

Réseau II

Jean-Philippe Collette

27 janvier 2012

Table des matières

1 Routing multicast	2
1.1 Routing en broadcast	2
1.1.1 Broadcast sur un spanning tree	2
1.2 Routing multicast	3
1.2.1 Adressage	3
1.2.2 Arbre de plus court chemin	4
1.2.3 Difficultés en wide-area	4
1.3 Multicast en local area	4
1.3.1 IGMP - Internet Group Management Protocol	4
1.3.2 Multicast avec le broadcast LAN	4
1.4 Multicast en wide area	5
1.4.1 Source-based tree	5
1.4.2 Center-based	7
1.5 Exemples de protocoles multicast	8
1.5.1 DVMRP	8
1.5.2 PIM - Protocol Independant Multicast	8
2 Réseaux ATM et MPLS	13
2.1 ATM - Asynchronous Transfer Mode	13
2.1.1 ATM Adaptation Layer - AAL	14
2.1.2 Circuits virtuels	14
2.1.3 Couche ATM	14
2.1.4 Couche physique	15
2.2 IP sur ATM	16
2.2.1 Topologies	16
2.3 MPLS - MultiProtocol Label Switching	18
2.3.1 FEC - Forwarding Equivalence Class	19
2.3.2 Label switching	20
3 Réseaux mobile et sans-fil	22
3.1 Introduction	22
3.1.1 Problème d'un terminal caché	24
3.1.2 CDMA - Code division multiple access	24
3.2 Réseaux locaux sans-fil et IEEE 802.11	25
3.2.1 Normes sans-fil	25
3.2.2 Architecture LAN	25
3.2.3 CSMA/CA	26
3.2.4 Adressage	28
3.2.5 802.15 : Personal Area Network - PAN	30
3.2.6 802.16 : WiMAX	30
3.3 Cellular Internet Access	30
3.3.1 Standards	31
3.3.2 2G	31
3.3.3 2.5G	32
3.3.4 3G	32
3.4 Adressage et routage vers des utilisateurs mobiles	33
3.4.1 Mobilité par les routeurs	33
3.4.2 Mobilité par les systèmes d'extrémité	33
3.5 Mobile IP	35

3.5.1	Découverte d'agent	35
3.5.2	Enregistrement avec l'home agent	37
3.5.3	Routage indirect	37
3.6	Mobilité dans les réseaux cellulaires	37
3.6.1	Gestion d'un handoff	38
3.7	Impact sur les protocoles de plus haut niveau	39
4	Applications et transport multimédia	40
4.1	Applications réseau multimédia	40
4.1.1	Streaming Stored Multimedia	40
4.1.2	Streaming Live Multimedia	40
4.1.3	Real-Time Interactive Multimedia	41
4.1.4	Audio digital et compression	41
4.1.5	Compression vidéo	42
4.1.6	L'exemple du JPEG	43
4.1.7	MPEG	44
4.1.8	Streaming audio et vidéo stocké	45
4.1.9	RTSP - Real-Time Streaming Protocol	47
4.2	Retirer le meilleur du service best effort	48
4.2.1	Utilisation d'un délai de playout	48
4.2.2	Récupération de perte de paquet	50
4.2.3	CDN - Content distribution networks	52
4.3	Protocoles pour des applications multimédia temps réel	53
4.3.1	RTP - Real-time Protocol	53
4.3.2	RTCP - Real Time Control Protocol	54
4.3.3	SIP - Session Initiation Protocol	54
5	Architectures et mécanismes QoS	57
5.1	Classes de service	57
5.1.1	Principes	57
5.1.2	Mécanismes de scheduling	57
5.1.3	Mécanismes de shaping	63
5.1.4	Mécanismes de policing	65
5.1.5	Stratégies de jet de paquets	66
5.1.6	IETF Differentiated Services	69
5.1.7	Les services différenciés et MPLS	71
5.2	Garanties de QoS	71
5.2.1	Principes	71
5.2.2	Garanties de bande passante	72
5.2.3	Garanties de délai	72
5.2.4	Réservation de ressources - RSVP signaling	73
5.2.5	IETF Integrated Services	77

Chapitre 1

Routing multicast

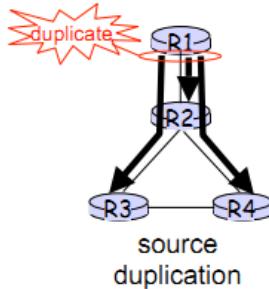
1.1 Routing en broadcast

Pour rappel, deux types de routage existent : les routages inter (BGP) et intra-domaines, soit à vecteur de distances (information globale envoyée en local, RIP), soit à état de lien (connaissance locale en diffusion globale, OSPF).

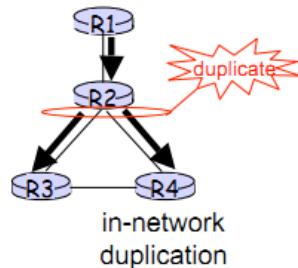
Le broadcast consiste en un envoi global, comme avec DHCP et ARP, au niveau lien. Le multicast consiste à n'atteindre qu'un sous-groupe de noeuds, pas tout le réseau.

Le broadcast multiplie les paquets

- soit à la source (création/transmission), mais il faut connaître toutes les destinations (multiple unicast)



- soit dans le réseau, en utilisant une adresse spéciale. Dans ce cas, tous les noeuds qui reçoivent le paquet le dupliquent et l'envoient aux voisins.



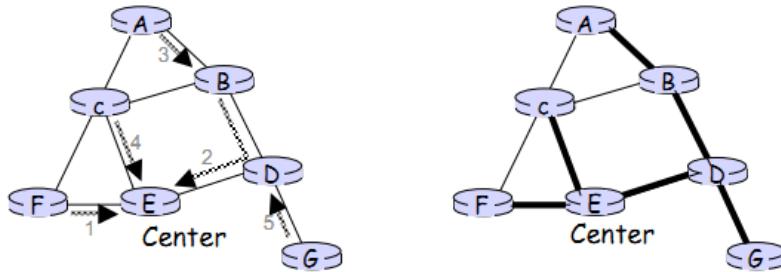
Il n'y a pas vraiment de duplication, mais cela peut mener à des cycles (broadcast storm). Il faut contrôler le flux pour les éviter, un état est nécessaire ; on a quelques méthodes qui permettent de le faire :

- TTL (protection ultime, au cas où aucune autre méthode n'aurait fonctionné). Peu efficace, mais c'est un bon garde-fou. Cependant, il n'existe qu'au niveau IP, pas Ethernet ;
- garder en mémoire des IDs de paquet déjà envoyés. Si le broadcast d'un paquet déjà enregistré est demandé, on ne fait rien ;
- reverse path forwarding (RPF) : le paquet sera forwardé s'il arrive par l'interface donnant lieu au plus petit chemin entre le noeud et la source ;
- créer un spanning tree dans le réseau, que chaque noeud aura, et broadcaster sur cet arbre uniquement (comme les switchs Ethernet). Non optimal.

1.1.1 Broadcast sur un spanning tree

Pour créer un spanning tree, on peut utiliser des BPDUs, ou bien utiliser un autre algorithme :

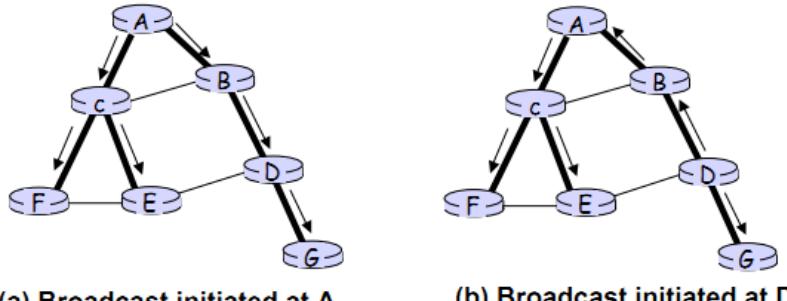
- choisir un noeud central dans le réseau, et le faire connaître à tous ;
- les noeuds calculent l'interface à utiliser pour accéder au noeud central. Pour ce faire, chaque noeud va envoyer un message en unicast pour rejoindre le noeud central. Ce message sera acheminé jusqu'à ce qu'il arrive à un noeud qui est déjà dans le spanning tree.



(a) Stepwise construction of spanning tree

(b) Constructed spanning tree

Ainsi, lorsqu'il y a un paquet à broadcaster, il est envoyé aux interfaces appartenant au spanning tree. Le flux de paquet va remonter jusqu'au noeud central, et passer par tous les noeuds.



On a un problème si le noeud central tombe en panne.

De nombreux spanning trees existent et il faut en choisir un. Il faudrait définir un coût pour chaque arbre et choisir le plus petit.

1.2 Routing multicast

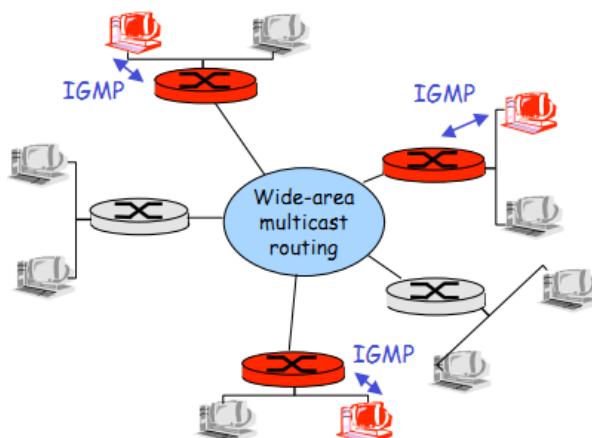
On veut atteindre un sous-ensemble de noeuds. Cela s'effectue au niveau d'IP.

Un groupe multicast est un ensemble de récepteurs et d'expéditeurs, indépendants des uns des autres, mais qui souhaitent interagir entre eux. Ce groupe est créé lorsqu'un expéditeur commence à envoyer de l'information à un groupe, ou lorsqu'un récepteur montre de l'intérêt à la réception des paquets d'un groupe, même si personne n'en enverra.

Le groupe est un point de rendez-vous logique ; l'expéditeur n'a pas besoin de connaître l'identité des récepteurs, c'est de l'**address indirection**. Pour multicaster à un groupe, le paquet est envoyé à une IP particulière. On ne sait pas à qui sera envoyé spécifiquement le paquet, car des personnes peuvent quitter ou entrer dans le groupe quand elles le souhaitent.

1.2.1 Adressage

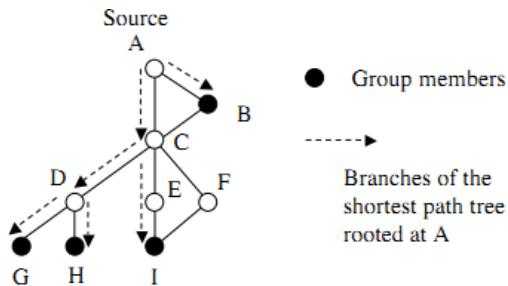
Des ranges d'adresses IP sont réservés pour les groupes, ce sont des IPs dites de classe-D. Les routeurs peuvent les détecter et les distinguer des paquets "classiques" à forwarder.



Ce sont les stations aux extrémités qui doivent indiquer qu'elles veulent recevoir des informations relatives à un groupe, mais elles restent sans connaissance des membres du groupe.

Cela conduit à de nombreux sous-problèmes : quels groupes sont actifs, comment en rejoindre un, comment le réseau connaît les récepteurs, etc.

1.2.2 Arbre de plus court chemin



Avec de l'unicast, 4 paquets seront envoyés par A, tandis qu'avec du multicast, 2 paquets seront envoyés.

Idéalement, on n'aimerait envoyer qu'un seul paquet multicast par lien ; on forme un arbre multicast dont la racine est la source. Cet arbre sera optimal s'il utilise les plus courts chemins entre la source et tous les récepteurs.

Pour avoir un arbre de plus court chemin en multicast, on peut réutiliser celui de l'unicast (obtenu avec Dijkstra ou autre).

Pour optimiser l'arbre, on peut minimiser le plus grand délai pour atteindre un noeud.

1.2.3 Difficultés en wide-area

On rencontre plusieurs difficultés avec le multicast en wide-area :

- Des membres peuvent joindre le groupe ou en partir dynamiquement. Il est donc nécessaire de mettre à jour dynamiquement l'arbre de plus court chemin.
- On aimerait qu'un récepteur puisse joindre ou quitter le groupe sans notifier explicitement la source, sinon cela ne se scalera pas (trop de trafic).
- Les feuilles comptent souvent des membres d'un même LAN, on aimerait donc exploiter les capacités de broadcast en LAN.

1.3 Multicast en local area

1.3.1 IGMP - Internet Group Management Protocol

IGMP est un protocole avec plusieurs instances dans un réseau, et établissant un dialogue entre un hôte et un routeur terminal. Lorsqu'un routeur sait qu'il y a des membres dans un groupe, il initialise lui-même les dialogues nécessaires avec les autres routeurs.

Ici, le routeur a l'initiative : il broadcast sur le réseau périodiquement un message indiquant les groupes disponibles. Il sait juste s'il y a des personnes dans le groupe ou non ; il ne sait pas combien il y a de membres.

Les hôtes répondent quand ils sont intéressés. Pour éviter un trafic trop élevé,

- ils répondent après un temps de réponse choisi aléatoirement, afin d'éviter d'avoir un flux massif de messages en même temps
- les réponses sont en broadcast, du coup les autres hôtes savent qu'il n'est pas nécessaire de répondre.

Au final, les hôtes ne connaissent pas les identités de tous les hôtes intéressés, mais ils savent qu'il y en a au moins un.

On peut filtrer les sources (IGMPv3) : un hôte peut entrer dans un groupe, et choisir de ne pas recevoir tous les paquets de tous les membres du groupe. Cela permet de se focaliser sur un ou des membres, ou supprimer du "bruit".

Si un hôte n'est pas intéressé par un groupe, il peut ne pas répondre aux requêtes du routeur.

Pour quitter un groupe, un hôte envoie un message explicite. Avant de faire partir un membre, le routeur doit savoir s'il y a encore des autres membres qui y sont connectés (car une réponse peut cacher un ou plusieurs hôtes).

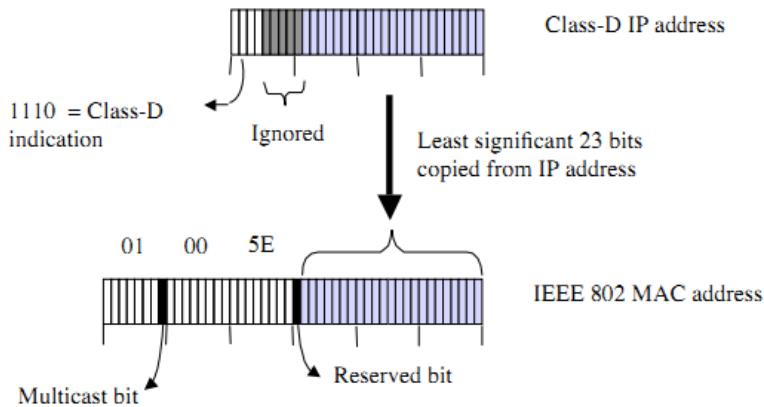
1.3.2 Multicast avec le broadcast LAN

Le multicast en wide area peut exploiter les capacités de broadcast d'un LAN, par exemple Ethernet va multicaster tous les paquets avec le bit de multicast à 1 sur les adresses de destination.

Pour qu'un routeur envoie un paquet à des membres d'un groupe, il faut utiliser une adresse MAC multicast, un problème similaire au mapping d'une IP unicast à une adresse MAC unicast. Un hôte devra écouter son adresse MAC, mais aussi celle du groupe.

Le protocole ARP est utilisé en unicast, mais ce n'est pas vraiment adapté pour le multicast.

Le problème est de mapper des IP multicast à des adresses MAC multicast, donc de déterminer la MAC adresse d'une adresse IP de classe D.



On a ainsi plusieurs adresses de classe D mappées sur la même adresse MAC.

Une IP multicast est détectée avec le préfixe, une adresse MAC avec un bit. Les bits sont recopiés de l'IP vers MAC.

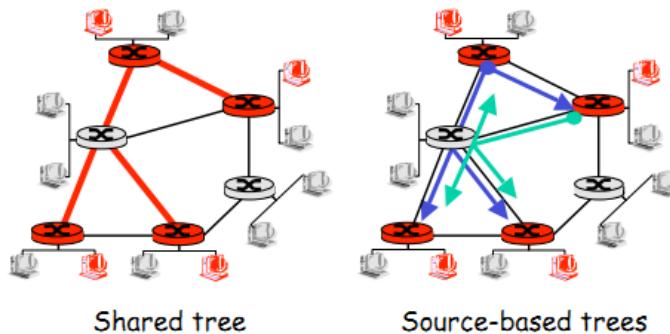
Il y a un problème de mapping (scénario peu probable) : deux groupes peuvent avoir une même adresse MAC (mais de deux adresses IP multicast différentes). Cependant, les paquets indésirables seront jetés dans la couche supérieure, car l'adresse IP n'est pas correcte.

1.4 Multicast en wide area

Le but est de distribuer les paquets destinés à un groupe venant de n'importe quelle source à tous les routeurs sur le chemin d'un membre du groupe.

Le problème est de connaître les routeurs associés à un groupe, donc de trouver un (ou des) arbre(s) connectant les routeurs qui ont des membres du groupe multicast. Deux types d'arbre permettent de router les paquets aux membres :

- group-shared tree : un seul arbre pour tous les routeurs. Il n'est pas nécessairement optimal, mais c'est une méthode plus facile et scalable (une entrée par groupe dans une table de forwarding) ;
- source-based tree : autant d'arbres que de sources. C'est optimal, mais plus lourd. On y utilise le reverse path forwarding (RPF).



1.4.1 Source-based tree

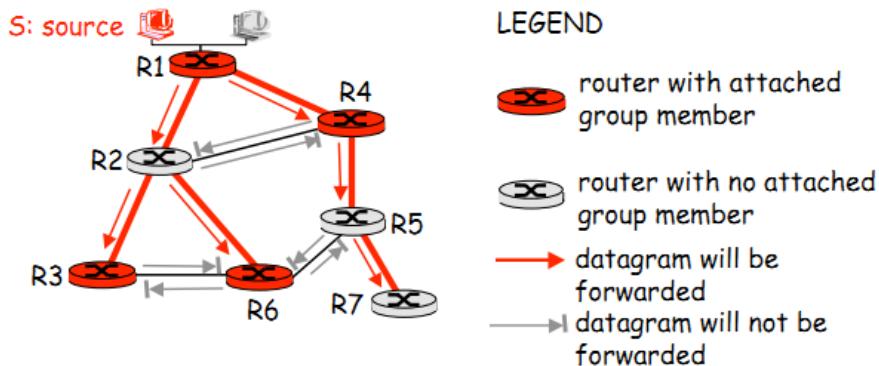
Chaque source va calculer le SPT (shortest path tree) vers tout les récepteurs ; avec la connaissance de la topologie, il suffit d'appliquer un algorithme comme Dijkstra.

Au lieu de partir d'un noeud racine vers les feuilles, on pourrait faire le contraire, c'est-à-dire partir des feuilles et aller à la racine. Les chemins seront les mêmes si le coût d'un lien est le même dans un sens et dans l'autre. On peut également utiliser Dijkstra pour le SPT, en utilisant le coût opposé des liens.

Reverse Path Forwarding

Pour du multicast, les SPTs ne sont pas calculés explicitement ; on va se baser sur la connaissance des plus courts chemins unicast du routeur. C'est la méthode du Reverse Path Forwarding, et qui applique une règle simple.

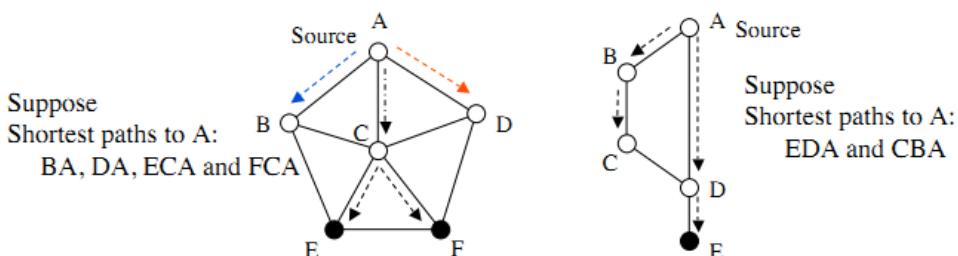
Si un routeur reçoit un paquet multicast par une interface qui est le lien qui aurait été utilisé pour atteindre la source du paquet, alors le paquet a suivi le plus court chemin, et donc on le propage sur tous les autres liens. Sinon, on ignore le paquet.



Le résultat de ce comportement est un SPT renversé, ce qui peut être pénalisant pour des liens asymétriques. On a cependant toujours un problème : on a des routeurs qui ne comptent pas de membre dans le groupe et qui forwardront quand même les paquets (ce soucis est réglé par le pruning).

Reverse Path Forwarding amélioré

L'idée est qu'un routeur n'enverra pas le paquet s'il n'est pas lui-même sur le chemin le plus court entre le routeur en aval et la source.



Un routeur peut le savoir s'il lance un protocole link-state ou distance vector avec le poison reverse. Ainsi, les paquets sont transmis une seule fois sur l'arbre de plus court chemin renversé.

Le routeur pourrait calculer un nouveau spanning tree en utilisant un algorithme à état de lien et en prenant la place du routeur. S'il est sur le chemin le plus court, il enverra le paquet, sinon il ne le fera pas. Il y a un plus grand coût en terme de calcul, mais on économise de la bande passante.

La différence avec un algorithme à vecteur de distances est qu'on ne connaît que le voisinage. Cependant, grâce au poisoned reverse, on saura si on est sur le chemin le plus court : si la distance du routeur à vérifier à la source est infinie.

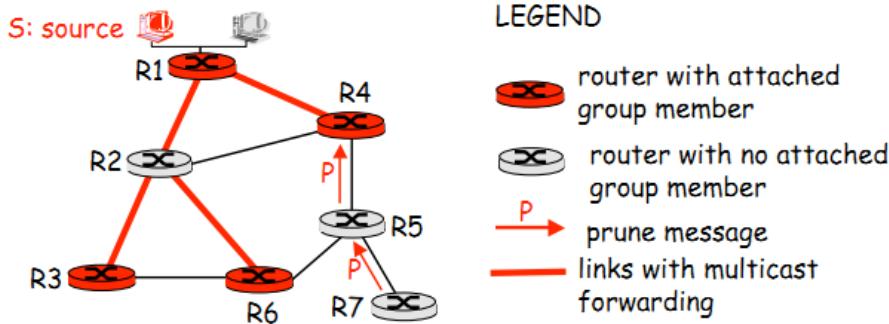
Il y a un risque de confusion si le routeur en aval a le choix entre plusieurs plus court chemins vers la source (par exemple, à gauche, ECA et EBA sont tous les deux des plus courts chemins vers A ; il y a un risque que B et C ne forwardent aucun paquet à E, car ils pensent que l'autre va le faire).

La solution est de quand même forwarder, ce qui fait que des duplicates sont possibles.

Reverse Path Forwarding avec pruning

Il est possible que l'arbre de forwarding contienne des sous-arbres qui n'ont pas de membres dans le groupe. Il est donc inutile de leur envoyer des datagrammes.

L'idée est que les routeurs qui ne comptent pas de membre envoient des messages dit "prune", qui empêche l'envoi de ces datagrammes.



L'arbre est ainsi optimal si on le considère inversé.

Table de forwarding multicast

Pour une table de forwarding unicast, on a l'association préfixe d'IP de destination → port de sortie. Une entrée d'une table de forwarding multicast est de la forme

(IP source, IP multicast de destination) : port d'entrée → {ensemble de port de sortie}

On a une entrée par paire source-destination. Si le port d'entrée ne correspond pas, on jette le paquet (mécanisme de reverse path forwarding).

L'adresse IP source doit être spécifiée, car il y a un arbre par IP source dans un routeur.

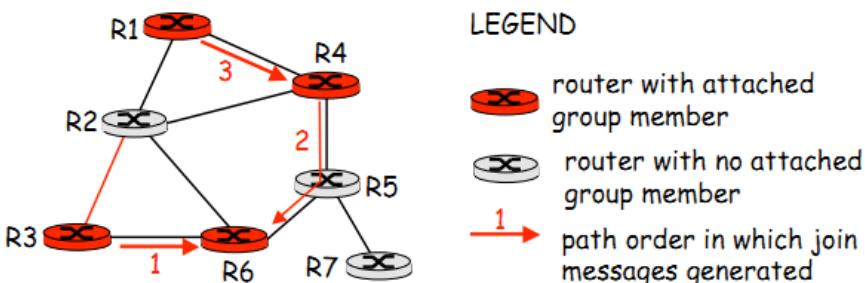
L'avantage de la technique source-based est que c'est facile à implémenter et l'arbre est optimal, mais dans le sens inverse uniquement (destination vers source). Cela pose problème si les métriques ne sont pas les mêmes.

1.4.2 Center-based

On va calculer un arbre dit de Steiner : c'est l'arbre de coût minimal connectant tous les routeurs possédant des membres d'un groupe. C'est un problème NP-complete, avec d'excellents heuristiques. Cet arbre sera unique et commun à tous les routeurs.

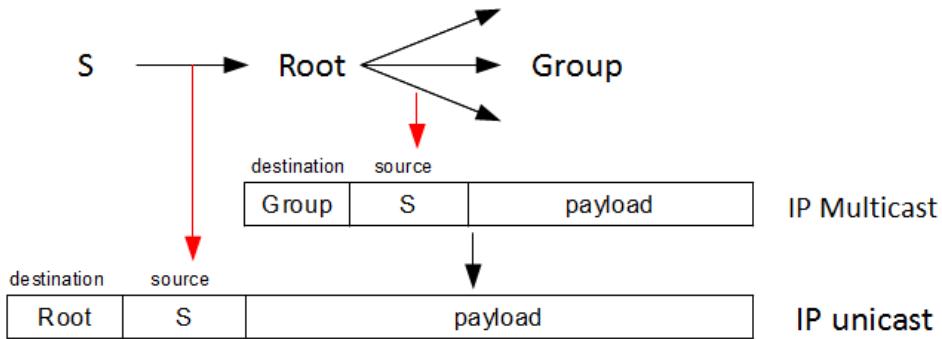
On va désigner un routeur comme le centre de l'arbre. Pour le rejoindre, les edge routeurs doivent envoyer un message unicast de type join, adressé au routeur central.

Le message est analysé par les routeurs intermédiaires, et forwardé vers le routeur central. Il ne sera plus forwardé si on arrive à une branche de l'arbre, ou si on arrive au centre. Le chemin parcouru par le message formera une nouvelle branche.



Si une source veut envoyer un paquet à un groupe, il va l'envoyer à la racine, qui va le forwarder à tout l'arbre. La source a donc besoin de deux adresses : celle du groupe et celle de la racine. Cependant, avec seulement l'adresse du groupe, le paquet sera dropé au premier routeur, car il n'y aura pas d'entrée dans la table de forwarding.

Pour faire parvenir le paquet au centre, la source va utiliser l'encapsulation/tunneling : le paquet multicast va être placé à l'intérieur d'un paquet unicast IP, qui sera envoyé au centre. Le centre va supprimer le header unicast et envoyer le paquet multicast aux membres du groupe.



Les entrées de la table d'acheminement multicast auront la forme suivante :

(* , IP multicast de destination) : port d'entrée → { ensemble de port de sortie }

La table d'acheminement est similaire, mais l'adresse source n'est pas nécessaire pour le forwarding (utilisation d'une wildcard). Le paquet doit également venir d'un port "appartenant" à l'arbre.

Le chemin n'est pas optimal, car il y a un détour à effectuer avant de fonder le réseau. Par contre, il y a peu d'entrées multicast dans la table d'acheminement des routeurs ; une entrée est valide pour toutes les sources, et il y a une entrée par destination (et non plus par paire source-destination).

Cette technique n'est pas utilisée en pratique car

- la complexité en temps de calcul est élevée,
- des informations sur tout le réseau sont nécessaires, et
- l'arbre est monolithique ; il faudra relancer l'algorithme chaque fois qu'un routeur rejoint/quitte le groupe.

1.5 Exemples de protocoles multicast

1.5.1 DVMRP

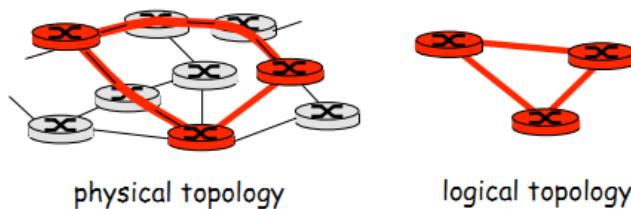
DVMRP est un protocole "historique", basé sur des vecteurs de distances. Il utilise un arbre source-based, avec le Reverse Path Forwarding et du pruning.

DVMRP est un protocole à part qui va construire sa table d'acheminement multicast à côté de celle des IP unicast. C'est une approche robuste, car on suppose que tous les routeurs ne supportent pas nécessairement DVMRP, et on ne fait pas d'hypothèses sur l'unicast.

Toutes les minutes, DVMRP relance les routeurs qui ont utilisé le pruning, cela permet de détecter les nouveaux membres. De plus, les routeurs peuvent notifier le routeur racine rapidement avec un mécanisme à part (IGMP), en supplément de la relance toutes les minutes.

Historiquement, il tournait au niveau des hôtes, car tous les routeurs ne l'implémaient pas. Cela donnait un résultat similaire au peer-to-peer (multicast backbone).

L'encapsulation/tunneling est utilisé pour traverser les liens entre deux noeuds du groupe multicast : tout est transparent pour les noeuds intermédiaires.



Les datagrammes multicast sont donc encapsulés dans des datagrammes "normaux" (pas avec une adresse multicast), et sont envoyés à travers un tunnel via de l'unicast IP "normal" jusqu'au routeur multicast. Ce dernier va alors décapsuler le datagramme pour retrouver le paquet multicast.

1.5.2 PIM - Protocol Independant Multicast

C'est le protocole le plus utilisé actuellement, et ne dépend pas d'algorithme de routage unicast spécifique. Il est compliqué car il utilise les deux approches pour construire l'arbre (source-based et center-based) et fonctionne sous deux modes :

- dense : les membres du groupe sont considérés comme compactes, "proches" en terme de proximité. On considère aussi qu'on a beaucoup de bande passante. Du coup, on va supposer qu'un routeur est membre du groupe tant qu'il n'envoie pas un message de pruning. La construction de l'arbre multicast se fait sur base des données (RPF). La bande passante est consommée sans modération.
- sparse : le nombre de réseaux avec des membres d'un groupe est plus petit que le nombre de réseaux interconnectés ; les membres du groupes sont plus éparpillés, ou peu nombreux. La bande passante est considérée comme réduite. Ainsi, les routeurs ne font pas partie du groupe tant qu'ils ne l'ont pas explicitement indiqué. La construction de l'arbre multicast se fait par les récepteurs (center-based). La bande passante est utilisée avec parcimonie.

Naturellement, si le groupe est dense, beaucoup de routeurs sont intéressés. Il est donc mieux d'effectuer une sorte de broadcast et d'utiliser le pruning, il y aura moins de gaspillage qu'avec un groupe peu dense.

Dense mode

En mode dense, PIM n'utilise pas son propre protocole unicast (contrairement à DVMRP), et ne connaît pas la technologie unicast. Du coup, il n'implémente pas le cleverer RPF (Reverse Path Forwarding, où en plus de vérifier l'interface d'arrivée du paquet, on veut savoir si un routeur voisin acceptera ou non notre paquet, en utilisant son FIB), cela pourrait mener à des anomalies.

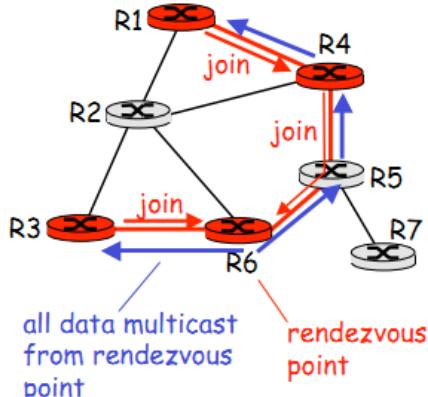
Pour rappel, la table de routage (RIB, routing information base) est utilisée comme structure de données pour générer la table de forwarding (FIB, forwarding information base), qui redirige les paquets vers les interfaces.

En multicast, soit on recréé un RIB, soit on réutilise le RIB de l'unicast (car il contient plus d'infos intéressantes que le FIB). Pour être indépendant, il faut éviter de reprendre RIB, et un protocole comme PIM peut réutiliser le FIB.

La version cleverer est impossible à implémenter, car on ne dispose que du FIB du noeud courant, et pas le RIB courant ni le FIB des routeurs voisins. C'est moins optimal, mais plus robuste.

Sparse mode

On va utiliser l'approche center-based, les messages join sont envoyés à un point de rendez-vous (RP). Après qu'un routeur ait rejoint l'arbre, il passe sur un arbre de source spécifique, ce qui augmente les performances (moins de concentration, chemins les plus courts).



Les sources vont donc envoyer les données en unicast au point de rendez-vous, qui va les envoyer aux arbres dont la racine est le point de rendez-vous.

Le point de rendez-vous peut envoyer des messages stop s'il n'y a aucun destinataire attaché.

Le terme "center" est remplacé par "rendez-vous" car, dans le cas de l'approche center-based, le centre peut recevoir beaucoup de trafic, et s'il tombe en panne, toute la connectivité est perdue. Ici, le point de rendez-vous ne supporte pas tout le trafic.

Certains routeurs dans un domaine sont ainsi configurés comme étant des candidats pour des points de rendez-vous. C'est un BSR (BootStrap Router) qui va être élu dynamiquement (parmi les points de rendez-vous) dans un domaine PIM. Ces élections sont basées sur des priorités et sur l'adresse IP la plus importante.

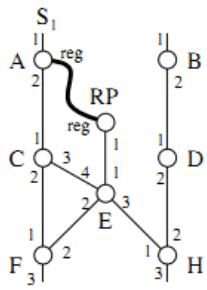
Le BSR va alors allouer les groupes aux points de rendez-vous, et distribuer les mappings RP → { groupes } à tous les routeurs du domaine, via des BSM (Bootstrap Messages) envoyés périodiquement. Des adresses IP multicast sont réservées pour ces usages.

Il y a une nouvelle élection si le routeur sélectionné tombe en panne (mécanisme keep-alive).

Chaque RP envoie périodiquement des messages aux routeurs qui y sont connectés, et signifiant qu'il est toujours "vivant". Ainsi, si les routeurs ne reçoivent plus ces messages (après un timeout), ils tenteront de trouver un nouveau RP, et de s'y joindre.

L'arbre peut être étendu à des routeurs pour éviter trop de tunneling. De plus, il est possible d'envoyer des messages dans le cas où il n'y a pas de récepteur, pour éviter du trafic inutile.

Exemple détaillé



1. A (on behalf of S_1) registers to RP as a source for group G

Forwarding tables:

A: (S_1, G) : 1 → {reg}

reg is a virtual "tunnel interface"

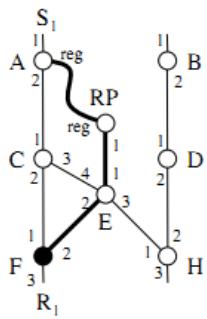
A sends a register message to RP.

A will later tunnel multicast packets to RP.

RP has no receivers, so it sends back a register-stop message to A.

Register messages are refreshed periodically (and register-stops retransmitted if still no receivers)

Other routers have empty tables



2. R_1 joins group G → F learns R_1 's interest by IGMP.

F sends JOIN (*, G) to E

E sends JOIN (*, G) to RP

RP doesn't send register-stops any more to A.

Forwarding tables:

A: (S_1, G) : 1 → {reg}

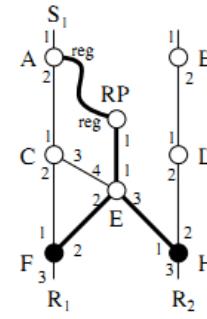
RP: $(*, G)$: reg → {1}

E: $(*, G)$: 1 → {2,3}

F: $(*, G)$: 2 → {3}

F: $(*, G)$: 2 → {3}

Other routers have empty tables



3. R_2 joins group G → H learns R_2 's interest by IGMP.

H sends JOIN (*, G) to E.

E need not propagate the join.

Forwarding tables:

A: (S_1, G) : 1 → {reg}

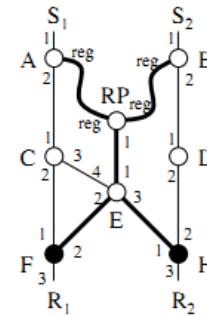
RP: $(*, G)$: reg → {1}

E: $(*, G)$: 1 → {2,3}

F: $(*, G)$: 2 → {3}

H: $(*, G)$: 1 → {3}

Other routers have empty tables



4. B (on behalf of S_2) registers to RP as a source for group G

Forwarding tables:

A: (S_1, G) : 1 → {reg}

B: (S_2, G) : 1 → {reg}

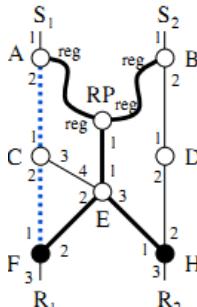
RP: $(*, G)$: reg → {1}

E: $(*, G)$: 1 → {2,3}

F: $(*, G)$: 2 → {3}

H: $(*, G)$: 1 → {3}

Other routers have empty tables



5. F creates a source-based tree for S_1

Forwarding tables:

- A: (S_1, G) : 1 → {reg, 2}
- B: (S_2, G) : 1 → {reg}
- C: (S_1, G) : 1 → {2}
- RP: $(*, G)$: reg → {1}
- E: $(*, G)$: 1 → {2, 3}
- F: $(*, G)$: 2 → {3}
- (S_1, G) : 1 → {3}
- H: $(*, G)$: 1 → {3}
- Other routers have empty tables

F sends JOIN (S_1, G) to C.

