
# TBXCAST

## *un protocole de routage multicast explicite*

## Rapport de pré-étude

Cyril BOULEAU
Hamze FARROUKH
Loïc LE HENAFF
Mickaël LECUYER
Jozef LEGÉNY
Benoît LUCET
Emmanuel THIERRY

Encadrants : Miklós MOLNÁR, Bernard COUSIN

# Sommaire

# 1 Introduction

Le projet TBXcast a pour objectif de créer un protocole[1] de communication entre les machines.

En tirant bénéfice des recherches et du travail de l'équipe de projet de l'année dernière, nous avons pu établir une certaine stratégie afin de découvrir le sujet. Ce rapport est là pour détailler nos étapes de découverte et orienter une direction pour l'avenir.

Afin de comprendre le « pourquoi » de notre sujet, c'est-à-dire de comprendre l'intérêt de développer un nouveau protocole de communication, nous avons du apprendre les bases d'un domaine bien particulier de l'informatique : le réseau. La première partie de cette pré-étude concernera donc notre apprentissage sur les technologies d'aujourd'hui – les technologies liées au contexte IPv4[2] - et même à venir dans un futur proche – IP version 6.

La deuxième partie de notre pré-étude se concentrera sur les différents modèles de routage, c'est-à-dire de mise en relation distante de différentes machines entre elles. En effet, c'est grâce aux technologies présentées dans la première partie que nous allons pouvoir modéliser tel ou tel type de communication. C'est ici que notre protocole TBXcast va trouver son sens : s'illustrer sur un type de communication encore peu représenté, le multicast explicite arborescent[3].

Le groupe de projet de l'année dernière sur le même sujet s'est rapidement rendu compte que la programmation système, donc de bas niveau, allait être délicate et a donc pris la décision de baser TBXcast sur le code d'un protocole déjà existant : Xcast. Dans la continuité de ce choix, nous avons étudié le protocole Xcast et la troisième partie de cette étude de domaine y sera entièrement consacrée.

Ce rapport présente donc notre apprentissage progressif sur les bases du réseau, puis sur les méthodes de routage qui nous ont finalement permis d'aborder Xcast.

# 2 Réseau

## 2.1 Le modèle OSI

Le modèle OSI (Open Systems Interconnection) définit sept couches différentes, correspondant à des niveaux d'abstraction croissants. C'est un modèle stable et pérenne car les méthodes de communication qu'il définit sont universelles et reposent sur une base immuable.

En effet, dans tout réseau ou système de mise en relation, on peut imaginer la notion d'adjacence qui définirait une zone de proximité dans laquelle la communication serait facilitée. On peut ensuite imaginer qu'au-delà d'une certaine limite, il faille mettre ces petites zones en relation au travers d'un moyen de communication unique. Et que faire si on veut échanger avec une zone située à très grande distance, dois-je alors faire appel à des relais ?

C'est sur ces bases de « bon sens » qu'est construit le modèle OSI, et à travers ses sept couches il définit alors rigoureusement un modèle de communication employé par tous nos ordinateurs.

### 2.1.1 Couche Physique

La couche Physique définit la manière dont les données circulent dans le matériel utilisé. Elle correspond à des questions du type : « Quelle est la forme de la prise ? Celle du câble ? Est-ce un câble électrique ou bien optique ? Quelle est la technique de modulation employée pour faire circuler les bits ? ».

On s'intéresse donc ici à la manière physique, technique de faire circuler les données.

[1] Un protocole de communication est une spécification de plusieurs règles pour un type de communication particulier.
[2] IP signifie « Internet Protocol » et est de nos jours très largement utilisé dans sa version 4.
[3] Protocole dont on spécifie une liste explicite des destinataires, et dans lequel la route est représentée par un arbre

### 2.1.2 Liaison de données

Dans cette couche, on s'intéresse à la synchronisation (début et fin) des trames ainsi qu'à la détection des erreurs et la gestion des collisions. La deuxième couche peut maintenant se reposer sur une première couche parfaitement définie, et ne plus se soucier de comment les données circulent physiquement. On va donc s'intéresser aux chaînes de bits reçues : les trames. Les trames (Ethernet) sont des chaînes dont la taille est limitée à 1500 octets, pour des raisons historiques mais également pour des raisons de contrôle d'erreur. En effet, la capacité de détection d'erreurs diminue avec l'augmentation de la taille des trames car les cartes réseaux doivent stocker la trame entière dans leur mémoire interne.

Les deux premières couchent définissent le réseau local, un lieu de communication privilégié entre les machines. Dans un réseau local, une machine est à l'écoute de toutes les autres et la communication y est très rapide. De manière imagée, on pourrait associer le réseau local à une pièce dans laquelle plusieurs participants parlent entre eux. Si tout le monde peut communiquer avec tout le monde, c'est parce que les participants sont peu nombreux et proches les uns des autres.

### 2.1.3 Réseau

La couche Réseau s'occupe de l'adressage : c'est ce qui permet d'identifier une machine en particulier. Dans un deuxième temps, et une fois la machine identifiée, la couche Réseau s'occupe de trouver un chemin pour lui acheminer les données : c'est le routage. Dans le protocole IP[4], les machines sont identifiées par des adresses IP et reçoivent des paquets IP (d'une taille pouvant aller jusqu'à 64 ko) pouvant passer par plusieurs routeurs. Les routeurs sont des relais qui forment une chaîne devant acheminer les informations provenant d'une source vers la bonne machine.

Les données sont donc encapsulées dans des paquets IP qui vont circuler entre les machines. Pour avoir l'illusion que la taille maximale des données échangées n'est limitée que par celle du paquet IP, on a recours au principe de fragmentation. En effet, nous avons appris au 2.1.2 que les trames sont limitées à 1500 octets. La fragmentation va donc revenir à couper le paquet IP en plusieurs petits morceaux qui devront être reconstitués une fois les données parvenues au destinataire (processus transparent en informatique grâce à la proximité spatiale des données dans la mémoire vive de la machine). La fragmentation est donc le fait de découper un bloc de données issu d'un système de communication supérieur ou inférieur et devant s'astreindre à certaines contraintes.

La couche Réseau correspond donc aux moyens mis en œuvre pour acheminer des données depuis une source vers un destinataire en passant par des routeurs.

### 2.1.4 Transport

La quatrième couche du modèle OSI marque une limite entre les couches dites « hautes » et les couches « basses ». On ne s'intéresse plus aux machines ni aux paquets mais aux processus qui ont permis l'envoi de ces paquets. Une relation procédurale se met en place « de bout en bout » entre deux machines (client – serveur) et permet d'ouvrir un dialogue. Sans cela, comment savoir si un paquet a bien été reçu, et par le bon destinataire ?

Un autre problème spécifique à la troisième couche était la gestion de la taille des données. Dans notre quatrième couche, il n'existe aucune contrainte de taille. Pour imager cela, on pourrait penser à une caméra de surveillance qui doit transmettre en permanence des images. Dans la couche de Transport peu importe la manière dont les paquets sont crées à partir du flux vidéo, on s'occupe seulement de savoir quand et à qui on transmet ces données vidéos.

La couche de Transport s'affranchit des limites physiques et propose la mise en relation des processus client avec les processus serveur indépendamment de la taille des données à transmettre.

### 2.1.5 Les couches 5, 6 et 7

La couche de Session (couche 5) définit des temps de parole et des changements de rôle au niveau applicatif.

[4] Internet Protocol : voir section 2.2

La couche de Présentation (couche 6) permet de répondre à la question suivante : « Comment faire en sorte que deux machines se comprennent quand elles parlent de la même chose mais sous deux représentations différentes ? ». Dans cette couche, on parle donc de la sémantique de l'information.

La couche d'Application (couche 7) correspond aux parties des applications qui sont sollicitées dans le cadre de communications de diverses sortes : FTP, messagerie, SSH, HTML etc.

Ces trois dernières couches sont fusionnées en une seule dans le modèle TCP/IP de l'Internet.

### 2.1.6 Schéma récapitulatif

La figure 1 décrit le scénario dans lequel un émetteur veut transmettre une donnée à un destinataire.

C'est la couche la plus haute (couche 7 d'application) qui va exprimer le besoin d'émettre une donnée. Petit à petit et en descendant de couche en couche, la donnée va prendre des formes différentes pour être finalement représentée par une chaîne de bits qui va traverser le câble jusqu'au premier routeur.

On a vu que la couche réseau s'occupait du routage, et c'est pourquoi la donnée va remonter jusqu'au troisième niveau : pour être traitée par le routeur.

Ainsi de suite la donnée va parvenir au destinataire, dont la machine sera capable de la faire remonter depuis son état de plus bas niveau (chaine de bits) jusqu'à sa représentation dans l'application concernée.

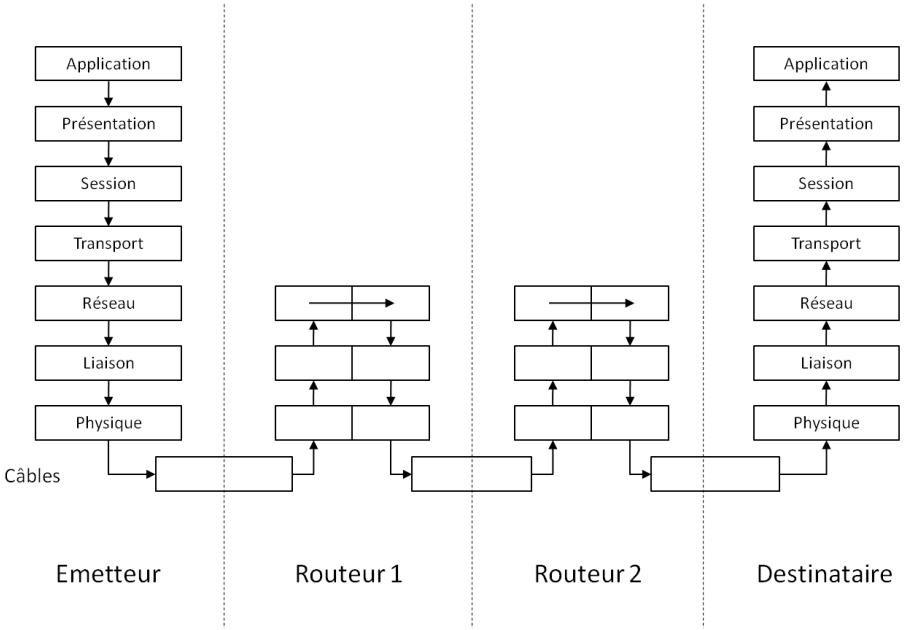



Figure 1 : modèle OSI

## 2.2 Protocoles de couches 3 et 4

Nous allons décrire les structures et le fonctionnement des protocoles utiles à notre étude. Nous allons commencer par étudier la version 4 d'IP qui marque le début de l'Internet grand public. Puis nous étudierons la version 6 d'IP qui palie à certains défauts d'IPv4 et qui par la suite pourrait devenir son successeur. Enfin nous nous intéresserons à deux protocoles de la couche 4.

### 2.2.1 Généralités de l'IP

Lorsque deux postes informatiques connectés à Internet désirent communiquer, plusieurs protocoles entrent en jeu. Dans un premier temps, nous allons nous focaliser sur IP (Internet Protocol), qui se situe au niveau 3 sur le modèle OSI. Lorsqu'un poste envoie des données à un autre poste, celles-ci sont découpées en paquet. Chaque paquet est envoyé à travers le réseau et réceptionné par le destinataire. Ces paquets sont acheminés grâce au protocole IP.

