RESEAUX A HAUT DEBIT : COMMUTATEURS

Auteur : Thierry USO

Version : 1.4

Date : 10 Aout 2001

Table des matières

FIGURES

TABLEAUX

Chapitre 1

INTRODUCTION

Ce document traite des commutateurs dans les réseaux à haut débit. Le terme de commutateur est utilisé ici dans le sens de commutateurs de PDUs; la commutation de circuits n'est pas abordée.

Le chapitre 2 présente les concepts permettant de comprendre le fonctionnement d'un commutateur. Le chapitre 3 présente les principales architectures de commutateurs. Le chapitre 4 présente les aspects spécifiques de la commutation Ethernet, ATM et IP.

Un glossaire à la fin du document explicite les nombreux acronymes utilisés.

Chapitre 2

CONCEPTS

Il n'existe pas de définition précise de ce qu'est un commutateur. Un équipement est souvent désigné comme commutateur pour des raisons marketings (cf. le changement de nom du DECbridge 900MX en DECswitch 900EF). La définition la plus large appelle commutateur tout périphérique d'interconnexion.

Un commutateur est habituellement décomposé en 3 blocs fonctionnels distincts :

- les *N* ports d'entrée qui assurent la réception des PDUs
- l'interconnexion (switching fabric) qui assure le forwarding des PDUs
- les *N* ports de sortie qui assurent l'émission des PDUs

Bien évidemment, pour l'utilisateur la distinction entre ports d'entrée et de sortie disparait au profit de la notion de ports, chaque port assurant la la réception et l'émission des PDUs. Un port peut se raccorder en liaison point-à-point à :

- un port d'un autre commutateur
- un port d'un concentrateur/feeder
- une interface d'un ES

Le commutateur fonctionne de manière synchronisée. L'axe du temps est découpé en tranches appelées slots. Les slots sont généralement égaux. Durant un slot, le commutateur peut au mieux recevoir une PDU par port, prendre *N* décisions de forwarding et émettre une PDU par port. Plus le niveau de parallèlisation de ces 3 fonctions sera élevé, plus le commutateur s'éloignera d'une implémentation store-and-forward (PDU-forwarding) pour tendre vers une implémentation cut-through (bit-forwarding).

Lorsque plusieurs PDUs traitées durant le même slot ont le même port de sortie, il y a conflit en sortie. Seule une des PDUs est émise sur le port. Les autres PDUs sont bufferisées ou perdues.

BRADNER (1991) propose les 3 critères suivants pour mesurer la performance d'un commutateur :

- le débit
  Le débit se définit comme étant la vitesse maximale de forwarding sans perte de PDUs exprimée en PDU/s ou bit/s.
- le taux de perte
  Le taux de perte se définit comme étant le % de PDUs perdues pour une charge donnée.
- le temps de traversée (ou latence)
  En store-and-forward, le temps de traversée est la durée entre l'instant où le premier bit de la PDU est émis et l'instant où le dernier bit de la PDU est reçu. En cut-through, le temps de traversée est la durée entre l'instant où la fin du premier bit de la PDU est émise et l'instant où la fin du premier bit de la PDU est reçue.

A ces 3 critères, on peut également en ajouter d'autres tels que l'équité ou la faculté à conserver l'ordre des PDUs d'un même circuit virtuel (X25, Frame Relay ou ATM).

Chapitre 3

ARCHITECTURES DES COMMUTATEURS

On distingue 2 grands types d'architectures pour l'interconnexion :

- médium partagé
- division spatiale

Ces 2 types d'architectures ont leurs avantages et leurs inconvénients. Les architectures à division spatiale semblent néanmoins les plus prometteuses pour les réseaux à haut débit.

3.1 ARCHITECTURES A MEDIUM PARTAGE

3.1.1 ARCHITECTURE A MEMOIRE PARTAGEE

L'interconnexion est constituée d'une mémoire accédée par toutes les lignes en entrée et en sortie.

L'accès à la mémoire doit être suffisamment rapide pour accepter le trafic entrant et sortant. Par exemple, la mémoire d'un commutateur de 32 lignes à 150 Mbit/s doit supporter un trafic de 2*32*150 Mbit/s, soit 9.6 Gbit/s. Pour ce faire, la mémoire doit avoir une organisation parallèle (bit-slicing).

La taille de la mémoire nécessaire pour ne pas dépasser un taux de perte de PDUs dépend du nombre de lignes *N*, de la capacité de ces lignes, de la nature du trafic (sporadicité) mais aussi de la facon dont la mémoire est distribuée entre les files d'attente de chaque ligne de sortie.

2 politiques de files d'attente peuvent être utilisées :

- complete-partitioning

La mémoire est divisée en *N* sections, chacune d'elle étant associée à la file d'attente d'une ligne de sortie. Dans ce cas, une PDU est perdue lorsque la section est déja pleine.

- full-sharing
  L'ensemble de la mémoire est partagé par les files d'attente des lignes de sortie. Dans ce cas, une PDU est perdue lorsque la mémoire est déja pleine.

Pour un taux de perte donné, le full-sharing minimise la taille de la mémoire nécessaire alors que le complete-partitioning assure une meilleure équité entre les files d'attente.

Le commutateur ATM Prelude conçu par le CNET (DEVAULT M. et al., 1988) est un exemple d'architecture à mémoire partagée.

3.1.2 ARCHITECTURE A BUS

L'interconnexion est constituée d'un bus parallèle accèdé en TDMA par toutes les lignes en entrée et en sortie.

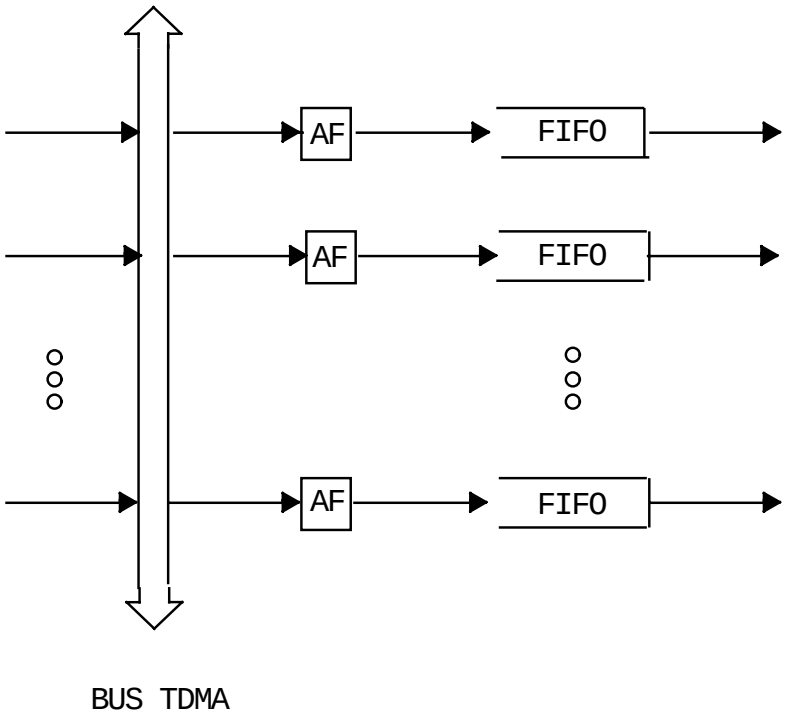
Chaque ligne en sortie est connectée au bus par une interface constituée d'un filtre d'adresse et d'une file d'attente. A chaque file d'attente est allouée une quantité de mémoire fixe (complete-partitioning). Le filtre d'adresse détermine si la PDU qu'il lit sur le bus doit être copiée dans la file d'attente qui lui est associée.