C sends JOIN (S_1, G) to A.

When F receives data from S_1 via C,

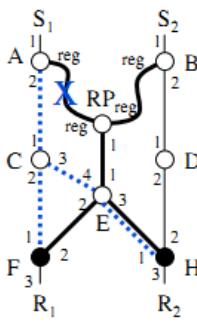
F sends a PRUNE (S_1, G) to E.

E does not forward the prune to RP because H still receives packets from S_1 via the shared tree.

And A still sends packets on both trees.

When a (S, G) entry and a $(*, G)$ entry co-exist in the table,

- if the source of the incoming packet is S, the rule is given by the (S, G) entry
- if the source of the incoming packet is not S, the rule is given by the $(*, G)$ entry
- There cannot be two entries in a given table with the same S and G.



6. H creates a source-based tree for S_1

Forwarding tables:

- A: (S_1, G) : 1 → {2}
- B: (S_2, G) : 1 → {reg}
- C: (S_1, G) : 1 → {2, 3}
- RP: $(*, G)$: reg → {1}
- (S_1, G) : reg → {}
- E: $(*, G)$: 1 → {2, 3}
- (S_1, G) : 4 → {3}
- F: $(*, G)$: 2 → {3}
- (S_1, G) : 1 → {3}
- H: $(*, G)$: 1 → {3}
- (S_1, G) : 1 → {3}
- Other routers have empty tables

H sends JOIN (S_1, G) to E.

E sends JOIN (S_1, G) to C.

When E receives data from S_1 via C, E sends a PRUNE (S_1, G) to RP.

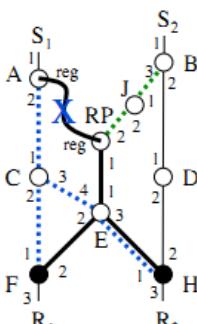
RP has now no receivers for S_1 via the shared tree, so RP will re-start sending register-stops to A

Entry (S_1, G) : 1 → {}

is useless because

(S_1, G) : 4 → {3} suffices

This entry may seem useless because it has the same forwarding as $(*, G)$: 1 → {3} but it is useful for triggering periodic refreshes.



6. RP switches to the shortest-path tree for S_2

Forwarding tables:

- A: (S_1, G) : 1 → {2}
- B: (S_2, G) : 1 → {3}
- J: (S_2, G) : 1 → {2}
- C: (S_1, G) : 1 → {2, 3}
- RP: $(*, G)$: reg → {1}
- (S_1, G) : reg → {}
- (S_2, G) : 2 → {1}
- E: $(*, G)$: 1 → {2, 3}
- (S_1, G) : 4 → {3}
- F: $(*, G)$: 2 → {3}
- (S_1, G) : 1 → {3}
- H: $(*, G)$: 1 → {3}
- (S_1, G) : 1 → {3}
- Other routers have empty tables

RP sends JOIN (S_2, G) toward B.

All routers on the path (here J) create (S_2, G) entries.

B updates its (S_2, G) entry.

B stops tunnelling multicast packets to RP and sends them natively in multicast to the group (actually to RP as only member of the shortest path tree at present).

Packets from S_2 still follow the shared tree from RP to receivers (until receivers switch to a shortest path tree).

Mécanisme de rafraîchissement

Pour la version Sparse de PIM, le même message est utilisé pour joindre et pruner : c'est un message Join/Prune (des bits dans le message lèvent l'ambiguïté). Ainsi, le même message peut joindre et pruner en même temps.

Chaque routeur envoie périodiquement des messages Join/Prune à ses voisins pour chaque entrée active de sa table. Un message est aussi envoyé à chaque fois qu'une nouvelle entrée est établie.

Il n'y a pas une connaissance explicite de l'ACK des messages de join/prune ; les pertes de paquets sont récupérées avec le mécanisme de rafraîchissement périodique.

Si les tables unicast changent, les arbres multicast doivent être mis à jour au prochain rafraîchissement ; PIM sera relancé, car il est entièrement basé sur la table d'acheminement.

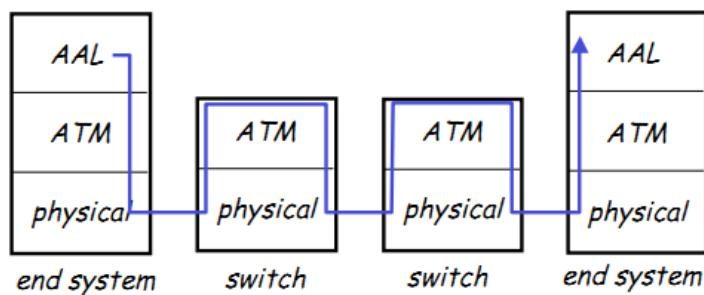
Chapitre 2

Réseaux ATM et MPLS

Ce sont des technologies de circuits virtuels qui offrent des architectures complètes, dérivées des compagnies de téléphone. Elles sont déployées à des plus petites échelles que IP. Pour améliorer la technologie IP, on peut passer par un réseau ATM.

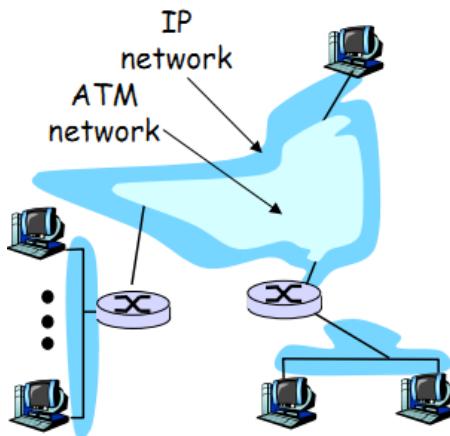
2.1 ATM - Asynchronous Transfer Mode

Cette technologie se distingue d'IP par le fait qu'elle est asynchrone, et qu'elle vise à avoir de bons taux de transfert tout en rassemblant toutes les fonctionnalités en un seul réseau. Le but premier est de donner un moyen de transport à la voix, la vidéo et les données, avec des garanties (contrairement au "best effort" d'IP). Il s'agit d'une évolution du réseau téléphonique qui supporte le packet-switching à travers des circuits virtuels.



On distingue trois couches dans un réseau ATM :

1. la couche adaptative (adaptation layer) : seulement aux extrémités d'un réseau, elle permet la segmentation et le rattachement des données. Elle correspond à la couche transport d'Internet ;
2. la couche ATM : possède les fonctionnalités réseau, c'est-à-dire le routage et le cell switching ;
3. la couche physique.



ATM est plutôt considéré dans la couche 2 d'un réseau IP (pas les applications) ; actuellement il est imbriqué dans IP et est invisible à l'utilisateur. Il est surtout utilisé pour les connexions entre des routeurs et des noeuds dans le réseau (edge-to-edge) et non aux extrémités (end-to-end).

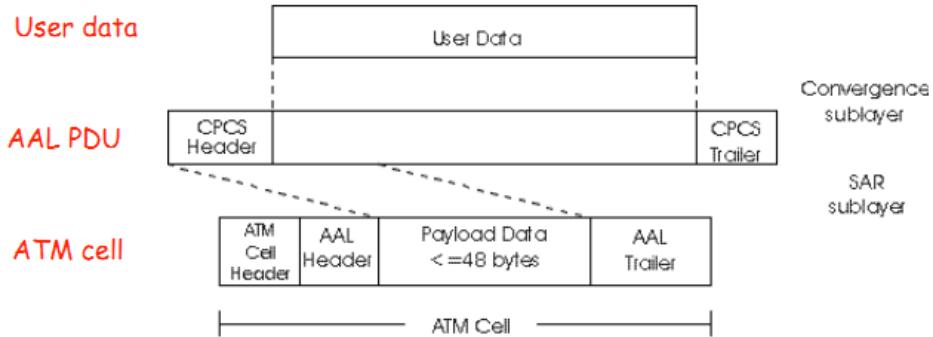
2.1.1 ATM Adaptation Layer - AAL

Cette couche permet d'adapter les couches supérieures (généralement IP, mais des applications ATM natives sont possibles) aux couches ATM inférieures. Elle n'est située qu'aux extrémités d'un réseau ATM (donc dans les routeurs IP), pas dans les switchs ATM.

Il existe plusieurs versions de couche AAL, selon la classe de service ATM souhaitée :

- AAL1 : pour des services CBR (Constant Bit Rate) (ex : émulation de circuits, VoIP)
- AAL2 : pour des services VBR (Variable Bit Rate) (ex : vidéo MPEG)
- AAL5 : pour des données (ex : datagrammes IP)

Comme IP, le payload est inséré dans une trame, mais au lieu de l'envoyer directement, elle est fragmentée en cellules. Cette fragmentation permet de satisfaire des critères de qualité selon les services souhaités.



2.1.2 Circuits virtuels

On a un mécanisme de transport des cellules à travers des circuits virtuels. Un appel au réseau est effectué afin d'en mettre un en place, et ce avant que les données soient envoyées.

Chaque paquet est estampillé par un identifiant de circuit virtuel (VCI), qui n'est pas l'adresse de destination.

Chaque switch sur le chemin entre la source et la destination va maintenir un état à chaque connexion qui passe par lui : à un identifiant de circuit virtuel est associé une interface de sortie. Des ressources peuvent être réservées (buffer, cpu, bande passante), ce qui permet d'avoir des performances similaires à des circuits.

Des circuits virtuels permanents (PVC) peuvent être définis, afin de ne pas chaque fois les rétablir. Ils établissent généralement une route permanente entre les routeurs IP.

Il existe aussi des circuits virtuels dynamiques (switched VC) qui sont utilisés pour des transferts courts, dont l'installation se fait lors d'un appel.

L'avantage des circuits virtuels réside dans les performances et les garanties de QoS (Quality of Service), que ce soit en bande passante ou en délais. Des désavantages :

- les datagrammes sont mal supportés, alors que c'est le type de trafic le plus courant ;
- pas scalable, beaucoup de connexions sont nécessaires entre des paires source/destination pour un circuit virtuel permanent ($\mathcal{O}(N^2)$) ;
- pour un circuit virtuel dynamique, il y a un délai dû à l'établissement du circuit et dans le processing de l'overhead pour des connexions courtes.

2.1.3 Couche ATM

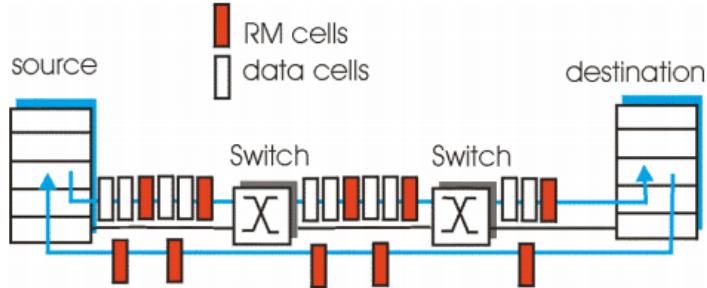
Le service qu'offre cette couche est le transport de cellules ATM à travers un réseau ATM. Elle correspond à la couche réseau d'IP, mais ses services différent.

Network Architecture	Service Model	Guarantees?				Congestion feedback
		Bandwidth	Loss	Order	Timing	
Internet	best effort	none	no	no	no	no (inferred via loss)
ATM	CBR	constant rate	yes	yes	yes	no congestion
ATM	VBR	guaranteed rate	yes	yes	yes	no congestion
ATM	ABR	guaranteed minimum	no	yes	no	yes
ATM	UBR	none	no	yes	no	no

Pour le CBR et le VBR, il n'y a pas de problème de timing, car de la bande passante est déjà réservée, donc pas de bufferisation.

Le mode ABR (Available Bit Rate) consiste en un bitrate variable selon l'état du réseau. Il y a la possibilité pour les applications de demander une diminution du débit s'il y a des pertes (congestion) ou de l'augmenter si le chemin est sous-utilisé. C'est similaire à TCP, si ce n'est que la congestion est donnée explicitement, alors que TCP la déduit des pertes.

Des cellules dites RM (ressource management) peuvent être utilisées pour modifier le débit de la source ou marquer de la congestion, par le destinataire ou les switchs.



Ces cellules sont entrelacées avec des cellules de données. Des bits à l'intérieur même de ces cellules sont modifiés par les switchs, pour donner un feed-back du réseau. Il y a notamment :

- un bit NI (no increase) : il ne faut pas augmenter le débit (congestion bénigne)
- un bit CI (congestion indication)
- deux bytes ER (explicit rate) : spécifient le débit maximal de l'expéditeur utilisable sur le chemin
- un bit EFCI (explicite forward congestion indication) : bit mis à un s'il y a des switchs congestionnés. Lorsqu'une cellule de données précédant une cellule RM a ce bit à 1, le destinataire met à 1 le bit CI de la cellule RM à renvoyer.

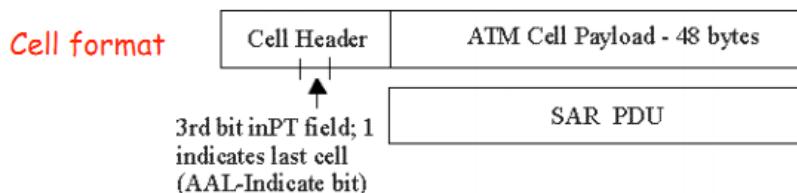
Ces cellules sont retournées à l'expéditeur par le receveur, sans modifier les bits.

Cellule ATM



Le header est de 5 bytes et comporte les champs suivants :

- VCI : le numéro de canal virtuel, qui change d'un lien à un autre
- PT : Payload type, permet de discriminer les cellules de données et les cellules RM
- CLP : Cell Loss Priority. S'il est à 1, cela indique que la cellule n'est pas prioritaire, et qu'elle peut donc être jetée en cas de congestion.
- HEC : Header Error Checksum, check de redondance cyclique permettant de détecter des erreurs.



Les paquets sont de taille fixe (48 bytes) et petits, sinon il faudra du temps pour collecter des bits afin de créer un paquet plus grand, ce qui peut être problématique pour l'application (pour la voix par exemple).

Une taille fixe implique une taille petite, sinon il y aurait un gaspillage de ressources. De plus, dans un switch, lors de la redirection d'un paquet, avec une matrice de commutation, le parallélisme est amélioré (tous les transferts prennent le même temps) et on peut calculer des débits de cellules et donner des garanties facilement.

2.1.4 Couche physique

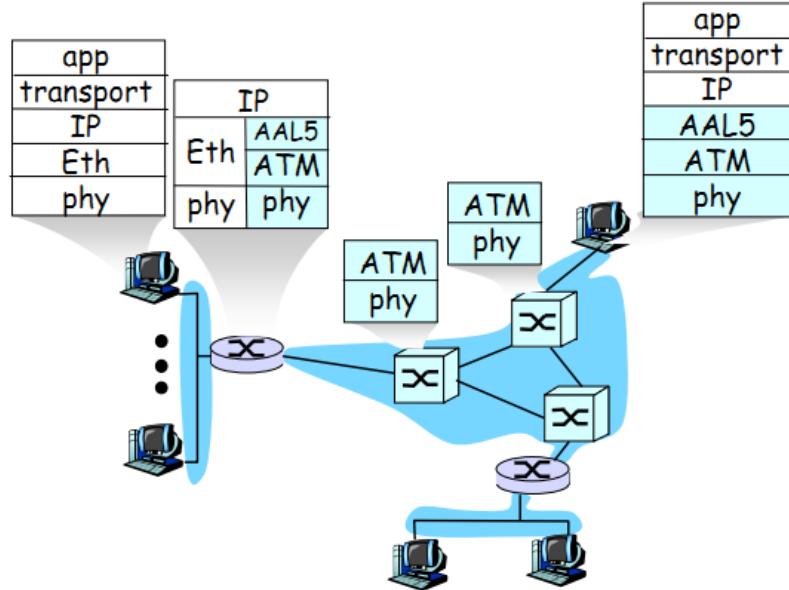
Les débits (OC3, OC12, etc) ne sont pas entiers, car les normes sont spécifiées pour utiliser des composants analogiques.

2.2 IP sur ATM

ATM a été placé sous IP (et non l'inverse), car IP a été très vite populaire. ATM se voulait un concurrent d'Ethernet, plus rapide et également disponible aux hôtes. Le problème était l'aspect bon marché d'Ethernet qu'ATM n'avait pas, car c'est une technologie beaucoup plus simple.

ATM s'est donc retrouvé dans le cœur, au niveau des ISP, au même titre qu'Ethernet. Certains routeurs disposent ainsi de deux types d'interface (ATM et Ethernet).

Les réseaux ATM remplacent généralement un réseau IP ; les trames IP sont encapsulées et fragmentées en cellules ATM.



Pour un datagramme, à la source (hôte ou routeur), la couche IP mappe l'adresse IP et l'adresse ATM de destination, en utilisant ATM ARP. Ensuite, le paquet IP est passé à la couche adaptative (AAL5) qui l'encapsule (c'est l'équivalent d'une trame Ethernet). Ensuite la couche adaptative segmente la trame en cellules ATM, qui sont ensuite passées à la couche ATM. Le réseau ATM va alors déplacer les cellules le long du circuit virtuel jusqu'au destinataire. A l'arrivée, AAL5 va réassembler le datagramme original, et si le CRC est valide, il est passé à IP.

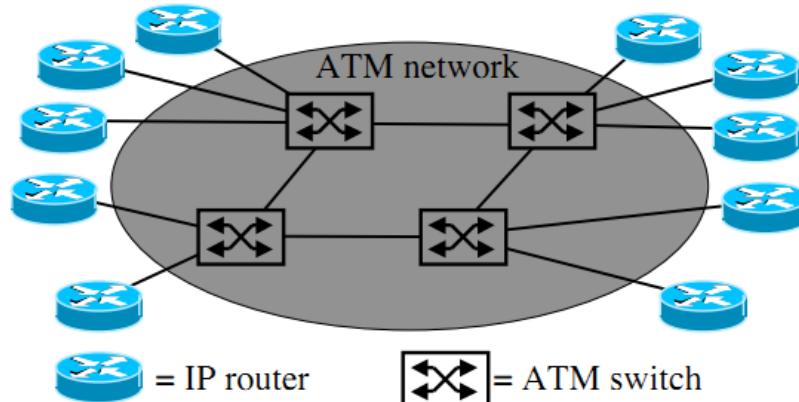
Le problème est de traduire l'adresse IP du routeur à atteindre en une adresse ATM. Il suffit de mapper les adresses IP à des adresses MAC, cependant, le broadcast n'est pas disponible aussi facilement qu'en Ethernet avec ARP ; il faudrait un circuit qui permet d'atteindre tous les périphériques. On définit pour cela l'ATMARP.

Un serveur sera désigné dans le système, auquel on adresse les requêtes de type ARP (résolution d'adresses, ne fonctionne pas comme Ethernet). Ensuite, il faudra établir un circuit. Ces étapes (résolution et établissement de circuit) peuvent être passées si le circuit est déjà établi.

2.2.1 Topologies

Full-mesh

Chaque routeur doit être connecté avec les autres routeurs grâce à un circuit virtuel permanent. Il y a donc $\frac{n(n-1)}{2}$ circuits virtuels permanents, ce qui donnera une topologie de type full-mesh.



Lors d'une perte de connexion, soit on peut tenter de la récupérer au niveau ATM, soit on recalcule des tables d'acheminement au niveau IP. En pratique, cela dépend des temps de récupération de ces deux méthodes.

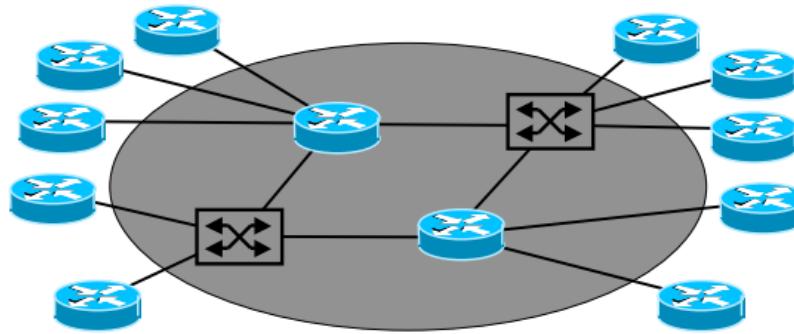
Cependant, il y a des problèmes de scalabilité :

- le nombre de circuits virtuels ATM est quadratique par rapport au nombre de routeur ;
- avec autant de liens, cela peut créer des problèmes lors de la création des tables d'acheminement. Par exemple, pour OSPF, lors de la découverte des voisins (pour générer la topologie), un paquet aura une taille N (car N voisins à décrire) qui sera envoyé à N liens, et ce pour chacun des N noeuds : la complexité en nombre de message est cubique ;
- si un lien ATM tombe en panne, on peut perdre énormément de liens IP. La récupération peut augmenter la charge, de l'ordre de N^4 (routing storm).

En pratique, on limitera le nombre de routeurs (environ 100).

Intégration de routeurs Ethernet

Une solution est de découper le réseau en plus petits sous-réseaux, en plaçant des routeurs Ethernet dans le coeur du réseau.



Cela réduit le nombre de voisin des routeurs, de même que le nombre de circuits virtuels (notamment ceux perdus lors d'une panne ; il n'y a plus de edge-to-edge). Cependant,

- il y a plus de hop à effectuer ;
- il peut y avoir une perte de qualité de service ATM (cas d'un routeur congestionné). En effet, l'ATM switching est délaissé pour l'IP switching ;
- certains routeurs peuvent se passer d'ATM pour communiquer entre eux (cas extrêmes) ;
- les routeurs introduisent des délais ; un routeur doit attendre toutes les cellules du paquet afin d'avoir la destination IP, puis réencapsuler et refragmenter le paquet et enfin le renvoyer. Les délais seront plus élevés.

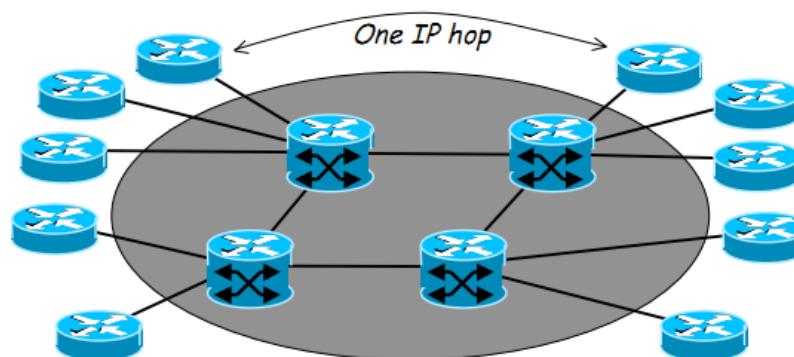
L'idéal serait de ne pas devoir réassembler et désassembler les cellules ATM.

Intégration de switchs IP

On a donc, pour les gros réseaux, un dilemme entre l'acheminement de données (où ATM est meilleur) et la signalisation (mise à jour du routage) (où IP est meilleur).

La solution serait d'avoir les switchs ATM au coeur et d'utiliser le routage IP sur eux. Cela réduit le nombre d'adjacences OSPF et il n'y a pas d'overload pour les mises à jour de routage. Le routing d'ATM est en quelque sorte remplacé par le routing d'IP ; on utilise du hardware ATM avec du software IP.

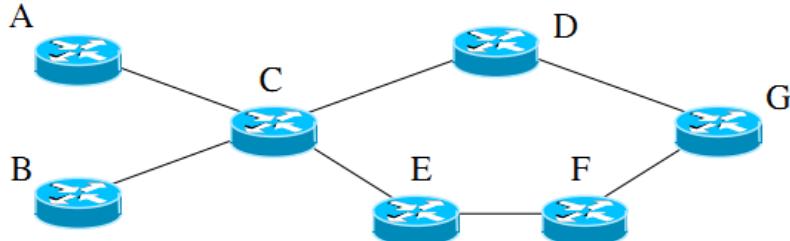
En découlent des switchs IP, des hybrides entre un routeur IP et un switch ATM. Des connexion ATM sont utilisées pour communiquer entre les switchs. Cependant, le chemin est généré par ces hybrides.



Le chemin suivi par les cellules ATM (donc les circuits virtuels) correspondent aux plus courts chemins d'IP. Quand un lien tombe en panne, seulement deux switch IP perdent une adjacence, il n'y a pas de routing storm pour la mise à jour de la topologie. Cependant, tous les circuits virtuels interrompus doivent être reroutés.

Cependant, on retrouve des problèmes avec le routage par IP :

- certains liens peuvent être surchargés alors que d'autres sont inutilisés (topologie de poisson)



Par exemple, si le plus court chemin de C à G est CDG, alors tous les flux qui viennent de A et de B vers G utiliseront le chemin CDG. Cela créera de la congestion, et c'est inefficace en sachant que le chemin CEFG est inutilisé.

La solution serait de mettre un poids sur les liens, mais si on assigne comme poids le trafic, on aura des oscillations. On pourrait introduire du load balancing ; OSPF pourrait conserver, pour une même destination, plusieurs routes qui auraient le même coût. Cependant, cela ne suffit pas dans la topologie en forme de poisson.

- les réseaux IP sont uniquement basés sur la destination. Cela peut être contourné en utilisant d'autres champs du header IP, comme l'adresse source ou le type de service (TOS). Par exemple, pour la topologie du poisson, le flux venant de A aurait un chemin différent de celui du flux venant de B. C'est mieux pour gérer le trafic, mais pas scalable car il y a potentiellement beaucoup d'entrées dans la table d'acheminement.

On pourrait aussi spécifier la route dans le paquet IP ; des adresses IP intermédiaires seraient spécifiées dans les options du paquet. Cependant, cela augmente la taille de l'en-tête, et beaucoup d'ISP ne s'en soucient pas.

On peut aussi spécifier des contraintes, par exemple avec une bande passante minimale, ou un délai à ne pas dépasser. Il faudrait pour cela attribuer plusieurs métriques à chaque lien, sans pour autant garantir quoi que ce soit.

On perd donc toutes les features du routage ATM.

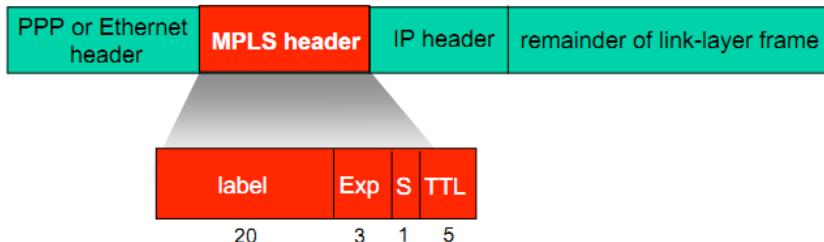
On se retrouve donc avec plusieurs problèmes :

- la nécessité d'avoir des bonnes performances de forwarding (ATM), ou de l'équipement bon marché
- la complexité du mapping IP vers ATM
- des problèmes de scalabilité
- la nécessité d'ajouter des nouvelles fonctionnalités de routing

Le but d'MPLS est de résoudre ces problèmes.

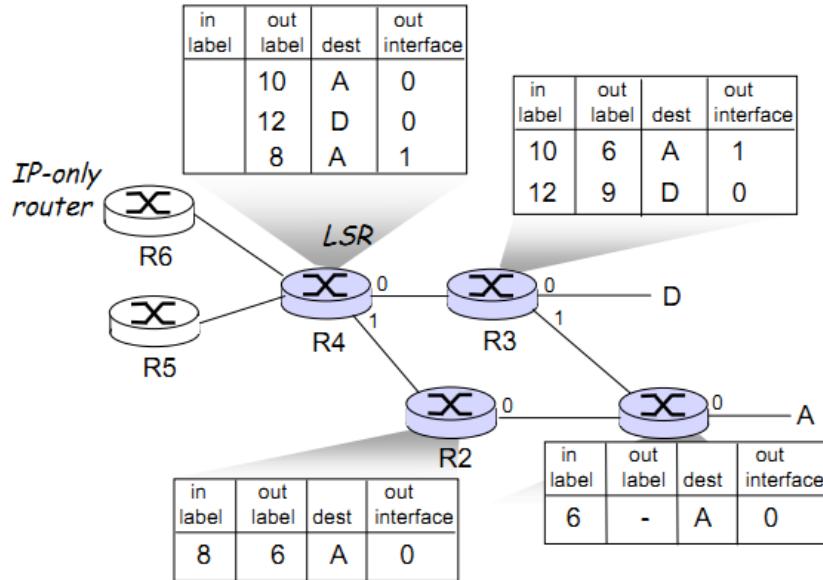
2.3 MPLS - MultiProtocol Label Switching

MPLS est similaire à de l'ATM sous IP, mais est mieux intégré. Le but initial est d'accélérer IP en utilisant des labels de taille fixe à la place d'adresse IP. Ce sont des idées venant de l'approche des circuits virtuels, mais les datagrammes IP gardent toujours leur adresse.



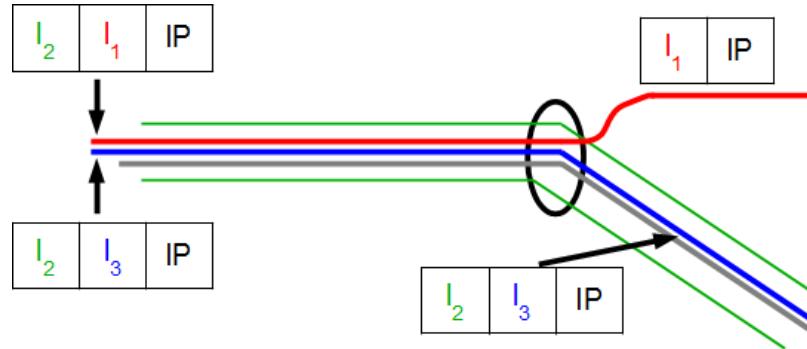
Il ajoute un header supplémentaire dans le paquet, avec un label (nombre). Ce dernier sera utilisé comme un numéro de circuit virtuel. Ces paquets seront traités par des routeurs dit LSR (label-switched router). Ces routeurs ne se basent plus que sur ce label, ils n'inspectent pas l'adresse IP. MPLS définit une table d'acheminement distincte des tables d'acheminement IP.

Un protocole de signalisation est nécessaire pour mettre en place l'état d'acheminement. MPLS permet de l'acheminement sur des chemins qu'IP ne saurait pas faire, par exemple le routage spécifique à l'adresse source. C'est pour cela que MPLS est utilisé pour l'ingénierie du trafic. Il doit cependant coexister avec les routeurs ne gérant qu'IP.



Dans l'exemple, le label de tête (out-label) est le même pour R3 et R2, sinon il y aurait deux entrées dans R1.

Généralement, pour l'acheminement d'un paquet, le routeur d'entrée est appelé Ingress, et celui de sortie Egress. Un label permet de regrouper des circuits, afin de diminuer le nombre d'entrées dans la table d'acheminement. Cela nécessite d'utiliser deux labels : un pour le regroupement des circuits, et un autre pour un circuit individuel.



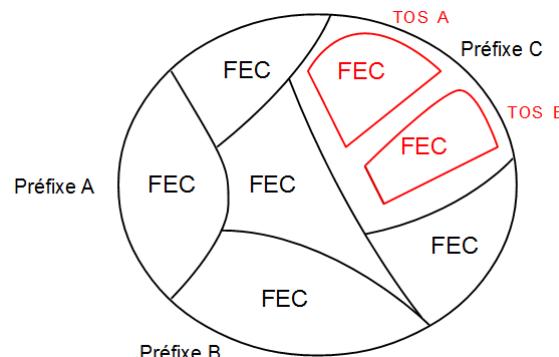
2.3.1 FEC - Forwarding Equivalence Class

Pour rappel, le routing consiste à construire les tables de routage, tandis que le forwarding est l'acheminement de paquets en se basant le contenu des tables de forwarding, dérivées des tables de routage.

On a différents types de forwarding pour IP :

- le forwarding IP unicast : on utilise le préfixe de l'IP de destination (règle du longest prefix match)
- le forwarding IP unicast avec un TOS : on utilise le préfixe de l'IP et la valeur TOS (longest prefix match pour l'adresse, et match exact pour le TOS)
- le forwarding multicast : on utilise les adresses source et de destination et l'interface d'entrée (match exact).

On peut définir, parmi un ensemble de paquets IP, un sous-ensemble (classe) dont les paquets seront acheminés de la même manière. On peut définir ces classes en se basant sur le préfixe IP, mais on peut aussi se baser sur d'autres paradigmes pour subdiviser les classes. Par exemple en utilisant le type de service (TOS).



Chaque classe aura un FEC. Si deux préfixes IP différents (A et B) suivent le même chemin, on peut les grouper.

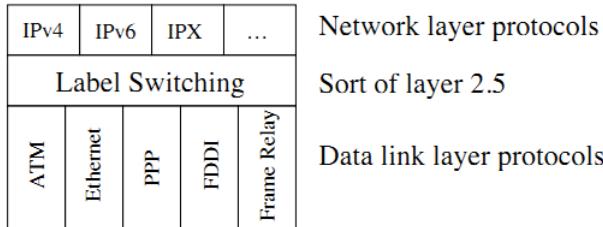
C'est au concepteur du réseau de définir ces classes. C'est permis par MPLS, alors que pour IP on est forcé d'utiliser le forwarding unicast, donc en se basant uniquement sur le préfixe.

2.3.2 Label switching

Tous les paquets ont un label, qui correspond à un FEC qui correspond à une classe, de taille courte et fixe (20 bits). Il agit comme un VCI d'ATM ; les routeurs se basent dessus pour décider de l'interface de sortie. On garde la possibilité d'utiliser aussi l'interface d'entrée. On n'a pas de contraintes sur la granularité du forwarding, cela signifie qu'un label peut être associé à n'importe FEC.

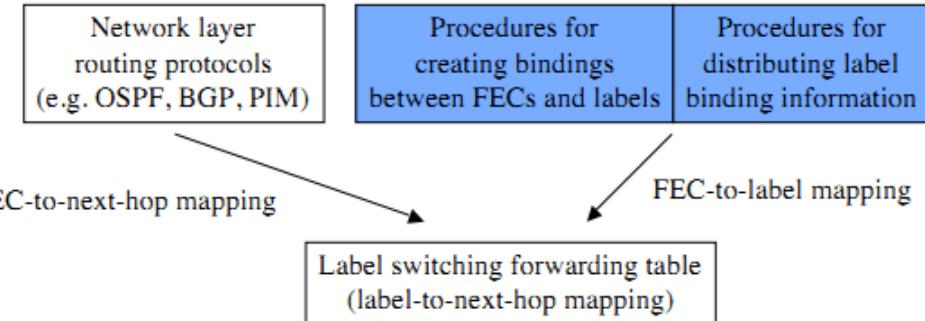
Le label permet de matcher toutes les informations nécessaires pour acheminer le paquet. On a un mapping **label → composants** (à l'inverse d'IP), qui comprend un label de sortie, une interface de sortie, le next-hop, etc. Pour du multicast, il faut prévoir plusieurs composants pour un label, vu que le paquet peut être dupliqué sur plusieurs interfaces. Pour du QoS, un champ spécifierait une liste de sorties.

Ainsi, il n'y a qu'un seul algorithme de forwarding (contrairement à IP qui en a plusieurs, un pour l'unicast, un pour le multicast, un pour l'unicast avec TOS, etc). Les chemins sont appelés LSP (label-switched path).



Le label switching se situe entre les protocoles réseau (IPv4, IPv6, etc) et les protocoles des couches link (ATM, Ethernet, etc). L'avantage est qu'on n'utilise pas IP pour le forwarding, ce qui facilite le changement de technologie (par exemple d'IPv4 à IPv6). Il est complètement indépendant de la couche réseau et peut opérer sur n'importe quelle couche lien, d'où le nom **Multiprotocol Label Switching**. Cependant, on a quand même besoin d'un mécanisme pour obtenir un label.

Le composant de contrôle



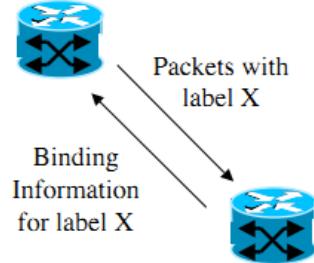
Ce composant est responsable

- de distribuer les informations de routage à travers les LSR
- des procédures pour convertir ces informations en une table de forwarding, c'est-à-dire créer des liens entre les labels et les FECs et les distribuer aux LSRs.

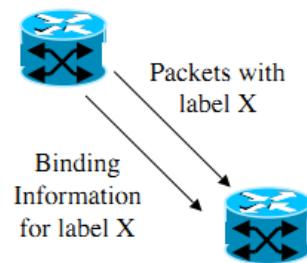
IP n'a ainsi pas besoin des composants bleus.

Binding local et distant Il existe deux manières de créer des liaisons entre des labels et des FECs :

1. le binding local (upstream binding) : un LSR crée la liaison avec un label qui est choisi et assigné localement.
2. le remote binding (downstream binding) : un LSR reçoit un label d'un autre LSR.

