273

Cette méthode permet de générer 1 bit de manière aléatoire. Pour générer un mot aléatoire de n bits, il suffit de prendre une séquence de n bits, ou de faire marcher en parallèle n registres initialisés différemment. Dans ce dernier cas, l'utilisation de registres peut être avantageusement remplacée par l'utilisation d'une mémoire comportant $(r-1)$ mots de n bits ainsi qu'un registre de n bits. Prenons, par exemple, $r = 127$ et $s = 1$ (le polynôme $1 + X + X^{127}$). On peut implanter celui-ci grâce à n registres de 127 bits en parallèle ou par une mémoire de 126 mots et un registre de n bits. De plus, si l'on dispose d'une mémoire double port, la lecture et l'écriture ont lieu durant un même top horloge. La figure (III.4) montre comment on peut implanter un tel système de génération de mots pseudo-aléatoires.

Ceci permet la génération de variables suivant une loi uniforme. Ainsi, ce type de générateur sera utilisé pour l'émulation des processus d'arrivée et de service lors de l'étude de réseaux de files d'attente (voir III.3).

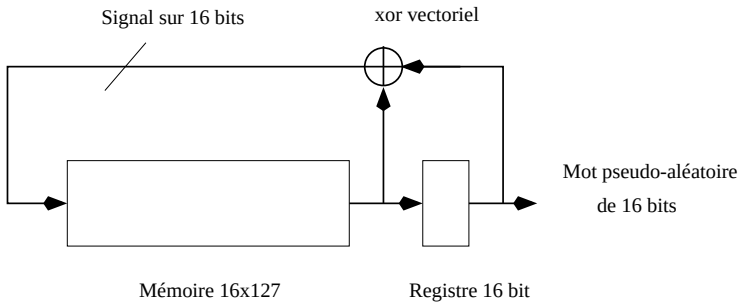

FIG. III.4: Implantation possible d'un générateur pseudo-aléatoire de mots de 16 bits. On utilise une mémoire et un registre de 16 bits, le tout fonctionnant comme un registre à décalage.

Discrétisation d'une fonction : log **et** exp

Ces calculs peuvent s'effectuer de différentes manières. Une première méthode consiste à utiliser des développements limités. Mais, dans le cas où le nombre de valeurs possibles des opérandes est limité, on peut utiliser la discrétisation dans une mémoire.

Prenons pour exemple le calcul d'une fonction $f(x)$, où x est un nombre entier codé sur n bits. Ce calcul demandera une mémoire de 2^n mots. Dans cette mémoire on trouve à l'adresse x la valeur de $f(x)$. Avec cette méthode, la réalisation des fonctions logarithme et exponentielle demande un pré-calcul dont le résultat est stocké dans une mémoire.

Le problème est alors de choisir un codage adapté pour la valeur $f(x)$. Pour le calcul du logarithme d'un nombre entier, le principal problème est donc de savoir comment choisir le codage des logarithmes. Le codage le plus simple consiste à utiliser un codage en virgule fixe du type $r.s$. Les r premiers bits codent la partie entière du nombre et les s bits suivants codent la partie décimale. Il existe des codages plus complexes comme par exemple l'utilisation d'une mantisse. Le choix doit ici se faire en fonction des précisions que l'on attend sur les résultats.

Si on choisit un codage en virgule fixe ($r.s$) alors les mots de la mémoire seront de taille $r+s$. La démarche consiste donc à considérer l'entier x codé sur n bits comme une adresse de n bits. Ce mot adresse une table dans laquelle on lit un mot de $r.s$ bits qui est la valeur de $\log(x)$.

Pour calculer la fonction inverse, on a un nombre y codé sur $r.s$ bits et on cherche à calculer l'entier $\exp(y)$ codé sur n bits. Là encore on considère le mot de $r+s$ bits comme l'adresse dans une table qui contient $r+s$ mots de n bits. Les $r.s$ bits adressent une table dans laquelle on lit un résultat n bits.

Lors de l'utilisation d'une telle table, il est nécessaire de prêter une grande attention aux problèmes de précision de calcul. Le paragraphe suivant montre comment utiliser ces tables pour réaliser à moindre coût un multiplicateur. En effet, ici le coût en temps est limité à

une lecture (un top horloge) et le coût en espace à une mémoire plus ou moins grande selon la précission demandée.

multiplication / division

Les multiplications et les divisions sont des opérations qui demandent un grand nombre de ressources, tant au niveau matériel qu'au niveau temporel. Ces opérations sont basiquement implantées dans les processeurs classiques, néanmoins en vue de porter nos modèles sur du matériel reconfigurable, où les ressources sont limitées, il est intéressant de trouver des méthodes d'implantation moins coûteuses (en temps et en espace). En effet, un multiplicateur (ou un diviseur) classique demande un ou plusieurs tops horloge avec un coût matériel très important en glu logique et en registres.

La solution classique de ce problème est de calculer le logarithme des opérandes, puis d'effectuer une addition ou une soustraction avant de calculer un exponentiel pour récupérer le résultat.

Ainsi, en utilisant les méthodes de calcul présentées ci-dessus, un multiplicateur utilisera deux ou trois tables mémoire. Une ou deux pour calculer le log des nombres à multiplier et une pour calculer l'exponentiel du résultat de l'addition ou de la soustraction. Il reste à être conscient de la précision des calculs effectués pour les employer a bon escient.

Le nombre de tops horloge nécessaires pour la réalisation d'une multiplication est donc d'un minimum de 3. Un top pour la lecture en mémoire des valeurs des log, un top pour l'addition est enfin un top pour la lecture du résultat dans la table des exp (voir figure III.5). Le coût matériel des opérateurs (utilisation des ressources glu logique et registre) a ainsi été transformé en une utilisation de la ressource mémoire.

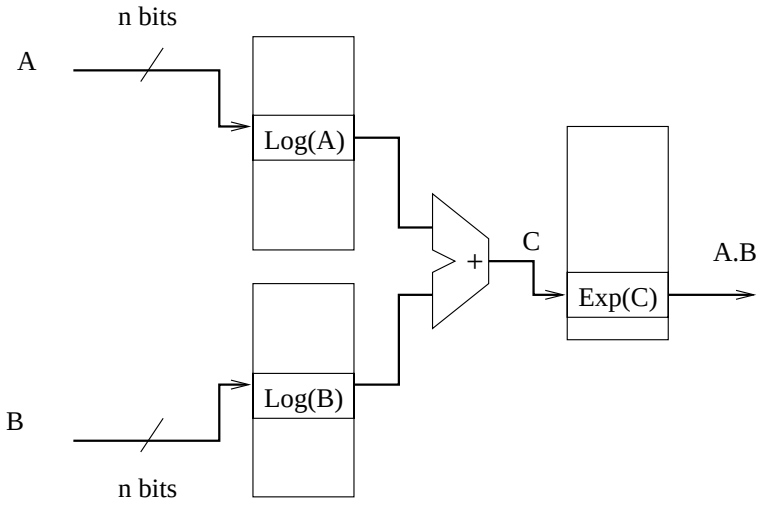

FIG. III.5: Réalisation d'une multiplication à partir de logarithmes et d'exponentiels pré-calculés dans des tables.

Ce multiplicateur sera utilisé lors de l'étude d'un algorithme de service de type "Weigted Fair Queuing" dans le chapitre (VII).

III.3 Simulation de files d'attente

Les files d'attente en temps discret sont particulièrement importantes pour la simulation des réseaux ATM (voir chapitre II). Cette section va permettre de voir comment des files d'attente peuvent être simulées grâce à des ressources matérielles. Le paragraphe (III.3.a) propose différentes manières de représenter une file d'attente. Le paragraphe (III.3.b) montre comment implémenter les processus de service et d'arrivée et enfin le paragraphe (III.3.c) propose différentes manières de collecter des informations sur les événements survenant dans un réseau de files d'attente.

III.3.a Différents modèles

Sans contenu ou avec contenu

La représentation matérielle d'une file dépendra de plusieurs paramètres, comme les caractéristiques des clients ou les caractéristiques du (ou des) serveurs.

Si tous les clients sont similaires, seul un registre et de la glu logique seront nécessaires pour implémenter une file. Le registre contiendra le nombre de clients en attente. Il sera mis à jour en fonction du nombre des nouvelles arrivées, de la capacité de la file, et du nombre de clients servis. Dans ce cas précis, on peut faire marcher la file d'attente avec l'horloge. Ainsi, un top horloge correspond à un slot. Cette méthode donne uniquement comme information le nombre de clients présents à un instant donné dans la file. C'est une représentation peu coûteuse en termes de matériel, car elle utilise juste un registre et un peu de glu logique (cf figure III.6). Cette solution a aussi l'avantage d'autoriser la simulation des modèles de files d'attente à capacité infinie.

Si les clients ont besoin d'être différenciés, ou de transporter réellement de l'information, il faut complexifier le modèle. En effet, il faut alors gérer l'individualité des paquets, et utiliser des "vrais" clients qui doivent être stockés dans une mémoire. Dans ce cas, une file d'attente est représentée par une mémoire qui correspond en taille à la capacité de la file d'attente. Par contre, en cas d'arrivées multiples dans un slot, il faut être capable d'écrire en mémoire plusieurs paquets en un seul slot. Il n'est donc plus possible de garder l'équivalence entre le slot et le top horloge. Ceci réduit la vitesse de la simulation. De plus, il n'est plus possible, avec cette représentation de simuler de files d'attente de capacité infinie. En contrepartie, cette méthode présente une extension du modèle, en effet, on peut ainsi marquer des paquets, prendre en compte des priorités, ou transporter des dates.

Il est cependant possible de conserver l'équivalence entre le slot et le top horloge en utilisant plusieurs mémoires pour une file d'attente. Si le modèle autorise n arrivées en un slot alors on utilise n mémoires en parallèle. Ainsi, en cas d'arrivées multiples, chaque paquet est écrit dans une mémoire différente. On utilise donc n fois plus de mémoire que la capacité de la file d'attente. Cette méthode est très coûteuse pour un gain en rapidité qui n'est pas forcément flagrant.

Traitement de la simultanéité (AF/DF)

Comme dans toute file d'attente en temps discret se pose le problème de la gestion de l'instantanéité (voir paragraphe II.1). Ce problème est tout à fait visible lors de l'implantation à l'aide de matériel. En effet, dans le modèle avec information, il faut soit écrire

les arrivées, soit lire les départs en premier. En choisissant l'une de ces deux méthodes, on choisit aussi l'un des deux modèles.

Dans le modèle sans information, il faut mettre à jour le registre d'occupation en fonction de la politique choisie (cf figure III.6).

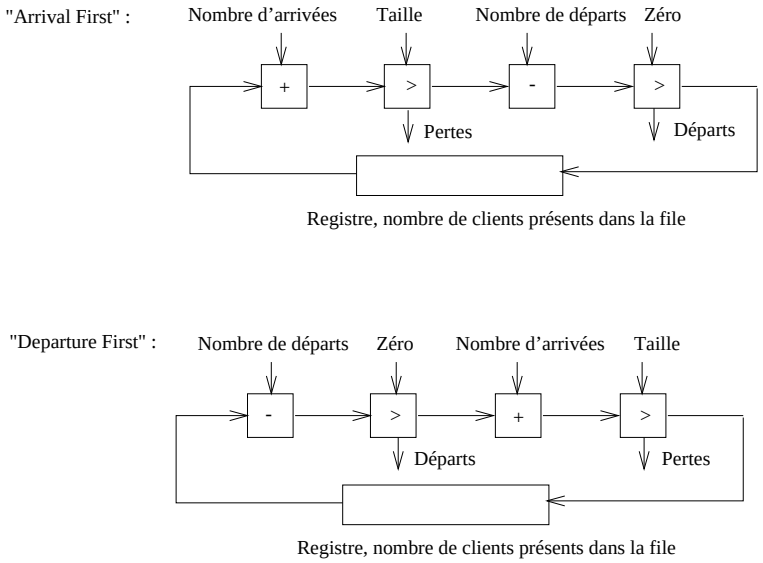

FIG. III.6: Réalisation et mise à jour du compteur qui comptabilise le nombre de clients présents dans une file. Deux choix sont possibles, chacun correspond à un traitement de la simultanéité (AF ou DF). Pour un modèle de files "sans information" ce compteur suffit pour représenter une file d'attente.

III.3.b Processus de service et d'arrivée

Après avoir vu comment représenter à l'aide de ressources matérielles, les capacités des files, il reste à montrer comment matérialiser la partie processus d'arrivée et de service des files d'attente.

Ce problème se ramène pour beaucoup au problème de la génération de nombres pseudo-aléatoires abordé au paragraphe (III.2.b). Le problème revient à transformer - en utilisant des ressource matérielles - la réalisation d'une loi uniforme en la réalisation d'une loi plus complexe.

Ainsi, si l'on souhaite générer la réalisation d'une variable aléatoire gaussienne on peut utiliser la méthode d'inversion, en tabulant la fonction dans une mémoire (cf Figure III.7). Cette méthode permet de générer n'importe quelle réalisation de variable aléatoire.

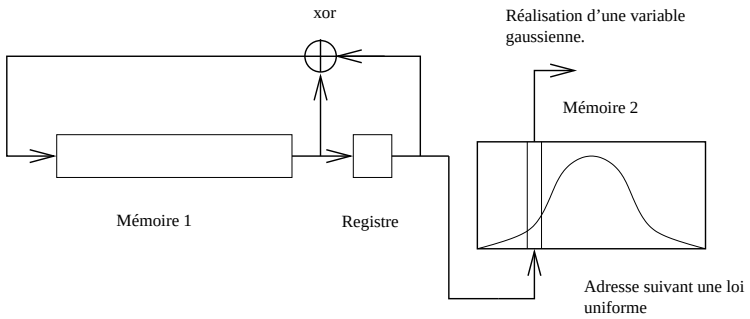

FIG. III.7: utilisation de deux mémoires pour simuler la réalisation d'une variable aléatoire gaussienne. La première mémoire (mémoire 1) sert à la génération d'un mot suivant une loi uniforme, la deuxième (mémoire 2) sert "d'inverseur".

Géométrique

Le processus géométrique est un processus simple et souvent employé (cf. II.2). En effet, sa facilité de manipulation est d'un grand intérêt pour l'étude des systèmes complexes. Le processus géométrique de paramètre ρ est simple à simuler. En effet, à chaque slot un événement a lieu avec une probabilité ρ.

En utilisant le générateur pseudo-aléatoire présenté dans (III.2.b) on dispose, à chaque top horloge, d'un mot pseudo-aléatoire w constitué de n bits. Soit w_i le mot généré au ième top. En considérant ce mot comme un entier on a :

$$0 \leq w_i \leq 2^n - 1.$$

Pour simuler un processus géométrique de paramètre ρ, il suffit de comparer à chaque slot, l'entier w_i à :

$$\rho_n = \lfloor \rho * 2^n - 1 \rfloor,$$

où $\lfloor . \rfloor$ est la partie entière inférieure. Un événement a lieu au ième slot si, et seulement, si $w_i \leq \rho_n$. Par exemple, si on souhaite simuler un processus géométrique de paramètre $\rho = 0.8$ et $n = 16$, on prendra $\rho_n = 52429$. La précision obtenue est de l'ordre de $1/2^n$.

Ainsi, ce processus s'obtient en ajoutant juste un comparateur (un peu de glu logique et un registre) au générateur pseudo-aléatoire.

MMBP

On peut vouloir utiliser des fonctions de génération un peu plus complexes comme un automate stochastique (de type Markov Modulated Bernouilli Process). Ces automates stochastiques permettent l'élaboration de sources $on - off$ ou auto-similaires (cf. II.3).

Ce type de sources peut être réalisé à l'aide d'une table mémoire. Chaque mot de cette table mémoire donne l'état suivant en fonction de l'état courant.

Par exemple, pour réaliser un automate ayant k états, on utilise une mémoire dans laquelle chaque mot est divisé en k parties de longeur $log_2(k)$. La $i^{ème}$ de ces parties code

l'état suivant dans le cas ou l'état courant est l'état i. Donc à chaque instant, seule une partie du mot mémoire est utilisée, c'est la partie du mot qui correspond à l'état courant.

L'utilisation d'une unique table mémoire ayant des mots de taille $k.log_2(k)$ peut éventuellement être remplacée par l'utilisation de n mémoires (une par état) ayant des mots de taille $log_2(k)$.

LB

La génération d'un trafic contraint, par exemple un trafic conforme aux spécifications de l'algorithme du Leaky-Bucket, on peut utiliser une file d'attente et un compteur de crédits. Comme pour les contrôleur réel, on peut choisir de tuer le trafic non conforme ou d'émettre des cellules avec un marquage en cas de non-conformité.

III.3.c Instrumentation

L'étude d'un réseau de files d'attente a pour objectif l'observation d'un certain nombre de phénomènes. On cherche à connaître l'occupation des files, le temps d'attente de chacun des clients, les pertes dues aux débordements. D'autre part, dans le cas des réseaux ATM, on cherche aussi à savoir comment le passage dans les files d'attente perturbe le trafic.

La modélisation de files d'attente par des ressources matérielles, nécessite donc la création d'un certain nombre de composants dont le rôle va être de récolter et de stocker les informations jugées importantes pour l'étude des files d'attente.

Cette section a pour but de présenter différentes manières de récolter ces informations qui sont l'objectif même de la modélisation par files d'attente.

Comptabilisation

La plupart des indices de performance sont des probabilités (voir chapitre I). L'approximation de ces probabilités nécessite une comptabilisation d'événements.

Ainsi, pour connaître le taux de pertes d'une file d'attente, il suffit de comptabiliser le nombre de pertes (les problèmes de précision seront traités au chapitre V). Pour le taux d'occupation du serveur, il suffit aussi de comptabiliser le nombre de fois que le serveur a été vide. Pour tout un nombre d'indices de performance, la comptabilisation d'événements suffit pour en avoir une bonne évaluation.

La comptabilisation du nombre de fois qu'a lieu un événement est assez simple. Cette opération nécessite un registre et un additionneur. Par exemple pour comptabiliser les pertes dans un modèle de file d'attente comme celui présenté ci-dessus, il suffit de prévoir un signal qui indique le nombre de pertes dans un slot. Ce nombre de pertes est additionné dans un registre qui comptabilise le nombre de pertes totales. Pour calculer le taux de pertes, il faut en plus comptabiliser le nombre total de clients traités. Ainsi, avec deux compteurs, il est possible de réaliser des bonnes estimations d'un certain nombre d'indices de performance.

Ce type d'instrumentation est peu coûteuse et permet de recueillir des informations d'ordre général sur le système. La figure (III.8) présente un exemple d'instrumentation pour le calcul d'un taux de pertes.

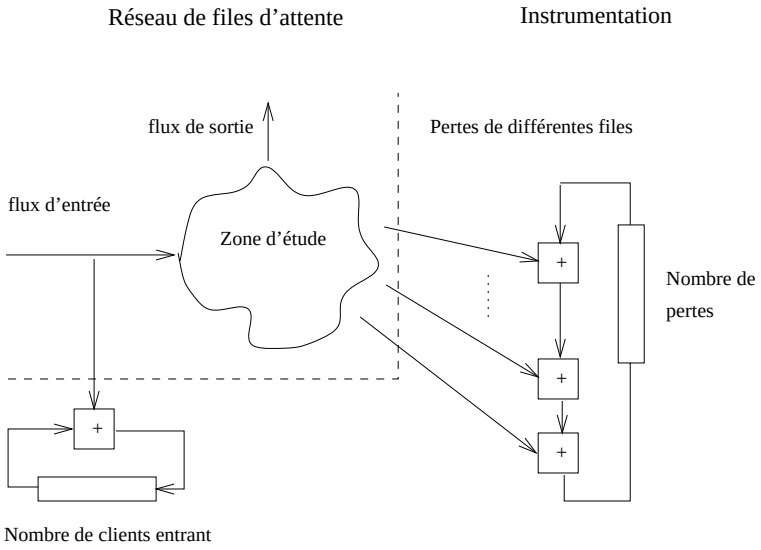

FIG. III.8: exemple d'instrumentation de type "comptabilisation". Pour étudier le taux de pertes sur une partie d'un réseau de files d'attente, deux compteurs et des additionneurs suffisent.

Histogrammes

Il est parfois important de pouvoir évaluer, non pas la probabilité d'un événement, mais la distribution d'une variable aléatoire discrète. Une solution consiste à utiliser un registre pour chaque valeur que peut prendre la variable aléatoire, mais il est plus logique d'utiliser une mémoire. En effet, à chaque slot on a besoin d'accéder à un seul des registres. Ainsi, la mémoire enregistre le nombre de réalisations de chacune des valeurs possibles de la variable aléatoire. Cette instrumentation va permettre de tracer un histogramme, et d'obtenir ainsi une estimation empirique de la distribution de la variable aléatoire étudiée.

Ce type d'instrumentation peut être utilisé pour évaluer les distributions de l'occupation des files du système, mais aussi pour mesurer les perturbations introduites sur le trafic par le passage dans les files d'attente. Dans ce dernier cas, on estime de manière empirique la distribution des inter-départs d'une file, et on la compare à la distribution (empirique ou théorique) des inter-arrivées.

Au niveau ressource, cette instrumentation nécessite l'utilisation d'une mémoire, et la division du slot en plusieurs tops horloge. En effet, les actions à réaliser en un slot sont : lecture en mémoire de la valeur à incrémenter, addition et enfin écriture en mémoire de la nouvelle valeur.

Historique

Il peut aussi être intéressant d'étudier la distribution de certains événements dans le temps et de pouvoir dater chacun d'eux.

L'opération nécessite un composant qui enregistre un "historique" de ces événements. Ceci peut être effectué à l'aide d'une mémoire qui sauvegarde la date et la valeur des variables étudiées.

Ce type d'instrumentation permet de (re)construire l'historique d'un événement pour les n derniers changements de valeur, n étant la taille de la mémoire utilisée pour stocker l'historique.

Bilan sur l'instrumentation

Ce paragraphe montre que, lors de la construction du circuit qui va représenter une modélisation, il faut prévoir un certain nombre de composants pour récolter l'information intéressante.

Le coût matériel de cette instrumentation ne doit pas être négligé. En effet, l'instrumentation peut très vite prendre plus de place que la partie modélisation du système à étudier.

Ainsi, les ressources utilisées pour réaliser la simulation d'un réseau de files d'attente sont de deux natures différentes. Certaines sont utilisées pour modéliser le système étudié et d'autres pour recueillir l'information jugée intéressante.

Il est important de noter que l'implantation d'un modèle en hardware va dépendre non seulement de la modélisation faite du problème, mais aussi des indices de performance auxquels on s'intéresse.

III.4 Conclusion

Ce chapitre a montré qu'il est "facile" et intéressant d'utiliser des ressources matérielles pour l'étude des réseaux de files d'attente en temps discret. Le paragraphe précédent montre que l'implantation de files d'attente en utilisant des ressources matérielles est simple et peu coûteuse. De plus, le parallélisme intrinsèque du matériel devrait permettre un gain important de rapidité par rapport aux méthodes classiques (voir chapitre II). C'est pour ces raisons que la méthode peut être utilisée dans l'espoir de résoudre des problèmes - comme la mise en évidence d'événements rares - inaccessibles avec l'utilisation de méthodes classiques.

Il est bien entendu inconcevable (aujourd'hui) de mener la construction d'un circuit à son terme uniquement pour réaliser une simulation. C'est pourquoi le chapitre suivant propose de présenter les outils qui peuvent être utilisés pour réaliser ces modèles à l'aide de matériels sans pour autant construire un circuit réel.

Bibliographie

[CLR94] Thomas Cormen, Charles Leiserson, and Ronald Rivest. *Introduction à l'algorithmique*. Dunod, Massachusetts Institute of Technology, 1994.

[Dev97] Luc Devroye. Random variate generation for multivariate unimodal densities. *ACM, Transactions On Modeling and Computer Simulation*, 7-4, 1997.

[Ent98] K. Entacher. Bad subsequences of well-known linear congruential pseudo-random number generators. *ACM, Transactions On Modeling and Computer Simulation*, 8-1, 1998.

[FLP98] Erwan Fabiani, Dominique Lavenier, and Laurent Perraudeau. Loop parallelization on a reconfigurable coprocessor. In *Workshop on Design, Test and Applications*, IRISA, Rennes, 1998.

[KR90] R. M. Karp and V. Ramachandran. Parallel Algorithms for Shared-Memory Machines. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A : Algorithms and Complexity, chapter 17, pages 870–941. North Holland, 1990.

[Len90] Lengauer. VLSI theory. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A : Algorithms and Complexity, chapter 16, pages 837–869. North Holland, 1990.

[Mat98] M. Matsumoto. Simple cellular automata as pseudo-random m-sequence generators for built-in self-test. *ACM, Transactions On Modeling and Computer Simulation*, 8-1, 1998.

[MN98] M. Matsumoto and T. Nishimura. Mersenne twister : A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM, Transactions On Modeling and Computer Simulation*, 8-1, 1998.

[SLC75] H. Daniel Schnurmann, Eric Lindbloom, and Robert G. Carpenter. The weighted random test-pattern generator. *IEE Transactions on computers*, 14-1, 1975.

[Tez95] Shu Tezuka. *Uniform Random Number : Theory and practice*. Kluwer Academic Piblishers, IBM Japan, 1995.

[Zak85] Rodnay Zaks. *Du composant au système : introduction aux microprocesseurs*. Sybex, 1985.

Chapitre IV

Les architectures reconfigurables et Sim-express

L'objectif du chapitre est de présenter les différents outils qui peuvent être utilisés pour simuler des files d'attente à l'aide de ressources matérielles. En effet, ceci est rendu possible par l'utilisation d'architectures reconfigurables. Ce type d'architecture permet l'émulation d'un circuit. Ce chapitre présente tous les outils utilisés pour la réalisation d'une simulation à l'aide des ressources matérielles fournit par une architecture reconfigurable.

Ce type d'architectures est de plus en plus utilisé dans un grand nombre de domaines. La section (IV.1) propose un éventail non exhaustif des domaines d'application pour lesquels ces architectures ont été ou sont encore utilisées. Ce sera l'occasion de présenter dans le paragraphe (IV.1.d) quelques unes des architectures reconfigurables les plus importantes.

Ceci conduira à la présentation de la machine Sim-express dans la section (IV.2) ainsi que des outils logiciels qui permettent de l'utiliser.

La dernière section (IV.3) montre l'utilisation de ces outils au travers de l'exemple de la file $Geo/Geo/1/k$.

IV.1 Les différentes applications des architectures reconfigurables

De nombreuses équipes à travers le monde se sont efforcées de trouver une application décisive (killer application) pour les systèmes à architecture reconfigurable. Cette effervescence a donné lieu à des recherches dans de nombreux domaines différents. Souvent l'utilisation d'une architecture reconfigurable a donné des résultats bien supérieurs au plus puissant calculateur du moment. Un certain nombre d'applications commerciales ont vu le jour, notamment en biologie et dans le domaine de la conception de circuits.

Le premier paragraphe (IV.1.a) de cette section présente l'utilisation de ce type d'architecture dans le cadre de la conception de circuits, qui reste l'application dominante des architectures reconfigurables. Le deuxième paragraphe (IV.1.b) énumère un certain nombre de réalisations qui ont toutes prouvé leur efficacité (biologie, calcul...). Enfin, la machine FAST sera présentée dans le paragraphe (IV.1.c). Cette machine est constituée d'une architecture reconfigurable dédiée à la simulation de commutateurs ATM.

IV.1.a Les architectures reconfigurables dans la conception d'un circuit

La conception de circuits est le domaine privilégié pour l'utilisation des architectures reconfigurables. En effet, elles sont de plus en plus couramment utilisées comme accélérateur de simulation (émulation) ou comme plates forme de tests pour des circuits. Elles s'intègrent parfaitement dans la chaîne de conception classique des circuits, et font partie intégrante des techniques de codesign.

Conception classique

La chaîne de conception classique d'un circuit peut se décomposer de manière succincte en trois grandes étapes : la description fonctionnelle, le placement-routage et le test.

- Description fonctionnelle : durant cette étape, le circuit est décrit en fonction de ce qu'il doit être capable de faire. Cette description contient les équations logiques décrivant le circuit. Elle contient aussi, de manière explicite, les choix matériels qui ont été faits : utilisation de tel ou tel type de mémoire, de telle ou telle fonction logique...etc.

  Il existe plusieurs possibilités pour réaliser cette phase, de nombreux langage peuvent être utilisés. Aujourd'hui l'utilisation du langage VHDL [10793] s'est imposée comme standard pour des raisons qui seront détaillées dans le paragraphe (IV.2.b).

  Ce code VHDL peut être validé par simulation. Il s'agit en général d'une simulation à événements discrets qui décrit le comportement concurrent du matériel.

  Une fois validé, le code VHDL est "synthétisé". Cette étape permet, à partir d'une bibliothèque de composants de transformer la description fonctionnelle du circuit en un schéma de portes logiques.
- Placement / Routage : le processus de placement essaie de trouver une affectation "optimale" pour chaque porte. Ce problème est NP-complet et sa résolution utilise typiquement des algorithmes de "recuit simulé", souvent très lourds en temps de calcul et aux résultats incertains. Le deuxième processus tente de trouver une configuration des ressources d'interconnexion du circuit permettant d'établir l'interconnexion désirée entre les cellules.

  Ces processus permettent de décider comment placer les différents blocs sur silicium. Ces étapes doivent elles aussi être validées par des simulations. Cette fois, il s'agit de simulations au niveau électrique. Elles permettent de voir si la manière dont les transistors ont été placés et reliés conduit aux résultats escomptés.

  Après validation, les dessins des masques de gravure sont fournis au fondeur qui construit le circuit à partir d'une galette de silicium.
- Test : La dernière phase est la phase de test, le circuit fourni par le fondeur est testé sur un banc de test. Plusieurs problèmes peuvent se poser, des problèmes électriques générés lors de la phase placement/routage ou des problèmes d'ordre plus fonctionnel générés lors de la phase de description.

  De la description à la phase de test, le cycle est long et coûteux. Les simulations électriques et fonctionnelles sont elles très coûteuses en temps machine, il existe donc un marché pour des architectures spécifiques capables d'accélérer ces simulations.

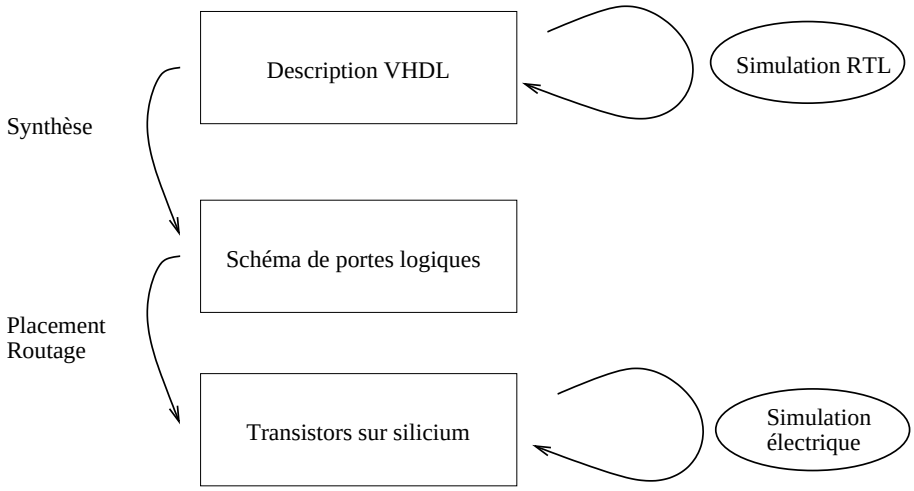

FIG. IV.1: La chaîne de conception classique comporte deux étapes de simulation. Ces simulations sont souvent très longues et très coûteuses.