Le protocole IP fonctionne en mode non connecté, c'est à dire que le chemin emprunté par les paquets sur le réseau n'est pas établi à l'avance. Il est "non fiable" dans le sens où il ne permet pas à l'émetteur de savoir si un paquet est arrivé à destination ou s'il est corrompu. Ces concessions au niveau de la fiabilité permettent une plus grande rapidité des échanges et une simplification de la complexité du traitement dans les routeurs.

## 2.2.2 IPv4

Internet Protocol version 4 ou IPv4 est la première version d'IP qui fut utilisée à grande échelle. Aujourd'hui encore, ce protocole est massivement utilisé pour établir des communications sur un réseau. Une adresse IPv4 est attribuée à chaque interface d'une machine lorsqu'il accède à Internet et permet alors de l'identifier. Cette adresse est constituée de 4 octets en représentation décimale pointée et est unique sur le réseau.

Depuis quelques années, la limitation d'IPv4 préoccupe. En effet, le développement rapide d'Internet a conduit à une saturation du nombre d'adresses disponibles, d'autant plus que toutes les adresses IPv4 ne sont pas utilisables (certaines plages entières sont réservées). Comme nous le verrons plus loin, la version 6 d'IP permet de s'affranchir de cette limitation. Pour accompagner la transition entre ces deux protocoles, des technologies ont dû être mise en place (DHCP, NAT, CIDR). Ces dernières permettent encore aujourd'hui de rallonger la durée de vie d'IPv4.

### 2.2.2.1 Structure d'un paquet IPv4

Un paquet IPv4 est divisé en plusieurs champs. Brièvement, on retrouve la version d'IP utilisée (ici la version 4) et des informations sur la longueur de l'en-tête et du paquet. Les champs identification, flags et fragment offset sont utilisés pour la fragmentation des paquets.

| **Bits** | **0-3** | **4-7** | **8-15** | **16-18** | **19-31** |
|---|---|---|---|---|---|
| **0** | Version | Header Length | Type of service (DiffServ, | Total length | |
| **32** | Identification | | | Flags | Fragment offset |
| **64** | Time to Live | | Protocol | Header Checksum | |
| **96** | Source Address | | | | |
| **128** | Destination Address | | | | |
| **160 ...** | Options | | | | |
| **160 ou 192+** | Data | | | | |

Tableau 1: Structure d'un paquet IPv4

Le champ Time to Live (TTL) est un compteur. Il est initialisé par l'émetteur et il est décrémenté à chaque saut de routeur. Lorsque ce compteur est égal à 0, le paquet est automatiquement détruit et un message ICMP est envoyé à l'émetteur pour le prévenir.
Le champ Protocol permet de définir quel protocole est utilisé pour décoder le champ de données du paquet IP (TCP, UDP, ICMP).
Le Header Checksum permet de vérifier l'intégrité de l'en-tête du paquet. On retrouve ensuite l'adresse de l'émetteur et du destinataire. Enfin, la fin du paquet est destinée à accueillir certaines options facultatives et les données.

### 2.2.2.2 La fragmentation des paquets IP

Dans un réseau informatique, une trame est un bloc d'information véhiculé au travers d'un support physique. Elle se situe sur le niveau 2 du modèle OSI. La taille des paquets pouvant être transmis dans une trame est donc limitée. Cette taille limite s'appelle Maximum Transmission Unit (MTU) et est de l'ordre du kilo-octet. Typiquement, la taille maximale d'un paquet IP est de 64 ko. Si la taille d'un paquet IP est supérieure au MTU, le paquet sera fragmenté en paquets plus petits pour pouvoir être transmis dans une trame. Arrivés à destination, les paquets fragmentés seront réunifiés grâce aux champs Identification, Flags et Fragment Offset du paquet IP.

Il est possible d'interdire la fragmentation d'un paquet, mais si la taille de ce dernier est supérieure au MTU, il sera automatiquement détruit.

#### 2.2.2.3 Les réseaux et sous-réseaux

Une adresse IP comporte deux parties: une partie permettant d'identifier le réseau IP et une partie permettant d'identifier un poste sur ce sous réseau IP. En IPv4, il existe cinq classes d'adresse IP différentes, nommées de A à E. Dans chaque classe, la partie identifiant le réseau change. Par exemple, dans la classe A, seul le premier octet de l'adresse permet d'identifier le réseau IP. Les trois octets suivants permettent d'identifier les postes de ce réseau IP. Les réseaux IP de classe B sont identifiés par deux octets et ceux de classe C par trois. La classe D est réservée au multicast tandis que le classe E est réservée pour une utilisation ultérieure. Ainsi, un réseau IP de classe A possède plus d'adresses à attribuer qu'un réseau IP de classe C.

Lorsqu'un paquet arrive sur un routeur, ce dernier lit l'adresse de destination, consulte la table de routage du routeur, en déduit le « next hop » et retransmet le paquet. L'extraction de l'adresse du réseau et de l'adresse du poste est obtenue en appliquant un masque de sous réseau sur l'adresse IP.

- Si le routeur possède le destinataire du paquet dans sa table de routage (c'est à dire que le destinataire est présent sur ce sous réseau IP), alors le routeur envoie directement le paquet à la machine correspondante,
- Sinon, le routeur détermine le routeur auquel il doit transmettre le paquet d'après sa table de routage. Les routeurs suivants procéderont alors au même traitement jusqu'à l'acheminement du paquet IP.

#### 2.2.2.4 Le protocole ICMP

Internet Control Message Protocol (ICMP) est utilisé pour véhiculer des messages d'erreur ou de contrôle. Il se situe tout comme le protocole IP sur la couche 3 (Réseau) du modèle OSI. Rappelons que le protocole IP ne permet pas d'envoyer des messages d'erreur. Dans certains cas le protocole ICMP permet de savoir s'il y a eu un incident sur le réseau. Un paquet ICMP peut être envoyé pour plusieurs raisons, par exemple lorsque le destinataire d'un paquet IP n'est pas accessible ou lorsque le Time to Live tombe à 0.

L'exemple le plus connu auprès des utilisateurs qui fait intervenir un paquet ICMP est la commande *ping*. Celle-ci permet d'envoyer un message à une adresse pour vérifier si elle est accessible. Si elle ne l'est pas, un message ICMP est renvoyé.

#### 2.2.2.5 En attendant IPv6 …

Pour palier au problème de pénurie d'adresses IPv4 trois solutions ont été mises en place.

##### *CIDR*

Le *Classless Inter-Domain Routing* (CIDR) fut mis en place afin de prolonger la durée de vie de la version 4 d'IP. Il a été mis au point dans le but de diminuer la taille des tables de routage des routeurs en agrégeant plusieurs entrées en une seule. Le CIDR a aboli l'utilisation des classes d'IP afin de permettre des attributions d'adresse plus fine (en frontière de bits et non plus en frontière d'octets).

##### *DHCP*

DHCP signifie *Dynamic Host Configuration Protocol*. Il s'agit d'un protocole qui permet à un ordinateur qui se connecte sur un réseau d'obtenir automatiquement et dynamiquement sa configuration (principalement, son adresse IP, l'adresse IP du routeur par défaut). Le but principal étant la simplification de l'administration d'un réseau. Le protocole DHCP sert principalement à distribuer des adresses IP sur un réseau.

##### *NAT*

Le mécanisme de translation d'adresses NAT (*Network Address Translation*) a été mis au point afin de répondre à la pénurie d'adresses IP, due aux limitations du protocole IPv4. NAT permet en effet d'identifier tout un réseau local d'ordinateur sur Internet, via une seule adresse IP.

Le principe du NAT consiste donc à utiliser une passerelle de connexion à Internet, possédant au moins une interface réseau connectée sur le réseau interne et au moins une interface externe connectée à Internet (possédant donc une adresse IP routable), pour connecter l'ensemble des machines du réseau. Chaque machine du réseau devant accéder à Internet est configurée pour utiliser la passerelle NAT (en précisant l'adresse IP de la passerelle dans le champ « Gateway » de ses paramètres TCP/IP). Lorsqu'une machine du réseau effectue une requête vers Internet, la passerelle effectue la requête à sa place, reçoit la réponse, puis la transmet à la machine ayant fait la demande.

## 2.2.3 IPv6

Le développement des télécommunications, des médias et de l'Internet a eu pour effet le dépassement des capacités d'IPv4. Le problème de pénurie d'adresses a été repoussé à plusieurs reprises mais au final il est évident que ce protocole doit être remplacé par une solution différente. Parmi les solutions proposées, la solution ayant la plus grande probabilité d'être finalement adoptée, notamment grâce au fait qu'elle soit déjà implémentée sur la plupart du matériel, est le protocole IPv6.

### 2.2.3.1 Différences par rapport à IPv4

#### *Plage d'adressage bien plus importante*

L'ancien protocole IPv4 codait les adresses sur 32 bits ce qui permettait de disposer de $2^{32}$ adresses uniques. Ce nombre est, de nos jours, très insuffisant et n'est viable que grâce au système NAT. Le protocole IPv6 prévoit une plage d'adresses bien plus large. Les adresses sont codées sur 128 bits ce qui devrait être suffisant pour les générations à venir.

#### *Simplification du routage*

Ce nombre n'est cependant pas choisi juste pour des raisons de capacité. Les adresses sont hiérarchisées avec des plages de bits de longueur fixe pour les différents niveaux hiérarchiques du réseau. Ceci est un grand changement par rapport au système CIDR de IPv4 où la longueur de l'identifiant du (sous)réseau IP était variable par rapport au masque.

#### *Diminution de charge sur les routeurs*

Le nombre de paquets traités par le routeur par seconde est très important, il est alors nécessaire de raccourcir le traitement de chaque paquet. Dans ce but, l'entête IPv6 a été simplifié. Même si l'adresse elle-même est quatre fois plus grande, l'entête n'est seulement que deux fois plus longue.

#### *Extensibilité*

IPv6 propose des entêtes optionnels destinés à l'extension de l'entête fixe IPv6. Ceux-ci se trouvent derrière l'entête IPv6 fixe et sont de longueur variable, limités seulement par la longueur du paquet lui-même. Ceci permet entre autres de garder un entête IPv6 obligatoire et de longueur fixe.

#### *Sécurité*

Le protocole IPsec est à part entière dans la suite des protocoles IPv6. Ainsi le support du service de confidentialité par chiffrement est obligatoire dans le monde IPv6.

## 2.2.3.2 Structure de l'entête IPv6

L'entête IPv6 fixe a une longueur de 256 bits. On peut remarquer que même si les adresses IPv6 sont quatre fois plus longues que les adresses IPv4, l'entête n'est que deux fois plus grande.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | **0** | **4** | | | **12** | **16** | | **24** | | **32** |
| | Version | Traffic Class | | | Flow Label | | | | | |
| **64** | Payload Length | | | | | Next Header | | Hop Limit | | |
| **128** | Source Address | | | | | | | | | |
| **192** | | | | | | | | | | |
| **256** | Destination Address | | | | | | | | | |

Tableau 2: Structure de l'entête IPv6

- Version – Contient la version du protocole, ici 6
- Traffic Class – Spécifie la priorité paquet
- Flow Label – Utilisé pour le contrôle de qualité (QoS[5]) – inutilisé à ce jour
- Payload Length – Longueur des données dans le paquet
- Next Header – Spécifie le protocole englobé
- Hop Limit – Limite la durée de vie d'un paquet (anciennement TTL)
- Source Address – l'adresse de l'expéditeur
- Destination Address – l'adresse du destinataire

## 2.2.3.3 Enchaînement des entêtes optionnels

IPv6 évite d'utiliser une longueur fixe pour les éventuelles options de l'entête comme IPv4. En revanche il spécifie un moyen d'utiliser les extensions IP. Les extensions sont constitués des entêtes enchaînés les uns à la suite des autres. Chacun d'entre eux spécifie le type de l'entête qui le suit. L'ordre est important car un routeur peut décider de ne pas poursuivre la lecture des extensions si cela n'est pas nécessaire et qu'il a déjà récupéré assez d'informations pour traiter correctement le paquet. La totalité des entêtes optionnels d'un paquet est lu par le destinataire.