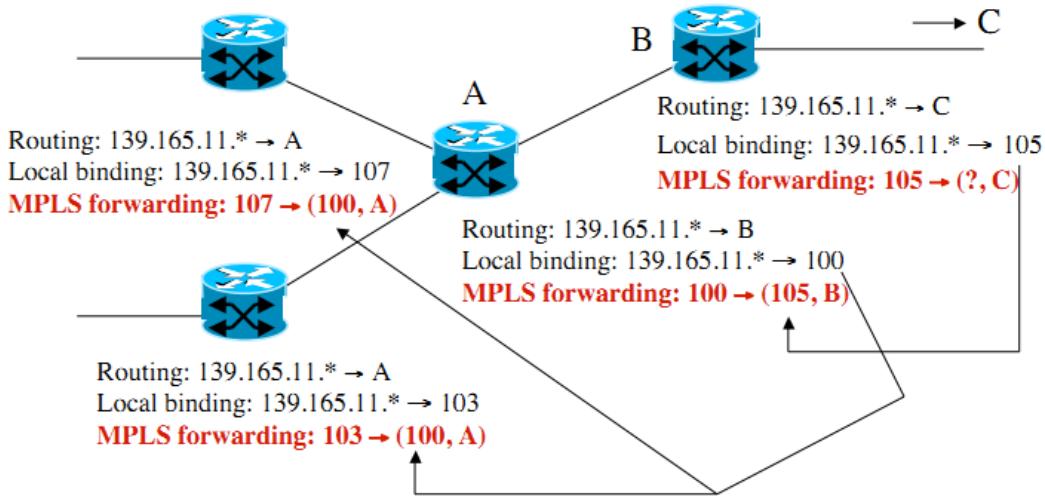


Downstream binding



Upstream binding

Exemple : B va prévenir qu'il a assigné un label à un FEC. En A, on est intéressé car on utilise B comme next hop pour ce FEC, on va donc stocker le label dans la table d'acheminement et l'associer à un autre label et la sortie vers B. De la même manière, A va informer ses voisins à propos de ce mapping, ainsi ils pourront envoyer les paquets en A labellisé par le label associé.



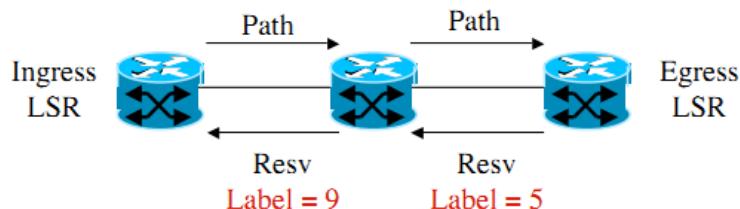
On a ainsi un merge naturel des circuits, par rapport à ATM qui les aurait gardés distincts. Le " ? " de B signifie qu'on a pop le label de tête (\leftrightarrow du tronçon commun) et qu'on utilise celui du circuit virtuel.

On utilise le downstream binding, car l'upstream serait plus compliqué à implémenter. ATM utilise l'upstream ; si c'était de l'upstream pour MPLS, par exemple pour le routeur A, les deux routeurs (non nommés) donneraient des labels différents.

LDP - Label Distribution Protocol C'est un mécanisme FEC-to-label, un plugin et non un substitut ; c'est un protocole qui distribue une liaison FEC-label à travers les LSRs, en utilisant un protocole de routage (comme OSPF).

Si les FEC sont les traditionnels préfixes IP, les LSPs MPLS vont simplement suivre les plus court chemins IP. C'est du label switching, mais naïf.

On utilise le protocole RSVP, qui permet une réservation de ressources et la récupération d'un label.



La source envoie un message PATH à la destination, qui va répondre en utilisant un message RESV tout en distribuant des labels MPLS. La route est déterminée par les tables d'acheminement IP.

Si le message PATH est étendue avec un objet de routage explicite (ERO - Explicit Route Object), le protocole RSVP-TE (pour trafic engineering) peut être utilisé pour établir un LSP qui a été précalculé (source routing). C'est utile quand les routes nécessitent un QoS qui nécessite des chemins particuliers (par exemple avec un minimum de bande passante), ou du load balancing.

Le LSR ingress doit calculer la route, il doit donc connaître la topologie et l'état de QoS de tous les liens. OSPF doit donc être étendu pour supporter l'état QoS d'un lien (par exemple la bande passante disponible). Le LSR ingress va alors calculer le plus court chemin sous contraintes (par exemple Dijkstra sur un graphe réduit).

Chapitre 3

Réseaux mobile et sans-fil

3.1 Introduction

Les technologies sans-fil sont partagées et se retrouvent aux extrémités des réseaux. Il y a deux grands challenges : l'aspect sans-fil et la mobilité, c'est-à-dire la gestion d'un utilisateur mobile qui peut changer de point d'attache à un réseau.

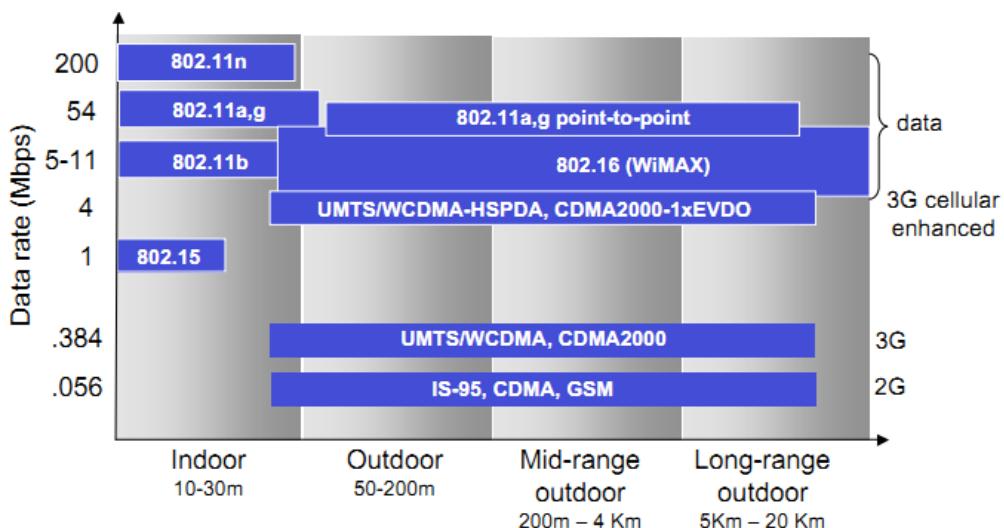
On distingue plusieurs types d'éléments dans un réseau sans-fil :

- Les hôtes sans-fils, des ordinateurs portables, des PDA, des smartphones, etc qui peuvent être stationnaires ou mobiles. La capacité d'être sans-fil n'implique pas nécessairement la mobilité.
- les points d'accès (base station), généralement reliés à un réseau câblé. Il s'agit d'un relais pour envoyer des paquets entre ce réseau et les hôtes sans-fil dans leur champ d'action.
- le lien sans-fil, avec un protocole à accès multiple. Les débits varient selon la distance de transmission. Ce lien connecte les hôtes sans-fil à une base station, mais aussi des éléments de l'infrastructure réseau entre eux. Un lien sans-fil est caractérisé par sa couverture et son débit.

Le fait de se trouver à portée de deux points d'accès est avantageux, car :

- la forme du signal est rarement uniforme, notamment à cause des éventuels obstacles ;
- si un spot est saturé, on peut toujours essayer sur un autre ;
- lorsqu'on entre dans le rayon d'un spot, la connexion avec celui-ci n'est pas instantanée. Le fait d'avoir des rayons qui se superposent permet de ne pas perdre de connectivité.

Il existe de nombreux standards de lien sans-fil.



La norme 802.15 correspond au Bluetooth

La norme 802.11a,g est utilisée en intérieur mais également à l'extérieur, avec une radiation en cône (alors qu'en intérieur, c'est une radiation omnidirectionnelle, où l'énergie utilisée décroît quadratiquement par rapport au rayon).

Il existe deux modes de connexion à un réseau :

- le mode infrastructure : les points d'accès connectent les terminaux à un réseau câblé. Le terme handoff/handover réfère à la mobilité d'un de ces terminaux entre plusieurs base stations.
- le mode ad hoc : il n'y a pas de base stations, les périphériques jouent le rôle de routeurs ; quand ils sont à proximité les uns des autres, ils peuvent communiquer. Si deux périphériques sont éloignés, ils peuvent passer

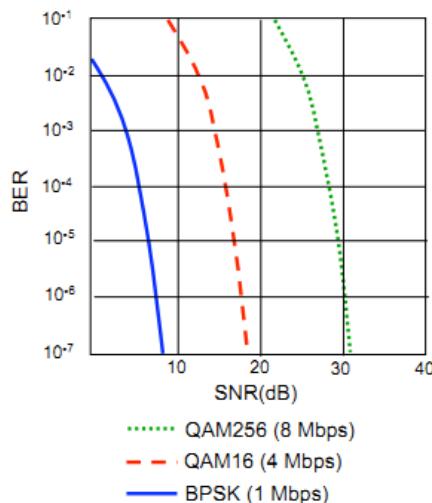
par des périphériques intermédiaires pour communiquer (MANET : mobile adhoc network). L'avantage est qu'il n'est pas nécessaire d'avoir une infrastructure pour supporter le réseau.

	single hop	multiple hops
infrastructure (e.g., APs)	host connects to base station (WiFi, WiMAX, cellular) which connects to larger Internet	host may have to relay through several wireless nodes to connect to larger Internet: <i>mesh net</i>
no infrastructure	no base station, no connection to larger Internet (Bluetooth, ad hoc nets)	no base station, no connection to larger Internet. May have to relay to reach another given wireless node (MANET, VANET)

Grandes différences entre un réseau sans-fil et un réseau câblé :

- la force du signal décroît de manière quadratique en fonction de la distance, alors que l'atténuation d'un câble est linéaire ;
- il y a des interférences avec d'autres sources ; certaines fréquences standardisées sont partagées avec d'autres périphériques (par exemple les téléphones) ;
- la propagation du signal fait qu'il peut se refléter et arriver décalé en un point. Un signal interfère avec lui-même en plus d'interférer avec d'autres objets.

Ce sont ces caractéristiques qui rendent une communication plus compliquée.

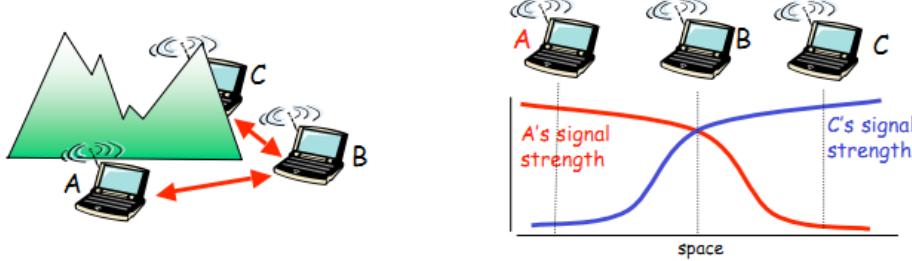


Un rapport signal/bruit grand signifie qu'il est facile d'extraire un signal du bruit ; idéalement, il faut qu'il soit le plus grand possible, mais cela implique une plus grande puissance de transmission. Pour une couche physique donnée, plus on augmente la puissance (le signal), plus le rapport signal/bruit augmente et plus le BER (bit error rate, taux d'erreur) diminue.

Par contre, étant donné un rapport signal/bruit, il faut choisir une couche physique que respecte certains pré-requis en terme de BER, tout en ayant la plus grande bande passante. De plus, le rapport signal/bruit peut varier avec la mobilité, la couche physique doit s'adapter dynamiquement.

Une solution est de passer d'un protocole à l'autre afin d'avoir un taux d'erreur faible (passer du rouge au bleu par exemple), mais en baissant le débit.

3.1.1 Problème d'un terminal caché



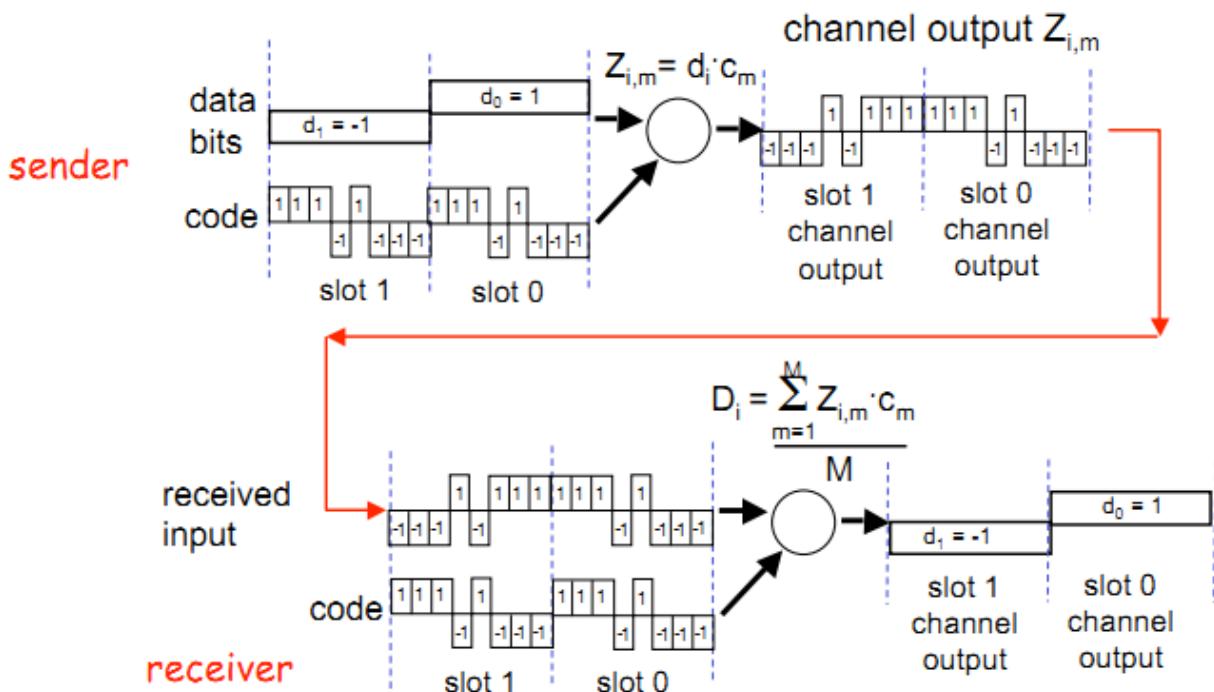
Avec un réseau câblé, tous les hôtes sont accessibles (partage d'un bus par exemple) et peuvent s'entendre. Avec un réseau sans-fil, ce n'est pas le cas, c'est le problème du terminal caché : un terminal peut ne pas être accessible et accéder à certains terminaux d'un réseau, à cause de l'atténuation du signal.

Cela s'ajoute au problème de collision ; le CSMA n'est pas utilisable, car si A et B communiquent, C ne le saura pas et communiquera à B quand même. À cause de ce problème de terminal caché, la détection de collision est beaucoup plus difficile.

3.1.2 CDMA - Code division multiple access

On avait déjà TDM (time division multiplexing) et FDM (frequency division multiplexing). Avec CDM, tout le monde parle en même temps à la même fréquence, mais on peut filtrer tout le bruit pour isoler une personne en particulier.

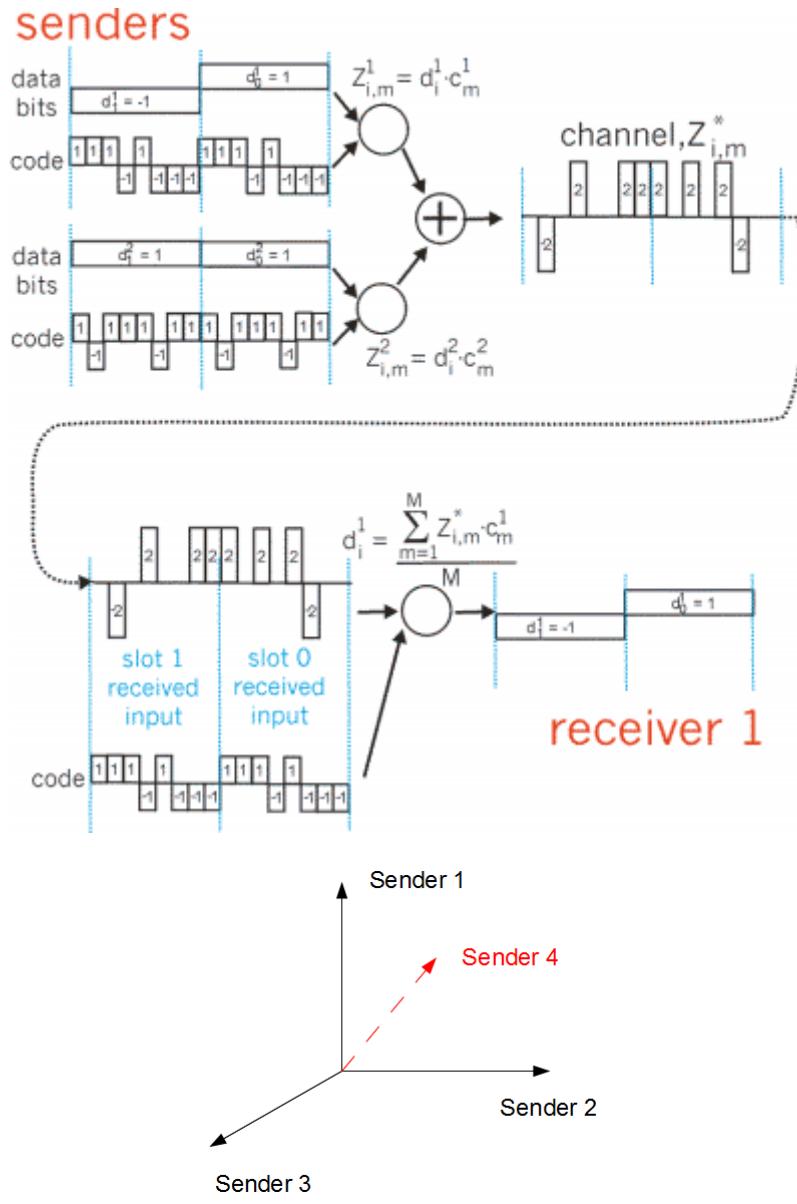
Un pattern est assigné à chaque interlocuteur, et chaque fois qu'il y a une trame à envoyer, les bits sont multipliés par ce pattern. 0 est encodé en -1, 1 reste 1. Le produit est envoyé, et peut être décodé avec le pattern par un produit scalaire.



Cela permet ainsi la coexistence de plusieurs utilisateurs sur la même gamme de fréquences, qui transmettent simultanément avec un minimum d'interférences.

Les codes doivent être soigneusement choisis ; il ne faut pas que deux senders aient le même code. De plus, les codes doivent être orthogonaux, afin que leur produit scalaire soit nul. Cependant, le nombre de vecteurs de bits orthogonaux est égal à la dimension (8 bits donne 8 vecteurs orthogonaux). Pour accepter plus de personnes, il faut augmenter la dimension.

Les signaux s'additionnent quand ils sont envoyés et qu'ils interfèrent entre eux. Pour décoder, il suffit de projeter un vecteur sur une direction. Si les vecteurs n'étaient pas orthogonaux, on ne pourrait pas le faire.



3.2 Réseaux locaux sans-fil et IEEE 802.11

3.2.1 Normes sans-fil

Il existe plusieurs normes pour le 802.11 :

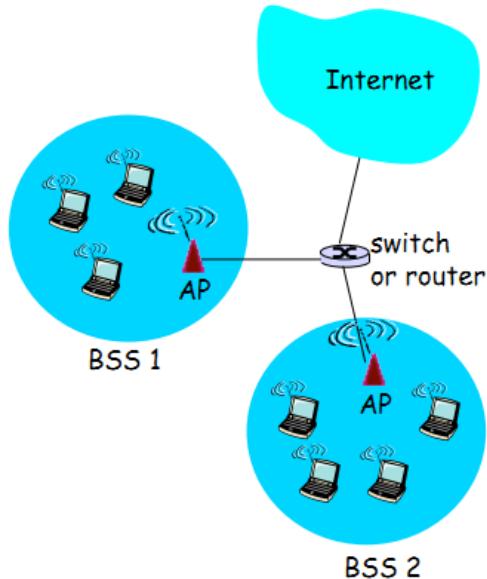
- 802.11b : utilisation d'un spectre de fréquences publiques (2.4-2.5GHz), jusqu'à 11Mbps, DSSS (direct sequence spread spectrum) dans la couche physique
- 802.11a : 5-6GHz, jusqu'à 54Mbps
- 802.11g : 2.4-2.5GHz, jusqu'à 54MBps
- 802.11n : 2.4-2.5GHz, jusqu'à 200Mbps, avec plusieurs antennes et MIMO

Tout ces protocoles utilisent CSMA/CA pour les accès multiples, et tous possèdent une version avec base-station et une version ad-hoc.

3.2.2 Architecture LAN

Une base station est nommé point d'accès ; c'est un élément opérant à la couche 2.

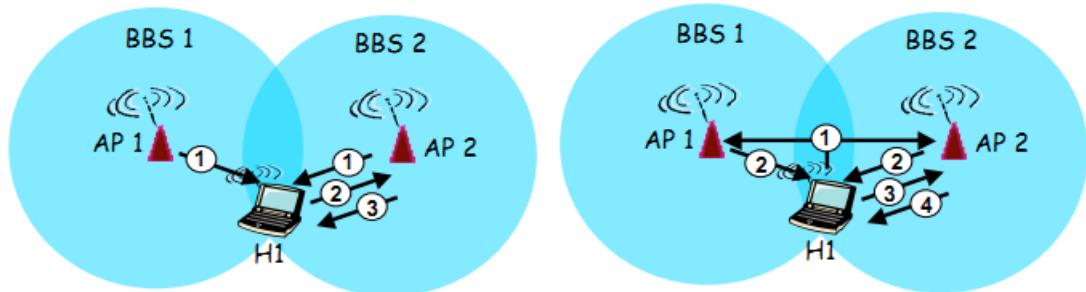
Un BSS (Basic Service Set), ou "cellule" est une infrastructure qui contient des hôtes sans fil et un point d'accès (si on est en mode ad hoc, il n'y a que des hôtes).



Pour une norme, la bande de fréquences est divisée en plusieurs canaux qui se chevauchent. Cela fait perdre de la bande passante (théorème de Shannon), mais cela permet de choisir des sous-canaux afin de ne pas subir des interférences venant de périphériques environnants. Ainsi, avant d'installer un point d'accès, il faut scanner l'environnement pour déterminer les sous-canaux. De plus, cela permet, si des points d'accès sont proches, d'éviter des interférences en opérant sur des sous-canaux distincts.

Un hôte doit s'associer à un point d'accès ; les points d'accès envoient des trames qui signalent leur présence (beacon frame), qui contiennent leurs SSID (nom) et adresse MAC.

Un hôte scanne et écoute ainsi les sous-canaux, à la recherche de ces trames. Ensuite il sélectionne un point d'accès, s'authentifie si nécessaire, et va utiliser DHCP pour récupérer une adresse IP sur le réseau du point d'accès.



Il y a deux types de scan de beacons :

– le scan passif :

1. les beacon frames sont envoyées depuis les points d'accès ;
2. un hôte sélectionne un point d'accès et envoie une frame de requête (Association Request frame) ;
3. le point d'accès renvoie une réponse (Association Response frame).

– le scan actif :

1. l'hôte envoie des trames de requête en broadcast (Probe Request frame) ;
2. des réponses sont envoyées par les points d'accès ;
3. l'hôte sélectionne un point d'accès et envoie une frame de requête (Association Request frame) ;
4. le point d'accès répond à l'hôte (Association Response frame).

Le scan actif est plus rapide, et est nécessaire pour les silent access points, des points d'accès qui n'envoient pas de beacon, pour plus de sécurité. Le mode passif est quand même utile, pour économiser de l'énergie par exemple.

3.2.3 CSMA/CA

CSMA est utilisable, mais pas pour détecter et éviter les collisions. CSMA/CD n'est pas suffisant non plus, car on pourrait ne pas entendre des hôtes qui sont éloignés (signal trop faible).

On définit donc le CSMA/CA (pour collision avoidance), qui va permettre d'éviter des collisions.

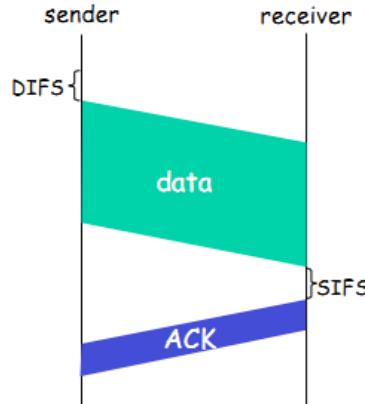
On définit un DIFS comme un slot durant lequel on écoute le canal. On a le fonctionnement suivant :

– pour l'expéditeur :

1. si le canal n'est pas occupé pendant un DIFS, alors on transmet toute la frame (pas de détection de collision)
2. si le canal est occupé, on attend pendant un temps choisi aléatoirement, et qui décroît quand le canal est libre.

On transmet quand le temps expire. S'il n'y a pas d'ACK, on augmente l'intervalle de temps choisi aléatoirement et on recommence.

– pour le récepteur : si la frame est bonne, on envoie un ACK après un SIFS ; cet ACK est nécessaire à cause du problème de terminal caché).



La principale différence est l'ACK envoyé par le récepteur, pour assurer qu'il n'y a pas eu de collision. Lorsqu'on le reçoit, on sait qu'il n'y a pas eu de collision ni d'erreur binaire. Généralement le délai est court comparé aux récupérations d'erreur des couches supérieures (TCP).

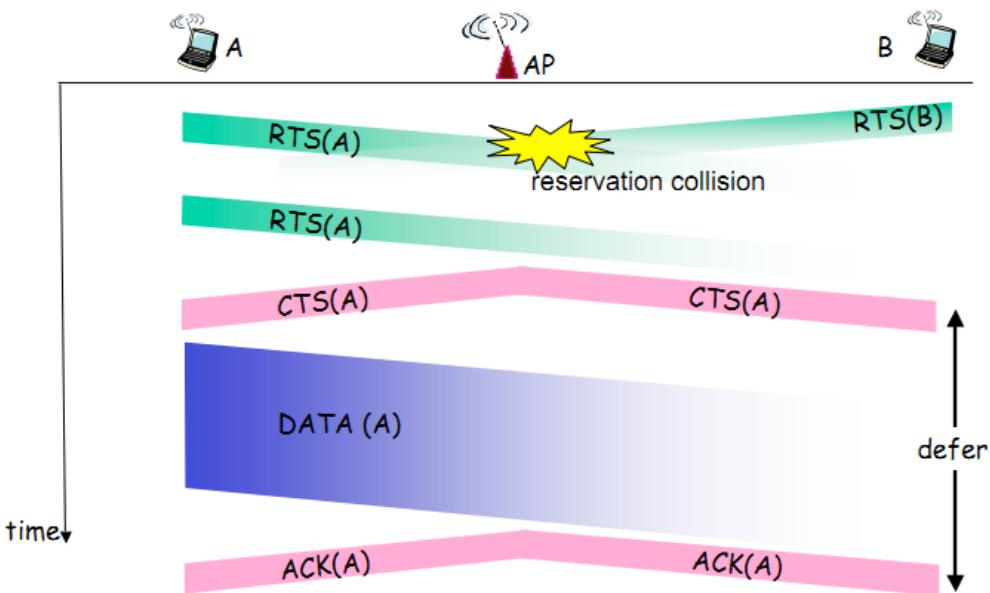
De plus, il n'y a jamais qu'une entité qui communique à la fois ; le canal est half-duplex. Cela donne un protocole robuste, même si le mécanisme est simple.

Pour Ethernet, quand on commence à transmettre, il peut y avoir une collision (à cause du temps de propagation) uniquement au début, mais vu que tout le monde écoute, il n'y en aura pas après $2 \times$ le temps de propagation. Du coup, il vaut mieux travailler avec des trames relativement grandes.

Dans le scénario actuel, on peut ne pas entendre le transfert, quelqu'un pourrait donc commencer à transmettre alors que le canal est occupé. Le risque de collision est donc tout le temps présent, et plus la trame à transmettre sera grande, plus il y a de chance d'avoir une collision. Du coup, il vaut mieux que les trames ne soient pas trop grandes.

De plus, les erreurs binaires sont plus fréquentes ; les probabilités sont petites pour des petits paquets, mais sont proportionnelles à la taille du paquet. Du coup, s'il y en a une dans une grosse trame, il faudra la renvoyer entièrement.

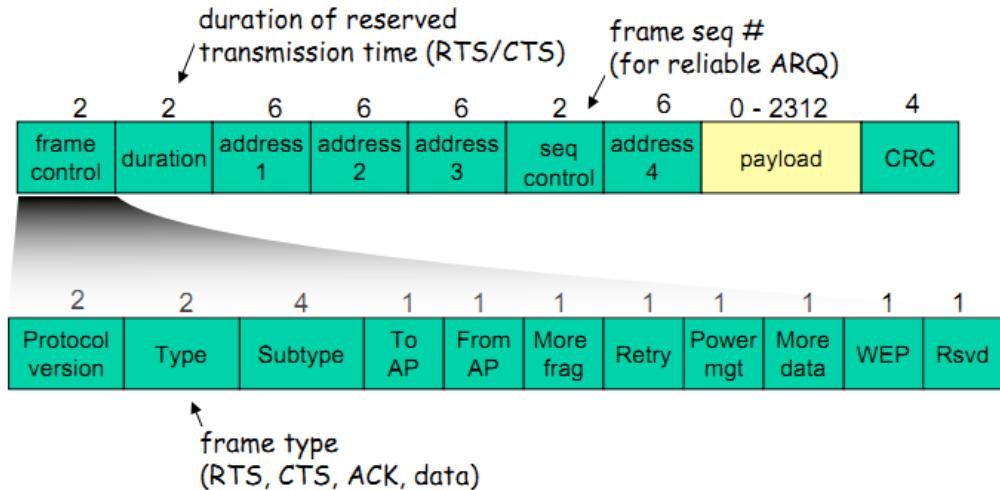
Pour Ethernet, il y a une réservation de ressources implicite avec le double du temps de propagation. Ici, l'idée serait de faire une réservation explicite.



Le transmetteur va envoyer une requête d'envoi (RTS, request-to-send) avec un petit paquet. Les confirmations (CTS) sont envoyées par le point d'accès en broadcast, afin que tous les hôtes sachent qu'il y a eu réservation. Même les hôtes qui sont éloignés le sauront, sinon ils ne seraient pas connectés au point d'accès, donc il ne saurait pas y avoir de collision.

Le CTS contient la taille des données à envoyer (communiquées dans le RTS à la base). Du coup, si un hôte ne reçoit pas l'ACK (comme B), il pourrait calculer le temps qu'il devra attendre en se basant sur cette taille.

3.2.4 Adressage



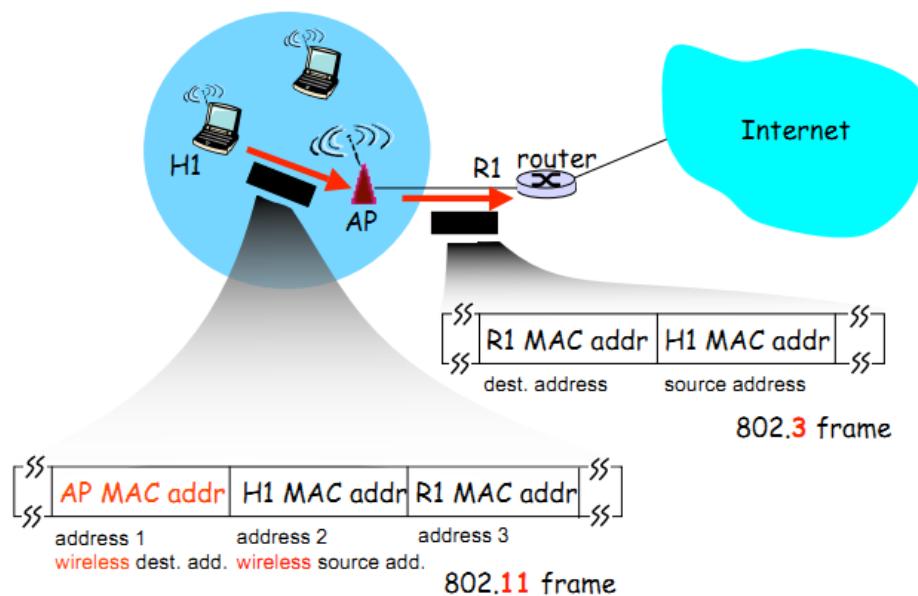
Une trame contient 4 adresses, soit deux adresses supplémentaires aux adresses MAC source et de destination :

- Adresse 1 : l'adresse MAC de l'hôte sans-fil ou du point d'accès, au cas où il y aurait plusieurs points d'accès à portée de l'hôte ; on n'enverra la trame qu'à un seul point d'accès.

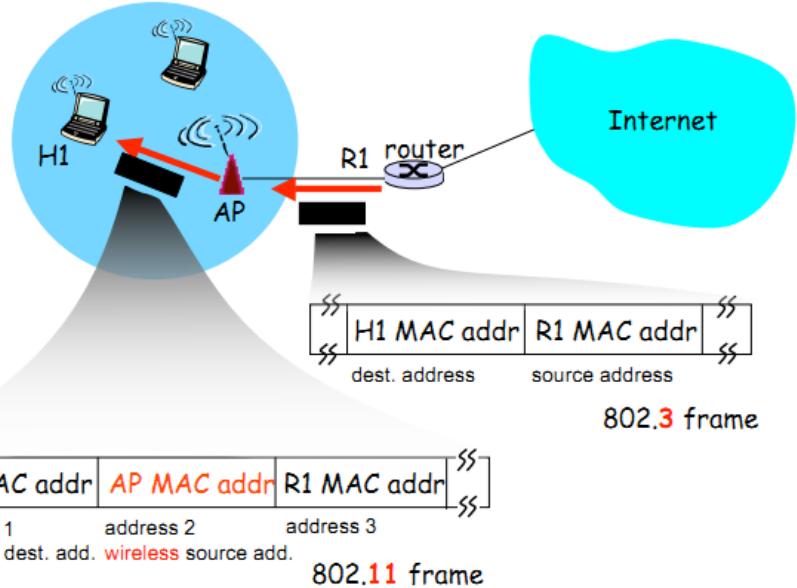
L'adresse du routeur est utilisée par le point d'accès au cas où il serait relié à plusieurs routeurs.

- Adresse 2 : l'adresse MAC source, de l'hôte ou du point d'accès qui transmet la trame ;
- Adresse 3 : l'adresse MAC de destination, celle de l'interface du routeur auquel le point d'accès est attaché
- Adresse 4 : utilisée pour le mode ad hoc.

Envoi d'une trame par l'hôte :

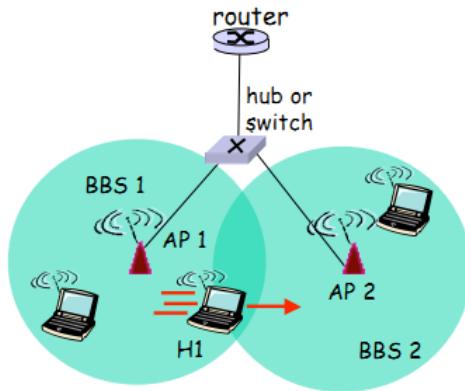


Réception d'une trame par l'hôte :



C'est différent d'Ethernet dans le sens où le point d'accès n'est pas transparent pour les hôtes sans-fil, contrairement à un switch Ethernet. Par contre, il est transparent pour les routeurs, il est considéré comme un bridge de niveau 2, au même titre qu'un switch.

Au sein d'un même subnet, on peut avoir un problème lorsqu'il y a de la mobilité.

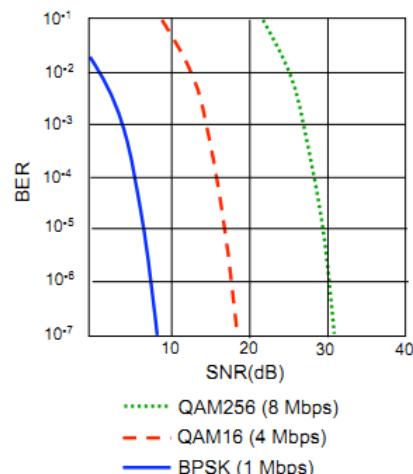


En changeant de point d'accès, l'hôte va conserver la même adresse IP (vu qu'il est dans le même subnet). Le problème est qu'en passant d'un à l'autre, le switch doit savoir vers lequel se diriger.

La solution est le self-learning : le switch va voir les frames de l'hôte, et se "rappellera" du port qu'il utilise.

Fonctionnalités avancées

Adaptation du débit En se déplaçant, le taux de transmission varie, de même que le rapport signal/bruit (SNR). Plus on s'éloigne d'un point d'accès, plus le SNR diminue et plus le BER augmente. Quand il est trop grand, la solution est de passer à un début plus faible, mais avec un BER plus petit.

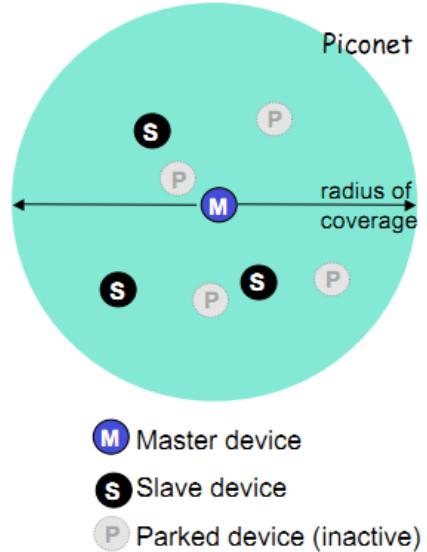


Power management Un noeud peut signifier à un point d'accès qu'il se met en veille. Ainsi, ce dernier sait qu'il ne faut pas transmettre de frames à cet hôte particulier. Cet hôte se réveillera avant la prochaine beacon frame.