Le co-design

Dans le contexte concurrentiel des circuits intégrés, la diminution des temps de production est particulièrement important, pour cela les développeurs ne sont pas censés attendre que la puce soit prête pour développer les logiciels qui vont fonctionner avec. Les circuits et les logiciels associés sont donc de plus en plus conçus en parallèle : c'est le co-dessign.

Dans ce cas, il est nécessaire de disposer de méthodes de conception qui soient résistantes au changement de spécification, mais surtout d'outil capable de simuler à grande vitesse un circuit intégré.

Dans ce genre de simulation c'est uniquement l'aspect fonctionnel du circuit qui est à prendre en compte. Là encore, des machines capables d'émuler le comportement d'un circuit sont particulièrement intéressantes.

L'utilisation de la machine Sim-express présentée dans le chapitre (VII) ne pourrait être qualifiée de co-design. Cette application qui couple aussi un émulateur de circuit est un logiciel soft doit plutôt être vue comme de la co-simulation. En effet, le but de la démarche présentée au chapitre (VII) n'est pas de réaliser un circuit mais d'accélérer la simulation d'un système complexe.

Accélération de la simulation par émulation

Lors du développement des gros circuits, ou des gros systèmes il n'est pas concevable d'utiliser des méthodes de simulation classiques. Il a donc fallu concevoir d'autre outils pour réaliser ces simulations pour gagner un temps précieux dans la conception des circuits.

Ainsi, il existe des machines permettant d'effectuer des simulations électriques accélérées. Ce type d'accélérateur est commercialisé notamment pas des entreprises comme Ikos [Iko99].

Il existe d'autre type "d'accélérateurs" basé sur des architectures reprogrammables. Ces accélérateurs sont des machines de simulation uniquement fonctionnelle. On configure des circuits FPGA (Field Programmable Gates Array) pour qu'il se comporte exactement comme le circuit que l'on souhaite émuler. Ces machines peuvent donc être vues comme un ensemble de ressources matérielles configurable à souhait. Elles sont utilisées dans plusieurs but. D'abord elles permettent de faire du débugage sur les fonctionnalités du circuit. Mais surtout ce type de simulateur peut être utilisé avec succès dans un contexte de co-dessign ou comme environnement de test [RKSC95, NT96, BRFL96].

Ce type de machine est commercialisé notamment par les sociétés Quickturn [qui99] et Metasystems (société de la corporation Mentorg Graphics [MG99]). La machine Sim-express vendue par la société Metasystems est présentée dans la section (IV.2).

IV.1.b Autres applications.

Recherche de séquences génétiques

La comparaison de séquences d'ADN (ou de séquences de protéines) est fondamentale en biologie moléculaire. L'objectif est de trouver des similarités (sous séquences communes) entre deux ou plusieurs séquences d'ADN (ou de protéines). Le problème consiste à explorer de grandes bases de données et à effectuer une comparaison entre la séquence de références et les éléments de la base. Pour améliorer les performances, une implantation des algorithmes de comparaison grâce à des ressources matérielles est particulièrement intéressante [LS95, Lav98].

L'algorithme classique pour réaliser cette recherche consiste à calculer une distance entre les séquences. Deux opérations élémentaires sont définies sur les séquences : la substitution et l'insertion/suppression. Grâce à une série de ces opérations unitaires, il est possible de transformer n'importe quelle séquence en une autre. Plus le nombre d'opérations à effectuer pour transformer une séquence en une autre est petit, plus ces séquences sont proches.

Considérons deux séquences $X = (x_1, ..., x_n)$ et $Y = (y_1, ..., y_m)$ à comparer. Notons $d(x_i, y_j)$ le coût de la substitution de x_i par y_j, et g le coût d'une insertion/suppression. On définit l'indice $H(i, j)$ qui augmente avec le nombre de similarités entre les deux séquences se terminant par x_i et y_j. On prend $g = -1$ et :

$$d(x_i, y_j) = \begin{cases} 2 & si\ x_i = y_j \\ -1 & sinon \end{cases}$$

L'équation à résoudre est alors :

$$H(i,j) = \max \begin{cases} 0 \\ H(i-1, j-1) + d(x_i, y_j) \\ H(i-1, j) - 1 \\ H(i, j-1) - 1 \end{cases} \qquad \text{(IV.1)}$$

avec $H(i, 0) = H(0, j) = 0$.

Cette équation peut être parallélisée en associant un élément de calcul pour chaque $H(i, j)$. On construit ainsi une matrice de $n * m$ processeurs connectés comme l'indique la figure (IV.2). D'autre part, à chaque instant les processeurs d'une diagonale effectuent un calcul. Il est donc possible de pipeliner plusieurs comparaisons ou d'utiliser uniquement un "vecteur" $\max(m, n)$ processeurs.

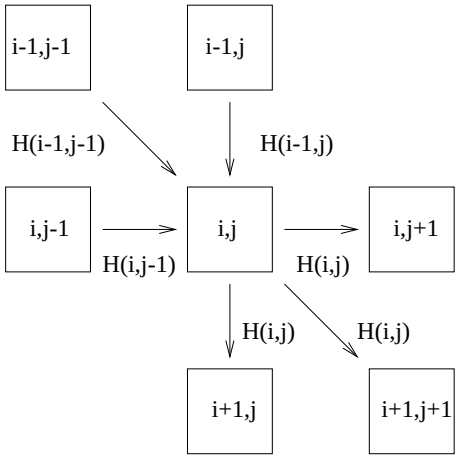

FIG. IV.2: flux des données dans un tableau de processeurs pour résoudre l'équation (IV.1). Le processeur (i, j) reçoit des données des processeurs $(i-1, j-1)$,$(i, j-1)$,$(i-1, j)$ et en envoie aux processeurs $(i+1, j+1)$,$(i+1, j)$,$(i, j+1)$.

Ce type de solution a été utilisé dans des machines commercialisées notamment par la société compugen [Com99]. Celle-ci vend deux générations d'accélérateurs de comparaison de séquences génétiques le Bioccelerator (1993) et le BioXL/P. L'utilité de la logique reprogrammable est que la structure de ces machines peut être adaptée à plusieurs algorithmes différents et ainsi être couplée aux différents logiciels utilisés par les chercheurs en biologie.

Entre les deux technologies la fréquence de l'horloge a peu évolué, par contre, le nombre de "processeurs" élémentaires a doublé. C'est un choix classique dans les applications des architectures reconfigurables qui utilisent le parallélisme. En effet, le gain de vitesse réalisé en augmentant le parallélisme est plus important que les gains réalisés en augmentant la fréquence de l'horloge.

D'autre part D.Lavenier (IRISA, Rennes) a mené plusieurs expériences dans ce domaine. La machine Samba [Lav98] utilise des circuits spécialisés et une implantation sur une architecture reconfigurable (carte PeRLe-1) a aussi été effectuée [VBR+96].

Le projet "Programmable Active Memories" (PAM)

Ce projet a donné lieu à de nombreuses expérimentations dans de nombreux domaines en utilisant les carte PeRLe-0 et 1 [VBR+96, Mol97, Ber93]. Les applications sont :

- RSA, Cryptographie [SV93, Sha97],
- Heat and Laplace équations,
- Réseaux de neurones [LLM95],
- Vidéo-compression,
- Physique de haute énergie : classification d'image, analyse d'image, détection de cluster,

- Acquisition d'images,
- Stéréo vision,
- Synthèse de sons,
- décodeur de viterbi.

Introduction du BOPS : Boolean Operations Per Second.

IV.1.c Utilisation pour la simulation de réseau

Les difficultés rencontrées pour la validation des protocoles de réseaux haut débit (voir chapitre II), justifie aussi le recours à des techniques d'accélération de simulations.

L'université de Santa-Cruz a développé dans ce but une plate-forme de test pour l'évaluation de protocoles ATM [SV97, Sti96]. Cette plate-forme intitulée FAST (FPGA-based ATM Simulation Testbed), utilise des circuits FPGA pour implanter et tester les algorithmes clefs des commutateurs ATM.

La première version de cette plate-forme a été construite pour simuler différentes politiques de service et de contrôle de flux.

Architecture

Cette plate-forme est constituée de quatre cartes FPGA "customizées", chacune ayant une capacité de 336.000 portes logiques et 17 Mbit de mémoire (static RAM). Chacune de ces cartes est utilisée pour modéliser un commutateur 4x4 et les algorithmes associés.

Le circuit utilisé comme brique de base est un circuit FPGA Altera [Alt99]. Chacune des quatre cartes est conçue pour modéliser un commutateur ATM. Les cartes peuvent être inter-connectées pour former la simulation d'un embryon de réseau. L'utilisation des circuits FPGA va permettre de tester différentes politiques de bufferisation (input / output) et de services (Roud Robin, Fair Queuing...).

Les sources de trafic peuvent être internes, avec des générateurs aléatoires, ou externes. L'architecture d'une carte est conçue pour simuler un commutateur $4x4$. Ainsi, une carte est composée de circuits reconfigurables Altera 81500 et 8050, destinés à jouer le rôle de tampons d'entrée, de sortie, de générateurs de trafic, et de politique de service.

Pour le contrôle de la simulation, une des cartes est maître et les autres esclaves. C'est la carte maître qui détermine le début de chaque slot.

Générateur de trafic

Les générateurs de trafic utilisent une implantation parallèle de l'algorithme de Tausworthe (III.2.b) pour avoir une séquence de mots uniformément distribués. Cette séquence peut être transformée en une autre distribution en utilisant une table mémoire. Les modules de génération de trafic sont donc formés d'un circuit FPGA et d'une mémoire.

L'injection d'un trafic extérieur est aussi possible. Dans ce cas les modules de génération de trafic sont utilisés dans un processus de communication et la mémoire est utilisée pour stocker les cellules ATM. Cette méthode utilise aussi une carte d'interface formée de six circuits Altera FLEX 815000, soit un total de 90000 portes logiques. Cette interface permet d'implanter un adaptateur qui intercepte le trafic réel d'entrée et génère uniquement ce qui est nécessaire pour la simulation (par exemple, des cellules ne contenant pas de charge utile).

Input / Output module

Les cartes contiennent aussi quatre modules d'entrée (input module) qui sont utilisés pour effectuer un éventuel tamponnage en entrée des commutateurs.

Les modules de sortie (output module) sont plus importants. C'est eux qui effectuent le multiplexage des cellules venant des différentes entrées. C'est aussi là que sont implantées les politiques de service. Chaque module de sortie est capable de recevoir 4 cellules en même temps, de les stocker, de choisir la cellule à servir et de diriger celle-ci vers la sortie. C'est dans ce module que se trouvent les fonctions les plus complexes.

Outil Logiciel

La configuration de la machine (choix des trafic d'entrée, politique de service) se fait au travers d'une description en VHDL. Une simulation fonctionnelle permet de valider les algorithmes sous leur forme matérielle. On effectue ensuite une synthèse avec pour cible la technologie FPGA. Les outils utilisés viennent de Altera et de Mentor Graphics.

Les outils de pilotage du simulateur FAST permettent d'accéder au contenu des mémoires et des registres des différents circuits. D'autre part, ce logiciel peut être adapté à l'architecture du commutateur simulé.

Un certain nombre de statistiques sont collectées par la machine hôte (une station de travail). Les résultats de la simulation sont enregistrés dans les mémoires des différents modules. La simulation est arrêtée, les résultats collectés, et la simulation est relancée. Les calculs sont fais "off line" par la machine hôte après simulation.

résultats

Les performances de la machine FAST sont particulièrement intéressantes. Cela découle du fait que la machine est construite et donc optimisée pour simuler des commutateurs.

Ainsi, chaque port de sortie peut gérer 1024 VC et la machine marche à 18 MHz. La vitesse de simulation est de l'ordre du tiers de la vitesse d'un lien ATM à 155 Mbits/secondes. Les comparaisons avec un simulateur logiciel basé sur CSIM ont montré un facteur d'accélération de 140 à 180. Ce test a été effectué pour la simulation de la politique de service weighted-round-robin avec de 4 à 32 VC. 10^6 cellules ont été simulées en 3 secondes contre 410-540 pour CSIM sur une DEC alpha 3000/400.

Pour des politiques de service plus complexes (Frame-based Fair Queuing) FAST simule 10^6 cellules en 12 secondes pour 256 VC. Une simulation de 1000 secondes pour uniquement 8 VC en CSIM permet d'obtenir les même résultats.

Cette machine est cependant peu évolutive, en effet, étant construite pour simuler des commutateurs 4x4, elle est difficilement utilisable dans un autre contexte.

IV.1.d Conclusion sur les systèmes à base de FPGA.

Cette section (IV.1) a permis de voir quelques applications et quelques types d'architectures reconfigurables. En fait, on peu distinguer trois types de machines à base de FPGA [Lis99].

Petites

Les petits systèmes reconfigurables peuvent être insérés directement dans un emplacement d'extension d'une station de travail ou d'un ordinateur personnel. Leur taille implique aussi un prix raisonnable, qui est un bon vecteur de propagation sur le marché. Parmi les applications de ces cartes ont trouve le prototypage rapide, l'acquisition de données et l'utilisation pour la formation dans les universités.

Ces petits systèmes sont en général composés d'une carte comportant un ou deux circuits FPGA couplés à des ressources d'interconection avec les machines hôtes. Un exemple type de ces systèmes sont ceux fournis par des constructeurs tels que Xilinx ou Altera.

Gros

Les gros systèmes reconfigurables sont composés de plusieurs cartes, ayant plusieurs circuits FPGA (au moins une dizaine de circuits). Ce type de cartes est utilisé pour le développement d'outils, et l'exploration des possibilités de calcul de ce genre de technologie, en particulier, pour le calcul intensif. C'est le cas des cartes Perle-0 et DECPeRLe-1 utilisées dans le projet PAM pour leurs évaluations dans différentes applications. La machine FAST peut aussi être classée ici.

Énorme

Le dernier type de machines est l'approfondissement des capacités de prototypage pour faire de l'émulation. Le principe est d'assembler des systèmes contenant des centaines de circuits, afin d'arriver à une capacité équivalente à celle d'un circuit sur mesure, d'une capacité de 500.000 à plusieurs millions de portes logiques. Ces systèmes ont en général une fréquence assez faible de l'ordre du Mhz. C'est le cas des machines des sociétés Quickturn et Metasystems pour l'émulation de circuits.

IV.2 La machine Sim-express

Ce paragraphe va permettre de présenter les outils qui seront utilisés pour la simulation de protocoles pour les réseaux haut débit à l'aide de ressources matérielles. Dans un premier temps, on se propose de détailler la structure matérielle de cette machine et d'expliquer son utilisation habituelle. Dans un deuxième temps, on verra les outils logiciels qui permettent de configurer et de piloter cette machine.

IV.2.a Architecture

La machine Sim-express est basée sur la répliquation d'un motif de base.

La brique de base : le Meta

Le circuit de base de la machine est nommé le Meta. Ce Meta est composé de 128 Blocs Logiques Programmables (BLP) ainsi que de la logique de commande pour les programmer et d'un réseau pour les interconnecter. Chaque BLP contient une RAM statique à 4 entrées

dont la sortie débouche sur un élément séquentiel programmable pouvant émuler un registre sur front ou sur état.

Contrairement à l'architecture Xilinx, chaque BLP peut être observé en temps réel à travers un réseau indépendant du système logique reprogrammable. Cette caractéristique importante évite de recompiler le circuit (reconfigurer la machine) lorsque l'on souhaite observer des registres qui n'auront pas été déclarés au préalable en tant que sonde.

Chaque carte logique est constituée de trois colonnes de huit Meta séparés par des colonnes de huit Matrices De Connexion (MDC). Le MDC est un circuit qui réalise une matrice partielle d'interconnexion sur 128 entrées / sorties. De plus à chacun des Meta est associée une mémoire vidéo dans laquelle est conservée la valeur des BLPs pendant les 7200 derniers cycles d'émulation.

Le fond de panier de la machine connecte 23 cartes logiques et une carte dédiée à la gestion des communications entre Sim-express et la machine hôte. Cette partie de l'architecture permet de disposer d'environ 500.000 portes logiques ou de 70.000 registres (voir figure IV.3). D'autre part, plusieurs machines de base peuvent être inter-connectées pour assurer l'émulation logique de circuits ayant plus de un million de portes.

Les mémoires

À chaque Meta est associé une mémoire de 32 Koctets. L'ensemble de ces mémoires peut être utilisé par l'utilisateur comme des mémoires de type varié : synchrones, asynchrones, simple port, double port... Et ceci sans restriction en adressage. L'émulateur met ainsi à disposition de l'utilisateur une capacité totale de 17 Moctets de mémoire.

Enfin, la machine Sim-express est pourvue d'entrées-sorties TTL (Transistor Transistor Logic) , programmables en entrée, en sortie ou bidirectionnelles. Cet accès peut être mis à profit pour utiliser des stimuli extérieurs. Par exemple, en provenance de la carte qui doit porter le circuit simulé.

Le tout

Sim-express est donc un émulateur construit pour répondre aux besoins de l'émulation logique. Cette machine peut être utilisée dans deux modes de fonctionnement. La vérification logique séparée et l'émulation in-circuit. Le premier mode peut-être vu comme une sorte de simulation en temps réel. Des stimuli sont alors stockés dans des mémoires et le comportement du circuit émulé est validé indépendamment du monde extérieur. Dans le second mode, la machine est directement connectée à la carte devant recevoir le circuit après fabrication. Ce mode permet de valider l'interaction du circuit émulé avec son futur environnement tout en observant et contrôlant l'état interne du circuit.

La machine est construite pour travailler de manière synchrone. Toutes les horloges sont dérivées d'une horloge maîtresse. Tant que le modèle émulé reste simple et utilise moins d'une carte, des vitesses de plusieurs MHz (jusqu'à 10 MHz) sont couramment atteintes. Dès que plusieurs cartes sont nécessaires, la vitesse tombe entre 1 et 2 MHz.

La technologie de cette machine est déjà rendue obsolète par la nouvelle génération d'émulateur Metasystems nommé Celaro (1998). En effet, la capacité de Celaro est 20 fois supérieure à la capacité de la machine Sim-express. Par contre, la fréquence de l'horloge n'a que peu changé avec une fréquence moyenne d'utilisation aux alentours de 2 MHz. Là

encore, le constructeur a considéré que le gain en rapidité d'émulation est moins important que le gain en capacité d'émulation.

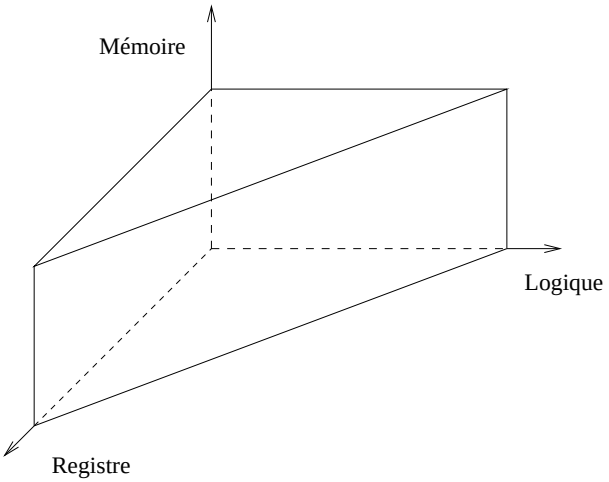

FIG. IV.3: la capacité de la machine Sim-express dans l'espace des ressources matérielles. La capacité d'une machine de deuxième génération est vingt fois supérieure pour une vitesse moyenne légèrement supérieure.

IV.2.b Chaîne logicielle

Aussi important que la configuration matérielle, la chaîne logicielle qui permet de configurer la machine se doit d'être simple et rapide. Ceci va permettre d'avoir un cycle de debug très rapide (de l'ordre de l'heure dans notre cas).

Le VHDL

Le choix du langage de description du circuit s'est porté sur VHDL (VHSIC Harware Description Language). En effet, VHDL a un certain nombre d'avantages. Outre le fait qu'il soit normalisé [10793] il permet une simulation à tous les niveaux. Un autre aspect très important est la possibilité de réutilisation des différents composants.

Le VHDL utilisé doit être du VHDL synthétisable. La synthèse est le processus qui transforme ou traduit une description dans un langage comportemental (comme VHDL) en une description structurelle dont chaque élément représente une ressource électronique de base prédéfinie. [ABO94, ABO98, ABO90, Per93].

Les caractéristiques les plus importantes qui font de VHDL un langage bien adapté à la modélisation et à la synthèse en particulier :

- Normalisation. Dès son lancement, le langage VHDL a été mis de façon ouverte sur le marché.
- Réutilisation. VHDL a plusieurs caractéristiques qui permettent de rendre réutilisables, donc plus générales, les descriptions matérielles : généricité, typage non contraint.

– Caractère général. Le langage VHDL est adapté aussi bien à la description du modèle en vue de la synthèse logique, qu'à la spécification fonctionnelle ou algorithmique ou encore à l'environnement de test. Autre intérêt, VHDL assure la continuité, tout au long du flot de conception, pour les validations par simulation.

De plus, le fait que VHDL soit utilisé pour décrire du matériel marchant de manière concurrente, il est particulièrement adapté à la description des problèmes parallèles, comme par exemple un ensemble de files d'attente. De nombreuses expériences ont été menées pour utiliser le VHDL, et les simulateurs associés, pour décrire des systèmes de files d'attente et des réseaux de Petri [MW95].

La synthèse synopsys

La synthèse est l'opération qui permet de passer de la description du circuit en langage VHDL à une description à base de portes logiques. Cette étape nécessite l'utilisation d'une bibliothèque dans laquelle sont décrites les portes logiques de base de la technologie utilisée.

Le langage VHDL n'a pas été défini pour la synthèse mais bien pour la simulation. Il en résulte que toutes les constructions VHDL ne peuvent pas être traduites en primitives matérielles. Cependant, un sous-ensemble s'est dégagé au fil du temps. Bien qu'aucun des sous-ensembles prônés par les vendeurs d'outils de synthèse ne repose sur une normalisation officielle de type IEEE, il faut admettre qu'il y a aujourd'hui quasi-identité au niveau du langage VHDL accepté pour la synthèse. La compatibilité s'est faite avec les outils de synopsys, le leader du marché qui a imposé son VHDL synthétisable comme une norme de fait.

La compilation Sim-express

La synthèse synopsys produit des fichiers de description du circuit que l'on souhaite porter sur la machine Sim-express. Les fichiers de configuration de la machine sont fournis par une compilation Sim-express. Cette compilation est automatique et sa vitesse est appréciable. Ce qui est important pour minimiser le temps du cycle Compilation-Emulation-Modifiction. C'est cette étape qui fixe la vitesse de l'horloge [BR96].

La première étape dans la compilation d'un circuit consiste à traduire la description du circuit (netlist) dans un format interne où les portes de la bibliothèque du client ont été substituées par des composants de base de la bibliothèque de la machine. La netlist est ensuite mise à plat puis partitionnée en deux niveaux de hiérarchie, les Metas puis les cartes. Finalement, le routage est réalisé sur trois niveaux d'interconnexion, entre les BLP, entre les Meta et enfin entre les cartes.

La phase de routage produit une configuration de la machine qui sera chargée avant d'effectuer l'émulation.

IV.2.c L'émulation, contrôle et vérification

Le contrôle de la machine

Une interface en Motif et un interpréteur de langage C sont fournis pour décrire chaque expérience d'émulation. Ce sont ces outils qui permettent un pilotage fin de la machine. Une expérience d'émulation définit les conditions dans lesquelles la machine fonctionne, c'est à

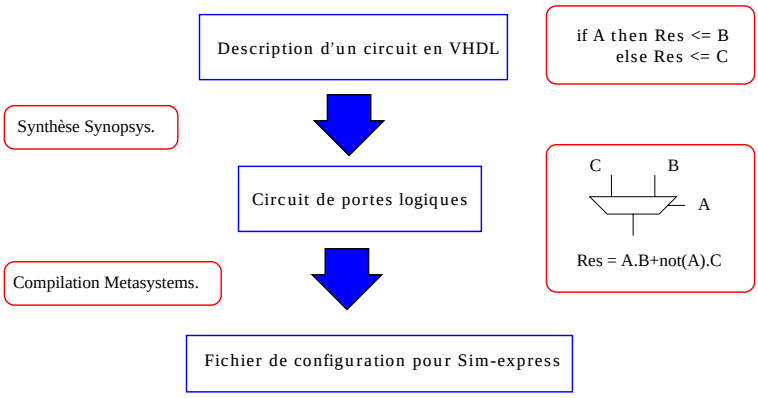

FIG. IV.4: la chaîne logicielle qui permet de configurer la machine Sim-express.

dire le nombre maximum de cycles, la vitesse maximale de fonctionnement, l'état initial des éléments séquentiels et les conditions d'arrêt. Une condition d'arrêt est définie en armant un détecteur qui teste si l'émulation amène un ou plusieurs BLP dans un état prédéfini. Non seulement les détecteurs peuvent être changés sans recompiler la configuration de la machine, mais en plus n'importe quel BLP peut être utilisé dans une condition d'arrêt.

IV.2.d Conclusion

La machine Sim-express et les outils logiciels qui l'accompagnent sont particulièrement adaptés à la modélisation. L'ensemble constitue un "General purpose emulator" qui permet de configurer la machine pour n'importe quelle simulation. Il ne s'agit pas d'une machine dédiée, les performances obtenues seront donc moindres que celles obtenues à l'aide d'une machine comme FAST. Cependant l'utilisation d'un émulateur non spécialisé offre un spectre d'utilisation beaucoup plus large. D'autre part, la chaîne logicielle particulièrement efficace permet une utilisation relativement facile.

Un autre grand avantage est l'utilisation du langage VHDL et d'un certain nombre de fonctionalités de Sim-express (en particulier la modularité). Tout ceci contribue à la formation d'un environnement où les composants développés sont très facilement réutilisables pour la construction d'autres composants. Ceci à la fois au niveau de la description VHDL que de l'implantation en machine.

IV.3 Exemple : la file Geo/Geo/1/k

L'objectif de ce paragraphe est de présenter l'utilisation des outils logiciels et matériels au travers d'un exemple. On a choisi comme exemple l'implantation de la file d'attente $Geo/Geo/1/k$. Avec λ et μ les paramètres des processus géométriques d'arrivées et de services. L'étude du taux d'utilisation de cette file sera l'objet de la simulation.

IV.3.a Le code / réalisation

La file $Geo/Geo/1/k$

La première question est de savoir quelle représentation de la file d'attente choisir. Sachant que l'objectif fixé est d'étudier le taux d'utilisation de la file $Geo/Geo/1/k$, il est inutile de différencier différents types de clients, l'information "nombre de clients présents dans le système" est suffisante. Le modèle de file "vide" présenté au chapitre (III.3) est donc le plus adapté. D'autre part, on choisit de se placer dans un modèle AF.

Voici une interface de file d'attente assez générale, permettant d'étudier la plupart des indices de performance classiques. Les noms à gauche des symboles `=>` sont les "broches" du composant, à leur droite se trouvent les noms des fils que l'utilisateur vient y connecter.

```
-- L'instantiation TheFifo du composant fifoAF.
TheFifo : fifoAF
   port map(CK   => CK,              -- in signal
            init => marche,          -- in signal (for init)
            k    => capa,            -- in signal (capacity)
        arrivees  => emission,       -- in (new customer)
         pertes   => loss,           -- out (loss customer)
      consommation => consumption,    -- in (consuption)
       occupation => N_customer,     -- out (number of waiting customer)
            out  => served_customer );-- out (served customer)
```

Les paramètres d'entrées de cette file sont :

- l'horloge `CK` (un bit). Ici le top horloge correspond au slot. Sur chaque front montant, la file change d'état.
- le signal `Init` (un bit). Ce signal permet l'initialisation ou le "gel" de la file d'attente.
- la capacité `k` codée sur plusieurs bits. C'est l'entier qui donne la taille de la file d'attente.
- les arrivées `arrivees` éventuellement codées sur plusieurs bits. C'est le nombre de clients qui arrivent pendant un slot. Ils doivent être ajoutés à ceux déjà présents dans le système. Dans le cas de la file $Geo/Geo/1/k$ ce signal peut prendre les valeurs zéro ou un. Le nombre d'arrivées est donc codé sur un seul bit.
- la consommation `consommation` éventuellement codée sur plusieurs bits. C'est le nombre de clients qui doivent être servis pendant le slot. Ils doivent être retranchés au nombre de clients présents dans le système. Pour la file $Geo/Geo/1/k$ il peut y avoir zéro ou un service. On utilise donc un codage sur un bit.

Un certain nombre d'informations doivent être extraites du système. Ce sont ces sorties qui seront connectées au(x) composant(s) d'instrumentation. Les paramètres de sorties de la file sont :

- les pertes `pertes`, un entier qui donne le nombre de pertes ayant eu lieu durant le slot courant. Dans le cas de la file $Geo/Geo/1/k$ il peut y avoir zéro ou une perte par slot. Un bit suffit donc pour coder ce signal.
- l'occupation `occupation` est un entier donnant le nombre de clients présent dans le système. C'est cette information qui est intéressante pour l'étude du taux d'utilisation de la file d'attente.
- le flux des clients servis `out`. Ce signal de sortie permet de construire des réseaux de files d'attente. Le flux de sortie peut ainsi être utilisé comme processus d'arrivée pour une autre file.

Les processus géométriques

Pour les processus de service et d'arrivée on utilise le générateur présenté au paragraphe (III.2.b). Les entrées de ce composant, qui permet de simuler un processus géométrique, vont être les taux d'arrivée λ et de service μ, une horloge et un signal d'initialisation.

```
-- instantiation Input_Traffic du composant Bernouilli_Process.
-- Geom(lambda).
Input_Traffic : Bernouilli_Process
   port map(CK       => CK,           -- in signal
            reset    => reset,        -- in signal (for init)
            seed     => germe1,      -- in signal (for init)
            rho      => lambda,       -- in signal (for comp)
            emission => emission );   -- out (new customer)

-- instantiation Service du composant Bernouilli_Process.
-- Geom(mu).
Service : Bernouilli_process
   port map( CK       => CK,           -- in signal
             reset    => reset,        -- in signal (for init)
             seed     => germe2,       -- in signal (for init)
             rho      => mu,           -- in signal (for comp)
             emission => consumption );-- out signal
```

Les composants `Bernouilli_Process` sont eux-mêmes composés de plusieurs "briques". Ils comprennent un processus qui initialise les mémoires des générateurs pseudo-aléatoires[1]. Les valeurs fournies par le générateur sont comparées aux valeurs `lambda` et `mu`. Ces composants comportent donc, outre l'automate d'initialisation, une mémoire, et un comparateur.