### *Partie fixe de l'entête de l'extension*

| **Next Header** | **HdrExtLen** | |
|---|---|---|

Tableau 3: Partie fixe de l'entête optionnel

Les seuls paramètres imposés sont l'identifiant de l'entête suivant et la longueur même de l'entête de l'extension.

## 2.2.3.4 Différents types d'adresses IPv6

La plage d'adressage plus importante permet à l'IPv6 de restructurer la représentation des adresses et ainsi faciliter le routage. Les parties définissant le sous réseau et l'adresse de l'interface de la machine sont bien définies et fixes, on n'a donc pas besoin d'utiliser des masques de sous réseau.

### *Adresse unicast globale*

| **001** | **TLA ID (13 bits)** | **Res (8 bits)** | **NLA ID (24 bits)** | **SLA ID (16 bits)** | **Interface ID (64 bits)** |
|---|---|---|---|---|---|
| Public Topology | | | | Site Topology | Interface of a node on a specific subnet |

Tableau 4: Adresse unicast globale

[5] Quality of Service : Qualité de la connexion entre deux machines, prenant en compte de nombreuses notions (débit, latence, coût financier,…)

- **001**– Identifie l'adresse en tant qu'adresse IPv6 unicast globale
- **TLA ID** (Top Level Aggregation) – Le plus haut niveau de routage. Administré par IANA
- **Res**- Réserve au cas où il s'avérait nécessaire d'augmenter la taille de TLA ID ou NLA ID
- **NLA ID** (Next Level Aggregation) – Spécifie un site.
- **SLA ID** – (Site Level Aggregation) – Spécifie un sous-réseau au sein d'un site.
- **Interface ID –** Identifiant d'un poste.

### *Adresse unicast locale à un site*

| **1111 1110 11**<br>**(10 bits)** | **000 … 000**<br>**(38 bits)** | **Subnet ID**<br>**(16 bits)** | **Interface ID**<br>**(64 bits)** |
|---|---|---|---|

Tableau 5: Adresse unicast locale à un site

- Préfixe : FEC0::/48
- La structure après les 48 premiers bits est identique à l'adresse globale.
- Utilisé dans les réseaux locaux.

### *Adresse unicast locale à un lien*

| **1111 1111 10**<br>**(10 bits)** | **000 … 000**<br>**(54 bits)** | **Interface ID**<br>**(64 bits)** |
|---|---|---|

Tableau 6: Adresse unicast locale à un lien

- Les ordinateurs dans un sous réseau peuvent se retrouver directement grâce à ces adresses

### *Adresse multicast*

| **1111 1111**<br>**(8 bits)** | **Flags**<br>**(4 bits)** | **Scope**<br>**(4 bits)** | **Group ID**<br>**(112 bits)** |
|---|---|---|---|

Tableau 7: Adresse multicast

- **1111 1111** - Identifie l'adresse en tant qu'une adresse multicast
- **Flags** – En ce moment on utilise seulement le flag T. Il identifie une adresse en tant que transitive s'il est mis à un. Sinon l'adresse est permanente.
- **Scope** – Définie la portée de l'adresse, telle que Interface-local, Site-local, Link-local ou Global.
- **Group ID** – L'identificateur du groupe.

### *Adresses spéciales*

Dans IPv6, on distingue quelques adresses réservées avec une signification bien précise.

- Adresse non spécifiée - **::**
- Loopback - **::1**
    - Boucle sur l'interface qui envoie le paquet

*Remarque : Dans la notation des adresses IPv6 on peut remplacer un nombre quelconque de 0 consécutifs par un double « deux-points ». Ce remplacement ne peut être effectué qu'une seule fois par adresse.*

## 2.2.3.5 ICMPv6

Les messages de contrôle font intégralement partie du protocole IPv6 et chaque implémentation et routeur IPv6 doit les supporter. Les messages ICMPv6 (*Internet Control Message Protocol Version 6)* sont transportés par les paquets IPv6, leur signature Next Header étant 58.

On distingue deux types de messages : messages de contrôle et messages d'erreur.

| Type | Meaning |
|---|---|
| ***ICMPv6 Error Messages*** | |
| **1** | Destination Unreachable |
| **2** | Packet Too Big |
| **3** | Time Exceeded |
| **4** | Parameter Problem |
| **100** | Private experimentation |
| **101** | Private experimentation |
| **127** | Reserved for expansion of ICMPv6 error messages |
| ***ICMPv6 Informational Messages*** | |
| **128** | Echo Request |
| **129** | Echo Reply |
| **200** | Private experimentation |
| **201** | Private experimentation |
| **255** | Reserved for expansion of ICMPv6 informational |

Tableau 8: types des messages ICMPv6

### *Ping6*

L'application IPv6 la plus simple est Ping6. En utilisant les messages ICMPv6 il permet de vérifier la présence d'une interface distante et le temps de réponse. L'émetteur émet un paquet Echo Request et son destinataire doit répondre avec un paquet Echo Reply.

## 2.2.4 Protocoles de transport

Il existe deux grands protocoles de transport sur Internet, l'un qui avantage la vitesse de transmission (UDP), l'autre qui avantage la fiabilité (TCP).

### 2.2.4.1 Protocole UDP

Le protocole UDP (*User Datagram Protocol*) est un protocole en mode non connecté de la couche Transport du modèle TCP/IP. Ce protocole est très simple étant donné qu'il ne fournit pas de contrôle d'erreur. Il est par exemple utilisé par les applications de type téléphonique où la minimisation du délai prime sur la fiabilité du transport des paquets. Un exemple connu d'utilisation de ce protocole est le logiciel de téléphonie Skype.

### 2.2.4.2 Protocole TCP

TCP (*Transmission Control Protocol*) est un des principaux protocoles de la couche Transport du modèle TCP/IP. Il permet de transmettre un flux de données (flux d'octets) entre deux processus situés chacun sur deux postes distants. Les caractéristiques principales du protocole TCP sont les suivantes :

- Remettre en ordre les datagrammes en provenance du protocole IP
- Contrôler le débit du flot de données afin d'éviter une saturation du réseau
- Formater les données en segments de longueur variable afin de les "remettre" au protocole IP
- Multiplexer les données, c'est-à-dire faire circuler simultanément des informations provenant de sources distinctes (applications par exemple) sur une même ligne
- Initialisation et fin d'une communication de manière « courtoise »

# 2.3 Routage

Sur un même sous-réseau, les machines sont directement interconnectées. Lorsqu'il faut envoyer un paquet à une machine, le protocole ARP pour IPv4 ou ICMPv6 pour IPv6 permet de traduire directement l'adresse IP en

adresse MAC (identification sur un réseau local). Les trames Ethernet sont ensuite envoyées vers toutes les machines, lesquelles acceptent ou rejettent le paquet.

Sur un réseau mondial, il paraît impossible que les machines envoient leurs paquets à toutes les autres, et plus encore, de relier toutes les machines directement. On doit alors placer des intermédiaires pour réduire le nombre de liens que chaque machine établit, tout en gardant la possibilité pour chaque machine sur le réseau de communiquer avec n'importe quelle autre : c'est le principe du routage.

## 2.3.1 Unicast

On organise le réseau en réseau de réseaux. Par ce mécanisme on permet de constituer une route entre deux machines, c'est à dire un chemin d'accès parmi tous les routeurs du réseau. La problématique est de découvrir ce chemin.

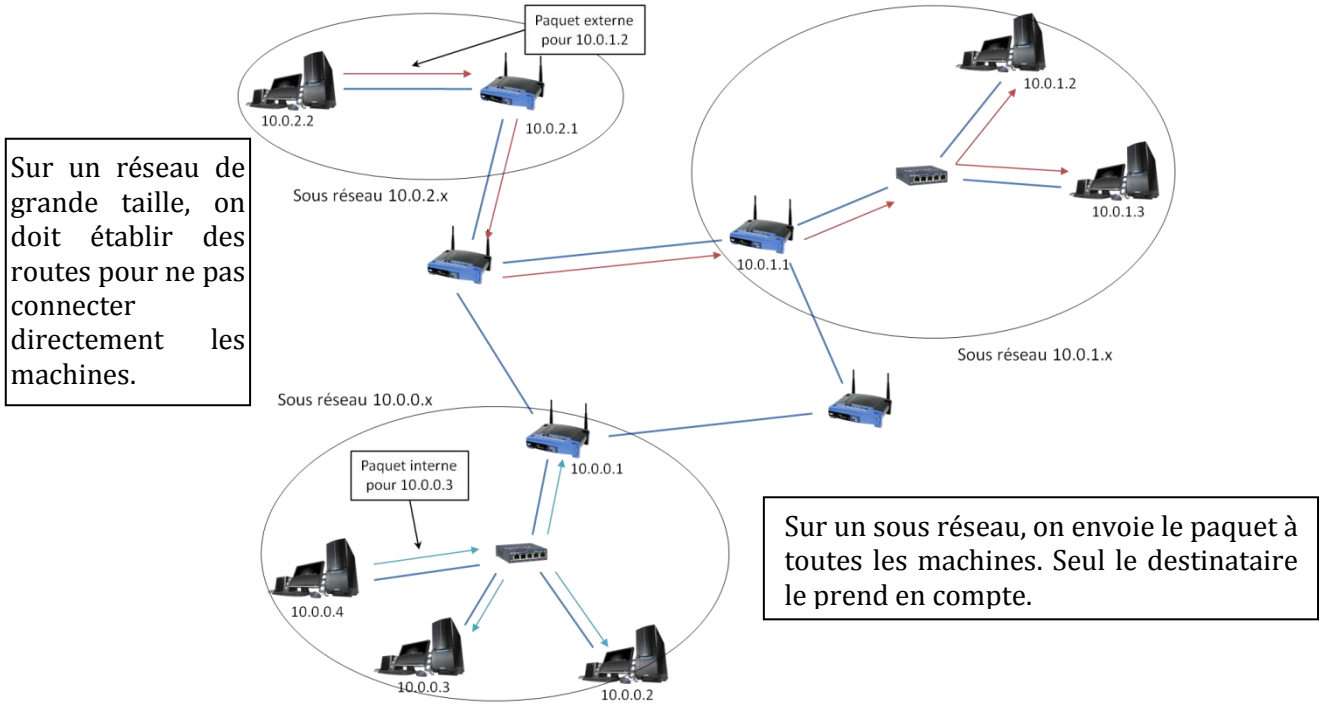



Figure 2: Exemple d'envoi d'un paquet unicast

### 2.3.1.1 Protocoles de routage

Il existe différents protocoles et algorithmes de routage pour la construction des tables de routage :

- RIP (Routing Information Protocol) : il se base sur un algorithme à vecteurs de distances, c'est à dire que chaque routeur est informé du nombre de routeurs entre lui-même et les autres. Le chemin privilégié sera alors celui qui passera par le moins de routeurs possible. Le RIP est adapté aux réseaux de petite taille.
- OSPF (Open Shortest Path First) : basé sur l'algorithme de Dijkstra, prend en compte la QoS de chaque route, dont entre autres le débit, la performance des routeurs etc. Par ailleurs, ses messages plus puissants permettent de réduire la charge du réseau et de repousser les limites du RIP. L'OSPF est donc adapté à des réseaux de taille moyenne.
- BGP (Border Gateway Protocol): Ce protocole est basé sur des politiques et des mécanismes de contrat. Chaque opérateur désigne les autres AS[6] avec lesquels il préférera établir des communications. Le BGP se base sur ces informations pour établir les relations entre les routeurs. Sur un Autonomous System, le choix du protocole de routage est très libre, dans la mesure où la politique de routage est commandée par la même entité. Pour relier les AS, le seul protocole disponible est BGP, qui est basé sur des

[6] Autonomous System : Système autonome géré par une seule entité administrative (fournisseur d'accès à internet, grosse entreprise ou université). Le réseau internet est l'agrégation de tous les AS.

politiques et des contrats inter-opérateurs. On distingue eBGP, destiné au routage inter domaine (sur internet) et iBGP qui, à l'instar de RIP ou OSPF, peut être utilisé pour le routage intra domaine (à l'intérieur d'un AS).