L'accés au bus doit être suffisamment rapide pour accepter le trafic entrant et sortant. Par exemple, le bus d'un commutateur de 32 lignes à 150 Mbits/s doit supporter un trafic de 32*150 Mbits/s, soit 4.8 Gbits/s.

La performance d'un commutateur à bus est très proche de celle d'un commutateur à mémoire partagée en complete-partitioning.

Le commutateur ATM Atom conçu par NEC (SUSUKI H. et al., 1989) est un exemple d'architecture à bus.

Figure 3–1: Commutateur à bus



3.2 ARCHITECTURES A DIVISION SPATIALE

Dans les architectures à division spatiale, le routage des PDUs s'effectue en établissant des chemins reliant les ports d'entrée aux ports de sortie, chacun d'eux ayant la capacité d'une ligne.

Ces architectures offrent 2 avantages par rapport aux architectures à médium partagé :

- il n'est pas nécessaire d'avoir des composants fonctionnant à des vitesses de N ou $2N$ fois la capacité d'une ligne
- le routage peut être décentralisé

Les architectures à division spatiale ont cependant des problèmes qui leur sont propres. Par exemple, certaines d'entre elles ne permettent pas toujours d'établir simultanément l'ensemble des chemins demandés. Ce phénomène est connu sous le nom de blocage interne. Les PDUs bloquées doivent être bufferisées soit à l'endroit du blocage, soit dans une file d'attente associée à leur port d'entrée. Le traitement du blocage interne a un impact important sur la performance et le coût du commutateur.

3.2.1 CROSSBAR

L'interconnexion est constituée d'une matrice carrée de N^2 portes. A chaque porte est associée une paire (i,j) où i est un port d'entrée et j un port de sortie. Les portes peuvent prendre 2 états appelés CROSS et BAR. les N^2 portes sont initialement dans l'état CROSS. L'établissement d'un chemin entre i et j s'effectue en passant la porte (i,j) dans l'état BAR (figure 3–2).

Le routage dans un Crossbar est décentralisé. La PDU donne son adresse de destination à chaque porte qu'elle traverse. En fonction de cette adresse, la porte passe ou non de l'état CROSS à l'état BAR. Le Crossbar est dit self-routing.

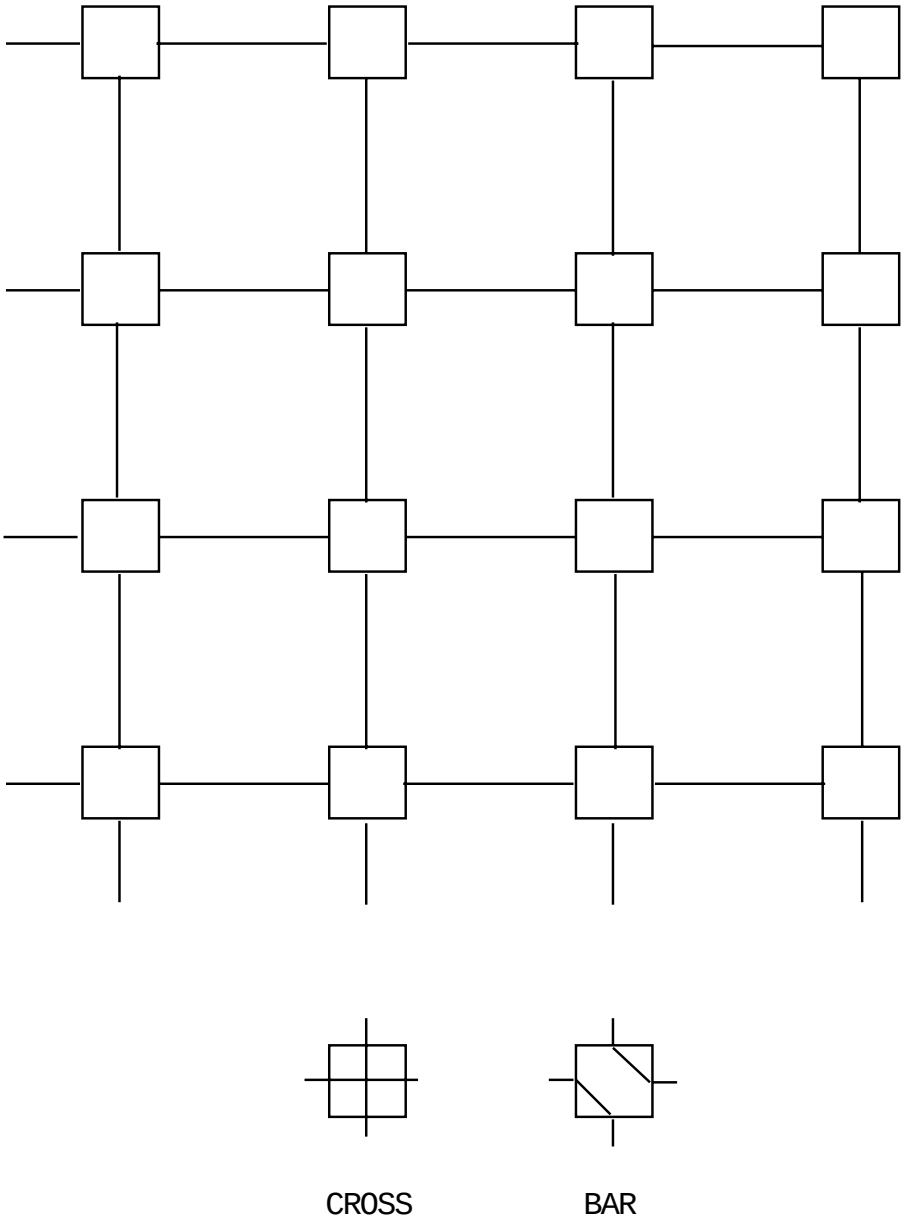
Le Crossbar ne manifeste du blocage interne que lorsqu'il y a des conflits en sortie (paires ayant la même valeur j). Pour pallier au blocage interne, des buffers peuvent être placés à 2 endroits :

- aux portes où le blocage apparait
  Cette situation est équivalente au complete-partitioning des architectures à médium partagé et conduit à une bonne performance. Cependant, les N^2 buffers nécessaires induisent un coût matériel élevé.