L'instrumentation

On souhaite étudier le taux d'utilisation de la file. Le plus simple pour cela est de comptabiliser le nombre de fois que la file est vide, on obtient ainsi le taux de "non utilisation" de la file. Pour réaliser cette comptabilisation, une instrumentation de type "comptabilisation" suffit. Le nombre de slots écoulés ainsi que le nombre de fois que la file d'attente est vide doivent être comptabilisés.

L'instrumentation sera donc constituée deux registres et de deux incrémenteurs.

Le composant "expérience"

L'ensemble de ces composants est regroupé dans un composant permettant de faire des expériences. Ce composant est constitué des générateurs de la file et de l'instrumentation. La figure (IV.5) montre de manière imagée l'utilisation et les connexions entre les différents composants.

[1] Cette initialisation se fait à partir du germe sur 32 bits et du polynôme $1 + X^3 + X^{31}$. Ce générateur fournit des mots pseudo-aléatoires qui sont écrit dans les mémoires.

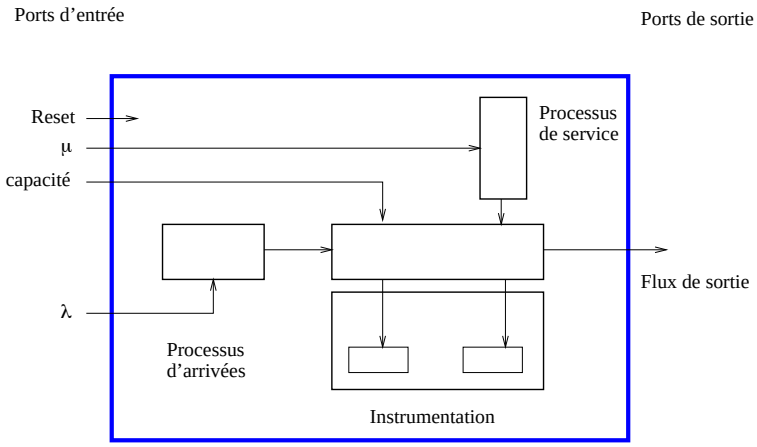

FIG. IV.5: le composant permettant de faire l'expérience utilise les composants de génération de lois de Bernouilli, une instrumentation et un composant pour comptabiliser le nombre de clients présents dans le système.

```
use_package Files_dAttente.pkg
entity Test_Geo_Geo is
       port(          reset : in bit ;
                     germe1 : in vecteur32bits  ;
                     germe2 : in vecteur32bits ;
                     lambda : in proba   ;
                         mu : in proba   ;
                       capa : in vecteur16bits  );
end Test_Geo_Geo;

Architecture A of Test_Geo_Geo is

signal CK : bit       ;
signal emission, consuption, loss : bit         ;
signal N_customer, served_customer : vecteur16bits ;

begin

 -- Clock generation
 clock:meta_ckpad
    port map(CK);                  -- CK is the clock, out signal

 -- instantiation Input_Traffic du composant Bernouilli_Process.
 -- Geom(lambda).
 Input_Traffic : Bernouilli_Process
    port map(CK       => CK,           -- in signal
             reset    => reset,        -- in signal (for init)
```

```
            rho      => lambda,       -- in signal (for comp)
            seed     => germe1,       -- in signal (for init)
            emission => emission );   -- out (new customer)

 -- instantiation Service du composant Bernouilli_Process.
 -- Geom(mu).
 Service : Bernouilli_process
    port map( CK       => CK,           -- in signal
              reset    => reset,        -- in signal (for init)
              rho      => mu,           -- in signal (for comp)
              seed     => germe2,       -- in signal (for init)
              emission => consumption );-- out signal

 -- L'instantiation TheFifo du composant fifoAF.
 TheFifo : fifoAF
port map(CK   => CK,              -- in signal
         init => reset,           -- in signal (for init)
         k    => capa,            -- in signal (capacity)
    arrivees  => emission,        -- in (new customer)
     pertes   => loss,            -- out (loss customer)
  consommation => consumption,     -- in (consuption)
   occupation => N_customer,      -- out (number of waiting customer)
         out  => served_customer );-- out (served customer)
```

Il faut noter l'utilisation d'un package dans lequel sont décris les interfaces des composants ainsi que les différents types utilisés.

Le code VHDL est synthétisé, on obtient ainsi une description à base de ressources matérielles. Ceci est réalisé grâce aux outils synopsys. La figure (IV.6) montre le résultat d'une petite partie de l'instantiation `TheFifo` du composant `FifoAF`.

Après cette étape de synthèse, la compilation Sim-express fournit la configuration adéquate de la machine. Des expériences sont alors réalisables au cours desquelles des données seront collectées par l'instrumentation . Le waveform de Sim-express permet la visualisation de la valeur de tous les nœuds du circuit.

Un exemple de waveform est visible sur la figure (IV.8). Celui-ci correspond à la simulation de la file $Geo/Geo/1/k$. La fenêtre "Monitor" permet de piloter les entrées du composant "expérience". On voit apparaître ici le signal d'initialisation `reset`, le signal donnant la capacité de la file `capa` et l'un des germes `germe`. La fenêtre "WaveForm" fait apparaître, entre autre, les signaux d'arrivée `arrivees`, de départ `consommation`, d'occupation `occupation`, de pertes `pertes`.

IV.3.b Les résultats

La charge du système est donnée par : $\rho = \frac{\lambda}{\mu}$. Prenons λ' et μ' tels que :

$$\lambda' = \frac{\lambda}{n}, \quad \mu' = \frac{\mu}{n}.$$

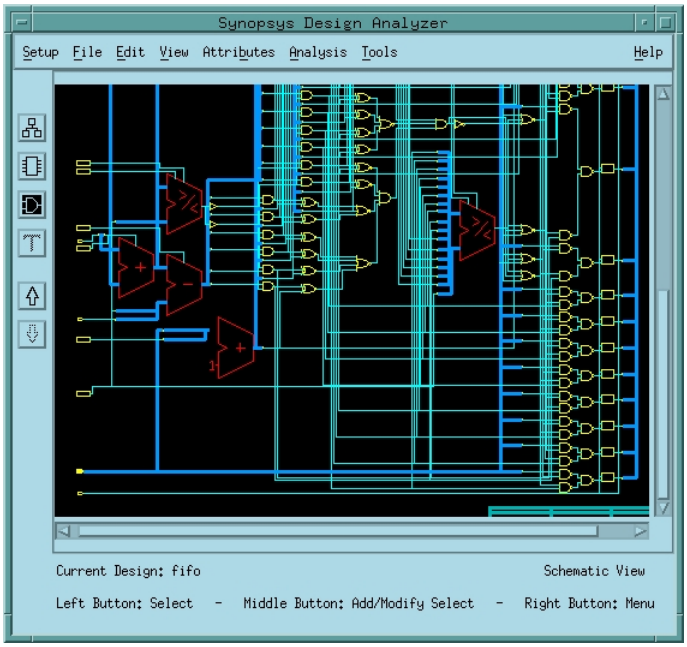

FIG. IV.6: Le résultat de la synthèse est une description à l'aide de ressources matérielles de l'algorithme décrit en VHDL.

Si n augmente, la charge ρ reste constante. On peut considérer $\frac{1}{n}$ comme la taille du slot. En effet, si λ' et μ' diminuent, tout en conservant une charge ρ constante, tout ce passe comme si la taille du slot avait été réduite. Augmenter n a donc le même effet que decrémenter la taille du slot. En faisant tendre n vers l'infini, on se rapproche du modèle continu.

La figure (IV.7) montre la probabilité empirique π_0 que la file soit vide en fonction de n. Cette probabilité est calculée à partir des résultats fournis par l'instrumentation. On voit que cette probabilité π_0 tend vers la valeur $1 - \rho$.

Cette valeur n'est pas surprenante, en effet les formules (II.4) et (II.5) permettaient de prévoir ce résultat. On a ainsi vérifié de manière expérimentale que le modèle asymptotique de la file $Geo/Geo/1/k$ est bien la file $M/M/1/k$ (voir [GH92] et II.1).

IV.3.c Conclusion

Il est donc possible d'utiliser des ressources matérielles pour l'évaluation de performances. De plus, l'environnement logiciel et matériel utilisé (VHDL et Sim-express) permet une grande portabilité des composants developpés. Ceux-ci peuvent être très facilement réutilisés.

Cette possibilité d'utilisation de ressources matérielles va permettre l'étude des protocoles pour les réseaux à haut débit. On espère ainsi mettre en évidence un certain nombre de phénomènes difficilement capturables à l'aide des outils classiques d'analyse.

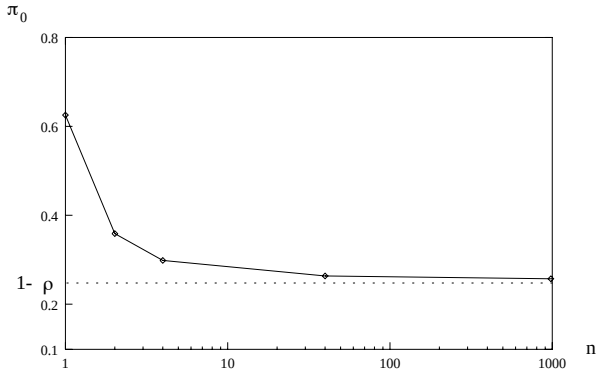

FIG. IV.7: la file $M/M/1/k$ est le modèle asymptotique de la file $Geo/Geo/1/k$. Quand le temps discret tend vers le temps continu la probabilité π_0 que la file soit vide tend vers la probabilité que la file $M/M/1/k$ soit vide.

FIG. IV.8: Visualisation du circuit. On voit ce qui ce passe grâce au waveform. La fenêtre "Monitor" permet de piloter les signaux d'entrées et l'horloge du composant que l'on émule. La fenêtre "WaveForm" permet de visualiser tous les nœuds du circuit. Ici on visualise les signaux d'un composant simulant une file $Geo/Geo/1/k$.

Bibliographie

[10793] IEEE Std 1076-1993. *IEEE Standard VHDL Language Reference Manual*. IEEE, Septembre 1993.

[ABO90] R. Airiau, J.-M. Berge, and V. Olive. *VHDL du langage à la modélisation*. Presses polytechniques et universitaires romandes, France Telecom, 1990.

[ABO94] R. Airiau, J.-M. Berge, and V. Olive. *Circuit Synthesis with VHDL*. Kluwer Academic Publishers, France Telecom, 1994.

[ABO98] R. Airiau, J.-M. Berge, and V. Olive. *VHDL language, modélisation, synthèse*. Presses polytechniques et universitaires romandes, France Telecom, 1998.

[Alt99] *http ://www.altera.com*. Altera, 1999.

[Ber93] Patrice Bertin. *Mémoire actives programmables : conception, réalisation et programmation*. PhD thesis, Université Paris 7, 1993.

[BR96] L. Burgun and F. Reblewski. Première génération d'emulateurs matériels metasystems. In *Quatrième symposium sur les architectures nouvelles de machines*, Metasystems, France, 1996.

[BRFL96] L. Burgun, F. Reblewski, G. J. Fenelon, Barbier, and O. Lepape. Serial fault emulation. In *Proceedings of the 33rd Design Automation Conference 1996 (DAC 96)*, pages 801–806, Metasystems, France, 1996.

[Com99] *http ://www.compugen.co.il/products*. 1999.

[GH92] A. Gravey and G. Hébuterne. Simultaneity in discrete-time single server queues with Bernouilli inputs. *Performance Evaluation North-Holland*, 14 :123–131, 1992.

[Iko99] *http ://www.ikos.com*. Ikos, 1999.

[Lav98] Dominique Lavenier. Speeding up genome computation with a systolic accelerator. *Society for Industrial and Applied Mathematics (SIAM News)*, 31-8, 1998.

[Lis99] *http ://www.io.com/ guccione/HW_list.html*. 1999.

[LLM95] L. Lundheim, I.C. Legrand, and L. Moll. A programmable active memory implementation of a neural network for second level triggering in atlas. In *Proceedings of AIHEN'95*, 1995.

[LMV98] C. Labbé, S. Martin, and J-M. Vincent. A reconfigurable hardware tool for high speed network simulation. In *International Conference on Modelling Techniques and Tools for Performance Evaluation, Performance TOOLS'98*, 1998.

[LS95] E. Lemoine and J. Sallantin. Un système reconfigurable dédier à la comparaison de séquence génétiques. *Rairo : technique et science informatiques*, 14-1, 1995.

[MG99] *http ://www.mentorg.com*. Mentor Graphics, 1999.

[Mol97] Laurent Moll. *Application des Mémoires Actives Programmables.* PhD thesis, École Polytechnique, 1997.

[MW95] Sidharta Mohanty and Philip A. Wilsey. Rapid system prototyping, system modeling, and analysis in a hardware-software environment. In *IEEE Rapid Systems Prototyping Workshop*, pages 154–160, 1995.

[NT96] Huy Nam NGUYEN and Michel THILL. Design verification based on hardware emulation. In *Seventh IEEE International Workshop on Rapid System Prototyping*, BULL S.A., 1996.

[Per93] Douglas L. Perry. *VHDL.* McGraw-Hill Series on Computer Engineering, 1993.

[qui99] *http ://www.quickturn.com.* Quickturn, 1999.

[RKSC95] O. Rasmont, J. Koulischer, J. Schaumont, and R. Crappe. Testing and optimizing a scale reduction algorithm for a multi-screen video wall application on the meta-100 asic emulator. In *Sixth IEEE International Workshop on Rapid System Prototyping*, 1995.

[Sha97] Mark Shand. A case study of algorithm implementation in reconfigurable hardware and software. In *Proceedings of the 7th International Workshop, FPL'97, Lecture notes in Computer Science. Springer-Verlag.*, Digital Equipment Corporation, Palo Alto, 1997.

[Sti96] Dimitrios Stiliadis. *Traffic Scheduling in Packet-Switched Networks : Analysis, Design, and Implementation.* PhD thesis, University of California Santa Cruz, 1996.

[SV93] M. Shand and J. Vuillemin. Fast implementations of RSA cryptography. In *11th IEEE Symposium on COMPUTER ARITHMETIC*, 1993.

[SV97] D. Stiliadis and A. Varma. A reconfigurable hardware approach to network simulation. *ACM Transaction on Modeling and Computer Simulation*, 7, 1997.

[VBR+96] J. Vuillemin, P. Bertin, D. Roncin, M. Shand, H. Touati, and P. Boucard. Programmable active memories : Reconfigurable systems come of age. *IEEE Transactions on VLSI Sytems*, 4-1, 1996.

Troisième partie

Applications

L'objectif de cette partie est de montrer que la méthode présentée dans la deuxième partie a été utilisée avec succès. On a tiré profit de la souplesse de l'architecture pour résoudre plusieurs problèmes de nature différente. Ainsi, trois applications ont été testées pour mettre à l'épreuve l'approche proposée. Ces trois réalisations ont toutes pour objectif l'analyse de protocoles réseaux.

Le chapitre (V) propose l'étude d'une modélisation assez simple des commutateurs ATM sous forme de réseaux multi-étages. L'examen de ces modèles, a permis de montrer que l'évaluation d'évènements rares est possible grâce à l'utilisation de ressources matérielles. On a ainsi mesuré de manière précise des taux de pertes réalistes.

Le chapitre (VI) a pour objet l'analyse des politiques de Fair Queuing. Ce chapitre permet de montrer qu'il est possible de réaliser des simulations de service complexe à l'aide de ressources matérielles. De plus, ce travail a permis de mettre à jour l'importance de la méthode de choix en cas d'égalité des marques.

Enfin, le chapitre (VII) propose une modélisation beaucoup plus complète d'un commutateur. Cette modélisation a pour objectif de permettre le test et la mise au point d'un régulateur de trafic. La machine Sim-express va donc être couplée à une autre application et être utilisée comme un accélérateur de simulation. On montre ainsi que l'utilisation de ressources matérielles permet d'étudier l'impact de décisions prises à une échelle de temps supérieur sur les phénomènes du niveau cellules.

Chapitre V

Performance de réseaux multi-étages (Taux de perte)

Ce chapitre présente une série de simulations réalisées à l'aide de ressources matérielles. Ces simulations ont pour objectif l'étude des performances d'un réseau de commutateurs. Ce type de réseaux a fait l'objet de nombreuseux travaux [Tru95, FMP97, BM93, Bey93, Abu98, BKYB95]. En effet, la structure simple du système examiné permet d'effectuer un certain nombre de calculs analytiques.

L'objectif principal est d'étudier des taux de pertes et les perturbations introduites, sur la structure du trafic, par le passage dans les files d'attente des commutateurs. L'une des grandes difficultés est d'estimer de manière correcte des taux de pertes réalistes (de l'ordre de 10^{-9}).

Ce chapitre se décompose en quatre sections. Dans un premier temps (V.1), la modélisation du système est présentée. L'implantation de ce modèle dans la machine Sim-express est détaillée dans la section (V.2). La section suivante (V.3) tente de répondre au problème de l'arrêt des simulations et réalise une petite étude sur le coût de ces simulations. Enfin, la dernière partie (V.4) est consacrée à l'analyse des résultats récoltés à l'aide de l'instrumentation.

V.1 Modélisation et résultats existants

V.1.a Modélisation de commutateurs

Le commutateur

L'objectif est d'analyser le comportement d'un commutateur composé de plusieurs éléments de commutation de base. L'architecture d'un élément de base est composée d'une matrice d'interconnexion et de tampons. Cette matrice est capable de commuter, sans blocage, toutes les entrées vers toutes les sorties en un temps cellule (un slot). L'élément de base doit donc gérer les conflits entre les cellules souhaitant partir sur la même ligne de sortie.

Il existe deux manières de procéder pour résoudre ce type de conflits (voir figure V.1) :

- On peut bloquer les cellules en entrée de l'élément de commutation (input buffering) et commuter les cellules en fonction des possibilités de routage. Cette méthode entraîne une baisse importante de l'utilisation de la bande passante. Il est possible de remédier à ce problème par un certain nombre de méthodes plus ou moins efficaces [YHP98, HYP98].

– L'autre option est de commuter toutes les cellules et de résoudre les conflits en utilisant des tampons de stockage en sortie de la matrice de commutation (output buffering). Dans ce cas, la matrice d'interconnexion doit être non bloquante. C'est cette architecture qui sera utilisée ici.

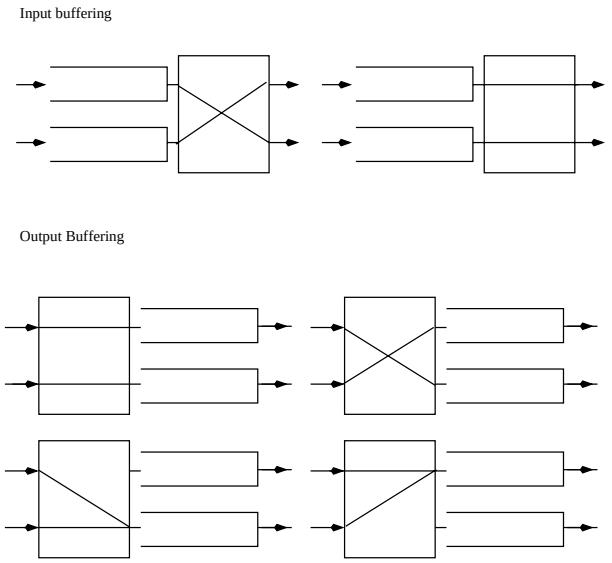

FIG. V.1: Deux types d'éléments de base possibles selon la politique de gestion des conflits (input ou output buffering) qui est choisie. L'option "input buffering" permet l'utilisation d'une matrice de connexion bloquante, ce qui n'est pas le cas de l'option "output buffering".

Les éléments de commutation sont ensuite connectés entre eux, selon une architecture de type Clos [GRI81]. Ainsi, à partir d'éléments de base réalisant la commutation de n entrées vers n sorties (nxn) on peut construire un commutateur réalisant la commutation de n^i entrées vers n^i sorties (n^ixn^i). i est le nombre d'étages d'éléments de base nécessaires pour la construction de ce commutateur. Un étage est défini, par la distance de l'élément de base à la source. Tous les éléments de base qui sont à une même distance des sources sont sur un même étage (voir figure V.2). L'assemblage ainsi obtenu peut être considéré comme la modélisation d'un commutateur multi-étages, ou comme la modélisation d'un réseau quelconque formé de plusieurs commutateurs.

Les sources

Le choix des caractéristiques des sources est particulièrement important. En effet, le type de trafic influe sur les résultats au même titre que les politiques de service ou de tamponage. Le choix du type de trafic à utiliser pour une simulation est extrêmement varié. Le plus simple, du point de vue analytique est de considérer un trafic ayant de bonnes propriétés mathématiques, comme par exemple le trafic uniforme. Du point de vue pratique, l'utilisation de traces en provenance d'un trafic réel peut permettre de contourner la difficulté de modélisation du trafic (cf II.3). Aucune impossibilité technique dans l'utilisation de la

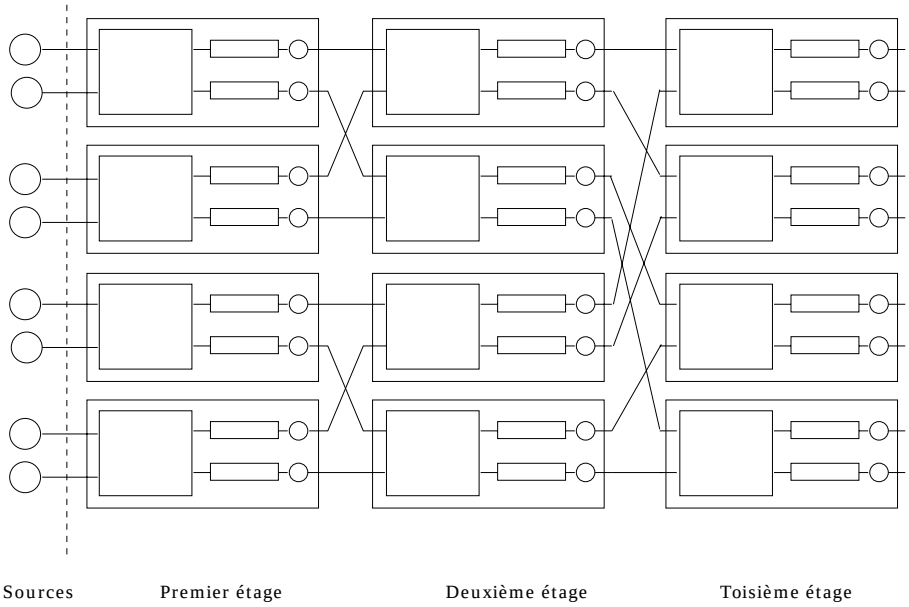

FIG. V.2: Un commutateur 8x8 formé de trois étages d'éléments de commutation 2x2. Dans notre cas les 8 sources émettent un trafic uniforme (processus géométrique et équiprobabilité de routage vers les différentes sorties possibles).

machine Sim-express s'oppose à l'utilisation de cette deuxième solution. Il aurait même été possible de la connecter directement à un flux réel. C'est cependant l'utilisation de sources facilement caractérisables analytiquement qui a été retenue. En effet, alors que l'approche analytique permet de tester un "type", une "classe" de trafic, l'approche utilisant des traces est beaucoup plus restrictive, elle ne permet de tester qu'un certain nombre de cas particuliers.

Parmi tous les choix analytiques possibles, le trafic uniforme est le plus souvent utilisé dans les études sur les réseaux multi-étages. En effet, l'avantage de la structure très régulière du système permet d'utiliser une résolution markovienne. Pour ne pas perdre cet avantage de calculabilité, le choix du processus géométrique est particulièrement intéressant.

L'utilisation de modèle de type MMBP, ou même un processus géométrique contraint par un Leaky-Bucket, complexifient les calculs [FPT95, FMH93].

V.1.b Résultats existants, premier et deuxième étage

Approches analytiques

Les approches analytiques permettent de calculer de manière exacte le taux de perte au premier étage [Tru95, Bey93].

Par contre l'impossibilité de caractériser le trafic en sortie du premier étage, pose un certain nombre de problèmes. Si l'on considère un réseaux de 24 files d'attente (figure V.2) de capacité 25 cellules, le nombre d'états possible du système est de l'ordre de 10^{34}. Calculer l'état stationnaire d'une telle chaîne de Markov sans la simplifier est illusoire.

Ainsi, il semble que l'on soit dans l'impossibilité de trouver de manière analytique des résultats exacts sur les autres étages.

Il faut alors procéder par agrégation d'états pour réduire la taille de l'espace d'état du système. Une autre solution consiste à procéder par encadrement stochastique [Abu98, Tru95, FMP97].

Simulation

La simulation est possible, soit à l'aide de simulateurs classiques de réseaux de files d'attente (Qnap ou SES-Workbench), soit à l'aide d'un simulateur ad-hoc (programmation en C). Le problème est que le nombre de files d'attente à traiter et le nombre d'événements à simuler, pour capturer des taux de pertes réalistes, conduisent à l'allongement des simulations et à l'impossibilité d'obtenir des estimations fiables.

V.1.c Conclusion

Ce problème est typiquement l'un de ceux où l'utilisation de ressources matérielles peut être intéressante. En effet, le matériel permet d'utiliser le parallélisme intrinsèque du problème. De plus les opérations à effectuer sont simples à réaliser à l'aide de matériel : génération aléatoire et tamponnage dans des files d'attente. D'autre part, l'étude des événements rares nécessite la génération d'un grand nombre d'événements.

Pour ces raisons, l'utilisation de ressources matérielles peut permettre de réussir là où les autres méthodes ont échoué.

V.2 Implantation

Comme l'a montré la section (III.3), plusieurs choix sont possibles pour l'implantation d'un réseau de files d'attente en utilisant des ressources matérielles. Ces choix doivent être effectués en fonction des objectifs fixés par l'étude. Ici, les taux de perte et l'étude des perturbations introduites dans le trafic par le commutateur sont les indices de performance que l'on souhaite étudier.

V.2.a Les files d'attente

Sans contenu

Le modèle de file sans contenu permet, en particulier, l'étude des taux de perte sur les différents étages du commutateur. De plus ce modèle peut aussi être utilisé pour l'observation des temps d'inter-arrivées entre les cellules à la sortie du commutateur, à condition de ne pas faire de distinction sur la provenance des cellules. Ces observations permettent de mettre en correspondance les temps d'inter-arrivées entre les cellules à l'entrée et à la sortie du commutateur.

Avec contenu

Le modèle sans contenu ne permet pas de différencier la provenance des cellules lorsque l'on se place en sortie du commutateur. Ainsi, pour dissocier les flux les uns des autres, l'utilisation d'un modèle où les cellules transportent de l'information est nécessaire. Ce modèle qui traite de vraies cellules sera utilisé pour étudier les perturbations introduites par un trafic de fond sur une connexion identifiée. Les cellules contiennent uniquement leur destination de sortie et une marque permettant de savoir à quel type de trafic elles appartiennent. Par exemple, pour un réseau 4x4, les cellules sont composées de 4 bits. Deux d'entre eux sont utilisés pour indiquer la destination de la cellule et les deux autres permettent d'identifier sa provenance.

V.2.b Les sources et le routage

Pour les sources, la technique de génération aléatoire présentée au paragraphe (III.3) est utilisée. Le routage est aussi effectué à l'aide de générateurs pseudo-aléatoires (GPA). Dans le cas du modèle sans contenu, les GPA effectuent directement le routage. Pour le modèle avec contenu, ils servent à remplir le champ des cellules réservées au routage.

V.2.c Instrumentation

Pertes

L'étude du taux de perte dans le système va principalement demander un comptage d'évènements. Ainsi, plusieurs registres sont utilisés pour compter les pertes qui surviennent à différents endroits du réseau de file d'attente.

Par exemple, un registre comptabilise toutes les pertes ayant lieu au premier étage, un autre est utilisé pour comptabiliser le nombre total de cellules émises. On peut ainsi calculer le taux de perte sur un étage, ce qui donne une estimation de la probabilité du taux de perte sur un tampon de cet étage. Le taux de perte sur l'ensemble du commutateur et sur les différents étages du système peuvent également être calculés.

Trafic

L'observation des caractéristiques du trafic nécessite l'étude de la distribution, et donc la création d'histogrammes. Plusieurs composants d'instrumentation vont être utilisés.

L'un compte les tailles des silences entre les cellules, il permet ainsi d'estimer la distribution de la durée des inter-arrivées. Un autre composant compte les tailles des rafales de cellules (cellules collées), il permet d'estimer la distribution de la taille des rafales. Dans le cas du modèle de file utilisant des cellules avec contenu, ce type de composant peut être utilisé pour étudier la structure d'un type de flux précis. Par exemple, l'étude de la taille des silences entre les cellules en provenance d'une source spécifique.

V.2.d Le composant "expérience"

Ces différents composants sont donc assemblés pour former un composant qui modélise un commutateur multi-étage. La figure (V.3) montre l'assemblage pour un commutateur 4x4 formé d'éléments de base 2x2.

La machine Sim-express est configurée pour émuler ce composant. Les paramètres d'entrées de la simulation sont :

– les paramètres des processus géométriques,

– la taille des files d'attente.

Les résultats récoltés dépendent de l'instrumentation utilisée. Typiquement ces résultats permettent de calculer des taux de perte ou des statistiques sur la structure du trafic.

Il reste à savoir quand arrêter la simulation, c'est-à-dire savoir dire à quel moment les informations récoltées par l'instrumentation vont fournir une précision suffisante.

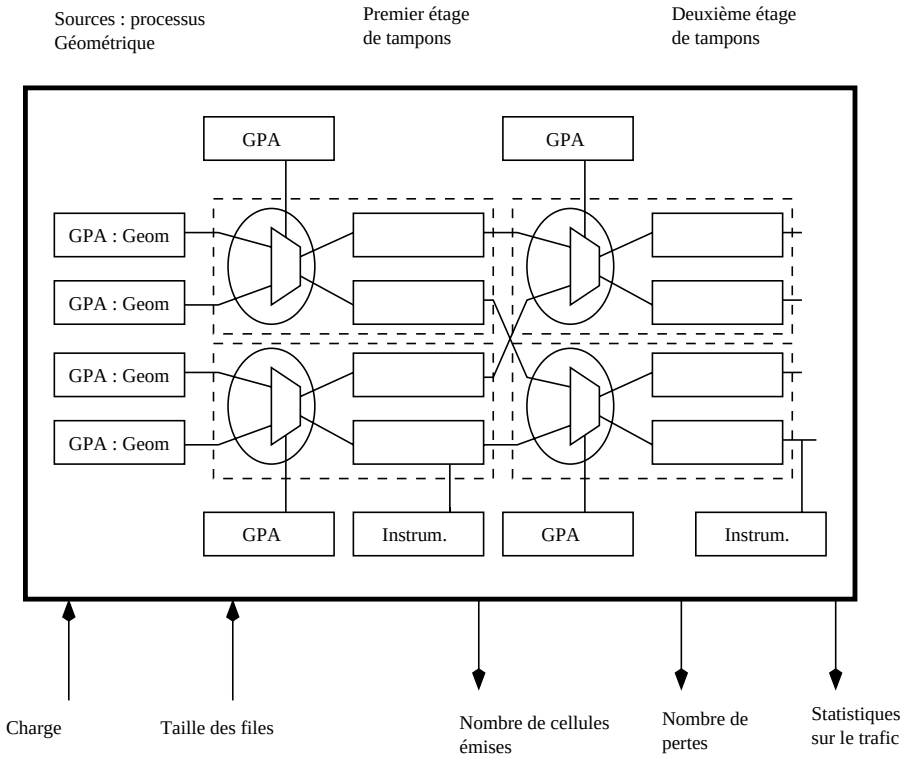

FIG. V.3: Un commutateur 4x4 formé de deux étages d'éléments de commutation 2x2 modélisé à l'aide de composants à base de ressources matérielles. Cette implantation est celle correspondant au modèle sans contenu. Une implantation avec contenu effectue le routage en utilisant le champ des cellules prévu à cet effet.