La plupart des paquets de ces protocoles se transmettent de proche en proche. Les routeurs transmettent des informations à leurs voisins, notamment l'existence d'un lien entre deux routeurs pour OSPF, et ces informations sont retransmises par le routeur qui reçoit le paquet. C'est une manière inévitable de procéder du fait qu'avant l'établissement des routes, le routeur ne connaît que ses voisins immédiats.

Chacun de ces protocoles permet de déduire, de manière plus ou moins fiable, la table de routage. Cette table permet alors de décider vers quelle machine rediriger un paquet en fonction de l'IP.

#### 2.3.1.2 Transmission des paquets

Un paquet unicast (à destination d'une seule machine) peut facilement trouver sa route. L'adresse de destination du paquet est constituée de telle manière que les premiers bits correspondront à un préfixe indiquant l'identification du sous réseau IP du destinataire. Ainsi, lorsqu'un routeur gère un sous-réseau, on lui assigne une plage d'adresses. Il est alors très facile de trouver la route vers le destinataire puisqu'il suffit de lire le préfixe d'une adresse pour identifier le prochain routeur.

### 2.3.2 Routage multicast

Le multicast est une réponse au problème de l'envoi de données depuis un expéditeur vers plusieurs destinataires. La minimisation de la bande passante et de la duplication des données étant essentielle, le multicast propose à la source de n'envoyer qu'une seule copie des données, qui sera par la suite acheminée à tous les membres du groupe multicast correspondant. La duplication se fait par les routeurs les mieux placés.

Pour des paquets multicast (à destination de plusieurs machines), il est impossible de se baser directement sur les principes sur lesquels unicast est fondé, dans la mesure où une même adresse pourra avoir des destinataires très éloignés.

#### 2.3.2.1 Notion de groupe

Un groupe multicast correspond à un ensemble de machines (donc d'utilisateurs) qui se sont volontairement abonnés à ce groupe. Chaque groupe multicast est identifié par une adresse multicast qui figure dans l'intervalle 224.0.0.0 – 239.255.255.255 (dans le cas d'IPv4), où certaines adresses sont réservées à des groupes multicast particuliers. Il est intéressant de noter que l'adresse réservée 224.0.0.1 correspond au « all-hosts group », ou encore la totalité des hôtes multicast du réseau local considéré.

Un groupe multicast peut avoir des membres répartis de manière clairsemée (c'est souvent le cas) ou dense, c'est pour cela que différents protocoles doivent être mis en place afin d'assurer au mieux l'envoi des données et leur duplication lorsque cela est nécessaire.

#### 2.3.2.2 Le protocole IGMP

C'est le protocole IGMP, ou encore Internet Group Management Protocol, qui va se charger de la gestion du groupe multicast de manière locale. Le routeur responsable de la jonction entre un réseau local donné (et ses machines) et le reste du réseau Internet doit supporter et implémenter le protocole IGMP.

De cette manière, les membres du réseau local peuvent émettre des demandes d'abonnement au routeur afin de rejoindre un groupe multicast désigné par une adresse. Cette opération de demande depuis l'hôte vers le routeur IGMP local est appelée « IGMP report ».

De son côté, le routeur émet régulièrement des messages de contrôle appelés « IGMP query » et qui lui permettent de vérifier à intervalle de temps régulier l'état et l'activité de chaque groupe multicast. Lorsqu'un hôte ne répond pas à l'IGMP query par un IMGP report, alors le routeur local ne le considère plus comme membre du groupe. Un hôte a également la possibilité de se retirer explicitement du groupe grâce au message « Leave ».

#### 2.3.2.3 Le protocole PIM

### *PIM sparse mode*

Le protocole PIM, ou « *Protocol Independant Multicast* » en mode clairsemé, travaille dans la continuité d'IGMP dans le sens où il va propager la demande de groupe à travers le réseau jusqu'à un routeur spécial désigné : le « Rendez-vous Point » ou RP. Comme le RP associé à un groupe multicast est connu de la source, il est facile de retrouver le chemin inverse « source -> RP -> destinataire » car les paquets unicast inverses qui ont servi à propager la demande de groupe vont également donner un ordre de mise à jour de la table de routage multicast des routeurs qu'ils ont traversé.

### *PIM dense mode*

En mode dense, le protocole PIM diffuse régulièrement par inondation les informations permettant le routage multicast. En inondant tous les hôtes du réseau, il ne garde que ceux qui ne répondent explicitement en transmettant en amont un message d'élagage (car ils sont membres du groupe multicast) et ainsi les routeurs peuvent mettre à jour leurs tables de routage multicast.

#### 2.3.2.4 Bilan

De part sa nature, le multicast gère bien les groupes peu éparses et contenant beaucoup d'utilisateurs. Mais lorsqu'il s'agit de transporter des données vers des groupes très clairsemés, il rencontre des difficultés : en effet, les tables de routage deviennent trop volumineuses. De plus, l'allocation unique d'une adresse à chaque groupe multicast est contraignante.

### *Multicast explicite*

Le principal défaut du routage multicast est la taille importante que prend la table de routage dans chaque routeur (une entrée par groupe multicast). Dans le chapitre suivant nous allons voir une solution qui permet de palier à ce problème : il s'agit d'indiquer explicitement les adresses IP des destinataires du message dans l'entête du paquet. Ainsi il n'y a pas besoin de stocker les correspondances adresse multicast – *next hop* dans les tables de routage.

Il existe deux implémentations différentes de ce système de routage :

- soit on place les entêtes IP des destinataires dans l'entête du paquet : c'est le routage multicast explicite plat
- soit on y place un arbre de routage : c'est le routage multicast explicite arborescent.

# 3 Exemple de protocole multicast explicite plat : Xcast

## 3.1 Introduction à Xcast

Le protocole Xcast est, comme le multicast, utilisé pour envoyer des messages à un groupe de machines. Un message Xcast est envoyé par une source qui connait l'ensemble de ses destinataires. Dans le protocole Xcast, la gestion de groupe n'est pas effectuée par les routeurs et un protocole spécial comme IGMP, mais c'est la source du message qui s'en occupe. Une machine qui veut adhérer à un groupe a juste à envoyer un message à la source qui l'ajoutera alors au groupe.

Pour émettre un message à l'ensemble des membres du groupe, la source va envoyer un paquet Xcast dont l'entête contient la liste des adresses IP des destinataires (contrairement aux paquets multicast dont l'entête ne contenait qu'une adresse IP multicast). Ensuite chaque routeur va faire le tri dans toutes ses adresses IP : le routeur va regarder, pour chaque adresse IP, quel est le prochain routeur dans sa table de routage, comme pour un paquet unicast. Le routeur va ensuite envoyer à chacun des routeurs trouvés un nouveau paquet Xcast dont l'entête va contenir la liste des adresses IP des destinataires qui doivent passer par ce prochain routeur. Si le destinataire est directement connecté au routeur, alors il va envoyer un simple paquet unicast à ce destinataire.

Sur l'exemple suivant, la source est désignée par s et les destinataires sont d1, d2, …, d6.

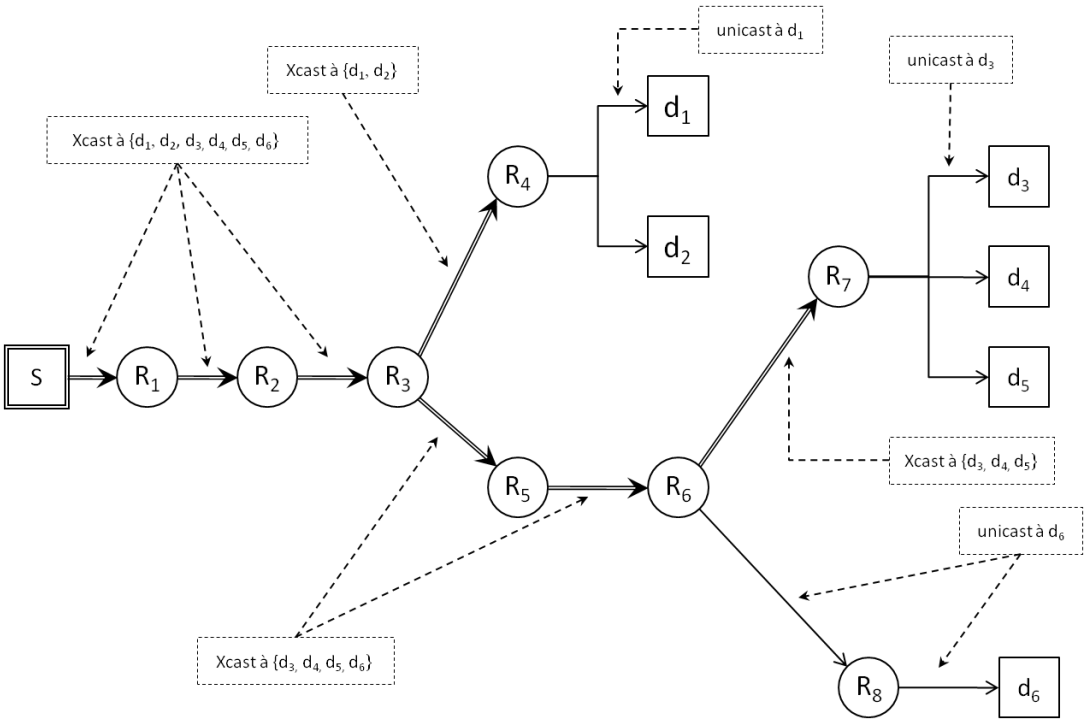



Figure 3: Exemple de l'acheminement d'un paquet Xcast

*S* veut envoyer un message aux six destinataires. Pour chaque destinataire, *S* va regarder par quel routeur doit passer le message pour atteindre ce destinataire. Ici tous les destinataires doivent passer par $R_1$. *S* va donc envoyer un paquet Xcast à $R_1$, contenant la liste $\{d_1, …, d_6\}$ des destinataires.