- aux ports d'entrée
  Si les buffers sont gérés comme des FIFO, il y a du HOL blocking ce qui conduit à une performance médiocre. A saturation et lorsque N tend vers l'infini, la performance est égale à 58.5% de la performance sans blocage interne. Pour améliorer la performance, un arbitre est en général utilisé. Celui-ci détecte les conflits en sortie, réorganise les buffers et envoie si néxessaire un signal blocant certaines PDUs dans leurs buffers.

Le commutateur GigaSwitch/ATM conçu par Digital (ANDERSON T.E. et al., 1993) est un exemple d'architecture Crossbar. Son fonctionnement est décrit au chapitre 4.2.

Figure 3–2: Crossbar



3.2.2 BANYAN

Le Crossbar a plusieurs inconvénients parmi lesquels :

- un nombre très élevé de portes
- un temps de traversée dépendant du chemin
- l'obligation pour chaque porte de connaitre l'adresse de destination complète

Pour pallier à ces inconvénients, les concepteurs de commutateurs ont imaginé une interconnexion constituée de portes binaires regroupées en plusieurs étages. Ces portes binaires pouvent prendre 2 états, CROSS et BAR. L'interconnexion est appelée Banyan lorsque les étages sont organisés de manière à obtenir un nombre minimal de portes égal à $(N/2)LogN$ (figure 3–3).

Le Banyan est self-routing comme le Crossbar mais la décision de routage au niveau de chaque porte s'effectue en testant un seul bit de l'adresse de destination de la PDU.

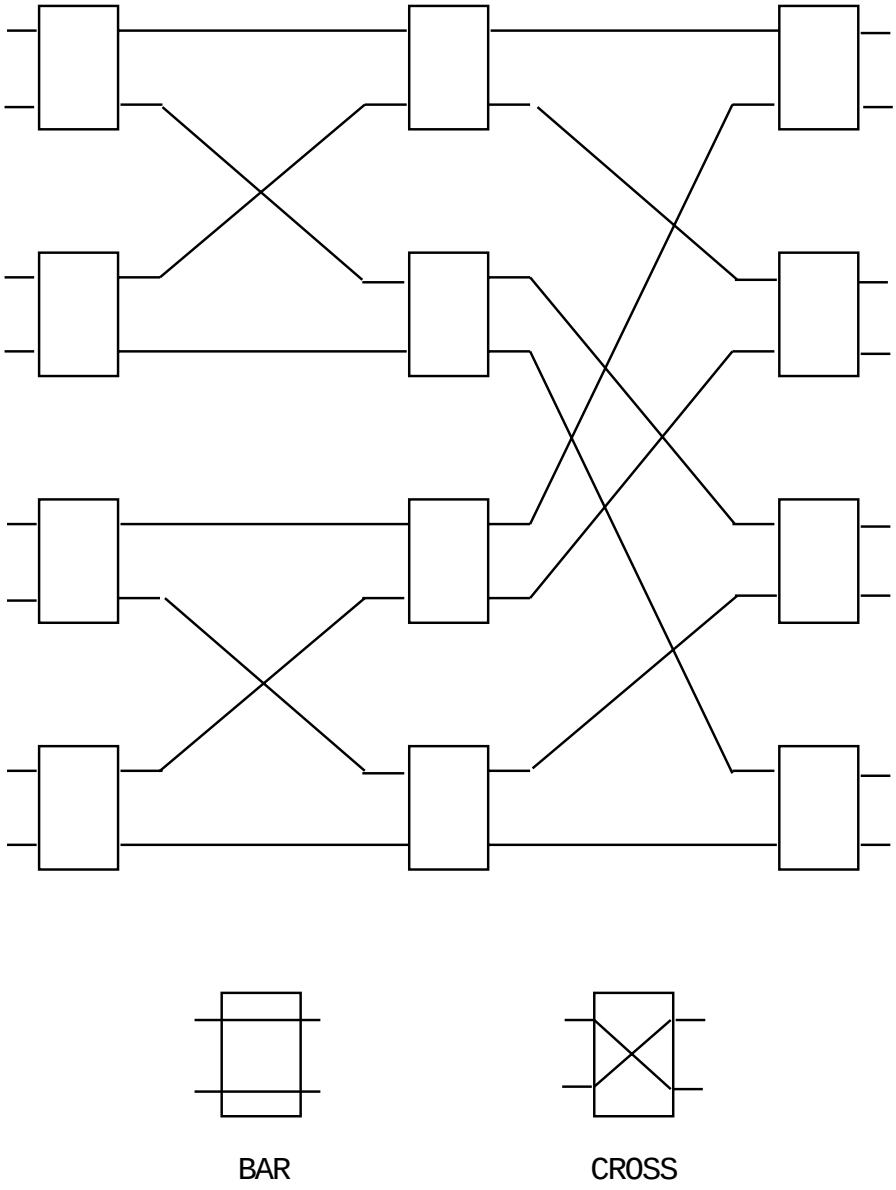
Le problème majeur du Banyan est le blocage interne. Celui-ci peut se produire même sans conflit de sortie. A saturation et lorsque N=1024, la performance est égale à 26% de la performance sans blocage interne.

Plusieurs variantes du Banyan ont été proposées pour lutter contre le blocage interne :

- Banyan avec buffers
  Les buffers sont placés à l'entrée des portes binaires. Le problème est alors de choisir une taille des buffers adéquate.
- Batcher-Banyan
  Les PDUs sont triées selon leur adresse de destination par l'algorithme de BATCHER puis envoyées dans le Banyan. Le trieur supprime la plupart des situations de blocage interne et ne laisse que celles liées à un conflit en sortie. Ces dernières peuvent alors être traitées comme dans le Crossbar.
- Banyans en tandem
  Les PDUs sont injectées dans un premier Banyan. Celles qui subissent un blocage interne sont routées vers un deuxième Banyan sauf une qui atteint son port de sortie. Cette opération se répète jusqu'à disparition des situations de blocage interne.
- Banyans en mirroir (réseau de BENES)
  Lorsque 2 Banyans sont placés en mirroir, l'interconnexion n'est plus self-routing car il existe plusieurs chemins possibles pour une PDU. Cependant, il existe toujours une combinaison de chemins sans blocage interne.

Le commutateur ATM Sunshine concu par Bell Communications Research (GIACOPELLI J.N. et al., 1991) est un exemple d'architecture Banyan.

Figure 3–3: Banyan 8x8



Chapitre 4

TECHNOLOGIES DES COMMUTATEURS

4.1 ETHERNET

La commutation Ethernet est une solution performante et économique d'évolution vers le haut débit. Le commutateur Ethernet peut être utilisé comme :

- hub
  Dans ce cas, une station Ethernet unique est raccordée à chacun des ports du commutateur.
- pont
  Dans ce cas, un segment Ethernet est raccordé à chacun des ports du commutateur.