V.3 Condition d'arrêt et coût

V.3.a Précision des estimateurs

Le problème est de connaître le moment où la précision des estimateurs est suffisante pour arrêter la simulation. Avec les outils de la machine Sim-express, la simulation peut être arrêtée avec un test sur la valeur d'un registre. Le registre du nombre de pertes a été choisi pour servir de test d'arrêt de la simulation. Le problème est de choisir la meilleure valeur possible de ce registre comme condition d'arrêt.

Notons X_i tel que :

$$X_i = \begin{cases} 1 & \text{si la i}^{ième} \text{ cellule est perdue} \\ 0 & \text{sinon} \end{cases}$$

L'indice de performance étudié est EX_i (le taux de perte), un estimateur de cet indice est :

$$\overline{X}_n = \frac{1}{n}\sum_{i=0}^{n} \mathbb{I}_{\{X_i=1\}}$$

Supposons pour l'étude du problème que, les variables $\{X_i\}^{i\in\mathbb{N}}$ sont des variables aléatoires indépendantes et identiquement distribuées.

En utilisant le théorème central limite [Ros91], et l'estimateur standard non biaisé

$$\hat{\sigma}_n^2 = \frac{1}{n-1}\sum_{i=1}^{n} \left(X_i - \overline{X}_n\right)^2$$

l'intervalle de confiance $z(\alpha)$ est calculé avec le niveau de certitude α.

Typiquement avec $\alpha = 0.05$, $n = 10^{10}$ et 10.000 pertes, l'intervalle de confiance est 2%, alors qu'il est de 6.3% avec 1.000 pertes.

En réalité, avec un trafic en rafales, les variables X_i ne sont plus vraiment indépendantes. Quand la file est pleine, la probabilité d'avoir une perte ($X_i = 1$) est plus grande, mais les *rafales* de pertes peuvent toujours être considérées comme indépendantes les unes des autres [GRSS94]. En pratique, pour produire des résultats ayant une précision satisfaisante, chaque expérience est effectuée plusieurs fois et chacune des simulations est arrêtée quand on a observé 1.000 pertes de cellules. Par la suite, chaque point présenté sur les graphiques (de ce chapitre et des suivants) est obtenu en effectuant la moyenne de ces expériences. Le résultat sera donné avec un l'intervalle de confiance de 6.3%.

V.3.b Coût

L'objectif de ce paragraphe est d'étudier le coût en temps et en utilisation des ressources matérielles des différentes possibilités d'implantation du modèle.

Coût en temps

Soit n le nombre de cellules à traiter, q le nombre d'opérations qui doivent être effectuées sur chacune des cellules (dans notre cas il s'agit clairement du nombre de services que requiert

chaque cellule, c'est-à-dire le nombre de files où doit passer chaque cellule) et s le nombre de sources.

La complexité du problème est donc $\mathcal{O}(nq)$, en effet q opérations sont à effectuer sur chacune des n cellules. Il s'agit aussi de la complexité d'un algorithme séquentiel. Dans le modèle matériel, les q opérations sont pipelinées, et les s sources émettent en parallèle avec un débit ρ. Le nombre de tops horloge nécessaires pour traiter n cellules est donc :

$$\mathcal{O}\left(\frac{n}{s\rho}\right)$$

Par exemple, pour $n = 10^{10}$ cellules sur un commutateur 4x4 ($s = 4$), et $\rho = 0.8$, le nombre de cycles d'horloge nécessaires est $3,1\ .\ 10^9$. Si la fréquence de l'horloge est de $10^6 Hz$, comme c'est le cas pour Sim-express, 50 minutes sont nécessaires pour traiter ces cellules. Il est important de noter que le temps de traitement de ces 10^{10} cellules est moins important si l'on étudie un commutateur 8x8. En effet, on a alors $s = 8$ et un parallélisme plus important.

Coût en espace

La complexité en utilisation des ressources matérielles est plus délicate à étudier. D'abord elle va dépendre du modèle de file adopté et, bien sûr, de la taille du commutateur que l'on simule (4x4, 8x8 ...). Comme le montre le paragraphe (III.3), une file peut être implantée de différentes manières. Chacune de ces solutions a un coût différent au niveau de l'utilisation des ressources matérielles. De plus, l'instrumentation ne représente pas un coût négligeable, le coût en ressources matérielles est donc difficile à définir de manière précise. Par exemple, 4 commutateurs 4x4 en parallèle, formés de deux étages de commutateurs 2x2, utilisent 7 cartes de la machine Sim-express, i.e. 30% de la capacité totale. Ceci en tenant compte de l'instrumentation permettant de calculer les taux de perte sur les deux étages des commutateurs.

Un point important à noter est que l'ajout d'autres files d'attente ne change pas la complexité en temps du système. L'ajout d'étages dans le système étudié ne va pas entraîner une simulation plus lente.

V.3.c Conclusion

Plusieurs versions du simulateur de commutateur multi-étages ont été implantées.

Une version simple à l'aide du modèle sans contenu a été utilisée pour l'étude des pertes et l'étude des caractéristiques du trafic. Les composants développés à cette occasion sont entièrement paramétrables et réutilisables. Ainsi, deux commutateurs multi-étages ont été simulés : un commutateur 4x4 à deux étages et un commutateur 8x8 à trois étages.

Une version, utilisant le modèle de file avec contenu, a été utilisé pour l'étude des perturbations introduites par un trafic de fond sur une connexion particulière. Là encore les composants développés sont paramétrables et réutilisables.

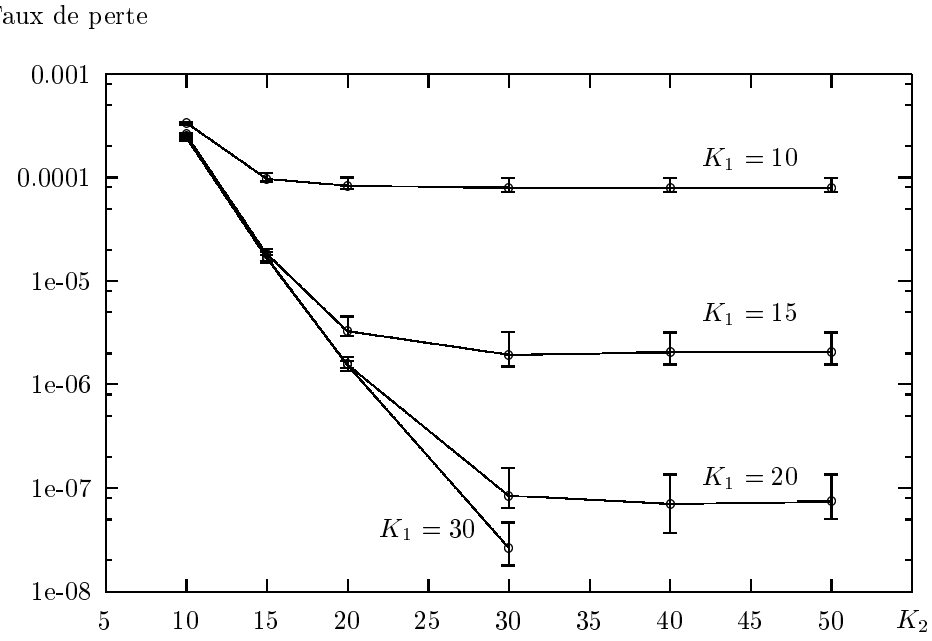

FIG. V.4: Taux de perte sur l'ensemble d'un commutateur 4x4 en fonction de la taille des tampons du deuxième étage (K_2). Chaque courbe correspond à une taille de tampon différente pour le premier étage (K_1). Trafic uniforme et charge d'entrée de $0,8$. Échelle logarithmique en ordonnée.

V.4 Résultats

Cette section est consacrée à l'étude d'un commutateur 4x4 (resp. 8x8) formé de deux (resp. trois) étages d'éléments de base 2x2 (voir figure V.2). Le premier paragraphe concerne les pertes sur l'ensemble du commutateur puis sur les différents étages. Cela conduit à l'étude des perturbations introduites par le commutateur sur le trafic. Dans le deuxième paragraphe de cette section, les propriétés du trafic de sortie sont comparées à celles du trafic d'entrée. Le modèle de trafic utilisé est le trafic uniforme. Chacune des sources émet des cellules selon un processus géométrique (voir III.3). Ces cellules choisissent leurs destinations de manière aléatoire, chacune des sorties ayant la même probabilité d'être choisie.

Dans ce qui suit, K_i est la taille des tampons (ie. la capacité des files d'attente) du $i^{ième}$ étage d'un commutateur 4x4 ou 8x8 ($i = 1, 2, 3$).

V.4.a Pertes

Sur l'ensemble d'un commutateur 4x4

La figure (V.4) montre le taux de perte global, c'est-à-dire le taux de perte sur l'ensemble du commutateur. La taille des files d'attente du deuxième étage (K_2) est représentée sur l'axe des abscisses. Cette taille varie de 10 à 50. Chaque courbe correspond à une taille différente pour les tampons des files d'attente du premier étage ($K_1 = 10, 15, 20, 30$). Les

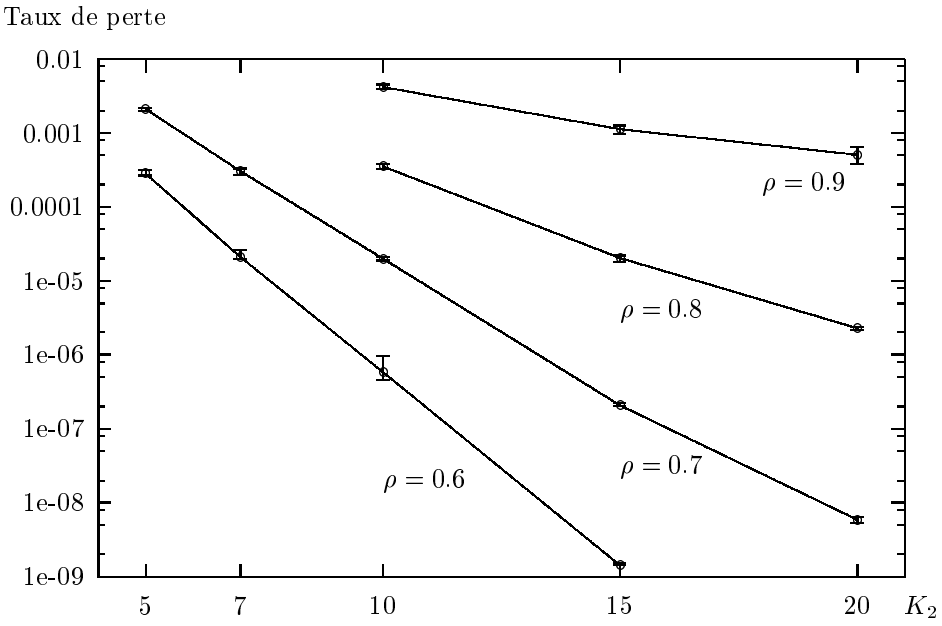

FIG. V.5: Taux de perte sur l'ensemble d'un commutateur 4x4 en fonction de la taille des tampons du deuxième étage (K_2). La taille des tampons du premier étage est fixée à 15. Trafic uniforme. Charge ρ variant de $0,9$ à $0,6$. Échelle logarithmique en ordonnée.

expériences ont été faites à charge constante : $\rho = 0,8$. Ce paramètre ρ est aussi le paramètre des processus géométriques des sources.

Un plateau peut être observé pour chaque courbe, pour les grandes valeurs de K_2. Ceci s'explique par le fait que, quand les tampons du deuxième étage sont assez grands, toutes les pertes observées surviennent au premier étage. Dans ce cas précis augmenter la taille des tampons du deuxième étage est parfaitement inutile.

D'un autre côté, pour les petites valeurs de K_2, les courbes sont confondues. Dans ce cas la capacité K_1 des tampons du premier étage est assez grande au regard des capacités du deuxième étage. Ainsi, toutes les pertes observées ont lieu au deuxième étage. Dans ce cas, il serait complètement inutile d'augmenter la taille des tampons du premier étage sans augmenter la capacité de ceux du deuxième.

Ainsi, pour $K_1 = 20$, la configuration "optimale", en termes de pertes au regard du coût en mémoire, est obtenue pour $K_2 \simeq 30$. Ceci signifie également que si $K_1 = K_2$ le taux de perte est différent pour chaque étage.

La figure (V.5) permet d'étudier le taux de perte pour l'ensemble du commutateur avec une taille K_1 fixée et une taille variable de K_2. Chaque courbe correspond à une charge ρ différente. La génération de ce graphique a nécessité 150 expériences (15x10), la plus longue d'entre elles ($K_1 = K_2 = 15$ et $\rho = 0.6$) a duré $10^9.100.\frac{1}{4.10^6} = 6h$. Quand un grand nombre d'expériences doivent être réalisées, le langage de pilotage de la machine Sim-express permet de réaliser des scripts de commandes. Ainsi, un seul script a permis de piloter ces 150 expériences et de générer cette figure.

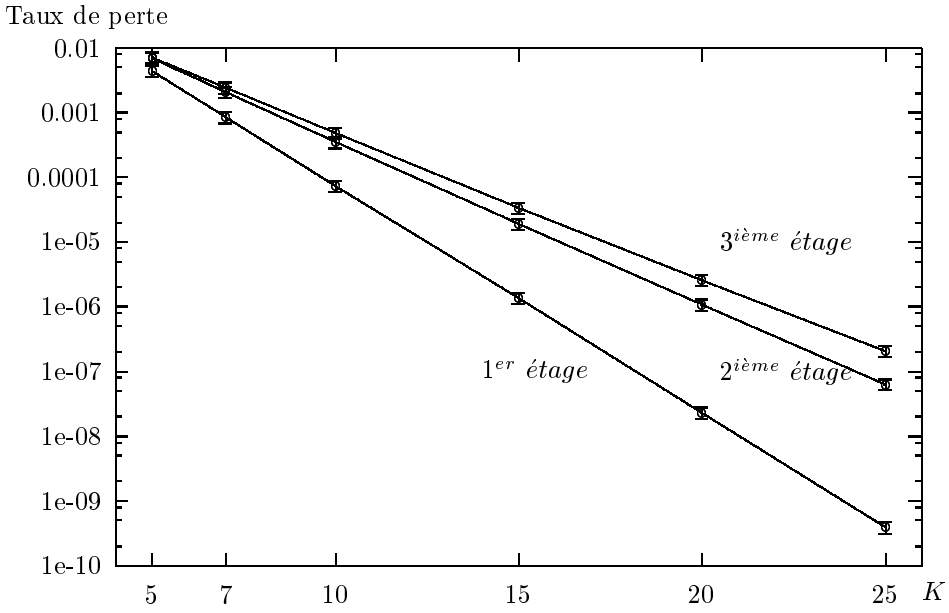

FIG. V.6: Taux de perte sur les différents étages d'un commutateur 8x8, en fonction de la taille $K = K_1 = K_2 = K_3$, des files d'attente. Trafic uniforme, charge de $0,8$. Échelle logarithmique en ordonnée.

Pertes par étage sur un commutateur 8x8

La probabilité de perte par étage peut être vue sur la figure (V.6), où les tampons des différents étages sont de capacité égale ($K = K_1 = K_2 = K_3$). Il est intéressant de remarquer que les pertes sont toujours plus importantes pour les étages supérieurs (les plus éloignés des sources).

Ceci peut être expliqué par le fait que le premier étage introduit des perturbations sur le trafic. En d'autres termes, le trafic sortant d'un étage de tampon semble être plus perturbé, il semble comporter plus de rafales que le trafic qui a été injecté dans cet étage.

C'est pour cette raison que l'étude des caractéristiques du trafic en sortie doit être réalisée. Ceci va pouvoir être fait grâce à l'instrumentation permettant d'obtenir des histogrammes. L'information récoltée par l'analyseur de trafic va être présentée au paragraphe suivant.

V.4.b Étude d'une connexion

Dans ce paragraphe, on étudie l'impact d'un trafic de fond sur une communication point à point. Pour obtenir les résultats sur les taux de perte, le modèle de file sans contenu a été utilisé (III.3). Pour cette étude les paquets du modèle transportent réellement une information qui permet de connaître l'origine de chacune des cellules ainsi que leur destination. Ainsi on utilise des cellules marquées pour différencier le trafic de fond de la connexion point à point.

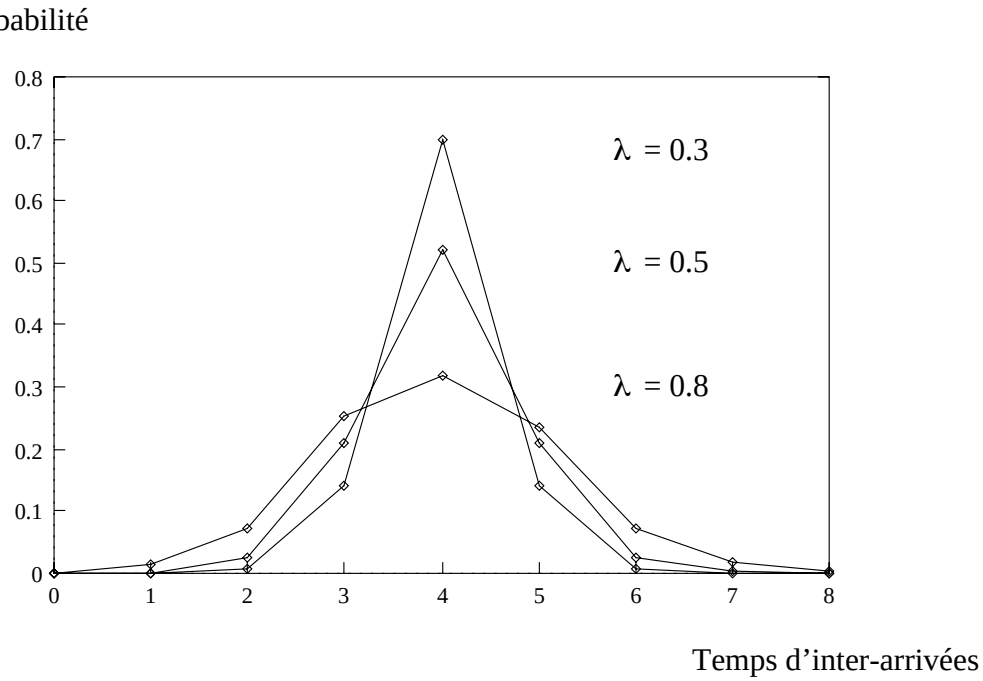

FIG. V.7: Distribution des temps d'inter-arrivées, en sortie d'un commutateur 4x4, des cellules en provenance d'une source de période 4. λ est la charge du trafic de fond, c'est aussi le paramètre des processus géométriques des sources aléatoires.

Une des sources est consacrée à l'émission d'un trafic périodique, les autres sont utilisées pour générer un trafic uniforme. Ainsi, une des sources est remplacée par une source périodique de période 4. Tout les quatre slots une cellule est émise par cette source.

La figure (V.7)[1] montre les perturbations introduites par le trafic aléatoire sur le trafic périodique. La courbe indique la distribution des temps d'inter-arrivées à la sortie du commutateur pour la connexion point à point périodique. Comme on peut s'y attendre ces perturbations augmentent avec la charge λ du trafic de fond. Plus le trafic de fond est important, plus augmente la dispersion (la gigue) des cellules de la connexion périodique.

Les techniques de contrôle-espacement et les politiques de service plus complexes (I) ont pour objectif de limiter ces perturbations [BGSC92].

[1] Cette figure et les suivantes représentent des histogrammes. Cependant, pour obtenir une meilleure lisibilité des différentes distributions, nous n'avons pas utilisé les bâtons usuels.

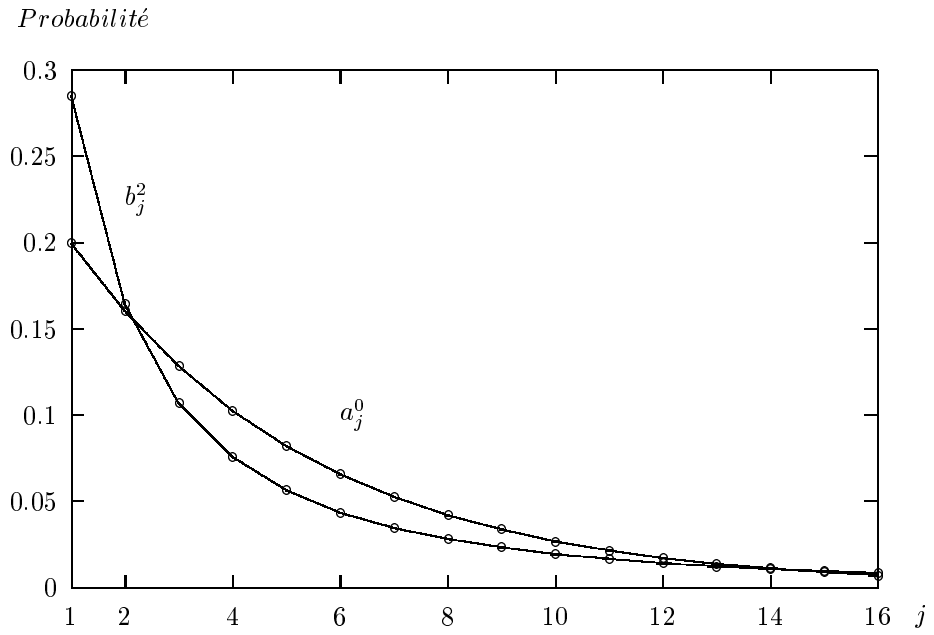

FIG. V.8: Distribution empirique de la probabilité d'apparition d'une rafale de taille j à la sortie des sources ($i = 0$) et à la sortie du deuxième étage ($i = 2$).

V.4.c Structure du flux de sortie d'un commutateur 4x4

Dans ce paragraphe, on étudie les perturbations introduites sur un trafic uniforme par le passage dans un commutateur . Toutes les sources sont des sources aléatoires de processus géométrique de paramètre ρ.

Les perturbations du trafic sont mises en évidence par l'instrumentation utilisée comme un analyseur de trafic. Cette analyseur de trafic donne le nombre de fois qu'une inter-arrivée a duré j slots à la sortie du $i^{ième}$ étage. Cela permet d'estimer la probabilité a_j^i d'observer un temps d'inter-arrivée de j à la sortie du $i^{ième}$ étage.

L'instrumentation donne aussi le nombre de rafales (cellules collées) qui ont duré j slots à la sortie du $i^{ième}$ étage. Ceci permet l'estimation de la probabilité b_j^i d'observer une rafale qui dure j slots à la sortie du $i^{ième}$ étage. Par convention, on prend $i = 0$ qui donne ces probabilités à la sortie des sources.

La figure (V.10) présente les distributions empiriques $\hat{a}_j^0$ et $\hat{a}_j^2$ et la figure (V.9) expose les mêmes distributions en utilisant une échelle logarithmique. La figure (V.8) montre les distributions empiriques $\hat{b}_j^0$ et $\hat{b}_j^2$ ($0 < j \leq 16$) et la figure (V.11) présente les mêmes distributions en utilisant une échelle logarithmique.

Comme le processus d'émission des sources est un processus géométrique de paramètre ρ, la distribution théorique a_j^0 de $\hat{a}_j^0$ peut être calculée :

$$a_j^0 = (1 - \rho) * \rho^{(j-1)}$$

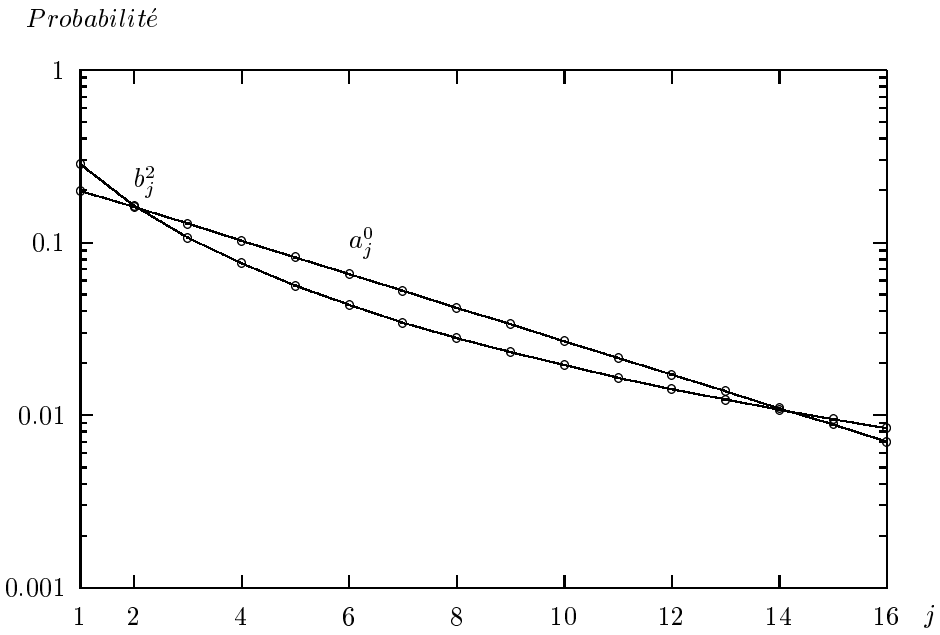

FIG. V.9: Distribution empirique de la probabilité d'apparition d'une rafale de taille j à la sortie des sources ($i = 0$) et à la sortie du deuxième étage ($i = 2$). Échelle logarithmique en ordonnée.

Un test[2] du χ^2 sur la distribution observée $\hat{a}_j^0$ permet de vérifier la justesse du générateur pseudo-aléatoire utilisé.

Pour $\hat{b}_j^0$ la distribution théorique b_j^0 est donnée par :

$$b_j^0 = \rho * (1-\rho)^{(j-1)}$$

Un test du χ^2 sur la distribution observée confirme que le trafic observé à la sortie des sources est bien conforme à celui que l'on attend.

Les distributions $\hat{a}_j^2$ et $\hat{b}_j^2$ permettent de se faire une idée des perturbations que le commutateur introduit dans le trafic. Sur la figure (V.10), il est clair que le temps moyen d'inter-arrivées a augmenté. La figure (V.8) montre que le nombre de petites rafales croît, tout comme le nombre des grandes rafales $\hat{b}_{j>16}^2 = 0.918$ et $\hat{b}_{j>16}^0 = 0.0282$ (voir tableau V.4.b).

Ainsi la différence de pertes entre les différents étages s'explique par le fait que la longueur moyenne des rafales a augmenté par le passage dans les étages du commutateur. Le trafic étant plus perturbé, les tampons saturent plus rapidement, ce qui provoque des pertes plus importantes.

V.5 Conclusion

Ce chapitre a montré que l'utilisation de ressources matérielles permet de mesurer des taux de perte qui sont réalistes, i.e. de l'ordre de 10^{-9} (voir figure V.5).

[2] Tous les tests statistiques sont effectués avec un intervalle de confiance de 95%

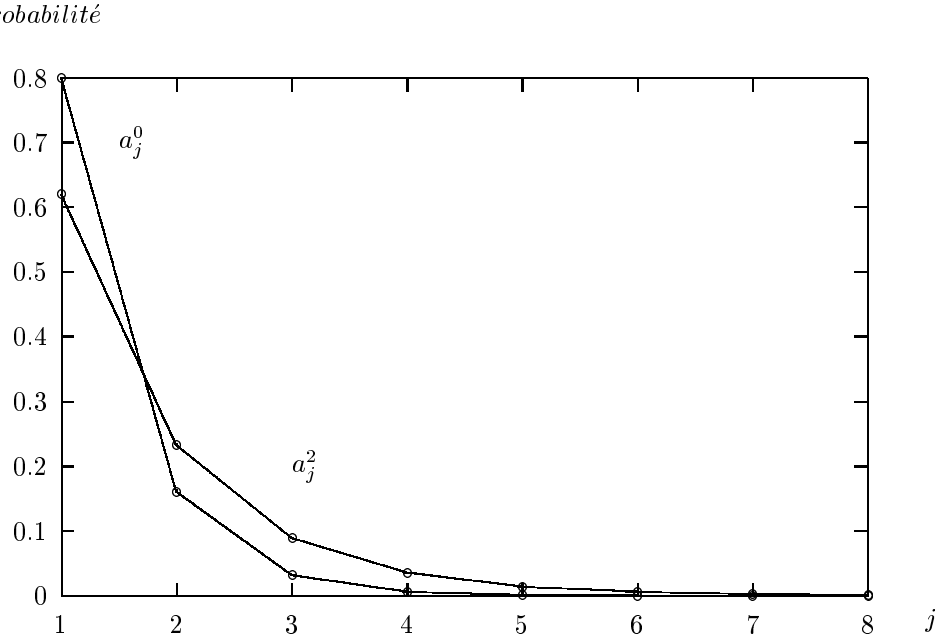

FIG. V.10: Distribution empirique de la probabilité d'apparition d'un silence de taille j à la sortie des sources ($i = 0$) et à la sortie du commutateur ($i = 2$).

Les paragraphes (V.4.b) et (V.4.c) montrent aussi que cette technique permet d'obtenir d'autres informations intéressantes. Ainsi, des informations concernant la gigue, et sur les caractéristiques du trafic ont pu être étudiées, et ceci sans augmenter la complexité temporelle du modèle.

De plus, la durée des simulations ne dépend pas du nombre de files d'attente étudiées mais uniquement du nombre de cellules à traiter (voir paragraphe V.3.b).

Cependant cette modélisation des commutateurs reste sommaire et ne permet pas de rendre compte d'autre phénomènes difficiles à étudier dans les commutateurs réels. En particulier, la complexité croissante des politiques de service et leur impact sur les caractéristiques du trafic restent des problèmes difficiles à étudier.