Une beacon frame contient la liste des périphériques mobiles pour lesquels il y a des frames en attente de transmission. Ainsi, un noeud se réveillera s'il y a des frames pour lui, sinon il se remet en veille jusqu'à la prochaine beacon frame.

Cela permet d'économiser de l'énergie ; un mobile peut être en veille durant 99% du temps.

3.2.5 802.15 : Personal Area Network - PAN



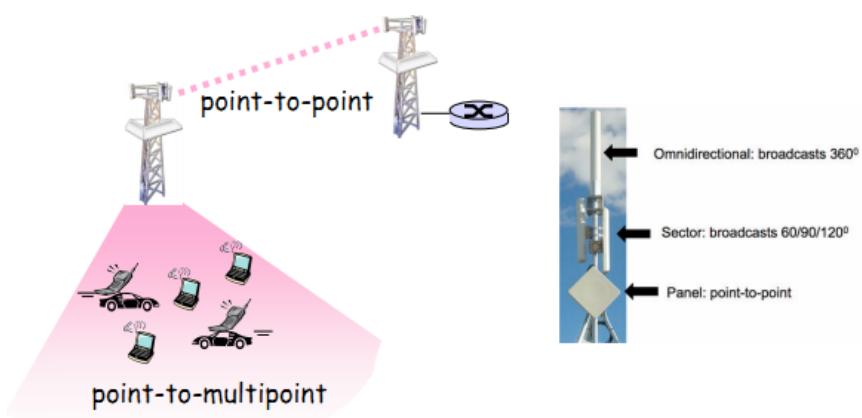
Un PAN est un petit réseau, généralement de pas plus de 10m de diamètre. Il est de type ad hoc, avec un système de maître/esclave. Il permet de se passer de câble (pour une souris, un clavier, un casque, etc).

Avec le FHSS (Frequency Hopping), la bande de fréquences est divisée en de nombreuses sous-intervalles de fréquences. Les périphériques passent d'une fréquence à l'autre très rapidement ; quelques bits sont passé pour chaque fréquence avant de passer à une autre. Cela permet d'éviter le bruit et les interférences sur une gamme de fréquences.

3.2.6 802.16 : WiMAX

Il est similaire au 802.11 et au réseau cellulaire et utilise le modèle base-station :

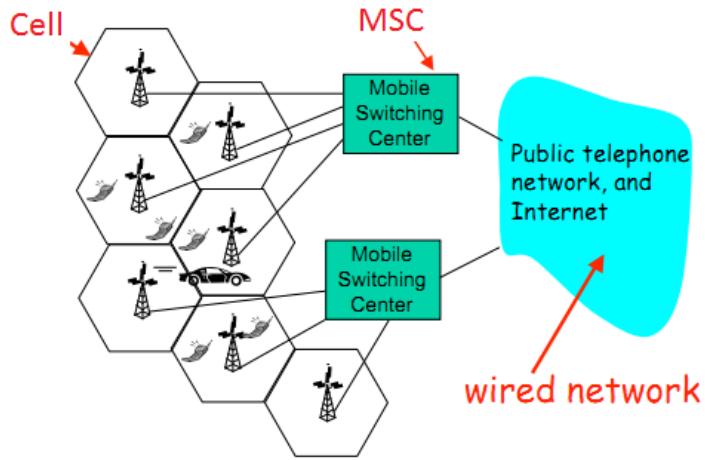
- les transmissions entre les hôtes et la base station se fait au moyen d'une antenne omnidirectionnelle ;
- les transmissions entre des base stations se fait avec une antenne permettant le point-to-point.



Cela permet d'avoir des ranges jusqu'à 10km, avec un débit d'environ 14Mbps.

3.3 Cellular Internet Access

Le nom "cellular" vient de la forme des couvertures des antennes.



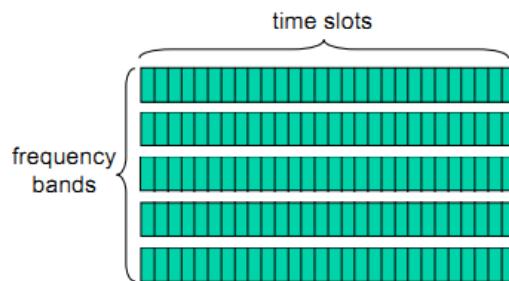
Un MSC (mobile switching center) est sorte de routeur, mais au niveau téléphonique. Il connecte les cellules à un réseau de plus grande ampleur. Il gère la mise en place des appels et la mobilité.

Une cellule est une petite région géographique, composée d'une base station, d'utilisateurs mobiles (attachés à la base station) et d'une interface sans support physique entre la base station et les utilisateurs.

Lorsqu'on veut passer un appel, un périphérique utilise un canal dit ouvert. S'il n'y a pas de collision, un canal sera réservé et communiqué au périphérique.

Il y a plusieurs manières de partager le spectre radio d'un périphérique mobile à une base station :

- on divise la bande de fréquences en sous-intervalles, qui sont ensuite divisés en slots "temporels" ; c'est un mélange de TDMA et de FDMA. La technologie GSM utilise cet manière de partager le canal, et est présente plutôt en Europe.



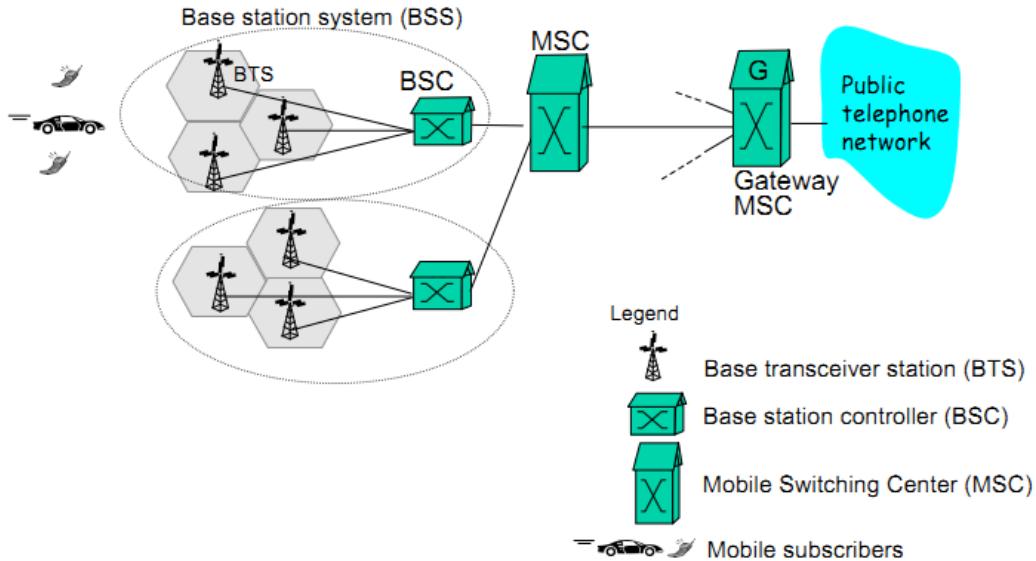
- CDMA, utilisé aux USA

3.3.1 Standards

La première génération de systèmes cellulaires était la 1G, uniquement analogique, et qui est à présent éteinte.

3.3.2 2G

La 2G a introduit l'aspect digital, mais ne concernait toujours que la voix.

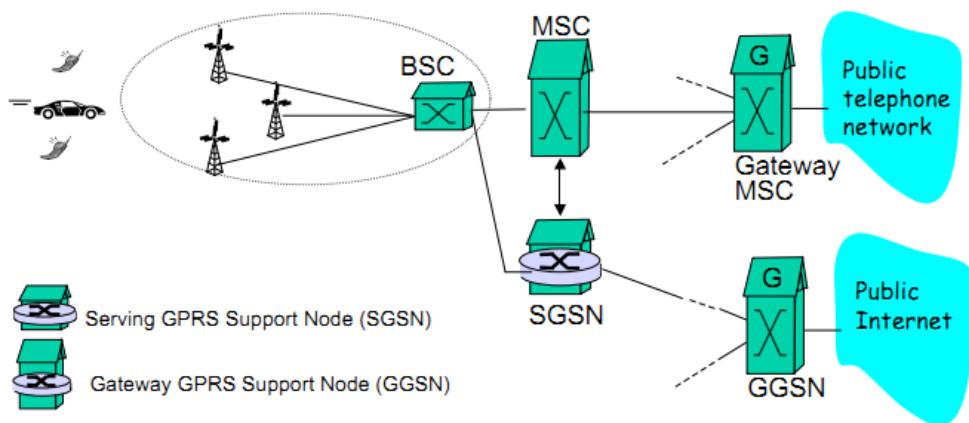


3.3.3 2.5G

Après la seconde génération (2G), il y eu une génération intermédiaire (2.5G), car la 3G a été standardisée, mais n'a pu être mise en place rapidement. La 2.5G est destinée à ceux qui n'ont pu attendre, la technologie GSM étant très lente (13K), ce qui la limitait à la voix.

La technologie GPRS est une évolution du GSM ; au lieu d'utiliser un seul canal, elle va en utiliser plusieurs. Vu que des paquets sont généralement envoyés en burst (et non pas un flux constant), il n'est pas nécessaire de réserver complètement de nombreux canaux, ce qui n'est pas compatible avec le GSM. Pour parer à cela, on va réserver des canaux pour les données et d'autres pour la voix. Ils sont ainsi alloués dynamiquement aux périphériques GPRS en fonction des besoins.

EDGE a été aussi basé sur GSM, un peu plus tard que GPRS, avec un débit plus élevé. La technologie CDMA américaine a également évolué.

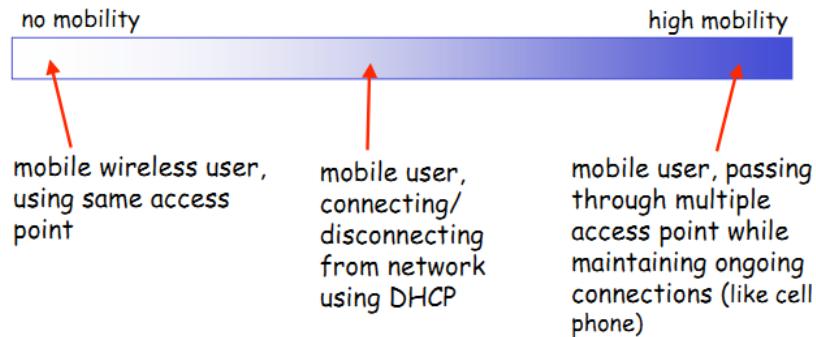


La voix utilise le même chemin (via BSC et MSC), mais les données sont détournées vers un SGSN. Cet installation est bon marché à mettre en place, les structures les plus chères (antennes) restent inchangées et seul quelques périphériques sont à installer à côté.

3.3.4 3G

La 3G, appelée aussi UMTS (universal mobile telecommunications service) en Europe, possède de bonnes propriétés pour les données et la voix. Elle combine le FDMA/TDMA et le CDMA : chaque slot temporel est assigné à plusieurs périphériques. Le CDMA-2000 américain combine aussi plusieurs technologies pour également augmenter les débits.

3.4 Adressage et routage vers des utilisateurs mobiles



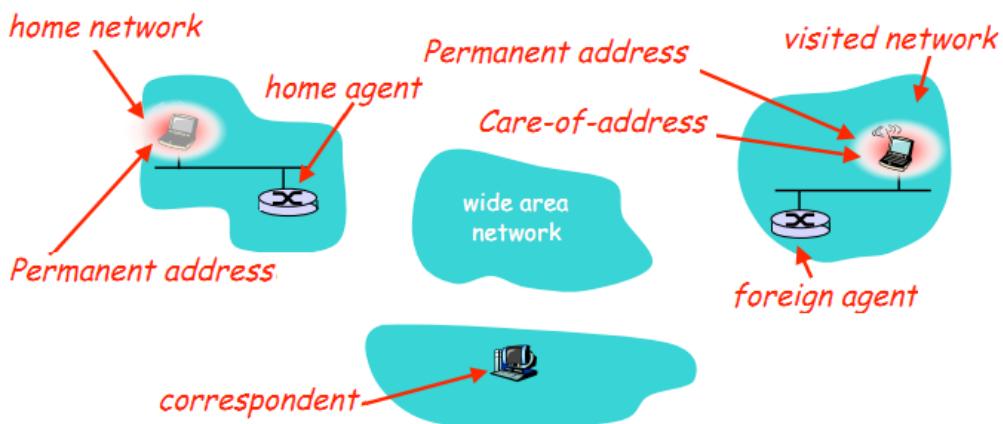
N'est pas considéré comme mobile le fait de se déplacer tout en gardant un contact avec le même point d'accès.

Le fait d'être nomade désigne principalement la connexion/déconnexion à un réseau, ce qui implique notamment un changement d'IP. Cependant, grâce à DHCP, il n'y a pas vraiment de mécanisme à mettre en place.

La "vrai" mobilité signifie passer d'un point d'accès à un autre tout en maintenant la connexion.

Terminologie :

- home network : le domicile permanent du mobile ;
- permanent address : elle ne change pas, quelque soit le réseau dans lequel on se trouve. Cela implique qu'être dans un réseau qui n'est pas dans un bon subnet fera qu'on ne sera pas joignable. Il est donc nécessaire d'avoir une autre adresse (care-of-address)
- home agent : l'entité qui offre les fonctionnalités de mobilité au mobile, quand il est à porté.
- foreign agent : l'agent d'un réseau étranger
- care-of-address : adresse utilisée dans un réseau étranger ; selon la technologie, assignnée au routeur ou au périphérique.



La recherche du correspondant peut donner des solutions extrêmes (recherche exhaustive, contacter des périphériques en amont du correspondant, attendre que le correspondant indique sa position, etc), elles ne sont pas scalables. Par exemple, pour la 3ème solution, tous les contacts potentiels doivent faire savoir où ils sont.

Il y a deux approches pour gérer la mobilité : délaisser la gestion aux routeurs ou aux systèmes d'extrémité.

3.4.1 Mobilité par les routeurs

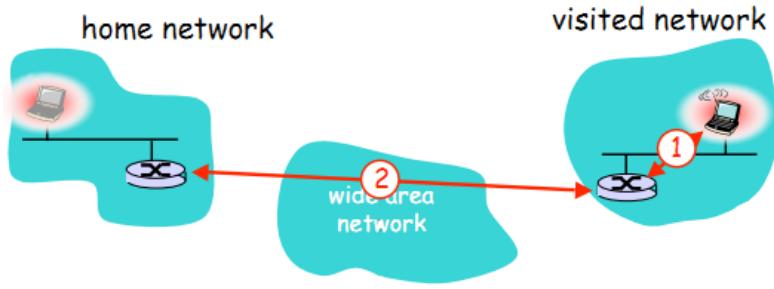
Un périphérique garde une adresse permanente, même dans un réseau étranger. Il y aura deux entrées dans la table d'acheminement des routeurs intermédiaires : une pour le subnet home, une autre pour le périphérique mobile, que l'on a "volée" du réseau d'origine et qu'on a temporairement donnée au réseau étranger. Ainsi, le routeur étranger communiquera, avec BGP, qu'il a accès au subnet étranger ET l'adresse individuelle du périphérique mobile.

Cela fonctionne grâce à la règle de matching du plus long préfixe, de plus on ne change rien d'autre. Enfin, cela permet d'éviter de modifier les systèmes d'extrémité.

Cependant, s'il y a des millions de périphériques mobiles, il y aura autant d'entrées dans les tables d'acheminement.

3.4.2 Mobilité par les systèmes d'extrémité

Une première étape est de s'arranger pour que le home agent sache où se trouve le mobile, et que le foreign agent sache qu'il y a un mobile dans le subnet.

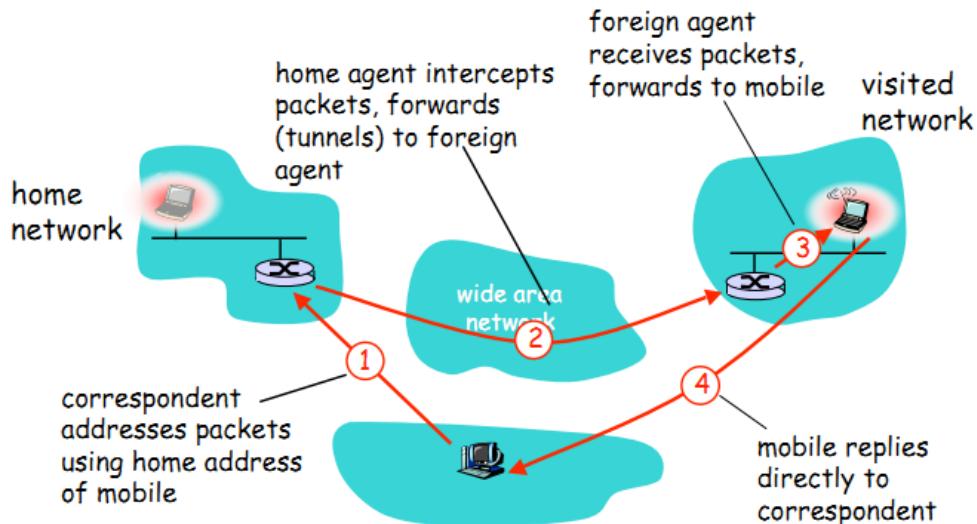


1. Le périphérique mobile communique au routeur étranger qu'il est entré dans son réseau, ainsi que son adresse permanente. Il découvrira alors le réseau d'origine.
2. Ce routeur étranger va contacter l'home agent pour signifier qu'il peut atteindre le périphérique mobile. L'home agent saura ainsi comment atteindre le périphérique mobile.

Après cette enregistrement, il y a deux manières de gérer la mobilité : par le routage indirect et direct.

Routage indirect

Avec le routage indirect, il y a une communication entre le correspondant et le mobile à travers l'home agent, qui forward les données.



Un mobile dispose donc de deux adresses : une adresse permanente (utilisée par le correspondant et qui est transparente, quelque soit la localisation), et une care-of-address, utilisée par l'home agent pour forwarder les datagrammes.

Un paquet dont l'adresse source est celle du périphérique mobile sera ainsi envoyé au home agent, qui le transmettra au routeur étranger en l'empaquetant (tunneling). Le routeur étranger aura une entrée particulière pour le périphérique mobile. Il connaîtra son adresse MAC (grâce au contact qu'il aura eu avec lui au préalable), il pourra l'envoyer.

Si le périphérique mobile doit répondre, il va répondre en utilisant le routeur étranger mais en utilisant comme adresse source son adresse permanente.

Il peut y avoir des problèmes, car le routeur étranger pourrait croire qu'il s'agit d'une source pour une attaque (spoofing), et pourrait ne pas transmettre de réponse. Mais on pourrait mettre une exception comme quoi on autorise les paquets envoyés à partir d'adresses IP permanentes.

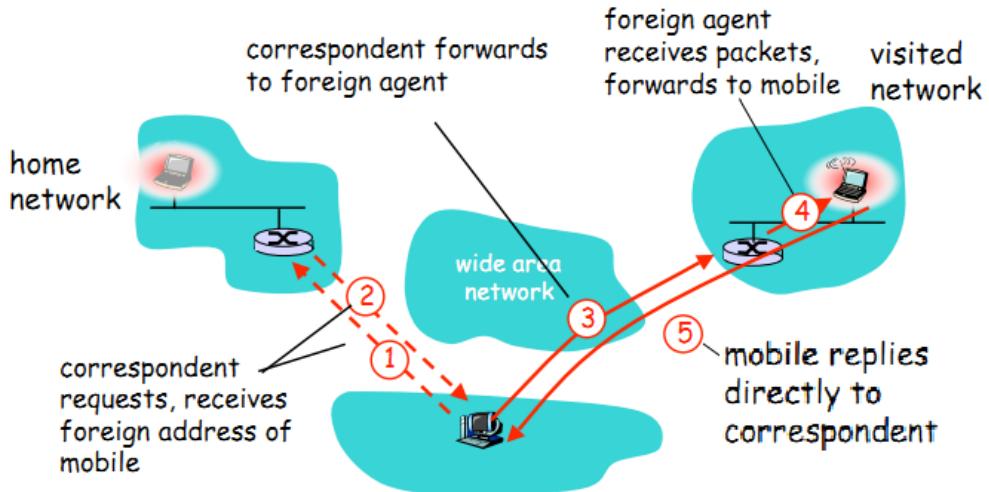
Ce système est transparent pour le périphérique correspondant, car les adresses restent les mêmes. Cependant, il s'agit d'un routing triangulaire, ce n'est pas efficace, il pourrait y avoir un énorme détour alors que le correspondant est proche.

Un avantage est que si le périphérique se déplace, il y aura un nouveau subnet avec un nouveau routeur étranger, mais tout restera transparent pour le correspondant. Une perte de paquets peut survenir, leur gestion revient à TCP/UDP.

Supposons que le mobile change de réseau. Il s'enregistrera alors avec le nouveau foreign agent, qui va contacter le home agent. Ce dernier va mettre à jour la care-of-address pour le mobile, et les paquets continueront à être forwardés, en utilisant la nouvelle adresse. Les connexions en cours sont ainsi facilement maintenues.

Routage direct

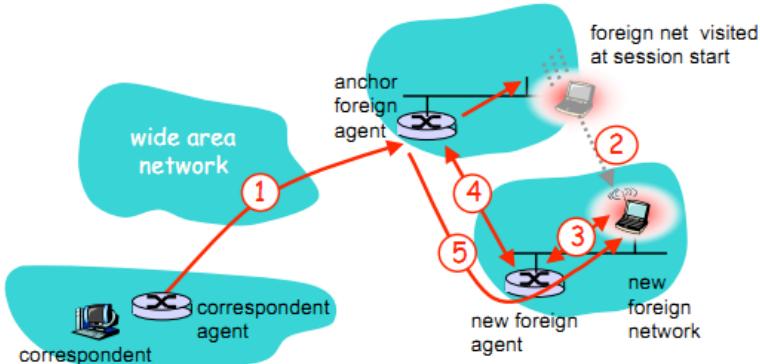
On va sacrifier la transparence pour obtenir un adressage non triangulaire, et donc plus efficace. Le dialogue entre le périphérique mobile et le routeur étranger, et entre le routeur étranger et le home agent est préservé. Cependant, le correspondant "demanderà" au home agent si le périphérique mobile est accessible, sinon de communiquer l'adresse du routeur étranger.



Ce routage permet de supprimer le problème de routage triangulaire, mais on perd la transparence.

Quand un périphérique mobile change de subnet, le nouveau routeur étranger va, en plus de contacter le home agent, contacter l'ancien routeur étranger pour que le trafic soit redirigé vers le nouveau subnet. Le problème est qu'il peut y avoir de nombreux changements de réseau, et donc de routeurs étrangers qui exécutent des redirections.

On va pour cela maintenir le premier routeur étranger (anchor foreign agent), le correspondant maintiendra le contact avec lui. Ensuite, les routeurs étrangers intermédiaires (entre le premier routeur étranger et le périphérique mobile) seront retirés de la chaîne.



Le routage direct n'implique ainsi plus le home agent ; ce dernier n'est contacté qu'une seule fois.

Chacun de ces types de routage a des avantages et des inconvénients, en terme de transparence et d'efficacité.

3.5 Mobile IP

La technologie mobile IP est définie par une RFC, et utilise

- la découverte d'agent
- l'enregistrement avec l'home agent, et
- le routage indirect.

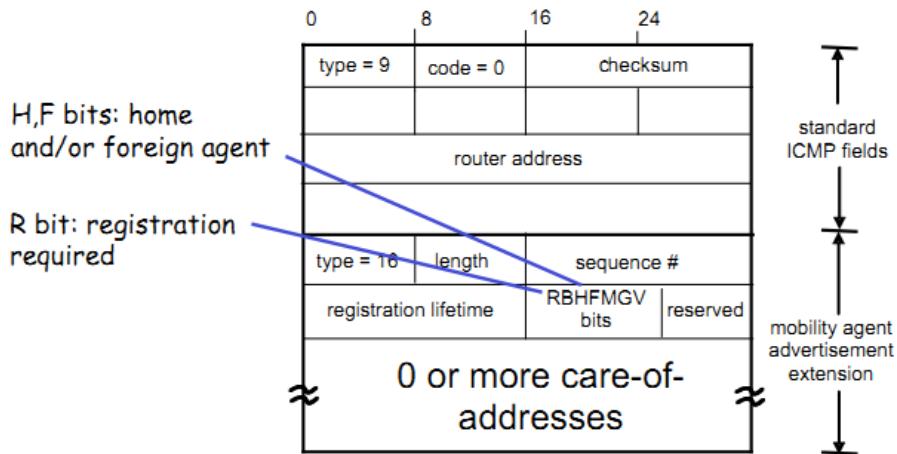
3.5.1 Découverte d'agent

Lorsqu'un périphérique mobile entre dans un nouveau réseau (qu'il soit étranger ou home), il doit connaître l'identité de l'agent. Cela peut se faire de deux manières :

- par une initiative de l'agent, ou
- par une sollicitation de l'agent.

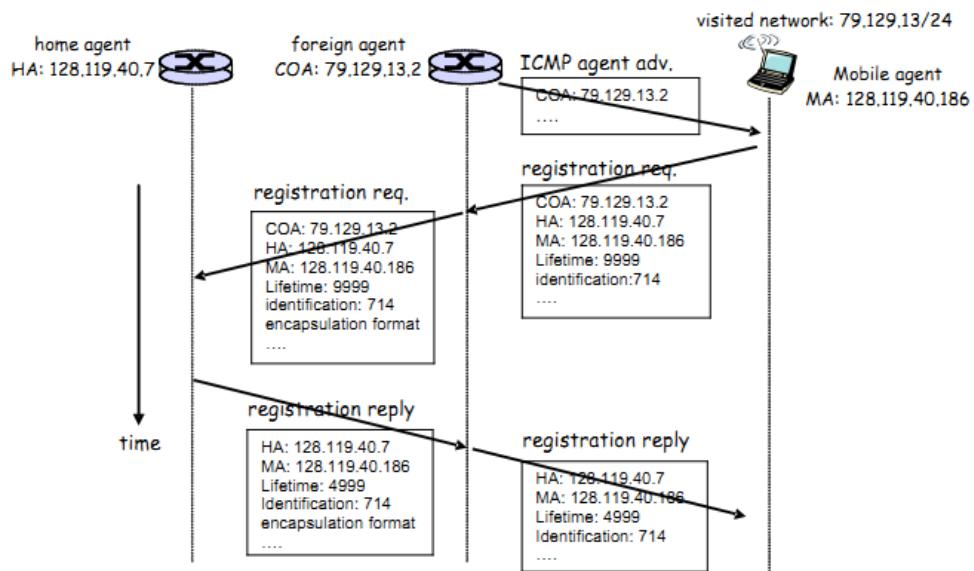
Initiative de l'agent

Un agent utilise des paquets ICMP étendus pour déclarer qu'il est un home ou un foreign agent. On communique également une care-of-address.



L'agent mobile envoie à un agent étranger un message (registration request) qui indique l'adresse du home agent. L'agent étranger va alors transmettre la requête à l'home agent.

L'home agent va alors envoyer un message de confirmation (registration reply) au foreign agent, qui va également le communiquer au mobile agent.

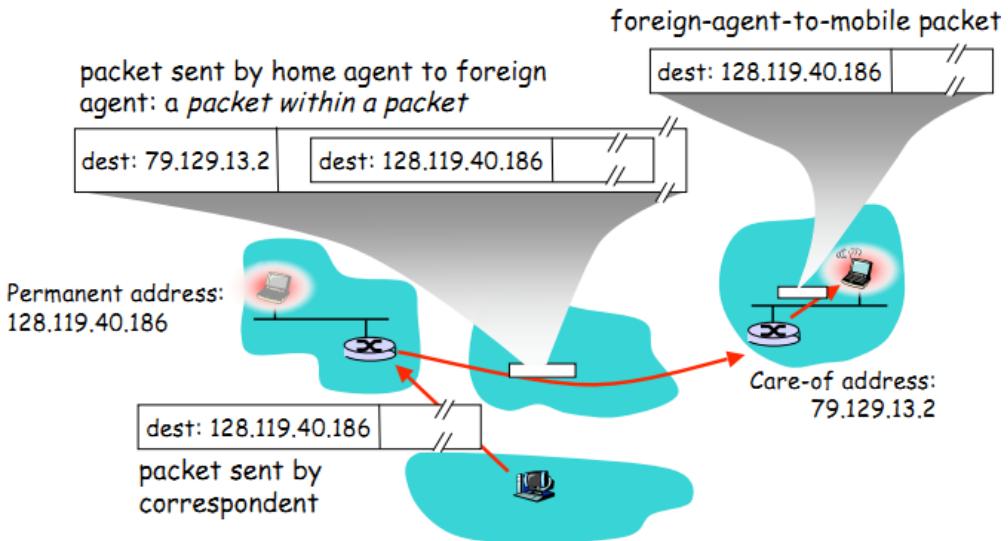


Sollicitation d'un agent

Le périphérique mobile broadcast des messages de sollicitation, qui sont simplement des messages ICMP.

3.5.2 Enregistrement avec l'home agent

3.5.3 Routage indirect



3.6 Mobilité dans les réseaux cellulaires

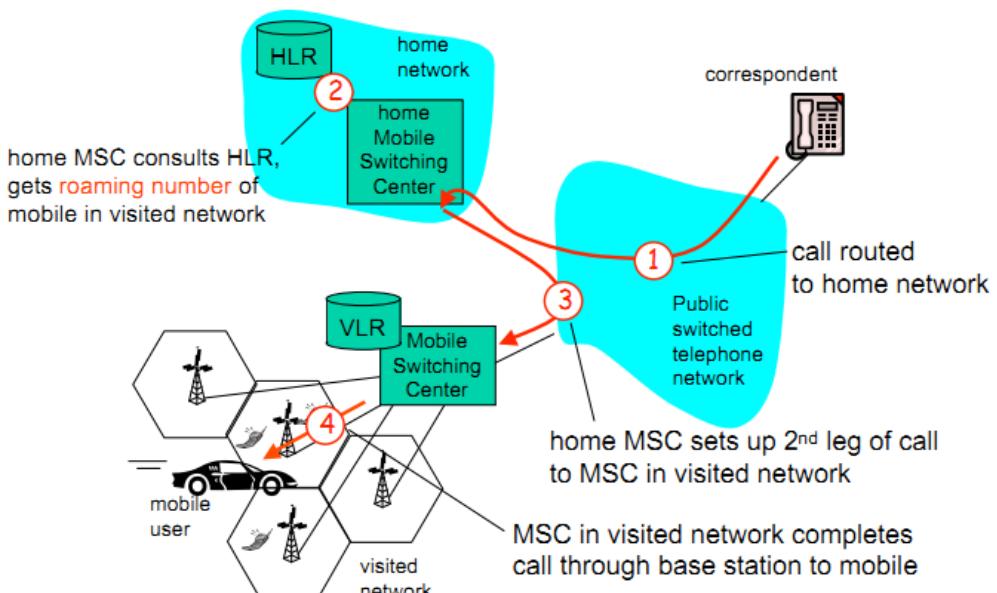
Chaque opérateur possède un réseau cellulaire, avec leurs propres antennes, qui couvrent le territoire.

Il y a deux types de réseau :

- le home network : le réseau cellulaire auquel on a souscrit. Dedans se trouve une base de données (HLR, home location register) qui contient des informations sur les utilisateurs (numéros de téléphone, profil, services, billing, etc), ainsi que des informations sur la localisation d'un utilisateur dans le réseau, autrement dit dans quelle cellule on se trouve.
- le visited network, le réseau dans lequel le mobile se trouve. Il contient également une base de données (VLR, visitor location register), avec des informations sur chaque utilisateur du réseau. Il peut être le même réseau que le home network.

Déroulement d'un appel :

1. un appel est redirigé vers le réseau du correspondant (home MSC)
2. le hMSC va consulter la BDD et regarder s'il se trouve dans le réseau. Il récupère le roaming number (sorte de care-of-address) pour localiser un mobile.
3. en utilisant le roaming number, le hMSC va rediriger l'appel vers le MSC du bon réseau.
4. l'appel est dirigé vers le correspondant à partir du MSC, en utilisant la BDD.



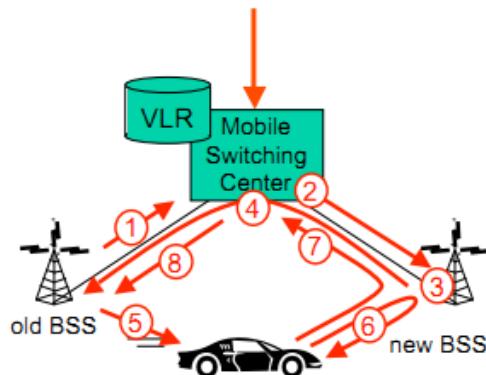
3.6.1 Gestion d'un handoff

Le but du handoff est de router un appel via une nouvelle base station, mais sans l'interrompre.

Raisons pour changer de BSS :

- un signal plus puissant avec une autre BSS, afin d'économiser de la batterie et d'avoir une meilleure connectivité ;
- load balance, répartir le trafic de manière optimale ;
- un comportement (policy) décidé par l'opérateur. Cependant, tous les mobiles doivent pouvoir changer de BSS de la même façon, quelque soit la raison.

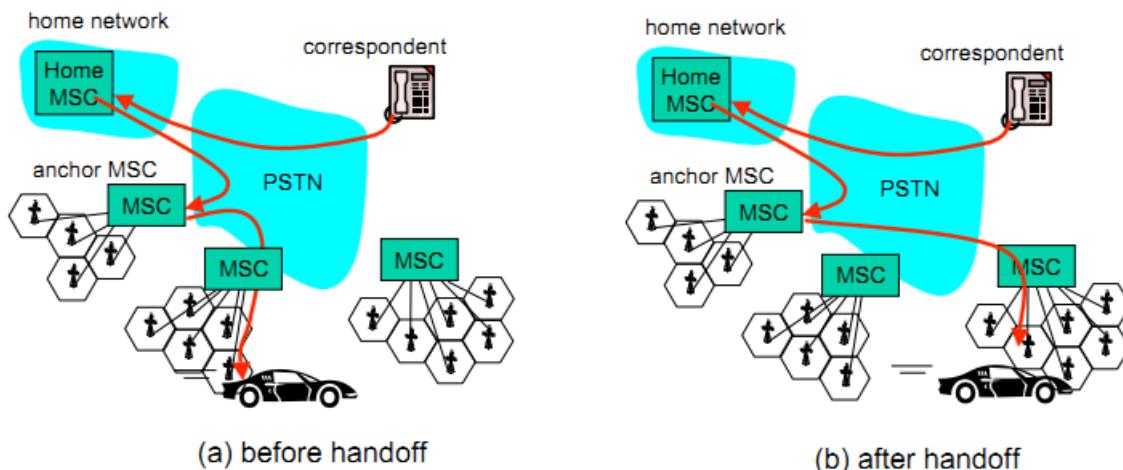
Il faut gérer le passage d'un BSS à un autre. C'est l'ancien BSS qui va déclencher le changement, lorsqu'il détectera que le signal se détériore et/ou si la cellule commence à être surchargée.



1. l'ancien BSS doit communiquer qu'il y a un changement de BSS, et doit envoyer un nouveau (ou plusieurs) BSS (le choix se fera selon la police de l'opérateur). Il faut cependant être le plus précis possible dans le choix du BSS, sinon à la seconde étape il y aura des réservations de ressources inutiles ;
2. le MSC met en place un chemin, il alloue des ressources pour atteindre le nouveau BSS ;
3. le nouveau BSS alloue un canal radio pour le mobile ;
4. le nouveau BSS signale au MSC et au vieux BSS qu'il est prêt pour le changement ;
5. le vieux BSS informe le mobile qu'il doit effectuer le handoff sur le nouveau BSS ;
6. le mobile contact le nouveau BSS pour activer le nouveau canal ;
7. le mobile communique à travers le nouveau BSS au MSC que le handoff est complet. Ce dernier va alors rediriger l'appel ;
8. les ressources allouées au MSC et au vieux BSS sont libérées.

Les stratégies d'allocations doivent aussi être précises, pour réserver correctement des ressources au bon moment. Par exemple, on pourrait mettre la priorité aux agents qui ont un appel en cours, afin de ne pas le perdre, en désavantageant les nouveaux appels. L'appel peut toujours être perdu s'il n'y a pas de canal libre dans le nouveau BSS.

Ce principe s'applique quand on passe d'un MSC à un autre, mais sous le même opérateur. On va également définir un anchor MSC. Comme pour mobile IP, on va éviter de chaîner les MSC, les MSC intermédiaires seront supprimés. Il y aura des signaux supplémentaires pour rediriger l'appel vers les MSC, mais il y a moyen de ne pas avoir plus de trois redirections.



Il y aura au maximum 3 MSCs sur le chemin de l'appel.

Un anchor MSC est utile, car pour changer le chemin home MSC vers un MSC, il y a beaucoup plus de ressources à mobiliser. Mobile IP est avantage par rapport à ceci, c'est plus flexible car il n'y a pas de ressources bloquées.