Plus de 50 commutateurs Ethernet existent sur le marché à ce jour. Comme l'illustre le tableau 4–1, ces commutateurs reposent sur des architectures et des technologies très diverses.

Tableau 4–1: Quelques commutateurs Ethernet

Modèle	Fournisseur	Architecture	Technologie
EtherSwitch	Kalpana	Crossbar	Cut-through (hard)
LanSwitch	Lannet	Bus + ASIC	Store & forward (hard)
PowerHub	Alantec	Mémoire partagée	Store & Forward (soft)
ESX-820	Cabletron	Bus	Store & Forward (soft)

Une étude effectuée par MANDEVILLE (1994) montre que certaines architectures et technologies sont plus adaptées à un usage de hub et d'autres à un usage de pont.

Les collisions sont inexistantes dans un hub. Par conséquent, le commutateur peut se passer de filtrer les trames erronées. Le choix du commutateur s'effectue alors en se basant sur la latence et le taux de perte. Dans ces conditions, les technologies cut-through ou store-and-forward hardware s'imposent (tableau 4–2).

Lorsque le commutateur est utilisé comme pont, le filtrage des trames erronées est indispensable pour éviter que les collisions ne consomment la bande passante du commutateur. De plus, le raccordement de commutateurs entre eux n'est possible que s'ils implémentent l'algorithme du spanning tree (IEEE 802.1d). Dans ces conditions, les technologies store-and-forward software s'imposent malgré leur latence supérieure (tableau 4–2).

Tableau 4–2: Performance de quelques commutateurs Ethernet

	EtherSwitch	**LanSwitch**	**PowerHub**	**ESX-820**
Latence	40 us	52 us	90 us	130 us
Taux de perte	0%	0%	1%	25%
filtrage des erreurs	non	partiel	oui	oui

Le choix d'un commutateur de conception récente repose souvent sur la présence de certaines fonctionnalités telles que :

- le choix cut-through vs store-and-forward
- l'autonégotiation 10/100 Mbits/s et half/full duplex
- le support du VLAN IEEE 802.1q
- le groupage de plusieurs ports physiques en un port logique
- la modularité par ajoût optionnel de cartes WAN ou GigaEthernet
- le mirroring de ports

Le choix cut-through vs store-and-forward peut être fait :

- manuellement
  Par exemple, l'ingénieur réseau envoie un set-request SNMP pour choisir le fonctionnement cut-through ou store-and-forward.
- automatiquement
  Par exemple, le commutateur détecte le dépassement d'un seuil de collisions ou un nombre élevé de stations sur un port et bascule de lui-même en fonctionnement store-and-forward.

L'autonégotiation de la capacité (10 ou 100 Mbits/s) et du mode de fonctionnement (half ou full-duplex) bien que définie par l'IEEE pose encore parfois des problèmes d'interopérabilité entre commutateurs et cartes Ethernet. Par conséquent, il est intéressant de pouvoir également fixer ces caractéristiques manuellement.

le groupage de plusieurs ports physiques en un port logique (appelé EtherChannel group chez Cisco) permet d'augmenter le débit entre 2 commutateurs de même marque tout en offrant une fonction de fail-over très efficace. En effet, lorsqu'un port physique participant à un port logique n'est plus opérationnel, son traffic bascule vers les autres ports physiques participant au même port logique et cela sans reconstruction du spanning tree IEEE 802.1d.

Le mirroring de ports consiste à duppliquer le traffic entrant et sortant d'un port vers un autre port défini comme port mirroir. Le mirroring facilite grandement l'analyse du réseau à l'aide d'outils de type sniffer.

4.2 ATM

4.2.1 ARCHITECTURE DU GIGASWITCH/ATM

La problématique de la commutation ATM est illustrée par l'étude du GigaSwitch/ATM de Digital (ANDERSON T.E. et al., 1993).

Le GigaSwitch/ATM est un commutateur ATM de 34 ports maximum qui vise le marché des réseaux d'entreprise. Il est constitué d'un chassis sur lequel peuvent se connecter plusieurs types de cartes. Certaines de ces cartes sont obligatoires (Switch Control Processor, Crossbar) et d'autres optionnelles (ports ATM, alimentation redondante...).

Digital a choisi une architecture Crossbar car elle offre le meilleur compromis coût/performance pour des commutateurs de la taille du GigaSwitch/ATM.

L'intelligence du commutateur est partagée entre les ports ATM et le Switch Control Processor (SCP). Les ports ATM sont implémentés sous forme d'ASIC. Le SCP est constitué d'un processeur R3000, de 2 caches de 64 Ko, d'un write buffer, de 16 Mo de DRAM, de 2 Mo de mémoire Flash et d'un composant appelé XAC.

Les opérations qui ne peuvent pas être exécutées au niveau de chacun des ports sont exécutées par le SCP. Par exemple :

- le traitement des protocoles de couches hautes (IP, ARP, SNMP...)
- le multicasting de cellules ATM

La DRAM contient les bases de données d'adresses ainsi que des buffers de cellules. La mémoire Flash contient le code exécuté au boot du commutateur. Le XAC assure un bon fonctionnement à forte charge et effectue le multicasting.

Dans sa version actuellement commercialisée, Le GigaSwitch/ATM supporte le Classical IP et le LAN emulation. Les ports ont une capacité de 155 Mbit/s.

4.2.2 TRAITEMENT DU HOL BLOCKING

Digital a choisi de placer les buffers au niveau des ports d'entrée.

Si les buffers étaient gérés comme de simples FIFOs, le phénomène de HOL blocking qui en résulterait dégraderait notablement la performance (voir paragraphe 3). Par conséquent, Digital optimise l'ordonnancement des cellules dans les buffers à l'aide d'un algorithme appelé Parallel Iterative Matching.