$R_1$ reçoit ensuite ce paquet et analyse la liste des destinataires. Pour chaque destinataire, $R_1$ regarde dans sa table de routage unicast quel est le routeur suivant. Il remarque que c'est $R_2$ pour tous les destinataires, il va donc envoyer un paquet Xcast à $R_2$ contenant la liste des destinataires $\{d_1, …, d_6\}$.

Le routeur $R_2$ fait la même opération que $R_1$ et transmet le paquet à $R_3$. $R_3$ analyse de même la liste des destinataires et remarque que le routeur suivant est $R_4$ pour les destinataires $d_1$ et $d_2$, et $R_5$ pour les destinataires $d_3$ à $d_6$. $R_3$ va donc envoyer un paquet Xcast à $R_4$ contenant la liste des destinataires $\{d_1, d_2\}$ et un autre paquet Xcast ayant le même contenu mais dont la liste des destinataires est $\{d_3, d_4, d_5, d_6\}$. $R_4$ reçoit son paquet Xcast et analyse la liste des destinataires de ce paquet : il remarque que $d_1$ et $d_2$ sont directement connectés à lui, il n'a donc qu'à envoyer un paquet unicast à $d_1$ et un autre à $d_2$ contenant l'information du paquet Xcast envoyé par la source. $d_1$ et $d_2$ peuvent alors enfin recevoir l'information envoyée par la source.

On applique le même genre de traitement pour les autres routeurs. On remarque le traitement spécial effectué en $R_6$ : tout comme $R_3$, $R_6$ va envoyer deux paquets Xcast, un à $R_7$ et un à $R_8$. Le paquet envoyé à $R_8$ ne contient qu'un seul destinataire, on fait donc une simplification en envoyant un simple paquet unicast à destination de $d_6$. Cette simplification est appelée X2U pour « Xcast to unicast ».

### 3.1.1 Les points forts de Xcast

Par rapport au multicast traditionnel, Xcast présente de nombreux avantages.

#### 3.1.1.1 Tables de routage

A ce niveau, Xcast est plus avantageux que le multicast traditionnel car il permet d'alléger grandement les tables de routage. En effet ces tables de routage doivent posséder une entrée pour chaque groupe multicast alors qu'il n'y a pas besoin de cela avec le multicast explicite.

#### 3.1.1.2 Diminution du trafic

En outre on allège le réseau car on n'a plus besoin d'inondation pour connaitre l'appartenance ou non des membres du groupe. De plus, on peut savoir immédiatement qu'un membre du groupe le quitte.

#### 3.1.1.3 Utilisation de unicast

Les traitements de Xcast sont grandement simplifiés par l'utilisation des fonctions d'unicast déjà existantes. De plus cela permet de s'adapter très rapidement à un changement de la configuration du réseau. En effet, si on supprime un routeur dans un réseau multicast, il faut développer un algorithme très lourd pour recalculer les routes alors que les routeurs peuvent généralement recalculer leurs routes unicast très rapidement. Avec Xcast, on a besoin d'utiliser que des nouvelles routes fournies par l'unicast sans y apporter de traitement supplémentaires.

L'utilisation des protocoles unicast permet aussi de profiter des algorithmes optimisés mis en place pour ce type de routage. Si une amélioration est apportée à unicast, Xcast peut en profiter directement.

#### 3.1.1.4 Gestion de groupe

Comme expliqué précédemment, la gestion de l'appartenance au groupe Xcast est gérée par la source seulement, il n'y a pas besoin de protocoles supplémentaires. Cela permet une gestion simple et rapide du groupe. En effet lorsqu'une machine veut rejoindre ou quitter le groupe, elle n'a qu'à envoyer un message à la source et elle s'occupera de l'ajouter ou de le supprimer du groupe.

Xcast présente donc de nombreux avantages par rapport au multicast mais ce protocole présente tout de même quelques défauts non négligeables.

### 3.1.2 Les défauts de Xcast

#### 3.1.2.1 Temps de traitement des paquets

Un des premiers défauts de Xcast est lié au temps de calcul. Dans l'exemple expliqué précédemment, on voit que les opérations effectuées sur les routeurs $R_1$ et $R_2$ sont inutiles car au final le paquet est envoyé avec la même liste de destinataire vers un seul routeur. Pourtant chaque routeur doit systématiquement, parcourir toute la liste des destinataires ce qui peut prendre beaucoup de temps. Dans certains cas comme pour $R_1$, un simple routage unicast aurait suffit.

Il existe tout de même quelques méthodes qui permettent de diminuer ce problème comme X2U ou l'utilisation de bitmap (voir le tunneling) mais cela ne le supprime pas.

#### 3.1.2.2 Problème de la fragmentation IP

Le problème de la fragmentation des paquets IP est l'un des inconvénients majeurs de Xcast. Comme nous l'avons expliqué dans la partie sur IP, tous les messages sont fragmentés en plusieurs petits paquets lors de leur envoi sur le réseau. Le problème avec Xcast est que l'entête de ces paquets est beaucoup plus important que

pour des paquets classiques à cause du nombre important d'adresses IP qui y est stocké. On perd donc de la place. C'est encore pire quand la taille de l'entête Xcast dépasse celle du paquet ; dans ce cas, on ne peut plus du tout envoyer le paquet. On peut remédier à ce problème en envoyant plusieurs paquets avec à chaque fois une partie des destinataires, mais cela reste assez lourd et peu optimal. Xcast sera plus adapté à des petits groupes et difficilement utilisable avec des grands groupes, contrairement au multicast.

#### 3.1.2.3 Sécurité et anonymat

Le fait de stocker toutes les adresses IP des destinataires dans l'entête du paquet Xcast pose un problème d'anonymat. En effet, si on récupère le paquet au niveau des premiers routeurs, alors on peut connaitre les adresses IP de tous les membres du groupe. On verra par la suite qu'avec l'utilisation de bitmap, on ne supprime plus les adresses IP et donc on est capable de connaitre tous les membres du groupe Xcast au niveau de n'importe quel routeur par lequel passe le paquet.

### 3.1.3 Bilan

Xcast est un protocole de routage multicast explicite très intéressant car il contourne la plupart des problèmes rencontrés avec multicast et il possède de plus une gestion des groupes plus efficace. Cependant il possède quelques inconvénients liés à la fragmentation IP et au temps de traitement. Xcast ne peut pas remplacer le routage multicast pour les grands groupes à cause de ces problèmes. Cependant Xcast reste très adapté pour des petits groupes pour lesquels il n'y aura pas trop de calcul à effectuer.

Notre projet va consister à utiliser le protocole Xcast et à le modifier pour tenter de supprimer quelques un de ces problèmes. C'est pourquoi nous allons dans la suite proposer une étude plus détaillée du protocole Xcast et de son implémentation.

## 3.2 Architecture de Xcast

### 3.2.1 Vue d'ensemble

Dans notre étude, nous avons utilisé Xcast6 : c'est la version d'Xcast utilisant le protocole IPv6. L'implémentation de ce protocole au sein du noyau NetBSD se compose ainsi :

- une interface du module Xcast standard à tous les modules du noyau permettant les échanges entre le noyau Xcast et le noyau standard de NetBSD,
- un noyau du module Xcast, contenant le code des fonctions de Xcast et situé dans le kernel de NetBSD
- une API, LibXcast, fournissant des méthodes d'utilisation des communications Xcast au niveau applicatif.

L'interface et le noyau de Xcast sont inclus dans le kernel de NetBSD. Cela permet une implémentation efficace pour la réception et l'expédition de paquets Xcast6. Le renvoi d'erreur est par ailleurs, géré par ICMPv6.

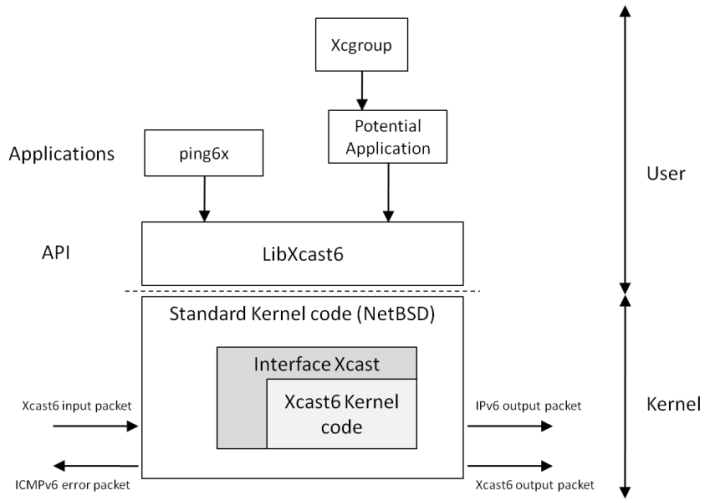



Figure 4: Insertion de Xcast dans le noyau de NetBSD

### 3.2.2 Noyau de Xcast

Le module Xcast est intégré dans NetBSD. Le cœur du protocole de routage se situe dans les fichiers xcast.c et xcast.h. Module Xcast communiquera avec le noyau de NetBSD grâce à une interface, disponible dans les fichiers if_xcst.c et if_xcst.h. Afin d'en savoir plus sur le fonctionnement interne de Xcast, nous avons identifié les méthodes les plus importantes du noyau et de son interface.