Si on passe d'un réseau à un autre, mais qui appartient à un autre opérateur, on perdra l'appel.

GSM element	Comment on GSM element	Mobile IP element
Home system	Network to which mobile user's permanent phone number belongs	Home network
Gateway Mobile Switching Center, or "home MSC". Home Location Register (HLR)	Home MSC: point of contact to obtain routable address of mobile user. HLR: database in home system containing permanent phone number, profile information, current location of mobile user, subscription information	Home agent
Visited System	Network other than home system where mobile user is currently residing	Visited network
Visited Mobile services Switching Center. Visitor Location Record (VLR)	Visited MSC: responsible for setting up calls to/from mobile nodes in cells associated with MSC. VLR: temporary database entry in visited system, containing subscription information for each visiting mobile user	Foreign agent
Mobile Station Roaming Number (MSRN), or "roaming number"	Routable address for telephone call segment between home MSC and visited MSC, visible to neither the mobile nor the correspondent.	Care-of-address

3.7 Impact sur les protocoles de plus haut niveau

Logiquement/fonctionnellement, l'impact est minimal, vu que les adresses IP ne changent pas, tout est transparent. Le modèle du best effort reste inchangé, et TCP et UDP peuvent fonctionner sur un support sans-fil mobile.

Cependant, vu qu'il y a des pertes de connectivité, les délais dus aux erreurs binaires et aux pertes de paquets sont plus grands, il y aura plus de retransmissions.

TCP va en particulier mal réagir, car il va interpréter une perte de paquet comme de la congestion, alors que cela pourrait être juste un problème de mobilité. La robustesse n'est pas touchée, mais le débit sera plus bas s'il y a trop de perte de connectivité.

Il faudrait que TCP puisse distinguer la "vrai" congestion, en séparant le cas d'une perte due à la connectivité. Il est difficile de communiquer un feed-back explicite du réseau en citant l'erreur et son origine (par exemple, l'erreur pourrait être dans l'identifiant d'un paquet). Cela peut être amélioré (inférence), mais c'est un gain de complexité.

Chapitre 4

Applications et transport multimédia

4.1 Applications réseau multimédia

Il y a trois types d'applications MM (MultiMedia) :

- stored streaming
- live streaming
- interactive, real-time

Les caractéristiques fondamentales de ces applications sont que

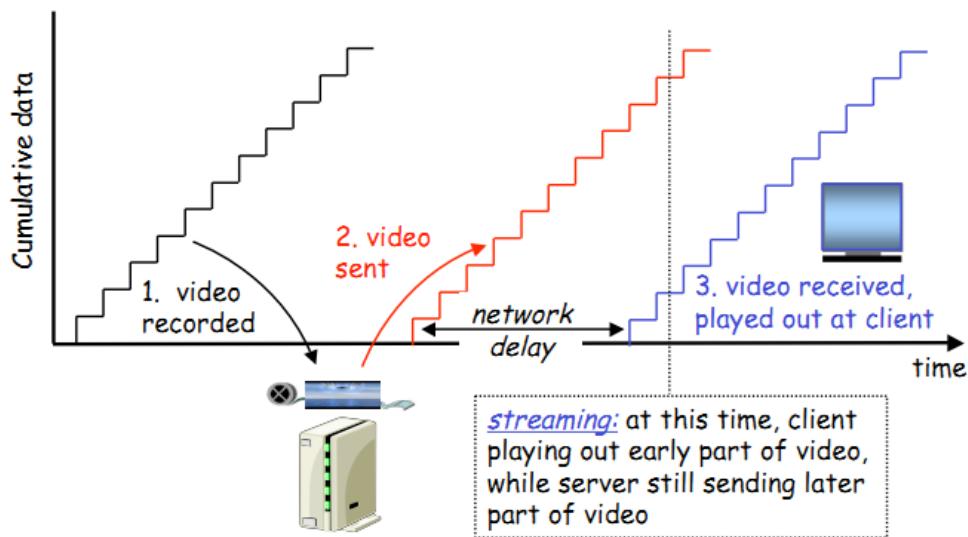
- elles sont sensibles au délai : le délai pour le transport end-to-end, ainsi que les variations de délai (delay jitter) ;
- elles sont sensibles à la perte : des pertes peu fréquentes causent des glitches mineurs ;
- c'est l'antithèse des données, qui sont intolérantes à la perte, mais tolérantes sur le délai.

4.1.1 Streaming Stored Multimedia

Il s'agit du cas d'un média stocké sur une machine distante et auquel on voudrait avoir accès.

Le streaming désigne le fait que le client peut commencer à jouer le fichier avant qu'il soit complètement transféré. On utilise un buffer, pour parer au risque de perte d'une transmission.

Le fait que le streaming soit stocké fait que le buffer est de taille conséquent ; cela revient à télécharger le fichier complètement.



On peut y ajouter de l'interactivité (pause, FF), cette fonctionnalité agit un peu comme un circuit virtuel (VCR). On tolère un délai d'initialisation de 10 secondes, et un délai de 1 à 2 secondes pour qu'une commande prenne effet.

4.1.2 Streaming Live Multimedia

Il s'agit toujours de streaming, si ce n'est que le buffer doit être plus petit et l'interactivité plus limitée (pas de FF). Cependant, on peut toujours revenir en arrière. On a toujours les contraintes de timing.

4.1.3 Real-Time Interactive Multimedia

Cela désigne la téléphonie par IP, la vidéo conférence, etc.

Il y a une nécessité de délais très courts. Le temps de propagation intervient, mais le plus gros délai vient de la création du paquet (accumulation de données) ou de la compression des données ("facile" en audio, compliquée en vidéo).

En général, en audio, un délai inférieur à 150ms est bon et inférieur à 400ms est tolérable.

Sur l'Internet d'aujourd'hui

TCP/UDP/IP sont des services du meilleur effort, il n'y a pas de garantie sur le délai ni la perte, or c'est ce qu'on souhaiterait avoir. Cependant, il est possible de mettre en place des techniques afin de limiter les effets (par exemple le playout buffer, qui permet d'avoir un flux constant mais en ajoutant du délai), avec des limites et des concessions à faire. Ces mécanismes sont à créer au niveau applicatif.

Il y a plusieurs manières d'améliorer le support multimédia d'Internet :

1. permettre la réservation de bande passante par les applications, ce qui nécessite du software plus compliqué dans les hôtes et les routeurs. Solution devant être standardisée.
→ Solution compliquée et coûteuse.
2. laisser faire, pas de changements majeurs à part augmenter la bande passante quand c'est nécessaire. On pourrait aussi mettre en place du multicast, et si ce n'est pas possible au niveau du réseau, l'utiliser au niveau applicatif. On peut aussi utiliser des réseaux de distribution de contenu ; les fichiers sont répliqués à travers le monde, et lorsqu'on demande l'accès à un d'entre eux, on serait redirigé vers le serveur le plus proche.
→ Solution facile au niveau du réseau.
3. introduire des changements plus légers, en ajoutant une discrimination entre deux types de trafic (prioritaire et best-effort).

Problème : qu'est-ce qui va contrôler la classe du paquet ? On pourrait profiter de la priorité de la première classe pour profiter de la bande passante. De plus, c'est compliqué à mettre en place et coûteux, et on n'a pas nécessairement de garantie.

D'un point de vue business, on pourrait appliquer un tarif plus élevé pour des paquets de première classe, avec un quota par utilisateur. C'est également une solution compliquée.

Cependant, de nos jours, on préfère les solutions peu coûteuses et la connectivité à la qualité des services. La deuxième solution est utilisée ; la troisième commence à être implémentée par les ISP.

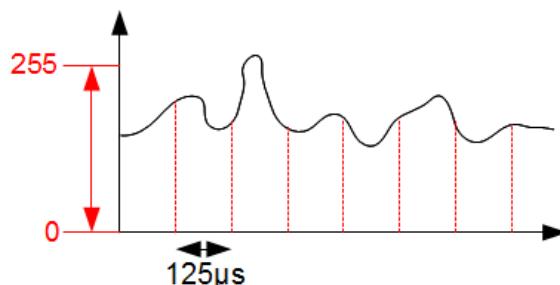
4.1.4 Audio digital et compression

Le PCM (pulse code modulation) désigne l'échantillonnage de la voix sur 8 bits (en Europe). Cependant, vu qu'il n'y a que 255 possibilités, il y a une erreur d'arrondi, et la reconstruction du signal ne sera pas parfaite.

L'échantillonnage doit être suffisamment grand que pour pouvoir capter toutes les fréquences. On restreint alors la voix dans la bande de fréquences de 0 à 4KHz. Grâce au théorème de Nyquist, on sait qu'il faut échantillonner au double de la fréquence maximale, soit 8KHz, pour pouvoir reconstruire le signal ; échantillonner plus ne servira à rien.

On prend donc, par seconde, 8000 échantillons de 8 bits, ce qui fait un débit de 64kps.

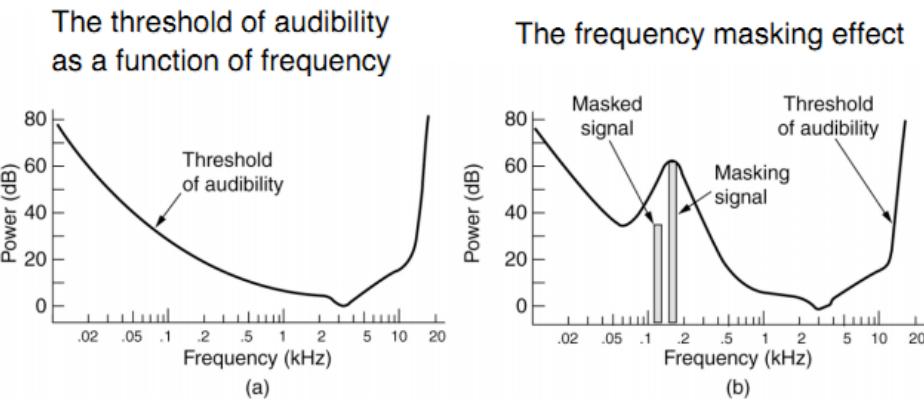
Le mapping de l'échantillon et de l'amplitude n'est pas linéaire ; on aimerait plus de valeurs (donc plus de précision) au début qu'à la fin de l'axe de l'amplitude. On utilise alors une échelle logarithmique.



Avec un débit de 64kbps, la qualité est excellente. Avec 32 kbps, on est à la limite de la bonne qualité.

Avec de la musique, on a un plus grand spectre de fréquences à échantillonner, de 0 à 22Khz.

Pour la téléphonie Internet, on peut mettre en place un modèle de la voix, et lors d'un échantillonnage, en extraire des paramètres qui permettront de reconstruire la voix via le modèle.

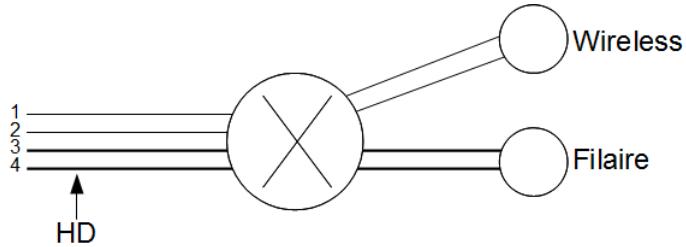


Le codeur va ignorer ce qui ne peut pas être entendu par l'oreille humaine, les signaux masqués. Cette technique est utilisée par le codec MP3. On arrive ainsi à compresser des CD stéréos jusqu'à 96-128kbps.

4.1.5 Compression vidéo

Pour compresser de la vidéo, on exploite la redondance spatiale (dans l'image) et temporelle (d'une image à l'autre).

Un domaine de recherche sont les layered videos : il y a un envoi de "couches de qualité", qui seront ignorées d'après certains critères (support de transmission, etc).



On dispose de plusieurs schémas d'encodage et de décodages. Il y a deux types de schémas :

- les schémas asymétriques ; par exemple, en VoD, l'encodage peut être lent (il n'est fait qu'une fois), mais le décodage doit être rapide ;
- les schémas symétriques : par exemple, pour des vidéos conférences (et en général pour les applications multimédia en temps réel), les deux schémas doivent être rapides.

On peut également envisager une compression avec une perte, ce qui permet d'obtenir des ratios de compression meilleurs.

Il y a deux principaux schémas de compression :

- l'encodage entropique (sans perte), et
- l'encodage de source (avec perte).

Encodage entropique

Il s'agit d'un encodage sans perte. Trois exemples typiques :

- run-length encoding : les symboles qui sont répétés sont marqués par un symbole spécial et le nombre d'occurrence ;
- encodage statistique : court codes pour des symboles fréquents ;
- look up table : on définit un tableau de couleurs qui sont utilisées, et encode un pixel en l'indice de sa couleur dans le tableau.

Source encoding

Il s'agit d'un encodage avec une perte. Trois exemples typiques :

- encodage différentiel : des séquences de valeurs sont représentées par la différences avec les précédentes valeurs. Cela est utile si on utilise moins de bits pour encoder. Il y a une perte si la différence est trop grande et ne peut être encodée.

On peut rendre ce schéma sans perte si on utilise un encodage à taille variable, mais on perd en décompression.

- transformations du signal, ce qui permet par exemple pour Fourier de récupérer les termes A_k , ω_k et ϕ_k . Cependant, vu qu'il y en a une infinité, on ne peut qu'en prendre un nombre fini, il y a donc perte.

$$\sum_{k=0}^{\infty/N} A_k \sin(\omega_k t + \phi_k)$$

Cela s'applique bien pour les fonctions périodiques.

- variante de la Look Up table, où la table pourrait être standardisée, afin de ne pas devoir l'envoyer. Dès lors, les valeurs des pixels sont arrondies aux plus proches valeurs dans la table.

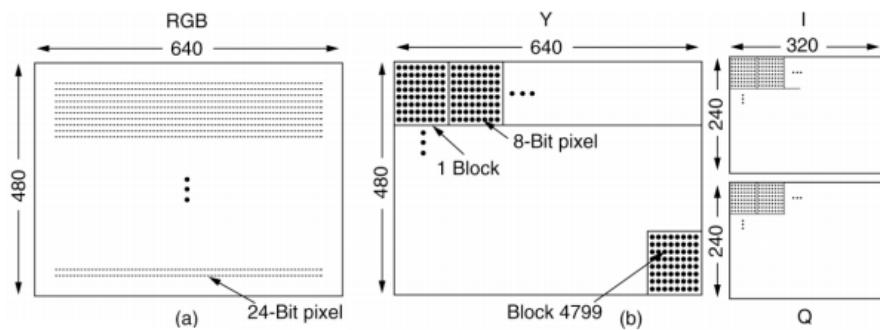
4.1.6 L'exemple du JPEG

JPEG (Joint Photographic Experts Group) est un standard pour la compression d'images fixes. Typiquement, on a un ratio de compression de 20 :1. C'est un schéma symétrique, avec perte, qui se déroule en 6 étapes.

1. l'encodage du pixel est changé, de RGB en YIQ (un canal de luminance et deux de chrominance). Cette conversion se fait à l'aide d'un produit matriciel.

$$\begin{pmatrix} Y \\ I \\ Q \end{pmatrix} = \begin{pmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

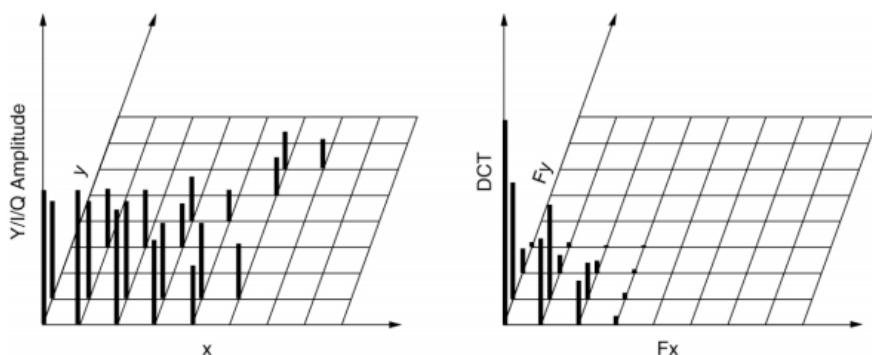
Un des avantages est qu'avec une image en noir et blanc, le canal Y suffit.



Un autre avantage de cet encodage est que les matrices I et Q sont facilement compressibles, qui sont déjà deux fois plus petites en encodant la moyenne de blocs de 4 pixels. On n'effectue pas cette approximation pour la matrice Y car l'oeil humain le remarquerait plus facilement, alors qu'avec I et Q on ne voit pas la différence.

On va ensuite soustraire 128 à chaque élément, afin d'avoir 0 comme valeur au milieu. Enfin, on divise la frame en blocs de 8×8 .

2. on effectue une transformation en cosinus discrète. La première valeur encodée (0,0), on a une approximation, c'est la moyenne du bloc.



Si on envoie plusieurs valeurs, on peut les utiliser pour assombrir ou éclaircir des zones dans le blocs, par exemple (0,0) donne la moyenne du bloc, (1,0) la moyenne pour la partie supérieure, (0, 1) la moyenne de la partie inférieure.

Quand on a le premier coefficient de la série (A_0), on a une moyenne de l'ensemble, car le reste du terme n'a pas d'influence ($k = 0$).

L'avantage de la transformation en cosinus discrète par rapport à Fourier est que l'énergie est concentrée à l'origine, les valeurs sont limitées, alors que Fourier "éteint" plus les valeurs.

Toutes les valeurs sont approximées, car elles sont réelles, il y a donc une perte.

Il en sort des blocs de 8×8 éléments, les coefficients de la transformation.

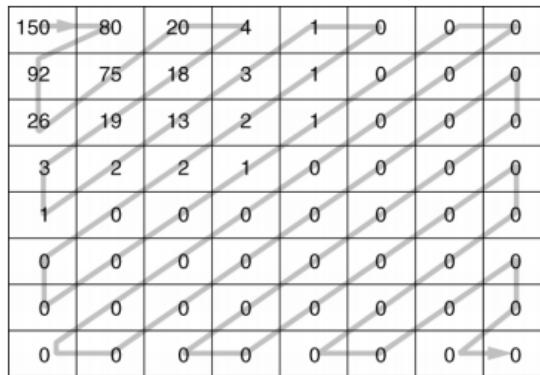
3. application d'une sorte filtre passe-bas, ce qui entraîne des pertes. On divise la matrice des coefficients DCT par une table de quantification. Cela permet d'avoir une table contenant beaucoup plus de zéros.

DCT Coefficients									Quantization table								Quantized coefficients							
150	80	40	14	4	2	1	0		1	1	2	4	8	16	32	64	150	80	20	4	1	0	0	0
92	75	36	10	6	1	0	0		1	1	2	4	8	16	32	64	92	75	18	3	1	0	0	0
52	38	26	8	7	4	0	0		2	2	2	4	8	16	32	64	26	19	13	2	1	0	0	0
12	8	6	4	2	1	0	0		4	4	4	4	8	16	32	64	3	2	2	1	0	0	0	0
4	3	2	0	0	0	0	0		8	8	8	8	8	16	32	64	1	0	0	0	0	0	0	0
2	2	1	1	0	0	0	0		16	16	16	16	16	16	32	64	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0		32	32	32	32	32	32	32	64	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0		64	64	64	64	64	64	64	64	0	0	0	0	0	0	0	0

Les nombres sont choisis pour mettre le plus de précision sur les coefficients qui auront le plus d'impact sur le rendu pour un oeil humain.

Les valeurs seront retrouvées en multipliant la matrice de quantification.

4. quantification différentielle ; on remplace les coefficients des blocs (en commençant par le coin supérieur gauche) par la différence par rapport au bloc précédent.
5. envoi du pattern en zigzag, ce qui permet de rassembler les zéros à la fin, et donc d'arrêter la transmission à la dernière valeur non nulle, avec un symbole spécial. On pourrait faire de même avec les symboles qui se répètent, en combinant le symbole et le nombre d'occurrences.



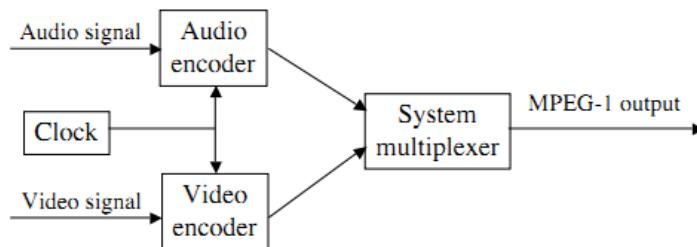
6. encodage statistique (encodage de Huffman)

4.1.7 MPEG

Le MPEG (Motion Picture Experts Group) est un standard ISO de compression audio et vidéo. Il y a plusieurs variantes : MPEG-1 (qualité CD-rom), MPEG-2 (qualité DVD) et MPEG-4 (video conferencing avec un framerate bas et une résolution moyenne).

MPEG-1

Généralement, un encodeur est utilisé pour l'audio et un autre pour la vidéo, qui seront ensuite multiplexés. Une clock ajoute un timestamp aux deux flux, afin de les synchroniser lors du décodage.



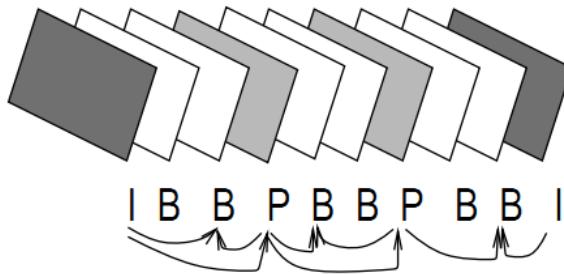
Pour la compression audio, on va utiliser MP3 et exploiter la redondance des deux canaux (dans le cas d'un stream en stéréo).

Pour la compression vidéo, on va profiter des redondances spatiale et temporelle. La redondance spatiale est gérée comme pour JPEG.

Pour la redondance temporelle, avant d'encoder, on va d'abord analyser les images et voir s'il y a des similitudes entre elles, et ce sans se limiter à la taille des blocs. Lors du décodage, il suffira alors de dupliquer la zone de pixels, ou bien de corriger un bloc (plus ou moins lumineux, etc).

On définit différents types de frame :

1. I frames (Intracoded) : elles contiennent des images encodées au format JPEG. Elles apparaissent périodiquement à la sortie, pour les changements de scènes, mais aussi pour ne pas propager des erreurs ou d'éviter le cas où il y a une perte d'une I-frame, ou bien lorsqu'il y a des sauts dans la vidéo (fast forward, rewind).
2. P frames (Predictive) : elles contiennent les différences bloc par bloc par rapport à la frame précédente. On cherche un macrobloc (Y, I, Q) dans la frame précédente qui est égale ou légèrement différente, et on encode l'offset en position et en différente.
3. B frames (Bidirectional) : idem que les frames P, mais on cherche dans la prochaine frame I ou la prochaine frame P.
4. D frames (DC-codec) : frames qui contiennent les moyennes des blocs, et qui permettent d'effectuer des fast forward (à basse résolution).



Les ordres d'encodage et de décodage ne sont pas les mêmes que l'ordre des frames, afin de pouvoir décoder dès qu'on a les données.

Profil	I	B	B	P	B	B	I
Ordre	1	3	4	2	6	7	5

Une B-frame engendre un encodage et un décodage plus difficile pour le CPU, mais sont moins lourdes.

Le network doit être adapté au profil ; la perte d'une I-frame aura un plus grand impact qu'une B-frame.

MPEG-2

Le principe de fonctionnement est similaire à celui de MPEG-1. On a une meilleure qualité ; on utilise des blocs de 10×10 coefficients de DCT, au lieu de 8×8 .

On a plusieurs niveaux de résolution, le plus bas correspond à du MPEG-1. Il y a également plusieurs profils (par exemple un sans frames B, pour simplifier l'encodage).

Généralement, le débit tourne dans les 3-4Mbps, mais on peut aller jusqu'à 100Mbps (HDTV).

4.1.8 Streaming audio et vidéo stocké

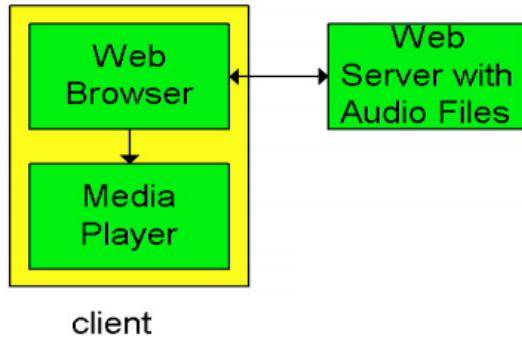
Le streaming au niveau applicatif tente de tirer le plus du service de best effort. On a plusieurs techniques :

- un buffering côté client,
- l'utilisation d'UDP et de TCP, et
- des encodages multiples de médias.

Un media player a pour but le retrait du jitter (délais variables), la décompression, la gestion d'erreurs et d'offrir une interface graphique avec des contrôles pour l'interactivité.

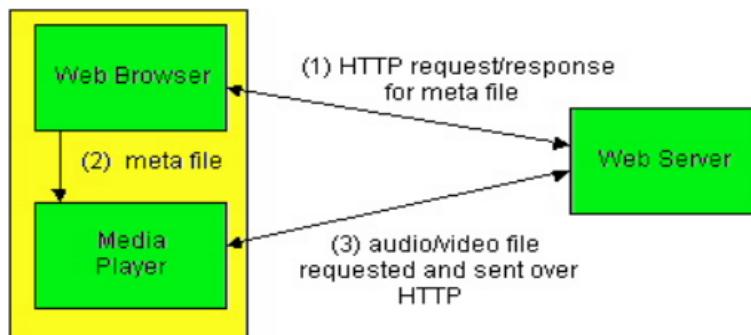
On a différentes approches pour gérer le streaming :

1. Approche simple : les fichiers audio et vidéo sont stockés dans des fichiers, qui sont transférés comme des objets HTTP. Ils sont donc reçus entièrement par le client, puis passés au lecteur.

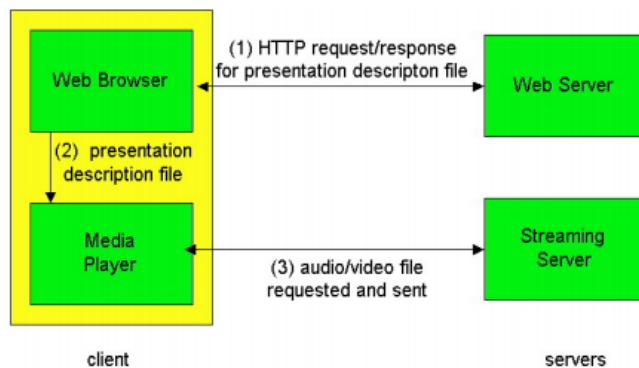


Ici, l'audio et la vidéo ne sont pas streamés ; il n'y a pas de streaming car on doit tout transférer pour pouvoir lire le fichier. De plus, il peut y avoir un long délai avant d'avoir le fichier complet.

2. Approche streaming : le browser récupère un metafile, qu'il passe au lecteur. Ce fichier contient toutes les informations nécessaires pour qu'il soit streamé en HTTP. Le lecteur contacte ensuite le serveur, qui stream les flux audio et vidéo au lecteur.



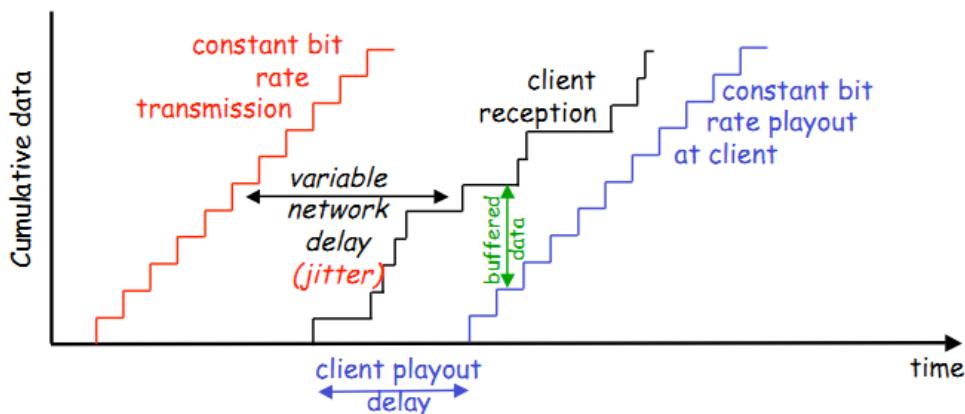
3. Approche streaming sans HTTP : cela permet l'utilisation de protocoles non HTTP entre le serveur et le lecteur. On a alors le choix entre UDP ou TCP pour l'étape 3.

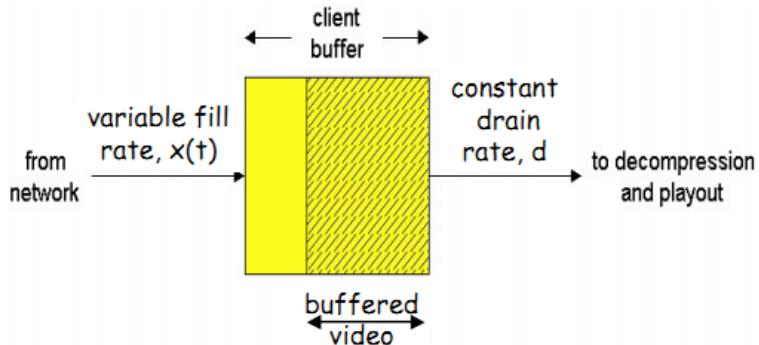


HTTP est juste utilisé pour récupérer le meta file, après c'est un protocole dédié qui prend le relais.

Buffering client

On va introduire un délai de playout, qui va permettre de compenser le délai du réseau et le jitter.





TCP ou UDP

UDP permet de mieux gérer la récupération d'erreur (au niveau applicatif). Avec un buffer, on dispose d'un budget temps qui permet d'attendre éventuellement qu'un paquet manquant arrive. Avec TCP, dès qu'un manquement est détecté, la retransmission aura lieu systématiquement.

UDP envoie au rythme de l'encodage, TCP ne va faire qu'essayer mais sera limité (slow start, congestion avoidance, receiver buffer, etc) ; il y a énormément de fluctuations dans le remplissage du buffer. Du coup, plus il y a de fluctuations, plus il faut un grand playout buffer, alors qu'avec UDP ce n'est pas nécessaire vu qu'il n'y a pas de variations de débit à compenser.

Par contre, TCP passe mieux à travers les firewalls et les périphériques réseaux, qui pourraient filtrer les connexions UDP.

Encodages différents

Le fait d'avoir plusieurs encodages, avec des débits différents, d'un même média permet de s'adapter au débit de réception des clients.

4.1.9 RTSP - Real-Time Streaming Protocol

HTTP n'est pas conçu pour le contenu multimédia et ne dispose pas de commandes pour effectuer des avances rapides, etc.

RTSP est un protocole client-serveur au niveau applicatif, qui permet le contrôle (rewind, pause, etc). Il ne définit pas

- comment les flux audio et vidéo sont encapsulés ;
- la manière de transporter les données (TCP ou UDP) ;
- les buffers du lecteur.

RSTP utilise le contrôle out-of-band, comme FTP ; ce dernier utilise deux connexions TCP : une pour le transfert de fichier en lui-même, et une autre (out-of-band) qui permet d'envoyer des commandes FTP (se déplacer dans l'arborescence du système, copier un fichier, etc).

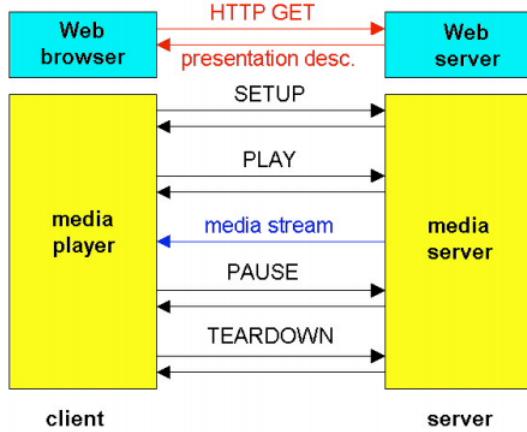
Cette séparation est faite car TCP pourrait toujours fixer le débit à un niveau très bas (à cause de congestion). Du coup, toutes les données à transférer seront bufferisées, de même que les commandes. L'autre connexion permet de ne pas bloquer immédiatement les commandes ; généralement elles sont de taille très réduite.

RTSP utilise la même technique, si ce n'est qu'on n'est pas limité à TCP.

Metafile

Lors de la lecture d'un streaming, un metafile est communiqué au web browser, qui sera ensuite envoyé au lecteur. Ce dernier établit alors une connexion de contrôle RTSP.

Pour une session, le meta-file peut définir plusieurs pistes audio, avec des encodages différents et avec une source par encodage. Il en est de même pour le flux vidéo.



4.2 Retirer le meilleur du service best effort

Le but est d'améliorer les interactions entre les applications à partir du service de best effort d'IP.

Pour la téléphonie sur Internet, on va utiliser les silences dans les conversations pour modifier les paramètres des protocoles. On définit un talk spurt comme une période où un interlocuteur parle.

Ainsi, pendant un talk spurt, on utilise le débit de 64 kbps ; les paquets ne sont générés que lors d'un talk spurt. On a des morceaux de 20 ms à un débit de 8KBytes/sec, soit 160 bytes de données.

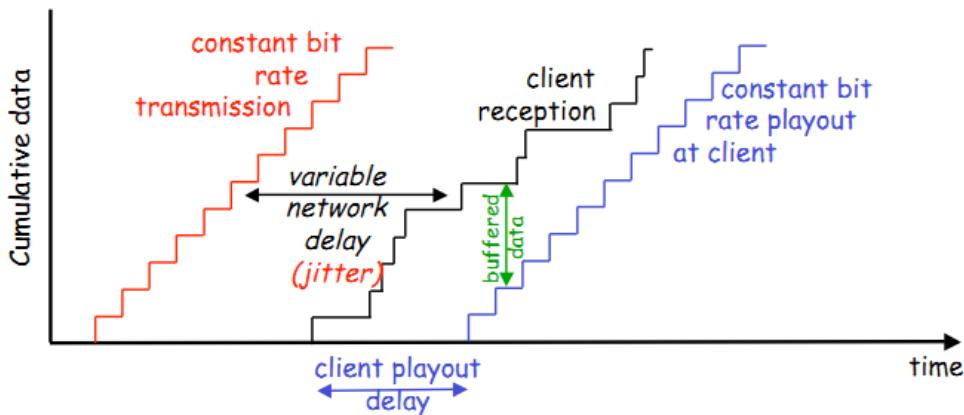
Les problèmes surviennent lors de la perte de paquet (network loss), due à de la congestion, et du délai (delay loss), où les paquets arrivent trop tard pour être joués par le récepteur. Le problème avec cet encodage est que la perte d'un paquet entraîne la perte de 20ms de voix.

Cela pourrait être pire avec certains schémas de compression qui peuvent rendre le schéma plus fragile. Par exemple, si on perd une frame I d'une vidéo MPEG, on perd aussi les frames B et P qui y réfèrent. Si on n'utilisait que des frames I, il n'y aurait pas d'effet de cascade, mais la compression serait mauvaise.

De plus, certaines frames sont plus lourdes que d'autres (une frame I générera plus de paquets IP qu'une frame B ou P). Selon la manière d'organiser la répartition des blocs dans les paquets, on peut avoir des précautions à prendre si on perd un paquet, qui ferait donc perdre des blocs. Les dépendances entre blocs rendent le schéma moins robuste face à la perte de paquet.

Pour la téléphonie sur Internet, on considère que le délai maximum tolérable est de 400ms. Les pertes de paquet sont tolérables si le ratio est compris entre 1% et 10%. Pour deux paquets consécutifs, la différence des temps de transmission doit valoir environ 20ms pour être tolérable.

4.2.1 Utilisation d'un délai de playout

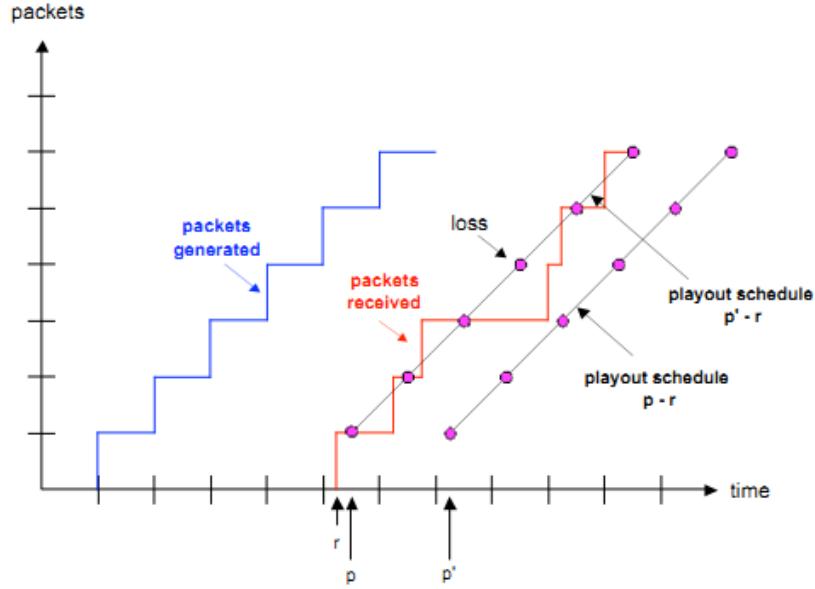


Playout fixé

On pourrait définir un délai de playout fixe ; supposons qu'un récepteur s'attende à jouer chaque morceau q ms après qu'il soit généré. Si le morceau a un timestamp t , il sera joué au temps $t + q$. Si le morceau arrive après le temps $t + q$, les données arrivent trop tard pour être jouées, elles sont donc perdues.