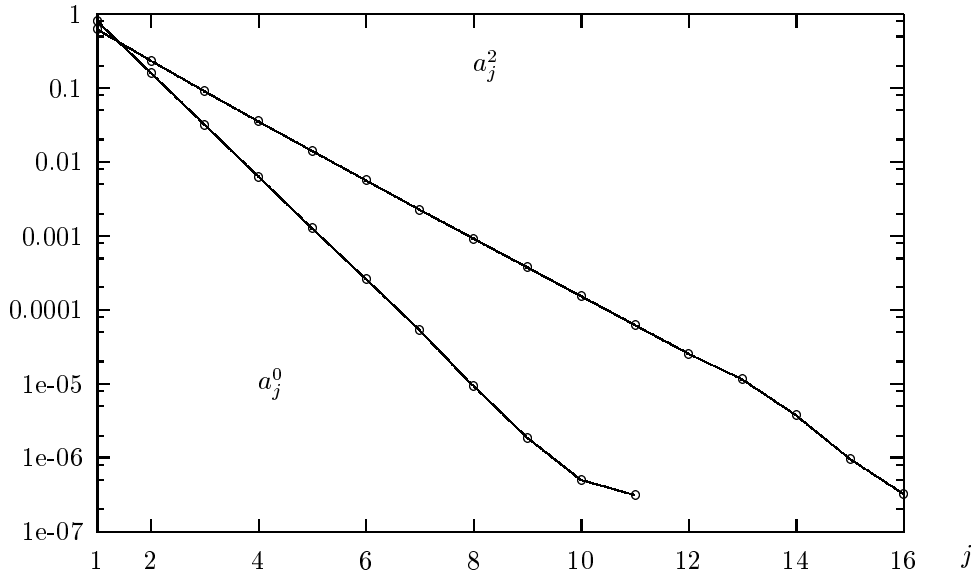

FIG. V.11: Distribution empirique de la probabilité d'apparition d'un silence de taille j à la sortie des sources ($i = 0$) et à la sortie du commutateur ($i = 2$). Échelle logarithmique en ordonnée.

j	b_j^0	b_j^2	a_j^0	a_j^2
1	0.1994	0.2851	0.7998	0.6196
2	0.1602	0.1641	0.1602	0.2323
3	0.1283	0.1069	0.0319	0.0894
4	0.1024	0.0756	0.0063	0.0351
5	0.0819	0.0562	0.0012	0.0139
6	0.0655	0.0433	0.0002	0.0056
7	0.0527	0.0344	$5.3E-5$	0.0022
8	0.0417	0.0280	$9.4E-6$	0.0009
9	0.0335	0.0232	$1.8E-6$	0.0003
10	0.0267	0.0194	$5.0E-7$	0.0001
11	0.0214	0.0165	$3.1E-7$	$6.1E-5$
12	0.0171	0.0141	0	$2.5E-5$
13	0.0137	0.0122	0	$1.1E-5$
14	0.0109	0.0107	0	$3.7E-6$
15	0.0087	0.0094	0	$9.7E-7$
16	0.0069	0.0083	0	$3.2E-7$
> 16	0.0282	0.0918	0	$6.4E-7$

TAB. V.1: Distribution a_j^i et b_j^i. ($K_1 = K_2 = 10$ et $\rho = 0.8$).

Bibliographie

[Abu98] O. Abuamsha. *Application des méthodes de la comparaison stochastique pour l'analyse des disciplines "Fair-Queueing"*. PhD thesis, PRiSM, Université de Versailles, 1998.

[Bey93] André-Luc Beylot. *Modèles de Trafics et de commutateurs pout l'Evaluation de la Perte et du Délai dans les Réseaux ATM*. PhD thesis, Université Paris VI, 1993.

[BGSC92] Pierre E. Boyer, Fabrice M. Guillemin, Michel J. Servel, and Jean-Pierre Coudreuse. Spacing cells protects and enhances utilization of atm network's links. *IEEE Networks*, pages 38–48, 1992.

[BKYB95] André-Luc Beylot, Josephina Kohlenberg, Haikel Yaiche, and Monique Becker. Performance analysis of an atm switch based on a three stage clos interconnection network under non-uniform ibp traffic patterns. In *Proceedings of the First Workshop on ATM Traffic Management WATM'95*, page 263, Paris France, Déc 1995. IFIP.

[BM93] André-Luc Beylot and Becker M. Performance analysis of an atm switch based on a three-stage clos interconnection network. Technical Report MASI 92.45, INT-MASSI, 1993.

[FMH93] W. Fischer and K. Meier-Hellstern. The markov-modulated poisson process (MMPP) cookbook. *Performance Evaluation North-Holland*, 18, 2 :149–171, 1993.

[FMP97] J.-M. Fourneau, L. Mokdad, and N. Pekergin. Bounding the loss rates in a multistage ATM switch. *Lecture Notes in Computer Science*, 1245 :193–205, 1997.

[FPT95] J.-M. Fourneau, N. Pekergin, and H. Taleb. An application of stochastic ordering to the analysis of buffers with SBBP arrivals. In *Proceedings of the First Workshop on ATM Traffic Management WATM'95*, page 263, Paris France, Déc 1995. IFIP.

[GRI81] GRINSEC. *La commutation électronique*. EYROLLES, 1981.

[GRSS94] F. Guillemin, G. Rubino, B. Sericola, and A. Simonian. Transient characteristics of an $M/M/\infty$ system applied to statistical multiplexing on an ATM link. Publication interne 874, IRISA, october 1994.

[HYP98] S. Hai, K. L. Yeung, and L. Ping. Optimal queuing policy for input-buffered ATM switches with two-fifo queues per input port. In *Proceedings of the International Conference on Telecommunications*, Porto Carras Greece, June 1998. IFIP.

[LRV98] C. Labbé, F. Reblewski, and J-M. Vincent. Performance evaluation of high speed network protocols by emulation on a versatile architecture. *RAIRO, Systèmes à événements discrets stochastiques : théorie, application et outils.*, 1998.

[RG92] J. Roberts and F. Guillemin. Jitter in ATM networks and its impact on peak rate enforcement. *Performance Evaluation North-Holland*, 16, 1-3 :35–48, 1992.

[Ros91] Sheldon M. Ross. *A Course in Simulation.* Mamillan Publishing Company, University of california, Berkeley, 1991.

[Tru95] Laurent Truffet. *Méthodes de Calcul de Bornes Stochastiques sur des Modèles de Systèmes et de Réseaux.* PhD thesis, Université Paris VI, 1995.

[YHP98] K. L. Yeung, S. Hai, and L. Ping. Thoughput analysis for input-beffered ATM switches with multiple fifo queues per input port. In *Proceedings of the International Conference on Telecommunications*, Porto Carras Greece, June 1998. IFIP.

Chapitre VI

Fair Queuing (service équitable)

L'objectif de ce chapitre est d'étudier les possibilités de simulation des politiques de Fair Queuing (FQ) à l'aide de ressources matérielles.

Le première section (VI.1) présente de manière assez succincte le principe des différentes politiques de FQ utilisées dans les réseaux à haut débit. Ces techniques sont relativement récentes et font donc l'objet de nombreuses recherches, on pourra notamment consulter les ouvrages [Abu98, GM92, Sti96, BZ97, GVC97]. Ces politiques FQ sont destinées à être implantées dans des commutateurs à l'aide de ressources matérielles. La modélisation d'un commutateur utilisant une politique FQ à l'aide de ces mêmes ressources peut donc être particulièrement intéressante.

L'une de ces disciplines FQ a donc été implantée sur la machine. Les différents choix fait pour cette implantation sont présentés dans le paragraphe (VI.2). Quelques résultats d'expériences réalisées à l'aide de cette implantation sont présentés dans la section (VI.3).

VI.1 Les politiques de Fair Queuing

Les politiques FQ ont pour objectif d'assurer l'équité du partage d'une ressource entre différents flux de clients (voir I.3). Ce paragraphe présente la philosophie générale et montre la grande diversité de ces politiques. Dans un premier temps, la politique de référence "Generalized Processor Sharing" (GPS) est présentée. Il s'agit d'un modèle théorique qui sert de référence aux politiques FQ qui tentent de l'approcher. Ainsi, la politique "Packet-by-packet Generalized Processor Sharing" (PGPS) est directement dérivée de la politique GPS. PGPS est présentée dans le paragraphe (VI.1.b). Enfin, un certain nombre de simplifications sont introduites pour faciliter l'implantation de ces politiques à l'aide de ressources matérielles. Ces simplifications sont passées en revue dans les paragraphes (VI.1.c) et (VI.1.d).

VI.1.a La discipline fluide "GPS" (Generalized Processor Sharing)

La discipline GPS est une discipline qui assure l'équité du partage d'une ressource fluide entre différents flux (ou sessions). Ces flux ont des exigences variées qui se traduisent par la réservation d'une certaine quantité de ressource. Même si ces flux sont composés de paquets, on considère ceux-ci comme indéfiniment divisibles.

Pour appliquer ce modèle aux réseaux, on assimile la bande passante à la ressource et les connexions (ou groupe de connexions) aux sessions.

Considérons N sessions se partageant les ressources d'un serveur GPS de taux [1] r. Ce serveur n'est jamais au repos s'il y a du travail en attente dans le système. Une session est dite "active" à l'instant t si son tampon n'est pas vide. A chaque session i est associé un nombre réel positif ϕ_i. Ce nombre ϕ_i doit être vu comme le taux de réservation de service pour la session i. La stabilité du système requiert donc que $\sum_i \phi_i \leq r$ (seul ce cas sera considéré).

La discipline GPS est dite "équitable" car elle est construite de telle manière que, pour toute session i active entre les instants t_1 et t_2 on ait :

$$\frac{W_i(t_1,t_2)}{W_j(t_1,t_2)} \geq \frac{\phi_i}{\phi_j},\ j = 1,2,..,N,$$

où $W_i(t_1,t_2)$ est la quantité de service fournie à la session i entre les instants t_1 et t_2.

Le service normalisé offert à la session i pendant l'intervalle de temps $[t_1,t_2]$ est donc donné par : $\frac{W_i(t_1,t_2)}{\phi_i}$. Ainsi, avec cette définition, GPS est une discipline qui offre à toutes les sessions actives la même quantité de service normalisée. On en déduit que :

$$W_i(t_1,t_2) \geq \frac{\phi_i}{\sum_j \phi_j}(t_2 - t_1)r$$

et que le taux de service garanti à la session i est :

$$r_i = \frac{\phi_i}{\sum_j \phi_j} r$$

Dans les cas où l'on considère des flux composés de paquets indivisibles, la politique GPS devient un modèle théorique. Il s'agit même d'un modèle idéal, en effet, le délai maximal de service pour une session ne dépend que de la taille de la file d'attente qui lui est associée. Ce délai est indépendant de l'état des files associées aux autres sessions. C'est une particularité très intéressante dans le cas des réseaux puisque une session n'est pas pénalisée par le trafic des autres.

De plus, la discipline GPS accomplit un partage équitable parfait entre les sessions. Si l'une ou plusieurs des sessions sont inactives, la discipline GPS distribue la part de la bande passante qui leur est réservée entre les sessions actives. Cette bande passante est partagée entre ces sessions proportionnellement aux réservations (ϕ_i) associées à chacune d'elles.

Ce modèle théorique qui suppose la fluidité des demandes (ce qui n'est pas le cas dans les réseaux à commutation de paquets) possède donc de nombreux avantages. Les nombreuses politiques dites "Fair Queuing" tentent d'imiter au mieux ce modèle idéal.

VI.1.b La discipline "PGPS" (Packet-by-packet Generalised Processor Sharing)

Cette politique de services est aussi appelée "Weighted Fair Queuing" (WFQ). Son principe est d'imaginer un serveur GPS qui évolue dans un espace de temps virtuel dans les mêmes conditions (même paramètres et mêmes paquets) que le système réel.

[1] Les notations utilisées ici sont celles de [Abu98].

Pour coupler le système réel et GPS, il faut représenter l'avancement du service dans le système virtuel en fonction du temps réel. Ceci est fait avec l'introduction d'un temps virtuel qui évolue comme le service normalisé. Ainsi, entre deux instants "virtuels" $v(t_0)$ et $v(t)$, la quantité de service normalisé fourni à une session active est :

$$\frac{W_i(t_0,t)}{\phi_i} = v(t) - v(t-0), \forall i \in B(t_0,t),$$

où $B(t_0,t)$ est l'ensemble des sessions actives pendant l'intervalle $[t_0,t]$.

Ceci permet de montrer que le temps virtuel $v^{PGPS}(t)$ de la discipline PGPS évolue linéairement selon l'expression :

$$v^{PGPS}(t) = v(t_0) + \frac{r(t-t_0)}{\sum_{i \in B(t_0,t)} \phi_i}. \tag{VI.1}$$

On remarque que chaque changement dans l'ensemble B (activation où désactivation d'une session) entraîne un changement de la "pente" du temps virtuel. Celui-ci est donc linéaire par morceau.

Pour chaque paquet j de la session i ayant une taille l_i^j et une date d'arrivée dans le système réel a_i^j, les instants virtuels de début S_i^j et de fin F_i^j de service sont donnés par :

$$S_i^j = \max\{F_i^{j-1}, v(a_i^j)\},$$

$$F_i^j = S_i^j + \frac{l_i^j.r}{\phi_i}.$$

La stratégie des politiques FQ pour simuler GPS est donc de calculer ces marques pour tous les paquets entrant dans le système. Puis ces marques servent à choisir le paquet qui doit être servi pour que le système réel s'éloigne le moins possible du modèle GPS.

Ainsi, PGPS marque les paquets qui intègrent le système, et il sert le paquet qui a la plus petite marque de fin virtuelle ($\min_{i \in B(t)}(F_i^j)$). Lorsque plusieurs paquets ont la même marque, un choix aléatoire ou un mécanisme de priorité peut être utilisé pour les départager.

En général, on considère que PGPS est le schéma qui simule le mieux GPS. De plus, pour un paquet p, le retard entre l'instant de fin de service réel ($F^{PGPS}(p)$) et l'instant de service virtuel ($F^{GPS}(p)$) est au maximum égal à la durée de service du plus long paquet :

$$F^{PGPS}(p) - F^{GPS}(p) \leq \frac{l_{max}}{r}$$

Ce résultat ne donne pas un écart maximal entre le comportement des deux systèmes. Il se contente de borner le retard des paquets. En effet, un serveur PGPS peut servir un paquet très en avance par rapport à la discipline GPS.

Cette situation peut être considérée comme gênante, en effet, elle peut entraîner une déformation importante du flux et laisser croire à une congestion du réseau ce qui pourrait influencer de manière négative les mécanismes de contrôle de congestion.

VI.1.c Changement de stratégie de choix

De nombreuses variantes de la politique PGPS ont été proposées. Oula Abuamsha présente une bonne synthèse de ces variantes dans [Abu98]. En effet, à la différence des disciplines du type "Round Robin" où l'ordre de service des sessions est fixé à l'avance, le serveur FQ doit choisir, après chaque fin de service, le paquet suivant. Ainsi, plusieurs stratégies de choix sont possibles :

- SFF (Smallest virtual Finishing time First) : cette stratégie est celle de la discipline PGPS, elle consiste à choisir le paquet ayant la marque de fin la plus petite.
- SSF (Smallest virtual Starting time First) : ici, le paquet ayant la marque de début minimale est choisie.
- SEFF (Smallest Eligible virtual Finishing time First) : cette stratégie consiste à définir un ensemble de paquets éligibles. Le paquet à servir est le paquet éligible qui possède la plus petite marque de fin. Cette technique est utilisée pour limiter la gigue du service. Par exemple, l'ensemble des paquets éligibles à l'instant t est l'ensemble des paquets qui ont virtuellement déjà commencé leur service : $E(t) = \{p_i^j, S_i^j \leq v(t)\}$.

Cette dernière stratégie permet d'éviter que les paquets prennent trop d'avance par rapport au modèle de référence PGP.

On peut remarquer qu'un grand nombre de possibilités existe parmi les politiques de type SEFF. En effet, on peut jouer sur la définition de l'ensemble éligible et sur le choix du calcul du temps virtuel. Ces possibilités de choix sont présentées au paragraphe suivant.

VI.1.d Simplification du calcul "temps virtuel"

Il ne faut pas perdre de vue que ces algorithmes sont destinés à être implantés dans des commutateurs et plus particulièrement dans des commutateurs pour des réseaux à haut débit. C'est pourquoi ces algorithmes doivent fonctionner très rapidement (au rythme du temps de service d'un paquet). Le problème de leur coût est donc un problème crucial.

La complexité de ces algorithmes a deux sources principales. La première est le coût de la sélection du paquet à servir. Cette sélection est classiquement faite par des algorithmes de tri avec un coût logarithmique. Certaines méthodes tentent de trouver des mécanismes de sélection à moindre coût (voir VII et [Sti96]).

La deuxième source de complexité est la mise à jour du temps virtuel. En effet, en utilisant la formule (VI.1) la mise à jour du temps virtuel est très coûteuse (de l'ordre de $\mathcal{O}(N)$ où N est le nombre de sessions qui se partagent le serveur soit plusieurs centaines dans le cas d'un commutateur). De plus l'ensemble des sessions actives peut varier très fortement, ce qui implique une remise à jour très fréquente de l'ensemble $B(t_0, t)$ et donc un surcroît de calcul lors de la mise à jour du temps virtuel. Ces mises à jour doivent être effectuées pendant le temps de commutation très court d'un paquet.

Pour résoudre ce problème de complexité des calculs, des disciplines avec un temps virtuel plus simple ont été proposées. Ces politiques gardent le même schéma de calcul des marques.

Voici quelque exemples de temps virtuel simplifié :

- $v(t) = t$. Dans ce cas on réduit le temps virtuel au temps réel.
- $v(t) =$ *la marque de fin du dernier paquet servi avant l'instant t.*
- $v(t) =$ *la marque du début du paquet en cours de service à l'instant t.*

Chacune de ces possibilités a des avantages et des inconvénients. On voit la diversité qu'offrent les politiques FQ de par le choix d'un mode de calcul, puis par le choix de la stratégie de sélection dans les paquets. Le lecteur peut trouver dans [Abu98] la description, et les comparaisons entre ces différentes politiques.

Un petit exemple devenu classique [BZ97, Abu98] permet de mieux comprendre ces différences. On considère 11 sessions qui partagent un serveur de capacité 1, avec $\phi_1 = \frac{1}{2}$ et $\phi_i = \frac{1}{22} \forall\ i \neq 1$. On suppose que tous les paquets émis sont de longueur 1. A l'instant 0, la session 1 émet 11 paquets tandis que chacune des autres sessions émet un seul paquet. Le calcul des marques donne les valeurs suivantes :

$S_1^1 = 0$ et $F_1^1 = 2$,

$S_1^i = 2(i-1)$ et $F_1^i = 2i$, pour les paquets $i = 2, ...11$,

$S_j^1 = 0$ et $F_j^1 = 2$, pour les sessions $j = 2, ...11$.

La figure (VI.1) montre les différences entre les politiques, celles qui n'utilisent pas l'éligibilité dispersent plus les cellules.

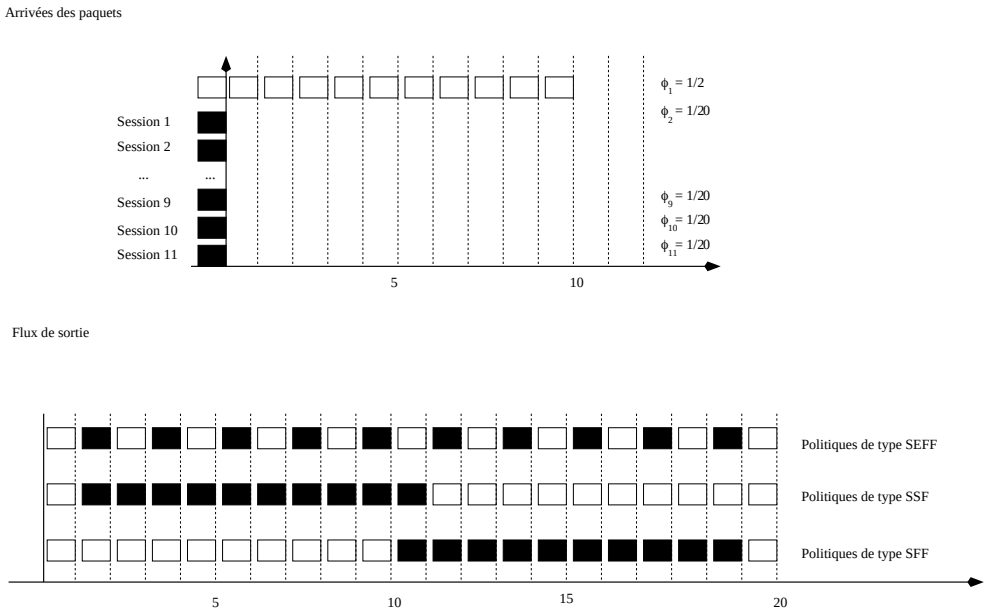

FIG. VI.1: Cet exemple (tiré de [BZ97, Abu98]) permet de visualiser certaines des différences existant entre trois types de politique FQ. Les disciplines avec éligibilité réduisent les dispersions.

VI.1.e Conclusion

Les politiques de service FQ sont des politiques de service complexes. Elles sont destinées a être implantées à l'aide de ressources matérielles dans les commutateurs des réseaux. Elles ont donc été adaptées en vue de cette utilisation

La simulation de commutateurs utilisant ces politiques à l'aide de ressources matérielles peut se révéler particulièrement intéressante. En effet, la complexité de ces politiques rend l'étude statistique de leur comportement assez délicate.

VI.2 Implantation

L'objectif est donc de décrire une discipline FQ assez simple à l'aide de ressources matérielles.

Une implantation assez générale pour être évolutive est présentée dans cette section. Cette implantation comporte un serveur, des sources et une instrumentation pour l'étude des perturbations introduites par le passage dans la file d'attente.

On a choisi d'implanter une discipline FQ avec un calcul simplifié du temps virtuel :

$$v(t) = \textit{ la marque de fin du dernier paquet servi avant l'instant } t.$$

Le choix du paquet à servir est fait avec la stratégie SFF. Pour simplifier l'implantation, un nombre de sessions de $N = 2^q$ sera considéré.

VI.2.a Le serveur

Le composant serveur recherche le minimum des marques virtuelles de fin de service parmi les paquets de tête des sessions actives. La file ayant le paquet de tête avec la marque la plus faible doit être servie. La valeur du temps virtuel doit être mise à jour avec la marque de fin de cette cellule.

Le serveur utilise donc un composant minimum identique à ceux présentés dans (III.2). Plusieurs tops horloge sont nécessaires pour effectuer le calcul du minimum des N nombres. La solution parallèle a été retenue de manière à minimiser le nombre de tops horloge nécessaires pour simuler un slot. Ceci permet de ne pas trop ralentir la simulation. Le temps de calcul du minimum est donc de q tops horloge.

Lorsque toutes les files sont vides, le temps virtuel est remis à 0. Ainsi, le registre de ce dernier, lors de très longues simulations, peut arriver à saturation et devenir nul. Cependant, cette technique ne permet pas de règler le problème, il est donc important de prévoir un registre suffisamment grand pour ne pas arriver à saturation ou à défaut de prévoir un signal d'alerte qui arrêtera la simulation.

En cas d'égalité des marques, le composant minimum a été modifié pour choisir alternativement l'une ou l'autre des files. Ce choix a un impact sur les performance du système lorsque la charge est élevée. Ceci est mis en évidence par les résultats présentés dans le paragraphe (VI.3).

Le composant qui simule le serveur est le composant "maître" de la simulation. C'est lui qui donne aux autres composants le signal de la fin du slot.

VI.2.b Les files

L'étude que l'on souhaite faire ici ne nécessite pas l'emploi du modèle de file avec contenu. En effet, les différentes sources sont connectées directement à leurs files respectives et l'étude du flux de sortie peut se faire sur le numéro de la file servie.

Le composant utilisé est légèrement modifié par rapport au modèle présenté au paragraphe (III.3). En effet, en plus de la mise à jour, à chaque slot, du nombre de clients en attente et du nombre de rejets, ce composant doit aussi réaliser la mise à jour de la date de fin virtuelle du client de tête. Le traitement de la simultanéité a été effectuée avec l'option DF.

VI.2.c Les sources

La plupart des études théoriques sur les politiques de service FQ utilisent des flux périodiques. Ceci permet de réaliser un certain nombre de calculs de pires cas. On a choisi de travailler avec des sources aléatoires de manière à pouvoir réaliser des études statistiques.

Les sources choisies sont donc des sources qui génèrent un trafic de type processus géométriques. On notera α le paramètre de ces processus.

VI.2.d L'instrumentation

L'instrumentation du circuit est chargée de mesurer l'occupation d'une des files d'attente et le débit de sortie du commutateur. Par conséquent, les instruments de mesure enregistrent à chaque période système, le nombre de clients dans la file étudiée. De plus, afin d'analyser le trafic de sortie, on stocke en mémoire la taille des intervalles de temps qui séparent deux clients quelconques et ceux séparant deux clients d'un même flux.

Ceci est réalisé grâce à l'utilisation d'un composant d'instrumentation de type "histogramme" (voir III.3).

VI.2.e Le composant "expérience" et son coût

Le composant "expérience"

Tous ces composants sont regroupés dans un seul qui permettra de réaliser un certain nombre d'expériences (voir figure VI.2). Certains des résultats sont présentés dans la section (VI.3).

Ce composant expérience comporte :

- les sources,
- les files d'attente,
- le serveur,
- l'instrumentation.

Les paramètres d'une simulation sont :

- les capacités des files,
- les paramètres des processus géométriques pour les sources,
- les paramètres ϕ_i associés à chacun des flux.

Le coût

Le coût en temps de cette implantation peut être mesuré par le nombre de tops horloge nécessaires pour simuler un slot. Avec les choix qui ont été effectués ici, le composant

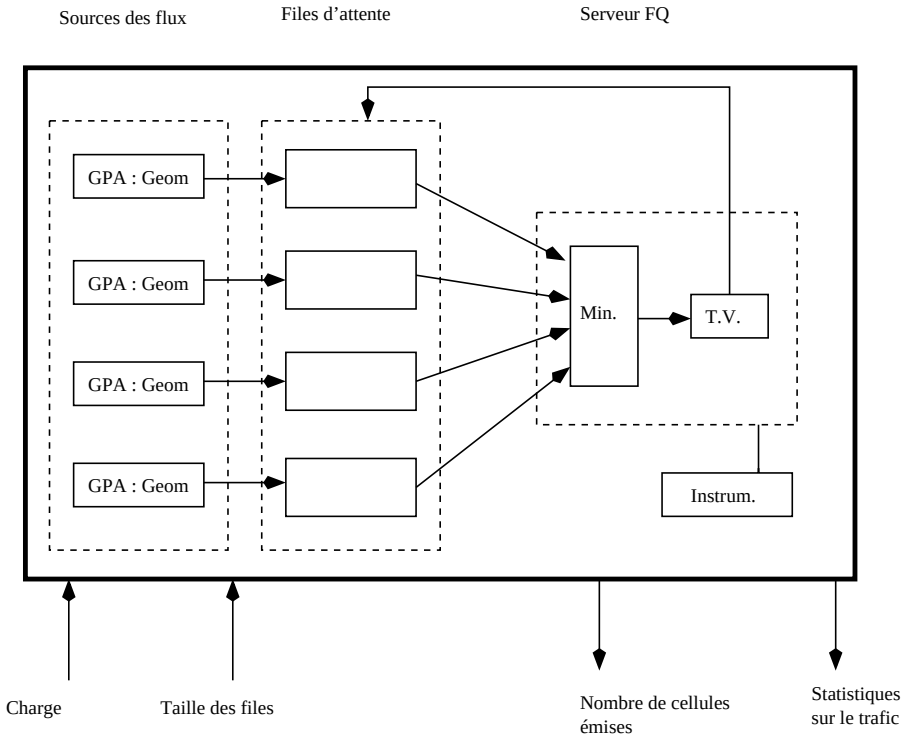

FIG. VI.2: Un composant réalisé à l'aide de ressources matérielles. Ce composant simule un serveur FQ et quatre flux qu'il tente de servir équitablement.

recherchant le minimum apporte la plus grande complexité. Si l'on considère un serveur répartissant une ressource entre N sources, le nombre de tops horloge nécessaires pour simuler un slot est $\log_2(N)$ soit q dans notre cas.

Le coût en ressources matérielles est composé de trois parties :

- le coût apporté par la recherche du minimum, soit $N-1$ comparateurs.
- le coût des N files d'attente associées aux N sources.
- l'instrumentation c'est-à-dire, dans notre cas, quelques mémoires.

VI.3 Résultats

Les résultats présentés ici mettent en évidence l'influence de la stratégie de choix en cas d'égalité des marques. L'étude des différences entre les politiques FQ peut justifier une plus longue campagne de comparaison, notamment avec des simulations comportant plus de sources. Mais la réalisation de la dernière application présentée dans le chapitre (VII) n'a pas permis de pousser plus avant cette étude.

VI.3.a Protocole expérimental

Pour étudier le système de manière statistique, il faut faire l'hypothèse de l'existence d'un régime stationnaire. C'est-à-dire que pour un temps assez grand, les lois des variables aléatoires étudiées sont indépendantes du temps.

Lors de la simulation, l'émulateur fonctionne pendant 30.000 tops horloge pour atteindre un état stationnaire (choix par la pratique). Le nombre de files étant fixé à quatre, ces 30.000 tops représentent 10.000 slots. Les composants d'instrumentation recueillent des informations pendant 30 millions de tops horloge soit 10 millions de slots.

Les conditions des expériences sont les suivantes :

- $\phi_i = \frac{1}{4}\ \forall i$
- les arrivées des clients dans chaque file suivent un processus géométrique de paramètre α. La charge du système ρ est donc $\frac{r}{4\alpha}$.
- par construction, le serveur a un taux de service $r = 1$.
- les capacités des files sont fixées arbitrairement grandes (300) pour éviter les pertes.
- La charge du système ρ prend les valeurs 0.5%, 0.6%, 0.7%, 0.8%, 0.9%, 0.95%.