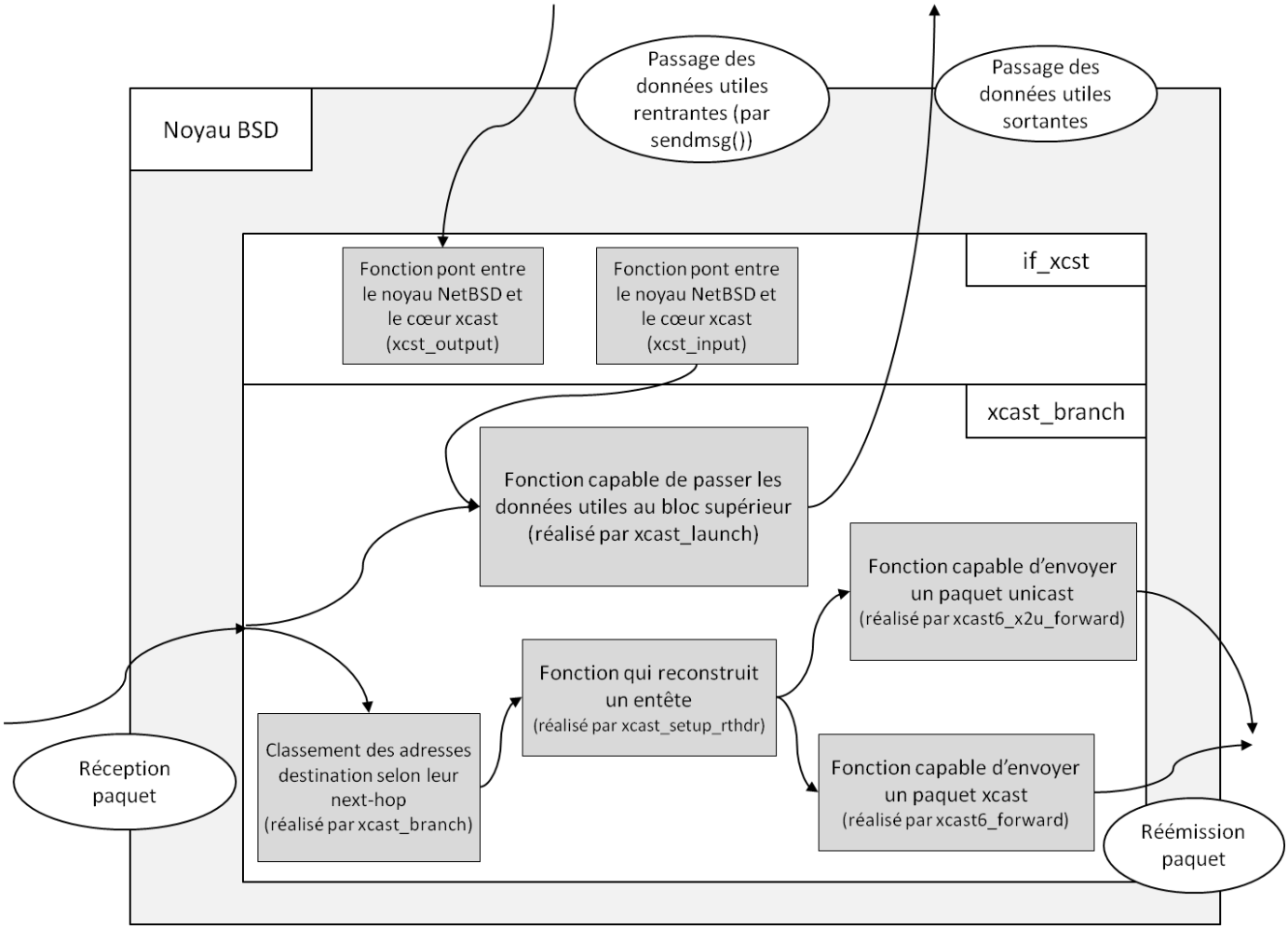



Figure 5: Les fonctions principales du noyau Xcast

#### 3.2.2.1 Code du noyau du module Xcast

La méthode *xcast_branch()* concentre la majorité du code de Xcast. Elle fait appel à un certain nombre de sous fonctions afin de gérer la réception, la création, la réémission d'un paquet Xcast et la transmission des données à la partie applicative.

Lorsqu'on paquet Xcast est réceptionné, la fonction *xcast_branch()* réalise plusieurs opérations. Tout d'abord, elle vérifie l'intégrité de l'en-tête afin de déterminer si celle-ci est correcte et donc analysable. Elle lit par la suite la liste des destinataires du paquet afin de déterminer si certaines adresses sont invalides (comme la présence d'adresse multicast par exemple). Trois cas sont ensuite à considérer:

- Si la machine courante fait partie des destinataires, la fonction *xcast_launch()* est appelée afin de passer les données à la partie applicative.
- Sinon, le paquet est traité en vue d'une réémission.
- Ou les deux

Lors d'une réémission du paquet, *xcast_branch()* détermine quelles sont les adresses de destination à unir au sein d'un même paquet (c'est à dire, les destinations qui ont le même next-hop). Le paquet va alors être divisé

en autant de paquets qu'il y a de next-hop différents. La fonction qui va reconstruire un entête Xcast est *xcast_setup_rthdr().*

Afin de gagner du temps de traitement dans le cas où il faut diviser un paquet Xcast, la fonction xcast_branch() fournit un algorithme permettant de manipuler des bitmaps au lieu des adresses. Un bitmap est un tableau de 64 bits. A chaque bit va correspondre une destination (les adresses sont toujours stockés intégralement) . S'il y a des bits non utilisés, ces derniers sont mis à 0. Un bit est à 1 si le destinataire associé est un destinataire de ce paquet, sinon il est mis à 0. Cette méthode permet de s'affranchir du traitement coûteux de la modification des adresses de destination, en mettant à jour uniquement un bit par adresse.

Avant de réémettre un paquet, deux choix sont possibles:

- S'il n'y a qu'un seul destinataire, on peut envoyer le paquet en unicast grâce à la fonction *xcast6_x2u_forward()*
- Sinon le paquet Xcast est envoyé grâce à la fonction *xcast_forward()*.

#### 3.2.2.2 Interface du module Xcast

Le module Xcast a été conçu sous la forme d'un pseudo-device qui se rajoute au niveau du système d'exploitation NetBSD. Il s'interface donc directement avec le noyau de NetBSD. Le pseudo-device sert à rajouter des fonctionnalités au hardware en place (dans notre cas, ce serait la carte réseau) ou à simuler des pilotes pour un hardware fictif créé par celui-ci. Dans les fonctions qu'il implémente, on retrouve entre autres :

- Xcstattach(int count) : cette fonction est appelée par le noyau et sert surtout à l'allocation mémoire. L'entier en paramètre désigne le nombre de cartes réseaux qu'elle devra gérer.
- Xcstinput(mp, offp, proto) : cette fonction est appelée lors de la réception d'un paquet IPv6. Elle sert à décapsuler l'entête puis appelle alors une nouvelle fonction se trouvant dans le kernel de Xcast : Xcast_launch().
- Xcst_output(if, m0, dst, rt) : cette fonction est appelée lors de la création d'un paquet Xcast afin de construire correctement le Routing Header et appelle la fonction Xcast_branch() .

### 3.2.3 LibXcast

A partir d'une application, il est possible de recourir aux fonctions du module de Xcast grâce à l'API Xcast. Elle fournit des méthodes pour gérer les groupes et l'envoi de message. La déclaration des fonctions sont regroupés dans le fichier libxcast.h.

Plusieurs structures font leur apparition:

- *xcast_group* : cette structure représente un groupe. On y trouve par exemple l'identifiant du groupe, le nombre de destinataires, et des informations sur le paquet IP (soit IPv4, soit IPv6).
- *xcast_grpentry*: cette structure contient la structure *xcast_group*. Elle va permettre de chaîner des groupes entre eux afin d'éviter d'avoir une liste de destinataires trop importante dans l'entête. Afin de réaliser ce chaînage, on trouve l'identifiant du groupe suivant, un pointeur sur le groupe correspondant, l'identifiant du dernier sous-groupe ainsi qu'un booléen déterminant si ce groupe est bien un sous-groupe. Plusieurs méthodes sont disponibles pour gérer ce chaînage: allocation mémoire, libération d'un groupe, création et suppression de groupes et sous-groupes.
- *xcast_member*: les membres d'un groupe sont représentés par cette structure. L'API fournit des méthodes relatives à la gestion des membres pour rechercher, ajouter, supprimer un membre dans un groupe.

L'API fournit également des méthodes pour la gestion des sockets et enfin pour l'envoi de message. La méthode principale utilisée pour l'envoi est *XCast_send* qui prend en paramètre un identifiant de groupe, des données, et la longueur des données. C'est cette méthode qui va se charger de créé le paquet (avec les entêtes Xcast mais aussi IPv6), dans une structure *msghdr*. Le paquet est ensuite envoyé grâce à la fonction standard *sendmsg* de NetBSD.

Il est également important de noter qu'il n'y a pas de réception des paquets Xcast. Ces derniers sont gérés en tant que paquets IPv6 par la fonction standard de NetBSD, recvmsg.

### 3.2.4 Exemple d'application : Ping6x

Ping est une commande souvent utilisée pour savoir si une machine avec une adresse IP est connectée au réseau. Grâce à elle, on peut récupérer le temps aller-retour entre la machine source et la machine distante. Ping6x est une version de Ping pour le protocole Xcast en utilisant des adresses de type IPv6. Un exemple de commande de ping6x est : *Ping6x –c 5 a::1, a::5 1* . Ici, on envoie un message à deux machines d'adresse : a::1 et a::5. Le chiffre se trouvant à la suite de la commande désigne la machine distante qui doit nous répondre (ici, le 1 désigne la deuxième adresse ; pour avoir la première, on aurait alors mis un 0). L'étude de la fonction Ping permettra de trouver les différences entre IPv6 et Xcast sur un cas simple. En comparant Ping6 et Ping6x, on a découvert quelques fonctions qui ont été modifiés :

- Les accesseurs
- *Make_opthdr()* : elle permet la création d'un entête contenue dans un entête de type IPv6(type hph).
- *Make_rthdr()* : elle initialise le bitmap et les destinataires dans l'entête contenu dans l'entête de type IPv6 (type Routing Header).

Cette étude nous permettra une meilleure compréhension des fonctions relatives à Xcast et donc de commencer sur de bonnes bases.

### 3.2.5 Structure d'un paquet Xcast

#### 3.2.5.1 Intégration de Xcast dans un paquet IPv6

Un paquet Xcast est véhiculé dans un paquet IPv6 sur un réseau IPv6. Son entête est une extension IPv6. Du point de vue de Xcast le paquet IPv6 est le paquet englobant et les données véhiculées sont l'entête du protocole de transport (TCP ; UDP) et les données elles-mêmes.

| IPv6 | Xcast6 | Transport | Données |
|---|---|---|---|

Tableau 9: Intégration de Xcast dans un paquet IPv6

#### 3.2.5.2 L'entête Xcast6

L'entête Xcast6 est composé d'une partie fixe de 96 bits et une partie variable.

| Next Header | | | | HdrExtLen | RouteType=Xcast | 0 |
|---|---|---|---|---|---|---|
| VERSION | A | X | D | R | NBR_OF_DEST | CHECKSUM |
| CHANNEL IDENTIFIER | | | | | | |
| BITMAP (64 bits) | | | | | | |
| Liste des destinations | | | | | | |

Tableau 10: L'entête Xcast

#### *Partie Fixe*

- Next Header – contient l'identifiant du protocole suivant, définit une extension spéciale Xcast qui définit les ports.
- HdrExtLen – longueur de l'entête de l'extension
- RouteType=Xcast – spécifie l'utilisation de Xcast
- Version – version de Xcast, ici 6
- A – bit d'anonymat, s'il est mis à 1 les adresses des destinataires qui ne sont pas dans la branche courante vont être mises à 0

- X – bit de Xcast, s'il est mis à 1 le protocole ne va pas utiliser X2U et va acheminer le paquet Xcast jusqu'au destinataire
- D – s'il est mis à 1 Xcast va ajouter à chaque adresse leur priorité
- R – réservé et mis à 00 pour le moment
- NBR_OF_DEST – nombre de destinataires sur 7 bits, donc 128 destinataires maximum
- CHECKSUM – somme le contrôle de l'entête
- CHANNEL IDENTIFIER – identificateur de canal, peut être exploité par l'application mais non nécessaire au routage

### *Partie Variable*

- BITMAP – suite de bits identifiant les adresses des destinataires. Un bit à 1 signifie que le paquet doit être envoyé à cette adresse, sinon le destinataire est dans une autre branche de l'arbre multicast et le paquet ne doit pas lui être envoyé par le routeur courant.
- Liste des destinations – Adresse de chaque destinataire.

## 3.2.5.3 Utilisation des ports

Si on veut spécifier les ports qui indiqueront par quel processus doivent être traités les données du paquet Xcast, on ajoute une autre extension IPv6 à la suite de Xcast. Cette extension n'est lue que par le destinataire.

| **Next Header** | **HdrExtLen** | **RouteType=Ports** | **Opt_Data_Len** |
|---|---|---|---|
| Liste des ports | | | |

Tableau 11: Extension Xcast rajoutante les ports

- Next Header – l'identifiant du protocole suivant
- HdrExtLen – longueur de cette extension IPv6
- RouteType=Ports – définie l'extension en tant que spécification des ports
- Opt_Data_Len – longueur des données optionnelles
- Liste des ports – une liste des ports pour chaque destinataire

## 3.2.6 Tunneling

Jusqu'ici, tout ce qu'on a présenté fonctionne parfaitement à condition que tous les routeurs connaissent Xcast. Le problème est que, dans la réalité, il faut un certain temps avant d'installer un protocole sur tous les routeurs. Il faut donc adapter le protocole Xcast pour qu'il fonctionne même si certains routeurs ne peuvent pas l'interpréter.