Cet algorithme se décompose en 3 étapes :

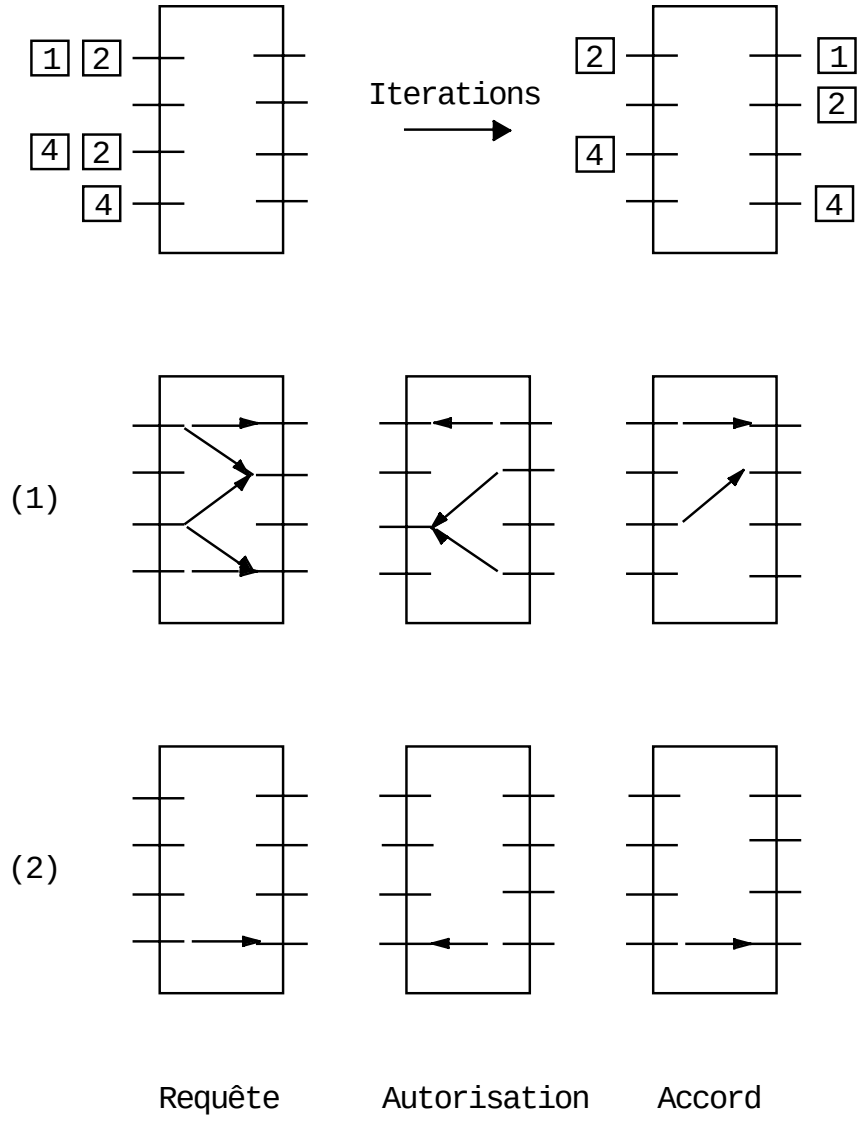
- requête
  Chaque port d'entrée envoie une requête à chaque port de sortie pour lequel il a une cellule dans un buffer.
- autorisation
  Si un port de sortie pas encore élu reçoit plusieurs requêtes, il en autorise une au hasard et avertit chacun des ports d'entrée de son choix.
- accord
  Si un port d'entrée reçoit plusieurs autorisations, il en choisit une et avertit le port de sortie correspondant.

Chacune de ces étapes s'exécute indépendament et en parallèle pour chaque paire de ports (entrée, sortie). Il n'y a donc pas de contrôle centralisé. La première itération de l'algorithme permet de découvrir plusieurs paires sans conflit de sortie. Les itérations suivantes mettant en jeu les ports d'entrée et de sortie non encore associés permettent d'en découvrir d'autres (figure 4–1).

Après 4 itérations, la performance obtenue est proche de celle des architectures utilisant des buffers au niveau des ports de sortie. Les itérations suivantes n'apportent qu'un gain négligeable.

Le Parallel Iterative Matching conserve l'ordre des cellules d'un VC mais n'est pas totalement équitable. Pour assurer une équité parfaite entre VCs et offrir les services synchrone et isochrone, Digital a conçu également un algorithme appelé Statistical Matching qui est une généralisation du Parallel Iterative Matching.

Figure 4–1: Parallel Iterative Matching



4.2.3 CONTROLE DE FLUX

Pour concurrencer les technologies à médium partagé, le LAN ATM doit permettre aux VCs d'avoir un débit crête égal à la capacité de la liaison de données. Cependant, en l'absence de contrôle de flux, les trafics combinés de plusieurs VCs peuvent dépasser momentanément la capacité de la liaison de données entrainant la pertes de cellules. Par conséquent, il est nécessaire d'effectuer du contrôle de flux.

Lorsque Digital a commencé à commercialiser le GigaSwitch/ATM, le contrôle de flux était laissé à l'initiative de chaque constructeur. Par conséquent, la 1ère génération de cartes de ports ATM implémente un contrôle de flux propriétaire. L'ATM forum ayant depuis peu standardisé le contrôle de flux, Digital commercialise désormais une 2ème génération de cartes de ports ATM conforme au standard.

Le contrôle de flux propriétaire est de type credit-based (par opposition à rate-based) et link-by-link (par opposition à end-to-end). Il utilise un algorithme appelé Flow Master que Digital a soumis à l'ATM Forum mais qui a été refusé par celui-ci.

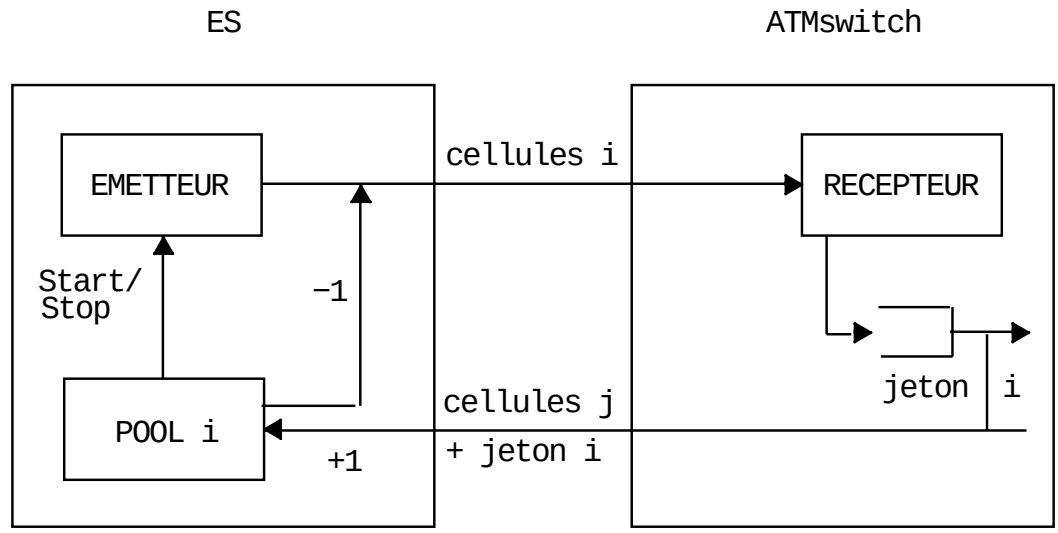
L'algorithme du Flow Master est le suivant :

- l'émetteur (ES ou commutateur) gère un pool de jetons pour chaque VC
- lorsque l'émetteur envoie une cellule sur un VC, le pool correspondant est décrémenté de 1
- lorsque le récepteur forward une cellule (commutateur) ou la passe aux couches supérieures (ES) libérant ainsi un buffer, il envoie un jeton à l'émetteur qui incrémente le pool correspondant de 1
- lorsqu'un pool est vide, l'émetteur s'interdit d'envoyer une cellule sur le VC correspondant

Le récepteur utilise les champs GFC, VPI et VCI de la cellule pour envoyer un jeton. Ces champs sont divisés en 2 zones de 14 bits chacune. La 1ère zone contient le VCI de destination et la 2ème le VCI à créditer d'un jeton.