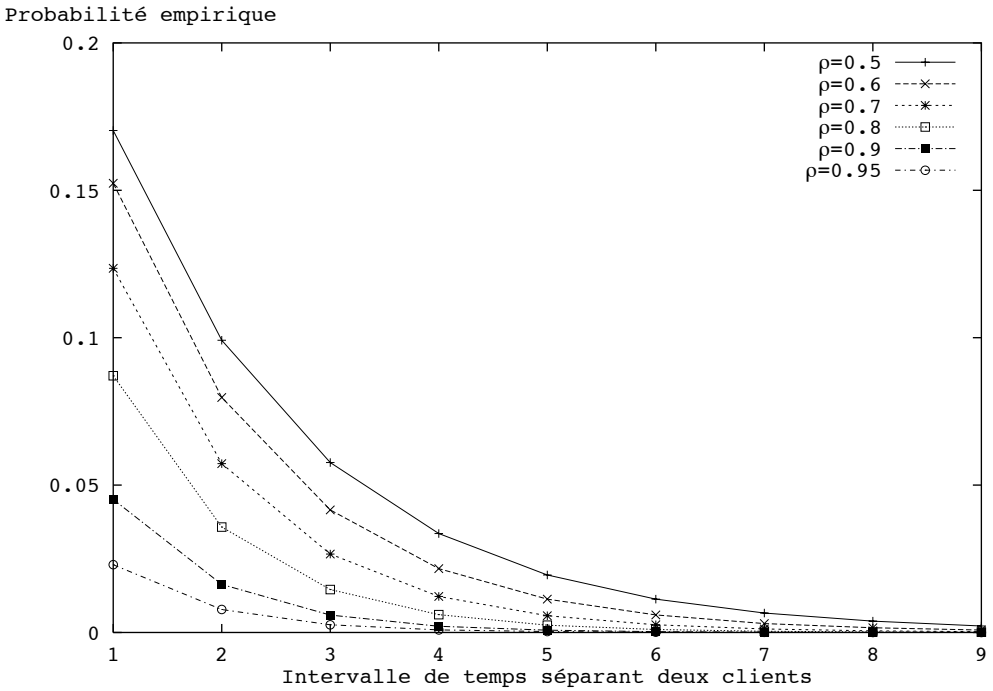

FIG. VI.3: Distribution empirique de la taille des intervalles de temps séparant les cellules du flux en sortie du serveur FQ. Chaque courbe correspond à une valeur différente de la charge ρ [3].

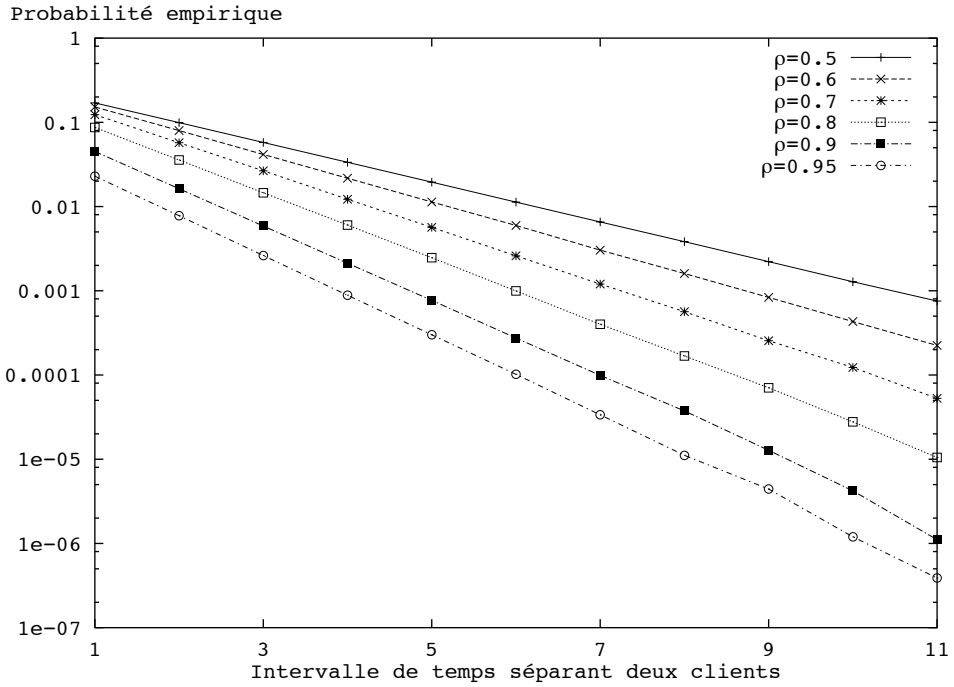

FIG. VI.4: Distribution empirique de la taille des intervalles de temps séparant les cellules du flux en sortie du serveur FQ. Chaque courbe correspond à une valeur différente de la charge ρ. Échelle logarithmique en ordonnée.

VI.3.b Étude du flux de sortie dans sa globalité

Une partie de l'instrumentation a été utilisée pour étudier le flux de sortie du serveur. Cette instrumentation a permis de tracer les histogrammes de la figure (VI.3). Comme dans le chapitre (V), on n'utilise pas les bâtons usuels pour mieux visualiser les différentes distributions.

La figure (VI.4) fait apparaître très clairement l'alignement des différents points. Grâce à un test du χ^2 on teste l'hypothèse.

VI.3.c Étude d'un flux particulier en sortie du serveur

Les figures (VI.5) et (VI.6) regroupent les distributions empiriques de l'intervalle de temps séparant deux clients consécutifs d'une même file. Ces figures mettent en évidence un phénomène d'oscillation pour les intervalles de temps inférieurs à 9. celui-ci s'amplifie à mesure que le trafic augmente. Tout se passe comme s'il y avait superposition d'un régime aléatoire ayant une distribution géométrique et d'un régime périodique.

En effet, en cas d'égalité de plusieurs dates de fin, un choix déterministe est effectué. Ce choix peut engendrer un phénomène périodique. Ainsi, les oscillations observées, proviennent

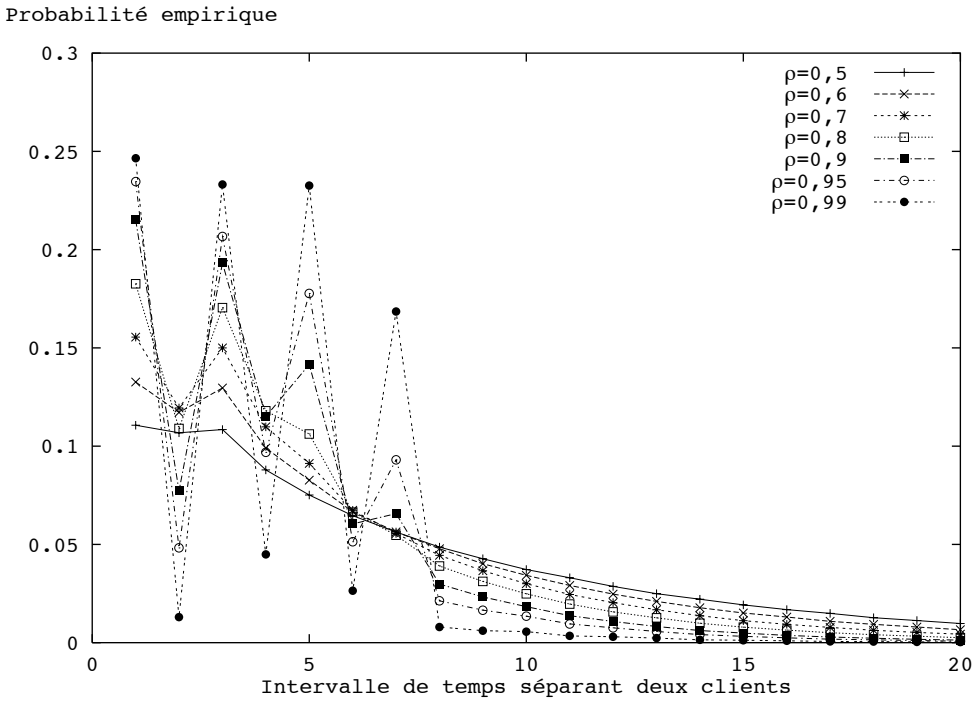

FIG. VI.5: Distribution empirique de la taille des intervalles de temps séparant les cellules d'un flux particulier en sortie du serveur FQ.

de la concurrence entre les échéanciers des différentes files. En effet, si le temps entre deux départs est supérieur à 9, ce qui correspond à un trafic fluide, il y a un seul client dans le système à chaque instant. Donc on n'observe pas de conflit entre les files et, par conséquent, pas d'oscillations.

VI.4 Conclusion

La discipline de Fair Queuing implantée respecte le trafic dans sa globalité. C'est-à-dire que, si on regroupe les clients consécutifs, les espaces vides entre ces groupes suivent une loi géométrique. Par contre, en considérant une file particulière, il apparaît que cette discipline de service perturbe le trafic, surtout si ce dernier est chargé. Ce phénomène est dû au mécanisme de choix qui est utilisé pour la désignation de la cellule à servir en cas d'égalité des marques.

Ce chapitre a donc montré que l'utilisation de ressources matérielles pour l'étude des disciplines FQ est largement envisageable. Dans ce chapitre, le calcul du temps virtuel est très simplifié, mais la sélection de la plus petite marque est réalisée sans simplification.

Le chapitre (VII) présente la modélisation d'un commutateur réel. Ce commutateur utilise une méthode de calcul du temps virtuel non simplifié. Par contre, le choix de la cellule à servir est effectué de manière à éviter le calcul du minimum.

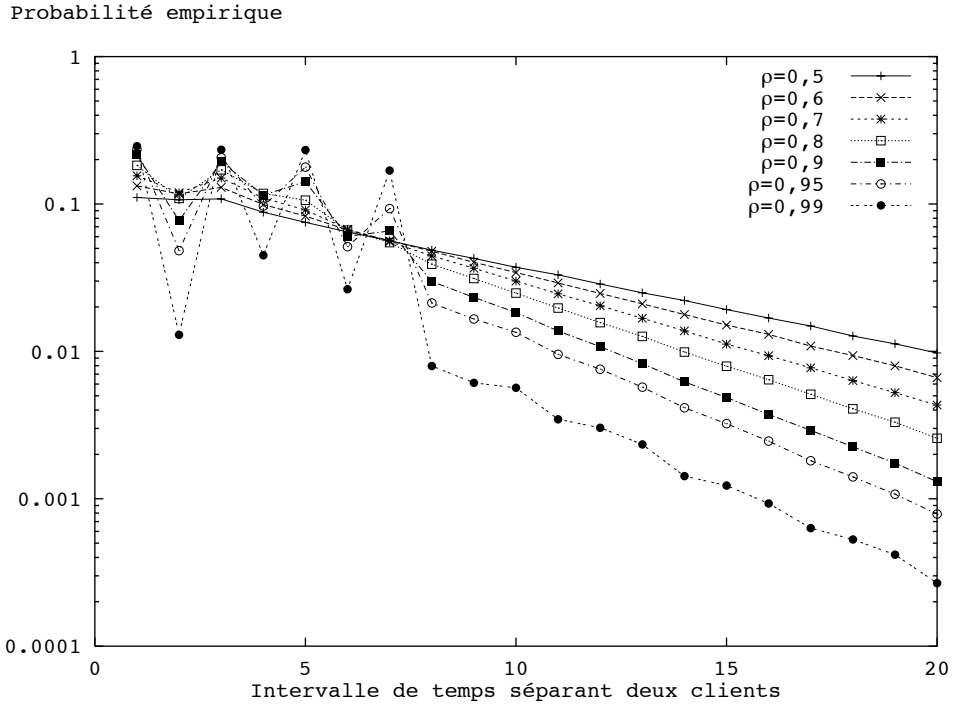

FIG. VI.6: Distribution empirique de la taille des intervalles de temps séparant les cellules d'un flux particulier en sortie du serveur FQ. Échelle logarithmique en ordonnée.

Bibliographie

[Abu98] O. Abuamsha. *Application des méthodes de la comparaison stochastique pour l'analyse des disciplines "Fair-Queueing"*. PhD thesis, PRiSM, Université de Versailles, 1998.

[BZ97] Jon C. R. Bennett and Hui Zhang. Hierarchical packet fair queuing algorithms. *IEEE/ACM Transaction on Networking*, 5, 1997.

[GM92] A. G. Greenberg and N. Madras. How fair is fair queuing? *J. of the ACM*, 39, 3 :568–598, 1992.

[Gre96] Albert G. Greenberg. Fair queueing architectures for high-speed networks. In *Proceedings of the Fourth International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, page 198, San Jose, California, feb 1996. IEEE Computer Society TCCA and TCS.

[GVC97] Pawan Goyal, Harrick M. Vin, and Haichen Cheng. Start-time fair queuing : A scheduling algorithm for integrated services packet switching networks. *IEEE/ACM Transaction on Networking*, 5, 1997.

[Sti96] Dimitrios Stiliadis. *Traffic Scheduling in Packet-Switched Networks : Analysis, Design, and Implementation.* PhD thesis, University of California Santa Cruz, 1996.

[SV98] D. Stiliadis and A. Varma. Efficient fair queuing algorithms for packet-switched networks. *IEEE/ACM Transaction on Networking*, 6, 1998.

Chapitre VII

Modélisation d'un commutateur réel (le CMS)

La mise au point, au CNET Lannion, d'un Commutateur Multi-Service (CMS) a demandé de nombreuses simulations. En particulier, pour les tests du régulateur de trafic que comporte le CMS. Ce régulateur de trafic a pour fonction d'adapter la taille des VP (Virtual Path) en fonction de l'état des files d'attente associées à chaque VC (Virtual Chanel). L'objectif est d'optimiser dynamiquement la taille des VP. Cet adaptateur est composé d'un algorithme neuro-mimétique qui doit être capable d'allouer un débit optimal aux VP.

Le test de ce régulateur nécessite un grand nombre de données, et la première version de celui-ci doit pouvoir être validée grâce à des simulations. Ces simulations doivent donc être capables de générer un grand nombre de données pour permettre de tester l'algorithme neuro-mimétique. Dans ce contexte, l'utilisation de ressources matérielles semble être parfaitement adaptée.

Ce chapitre présente donc la manière dont des ressources matérielles - et plus particulièrement comment la machine Sim-express - peuvent être utilisées dans un cadre de co-simulation. On verra qu'ainsi l'utilisation de ressources matérielles devient complètement transparente pour l'utilisateur.

La première section (VII.1) de ce chapitre présente les caractéristiques et les objectifs du CMS. Cette section expose aussi la modélisation du commutateur qui va être utilisée pour la mise au point du régulateur de trafic. La section (VII.2) présente les implantations qui ont été choisies pour la matérialisation de la plupart des composants nécessaires à la simulation (sources, files d'attente...). La section suivante (VII.3) détaille l'implantation du serveur (l'arbitre) utilisé dans le commutateur. La section (VII.4) montre comment la simulation est pilotée par une autre application. C'est-à-dire comment la machine Sim-express est couplée à l'application simulant le réseau neuro-mimétique de régulation de trafic. Enfin, la dernière section (VII.6) propose un bilan de cette expérience.

VII.1 CMS et Objectifs

Cette section présente, de manière rapide, les objectifs et les caractéristiques du CMS. Les principaux choix architecturaux du commutateur sont donc présentés. Le deuxième paragraphe présente la modélisation du CMS qu'il est nécessaire de réaliser pour pouvoir tester l'algorithme de régulation du trafic.

VII.1.a Le Commutateur Multi-Service

Objectifs et caractéristiques

L'objectif du CMS est, bien sûr, d'assurer le transfert de données à haut débit dans un réseau ATM, mais aussi de prendre en compte la spécificité des différents services de données. L'un des autres grands objectif du CMS est d'être capable d'optimiser l'utilisation du réseau en adaptant le débit alloué à chaque VP en fonction de l'état des VC.

Le CMS a donc pour objectif d'être capable d'offrir :

- la garantie d'un débit minimum pour chaque source,
- une perturbation minimale du trafic,
- pas (ou très peu) de pertes,
- l'utilisation maximum des ressources du réseau.

Ainsi, le commutateur doit être capable de fournir ces bonnes propriétés pour tous les types de trafic. Par exemple, un trafic sans exigence temps réel obtiendra un débit minimum garanti bas au contraire d'une communication ayant des exigences temps réel qui demandera un débit minimum plus important.

Les techniques utilisées

Pour garantir un débit minimum à chacune des sources tout en évitant au maximum les perturbations, les techniques de Fair Queueing sont particulièrement intéressantes (cf. chapitre VI). De plus, la réallocation du débit libéré par les connexions inactives permet d'optimiser l'utilisation des ressources.

Ainsi un algorithme de type "Weighted Fair Queueing" est utilisé dans le CMS. Cet algorithme permet de garantir :

- la réallocation du débit en temps réel (au niveau cellule) en partageant dynamiquement la bande passante entre les connexions actives (une connexion est active dans un nœud si le nombre de cellules de la connexion est non nul dans ce nœud). Cette réallocation permet de prendre en compte des informations au niveau du commutateur (activité de la connexion, présence de flux additionnels).
- l'indépendance entre les connexions grâce à l'utilisation d'une file d'attente logique par connexion dans chaque commutateur.

L'utilisation d'un algorithme d'espacement en entrée du réseau permet d'obtenir un flux de cellules correctement espacé en entrée des premiers commutateurs. Pour assurer la propagation de cette bonne propriété du trafic, chaque commutateur utilise un algorithme d'espacement. Ainsi chaque VP est correctement espacé à la sortie des commutateurs.

Un mécanisme de congestion bond par bond est utilisé pour limiter les pertes. Ce mécanisme de rétroaction fonctionne connexion par connexion. La congestion (ou la décongestion) d'une connexion dans un commutateur est décidée par le nœud aval. Le commutateur aval considère qu'une connexion est congestionnée si le nombre de cellules en attente dans la file dépasse un seuil haut. Dans ce cas, une notification de congestion est envoyée par l'intermédiaire d'une cellule RM. Cette émission n'a pas forcément lieu au moment du dépassement de seuil. En effet, l'état des files d'attente est testé périodiquement, les émissions de cellules RM sont donc elles aussi périodiques. Le temps de transfert des notifications de congestion dépend principalement de la distance entre les commutateurs. Le flux est considéré

comme décongestionné quand le nombre de clients repasse sous le seuil bas de la file. Dans ce cas, une notification de décongestion est émise. La politique de service de type FQ utilise le fait qu'une connexion est en congestion ou non pour l'allocation de son débit. Ainsi, une connexion active mais en congestion sera servie à son débit minimum (voir VII.3 pour plus de détails). Ceci peut permettre une décongestion en aval avant que le mécanisme de rétroaction ne remonte jusqu'à la source.

L'utilisation d'un arbitre de type FQ qui réalloue le débit du VP entre chaque VC permet aussi d'optimiser l'utilisation des ressources du réseau. Mais ceci n'empêche pas le VP d'être surdimentionné (ou sousdimentionné) par rapport au besoin des VC qui le composent. C'est pour cette raison que le CMS utilise un régulateur de VP capable d'ajuster au mieux la taille du VP par rapport à la demande des VC.

Le résultat

Avec l'utilisation de ces techniques, chaque connexion présente dans le réseau est caractérisée par :

- son couple VP/VC,
- son débit minimum garanti : Dmg,
- son débit crête : DC.

En plus de ces caractéristiques vraies sur tout le réseau, chacune des connexions possède des caractéristiques locales qui ne sont vraies que dans un commutateur :

- active ou inactive,
- libre ou en congestion,
- les seuils haut (SH) et bas (SB) des files qui lui sont associées.

Le couplage des méthodes du service FQ et des mécanismes de notification de congestion conduisent à définir un arbitre ayant les caractéristiques suivantes :

- toutes les connexions actives ont au moins un débit égal à leur Dmg. En effet, l'état de congestion n'affecte pas le Dmg. La congestion affecte uniquement le trafic participant au sur-débit de cette connexion.
- la bande passante allouée aux connexions inactives est réallouée dynamiquement aux connexions actives non congestionnées. Ceci au prorata de leur DC, celui ci joue le rôle des ϕ_i du chapitre (VI).

Pour éviter une recherche de minimum coûteuse dans l'implantation de l'arbitre (cf VI), on utilise une "ronde". Cette ronde est une mémoire dans laquelle sont rangés les identificateurs des flux à servir. Chaque mot de la mémoire correspond à un slot (un instant d'émission). Dans la case mémoire de la ronde d'adresse IC (instant courant) on lit l'identificateur du flux à servir et la date à laquelle il aurait dû être servi ITE (Instant Théorique d'Emission). Ceci permet de calculer, en fonction du débit instantané alloué au flux, le nouvel instant d'émission théorique ($NITE$) :

$$NITE = ITE + T_{inst},$$

où T_{inst} est la période instantanée associée au débit instantané. Le détail du calcul de la période T_{inst} sera présenté dans la section (VII.3).

L'identificateur du flux et son $NITE$ sont alors écrit dans la ronde à la première case (slot) libre suivant l'adresse $NITE$. Le fait d'utiliser un instant théorique d'émission pour

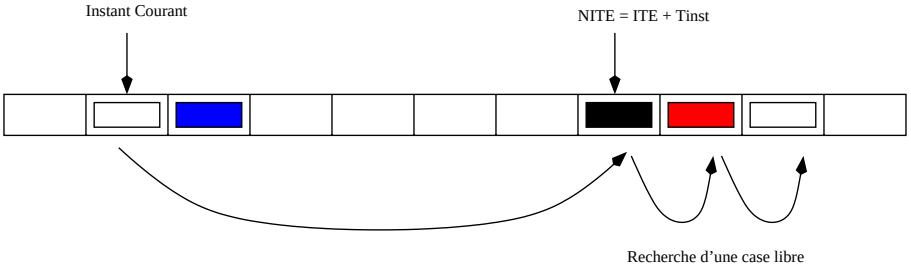

FIG. VII.1: Après avoir calculé le nouvel instant théorique d'émission ($NITE$) on trouve la case libre la plus proche pour y insérer l'émission suivante.

calculer la prochaine date de service permet d'éviter qu'un flux prenne trop de retard par rapport aux émissions souhaitées.

L'utilisation d'une ronde modifie les caractéristiques du serveur, en effet, il n'est plus "work conservative" : il peut y avoir des cellules en attente, mais pourtant aucun service n'est effectué.

Structure générale du commutateur

Le commutateur est composé de plusieurs cartes d'entrées qui réceptionnent les cellules sur les différents ports d'entrée. Les cellules sont stockées dans une mémoire partagée entre toute les connexions. Les cartes de sortie émettent les cellules. Dans chacune de ces cartes de sortie, un arbitre par VP gère les conflits d'émission entre les VC.

Chaque carte de sortie possède un mécanisme de dimensionnement des VP, implanté sur un DSP et basé sur un algorithme neuro-mimétique. C'est pour la mise au point de cet algorithme qu'une simulation sur la machine Sim-express va être effectuée.

VII.1.b La modélisation du commutateur

L'objectif est de mettre au point une méthode de régulation de la taille des VP. Le but de cette régulation est d'obtenir un taux de congestion moyen constant sur les différents VC du VP considéré. L'approche pour contrôler la taille du VP dans le cadre du CMS est basée sur la mesure de variables reliées au trafic. Le simulateur porté sur Sim-express doit donc être capable de générer ces variables nécessaires au contrôle du débit alloué au VP.

CMS modélisé

La modélisation qui a été définie avec les concepteurs du régulateur revient à considérer le commutateur vu d'un port de sortie (cf. figure VII.2). On étudie n sources qui arrivent dans n files d'attente. Ces files sont servies par un arbitre. Elles ont des seuils qui permettent de détecter les congestions. Un émetteur de congestion est aussi implanté pour modéliser le flux de notification de congestion en provenance de l'aval. Les notifications émises par

le commutateur simulé doivent mettre un certain temps (distance) pour parvenir jusqu'aux sources.

Dans une telle simulation utilisant des ressources matérielles, on peut distinguer deux types de paramètres. Les paramètres "statiques" nécessitant une "recompilation" du circuit en cas de modification, et les paramètres "dynamiques" qui sont modifiables en cours de simulation.

Les paramètres statiques sont :

- le nombre maximum de connexions (nombre de files d'attente à simuler),
- le type de sources (aléatoires, contrôlées par un LB, périodiques),
- le type de la source des notifications de congestion provenant de l'aval (aléatoires, périodiques).

Les paramètres dynamiques sont :

- les caractéristiques des sources (Dmg, DC, Débit courant, paramètres d'une loi aléatoire...),
- les caractéristiques du flot de retour en provenance de l'aval (paramètres d'une loi, période...),
- les seuils SB et SH des files d'attente et leurs capacités,
- les distances séparant les sources du commutateur.

Dans la suite du document, on présente chacun des éléments de la simulation : les sources, les distances, l'arbitre, les files d'attente, l'émetteur de notification et enfin l'instrumentation.

Récapitulatif

Les types de sources ont été choisis avec les concepteurs du régulateur. Ainsi, on a retenu des sources de cellules périodiques. En effet, le trafic étant contrôlé en entrée du réseau et à la sortie de chaque commutateur, on considère que l'on peut étudier un trafic périodique.

Par contre, on ne sait rien sur la manière dont apparaissent les congestions en aval du commutateur. Il a donc été choisi de modéliser le flux des notifications de congestion en provenance de l'aval par un processus géométrique.

Les paramètres d'entrée de la simulation modifiable par un utilisateur sans recompilation sont donc :

- les paramètres des sources (le Débit Crête, le Débit minimum garanti, et la période d'émission),
- les distances entre les sources et le commutateur (mesurées en temps cellules),
- la fréquence des congestions se produisant en aval,
- les seuils et les tailles des files d'attente,
- la période d'émission des notifications de congestion du commutateur vers l'amont (les sources).

Les résultats que doit fournir la simulation au régulateur de VP sont :

- l'historique de l'état de l'occupation des files d'attente (bas, moyen, haut),
- l'historique des notifications de congestion en provenance de l'aval.

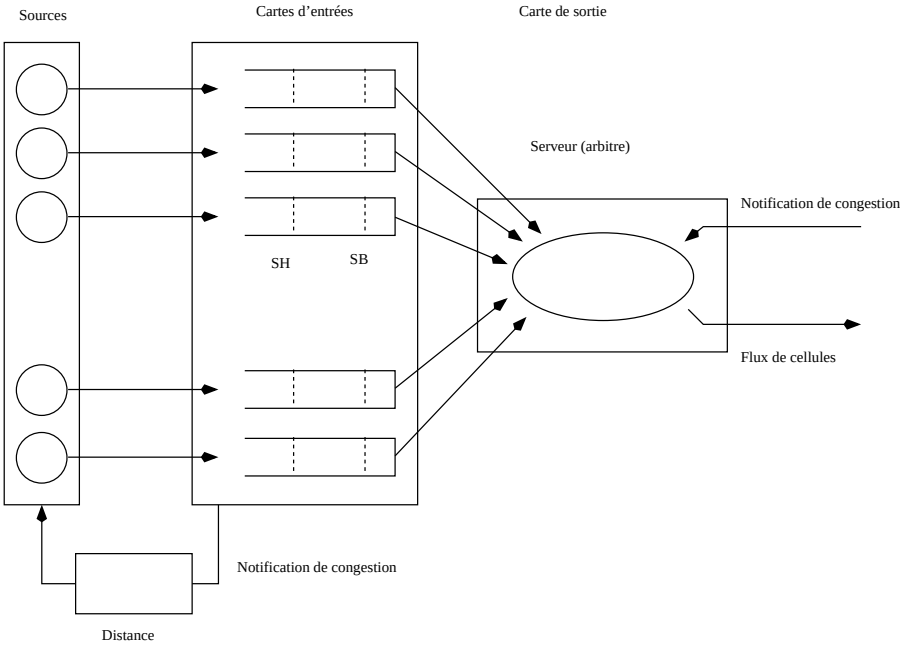

FIG. VII.2: La simulation du commutateur CMS que l'on souhaite réaliser. On considère n flux et les n files d'attente associées. Ces flux sont servis par un algorithme de type "Weighted Fair Queueing".

VII.2 Réalisation : les files, l'instrumentation, les distances...

VII.2.a Les sources

Les sources utilisées sont des sources ayant deux modes : un mode "libre" et un mode "congestionné". Ce sont des sources déterministes qui émettent des cellules de manière périodique. Les périodes d'émission courante font partie des paramètres de la simulation. En effet, l'échelle de temps des changements de période est grand vis à vis du temps cellule.

La source bascule de l'état libre à l'état congestionné en fonction des notifications de congestion envoyé par l'arbitre. Si DC est le débit crête et Dmg le débit minimum garanti alors, la période d'émission T de la source vérifie :

- en mode libre : $0 \leq T \leq \frac{1}{DC}$,
- en mode congestionné : $0 \leq T \leq \frac{1}{Dmg}$.

Les paramètres d'entrée du composant source sont :

- la période d'émission T.
- le flux de notification de congestion et de décongestion.
- les débits DC et Dmg.

La sortie d'un composant "source" est un flux de cellules périodique. Ce flux indique la présence ou l'absence d'une cellule par un bit. En effet, en reliant directement les sources aux files qui leur sont associées, les cellules n'ont pas besoin de contenir de l'information.

VII.2.b Délai induit par la distance

La distance permet de prendre en compte le temps de traversée du réseau. L'unité de mesure est le temps cellule. Les sources doivent pouvoir être disposées à une certaine distance du commutateur. Ceci pour modéliser le temps de réaction aux notifications de congestion.

Pour modéliser une distance de δ temps cellule entre le commutateur et les sources plusieurs solutions sont possibles. On peut par exemple, choisir d'introduire un délai δ sur les flux de cellules et le même délai δ sur le flot des notifications de congestions. La solution adoptée est différente, elle consiste à introduire un délai 2δ sur les notifications de congestion et aucun délai entre les sources et le commutateur. Dans les deux cas, on modélise bien un "temps de réaction" de 2δ.

Pour réaliser le délai sur les flux deux solutions sont possibles :

- L'utilisation d'une mémoire pour stocker les notifications de congestion pendant 2δ slots. Les notifications sont réémises 2δ slots après avoir été écrites en mémoire.
- Si on considère que l'intervalle de temps entre deux émissions du flux de notification est supérieur à 2δ, la mémoire est inutile et un registre est utilisé pour stocker la seule notification en "transfert".

C'est cette dernière solution qui a été retenue pour la simulation.

VII.2.c Les files

On utilise des files sans contenu, en effet, on considère que l'information transportée par les cellules n'est pas pertinente pour l'étude. Seule la présence ou non d'une cellule est importante. Dans ce cas, un flux de cellules est réduit à un flux de 1 bit indiquant la présence ou non d'une cellule. Les files utilisées sont celles présentées dans la section (III.3). L'état de congestion sera testé par un autre composant : le notificateur de congestion.

Paramètres d'entrée du composant file :

- un flux de cellules.
- une capacité.
- un signal de service (indique si l'on doit servir une cellule).

Paramètres de sortie :

- un flux de cellules.
- l'occupation, les pertes.

VII.2.d Notificateur de congestion

Le notificateur de congestion génère le flux des notifications de congestion ou de décongestion en fonction des états des différentes files d'attente. Les notifications sont émises de manière périodique. Cette période est l'un des paramètres de la simulation.

Paramètres d'entrée du composant notificateur :

– une période d'émission.
– l'occupation des différentes files d'attente.

Paramètres de sortie : Un flux de notification de congestion en direction des composants distances.

VII.2.e Instrumentation

Les informations que l'on souhaite récolter sur le système sont : l'historique des notifications de congestion en provenance de l'aval et l'historique de l'état des files. L'instrumentation est donc constituée de deux composants de type historique utilisant des mémoires (voir III.3).

VII.3 L'arbitre

L'arbitre pilote les signaux de service des files d'attente. Ceci se fait en fonction des paramètres des connexions et de leurs états (active, inactive, congestionnée ou libre). Les différents débits sont exprimés en fonction d'un nombre de granules. Le granule est le plus petit débit allouable.