On a donc un équilibre à trouver entre un grand q , qui gère mieux la perte de paquet, et un petit q qui offre une meilleure interactivité.

Supposons qu'un émetteur envoie des paquets toutes les 20ms durant un talk spurt. Le premier paquet est reçu au temps r et sera joué au temps p pour un premier délai de playout, en p' pour un second délai de playout plus long.



Playout adaptatif

Le but est de minimiser le délai de playout tout en gardant une perte minimale.

L'idée est d'utiliser un playout adaptatif, en estimant le délai du réseau et en ajustant le délai de playout au début de chaque talk spurt. Les périodes de silences sont compressées.

On va utiliser moyenne exponentielle, car plus on avance dans les échantillons plus les échantillons anciens ont moins d'importance $((1 - u)^\alpha)$. Cette méthode est également utilisée pour estimer le RTT de TCP.

Soient

- t_i le timestamp de génération du i ème paquet,
- r_i le temps auquel le paquet i est reçu par le récepteur,
- p_i le temps auquel le paquet i est joué par le récepteur,
- d_i l'estimation du délai du réseau moyen après avoir reçu le i ème paquet.

On a que $r_i - t_i$ est le délai du réseau pour le i ème paquet, et donc

$$d_i = (1 - u)d_{i-1} + u(r_i - t_i)$$

où u est une constante fixée (par exemple $u = 0.01$).

On peut également calculer la variance moyenne :

$$v_i = (1 - u)v_{i-1} + u|r_i - t_i - d_i|$$

Pour chaque paquet reçu, on va donc estimer d_i et v_i , mais ça ne sera utilisé qu'une fois au début d'un talk spurt (les paquets restants du talk spurt sont joués périodiquement). On a alors le délai de playout estimé p_i :

$$p_i = t_i + d_i + Kv_i$$

avec K une constante positive.

Un talk spurt peut être détecté lorsqu'on reçoit des paquets espacés de 20ms. Lorsqu'on a un trou de plus de 20ms entre deux paquets, on a un silence.

S'il n'y a pas de pertes, le premier paquet d'un talk spurt peut être détecté en regardant si la différence de timestamp de deux paquets consécutifs vaut plus de 20 ms. Cependant, à cause de la perte de paquet, on pourrait considérer des périodes de silence qui n'en sont pas. En ajoutant un numéro de séquence aux paquets, on peut différencier les pertes et les silences.

Ainsi, un talk spurt commence si la différence de timestamp de deux paquets consécutifs est de plus de 20ms et s'il n'y a pas de trou dans la séquence de numéros.

Les clocks des deux systèmes peuvent être décalées, mais la différence de temps restera constante et sera considérée comme un délai. Une clock parfaitement synchronisée vaudra t_i , une désynchronisée aura $t_i + \Delta$. Il y a une erreur dans le calcul $r_i - t_i$, mais elle sera constante. d_i aura aussi le délai supplémentaire, vu qu'il effectue une moyenne sur des valeurs qui comportent l'erreur.

On a alors $t_i \rightarrow t_i + \Delta$, $r_i - t_i \rightarrow r_i - t_i - \Delta$, $d_i \rightarrow d_i - \Delta$.

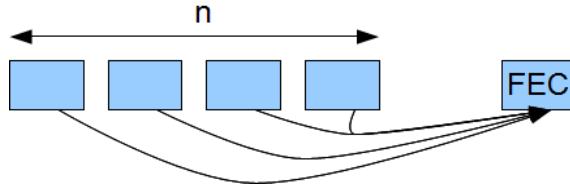
La variance n'est pas impactée ; $r_i - t_i - d_i \rightarrow r_i - t_i - \Delta - d_i + \Delta$.

Au niveau du playtime, $p_i = t_i + d_i + Kv_i \rightarrow t_i + \Delta + d_i - \Delta + Kv_i = p_i$; l'erreur faite sur le délai estimé est supprimée car elle est ajoutée au mauvais timestamp, elle se neutralise. Donc les clocks de deux systèmes n'ont pas besoin d'être synchronisées.

4.2.2 Récupération de perte de paquet

Schéma simple : FEC

FEC signifie Forward Error Correction. L'idée est d'insérer un paquet FEC dans un groupe de n paquets. Si l'un d'eux est perdu, on doit pouvoir le reconstruire à partir des autres paquets et du paquet FEC.



Pour créer le paquet FEC pour un groupe de n paquets, on prend le XOR des n paquets ; il agit comme des bits de parité pour chaque colonne. Les $n + 1$ paquets sont envoyés, en augmentant la bande passante de $\frac{1}{n}$. Le playout devra être suffisamment grand pour recevoir les $n + 1$ paquets.

Si on perd un paquet, avec le FEC on connaîtra la parité de chaque colonne ; en comparant les colonnes des paquets reçus, on peut retrouver le bit de chaque colonne du paquet perdu (soit on a un nombre pair soit un nombre impair de 1).

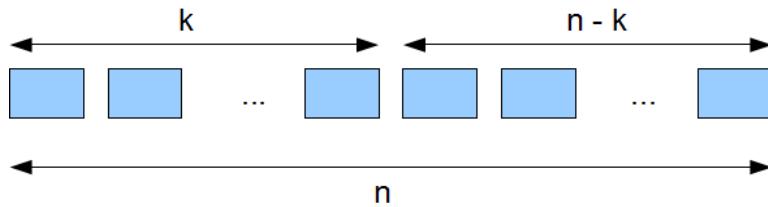
On a un équilibre à trouver : augmenter n revient à

- gaspiller moins de bande passante,
- augmenter le délai de playout, et
- augmenter la probabilité que deux ou plus morceaux soient perdus.

Si on perd plus d'un paquet, le schéma est inutilisable.

FEC : Reed-Solomon

Un code $RS(n, k)$ encode k paquets source dans n paquets.



On distingue plusieurs versions :

1. une version systématique : pour k paquets source, on ajoute $n - k$ paquets de redondance. Le schéma permet de perdre jusqu'à $n - k$ paquets, si on en perd plus, on ne peut pas tout reconstruire.
2. une version optimale : le code est optimal si on peut perdre $n - k$ paquets et reconstruire les k paquets ; les k paquets originaux peuvent être récupérés, en supposant qu'on récupère au moins k paquets parmi les n envoyés, quels qu'ils soient.

Code linéaire Cela peut se faire au moyen d'un système linéaire : on a x le vecteur des k paquets source. G est une matrice $n \times k$ et y est le vecteur des n paquets transmis. On a alors

$$y = Gx$$

$$\begin{pmatrix} y \\ \vdots \\ y_{n \times 1} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & & & \\ \alpha_{k+1} & & & \\ \alpha_n & & & \end{pmatrix}_{n \times k} \begin{pmatrix} x \\ \vdots \\ x_{k \times 1} \end{pmatrix}$$

Les 1 en diagonale permettent de garder les paquets originaux. Les coefficients après créent des combinaisons linéaires.

G n'est pas inversible (matrice rectangulaire), or on aimerait utiliser le processus dans l'autre sens.

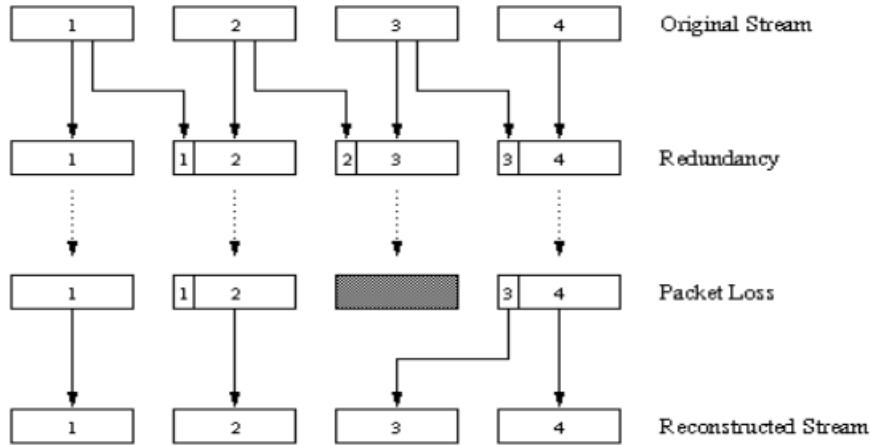
Pour le décodage, soit y' le vecteur des k paquets reçus, quel qu'ils soient. G' est la sous-matrice $k \times k$ de G dont les lignes correspondent à ces paquets. On a alors

$$x = G'^{-1}y'$$

En fait, parmi tous les paquets que l'on reçoit dans z , il peut contenir des trous (paquets perdus). On va alors supprimer les lignes qui correspondent à ces indices. On peut en retirer jusqu'à ce qu'il en reste k , ce qui donne y' . Dès lors, G' sera une matrice carrée, qui admet un inverse.

Autre schéma FEC

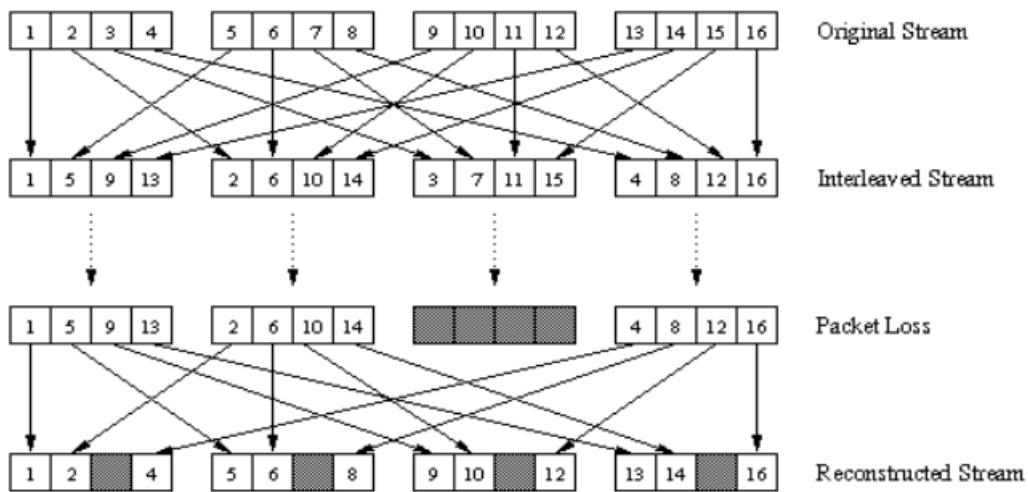
Pour un paquet i , on le compresse (PCM vers GSM par exemple) et on l'attache au paquet $i + 1$. Si un paquet est manquant, on peut utiliser la version low quality du paquet suivant. Il faut cependant, en cas de perte, attendre le paquet suivant.



On peut généraliser en ajouter le paquet i version low quality aux paquets $i + 1, i + 2$, etc.

Entrelacement de paquets

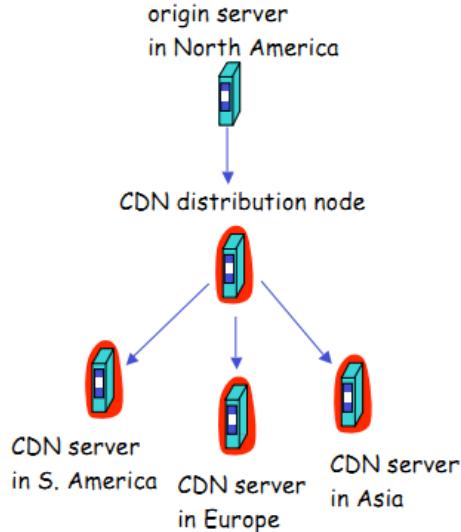
On découpe les 20ms en plusieurs parties, que l'on place dans des paquets de manière entrelacée (le premier paquet les 5 premières ms, le deuxième les 5ms suivantes, etc).



Cela permet de ne pas ajouter de redondance et une perte est difficile à remarquer, mais il y a un délai supplémentaire du au temps qu'il faut pour récupérer les paquets.

4.2.3 CDN - Content distribution networks

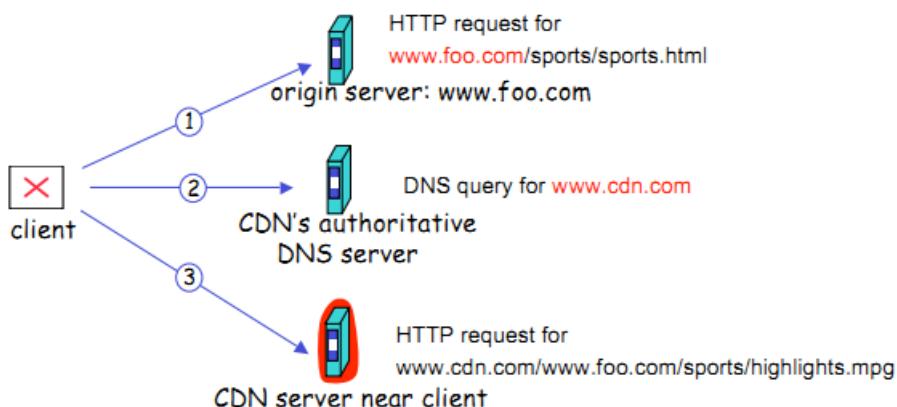
Une situation possible est qu'on stream un fichier à partir d'un serveur unique en temps réel.



La méthode consiste en la réplication d'un contenu sur des serveurs différents à travers l'Internet. Ainsi, on ne stream plus à partir du fichier original, mais à partir d'une copie d'un serveur plus proche.

Le serveur d'origine est vu comme un client pour le réseau. À chaque nouveau fichier à streamer, il va l'envoyer au CDN, qui va le placer à des emplacements qui permettent aux utilisateurs d'éviter les désagréments (perte, délai) dus à l'envoi de contenu sur de longues distances. À chaque mise à jour d'un fichier, le CDN mettra à jour ses copies.

Exemple d'un streaming avec un CDN.



1. La réponse du serveur HTTP sera modifiée de manière à rediriger vers un serveur CDN.
2. La résolution DNS est l'étape où on doit choisir le serveur le plus proche (grâce à une technique de mapping) et retourner son adresse IP.

Ainsi, le serveur d'origine ne fait plus que distribuer du HTML, et modifier les liens de streaming. Le CDN lui va distribuer les streaming, en utilisant son serveur DNS pour rediriger les requêtes vers un serveur CDN le plus proche.

Les avantages de la méthode sont la facilité de déploiement et le fait que cela reste totalement transparent.

Il est relativement facile de mapper un range d'adresses IP si on connaît les ranges d'IP des ISP. Ainsi, on peut approximer l'emplacement géographique d'un utilisateur. On n'aura pas d'informations intéressantes si l'ISP est situé au niveau mondial.

Cette détection n'est généralement pas assez. En plus de cela, les CDN surveillent les RTT entre leurs serveurs et des IP adresses. On associe ces IP à des adresses. On se base ainsi sur un délai pour localiser, et non sur des distances physiques.

Cependant, pinger de nombreuses IP avec beaucoup de serveurs ne donnera pas des résultats de manière efficace, des techniques doivent être mises en place. Par exemple générer une matrice de RTT entre des noeuds, pinger un petit sous-ensemble des noeuds choisis de manière aléatoire et inférer les mesures non effectuées. Grâce à la corrélation entre des noeuds (même localisation, même bottleneck), le système fonctionne, même avec une matrice relativement petite.

C'est au CDN de choisir la technique qu'il veut, ce n'est pas standardisé.

4.3 Protocoles pour des applications multimédia temps réel

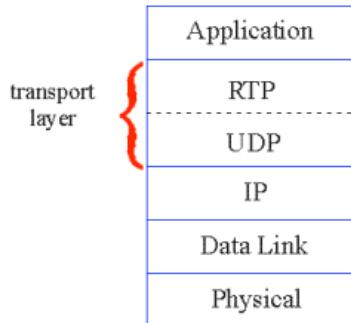
4.3.1 RTP - Real-time Protocol

RTP est un protocole spécifiant une structure pour les paquets transportant des données vidéo ou audio. Il est spécifié par un RFC et tourne dans les systèmes d'extrémités. Le but de ce protocole est de permettre l'interopérabilité entre les périphériques.

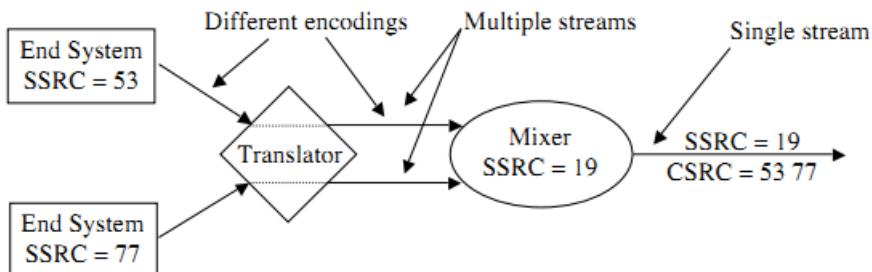
Les paquets RTP sont encapsulés dans des segments UDP, ils les étendent et fournissent

- un identifiant pour le type de donnée (payload)
- un numéro de séquence
- un timestamp.

On a besoin du timestamp ET du numéro de séquence. Cela permettait, lors de l'adaptation du playout delay, de savoir quand un talk spurt commence, et de ne pas le confondre avec une perte de paquets.



RTP reste un protocole du best effort, il n'y a pas de garanties de QoS. Les paquets RTP ne sont pas discriminés par rapport aux autres par les routeurs ; le caractère RTP d'un paquet n'est vu qu'aux systèmes d'extrémité.



Les end systems sont les applications qui génèrent et consomment le contenu des paquets RTP.

Un translator est un système intermédiaire qui change le schéma d'encodage sans altérer le timing. Il est utilisé au cas où il faudrait changer le type d'encodage, et peut aussi convertir un stream multicast en plusieurs streams unicast.

Un mixer est un système intermédiaire qui peut combiner plusieurs flux. Le nouveau flux généré a son propre timing (un nouveau SSRC).

Un SSRC (Synchronisation Source identifier) permet d'identifier la source. S'il y a un mixer, le SSRC l'identifie (et non pas un des flux d'entrées du mixer, vu que c'en est la composition), et on joint à part, dans le CSRC (Contributing Source identifier), des identifiants pour les "vraies" sources.

Header RTP

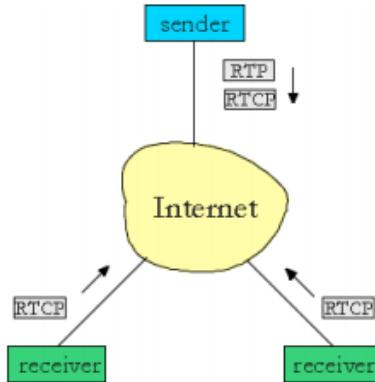
Un header RTP contient donc 4 éléments :

- le type de donnée (7 bits) : il indique le type d'encodage. Ainsi, si un expéditeur change de schéma d'encodage durant une conversation, c'est un moyen de le faire savoir au récepteur.
- un numéro de séquence (16 bits) : il est incrémenté de 1 pour chaque paquet RTP envoyé, et peut être utilisé pour détecter la perte d'un paquet et restaurer une séquence de paquet.
- un timestamp (32 bits) : généralement, pour l'audio, il incrémente toutes les périodes d'échantillonage. Il est relatif, on peut l'utiliser de la manière que l'on souhaite ; la clock continue à augmenter à un rythme constant, même lorsque la source est inactive.
- un SSRC (32 bits) : il identifie la source de chaque stream RTP. Chaque stream dans une session RTP doit en avoir un distinct.

4.3.2 RTCP - Real Time Control Protocol

RTCP travaille en conjonction avec RTP, ce n'est pas vraiment un protocole pour transporter les données. Il permet, entre des participants, de communiquer des informations sur le streaming (nombre de paquets perdus, de paquets envoyés, le jitter, etc.).

Ces informations permettent de contrôler les performances, par exemple diminuer le débit s'il y a des pertes de paquets, en changeant le débit de l'encodage, la résolution, en ignorant des images, etc.



RTP se scale parfaitement, car c'est un schéma multicast. RTCP moins, car le sender doit gérer tous les récepteurs, il peut être débordé par les arrivées de paquets. Il faudrait pouvoir adapter le rythme de paquets RTCP envoyés par les récepteurs. Plus il y aura de participants, moins il y aura de reports envoyés, afin de ne pas surcharger l'expéditeur.

Autre problème : l'hétérogénéité des récepteurs. Certains peuvent recevoir de meilleurs flux que d'autres, et donc demander un plus petit flux, ce qui pénalise les autres récepteurs. Il faut alors une méthode plus sophistiquée, par exemple définir des groupes de récepteur (low et high quality), ou bien donner un stream particulier à un récepteur.

Il existe trois types de paquet RTCP :

- les Receiver Report (RR) : la fraction de paquets perdus, le dernier numéro de séquence arrivé, le jitter moyen, etc ;
- les Sender Report (SR) : le SSRC du stream RTP, le temps actuel, le nombre de paquet envoyés, le nombre de bytes envoyés, etc ;
- les Source Description (SDES) : e-mail de l'expéditeur, son nom, le SSRC du flux associé. Cela permet donc un mapping entre le SSRC et l'utilisateur/le nom de l'hôte.

RTCP permet de synchroniser des flux différents au sein d'une même session. Les timestamps des paquets RTP sont liés à la vidéo et aux samples audio, et non à une horloge. Chaque paquet RTCP envoyé par l'expéditeur contient le timestamp du paquet RTP et l'heure (horloge) à laquelle le paquet a été créé. Ainsi, les expéditeurs peuvent utiliser cette association pour par exemple synchroniser un flux audio et un flux vidéo venant de périphériques différents.

RTCP va essayer de limiter son trafic à 5% de la bande passante disponible. Le problème est de connaître le nombre de récepteur (R) pour calculer le trafic RTCP.

Une solution est que l'expéditeur peut le calculer en fonction des reports RTCP qu'il reçoit, et ensuite l'envoyer à tous les récepteurs via ses RTCP. Le problème est que ce nombre est dynamique.

Autre solution : lorsqu'un récepteur doit envoyer un report, il n'a qu'à l'envoyer à tous les autres récepteurs, en broadcast à tous les membres du groupe (le récepteur est inclus dedans). A priori, cela génère plus de trafic, mais cela reste gérable (au plus $3 \times$ le trafic de l'expéditeur) car [...].

Ce broadcast peut aussi éviter du reporting inutile (par exemple, si un récepteur signale des pertes de paquets, un autre récepteur qui en aurait aussi recevra le paquet RTCP, mais n'en enverra pas un lui-même, car le paquet qu'il a reçu était déjà en broadcast).

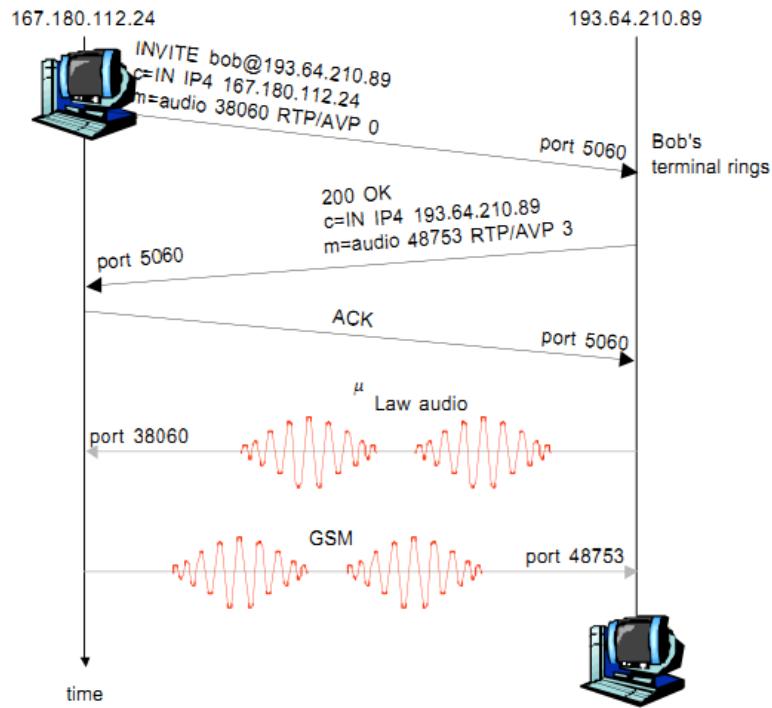
Il existe des techniques comme celle-ci qui permettent de diminuer le trafic. Le problème est que certains (voir trop de) récepteurs pourraient rester silencieux, ce qui fait que le comptage des récepteurs sera faux. On pourrait alors, en fonction des reports reçus et de leur fréquence, en déduire une probabilité d'avoir un groupe petit ou grand, et donc d'adapter le débit.

4.3.3 SIP - Session Initiation Protocol

Le but est d'atteindre quelqu'un et d'établir une communication. Les services qu'offre SIP :

- la mise en place d'un appel : un appelant doit pouvoir faire savoir à un appelé qu'on cherche à l'atteindre, avec un accord sur le type de média et l'encodage.
- la détermination de l'adresse IP d'une personne à contacter, on aurait un mapping entre un mnémonique et une adresse IP.
- la gestion d'un appel comporte l'ajout de nouveaux flux durant l'appel, le changement d'encodage durant l'appel, l'invitation de tiers, le transfert d'appel, etc.

SIP peut être utilisé avec TCP ou UDP. Par exemple, voici la mise en place avec TCP.



Pour négocier un codec, si une des parties ne possède pas un codec spécifié dans le message INVITE, elle envoie un message de type 606 qui liste tous les encodeurs disponibles. L'autre partie va alors renvoyer un message INVITE, avec un encodeur différent.

Une des parties peut rejeter un appel. Des médias peuvent être envoyés avec RTP ou un autre protocole.

```

INVITE sip:bob@domain.com SIP/2.0
Via: SIP/2.0/UDP 167.180.112.24
From: sip:alice@hereway.com
To: sip:bob@domain.com
Call-ID: a2e3a@pigeon.hereway.com
Content-Type: application/sdp
Content-Length: 885

c=IN IP4 167.180.112.24
m=audio 38060 RTP/AVP 0

```

La syntaxe des messages est celle des messages HTTP. Le sdp est le "session description protocol". Call-ID est unique pour chaque appel.

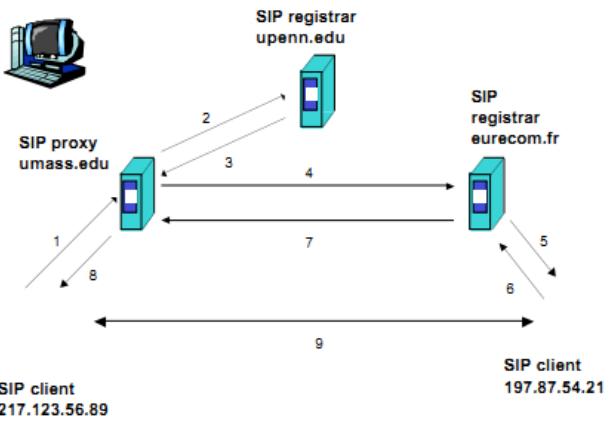
Le challenge est de mapper le nom d'un appelé vers son adresse IP, en sachant qu'il peut se déplacer. L'idée est d'utiliser des serveurs SIP, où les clients peuvent s'enregistrer (dans le SIP Registrar) via des messages REGISTER et communiquer leurs préférences d'utilisation de SIP.

On a, en plus de serveurs SIP d'enregistrement, des proxy SIP qui vont résoudre l'adresse (comme un DNS). Les proxy SIP vont ainsi retourner un message contenant l'adresse IP de la personne à appeler.

Exemple :

Caller `jim@umass.edu`
places a call to
`keith@upenn.edu`

(1) Jim sends `INVITE` message to umass SIP proxy. (2) Proxy forwards request to upenn registrar server
(3) upenn server returns redirect response, indicating that it should try `keith@eurecom.fr`



(4) umass proxy sends `INVITE` to eurecom registrar. (5) eurecom registrar forwards `INVITE` to `197.87.54.21`, which is running keith's SIP client. (6-8) SIP response sent back (9) media sent directly between clients

A noter que SIP utilise des ACKs.

Comparaison avec H.323

H.323 est un autre protocole de signalisation pour de l'interactivité en temps réel. Il est une suite complète de protocoles pour les conférences multimédia : signalisation, enregistrement, transport, codec, etc. Il s'agit d'un protocole venant du monde de la téléphonie.

Par rapport à H.323, SIP est plus simple et factorise le problème (ce n'est pas une réponse tout en un compliquée). SIP tire ses concepts d'HTTP.

Chapitre 5

Architectures et mécanismes QoS

5.1 Classes de service

5.1.1 Principes

IP est un modèle du type "ones-size fits all", il est sensé être utilisé pour toutes les applications. Le but est de retirer le meilleur de lui (qui est un service de type best effort).

Une alternative serait de proposer plusieurs classes de services, de partitionner le trafic en classes. On utiliserait pour cela le byte de ToS (type of service) dans le header IP, qui est redéfini et qui permet de déployer facilement des classes de service.

La granularité désigne la présence de services différents parmi les classes, et pas parmi les connexions individuelles.

Premier principe : marquage de paquets

Le marquage de paquets est nécessaire à un routeur pour distinguer les différentes classes, et donc appliquer des traitements différents.

Par exemple, on peut marquer des paquets de VOIP pour qu'ils soient protégés contre les bursts d'un flux FTP.

Deuxième principe : isolation des classes

Il faut fournir une protection/isolation d'une classe par rapport aux autres.

Une police pourrait par exemple, si trop de données d'une même classe sont envoyées, forcer la source à se limiter à son allocation de bande passante.

Les classes permettent aussi d'isoler les trafics, et même s'il y a des flux moins prioritaires, d'assurer qu'il y a un minimum d'équitabilité.

Troisième principe : utilisation efficace

Il faut utiliser les ressources autant que possible, tout en proposant une isolation.

Ainsi, l'allocation fixe de bande passante est inefficace si des flux ne l'utilisent pas.

5.1.2 Mécanismes de scheduling

Le scheduling (ordonnancement) détermine quel est le prochain paquet à envoyer sur un lien.

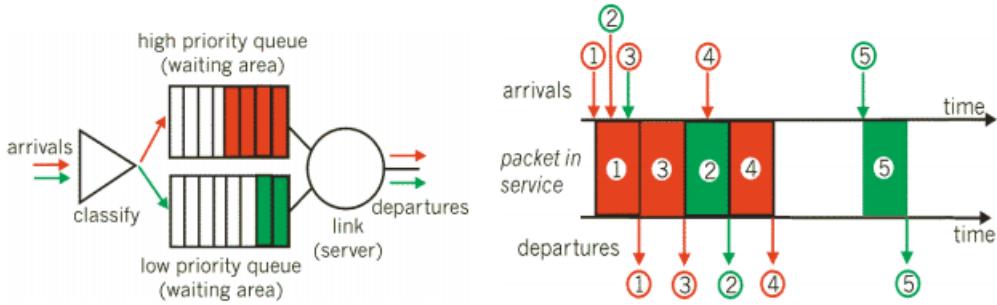
FIFO - FCFS

FCFS pour First Come First Served ; on envoie les paquets dans l'ordre d'arrivée dans la file. Si elle est remplie, on a plusieurs politiques de jet de paquets :

- tail drop : on supprime les paquets arrivant
- priority drop : on jette les paquets sur base d'une priorité
- random : on jette les paquets aléatoirement

Scheduling avec priorité

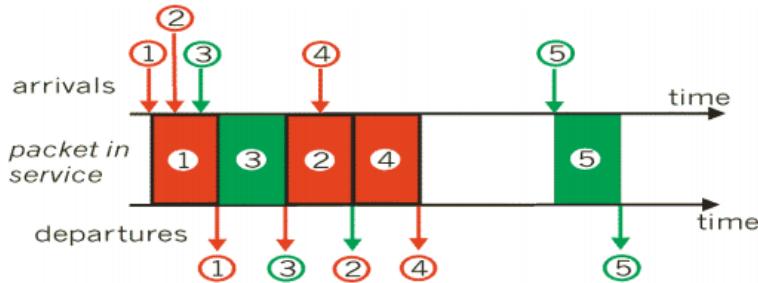
Cela consiste à transmettre d'abord les paquets de plus haute priorité dans une file. Chaque classe définit ainsi une priorité, sur base du ToS, de l'IP source/destination, des ports, etc.



Un tel système est non préemptif, c'est-à-dire qu'il n'interrompt pas une tâche qui a été commencée.

Round Robin

On crée des files, une pour chaque classe de paquet. Ensuite, on cycle dessus, en servant l'une après l'autre (pour autant qu'il y ait des paquets à servir).



Comme le priority scheduling, ce scheduling est work-conserving car, dès qu'il y a un paquet dans une des files, il est traité.

L'intérêt d'un mécanisme qui n'est pas work-conserving est par exemple de réduire/supprimer le jitter.

Loi de la conservation : la somme de la moyenne des délais dans les files (d_i) reçus par un ensemble de connexions, avec un poids dépendant du partage de la charge du lien (ρ_i) est indépendant de la discipline de scheduling.

d_i est le délai moyen des paquets pour une classe. $\rho_i \in [0, 1]$ est la proportion d'un flux par rapport aux autres. Si c est l'utilisation moyenne du lien de sortie, $\frac{c}{c} = \rho_i$.

Le scheduler est toujours actif, il ne s'arrête jamais (sauf si les files sont vides). On a donc que $\sum \rho_i \leq 1$ (= 1 : toujours occupé, < 1 : files parfois vides).

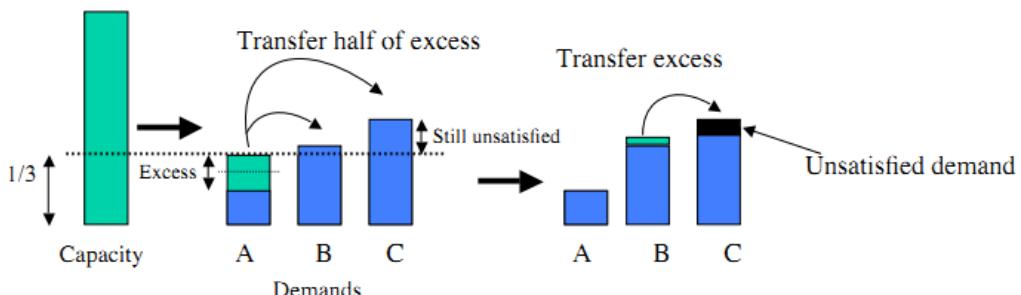
Si on veut mettre une priorité à un flux j , il faudra diminuer d_j . Vu que la somme des $d_i \rho_i$ est constante, et que les ρ_i sont aussi constantes, alors les d_i ($i \neq j$) augmenteront. Toute mise en priorité se fait au détriment des autres classes, quand on est en présence de ressources limitées.

La somme pondérée est un délai que l'on ramène à $d \times (\sum_i \rho_i)$ (la somme est toujours constante) avec d un délai qui caractérise tout le système, c'est le délai moyen d'un paquet, toutes classes confondues.

Scheduling avec équité max-min

Principe Une discipline de scheduling alloue une ressource. Cette allocation est équitable si elle satisfait l'équité max-min.

C'est une attribution récursive d'un ensemble de ressources ; chaque connexion n'a pas plus que ce qu'elle veut. Les petites demandes sont satisfaites, tandis que les grosses demandes peuvent ne pas l'être. Les excès, s'il y en a, sont partagés équitablement parmi les autres connexions.



En bleu, la demande de bande passante qui a été satisfaite.

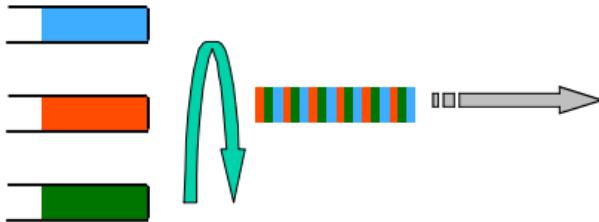
Un utilisateur ne peut pas tricher, dans le sens où il demande plus de bande passante pour être sûr d'avoir une bande passante minimale. L'effet inverse se produira.

L'algorithme de Water Filling permet de calculer cette équité ; on utilise le même principe, mais en prenant en compte tous les flux. Quand la capacité d'un lien est atteinte, on freeze les valeurs, et on continue sur les autres.

Il s'agit d'un algorithme centralisé qui nécessite la connaissance globale de tous les flux, de leurs chemins, et de toute la topologie.

Les algorithmes décentralisés pour résoudre le problème sont plus compliqués ; ATM ABR (available bit rate) en est un : un paquet est envoyé à travers les noeuds, et revient avec le fair share du flux (en ayant récupéré l'état des noeuds).

GPS On peut simuler ce mécanisme avec GPS (Generalized Processor Sharing) : on va visiter chaque file non vide, et on va servir une quantité infinitésimale de chacune.



On implémente bien l'équité max-min. Si tous les flux ont une demande, la même capacité sera allouée à tous les flux. Si un flux se termine ou est satisfait, les autres recevront la même quantité.

En théorie, l'algorithme est parfait, mais en pratique on ne peut pas couper des paquets en des quantités infinitésimales, du coup à un instant donné (sur une très petite échelle de temps) on a que 100% de la bande passante sera assignée à un flux. Si on utilisait un modèle à base de fluides, où le trafic en serait un, on aurait des quantités infinitésimales.

On va alors émuler l'algorithme, calculer le temps auquel un paquet doit être envoyé, et placer une borne sur le degré de non-équité.