La solution adoptée est le tunneling. Le principe est de créer un « tunnel » entre deux routeurs Xcast en « sautant » les routeurs unicast qui sont entre les deux.
Chaque routeur Xcast a dans sa table de routage une entrée pour l'adresse multicast *All_Xcast_Routers*. Un routeur Xcast peut donc savoir si ses voisins connaissent ou non Xcast (s'il appartient au groupe multicast ou non). Si jamais un message doit passer par un routeur non Xcast pour atteindre un des destinataires, le routeur Xcast va créer le message Xcast correspondant et va ensuite l'englober dans un paquet unicast dont l'adresse destinataire va être un des destinataires (généralement le premier) et dont le Next Header va être Xcast. Ainsi les routeurs non Xcast qui vont recevoir ce paquet vont simplement faire circuler le paquet par routage unicast et le routeur Xcast qui va recevoir ce paquet va savoir que c'est un paquet Xcast grâce au Next Header et va donc pouvoir l'interpréter.

Dans l'exemple présenté sur la figure, on désigne par $X_i$ un routeur connaissant Xcast et par $R_i$ un routeur ne connaissant pas Xcast.

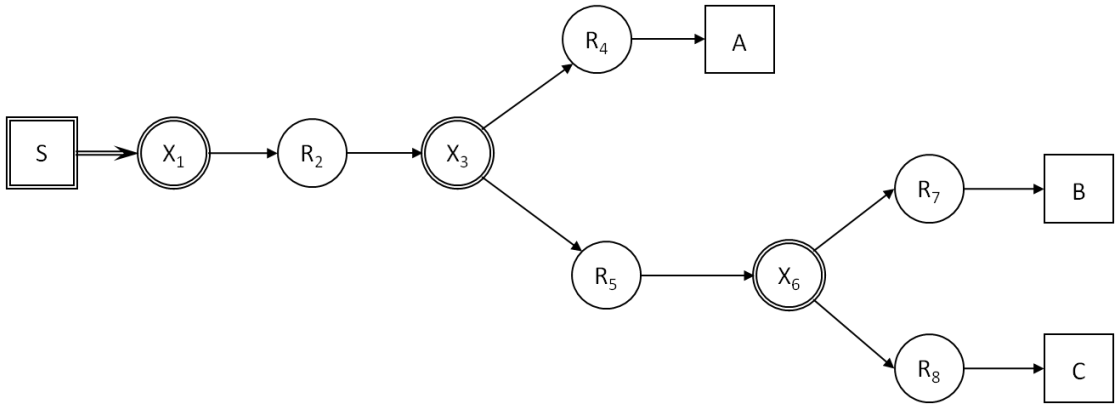



Figure 6: Exemple de tunneling Xcast

S envoi un message Xcast à A, B et C. Comme on l'a vu précédemment, $X_1$ va regarder la liste des destinataires et déterminer le prochain routeur pour chaque destinataire. Ici c'est $R_2$ pour chaque destinataire, $X_1$ va donc envoyer un message Xcast à $R_2$. Or $R_2$ ne connait pas Xcast, $X_1$ va donc envoyer un message unicast à $R_2$ contenant le message Xcast et dont le destinataire est le premier destinataire : A. $R_2$ va faire un simple routage unicast de ce message vers A et l'envoyer à $X_3$. A sa réception, $X_3$ va lire le message Xcast contenu dans le paquet unicast et va pouvoir faire les traitements habituels.

Si jamais $X_3$ n'était pas un routeur Xcast, le message aurait circulé en unicast jusqu'à A. C'est donc A qui devra renvoyer le message Xcast vers B et C.

Le tunneling est donc une technique permettant d' « ignorer » les routeurs ne connaissant pas le protocole Xcast et de pouvoir faire circuler un message Xcast sans problèmes dans n'importe quel réseau.

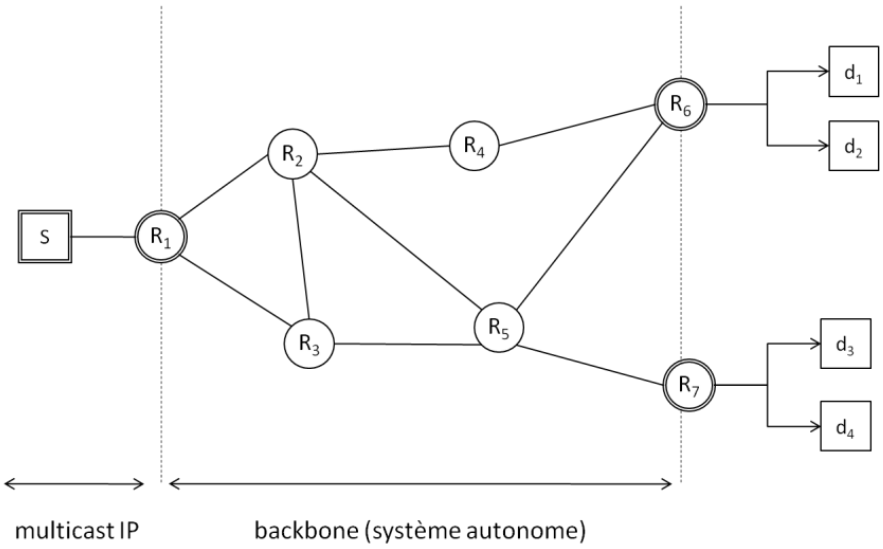
# 4 Protocoles multicast explicite arborescent existants

L'étude de domaine qui précède porte essentiellement sur Xcast, car c'est sur son code source que nous allons nous baser. Malgré tout il existe des protocoles qui ressemblent davantage à celui que nous souhaitons réaliser, le protocole multicast explicite arborescent TBXcast. Les études de l'année passée à propos des deux protocoles multicast explicites arborescents ERM et LINKCAST se sont montrées instructives quant à la conception de TBXcast, et c'est pourquoi nous allons brièvement les présenter dans cette spécification générale.

## 4.1 ERM

Dans ce protocole le routeur responsable de la source gère les groupes multicast. Selon ce protocole on a deux parties dans le réseau :

- backbone – système autonome pour le routage
- les réseaux locaux contenant les machines



Dans le backbone le protocole utilisé est ERM, en dehors il se présente en tant que multicast classique et au cas ou un nouveau destinataire veuille s'abonner au groupe, le routeur correspondant envoie un paquet unicast « trace » au routeur de la source. Ce paquet suit le plus court chemin et va mémoriser les routeurs ERM et non-ERM qu'il a rencontré sur ce chemin.

La superposition de tous les plus courts chemins envoyés par les destinataires permet de trouver les routeurs ERM du réseau et on peut ainsi créer l'arbre de routage au niveau du routeur de la source.

Pour coder l'arbre on va avoir besoin des adresses IP des routeurs et le codage se fait par des pointeurs de chacun des routeurs vers son prédécesseur et cela suffit étant donné que chaque nœud a un seul père dans l'arbre. Cet arbre de routage est codé dans l'entête du paquet à transmettre.

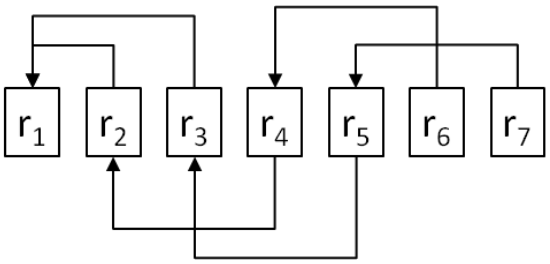



Figure 7: Représentations de l'arbre dans ERM

**Déroulement de l'envoi d'un paquet :**

La source envoie un paquet multicast IPv6 normal avec une adresse multicast et le transmet au premier routeur ERM ($r_1$). $r_1$ fait les branchements selon le protocole ERM en décodant l'arbre de routage et retransmet le paquet aux successeurs concernés ($r_2$ et $r_3$). $r_2$ envoie le paquet à $r_4$ qui le renvoie à son tour à $r_6$. Idem pour la route $r_3$, $r_5$, $r_7$. Enfin, les routeurs terminaux ($r_6$, $r_7$), correspondant aux destinations, vont retransformer le paquet en un paquet multicast IP classique.

## 4.2 LINKCAST

C'est un protocole qui ressemble beaucoup à ERM mais qui mémorise les numéros de routeurs au lieu de leurs adresses IP. L'arbre de routage est codé de manière différente mais utilise l'arbre des plus courts chemins comme ERM. Chaque routeur comporte plusieurs entrées et sorties numérotées et le codage de l'arbre se fait en listant pour chaque numéro de routeur les numéros de sortie qui emmènent aux routeurs LINKCAST voisins mais cela bien évidemment ne peut pas fonctionner avec des réseaux de grande taille.

# 5 Conclusion

Devant les limites de nombreuses technologies et solutions de routage, on voit la nécessité d'un protocole de multicast explicite arborescent. L'unicast n'est pas adapté pour envoyer des données à plusieurs destinataires, le multicast trop lourd lorsqu'il faut gérer beaucoup de groupes, le multicast explicite nécessite encore un traitement conséquent du routeur.

TBXcast vise à proposer une nouvelle solution pour pallier à ces problèmes. Nous avons aussi présenté la structure de Xcast, qui est déjà implémenté dans NetBSD. En effet Xcast présente des fonctionnalités proches de TBXcast, et nous servira de base pour la suite du projet.

# Cahier des charges

Nous allons dans un premier temps présenter la plateforme de développement sur laquelle nous allons travailler. Puis nous présenterons les fonctionnalités de base de TBXcast.

# 1 Plateforme de test et environnement de développement

Pour tester notre protocole et ses algorithmes il faut s'assurer d'avoir une plateforme de test suffisamment réaliste. Pour cela, nous disposons déjà de plusieurs machines contenant elles-mêmes plusieurs cartes réseaux, ce qui va nous permettre de modéliser un réseau.

Grâce à un switch nous allons pouvoir modifier à la volée la topologie de ce réseau, et ainsi représenter les routeurs (par les doublets ou triplets de cartes) ou des ordinateurs (par des interfaces simples).

On disposera de cinq machines dans notre salle, avec trois cartes réseau par machine. On distinguera les activités suivantes :

- Installation de NetBSD 3.1 sur les anciennes machines
- Test de bon fonctionnement d'IPv4, IPv6 et Xcast6 dans cette configuration
- Migration vers les nouvelles machines, test de bon fonctionnement de la configuration précédente sur ces machines
- Passage à NetBSD 4.0 et tests de bon fonctionnement, le tout sur les nouvelles machines.

A noter que si on rencontre des difficultés trop coûteuses en temps à résoudre pour une étape donnée, alors il sera préférable de rester avec la configuration précédemment établie.

# 2 TBXcast

## 2.1 Les apports de TBXcast

Comme nous venons de le voir, les protocoles ERM et LINKCAST se basent sur le plus court chemin, de part la nature des paquets unicast. Malgré tout dans un réseau il y a plusieurs paramètres à prendre en compte pour juger de la validité d'une route ou d'une autre. Ces paramètres sont regroupés dans ce qu'on appelle la QoS ou « Quality of Service » qui considère le délai, la variation du délai, le taux de pertes etc.