L'émetteur et le récepteur négotient l'utilisation ou non du Flow Master en s'échangeant 2 trames SNMP. L'algorithme lui-même est implémenté de manière matérielle.

Figure 4–2: Algorithme du Flow Master



4.3 IP

4.3.1 ROUTEUR HAUT DEBIT

Jusqu'à récemment, un routeur était conçu de la manière décrite par la figure 4–3, à savoir :

- Intelligence centralisée dans un processeur généraliste
- Information volatile centralisée dans de la mémoire RAM
- Information permanente centralisée dans de la mémoire flash
- Interconnexion par médium partagé

Dans ce cas, l'essentiel des opérations du routeur est effectué de manière logicielle sous le contrôle de processeur. La RAM contient l'information permanente de la mémoire flash (système d'exploitation, configuration) chargée lors du boot du routeur, la table de routage et les buffers d'émission/réception.

Ce type de routeur est un goulot d'étranglement dans un réseau à haut débit. Au mieux, il commute un million de pps alors que l'objectif à court terme est d'atteindre les 100 millions de pps.

Les fabricants de routeurs haut débit proposent des produits dont la conception est radicalement différente de celle décrite ci-dessus. Ces produits peuvent être classés grossièrement en 2 grandes catégories selon l'importance accordée respectivement au hardware et au software.

La figure 4–4 schématise un routeur haut débit privilégiant le hardware. Dans cette catégorie de routeur haut débit, l'interconnexion par division spatiale est la plus fréquente. De plus, les interfaces d'E/S sont dotées (i) d'intelligence sous la forme d'ASICs et (ii) de caches contenant les routes des destinations les plus récentes. Les ASICs se chargent du traitement hardware de l'entête des paquets IP. Les caches ont pour rôle de limiter les accès à la mémoire lors des opérations d'address lookup.

Bien qu'ayant des performances remarquables, les routeurs haut débit privilégiant le hardware ont l'inconvénient majeur d'être peu réactifs vis à vis de l'évolution des normes. En effet, toute modification de l'algorithme de traitement de l'entête du paquet IP nécessite le développement d'interfaces d'E/S avec de nouveaux ASICs. Ceci se traduit par un coût et des délais non négligeables (le cycle de développement d'un ASIC est de 12 à 18 mois).

Actuellement, la tendance est plutôt à la conception de routeurs haut débit privilégiant le software (BUX W. et al., 2001). Cette catégorie de routeurs à l'avantage d'être réactive vis à vis de l'évolution des normes puisqu'une mise à jour logicielle suffit et cela sans sacrifier les performances. Pour ce faire, les concepteurs utilisent non plus un processeur centralisé généraliste mais un ou plusieurs processeurs centralisés spécialisés que l'on appelle network processors (NP).

La figure 4–5 schématise un prototype IBM de routeur haut débit prvilégiant le software. Ce prototype fonctionne de la manière suivante :

1. bitstream processor en réception
   Un bitstream processor prend en charge le paquet IP en provenance de l'interface d'E/S. Il extrait du paquet IP les informations nécessaires à son traitment (adresses, ToS...). Il passe ces informations et le paquet IP au processor complex.
2. processor complex
   Le processor complex prend en charge les informations et le paquet IP en provenance du bitstream processor. Il passe le paquet IP au scheduler qui le copie dans un buffer. Il traite les informations en parallèle en faisant appel à des co-processeurs pour certaines opérations (calcul de checksum, address lookup...). Une fois les résultats obtenus, il indique au scheduler comment émettre le paquet IP.
3. scheduler
   Grâce aux informations du processeur complex, le scheduler copie le paquet IP du buffer vers le bitstream processor d'émission via le bitstream processor de l'interconnexion.