Weighted Round Robin

On reprend Round Robin, mais en assignant différents poids aux flux, pour compenser des tailles de paquets différentes, mais aussi pour donner plus de services à d'autres flux. Chaque flux aura un poids w_i .

L'algorithme fonctionne bien avec des paquets de taille égale (ex : ATM, qui en plus a des petits paquets). Avec IP, on a des paquets plus gros que d'autres, on ne peut donc pas se fier à un découpage en paquets. Un découpage en bit/byte serait idéal, mais on ne peut pas le faire.

On va alors assigner des poids aux flux (cela revient à modifier la surface des récipients, dans l'algorithme du water filling). On a un weighted Max-min implémenté par un weighted GPS.

Si on a des poids différents et une taille de paquet fixe, à chaque visite d'un flux, on servira plus d'un paquet (le nombre est obtenu en normalisant les poids pour les rendre entiers).

Si on a des poids différents et des tailles de paquets variables, il faut normaliser les poids par la taille moyenne d'un paquet.

Problèmes :

- En pratique, le nombre de paquets calculé peut être très grand. Il faut tout un round (le temps de servir tous les flux) pour que le système soit équitable. Si le round est très long, sur des sous-intervalles, on a des flux qui seront privilégiés. On n'est pas équitable sur des petites échelles de temps.
- Il faut déterminer la taille moyenne d'un paquet. Un flux lui-même ne saurait pas prédire cette taille.

WFQ - Weighted Fair Queuing

Principe Pour chaque paquet, WFQ va le marquer avec un timing, qui est calculé par une émulation de GPS. Ce timing est le moment auquel le paquet doit être déservi si on utilisait GPS, soit le moment où on a poussé le dernier bit du paquet sur la ligne. Ensuite, le paquet est placé dans une file.

C'est ensuite au tour du scheduler de s'arranger pour coller le plus à la réalité. Un moyen de servir les paquets est de les servir par ordre croissant.

Problème : si on reçoit des paquets au même moment et de même poids, en théorie, ils seront envoyés au même temps t , en parallèle. En pratique, on doit en choisir un des deux, du coup cela ne veut pas dire que les paquets seront servis à leur temps GPS.

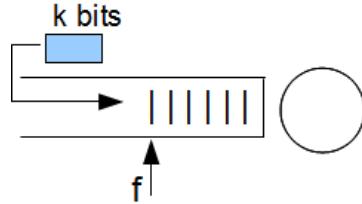
Le défi est de calculer le temps auquel envoyer le paquet.

FQ - Fair Queuing Dans un premier temps et sans considérer les poids, on va essayer d'émuler un Round Robin bit par bit. Donc, à chaque round, un bit de chaque paquet d'une file active (c'est-à-dire non vide) est traité. Pour un paquet de k bits, il faudra k rounds pour qu'il soit complètement servi ; ce nombre est appelé *round number* (R), on va l'utiliser pour le calcul du timestamp.

Pour une file vide, on assignera comme tag le finish number $FN = R + k$.



Si la file n'est pas vide, le finish number sera celui du dernier paquet dans la file auquel on ajoute k ; $FN = f + k$.



Avantages :

- on n'a pas besoin de connaître de faire des spéculations sur ce qu'il va se passer dans le futur. Une fois le tag assigné, il ne change plus.
- quand on assigne un tag, on ne se soucie pas des autres flux

On a toujours un manque d'équité à des très petites échelles de temps : au milieu d'un round, un flux pourrait avoir un bit traité, et pas un autre.

On émule GPS, donc une connexion devrait être considérée comme active si la file GPS émulée (pas la file WFQ) n'est pas vide. De ce fait, une file peut être considérée comme active même si elle n'a pas de paquets.

Par exemple, pour une file vide et deux connexions, avec des paquets de longueur 1 et un lien de débit égal à 1 bit/src, d'après GPS, on aurait traité 0.5 bit et la file ne serait pas vide, alors qu'avec le WFQ on aurait traité le bit. Il faut donc considérer ce qu'on aurait eu en théorie, avec GPS.

Le vrai test est donc de savoir si une file est vide ou non, pour choisir une des deux formules. Si une file contient des éléments, elle sera active si le *finish number* du dernier paquet est plus grand que R . Pour une file vide, on conserve le tag du dernier paquet envoyé, qui sera aussi comparé à R . Là où un paquet serait envoyé par un routeur, GPS serait encore en train de le servir.

On a alors, pour un paquet de longueur p ,

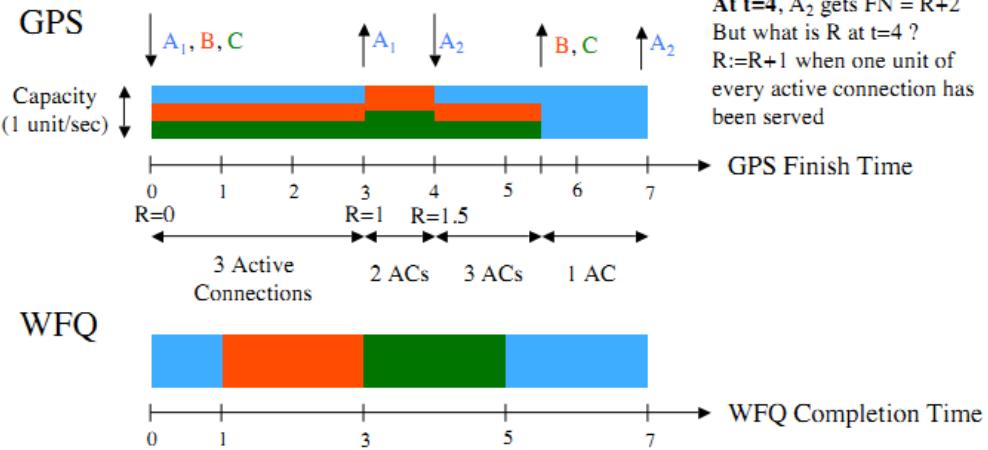
- s'il arrive d'une connexion active i , $FN = FN_i$ précédent + p
- s'il arrive d'une connexion inactive i : $FN = R + p$

En insérant les poids, on a pour un paquet de longueur p :

- s'il arrive d'une connexion active i , $FN = FN_i$ précédent + $\frac{p}{w_i}$
- s'il arrive d'une connexion inactive o : $FN = R + \frac{p}{w_i}$

Pour l'implémentation, on devra savoir si la connexion est active ou non, et pour cela on devra calculer R à chaque arrivée de paquet.

- Three connections : A, B and C
- On A : packet of size 1 at time 0, packet of size 2 at time 4
- On B and C : packet of size 2 at time 0

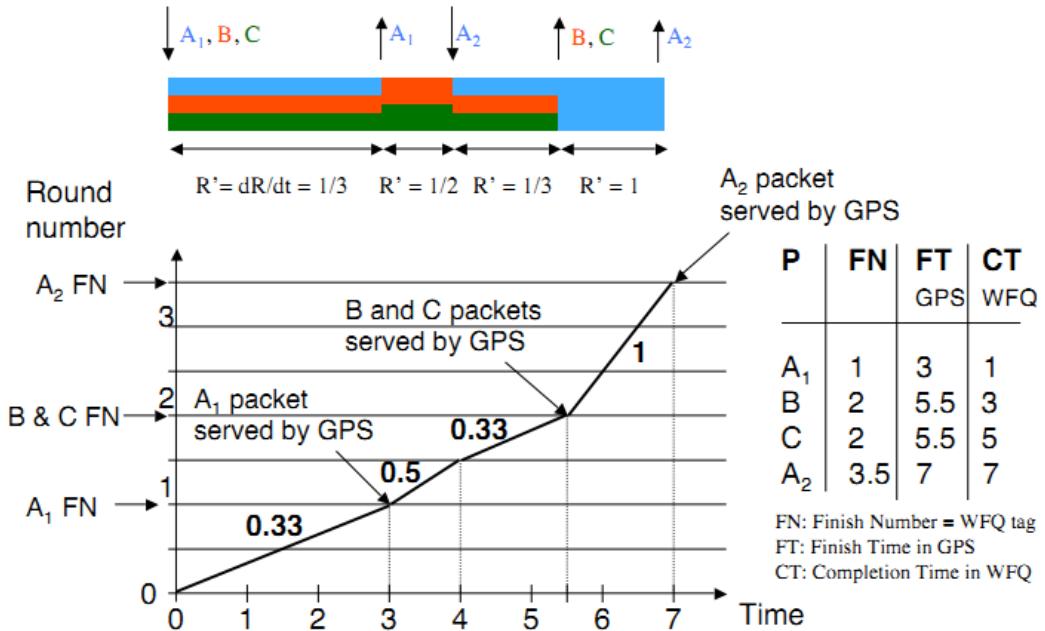


On a ajouté .5 à R car on a servi la moitié de la capacité à chaque flux (?). Le partage de la connexion varie, ce qui fait varier le taux d'incrémentation de R , la vitesse est inversement proportionnelle au nombre de connexion.

Calcul de R Naïvement, R pourrait être le nombre de rounds de service qui ont été complétés. Le problème vient du fait qu'un serveur peut ne pas avoir servi toutes les connexions en un round, ou que des nouvelles connexions peuvent s'ajouter en plein milieu d'un round. Une incrémentation constante n'est pas pensable.

Il faut redéfinir R comme une valeur réelle qui s'incrémente à un rythme inversement proportionnel au nombre de connexions actives (dans l'émulation GPS).

Avec ce changement, WFQ émule GPS au lieu de RR en bit par bit. On a des adaptations de débit instantanées avec le nombre de connexions actives.



Le futur est ainsi prédit en fonction du nombre de round. FN est calculable, pas FT.

Garantie de bande passante Une bande passante minimale est garantie, de même que le délai peut être borné.

Pour un lien de capacité C , pour i connexions actives de poids w_i et AC l'ensemble des connexions actives, la i ème connexion obtient une bande passante

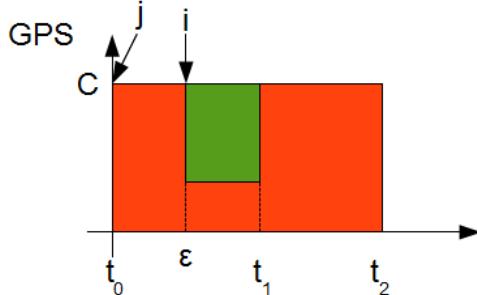
$$\frac{w_i}{\sum_{k \in AC} w_k} \times C \geq \frac{w_i}{\sum_{\text{tout } k} w_k} \times C$$

$\frac{w_i}{\sum_{\text{tout } k} w_k} \times C$ est la bande passante minimale garantie. La formule est également valable pour WRR. On a la propriété suivante du WFQ, avec M la taille maximale de paquet et C la capacité du lien :

$$CT_{WFQ} \leq FT_{GPS} + \frac{M}{C}$$

Supposons le pire des cas, avec des petits paquets très prioritaires et des gros paquets peu prioritaires : soit un flux i avec une taille de paquet minimale m et un flux j avec une taille maximale de paquet M . On pose $w_j \ll w_i$; $\frac{w_i}{w_i+w_j} C \approx C$.

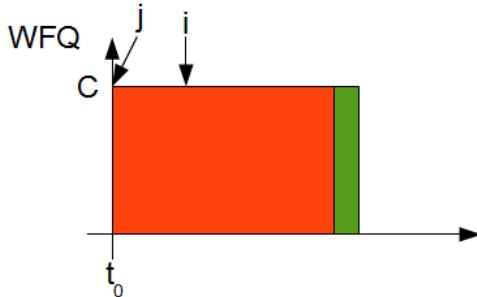
Avec GPS, les deux paquets sont servis simultanément, mais i est prioritaire.



On a $t_1 = \epsilon + \frac{m}{\epsilon}$ et $t_2 = \frac{m+M}{C}$. Dès lors,

$$FT_{j,GPS} = \frac{m+M}{C} \quad FT_{i,GPS} = \frac{w_i}{\sum w_k} \times \frac{m}{C} \approx \frac{m}{C}$$

Avec WFQ, j est servi en premier.



$$CT_{j,WFQ} = \frac{M}{C} \quad CT_{i,WFQ} = \frac{M+m}{C}$$

Le gros paquet ne verra pas de grosse différence, entre GPS et WFQ. Par contre, le plus petit aura une différence de $\frac{M}{C}$

DRR - Deficit Round Robin

FQ et WFQ sont en $\mathcal{O}(\log N)$ (N étant le nombre de files), dû à la structure de données à maintenir pour avoir l'ordre de sortie des premiers paquets de chaque flux.

Le DRR est une approximation de FQ, avec une complexité $\mathcal{O}(1)$, la même complexité que RR. Il peut gérer les paquets de taille variable sans connaître la taille moyenne, comme FQ.

On assigne une variable à chaque file, un deficit counter, qui quantifie le fait qu'une file n'a pas été traitée équitablement.

Si ce counter satisfait une condition (la file a été trop désavantagée), le paquet de tête sera traité ; si le déficit est plus grand que la taille p du paquet, on sert le paquet, et on diminue le déficit par la taille du paquet. Si le déficit est plus petit que p , on ne sert pas le paquet, et on augmente le déficit de p . On passe ensuite à la file suivante.

Cet algorithme est beaucoup plus simple, mais n'a pas toutes les propriétés des autres algorithmes (par exemple, il n'y a pas de borne supérieure sur le délai).

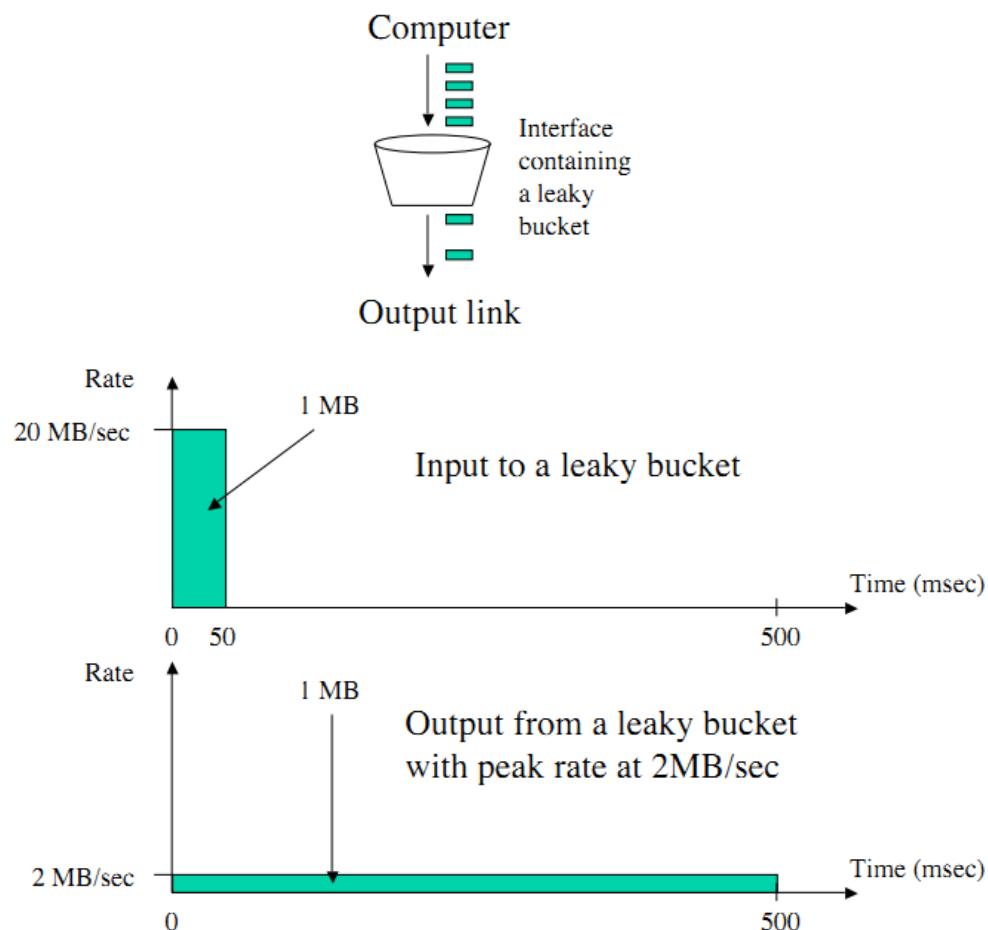
5.1.3 Mécanismes de shaping

On modèle le trafic, on s'arrange pour qu'il ne dépasse pas certaines quantités :

- Average rate : combien de paquets sont envoyés par unité de temps (à long terme). Il faut savoir de quel intervalle de temps on parle (car les moyennes de 100 paquets/s et de 6000 paquets/min sont les mêmes) ;
 - Peak rate ;
 - La taille de burst maximale, c'est-à-dire le nombre de paquets envoyés consécutivement à un peak rate.
- Le trafic qui sort d'une file subit ce modelage.
- Average rate : on ne dit pas comment on distribue les paquets. Par exemple, on pourrait envoyer les 6000 paquets dans les 10 premières secondes, et ne rien envoyer pendant 50 secondes. Cela ne suffit pas à shaper le trafic. On considère une fenêtre temporelle relativement grande.
 - Peak rate : idem que l'average rate, mais on considère une fenêtre temporelle beaucoup plus petite.
 - Maximum Burst size

Leaky Bucket shape

On utilise le modèle d'un seau troué.



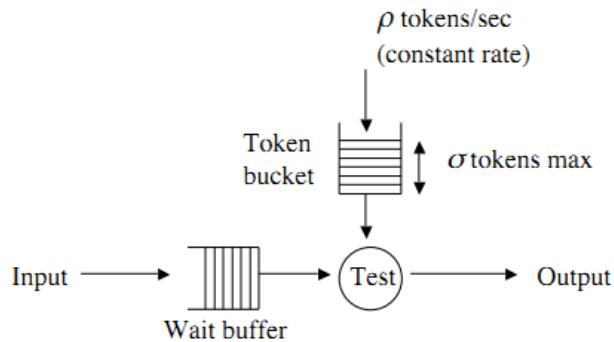
Il faut cependant prendre en compte que

- les flux de paquets ne sont pas des fluides
- chaque paquet est transmis à la vitesse du lien de sortie, donc à des très petites périodes de temps le débit de sortie vaut tout le temps la vitesse de sortie du lien
- les paquets ont une taille variable
- seul le peak rate est régulé, on aimerait bien limiter le burst size et l'average rate

Pour régler ces problèmes, on utilise le mécanisme du token bucket.

Token Bucket

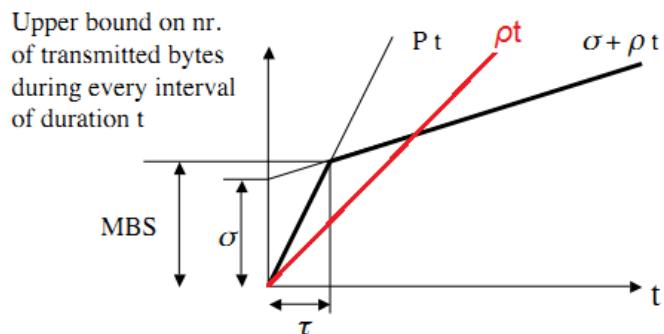
On dispose d'un seau de tokens qui se remplit à une vitesse ρ constante. Le plus grand nombre de token qu'il peut compter est σ , après quoi les nouveaux tokens sont jetés. Le seau est initialement plein (σ tokens).



Chaque token permet la transmission d'un certain nombre de bytes (par exemple 1 byte). Les paquets sont ainsi retardés dans un buffer d'attente, et ce tant qu'il n'y a pas assez de tokens dans le bucket pour les envoyer.

Au début de l'exécution, on va envoyer les paquets à la vitesse du lien, tant que le token bucket n'est pas vide. Ensuite, on ne peut plus envoyer qu'à une vitesse similaire à la vitesse d'arrivée des tokens (ρ).

Soit P la capacité en bytes/sec du lien, avec $P > \rho$. Si la capacité est plus petite que le débit d'arrivée des tokens, ces derniers n'ont plus d'intérêt.



On a

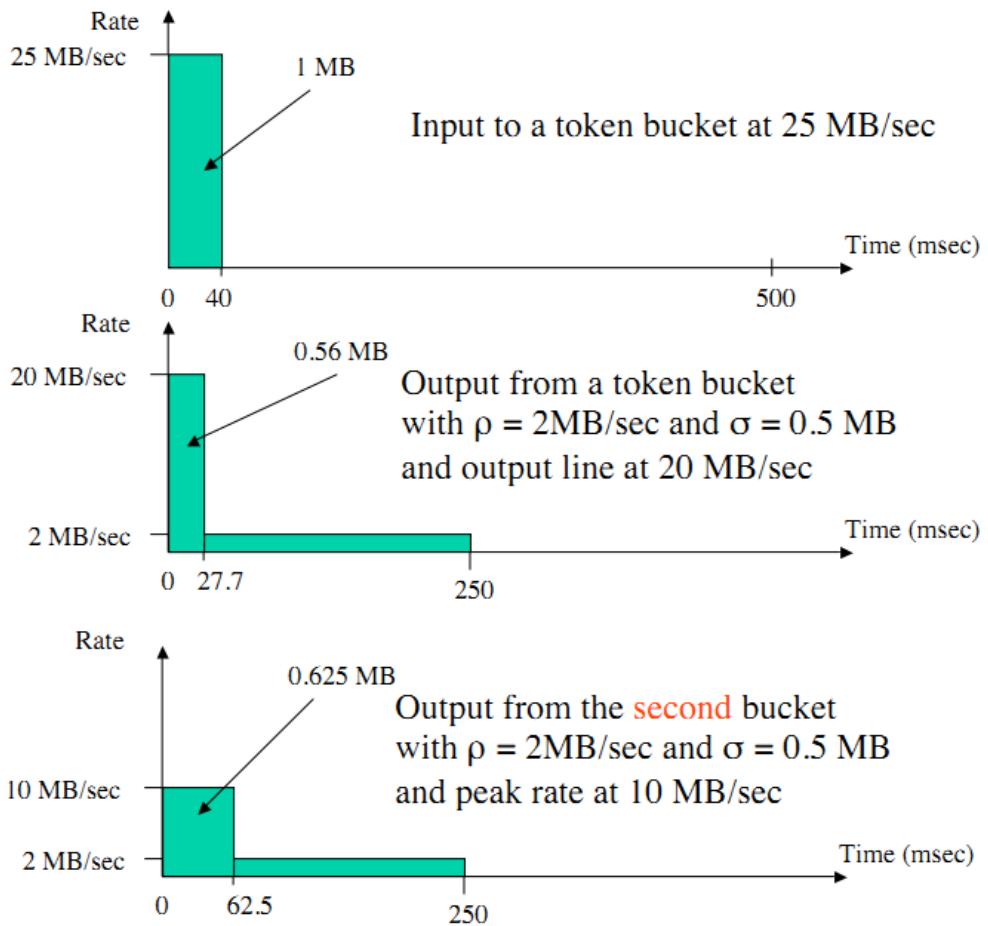
$$- \text{MBS le maximum burst size : } MBS = \sigma \frac{P}{P-\rho} > \sigma$$

$$- \tau \text{ la durée maximale de burst : } \tau = \frac{MBS}{P} = \frac{\sigma}{P-\rho}$$

On a $\sigma < MBS$, car des tokens s'ajoutent pendant qu'on les utilise dans le MBS.

Pour réguler le peak rate, on ajoute un second régulateur, avec des paramètres (ρ_2, σ_2) . ρ_2 sera le peak rate, σ_2 ne peut pas être nulle sinon un paquet de taille maximale M risque de rester indéfiniment dans le buffer. Il faut donc que σ_2 soit égal à M .

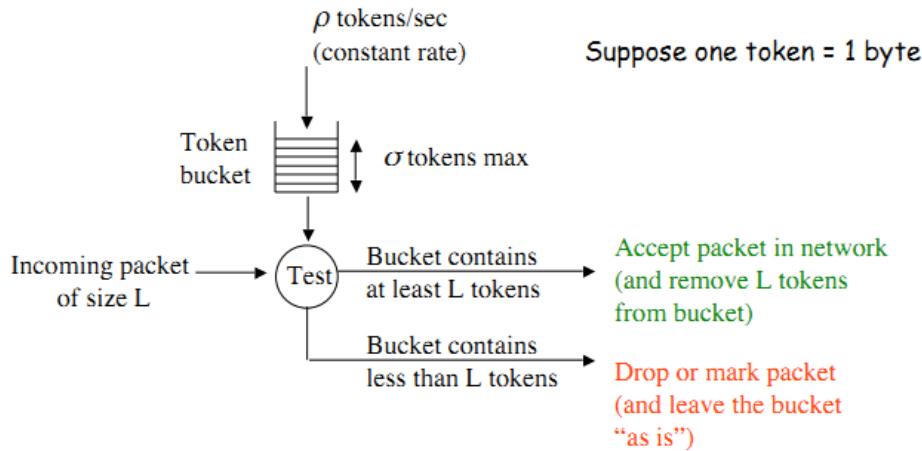
ρt permet donc de réguler le peak rate.



5.1.4 Mécanismes de policing

On vérifie que le trafic a bien été shapé. Un policer choisira si un paquet peut entrer dans le réseau, sinon il le jette.

Le policer n'a pas pour but de changer la forme du trafic, il ne fait que vérifier si le "contrat" entre la source et le destinataire est respecté. Cette vérification s'effectue généralement à l'entrée (ingress point) d'un réseau.



Quand un paquet est jeté, aucun token n'est utilisé, vu qu'il n'est pas envoyé sur le réseau (et donc aucune ressource n'est utilisée).

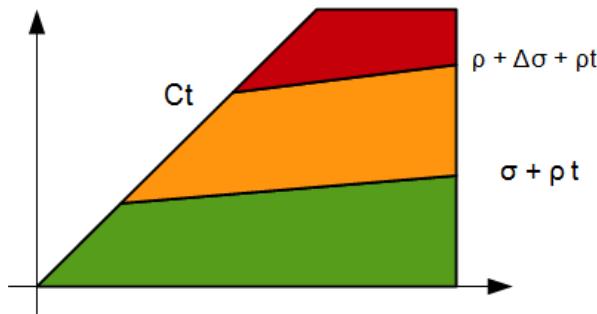
On peut s'arranger pour que les paquets à jeter soient quand même envoyés dans le réseau, même s'ils ne satisfont pas le contrat. Cependant, si des paquets doivent être jetés dans le réseau, ce seront les premiers éliminés. Pour les différencier des paquets qui ont respecté le contrat, ils sont marqués.

Le token bucket n'est pas modifié, on ne consomme pas de tokens. Si on le fessait, cela réduirait l'enveloppe pour les paquets admis à entrer dans le réseau.

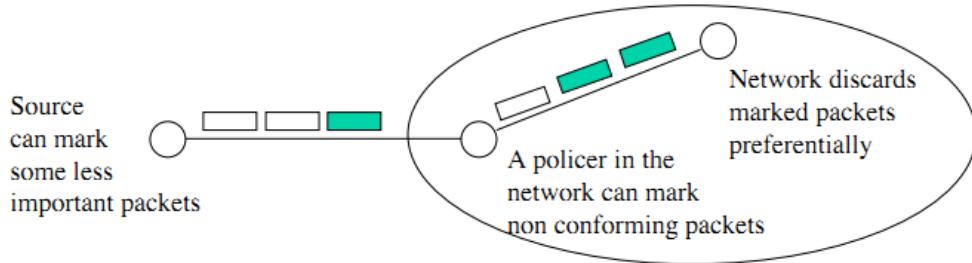
Si on a plusieurs critères, comme pour le shaping, il faut mettre en cascade plusieurs token buckets pour le policing : le premier servira à jeter les paquets qui excèdent le peak rate, le second ceux qui excèdent l'average rate et le maximum

burst size.

On niveau des marquages, on peut par exemple marquer les paquets en vert s'ils respectent l'enveloppe, en orange s'ils ne dépasse pas trop le maximum burst size, en rouge sinon.



5.1.5 Stratégies de jet de paquets



Les paquets marqués seront donc jetés en premiers, ils sont considérés comme non prioritaires. En les faisant entrer dans le réseau, on ne peut pas prédire ce qu'il va se passer.

Des policiers dans le réseau même peuvent aussi marquer des paquets, et peuvent avoir des marqueurs différents des marqueurs de la source. La source peut aussi mentir et mal marquer les paquets.

Avantages d'une priorité dans le dropping

- si le réseau a une capacité éparsillée, tout le trafic est géré.
- lors de congestion, la charge est automatiquement diminuée.

Désavantage : la séparation des priorités au sein d'un même flux est compliquée pour une source, de plus rien n'empêche de marquer tous les paquets comme prioritaires. Une police doit être mise en place pour définir le marquage.

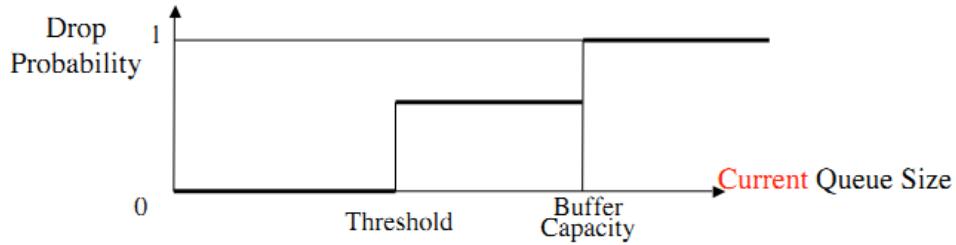
Cas particulier d'ATM : perdre une cell fait perdre tout le paquet qui est formé par elle, donc on peut jeter toutes les cells du paquet, car elles ne serviront à rien (on ne saura jamais reconstituer tout le paquet).

Une stratégie idéaliste serait de jeter les paquets d'hôtes à proximité en premier, car ils n'ont pas trop utilisé de ressources dans le réseau. On ne peut cependant pas le faire pour Internet à cause du TTL qui diminue, ce qui fait que la valeur initiale n'est pas connue.

Early drop

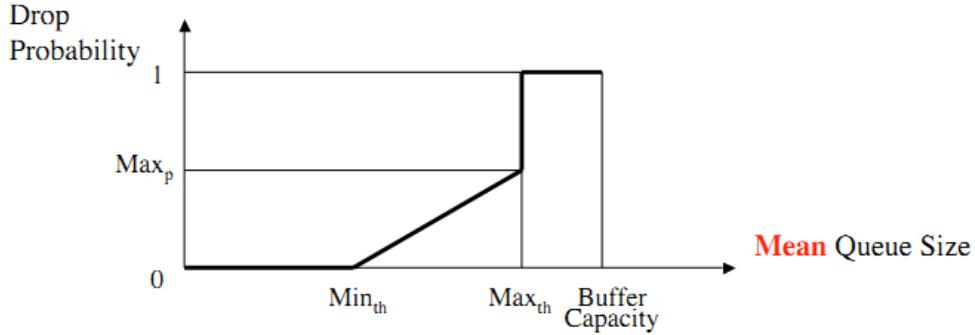
L'idée est de jeter des paquets même s'il reste de l'espace libre. Cela permet de signaler aux sources qu'il y a un début de congestion et qu'il faut ralentir les débits ; les connexions TCP vont diminuer le débit, ce qui permettra (peut-être) d'éviter la congestion. Ainsi, les sources coopératives auront des délais plus courts, tandis que les sources non coopératives subiront de sévères pertes de paquets.

Lorsque le buffer a atteint une certaine capacité, les paquets sont dropés avec une certaine probabilité.



Problème : on pourrait avoir des fluctuations importantes qui passent sur le seuil, il faudrait une stratégie plus progressive. De plus, on se base sur la taille courante (instantanée) du buffer.

RED - Random Early Detection



On utilise la taille de la file **moyenne**. Par exemple, si on envoie un burst, si on se base sur la taille instantanée, il y aura une grosse variation et un drop est possible. Si on utilise la moyenne, la variation ne sera pas grande. Seule une source qui envoie beaucoup de données (gros burst) constamment sera pénalisée ; TCP ne sera pas touché car généralement il envoie des petits bursts.

Le seuil maximal est plus petit que la capacité, ce qui est logique car si on se situe près de la capacité, cela signifie que la moyenne est très grande (fluctuations élevées), donc très proche de la congestion.

La moyenne a donc une borne (max_{th}), mais la taille instantanée peut la dépasser.

RED augmente les performances d'un réseau composé de sources TCP :

- les pertes surviennent aléatoirement (et pas par burst comme avec un jet de type drop-tail), du coup TCP reste en mode congestion avoidance (alors que des pertes par burst fait rentrer TCP en slow start). De plus, il y a moins de synchronisation entre les sources TCP, elles ne ralentissent pas toutes en même temps
- les pertes surviennent avant que la congestion ne soit réellement présente, du coup TCP anticipe plus
- les files des buffers sont gardées petites, il y a moins de queuing delay, un plus petit RTT, et donc un meilleur débit du côté de TCP.

Si on ne perd qu'un seul paquet dans un burst, c'est bon pour TCP car si on perdait tout le burst, le seul moyen de retransmettre est d'attendre la fin du timer (les ACK ne viendront jamais), ce qui implique que le throughput sera mis à 1.

De plus, si on perdait systématiquement les paquets après une certaine capacité sur un buffer, toutes les connexions TCP qui l'utilisent vont diminuer (car congestion), puis remonter, puis redescendre lorsque la capacité maximale est utilisée, etc. Cette synchronisation est néfaste, RED va la casser.

Grâce à RED, la capacité moyenne est petite (comprise entre les seuils min et max), ce qui diminue le queuing delay, donc des RTT plus petits, et donc un débit TCP plus grand (car le débit est proportionnel à son inverse : $\frac{MSS}{RTT\sqrt{p}}$).

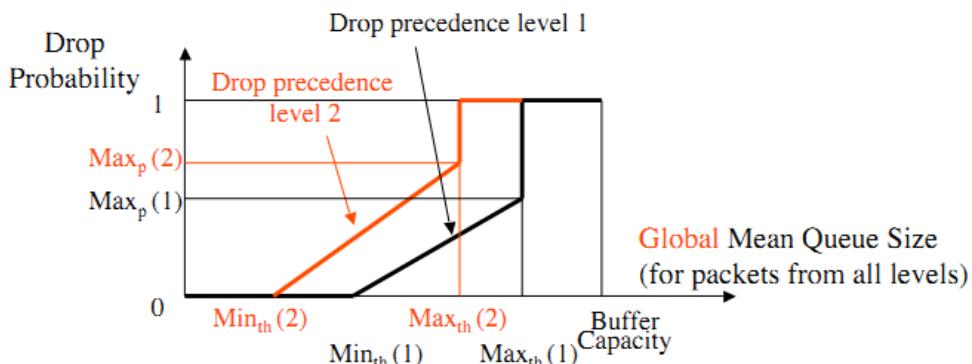
Cependant, RED est difficile à paramétrier, et des pertes de paquets sont parfois inutiles (cela réduit le débit de TCP pour rien), notamment dans les cas où il n'y a pas de congestion.

ECN - Early Congestion Notification

C'est une variante de RED ; on ne va pas jeter les paquets, mais les marquer. Ainsi, le destinataire des paquets saura qu'il y a de la congestion, et va prévenir la source via les ACK (car IP est connectionless), qui indiqueront le niveau de congestion en plus de signaler jusqu'à quel byte on a reçu les données.

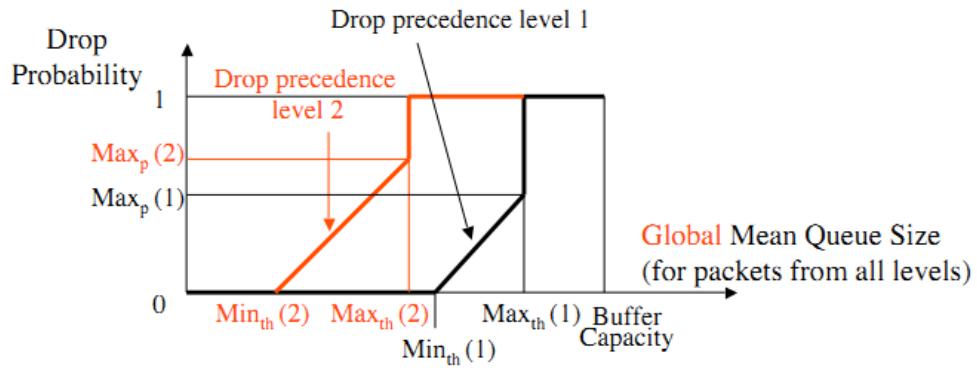
WRED - Weighted RED

On est plus agressif sur les paquets marqués que sur les autres. On peut ainsi définir n niveaux de drop, n couleurs différentes avec des seuils particuliers.



Les paquets admis ne sont pas nécessairement protégés des paquets marqués : le fait d'avoir ajouté le trafic des paquets marqués pousse la courbe de probabilité.

Solution : décalage des paramètres.



