Un Etat de l'Art sur les Techniques de Transmission Multipoint Fiable *†

Vincent Roca
INRIA Rhône-Alpes, projet Planète, Grenoble, FRANCE
vincent.roca@inrialpes.fr
http://www.inrialpes.fr/planete/people/roca/

19 septembre 2001

Abstract

Après une dizaine d'années d'intenses efforts de recherche et de déploiement, les technologies de communication multipoint connaissent aujourd'hui une certaine maturité. Dans cet article nous abordons les aspects applicatifs (par opposition aux techniques de routage multipoint) et plus particulièrement :

- la fourniture d'un service de communication de groupe fiable, et
- les solutions de contrôle de congestion qui leur sont associées.

Ces deux aspects sont en effet des points clef et font l'objet depuis un an et demi de travaux de standardisation au sein du groupe RMT ("Reliable Multicast Transport") de l'IETF. Nous décrivons les principales approches suivies et leur champ d'application. Ainsi, concernant la fourniture d'un service fiable, nous nous intéressons :

- aux protocoles NORM, TRACK et ALC qui sont les trois instances identifiées par l'IETF. Nous insisterons plus particulièrement sur le protocole ALC qui est une technologie majeure pour la diffusion de contenus très populaires sur le WEB (vidéos, logiciels...).
- à PGM, proposé par CISCO, qui est une instanciation du protocole TRACK.
- à l'utilisation de codes FEC ("Forward Error Correction"), indispensables pour un passage à l'échelle.

Concernant les technologies de contrôle de congestion, nous nous intéressons aux principaux protocoles en cours de standardisation : PGMCC et RLC/FLID-SL/FLID-DL.

mots clef: multicast fiable, communication de groupe fiable, NORM, TRACK, ALC, LCT, PGM, FEC.

*This work is partially supported by the Distributed Systems Engineering (DSE) European Community project IST-1999-10302. For more information see the official DSE home page: http://cec.to.alespazio.it/DSE/

†Publié lors des 4èmes Journées Réseaux (JRES'01), Lyon, France, décembre 2001.

1 Introduction

1.1 Motivations pour l'Utilisation des Techniques de Communication Multicast

Plusieurs raisons peuvent motiver l'utilisation des techniques de communication de groupe, et en particulier du service de routage multicast.

La première est la *scalabilité (ou passage à l'échelle) en terme de nombre de participants.* Chaque information n'étant transmise qu'une seule fois (au lieu d'être transmis individuellement pour chaque récepteur), une session peut contenir un très grand nombre de récepteurs sans que cela ne pose de problème de ressources (réseau d'accès de l'émetteur et capacités de traitement). Ce point est cependant à relativiser lorsque l'on considère la fourniture de services supplémentaires (telle la fiabilité), les mécanismes utilisés pouvant induire d'autres limitations.

La seconde motivation, conséquence directe de l'aspect scalabilité, est la *réduction des coûts de transmission.* Une diffusion à plusieurs millions de personnes d'un contenu populaire peut se faire sans disposer d'un très gros réseau d'accès.

Une troisième motivation est l'augmentation des vitesses de transfert par rapport à la solution point-à-point lorsque le nombre de récepteurs augmente, la bande passante du réseau d'accès n'étant pas à partager.

1.2 Quelques Applications Concrètes

On peut distinguer plusieurs types d'applications pouvant tirer profit du multicast :

- *le streaming audio/vidéo* : la quantité de données transmises à un nombre important d'utilisateurs requiert en effet des moyens de transmission efficaces, et un service temps réel semi fiable.
- *la diffusion de contenu "à la demande"* : des serveurs diffusant des contenus très populaires (clip vidéo, logiciels, mise à jour, etc vus en pratique comme des fichiers) peuvent tirer profit du multicast pour transmettre en continu les données que les utilisateurs pourront télécharger à la demande. Les transmissions se faisant en boucle, le service est totalement fiable. Il doit être scalable mais en revanche il n'y a pas de contrainte temps-réel.
- *la diffusion de contenu en mode "push"* : il s'agit d'une application de diffusion qui se fait à l'initiative de la source et non des récepteurs. Le contenu est là aussi constitué d'un ou plusieurs fichiers. Le service doit être totalement fiable et scalable.

Ces scénarios serviront dans différents cas. Ainsi la diffusion au sein d'un groupe multi-localisé de nouvelles versions de catalogues, de documents internes, etc pourra se faire avec une approche "push". L'aspect sécurité sera souvent déterminant dans ce cas. Un autre exemple d'utilisation est constitué par la diffusion de données issus des marchés financiers. Le modèle d'utilisation pourra être à cheval entre les modes "push" et "à la demande". La fiabilité est ici essentielle. Les outils de visio-conférence, de travail coopératif, les jeux multi-utilisateurs, les techniques de synchronisation de caches ou de serveurs

utilisées par exemple par les "content delivery network" (CDN), etc sont autant d'exemples d'utilisation.

1.3 Le Problème de Déploiement du Routage Multicast

Nous avons vu les utilisations possibles des services de communication de groupe. Mais qu'en est-il de la disponibilité effective de ces services ? Plus précisément le routage multicast permet-il d'atteindre tous les récepteurs potentiels ?