Figure 4–3: Architecture de routeur traditionnel

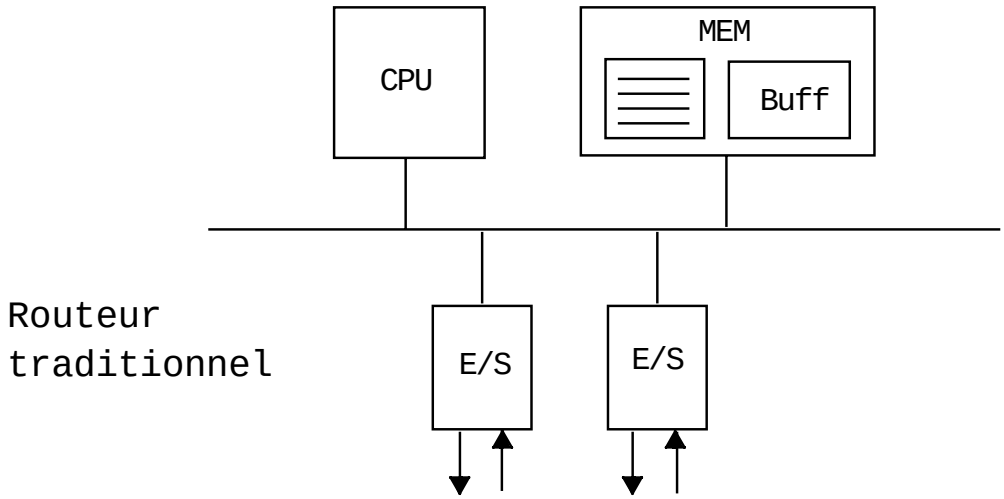


Figure 4–4: Architecture de routeur hardware

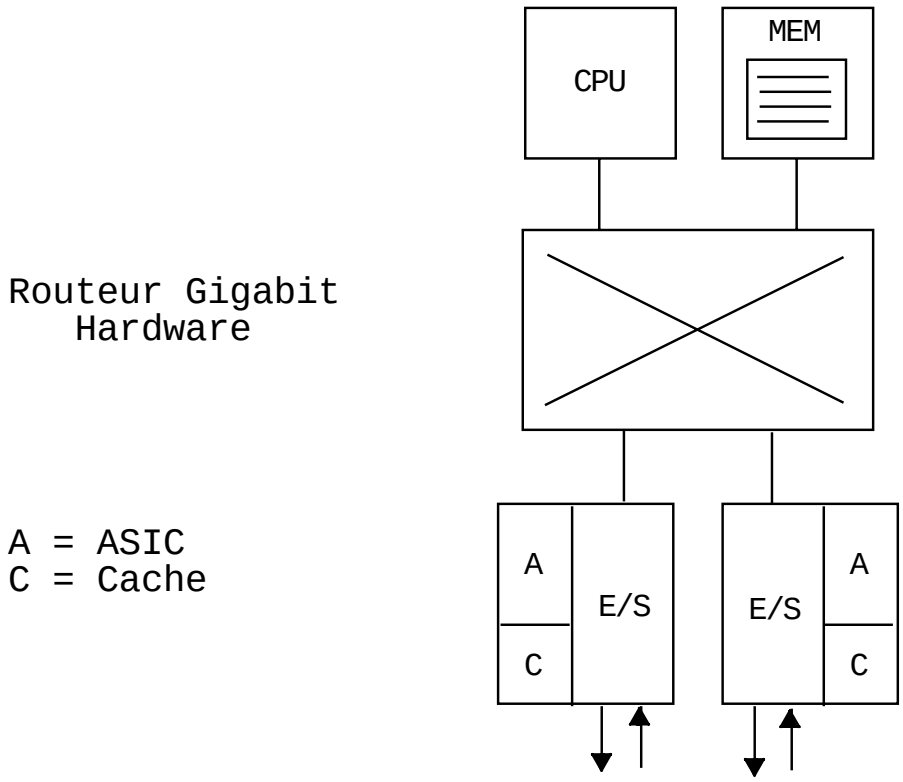
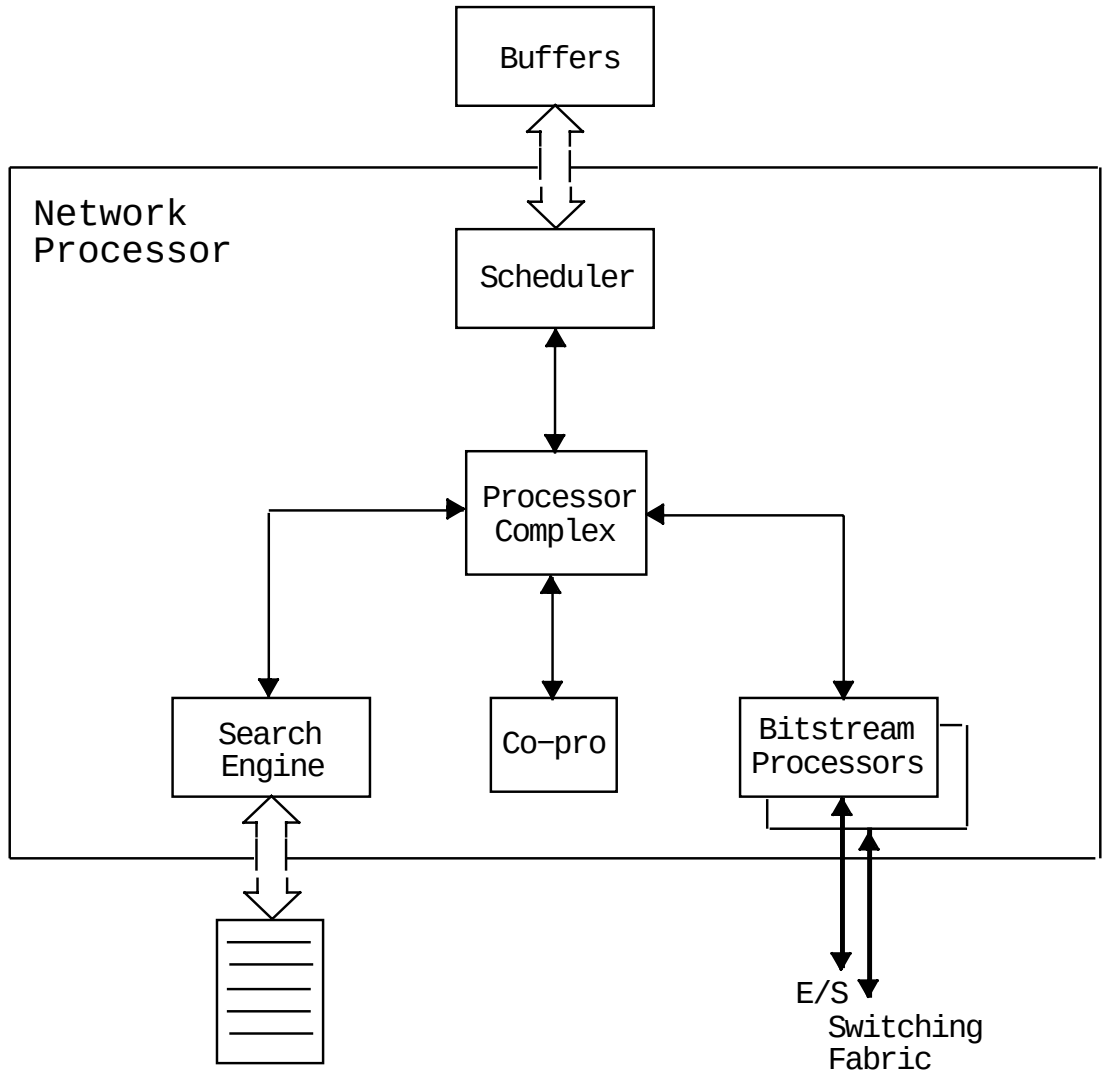


Figure 4–5: Architecture de routeur software



4.3.2 ADDRESS LOOKUP

L'address lookup consiste à consulter la table de routage pour trouver le port d'émission correspondant à l'adresse de destination du paquet IP. Un préfixe de l'adresse de destination dont la longueur n'est pas initialement connue (supernet, net, subnet...) sert de clé de recherche.

Parmi les opérations effectuées par un routeur IP, l'address lookup est la plus difficile à implémenter de manière performante. Un algorithme d'address lookup performant se doit d'être rapide tout en assurant une mise à jour de la table de routage rapide, une occupation mémoire faible et une bonne scalabilité. Le tableau 4–3 compare la complexité de différents algorithmes d'address lookup, N et W étant respectivement le nombre d'enregistrements de la table de routage et la longueur du préfixe.

Tableau 4–3: Complexité de différents algorithmes d'address lookup

	Recherche	**Mise à jour**	**Mémoire**
Force brute	O(N)	1	O(NW)
Arbre binaire simple	O(W)	O(W)	O(NW)
Arbre binaire compressé	O(W)	O(W)	O(N)
Arbre k-bit	O(W/k)	$O(W/K+2^k)$	$O(2^kNW/k)$
Hashing b.s.p.l.	O(logW)	O(NlogW)	O(logW)

L'algorithme d'address lookup le plus simple est appelé "force brute". Dans cet algorithme, la table de routage a la forme d'une liste de N enregistrements non triés. Chaque enregistrement commence par un préfixe d'adresse IP. La recherche consiste en N fois la comparaison de l'adresse de destination du paquet IP avec le préfixe de chaque enregistrement. L'enregistrement retenu est celui ayant le plus long préfixe de l'adresse de destination du paquet IP. La performance de cet algorithme étant particulièrement médiocre (recherche en N accès mémoire), les concepteurs de routeurs IP lui préfèrent des algorithmes où la table de routage a la forme d'un arbre binaire (simple, compressé, multibit...) ou de tables de hashing (RUIZ-SANCHEZ M.A. et al., 2001).