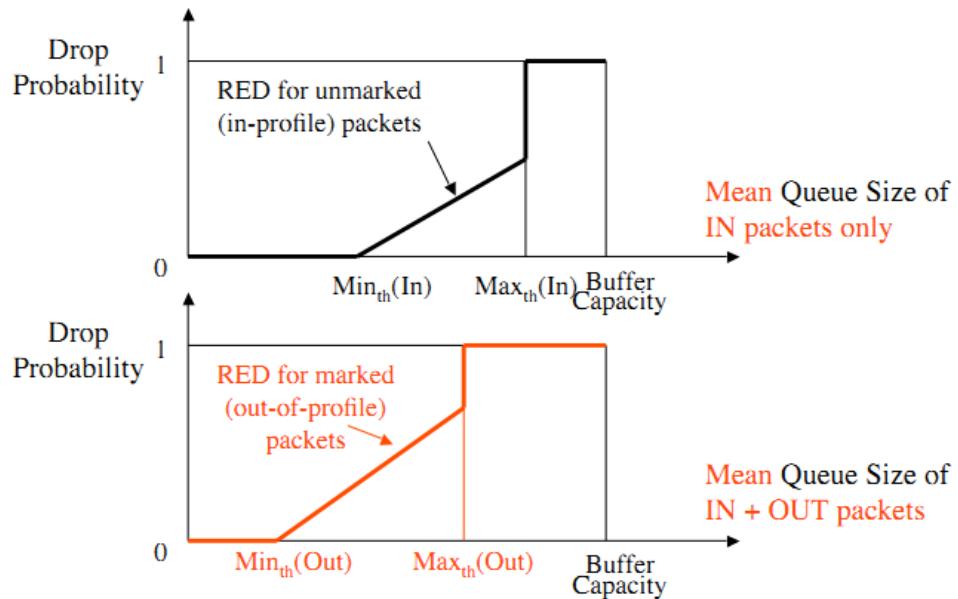
Un trafic au niveau n est protégé des trafics de niveau supérieur si les charges du trafic des niveaux supérieurs n'a que des effets mineurs sur le taux de perte au niveau n .

WRED offre une telle protection si $\max_{th}(n+1) \leq \min_{th}(n) \quad \forall n$

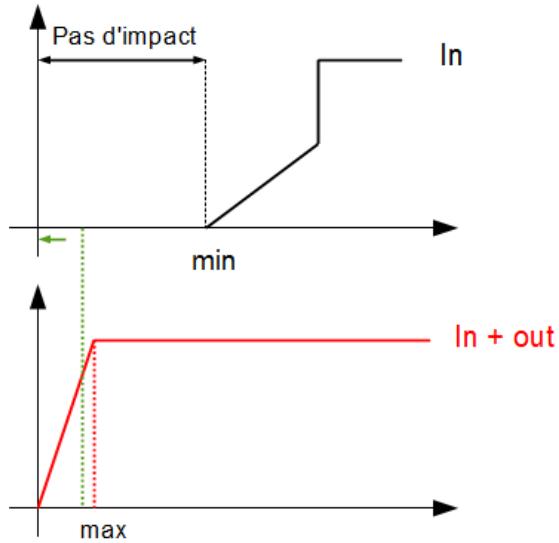
Un trafic de paquets rouges pénalise moins le trafic des paquets verts, mais contribue toujours à augmenter la moyenne de la taille du buffer. Sans trafic rouge, un trafic vert commence à être perdu à partir de $\min_{th}(1)$. Avec un trafic rouge élevé, le trafic vert commencera plus vite à être perdu. On n'a donc pas d'isolation parfaite entre les deux.

RIO - RED with In and Out

On définit deux courbes de probabilité, mais avec des axes X différents. Le trafic vert est In car "in profil", le rouge est Out. Chacun de ces trafics sera assigné à une des courbes.



Le système est amélioré avec ces deux courbes avec des bons paramètres : il faut que $\max_{th}(Out) < \min_{th}(In)$.



Avec les mêmes paramètres, RIO est mieux que WRED avec deux niveaux dans la protection des paquets In des paquets Out (car ...).

Avec RIO, le taux de drop des paquets In ne dépend pas des paquets Out dans le système, excepté dans des circonstances exceptionnelles (overflow du buffer). Donc RIO fournit une protection du trafic (traffic sheltering).

Si $\max_{th}(Out) \leq \min_{th}(In)$, alors les paquets In peuvent être jetés si tous les paquets Out sont déjà tous jetés.

Un cas particulier pourrait être :

- une file de type drop tail (courbe en step) pour les paquets In, et
- une file RED pour les paquets Out.

Avec des trafics élevés, WRED et RIO peuvent diminuer le trafic à des hauts taux de jet.

On a une certaine flexibilité, par exemple on pourrait implémenter un mécanisme qui marque les paquets verts et qui drop les paquets rouges.

5.1.6 IETF Differentiated Services

On va définir des classes de services (Platinum, Gold, Silver).

Le système a été pensé pour la scalabilité : les fonctions sont basées sur les classes et non sur les flux. On peut avoir des millions de flux, on a toujours 3 classes à traiter ; du coup, on n'a plus de signaling (à part celui qui définit le chemin pour de l'unicast et le multicast).

IETF définit des blocs qui peuvent être utilisés pour créer une classe de service, on ne définit rien d'autre.

Dans une architecture Diffserv, on définit deux types de routeur :

- les edge routers : ils gèrent le trafic par flux et marquent les paquets de deux manières :
 - ils leur assignent le code d'une classe
 - ils marquent les paquets comme In ou Out

Le management est par flux, car il y en a moins sur les bords que dans le réseau même.

- les core routers : ils gèrent le trafic par classe. Ils bufferisent et ordonnancent les paquets sur base du marquage, et mettent en avant les paquets marqués comme In dans les stratégies de drop. Ils ne font que lire les classes assignées aux paquets.

Edge routers

Le paquet est marqué dans le champ ToS en IPv4, ou dans le champ Traffic Class en IPv6 ; ces bytes sont renommés DS bytes.

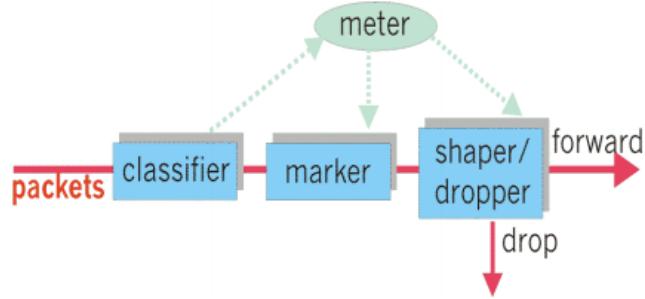


6 bits sont utilisés pour le DSCP (Differentiated Service Code Point), les deux autres sont inutilisés (CU - currently unused). Un usage possible serait dans la stratégie de drop ECN (qui doit signaler la congestion).

Ainsi, on doit coder la classe et la couleur dans les 6 bits. Les deux derniers bits sont utilisés par ECN : le premier sert à indiquer si le paquet peut être marqué ou non, et le second est le marquage.

Un utilisateur accepte de limiter son trafic selon un profil de trafic : c'est le concept de SLA/SLS (Service Level Agreement/Specification).

Le SLA, défini par un ISP, assigne des quotas de classe à une source en fonction du contrat passé. Ainsi, cela évite qu'une source n'envoie que des paquets platinum par exemple. Le SLA se trouve entre le shaper/dropper et le policer.



On a les trois composantes :

- Classifier : classification basée sur le header du paquet.
- Marker : assignation DSCP, downgrading, réassignement.
- Shaper/dropper : retardement ou jet des paquets s'ils ne sont pas conformes.

Core routers

Le PHB (Per-Hop Behavior) décrit l'action qu'effectuera le routeur en fonction de la classe d'un paquet, sous forme de service. Cela définit de manière abstraite le traitement à effectuer, mais rien de concret (aucun mécanisme n'est défini).

On distingue trois types de forwarding défini par PHB :

- EF - Expedited Forwarding : trafic à haute priorité, avec une file FIFO spécifique. Le taux de départ des paquets d'une classe est égal ou excède un débit spécifique, même à des petites échelles de temps ; cela équivaut à un lien logique avec un minimum de garantie de débit.

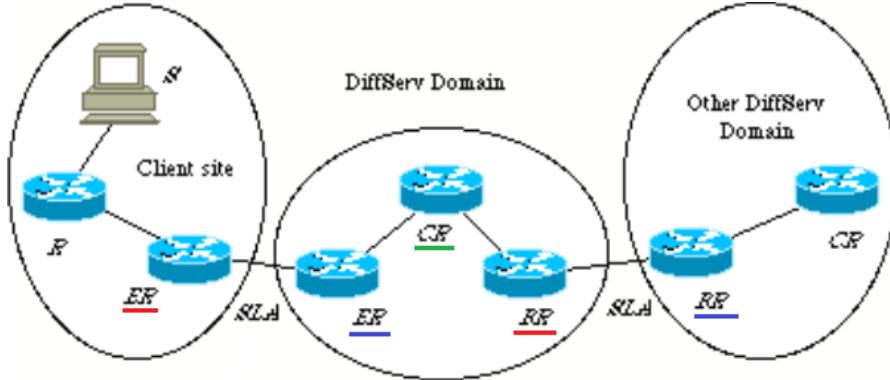
Pour qu'il n'écrase par les autres classes, il faut qu'il soit contrôlé. Ainsi, au niveau du policing, le paquet est droppé s'il n'est pas vert.

- AF - Assured Forwarding : moins de pertes de paquet que le best effort. Les flux de différentes classes sont isolés, mais les flux d'une même classe sont en compétition. Les garanties de bande passante s'appliquent aux classes, pas aux flux individuels.

Le standard définit seulement qu'il y a 4 classes (AF1, AF2, AF3, AF4), mais pas ce qu'on doit mettre dedans. Chaque AF garantit un minimum de bande passante, il y a une file FIFO par AF. Elles peuvent être utilisées pour isoler les types de trafic les uns des autres (par exemple ne pas mettre TCP et UDP dans la même classe). Chaque AF peut avoir trois niveaux de jet de paquet, représentés par 3 couleurs (il n'y a pas de couleur dans EF car si un paquet n'est pas conforme, il est jeté).

- BE - Best Effort
- LBE - Less Than Best effort

Interconnexion de domaines typique



La source S va donc émettre des paquets. Elle pourrait les marquer avec un DSCP approprié, mais c'est dangereux dans le sens où elle pourrait en abuser (en marquant tout en EF par exemple). Il vaut mieux que le premier routeur, R, classifie les paquets et assigne le DSCP en accord avec une police de QoS. Cette classification est multi-field (MF), mais n'est faite qu'une fois.

Les routeurs rouges vont appliquer un mécanisme de shaping sur les flux agrégés avant de les faire entrer sur un autre domaine, de manière à ce qu'ils remplissent les conditions de SLA en place. A noter que le routeur R peut déjà l'avoir fait sur des flux individuels.

Il est logique que ce soit ER qui marque les paquets (si ce n'est pas déjà fait, par R par exemple), car tout le flux du réseau passe par ce point. Du coup, le marquage tient compte de l'entièreté du trafic, tout comme le SLA.

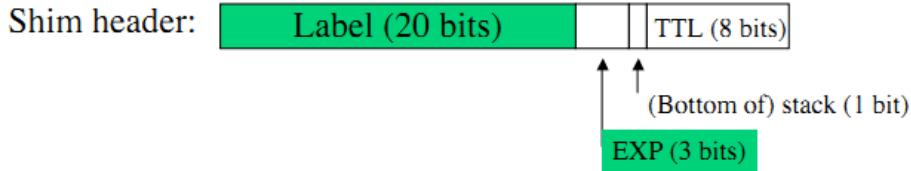
Ce sont les routeurs bleus qui vont filtrer le trafic entrant et appliquer un mécanisme de policing, afin d'être en accord avec le SLA établit. Ils peuvent

- remarquer le paquet,
- jeter des paquets non conformes, ou
- diminuer l'importance de paquets non conformes (plus de chance de drop).

Le routeur vert n'effectue que du packet forwarding et du scheduling/dropping par classe ; il ne fait que lire le DSCP.

5.1.7 Les services différenciés et MPLS

Les paquets IP sont encapsulés dans des frames MPLS, du coup DSCP est invisible aux yeux des LSRs. On aimerait appliquer le bon PHB aux frames MPLS.



On a deux possibilités :

- E-LSP (EXP LSP) : on peut utiliser les trois bits EXP, mais le problème est que DSCP est défini sur 6 bits ; on ne peut définir que 8 PHBs par domaine. On doit donc faire un mapping qui fait perdre des propriétés (par exemple définir EF, AF1 3 couleurs, AF2 3 couleurs et BE ; AF3 et AF4 sont mappés sur AF1 et AF2).
- L-LSP (Label LSP) : on utilise les labels ; une relation label-PHB doit être mise en place.

Le préséance de jet est toujours placée dans le champ EXP, et cela consomme plus de labels.

On va encoder la couleur dans EXP, car il ne faut pas changer le level : on peut dégrader un paquet, mais pas changer sa route, au risque d'avoir des paquets désordonnés.

5.2 Garanties de QoS

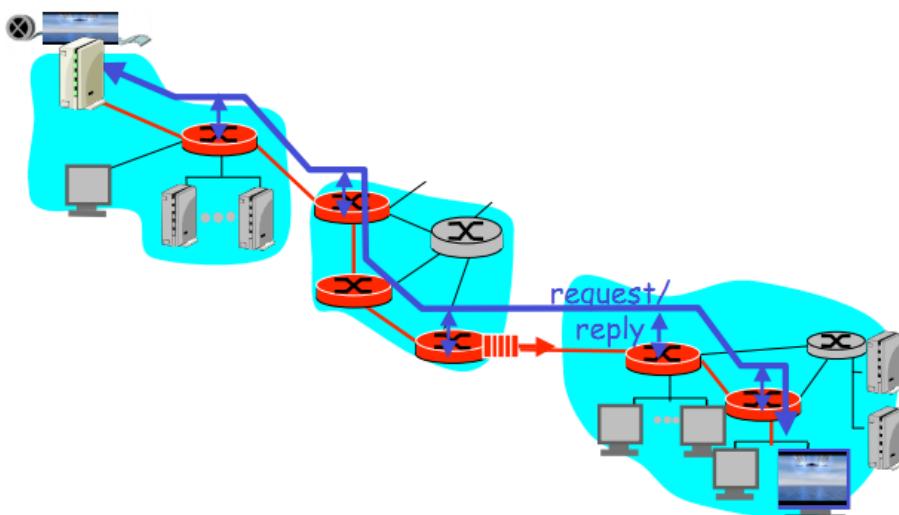
5.2.1 Principes

Quatrième principe

Call Admission : lorsqu'un flux déclare ses besoins, le réseau peut bloquer l'appel s'il ne peut les satisfaire.

Cela vient du principe qu'on ne peut pas utiliser un lien au delà de sa capacité.

On va mettre en place un mécanisme de réservation de ressources. Il y aura un appel à la mise en place, et de la signalisation (avec RSVP). Il y aura une déclaration de QoS, ce qui sensibilisera les mécanismes de scheduling.



IETF Integrated Services

Il s'agit d'une architecture qui offre des garanties de QoS dans les réseaux IP, pour des sessions individuelles. Il y a une réservation de ressources, les routeurs maintiennent un état (comme avec les circuits virtuels) de ressources allouées.

La question est de savoir si des nouveaux flux peuvent être admis avec les garanties de performances, et ce sans violer les garanties de QoS établies pour les flux admis.

5.2.2 Garanties de bande passante

Soit un lien de capacité C . On considère qu'un flux est admis si et seulement si la somme de tous les flux n'excède pas C .

On va assigner un poids w_i à un flux i . Il aura ainsi une certaine bande passante B_i . Par exemple, avec WFQ, on aura

$$B_i = \frac{w_i}{\sum_{\forall k} w_k} \times C$$

et même plus si les autres flux ne sont pas actifs.

L'assignation des poids est un mapping entre des demandes B_i et w_i . On a que $w_i = \alpha B_i$.

Par exemple, supposons que $w = 1$ correspond à 1 kbps ($\alpha = 0.001$) ; un flux de 1 Mbps se voit assigné $w = 1000$. Si le flux est seul, il aura $\frac{1000}{1000} \times C = C$. Si tous les flux admis valent collectivement C , la somme des poids vaut αC , et le flux en question aura $\frac{1000}{\alpha C} \times C = \frac{1000}{\alpha} = \frac{1000}{0.001} = 1\text{ Mbps}$.

5.2.3 Garanties de délai

Théorème de Parekh-Gallager

Soit un flux qui a des poids définis à chaque scheduler WFQ sur son chemin, de manière à ce que g soit la plus petite bande passante réservée.

Supposons qu'il soit régulé par un token bucket de paramètre (ρ, σ) , avec $g \geq \rho$.

Supposons que le flux passe à travers K schedulers WFQ, où le k ème scheduler possède un débit $r(k)$ égal à la capacité du lien. On a $g \leq r(k)$.

Soit le plus gros paquet admis dans le réseau de taille P et le plus gros paquet admis par le flux de taille M .

On a la formule de Parekh-Gallager, qui borne le délai dans le réseau :

$$\text{délai end-to-end} \leq \text{délai de propagation} + \frac{\sigma}{g} + \sum_{k=1}^{K-1} \frac{M}{g} + \sum_{k=1}^K k = 1K \frac{P}{r(k)}$$

$\frac{\sigma}{g}$ est le délai obtenu par le dernier paquet d'un burst de taille maximale σ et arrivant à une file déservie à un débit g . Cela comprend le délai de queuing et de transmission.

C'est un délai car on a une quantité de bits divisée par une bande passante (bits par seconde). C'est le temps qu'il faut pour que tous les bytes soient traités dans le buffer ; dans le pire des cas, c'est le dernier paquet du burst. C'est le terme principal ; il serait le seul si les paquets étaient des fluides.

A noter que ce terme n'apparaît qu'une fois. En effet, si le délai survient au premier noeud, les noeuds suivants ne recevront pas de burst. Les buffers sont en série ; quand un premier paquet est traité par un deuxième buffer, le deuxième paquet entre dans le deuxième buffer.

Les autres termes permettent de corriger le fait qu'il y a des bufferisations et des temps de transmission à chaque noeud.

Ainsi, les $K - 1$ sous-termes sont les délais de transmission pour un paquet de taille M dans un noeud déservi par un scheduler GPS (où les paquets arrivent dans des files GPS inactives à un débit g).

Dans le pire des cas, on envoie le plus gros burst (M) et ensuite coller à la limite (g).

Le dernier terme fait correspondre la borne au WFQ (sans cela, on a le GPS). A chaque noeud k , le paquet peut être transmis, dans le pire des cas, $\frac{P}{r(k)}$ secondes après son temps GPS théorique.

Ce théorème signifie que

- il est possible, pour des WFQ, de fournir une borne sur le délai end-to-end,
- plus g est grand (donc plus de bande passante), plus le délai sera court, et
- plus on envoie en burst, plus on a des fluctuations dans les buffers, et la possibilité d'avoir des grands délais.

Au niveau des flux régulés par des token bucket, les autres schedulers assurent que le délai end-to-end satisfait aussi la formule

$$\text{délai end-to-end} \leq \frac{\sigma}{g} + \frac{C}{g} + D$$

Le problème est que, pour avoir une borne, il faut choisir g :

- plus la borne sur le délai est petite, plus g doit être grand (quand σ est fixe)
- un grand g signifie l'exclusion de plus de compétiteurs pour le lien
- g peut alors devenir très grand, dans certains cas plusieurs fois le peak rate de la source.

De ce fait, les sources doivent être régulées par un leaky-bucket.

Les schedulers en WFQ couplent une allocation de bande passante et un délai ; un délai petit nécessite une grande bande passante. On a un gaspillage de bande passante pour les sources qui nécessitent des petits délais avec une petite bande passante (par exemple la voix). Ce n'est pas un problème quand plusieurs flux sont agrégés dans la même classe (et donc dans la même file).

5.2.4 Réservation de ressources - RSVP signaling

A l'origine, il n'y a pas de protocole de signaling dans IP ; on a seulement des routeurs connectionless qui forwardent par l'IP.

On a un nouveau prérequis : la possibilité de réserver des ressources sur le chemin d'un paquet, pour du QoS pour des applications multimédia.

Ainsi, pour une session, on doit

- déclarer les services QoS que l'on désire, ce sont les R-spec, et
- caractériser le trafic qui sera envoyé, ce sont les T-spec.

Un protocole de signalisation est nécessaire pour communiquer les R-spec et les T-spec aux routeurs (où une réservation est nécessaire), c'est le rôle de RSVP.

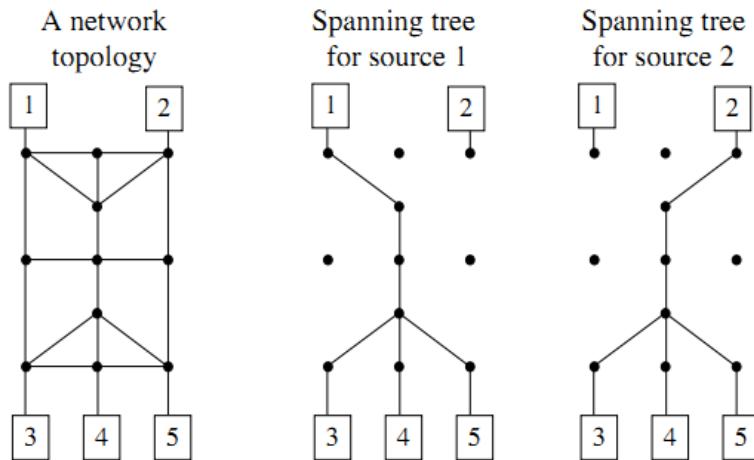
On peut distinguer deux types de protocole :

- Les data planes : couches réseaux qui interagissent directement avec les données utilisateurs, par exemple IP.
- Les control planes : couches en arrière-plan, qui définissent les états (par exemple RSVP, un algorithme de routage).

RSVP est de type control plane, son but est avant tout de supporter les applications multicast avec une réservation de ressources. On est donc dans le cas où on peut avoir plusieurs sources et récepteurs. Cela n'a rien à voir avec le routing multicast, qui est géré par exemple par PIM, un protocole de type control plane.

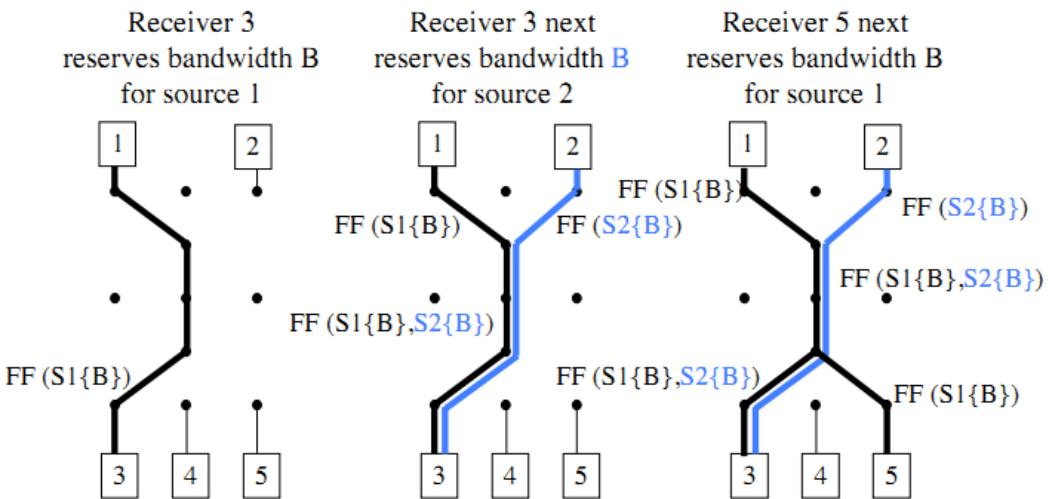
On définit plusieurs styles de réservation :

- une seule réservation partagée, pour toutes les sources,
- une réservation distincte par source, ou
- une réservation distincte partagée par un ensemble de sources.

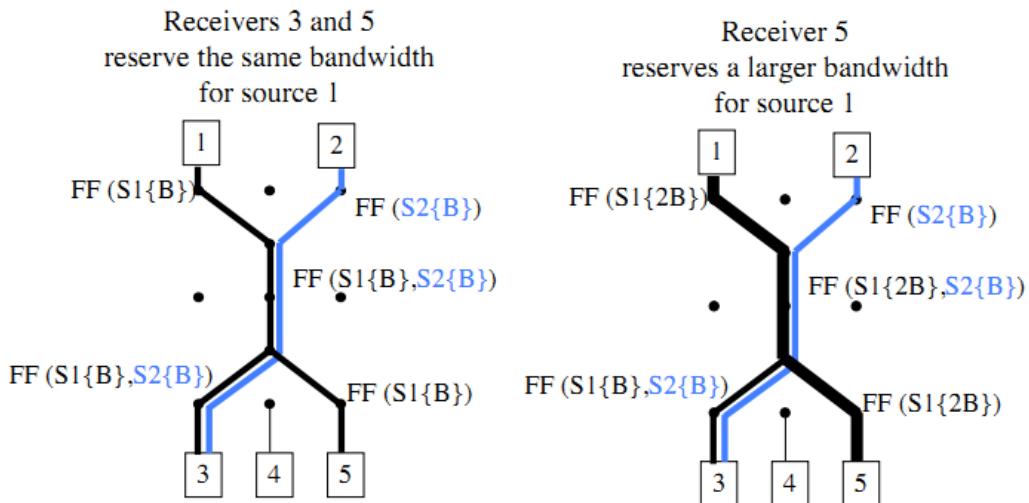


Réservation par source

Il s'agit d'une réservation dite FF (Fixed-Filter).



Une source peut avoir différentes réservations.

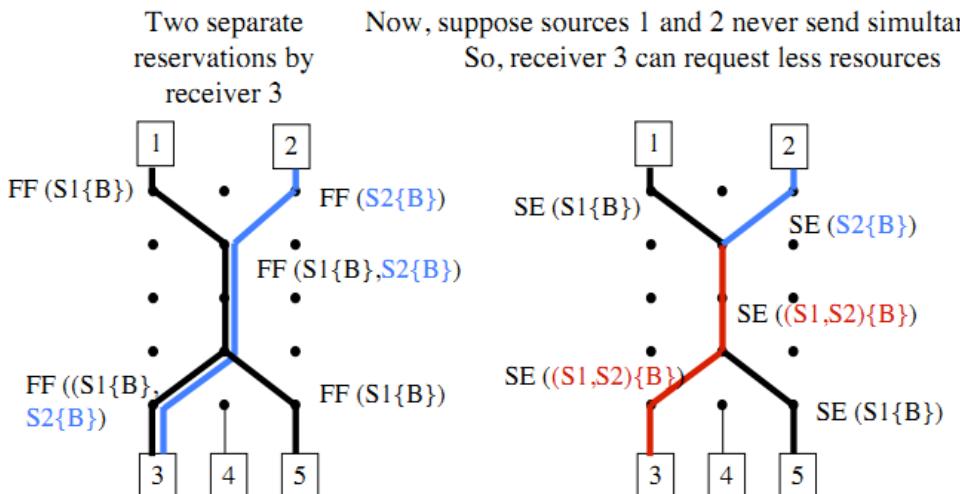


L'host 3 va demander une réservation de B bits/s. FF : réservation fixe pour une source. Lorsque l'hôte 5 voudra faire une réservation, seul un segment sera réservé, car une partie du chemin est déjà en place.

L'hôte 4 profite de la réservation jusqu'au dernier lien, et tant que la réservation B n'est pas dépassé.

Réservation partagée par des sources

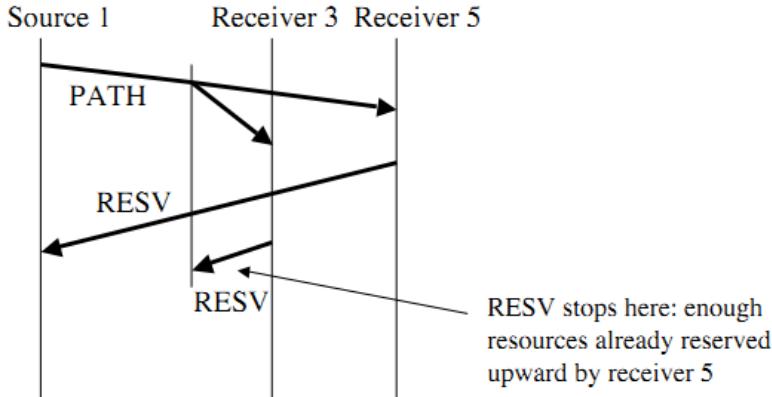
Ce schéma est appelé réservation SE (Shared-Explicit). Cela convient pour les applications où une seule source peut émettre à la fois (ex : conférence téléphonique).



$SE(S1\{B\})$ peut être fusionné avec $SE((S1, S2)\{B\})$, mais $FF(S1\{B\})$ ne peut pas, car...

Messages RSVP

La source envoie des messages PATH en multicast, sur le spanning tree du groupe, qui décrit l'enveloppe du trafic. Les récepteurs peuvent dès lors envoyer un message RESV qui effectue la réservation.



Ce sont donc les récepteurs qui s'occupent des réservations. C'est logique, car les récepteurs peuvent être très hétérogènes, et certains peuvent avoir les moyens de faire des réservations, d'autres pas.

Les messages PATH sont donc envoyés par la source, et décrivent les T-SPEC.

Les messages RESV contiennent un message qui fait la réservation (R-SPEC) ; ils contiennent la quantité de ressources qu'ils veulent réserver. Ces messages suivent donc la même route que le chemin suivi par les messages PATH.

Le chemin de retour n'est pas nécessairement le même que pour l'aller (car le spanning tree provient de sources différentes), or il faut que le RESV suive le même chemin (à l'envers) que le PATH. Un état est alors mis en place dans les routeurs pour garder une trace de l'interface d'entrée d'un message PATH. Ainsi, quand un RESV est reçu, on connaît l'interface de sortie.

Les messages RESV vont revenir aussi loin que c'est nécessaire vers la source ; ils ne sont forwardés vers la source que si la requête de réservation est plus grande que la réservation déjà mise en place pour le groupe multicast. On a ainsi une réservation Hop-by-hop.

Types de réservation

Le type de réservation est spécifié dans les messages RSVP. On en a trois :

- Fixed-filters (FF) : réservation pour recevoir depuis un seul émetteur, les autres sources ne l'utiliseront pas ;
- Shared-Explicit (SE) : réservation pour recevoir depuis un ensemble de sources ;
- Wildcard-Filter (WF) : réservation pour le groupe, c'est-à-dire pour recevoir depuis toutes les sources (c'est un SE avec toutes les sources).

Seuls les paquets qui satisfont les spécifications du filtre utilisent les ressources réservées. Les autres utiliseront ce qu'il reste de la bande passante, sinon ils rentreront en mode best effort.

Etat dans les routeurs

Les protocoles dit Hard state sont ceux où on met en place et où on supprime un état explicitement. A l'inverse, un protocole dit Soft State supprime l'état tout seul, après une période. Il doit être périodiquement rafraîchi pour rester en place.

Avec RSVP, l'état est soft ; il est périodiquement rafraîchi, des messages PATH et RESV doivent être envoyés fréquemment.

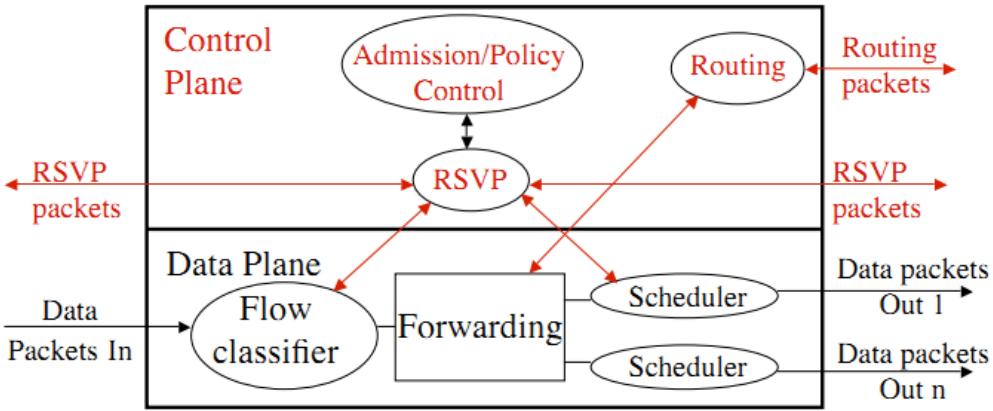
Lors d'une failure, un protocole de routage unicast sera lancé pour reconstruire les routes. Ensuite, l'arbre multicast sera reconstruit dessus, et enfin on rétablira les réservations. On peut oublier les réservations précédentes, vu que c'est un état de type soft. Si c'était en hard, on pourrait avoir des ressources qui resteraient réservées, mais jamais utilisées.

La période de rafraîchissement de la réservation ne peut pas être trop longue (car s'il y a une failure, on perdra la réservation pendant un moment) ni trop courte (sinon overhead).

Lors d'un changement de topologie (link failure, ou changement du poids d'un lien), la priorité est de rétablir le routing avant de s'occuper de la réservation. De ce fait, cette dernière peut se remettre en place après un certain temps. La factorisation des problèmes de routage et de réservation est ici pénalisante ; par exemple en ATM ou MPLS il y a une réservation de ressources en même temps que l'établissement des routes.

De plus, avec ATM ou MPLS, (sauf failure), lorsqu'un VC est établit, le chemin va rester stable. Avec IP, le chemin peut changer dynamiquement, ce qui fait que les ressources peuvent ne pas suivre. De ce fait, on peut arriver à du best effort si la dynamique est trop importante.

Architecture d'un noeud avec RSVP



Le contrôle Admission/Policy "décide" si on peut accepter une réservation ou non.

La classification de flux permet de distinguer les flux qui ont une réservation de ressources ou non. S'ils n'en ont pas, on passe en best effort, sinon on lit le tag du paquet (utilisé seulement localement, il sera supprimé à la sortie).

Le scheduler applique le service de scheduling/dropping pour les flux.

Description d'un flux

Il y a trois spécifications.

FilterSpec Elle permet d'identifier la(les) source(s).

- avec IPv4, un flux est un flux de couche 4; la source est identifiée par son adresse IP et le port ;
- avec IPv6, idem qu'IPv4, ou alors on utilise l'adresse IP de la source et le label du flux (header IPv6) ;
- avec MPLS, un flux est identifié par le label MPLS.

Avec IPv6, le label de flux permet de faire une meilleure discrimination qu'avec le port, car une même application pourrait envoyer plusieurs flux différents. La source a ainsi plus de flexibilité (rassembler plusieurs petits flux en un plus gros par exemple).

Session Elle identifie le(s) destinataire(s). Il s'agit de l'adresse IP (unicast ou multicast), de l'id du protocole et du numéro de port (optionnel)

FlowSpec Cela comprend les T-SPEC et les R-SPEC.

Les TSPEC sont un quintuplet (r, b, p, m, M) , où

- r le débit du token bucket (lié au débit moyen de la source) ($= \rho$)
- b la taille du token bucket (lié à la taille de burst maximale) ($= \sigma$)
- p le peak rate
- m la taille de paquet minimale
- M la taille de paquet maximale

Le couple (M, p) formera le token bucket pour le peak rate. Les paquets plus grands que M seront envoyés en best effort.

Le couple (b, r) formera le token bucket pour le débit moyen, avec une tolérance de burst b .

Les paquets plus petits que m seront considérés comme étant de taille m , afin de les pénaliser, car un grand overhead relatif est créé.

Les RSPEC se limitent à R , le débit réservé en bytes/sec. D'après la formule de Parekh-Gallager, plus R sera grand et plus le délai garantit sera bas.

Le problème est que le récepteur doit connaître K , P et $P(k)$ pour tous les k afin de calculer R en connaissant son délai end-to-end maximum, M et b .

La solution est que les messages PATH contiennent aussi les ADSPEC, un couple (C, D) mis à jour à chaque hop sur le chemin, et défini ainsi :

$$C = \sum_{\text{Hop visités}} M$$

$$D = \sum_{\text{Hop visités}} \frac{P}{P(k)} + \text{délai de propagation du lien}(k)$$

En connaissant C et D , qui s'accumulent le long du chemin, le récepteur peut calculer R pour obtenir le délai maximum. R peut être plus grand que r et même plus grand que p .

Si aucun délai n'est nécessaire, on prend $R = r$, qui permet de garantir le débit moyen.

Ce que ne fait pas RSVP

Spécifier comment les ressources doivent être réservées Il ne fournit qu'un moyen pour les communications.

Généralement, la réservation est faite en appliquant un poids approprié aux flux dans chaque scheduler WFQ sur le chemin.

Déterminer la route des paquets C'est le travail de protocoles de routage (PIM par exemple) ; la signalisation est découpée du routage.

L'exception est que RSVP est utilisé avec ERO (Explicit Route Object), pour mettre en place les LSPs MPLS point à point.

Interagir avec les paquets forwardés Il y a une séparation du contrôle (signaling) et des données (forwarding).

5.2.5 IETF Integrated Services

Deux types de services spécifiques sont décrits dans des RFCs : guaranteed service et controlled load service.

Guaranteed service

Il s'applique aux applications qui nécessitent des délais courts, aucunes pertes et des garanties de bande passante. Les routeurs allouent ainsi de la bande passante et de l'espace dans les buffers.

La source est décrite par le token bucket (ρ, σ) , le peak rate et le MTU.

On utilise la formule de Parekh pour calculer le délai maximal ; la bande passante réservée permet de parer au pire cas d'arrivée de trafic.

Ce service s'applique "pour tous les paquets qui transportent la voix", etc.

Controlled load service

Il s'agit d'une qualité de service qui approxime le QoS qu'un même flux aurait eu dans le cas d'un réseau non chargé. On ne détériore pas la qualité quand il y a des augmentations de charge (plus ou moins le même délai, taux de perte et bande passante).

Ce service est bon pour les applications qui peuvent tolérer une certaine quantité de délai (variations) et des pertes.

Avec $R = r$ dans les RSPEC, on y arriverait.