VII.3.a Présentation des contextes et notations

Ce paragraphe présente les différents contextes qui permettent à l'arbitre de calculer le $NITE$ d'une connexion i durant l'instant courant IC.

Contexte fixe général : débit alloué au VP, noté Dvp.

Contexte fixe de la $ième$ connexion :
– le débit minimum garanti, noté Dmg_i,
– période du Dmg, notée Tmg_i,
– le débit crête, noté DC_i,
– période minimum, notée Tc_i.

Variables internes attachées à chaque connexion :
– l'état de la connexion (active, inactive, congestionnée, libre),
– l'instant théorique d'émission, noté ITE_i.

On dit qu'une connexion est en réallocation si elle est active et non congestionnée. Elle profite donc de la réallocation du débit laissé libre par les connexions inactives.

On notera par $\mathcal{R}$ l'ensemble des connexions en réallocation et par $\mathcal{A}$ l'ensemble de connexions actives à l'instant IC. $\mathcal{E}$ désignera l'ensemble de toutes les communications.

Variables générales :
– débit total virtuellement alloué, noté $Dtot = \sum_i DC_i \ i \in \mathcal{R}$,
– débit restant à partager, noté $Dp = Dvp - \sum_j Dmg_j \ j \in \mathcal{A}$.

Un certain nombre de simplifications étaient envisageables, aucune d'elles n'a été retenue :
– $Dtot = \sum_i DC_i \ i \in \mathcal{E}$,
– $Dp = Dvp - \sum_j Dmg_j \ j \in \mathcal{E}$.

On note Tn_i la période d'émission instantanée (au slot n) et Dn_i le débit alloué en plus du Dmg_i au slot n.

VII.3.b Calcul NITE

À chaque temps cellule, on lit dans la ronde le numéro de la file qui doit être servie et on calcule le $NITE$ de cette connexion. Si la case de la ronde est vide, on ne fait rien.

Pour le calcul du $NITE$ quatre cas sont possibles :

- 1^{ier} Cas. Si une connexion s'active (arrivée d'une cellule dans une file vide) alors,

$$Tn_i = \frac{1}{2Dmg_i}$$

et

$$NITE = IC + Tn_i$$

- $2^{ième}$ Cas. Si la connexion à servir se désactive (on sert la dernière cellule de la file) alors le calcul de la $NITE$ n'a pas lieu d'être.
- $3^{ième}$ Cas. Si la connexion à servir est en congestion :

$$Tn_i = \frac{1}{Dmg_i}$$

et

$$NITE = ITE + Tn_i$$

- $4^{ième}$ Cas. Si la connexion à servir est en réallocation :

$$Dn_i = \frac{DC_i}{Dtot}.Dp$$

$$Tn_i = \frac{1}{Dmg_i + Dn_i}$$

et

$$NITE = ITE + Tn_i$$

VII.3.c Implantation à l'aide de ressources matérielles

Tous les débits sont codés par des entiers sur 10 bits. Les débits sont donc exprimés en pour 1024. Le débit alloué au VP de sortie est toujours le même par construction $Dvp = 1024$. Le slot du VP est l'unité de temps de la simulation.

Fonctionnement général

La figure (VII.3) tente de montrer le fonctionnement du composant simulant l'arbitre. Le travail que doit réaliser ce composant à chaque slot est le suivant :

- Il faut insérer le nouveau trafic dans la ronde (les connexions qui viennent de s'activer).
- En cas de service d'une file, il faut, si nécessaire, faire le calcul du $NITE$. Il n'est pas forcément utile de réaliser ce calcul à chaque top. Il est mis à jour uniquement quand les contextes changent.
- Si on n'a pas servi la dernière cellule de la file d'attente, il faut faire la recherche dans la ronde de la première place libre.

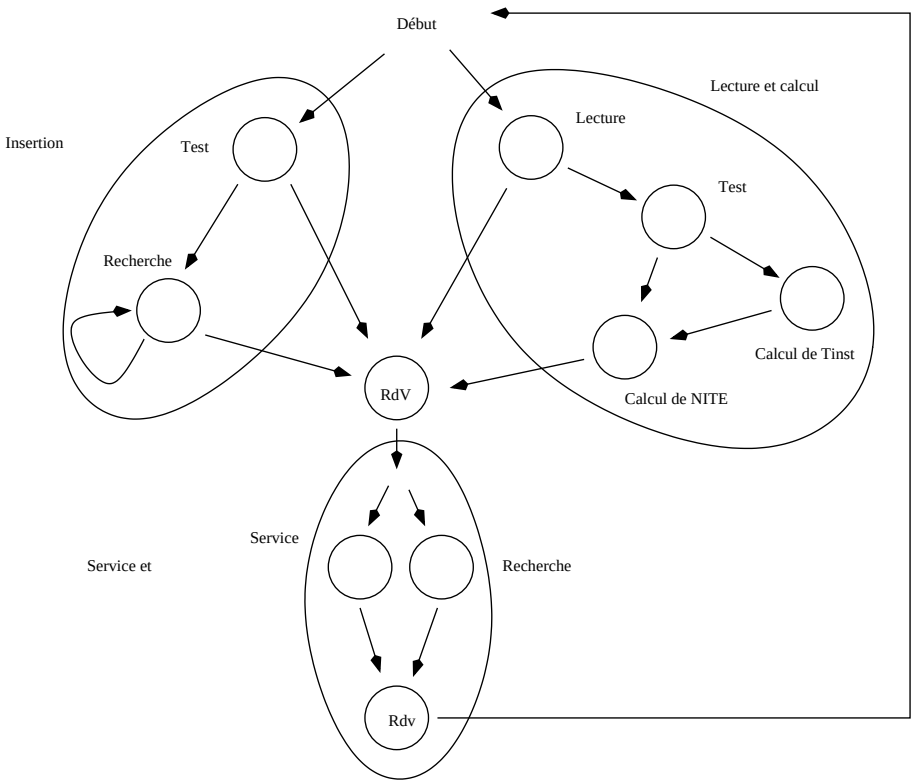

FIG. VII.3: L'automate de l'arbitre, composé principalement de trois parties. (1)L'insertion dans la ronde lors d'une activation. (2) Lecture dans la ronde et le calcul du $NITE$. (3) Service et placement dans la ronde de la prochaine émission.

La ronde et les vecteurs d'état

La ronde est donc une mémoire qui contient le numéro de la file à servir et l'ITE de cette cellule. Deux vecteurs d'état sont utilisés pour connaître l'état de chacun des flux. Le premier contient l'information active/non active, le second contient l'information congestion/libre.

Si l'un de ces deux vecteurs change (changement de contexte) les valeurs des périodes instantanées (Tn_i) attribuées à chaque connexion doivent être mises à jour. Un troisième vecteur de bits permet de savoir si un Tn_i est à jour ou non. En cas de changement de contexte, ce vecteur est mis à zéro et à chaque fois qu'un calcul de Tn_i est réalisé, la composante du vecteur est mis à 1 et le résultat du calcul est stocké dans une table.

Insertion

Une partie de l'arbitre doit réaliser l'insertion du nouveau trafic dans la ronde. Toute connexion j qui s'active doit être insérée dans la ronde. Chacune de ces insertions lance une recherche de la première case libre à partir de $NITE_j = IC + \frac{1}{2Dmg_j}$. Cette insertion se

fait en parallèle avec la lecture à l'adresse IC de la numéro i du flux à servir et le calcul du $NITE_i$.

Lecture et calcul

La lecture dans la ronde à l'adresse IC. Si la case est vide, il n'y a rien à faire. Si on lit un numéro i de file et un ITE_i, il faut regarder si Tn_i est à jour. Si c'est le cas, le calcul du $NITE_i$ peut être fait immédiatement. Sinon (Tn_i n'est pas à jour), si le calcul des sommes est à jour on peut calculer Tn_i. Dans le cas contraire (le contexte vient de changer), il faut les calculer avant de mettre à jour la valeur de Tn_i.

Le calcul de Tn_i fait intervenir des composants de multiplication et de division identiques à ceux présentés dans le paragraphe (III.2.b).

Service et recherche

Une fois les insertions et le calcul réalisés on va pouvoir effectuer le service et la recherche de la première case libre dans la ronde à partir de l'adresse $NITE_i$.

VII.3.d Paramètres d'entrée/sorties

L'arbitre pilote les signaux de service des files d'attente, ces paramètres d'entrées sont :

- les paramètres des flux (DC et Dmg),
- les notifications de congestion en provenance de l'aval,
- état des files d'attente (congestion ou réallocation).

En sortie : le numéro du flux servi.

VII.4 Synchronisation avec une application extérieure

Cette section montre comment coupler une application logicielle avec la machine. La communication entre les applications se fait par l'intermédiaire de fichiers. La machine Simexpress est pilotée par un script.

VII.4.a Déroulement de la simulation

À la mise en marche de l'émulateur un script de contrôle est automatiquement lancé. Ce script de contrôle initialise la machine et la configure avec le circuit modélisant le CMS, ensuite il comporte quatre phases :

- Phase 0 : Phase d'attente. L'horloge de l'émulateur est arrêtée, le circuit est donc gelé. Le script attend le signal de départ en scrutant un fichier. Quand ce signal est lu, on passe à la phase 1.
- Phase 1 : Lecture d'un fichier qui contient toute les informations nécessaires à la simulation, taille des files, les seuils, le germe du générateur aléatoire.... (cf format dans le README de l'annexe). Ce fichier contient aussi le nombre de slots (temps cellule) que l'émulateur doit simuler. Une fois ces paramètres transmis à l'émulateur on passe à la phase 2.

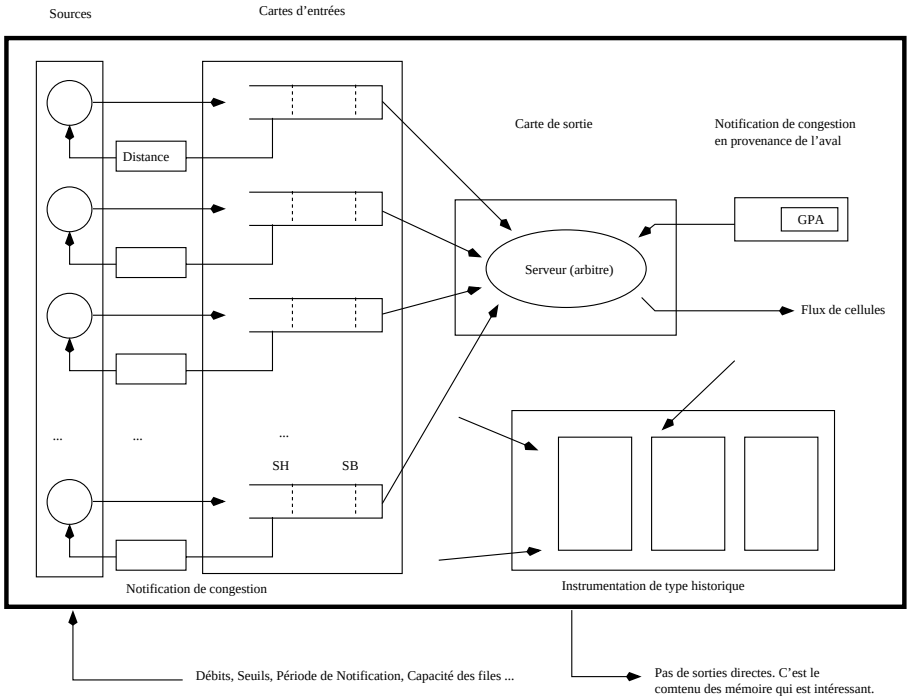

FIG. VII.4: Le composant qui simule le comportement du commutateur CMS. Les informations sont stockées dans les mémoires de l'émulateur.

- Phase 2 : L'émulateur simule le nombre de slots voulu... Les différents états des files d'attente ainsi que les notifications de congestion en provenance de l'aval sont collectés dans des mémoires. Si ces mémoires d'instrumentation sont saturées, leur contenu est vidé et concaténé dans deux fichiers différents. L'un pour l'historique des notifications de congestion et l'autre pour l'historique de l'état des files.
- Phase 3 : Après avoir simulé le nombre de slots voulu, les mémoires d'instrumentation sont vidées et concatenées dans les deux fichiers contenant les historiques. Puis, on retourne à la phase 0.

On sort de cette boucle quand le nombre de slots à simuler (lu lors de la phase 1) est égal à zéro. La figure (VII.5) montre le déroulement d'une simulation du CMS. La partie concernant le service des cellules est simulé à l'aide de ressources matérielles et la partie concernant le régulateur de VP à l'aide de ressources logicielles.

VII.4.b Paramètres d'entrée

Les paramètres d'entrée sont fournis par l'utilisateur dans un fichier qui est lu par le script (phase 1) qui pilote la machine Sim-express.

Ce fichier comporte :

- le nombre de slots que doit simuler l'émulateur. Si ce nombre est zéro la simulation s'arrête (on sort du script de contrôle).

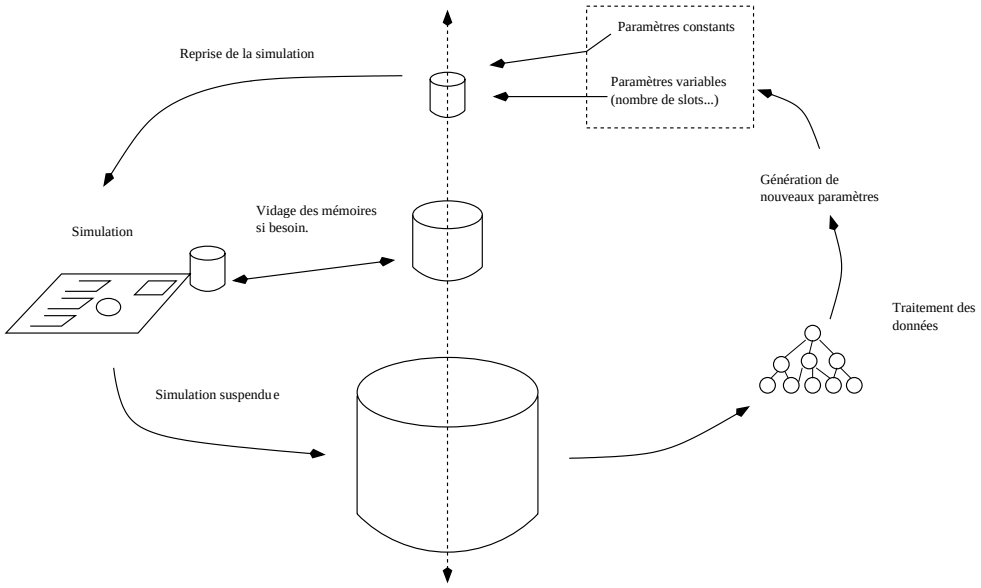

FIG. VII.5: Le déroulement d'une simulation. La machine Sim-express est couplée à l'application de régulation du VP

- La probabilité d'émission d'une notification de congestion en provenance de l'aval. Cette probabilité est donnée en proportion de 65536. Ainsi 6553 correspond à une probabilité de 0.1.
- Le germe du générateur aléatoire. 32 bits en hexadécimal.
- La période d'émission des notifications de congestion de l'arbitre. C'est la période avec laquelle les états des files sont évalués.

Le fichier comporte aussi les paramètres de chaque flux et de la file d'attente qui lui est associée. Pour chaque flux sept paramètres doivent être donnés :

- La capacité de la file,
- Le seuil bas de la file,
- Le seuil haut de la file,
- Le débit crête du flux (donné en pour 1024),
- Le débit minimum garanti du flux (donné en pour 1024),
- Le débit courant de la source associé au flux (donné en pour 1024),
- La distance en temps cellule entre l'arbitre et la source du flux.

VII.4.c Paramètres de sortie

Quand les mémoires sont pleines, elles sont vidées et leur contenu est concaténé dans deux fichiers. Quand le nombre de slots voulu a été simulé, le simulateur est mis en attente pendant que l'algorithme neuro-mimétique traite les données.

Le premier fichier contient les dates des changements d'état des files. Ce fichier est la concaténation des contenus de la mémoire d'instrumentation qui enregistre cet historique. Ce fichier est un fichier ASCII dans lequel on peut lire la date du changement d'état (la date est codée sur 32 bits), ainsi que pour chaque file d'attente, 2 bits donnant leurs états.

Le deuxième fichier contient l'historique sur les notifications de congestion. Ce fichier est la concaténation des contenus de la mémoire d'instrumentation qui enregistre cet historique. Ce fichier est un fichier ASCII dans lequel on peut lire la date (la date est codée sur 32 bits) du changement de valeur du vecteur de notification de congestion/décongestion en provenance de l'aval. Ce vecteur contient un bit pour chaque flux qui indique si ce flux est en congestion ou non en aval.

VII.5 Exemple d'utilisation[1]

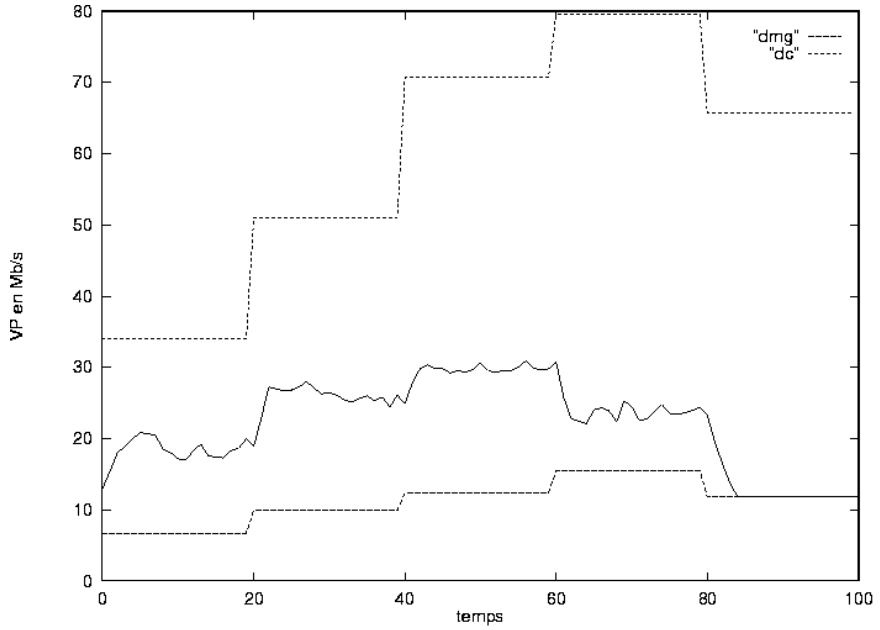

FIG. VII.6: Valeur du VP en fonction du temps. Toutes les minutes (20 unitées de temps), un changement de tous les contrats de trafic a lieu (32 sources).

Les expériences menées depuis Lannion et utilisant le simulateur ont permis de générer des courbes telles que celles des figures (VII.6) et (VII.7).

Dans l'expérience présentée ici, l'application neuro-mimétique redimensionne la taille du VP pour une période de $H = 5$ secondes (l'horizon de prédiction). Les émissions de notifications de congestion ont une période de $T = 1$ milliseconde. Ainsi, le commutateur

[1] Cette section expose le travail réalisé par Raphael Ferraux (CNET Lannion) pour la mise au point du régulateur de trafic à l'aide de l'émulateur.

fournit son état E à chaque période T. Tous les H, le régulateur de VP réalise un calcul du type :

$$Dvp(t+H) = F(E_{t-kT}, ..., E_{t-T}, E_t).$$

Le nombre de sources (VC) gérées par le commutateur est de 32. Ainsi, pour $VP = 30Mb/s$, chaque itération de l'agorithme (5 secondes) nécessite la simulation de 1,7 million de temps cellules soit 20 millions de slots pour la simulation d'une minute de fonctionnement réel. Les figures montrent la simulation de 5 minutes de temps réel. Cette simulation a donc demandé approximativement l'émulation de 100 millions de slots.

La figure (VII.6) montre trois courbes qui correspondent à la somme des DC, à la somme des Dmg, et à la bande passante allouée au VP par le régulateur au cours du temps. La figure (VII.7) montre que le taux moyen de sources congestionnées oscille autour de 20% ce qui été l'objectif. D'autre part, l'utilisation du VP est bien toujours proche de 1. Lors de la dernière minute, la somme des débits associés aux sources est inférieur à la somme des Dmg ce qui explique que le VP, ne soit pas rempli.

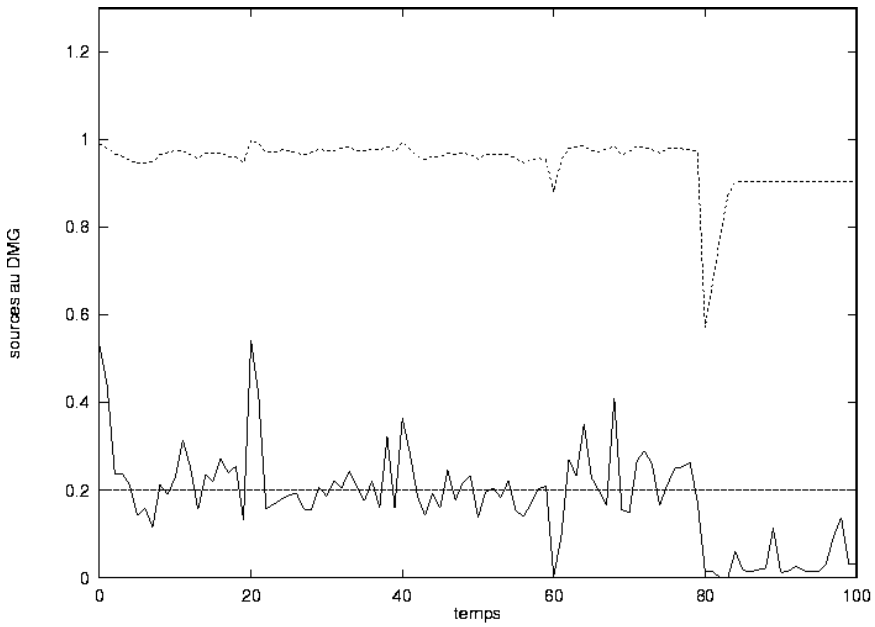

FIG. VII.7: Le pourcentage de sources congestionnées au cours du temps oscille autour de 20%. D'autre part, le pourcentage d'utilisation du VP reste proche de 1.

VII.6 Conclusion

Ce simulateur composé de deux applications, l'une constituée à l'aide de ressources matérielles et l'autre logicielle a permis de simuler 64 files d'attente. Pour une occupation de la machine de l'ordre de 17 cartes (70 % de la capacité totale).

Dans cette simulation, le slot n'est pas constant, selon les cas il peut durer de 3 à $2n$ tops horloge avec n le nombre de connexions. En effet, un slot pendant lequel aucun calcul et aucune insertion ne doit être effectué, dure beaucoup moins longtemps qu'un slot où toutes les connexions s'activent.

Tous les traitements sont réalisés en local sur la station qui pilote la machine Sim-express. Cela a permis à l'équipe qui développe le régulateur de VP d'utiliser le simulateur en utilisant le réseau interne du CNET.

L'utilisation de ressources matérielles comme simulateur de commutateur pour la génération d'un grand nombre de données a été appréciée. Les données ainsi obtenues ont été utilisées pour le calibrage et le test d'un algorithme neuro-mimétique de régulation de trafic. Les résultats ont été satisfaisants et largement supérieurs à ceux que l'on aurait pu attendre d'un simulateur logiciel.

La capacité des nouvelles générations de machine reconfigurable aurait permis la simulation de plusieurs commutateurs. Ceci aurait pu être utilisé pour le calibrage des files, des seuils, des périodes d'émission du mécanisme de contrôle des congestions.

Le couplage matériel/logiciel permet d'étudier les conséquences, à l'échelle des cellules, des décisions prises à l'échelle de la connexion (notification de congestion).

Conclusion

La thèse avait pour objectif de valider une méthode de simulation pour l'étude des réseaux à haut débit. Cette méthode consiste, dans un premier temps, à modéliser le fonctionnement d'un réseau notamment à l'aide de files d'attente. Ces files sont ensuite décrites à l'aide de ressources matérielles (portes logiques, registres et mémoires) grâce au langage VHDL. Le circuit ainsi décrit comprend une partie dite "d'instrumentation" qui récolte des informations sur le comportement du système étudié. Enfin, l'utilisation de la machine à architecture reconfigurable Sim-express permet d'émuler le comportement de ce circuit qui modélise le réseau.

Nous avons montré qu'un certain nombre de techniques utilisées pour les réseaux peuvent être testées et étudiées de manière efficace en appliquant une telle démarche. En particulier, l'application de cette méthode a permis la mise en évidence de phénomènes difficilement observables à l'aide de techniques classiques. En effet, en utilisant une architecture reconfigurable, on profite pleinement du parallélisme intrinsèque du matériel. C'est principalement cette capacité qui a été utilisée.

D'une part, la mise en évidence d'événements rares a été possible, notamment pour l'estimation de taux de pertes réalistes dans les réseaux ATM. D'autre part, il est difficile de déterminer l'impact des décisions prises à l'échelle de temps des connexions sur les phénomènes qui apparaissent à l'échelle de temps des cellules. En effet, là encore, pour comprendre ces phénomènes complexes, il est nécessaire de générer un grand nombre d'événements. L'utilisation d'une machine à architecture reconfigurable a permis de réaliser ce type d'étude. Ainsi, les ressources matérielles ont servi à simuler un commutateur à l'échelle du temps cellule. En couplant Sim-express à un simulateur logiciel de régulation de trafic, l'étude des interactions entre les différents niveaux d'échelle a pu être effectuée.

Le travail réalisé

Le travail réalisé pendant la thèse a conduit à l'élaboration d'une bibliothèque de composants écrit en VHDL. Ces composants sont génériques et peuvent être réutilisées pour d'autres applications. Cette bibliothèque comprend notamment : des files d'attente, des générateurs de lois pseudo-aléatoires, des composants d'instrumentation, des calculateurs de fonctions tabulées et des composants réalisant des multiplications et des divisions à moindre coût... Pour chacun de ces composants, différentes architectures sont possibles. Ainsi, selon l'optimisation que l'on souhaite réaliser (temps ou occupation), on peut choisir parmi ces différentes options. Il faut noter qu'au cours de l'élaboration de ces composants certaines architectures ont été testées puis écartées.

Ces composants ont été utilisés pour traiter trois problèmes différents. Lors de la réalisation de ces applications, la souplesse du matériel a permis de choisir la méthode la plus adaptée au problème en fonction des contraintes de coût.

La première application présentée a permis l'étude des réseaux multi-étages. Cette architecture a montré que l'on pouvait mettre en évidence des événements rares, tout en réalisant une étude sur les propriétés du trafic. On a ainsi montré comment les caractéristiques du trafic sont modifiées par le passage dans des commutateurs simplifiés. De plus, après complexification du modèle initial, on a pu obtenir des informations sur la gigue introduite par un trafic de fond sur une connexion.

La description d'une architecture matérielle, permettant l'étude des politiques de type Fair Queuing, a montré qu'il était possible d'utiliser des ressources matérielles pour l'étude de ces politiques. De plus, les expériences réalisées ont mis en évidence l'importance du choix en cas d'égalité des marques.

Enfin, la description d'une architecture modélisant le commutateur CMS et la mise au point d'un script de contrôle de la machine ont permis la construction d'un simulateur de commutateur. Ce simulateur a été utilisé comme générateur de données pour une application de régulation de trafic. Les ressources matérielles ont été ainsi utilisées par un simulateur logiciel pour accélérer une partie de l'exécution et mettre en évidence les interactions entre les différents niveaux d'échelle.

Perspectives

Prolongements immédiats du travail

Le travail réalisé dans la thèse peut être prolongé de différentes manières. En particulier, la librairie de composants peut être utilisée pour continuer les études réalisées.

Ainsi, l'étude sur les réseaux multi-étages peut être affinée. On peut, par exemple, remplacer les sources utilisées par des traces de trafic réel ou par des sources aléatoires contrôlées par un algorithme de lissage. On peut aussi implanter des politiques de services plus évoluées. Ceci peut permettre de vérifier la propagation des bonnes propriétés du trafic dans les étages du commutateur, et tenter de mesurer l'impact de ces politiques sur des indices de performance comme les taux de perte. De plus, l'utilisation du modèle de file dite "avec information" peut permettre de mesurer des délais de bout en bout.

La partie sur l'étude des politiques de Fair Queuing peut aussi être approfondie. En particulier, il serait intéressant de mener une évaluation plus poussée des propriétés statistiques des différentes politiques FQ. De plus, l'utilisation de ressources matérielles présente l'avantage de rendre très visibles les problèmes de coût qui se posent lors de l'implantation réelle de ces politiques.

La modélisation du Commutateur Multi-Service bien qu'ayant rempli le rôle qui lui était attribué (le test du régulateur de VP) peut elle aussi donner lieu à de nombreuses expérimentations. En effet, dans le CMS, un grand nombre de paramètres sont fixés "au jugé". Par exemple, la taille des files d'attente ou leurs seuils haut et bas sont calibrés de manière empirique. Il est donc intéressant d'en tester différentes valeurs. De plus, une étude peut être menée pour mieux comprendre la manière dont fonctionne le mécanisme de décongestion. Ainsi, y a-t-il un risque pour que des sessions passent en permanence de l'état congestionné à l'état libre ? Les congestions sont-elles équitablement réparties ?...

Dans ce contexte, il faut remarquer que les nouvelles générations de machine à architecture reconfigurable fournissent des capacités vingt fois supérieurs à celle de Sim-express. Il est donc possible, à l'aide de telles machines, de modéliser plusieurs commutateurs.

Architectures reconfigurables pour les réseaux

L'utilisation de ressources reconfigurables pour l'étude d'un problème peut être réalisée de différentes manières. Ainsi, la machine FAST est un exemple d'une machine à architecture reconfigurable dédiée à l'étude des protocoles réseaux. Ce type d'approche a l'avantage d'offrir une configuration optimisée pour le type de problème que l'on souhaite traiter, cependant de telles configurations sont, par nature, peu évolutives. L'utilisation de telles machines dans un contexte parfois légèrement différent ne sera pas forcément possible.

L'utilisation d'une machine provenant de la conception de circuits permet de conserver une grande liberté dans le choix des phénomènes que l'on souhaite étudier. Ce type de machine peut être vu comme un émulateur non spécialisé capable d'émuler n'importe quel circuit et donc en particulier un circuit modélisant un réseau. Une telle machine peut permettre le test de nombreux algorithmes. Comme l'a montré notre travail, elle peut être utilisée pour tester des techniques de régulation de trafic ainsi que différentes politiques de service. Cependant, notre démarche peut être utilisée pour l'évaluation d'autre types de protocoles comme, par exemple, des procédures de CAC ou des mécanismes de routage. L'utilisation d'une architecture reconfigurable permettant de ne pas perdre de vue les phénomènes qui se produisent à l'échelle de temps des cellules. On pourrait aussi calculer des probabilités d'existence de chemins dans un réseau où les liens ont une certaine probabilité de panne.