Avec le protocole TBXcast, on souhaiterait prendre en compte ces différents paramètres afin de construire un arbre de routage « personnalisé », c'est-à-dire qui soit capable de répondre à des contraintes combinatoires sur les différents aspects de la QoS. En effet, les soucis de taux de pertes et de variation de délai ne seront pas du tout les mêmes si on a affaire à des transferts bancaires ou bien à de la diffusion vidéo.

## 2.2 Problématiques

### 2.2.1 Calcul de l'arbre

Les problèmes d'algorithmique des graphes se cachant derrière la construction de l'arbre sont des problèmes de découverte d'un arbre de coût minimal, et non plus d'un plus court chemin comme le font les protocoles ERM et LINKCAST. Cela engendre des difficultés supplémentaires, et en particulier la complexité du problème s'alourdit fortement : on passe d'une complexité polynômiale à une complexité exponentielle.

### 2.2.2 Segmentation de l'arbre

Lorsque l'arbre atteint une taille trop importante, il est nécessaire de le segmenter afin de l'étaler sur plusieurs entêtes. Si on effectue un découpage brutal, on se rend compte que des redondances apparaissent (de mêmes

nœuds sont présents plusieurs fois) et que la quantité de données transmise sera conditionnée par la taille du plus gros sous-arbre.

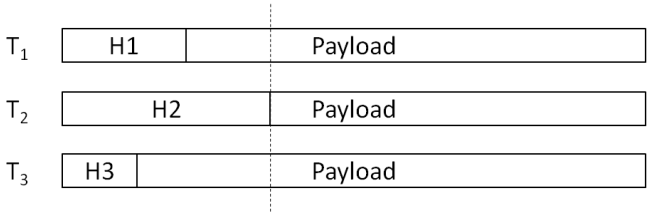



Figure 8: Différence de longueur des entêtes lors de la segmentation

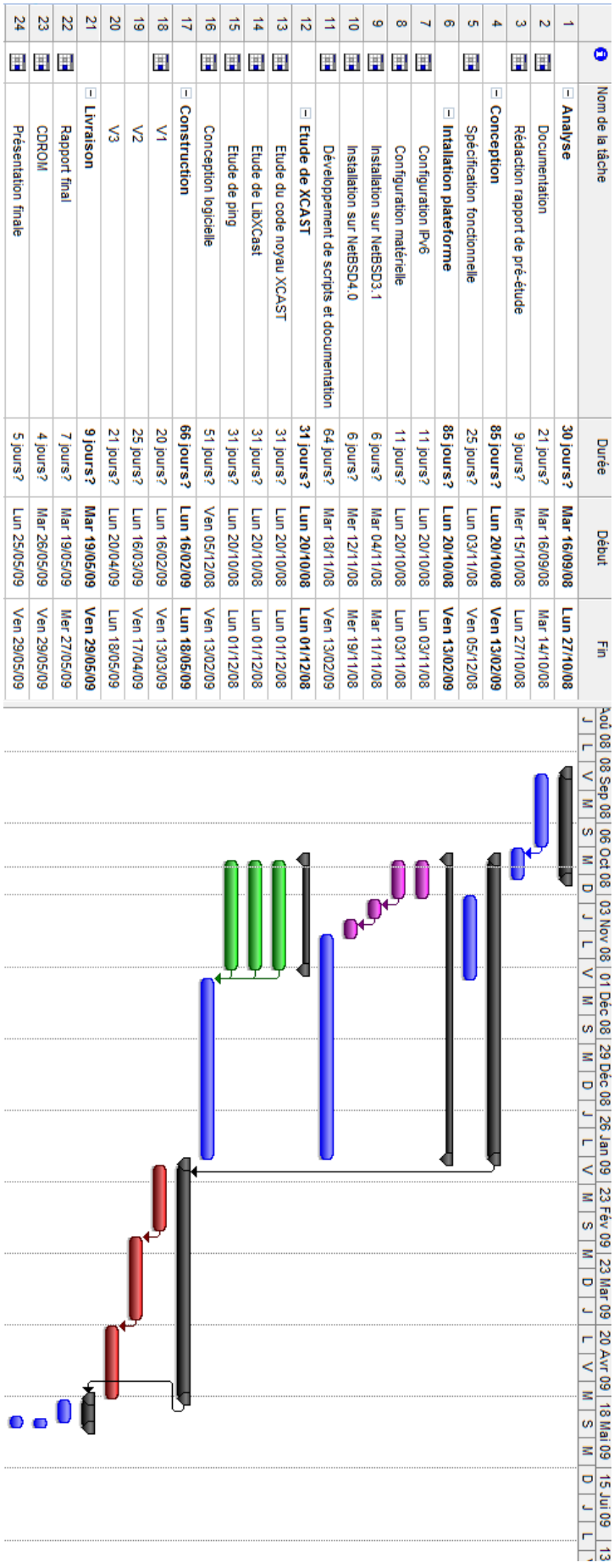
## 2.3 Bilan

La spécification du protocole TBXcast soulève donc de nouveaux problèmes :

- Segmentation de l'arbre
- Gestion des groupes : allons-nous nous baser sur le modèle Xcast ?
- Gestion de la topologie
- A-t-on à faire à un réseau hétérogène ? En d'autres mots, que ferons-nous si le réseau contient des routeurs ne supportant pas TBXcast ?
- Quelles sont les limites de TBXcast ? Son domaine de routage ?

Ces questions représentent la transition de notre projet vers une phase de spécification plus ciblée, dans laquelle nous détaillerons le fonctionnement précis de notre protocole de routage.

# Planification



| | Nom de la tâche | Durée | Début | Fin |
|---|---|---|---|---|
| 1 | **Analyse** | **30 jours?** | **Mar 16/09/08** | **Lun 27/10/08** |
| 2 | Documentation | 21 jours? | Mar 16/09/08 | Mar 14/10/08 |
| 3 | Rédaction rapport de pré-étude | 9 jours? | Mer 15/10/08 | Lun 27/10/08 |
| 4 | **Conception** | **85 jours?** | **Lun 20/10/08** | **Ven 13/02/09** |
| 5 | Spécification fonctionnelle | 25 jours? | Lun 03/11/08 | Ven 05/12/08 |
| 6 | **Intallation plateforme** | **85 jours?** | **Lun 20/10/08** | **Ven 13/02/09** |
| 7 | Configuration IPv6 | 11 jours? | Lun 20/10/08 | Lun 03/11/08 |
| 8 | Configuration matérielle | 11 jours? | Lun 20/10/08 | Lun 03/11/08 |
| 9 | Installation sur NetBSD3.1 | 6 jours? | Mar 04/11/08 | Mar 11/11/08 |
| 10 | Installation sur NetBSD4.0 | 6 jours? | Mer 12/11/08 | Mer 19/11/08 |
| 11 | Développement de scripts et documentation | 64 jours? | Mar 18/11/08 | Ven 13/02/09 |
| 12 | **Etude de XCAST** | **31 jours?** | **Lun 20/10/08** | **Lun 01/12/08** |
| 13 | Etude du code noyau XCAST | 31 jours? | Lun 20/10/08 | Lun 01/12/08 |
| 14 | Etude de LibXCast | 31 jours? | Lun 20/10/08 | Lun 01/12/08 |
| 15 | Etude de ping | 31 jours? | Lun 20/10/08 | Lun 01/12/08 |
| 16 | Conception logicielle | 51 jours? | Ven 05/12/08 | Ven 13/02/09 |
| 17 | **Construction** | **66 jours?** | **Lun 16/02/09** | **Lun 18/05/09** |
| 18 | V1 | 20 jours? | Lun 16/02/09 | Ven 13/03/09 |
| 19 | V2 | 25 jours? | Lun 16/03/09 | Ven 17/04/09 |
| 20 | V3 | 21 jours? | Lun 20/04/09 | Lun 18/05/09 |
| 21 | **Livraison** | **9 jours?** | **Mar 19/05/09** | **Ven 29/05/09** |
| 22 | Rapport final | 7 jours? | Mar 19/05/09 | Mer 27/05/09 |
| 23 | CDROM | 4 jours? | Mar 26/05/09 | Ven 29/05/09 |
| 24 | Présentation finale | 5 jours? | Lun 25/05/09 | Ven 29/05/09 |

Notre projet va se décomposer en quatre grandes phases : l'analyse, la conception, la construction et la livraison.

La phase d'analyse, qui se termine avec ce rapport, nous a permis de nous familiariser avec les notions de base sur le réseau et à commencer l'étude en profondeur de Xcast.

La phase de conception nécessitera la remise en fonctionnement de la plateforme de test. En parallèle, nous finirons d'étudier le code de Xcast ce qui nous permettra d'établir les fonctionnalités de TBXcast. Pour la mi-février, nous aurons déterminé avec précision les spécificités de TBXcast ainsi que les méthodes à adopter lors de la phase de construction qui suivra.

La phase de construction se fera sous la forme d'une succession de versions - phases de développement / tests – qui apporteront chacune des fonctionnalités supplémentaires au protocole.

Enfin, la phase de livraison correspondra aux finitions, à la rédaction du rapport final ainsi qu'à la présentation de notre projet.

# Bibliographie


Basseville, R., Bonnet, L., Chene, A.-S., Gao, F., Garnier, M., Georgescu, N., et al. (2007/2008). *Projet TBXCast : Rapport de préétude.*

*Border Gateway Protocol*. (s.d.). Récupéré sur Wikipedia: http://fr.wikipedia.org/wiki/Border_Gateway_Protocol

Boudani, A., Guitton, A., & Cousin, B. (s.d.). *GXcast : une généralisation du protocole Xcast*. Récupéré sur IRISA: http://www.irisa.fr/prive/bcousin/Articles/Majecstic-2003.pdf

Cousin, B. (s.d.). *Les protocoles de routage multicast*. Récupéré sur IRISA: http://www.irisa.fr/prive/bcousin/Cours/Le_multicast.routage.2P.pdf

*Explicit Multicast (Xcast) Concepts and Options*. (2007, Novembre). Récupéré sur RFC Archive: http://www.rfc-archive.org/getrfc.php?rfc=5058

Guitton, A. (2005, Octobre). Communications multicast : contributions aux réseaux optiques et au passage à l'échelle. *Thèse de doctorat* . Université de Rennes I.

*Internet*. (s.d.). Récupéré sur Comment ça marche: http://www.commentcamarche.net/contents/internet/internet.php3

*Internet Group Management Protocol*. (s.d.). Récupéré sur Wikipedia: http://en.wikipedia.org/wiki/Internet_Group_Management_Protocol

*IP Multicast*. (s.d.). Récupéré sur Cisco: http://www.cisco.com/en/US/products/ps6552/products_ios_technology_home.html

*IPv6 Address Types*. (2003, Mars). Récupéré sur Microsoft Technet: http://technet.microsoft.com/en-us/library/cc757359.aspx

*Multicast*. (s.d.). Récupéré sur Wikipedia: http://en.wikipedia.org/wiki/Multicast

*Open Short Path First*. (s.d.). Récupéré sur Wikipedia: http://fr.wikipedia.org/wiki/Open_shortest_path_first

*Routing Information Protocol*. (s.d.). Récupéré sur Wikipedia: http://fr.wikipedia.org/wiki/Routing_information_protocol

Seret, D. (2005, Octobre). *Réseaux et Protocoles*. Récupéré sur http://www.math-info.univ-paris5.fr/~seret/Reseaux_Protocoles_partie2.pdf