La figure 4–6 représente une table de routage sous la forme d'un arbre binaire simple. La recherche s'effectue par progression dans l'arbre en partant de la racine jusqu'à la feuille. Elle prend au maximum 32 accès mémoire. La recherche est accélérée et l'occupation mémoire diminuée si l'arbre binaire est compressé dans les portions de branche vides ou si l'arbre binaire est constitué de noeuds multibits.

La table de routage peut aussi être représentée sous la forme de 32 sous-tables de hashing, une par longueur de préfixe W. A chacune de ces sous-tables est associée une fonction de hashing Hw telle que Hw(préfixe) est un pointeur sur l'enregistrement correspondant. La recherche obéit alors a l'algoritme suivant :

```
W=32;
loop
  if Hw(préfixe de longueur W du paquet) -> enregistrement
    exit;
  else
    W=W-1;
endloop;
```

Comme dans le cas de l'arbre binaire simple, la recherche prend au maximum 32 accès mémoire. Cependant, une modification mineure des sous-tables conduit à un algorithme plus performant (binary search by prefix length).

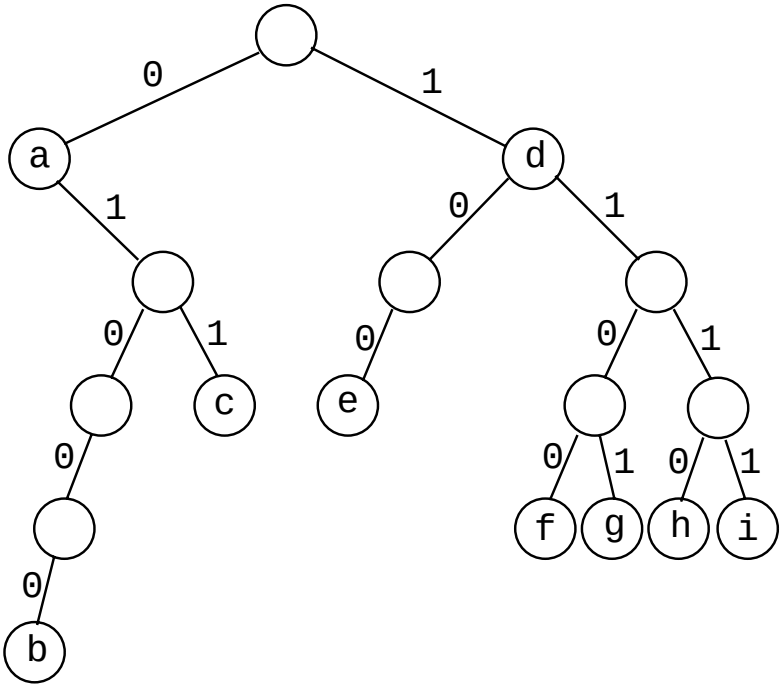
Enfin, récemment est apparue une alternative à ces différents algorithmes implémentés de manière logicielle. Cette alternative consiste à stocker la table de routage dans de la mémoire d'un type particulier appelé T-CAM. Dans cette mémoire, les données sont stockée sous la forme de paires {valeur, mask} de W bits. Par exemple, le préfixe d'adresse 10* est stocké dans la paire de 32 bits {10 suivi de 30x0, 11 suivi de 30x0}. La recherche s'effectue alors en comparant en parallèle l'adresse de destination du paquet IP avec toutes les paires et la réponse est obtenue en un accès mémoire inférieur à 10 ns. La mémoire T-CAM est actuellement peu utilisée dans les routeurs nécessitant une mémoire supérieure à 2 Mo pour des raisons à la fois techniques et économiques :

- densité de 11-15 transistors/bit (4-6 transistors/bit pour la SRAM)
- dissipation de chaleur de 7 Watt (2 Watt pour la SRAM)
- prix 15 fois supérieur à celui de la SRAM

Figure 4–6: Address lookup par arbre binaire simple

Préfixes

a = 0*
b = 01000*
c = 011*
d = 1*
e = 100*
f = 1100*
g = 1101*
h = 1110*
i = 1111*



Annexe A

BIBLIOGRAPHIE

ANDERSON T.E. et al. (1993) Digital Systems Research Center
High speed switch scheduling for local area networks

BRADNER S. (1991) IETF RFC 1242
Benchmarking terminology for network interconnection devices

BUX W. et al. (2001) IEEE Communications Magazine 39(1) pp. 70-77
Technologies and building blocks for fast packet forwarding

DEVAULT M. et al. (1988) JSAC 6(9) pp. 1528-1537
The Prelude ATD experiment: assessments and future prospects

GIACOPELLI J.N. et al. (1991) JSAC 9(8) pp. 1289-1298
Sunshine: A high-performance self-routing broadband packet switch architecture

JAIN R. et ROUTHIER S.A. (1986) JSAC 4(6) pp.1162-1165
Packet trains: measurements and a new model for computer network traffic

MANDEVILLE B. (1994) Réseaux & Telecoms 73 pp. 1-10
Switches Ethernet

PARTRIDGE G. (1994) Ed. Addison-Wesley
Gigabit Networking

RUIZ-SANCHEZ M.A. et al. (2001) IEEE Networks 15(2) pp. 8-23
Survey and taxonomy of IP address lookup algorithms

SUSUKI H. et al. (1989) Proc. Int. Conf. on Communications pp. 4.1.1-4.1.5
Output-buffer switch architecture for ATM

TOBAGI F.A. (1990) Proceedings of the IEEE 78(1) pp. 133-167
Fast packet switch architectures for broadband integrated services digital networks

WALSH R.J. et OZVEREN C.M. (1995) IEEE Networks 9(1) pp. 36-43
The GigaSwitch control processor

ZEGURA E.W. (1993) IEEE Communications Magazine pp. 28-37
Architectures for ATM switching systems

Glossaire

ASIC Application Specific Integrated Circuits

ATM Asynchronous Transfert Mode

GFC Generic Flow Control

LAN Local Area Network

MAC Medium Access Control

NP Network Processor

PDU Protocol Data Unit

SRAM Static Random Access Memory

T-CAM Ternary-Content Addressable Memory

VC Virtual Channel

VCI Virtual Channel Identifier

VPI Virtual Path Identifier

VLAN Virtual Local Area Network

WAN Wide Area Network