La "co-simulation", un avenir ?

Il faut aussi noter que le couplage matériel - logiciel peut servir dans bien d'autres domaines. De manière générale, on peut se demander si le type de co-simulation réalisée dans le chapitre (VII) a un avenir dans le cadre de la simulation de systèmes complexes. En effet, ce type de technique ne s'applique pas seulement à des modèles de type files d'attente. Des expériences pourraient être menées sur des réseaux de Petri, sur des graphes complexes ou encore dans le cadre de simulatioons de type Monte-Carlo.

Ainsi, de manière générale, tous les problèmes nécessitant un grand nombre d'opérations simples et pouvant être fortement parallélisés peuvent potentiellement être traités de manière efficace avec une architecture reconfigurable.

Le dernier chapitre montre un exemple où, grâce aux outils qui permettent de contrôler la machine Sim-express, les limites entre les fonctions logicielles et les fonctions réalisées à l'aide de circuits reconfigurables s'estompent. L'utilisation de l'architecture reconfigurable sert alors de "cartes accélératrices" pour les parties coûteuses de la simulation. Il faut ajouter que, au-delà de l'efficacité de la méthode en termes de vitesse sur un problème donné, la démarche reste particulièrement intéressante dans le sens ou l'on utilise une machine dont l'architecture change en fonction du problème à résoudre. La démarche consiste donc à imaginer une architecture capable de résoudre au mieux le problème posé. On pourrais ainsi imaginer, par exemple dans le cadre de la programmation orientée objet, la construction d'applications qui mélangeraient de manière transparente les objets "matériels" et les objets "logiciels".

Bibliographie

[10793] IEEE Std 1076-1993. *IEEE Standard VHDL Language Reference Manual*. IEEE, Septembre 1993.

[ABO90] R. Airiau, J.-M. Berge, and V. Olive. *VHDL du langage à la modélisation*. Presses polytechniques et universitaires romandes, France Telecom, 1990.

[ABO94] R. Airiau, J.-M. Berge, and V. Olive. *Circuit Synthesis with VHDL*. Kluwer Academic Publishers, France Telecom, 1994.

[ABO98] R. Airiau, J.-M. Berge, and V. Olive. *VHDL language, modélisation, synthèse*. Presses polytechniques et universitaires romandes, France Telecom, 1998.

[Abu98] O. Abuamsha. *Application des méthodes de la comparaison stochastique pour l'analyse des disciplines "Fair-Queueing"*. PhD thesis, PRiSM, Université de Versailles, 1998.

[AC91] P. Adam and J-P. Coudreuse. Atm et réseau : domaines d'application. *L'écho des recherches*, 145 :33–44, 1991.

[Ald89] D. Aldous. *Probability Approximations via the Poisson Clumbing Heuristic*. Springer-Verlag, 1989.

[Alt99] *http ://www.altera.com*. Altera, 1999.

[AM95] R.Y. Awdeh and H.T. Mouftah. Survey of ATM switch architectures. *Lecture Notes in Computer Science*, 27 :1567–1613, 1995.

[BDM92] Marc Boisseau, Michel Demange, and Jean-Marie Munier. *Réseaux haut débit*. Eyrolles, 1992.

[Ber93] Patrice Bertin. *Mémoire actives programmables : conception, réalisation et programmation*. PhD thesis, Université Paris 7, 1993.

[Bey93] André-Luc Beylot. *Modèles de Trafics et de commutateurs pout l'Evaluation de la Perte et du Délai dans les Réseaux ATM*. PhD thesis, Université Paris VI, 1993.

[BGSC92] Pierre E. Boyer, Fabrice M. Guillemin, Michel J. Servel, and Jean-Pierre Coudreuse. Spacing cells protects and enhances utilization of atm network's links. *IEEE Networks*, pages 38–48, 1992.

[BKYB95] André-Luc Beylot, Josephina Kohlenberg, Haikel Yaiche, and Monique Becker. Performance analysis of an atm switch based on a three stage clos interconnection network under non-uniform ibp traffic patterns. In *Proceedings of the First Workshop on ATM Traffic Management WATM'95*, page 263, Paris France, Déc 1995. IFIP.

[BM93] André-Luc Beylot and Becker M. Performance analysis of an atm switch based on a three-stage clos interconnection network. Technical Report MASI 92.45, INT-MASSI, 1993.

[BR96] L. Burgun and F. Reblewski. Première génération d'emulateurs matériels meta-systems. In *Quatrième symposium sur les architectures nouvelles de machines*, Metasystems, France, 1996.

[BRFL96] L. Burgun, F. Reblewski, G. J. Fenelon, Barbier, and O. Lepape. Serial fault emulation. In *Proceedings of the 33rd Design Automation Conference 1996 (DAC 96)*, pages 801–806, Metasystems, France, 1996.

[BZ97] Jon C. R. Bennett and Hui Zhang. Hierarchical packet fair queuing algorithms. *IEEE/ACM Transaction on Networking*, 5, 1997.

[CLR94] Thomas Cormen, Charles Leiserson, and Ronald Rivest. *Introduction à l'algorithmique*. Dunod, Massachusetts Institute of Technology, 1994.

[Com99] *http ://www.compugen.co.il/products*. 1999.

[Cou91] J-P. Coudreuse. Atm : principes généraux. *L'écho des recherches*, 144 :7–19, 1991.

[DA93] Alan Dolan and Joan Aldous. *Networks and Algorithms*. Johnn Wiley & Sons, 1993.

[Dev97] Luc Devroye. Random variate generation for multivariate unimodal densities. *ACM, Transactions On Modeling and Computer Simulation*, 7-4, 1997.

[ED96] Pierre-Jean Erard and Pontien Déguénon. *Simulation par événements discrets*. Presses polytechniques et universitaires romandes, 1996.

[Ent98] K. Entacher. Bad subsequences of well-known linear congruential pseudo-random number generators. *ACM, Transactions On Modeling and Computer Simulation*, 8-1, 1998.

[FLP98] Erwan Fabiani, Dominique Lavenier, and Laurent Perraudeau. Loop parallelization on a reconfigurable coprocessor. In *Workshop on Design, Test and Applications*, IRISA, Rennes, 1998.

[FMH93] W. Fischer and K. Meier-Hellstern. The markov-modulated poisson process (MMPP) cookbook. *Performance Evaluation North-Holland*, 18, 2 :149–171, 1993.

[FMP97] J.-M. Fourneau, L. Mokdad, and N. Pekergin. Bounding the loss rates in a multistage ATM switch. *Lecture Notes in Computer Science*, 1245 :193–205, 1997.

[For96] ATM Forum. *Traffic management specification*. Version4.0, Avril 1996.

[FPT95] J.-M. Fourneau, N. Pekergin, and H. Taleb. An application of stochastic ordering to the analysis of buffers with SBBP arrivals. In *Proceedings of the First Workshop on ATM Traffic Management WATM'95*, page 263, Paris France, Déc 1995. IFIP.

[GH92] A. Gravey and G. Hébuterne. Simultaneity in discrete-time single server queues with Bernouilli inputs. *Performance Evaluation North-Holland*, 14 :123–131, 1992.

[GM92] A. G. Greenberg and N. Madras. How fair is fair queuing ? *J. of the ACM*, 39, 3 :568–598, 1992.

[GP87] E. Gelenbe and G. Pujolle. *Introduction to Queueing Networks*. John Wiley & Sons Limited, 1987.

[Gre96] Albert G. Greenberg. Fair queueing architectures for high-speed networks. In *Proceedings of the Fourth International Workshop on Modeling, Analysis and*

Simulation of Computer and Telecommunication Systems, page 198, San Jose, California, feb 1996. IEEE Computer Society TCCA and TCS.

[GRI81] GRINSEC. *La commutation électronique*. EYROLLES, 1981.

[GRSS94] F. Guillemin, G. Rubino, B. Sericola, and A. Simonian. Transient characteristics of an $M/M/\infty$ system applied to statistical multiplexing on an ATM link. Publication interne 874, IRISA, october 1994.

[GVC97] Pawan Goyal, Harrick M. Vin, and Haichen Cheng. Start-time fair queuing : A scheduling algorithm for integrated services packet switching networks. *IEEE/ACM Transaction on Networking*, 5, 1997.

[Héb85] G. Hébuterne. *Écoulement du trafic dans les autocommutateurs*. Masson, Institut National de Télécommunications, 1985.

[Héb98] Gérard Hébuterne. Quality of service and related issues in broadband networks. In *Proceedings of ICT'98*, 1998.

[HYP98] S. Hai, K. L. Yeung, and L. Ping. Optimal queuing policy for input-buffered ATM switches with two-fifo queues per input port. In *Proceedings of the International Conference on Telecommunications*, Porto Carras Greece, June 1998. IFIP.

[I.396] ITU-T I.371. *Traffic control and congestion control in B-ISDN*. Genève, octobre 1996.

[Iko99] *http ://www.ikos.com*. Ikos, 1999.

[J.L80] J.Leroudier. *La simulation à événement discrets*. Édition hommes et techniques., 1980.

[Kel79] F. P. Kelly. *Reversibility and Stochastic Networks*. John Wiley and Sons, Chichester, 1979.

[KR90] R. M. Karp and V. Ramachandran. Parallel Algorithms for Shared-Memory Machines. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A : Algorithms and Complexity, chapter 17, pages 870–941. North Holland, 1990.

[Lav98] Dominique Lavenier. Speeding up genome computation with a systolic accelerator. *Society for Industrial and Applied Mathematics (SIAM News)*, 31-8, 1998.

[Len90] Lengauer. VLSI theory. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A : Algorithms and Complexity, chapter 16, pages 837–869. North Holland, 1990.

[Lis99] *http ://www.io.com/ guccione/HW_list.html*. 1999.

[LLM95] L. Lundheim, I.C. Legrand, and L. Moll. A programmable active memory implementation of a neural network for second level triggering in atlas. In *Proceedings of AIHEN'95*, 1995.

[LMV98] C. Labbé, S. Martin, and J-M. Vincent. A reconfigurable hardware tool for high speed network simulation. In *International Conference on Modelling Techniques and Tools for Performance Evaluation, Performance TOOLS'98*, 1998.

[LRV98] C. Labbé, F. Reblewski, and J-M. Vincent. Performance evaluation of high speed network protocols by emulation on a versatile architecture. *RAIRO, Systèmes à événements discrets stochastiques : théorie, application et outils.*, 1998.

[LS95] E. Lemoine and J. Sallantin. Un système reconfigurable dédier à la comparaison de séquence génétiques. *Rairo : technique et science informatiques*, 14-1, 1995.

[Mat98] M. Matsumoto. Simple cellular automata as pseudo-random m-sequence generators for built-in self-test. *ACM, Transactions On Modeling and Computer Simulation*, 8-1, 1998.

[MG99] *http ://www.mentorg.com*. Mentor Graphics, 1999.

[Mis87] Schwartz Mischa. *Telecomunication Networks. Protocols, Modeling and Analysis*. Addison-Wesley Publishing Company, November 1987.

[MN98] M. Matsumoto and T. Nishimura. Mersenne twister : A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM, Transactions On Modeling and Computer Simulation*, 8-1, 1998.

[Mol97] Laurent Moll. *Application des Mémoires Actives Programmables*. PhD thesis, École Polytechnique, 1997.

[MW95] Sidharta Mohanty and Philip A. Wilsey. Rapid system prototyping, system modeling, and analysis in a hardware-software environment. In *IEEE Rapid Systems Prototyping Workshop*, pages 154–160, 1995.

[NT96] Huy Nam NGUYEN and Michel THILL. Design verification based on hardware emulation. In *Seventh IEEE International Workshop on Rapid System Prototyping*, BULL S.A., 1996.

[PBF98] C. D. Pham, H. Brunst, and S. Fdida. Conservative simulation of load-balanced routing in a large ATM network model. In *Proceedings of the 12th Workshop on Parallel and Distributed Simulation (PADS-98)*, pages 142–153, Los Alamitos, May 26–29 1998. IEEE Computer Society.

[PBL97] Jean Pellaumail, Pierre Boyer, and Patrice Leguesdron. *Autoroutes ATM*. HERMES, 1997.

[PEF97] C. D. Pham, J. Essmeyer, and S. Fdida. Simulation of a routing algorithm using distributed simulation techniques. *Lecture Notes in Computer Science*, 1300 :1001–? ?, 1997.

[Pel92] Jean Pellaumail. *Graphes, Simulation, L-matrices, application aux files d'attente*. Hermes, 1992.

[Per93] Douglas L. Perry. *VHDL*. McGraw-Hill Series on Computer Engineering, 1993.

[Puj92] G. Pujolle. Commutateurs ATM : Classification et architecture. *Technique et Science Informatique*, 11, 1 :11–29, 1992.

[qui99] *http ://www.quickturn.com*. Quickturn, 1999.

[RG92] J. Roberts and F. Guillemin. Jitter in ATM networks and its impact on peak rate enforcement. *Performance Evaluation North-Holland*, 16, 1-3 :35–48, 1992.

[RKSC95] O. Rasmont, J. Koulischer, J. Schaumont, and R. Crappe. Testing and optimizing a scale reduction algorithm for a multi-screen video wall application on the meta-100 asic emulator. In *Sixth IEEE International Workshop on Rapid System Prototyping*, 1995.

[RLB96] S. Robert and J.-Y. Le Boudec. Can self-similar traffic be modeled by markovian processes ? *Lecture Notes in Computer Science*, 1044, 1996.

[RLB98] S. Robert and J.-Y. Le Boudec. Properties of a new class of models designed for self-similar traffic. In *Proceedings of the First Workshop on ATM Traffic Management*, Paris France, Decembre 1998. IFIP.

[Rol95] Pierre Rolin. *Réseaux haut débit*. réseaux et télécommunications. Hermes, Paris, 1995.

[Ros91] Sheldon M. Ross. *A Course in Simulation.* Mamillan Publishing Company, University of california, Berkeley, 1991.

[Sha97] Mark Shand. A case study of algorithm implementation in reconfigurable hardware and software. In *Proceedings of the 7th International Workshop, FPL'97, Lecture notes in Computer Science. Springer-Verlag.*, Digital Equipment Corporation, Palo Alto, 1997.

[SLC75] H. Daniel Schnurmann, Eric Lindbloom, and Robert G. Carpenter. The weighted random test-pattern generator. *IEE Transactions on computers*, 14-1, 1975.

[Ste94] William J. Steward. *Introduction to the Numerical Solution of Markov Chains.* Princeton University Press, 1994.

[Sti96] Dimitrios Stiliadis. *Traffic Scheduling in Packet-Switched Networks : Analysis, Design, and Implementation.* PhD thesis, University of California Santa Cruz, 1996.

[SV93] M. Shand and J. Vuillemin. Fast implementations of RSA cryptography. In *11th IEEE Symposium on COMPUTER ARITHMETIC*, 1993.

[SV97] D. Stiliadis and A. Varma. A reconfigurable hardware approach to network simulation. *ACM Transaction on Modeling and Computer Simulation*, 7, 1997.

[SV98] D. Stiliadis and A. Varma. Efficient fair queuing algorithms for packet-switched networks. *IEEE/ACM Transaction on Networking*, 6, 1998.

[Tan97] A. Tanenbaum. *Réseaux.* Prentice Hall Inter-Editions, 1997.

[Tez95] Shu Tezuka. *Uniform Random Number : Theory and practice.* Kluwer Academic Piblishers, IBM Japan, 1995.

[Tru95] Laurent Truffet. *Méthodes de Calcul de Bornes Stochastiques sur des Modèles de Systèmes et de Réseaux.* PhD thesis, Université Paris VI, 1995.

[VBR+96] J. Vuillemin, P. Bertin, D. Roncin, M. Shand, H. Touati, and P. Boucard. Programmable active memories : Reconfigurable systems come of age. *IEEE Transactions on VLSI Sytems*, 4-1, 1996.

[Wei98] D. Weil. *Architectures de Circuit pour la Commutation et la Gestion de Ressources en ATM.* PhD thesis, Institut National Polytechnique de Grenoble, 1998.

[Woo94] Michael E. Woodward. *Communication and Computer Networks, Modelling with discrete-time queues.* IEEE Computer Society Press, 1994.

[WV96] Jean Walrand and Pravin Varaiya. *High-Performance Communication Networks.* Morgan Kaufmann Publishers, Inc., San Francisco, 1 edition, 1996.

[YHP98] K. L. Yeung, S. Hai, and L. Ping. Thoughput analysis for input-beffered ATM switches with multiple fifo queues per input port. In *Proceedings of the International Conference on Telecommunications*, Porto Carras Greece, June 1998. IFIP.

[Zak85] Rodnay Zaks. *Du composant au système : introduction aux microprocesseurs.* Sybex, 1985.

Annexes

Annexe A

Comparaison entre NS et Sim-express

Cette annexe présente les résultats de simulations réalisées à l'aide du simulateur Network-Simulator sur un PC bi-processeur (pentium II à 333 Mhz). Les courbes présentées mesurent le temps nécessaire pour simuler 10^6 slots. Dans les différents cas présentés, sur l'émulateur on a équivalence entre slots et top horloge. L'émulateur ayant une fréquence de 1 Mhz le temps nécessaire à la simulation de 10^6 slots est toujours d'une seconde.

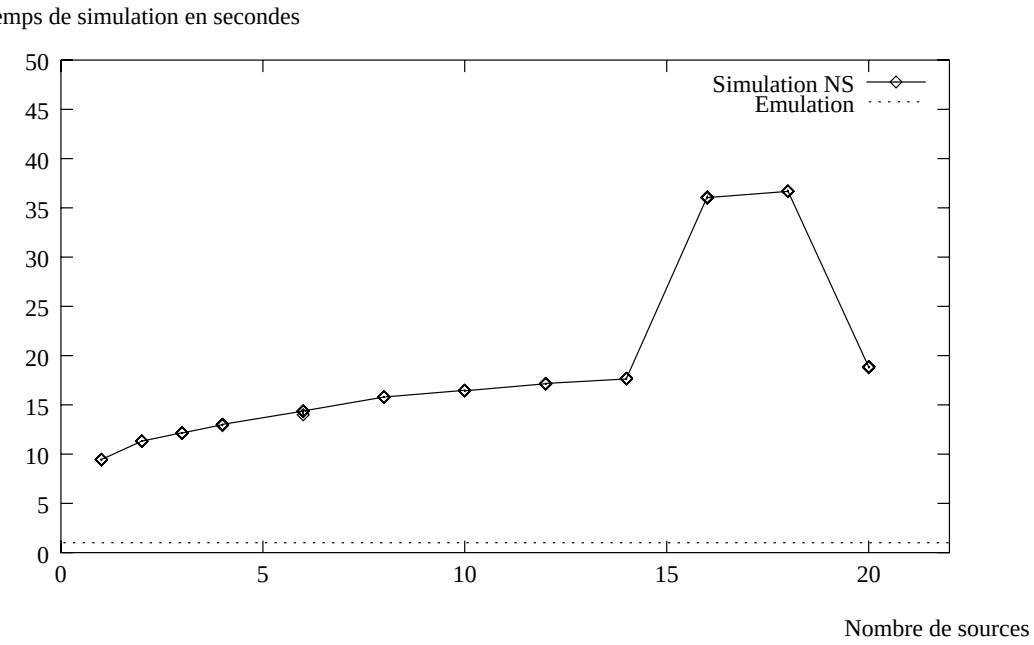

FIG. A.1: Temps de simulation d'une file d'attente FIFO avec NS et l'émulateur. La charge est de 0.8, le nombre de slots simulés est de 10^6. Le temps de simulation sur l'émulateur est constant de 1 seconde.

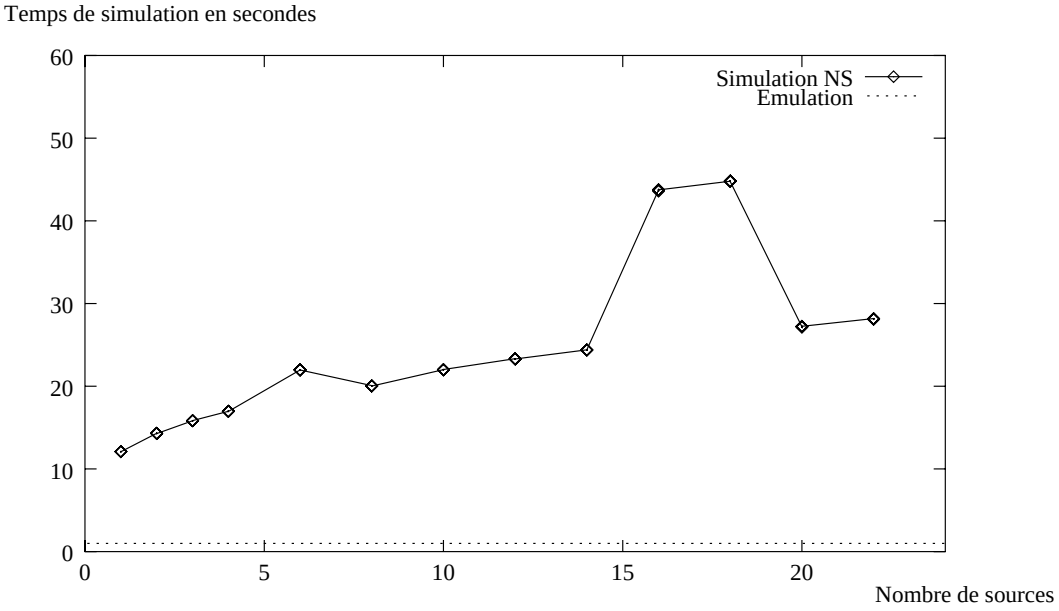

FIG. A.2: Temps de simulation d'une file d'attente de type Fair Queueing avec NS et avec l'émulateur. La charge est de 0.8 et le nombre de slots simulés est de 10^6. Le temps de simulation sur l'émulateur est constant de 1 seconde.

Une file d'attente unique

Les courbes (A.1) et (A.2) représentent le temps de simulation d'une file d'attente et de plusieurs sources. L'émulateur permet une accélération d'un facteur compris entre 10 et 50.

Commutateur multi-étages

Les courbes (A.3) et (A.4) représentent le temps de simulation de commutateurs multi-étages 4 vers 4 et 8 vers 8. L'émulateur permet une accélération d'un facteur compris entre 40 et 600.

Il faut noter que les commutateurs peuvent être implantés plusieurs fois dans la machine. Ainsi quatre commutateurs 8x8 peuvent fonctionner en parallèle. De cette manière, pour une charge de 0.8, l'évaluation de paramètres statistiques peut être réalisée 2000 fois plus vite qu'avec le simulateur NS.

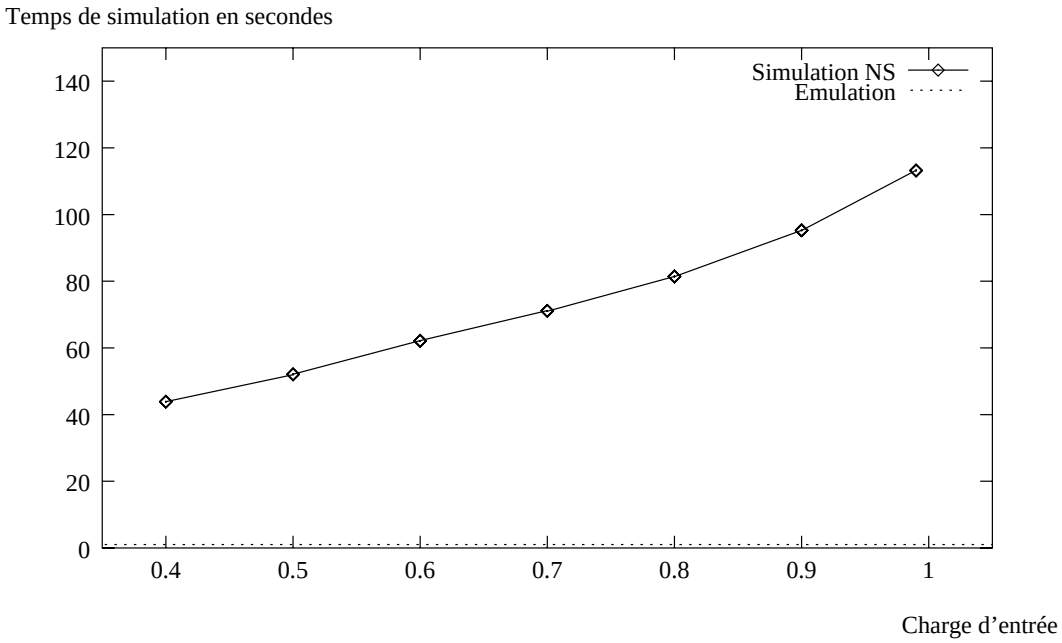

FIG. A.3: Temps de simulation d'un commutateur 4x4 à deux étages (8 files d'attentes). Le nombre de slots simulés est de 10^6 et le temps de simulation sur l'émulateur est constant de 1 seconde.

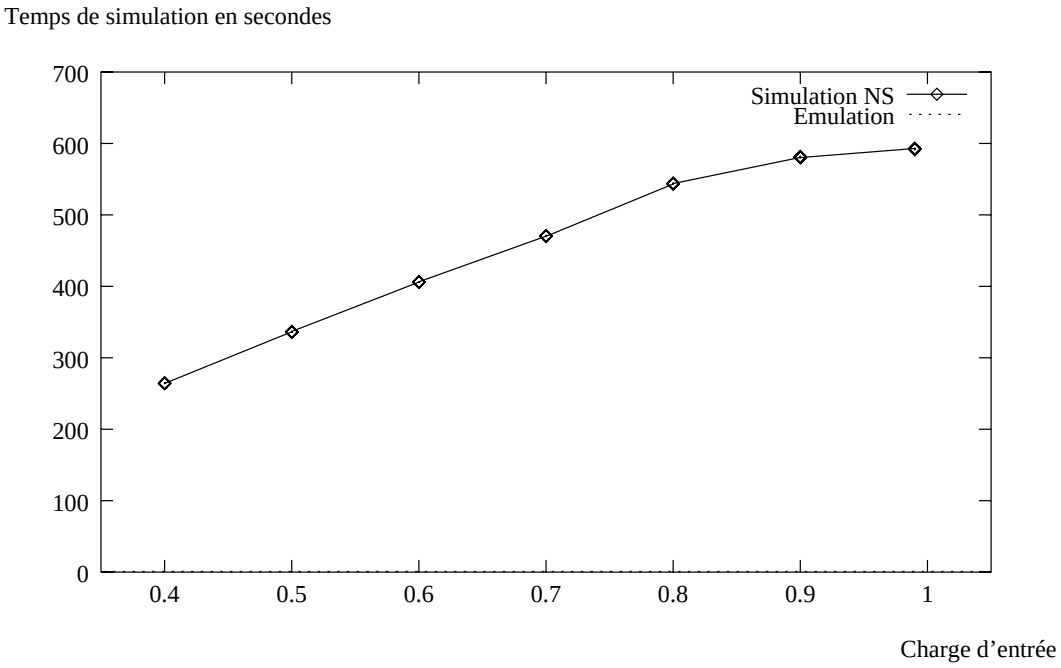

FIG. A.4: Temps de simulation d'un commutateur 8x8 à trois étages (24 files d'attentes). Le nombre de slots simulés est de 10^6 et le temps de simulation sur l'émulateur est constant de 1 seconde.

Annexe B

Exemples extrait de la bibliothèque de composants

Cette annexe présente quelques composants extrait de la bibliothèques. Les types qu'ils utilisent y sont aussi déclarés. Les principaux types et constantes utilisés pour le modèle de file dite "sans information" sont :

```
constant Pfin : natural := 16;
constant Valmax  : natural := 2**Pfin-1;
subtype vecteurPfin is std_ulogic_vector(Pfin-1 downto 0);
subtype vecteur32 is std_ulogic_vector(31 downto 0);

subtype proba is natural range 0 to Valmax;

-- Def de la puissance max du Gen et d'un type vecteur de bit.
constant Deg  : natural := 127;
subtype vecteurDeg is std_ulogic_vector(Deg-1 downto 0);

-- Def du type vectadd sur 7 bit.
constant Tad : natural := 7;
subtype vectadd is std_ulogic_vector(Tad-1 downto 0);


-- Nb de sources max.
constant nbsources : natural := 64;

-- Def du type Transfert : entre les diff elements du com.
subtype nb_cell is natural range 0 to nbsources;
type Transfert is array(nbsources-1 downto 0) of nb_cell;

subtype routage is std_ulogic_vector(2*nbcross-1 downto 0);
subtype capacite is natural range 0 to Valmax;
type vect_entiers is array(nbsources-1 downto 0) of integer;

constant ValmaxGeo   : positive := 16 ;
type petittab is array(ValmaxGeo downto 0) of integer;
-- Ce type est pour le composant AnaTraf !!
type Tableau is  array(1 downto 0) of petittab;
-- !! c'est type sont pour le compisant RegTraf :
```

```
--type Tableau is  array(nbsources-1 downto 0) of petittab;
--type tabcompt is array(nbsources-1 downto 0) of integer;
--type tabavant is array(nbsources-1 downto 0) of std_ulogic;
```

Ces types et constantes sont utilisé pour la définition et l'utilisation des composants de la bibliothèque. On trouve ci-dessous quelques exemples des composants présent dans la bibliothèque. Ceux présentés ici concernent uniquement le modèle sans information, dans le modèle avec information le type `nb_cell` change : il est remplacé par un tableau dont les éléments sont de type `cell`. Ce dernier type est un vecteur de bit comportant plusieurs champs dont le nombre et la taille sont a fixer par l'utilisateur.

```
-- Composant d'initialisation pour les generateurs aleatoires
component GenIni
    port( CK, init : in  std_ulogic  ;
          germe    : in  vecteur32   ;
          Vectmem  : out vecteur32   ;
           marche  : out std_ulogic  ;
              top  : out std_ulogic  ;
          mem_top  : out std_ulogic  ;
            initV  : out std_ulogic_vector(nbsources downto 0) );
end component;

-- Generateur aleatoire a base de Memoire
component GenMem
    port( CK, reset,top : in  std_ulogic   ;
              mem_top   : in  std_ulogic   ;
              m         : in  proba        ;
              germe     : in  vecteurPfin  ;
              res       : out nb_cell     );
end component;

-- Cross-bar pour les commutateurs multi-etages.
component
    generic(N : natural);
    port(CK               : in std_ulogic;
         entre_1, entre_2 : in nb_cell;
         tirage           : in std_ulogic_vector(1 downto 0);
         sort_1, sort_2   : out nb_cell);
end component;

-- Fifo de type arival first Dans le cas du modèle
-- avec information, le type nb_cell et un tableau de cell.
component fifoAF
    port(      init, CK : in std_ulogic;
                     k  : in capacite ;
            consomation : in capacite ;
               arrivees : in nb_cell;
     pertes, occupation : out integer ;
                   sort : out nb_cell );
end component;
```

Résumé :

Ce travail présente...

Mots clés : clé 1, clé 2

Title:

Modelisation of high speed network protocol with a versatil architecture.

Abstract:

This work presents...

Keywords: key 1, key 2