Au sein d'un réseau local, la réponse est oui. Les technologies physiques de communication sont pour la quasi totalité basées sur la diffusion (Ethernet, Wavelan). Même dans le cas où les équipements (un commutateur Ethernet par exemple) offrent une séparation des trafics, le trafic multicast est correctement et automatiquement acheminé vers toutes les machines intéressées (les commutateurs sont des équipements intelligents qui analysent les paquets IGMP grâce auxquels les machines expriment leur intérêt à recevoir le trafic multicast).

Au sein d'un site, la réponse est probablement oui. En effet mettre en place un service de routage multicast est relativement aisé, les protocoles largement répandus et les problèmes techniques limités.

Par contre *au niveau de l'Internet la situation est toute autre.* Si les réseaux universitaires offrent un service de routage multicast (tels le FMBONE, ou French Multicast BackbONE, en France [8]), il en est différemment des ISP privés [7]. Plusieurs raisons existent, d'ordre technique (ainsi certains aspects du routage multicast inter-domaines sont encore du domaine de la recherche), ou marketing (quel modèle de tarification utiliser ?), ou plus simplement liées à la demande (problème de l'oeuf et de la poule). Pour parer à ces problèmes, différentes solutions seront utilisées, la plus courante et la plus simple étant le recours à un réflecteur, c'est à dire une machine connectée au MBONE et servant de relais aux correspondants n'ayant accès qu'au routage point-à-point.

2 Le Service de Transmission Multicast Fiable

2.1 Les Challenges

Nous avons vu que nombre d'applications reposant sur le routage multicast ont besoin de services supplémentaires, et en particulier de la fiabilisation totale ou partielle des échanges. Ce problème est infiniment plus complexe à traiter en multicast qu'en point-à-point vu que la source ne connaîtra en général pas le nombre et l'identité des récepteurs (contrairement à TCP).

Plusieurs points ont été identifiés comme critiques par le RFC 2357 [14] :

- *la scalabilité* : il s'agit du passage à l'échelle en terme de nombre de récepteurs essentiellement (cf. section 2.2)
- *le contrôle de congestion* : les sessions multicasts doivent avoir un comportement équitable vis à vis des flux TCP (cf. section 5)
- *la sécurité* : ce point est adressé par le groupe SMUG, ou Secure Multicast Group, de l'IETF. Déja complexe en point-à-point, la sécurité appliquée au multicast l'est encore plus du fait de problèmes tels la dynamicité du groupe et l'anonymat des membres.

D'autres challenges existent pour ce service :

- faire face aux *besoins très variés* des différentes applications et de leur modèle de transmission : streaming, à la demande, push, avec ou sans contraintes temps-réel. Une solution universelle, applicable partout, n'est donc guère envisageable (cf. section 3.1).
- faire face aux *différents modèles de groupe*, qui peuvent être soit fermés (on connaît les membres et ceux-ci sont fixes), soit semi-fermés (les membres sont connus mais évoluent dynamiquement), soit ouverts (on ne connaît ni le nombre ni l'identité des membres). Le modèle de communication multicast, tel qu'introduit par S. Deering, est totalement ouvert et une source ne connaît rien des récepteurs, du moins au niveau IP (on peut malgré tout le faire au niveau applicatif). Le module de fiabilisation disposera donc de plus ou moins d'informations pour mener à bien son travail.
- être *simple d'utilisation*, que ce soit vis à vis de l'utilisateur final ou de l'API (Application Programming Interface) offert aux developpeurs. Ceci pourra passer en particulier par des techniques d'autoconfiguration (par exemple dans l'approche TRACK, section 3.3).
- être *facilement monitorable*. La complexité des solutions employées rend les dysfonctionnements difficiles à tracer.

2.2 Scalabilité et Multicast Fiable : les Problèmes

La notion de scalabilité appliquée au service de transmission multicast fiable pose de nombreux problèmes. Si l'on ne considère que la scalabilité en terme du nombre de récepteurs, il faudra offrir :

- *Un trafic de contrôle scalable :* Il est clair qu'envoyer un acquittement (ou ACK) à chaque paquet transmis (à la TCP) n'est guère envisageable. Envoyer un acquittement négatif (ou NAK) systématiquement à chaque paquet non reçu pose également problèmes. Ainsi si la perte a lieu près de la source, alors un très grand nombre de récepteurs vont générer un NAK qui risquent alors de faire écrouler la source.
- *Des retransmissions scalables :* Transmettre les paquets perdus en point à point à celui qui en fait la demande n'est clairement pas envisageable. Mais transmettre systématiquement en multicast à tout le monde les paquets perdus ne l'est guère plus.
- *Une gestion de l'hétérogénéité :* Au sein d'un groupe important les récepteurs ont une forte chance d'être largement hétérogènes des points de vue du réseau d'accès et des capacités de traitement.

Nous allons maintenant aborder chacun de ces points et voir les principales solutions adoptées.

2.3 Scalabilité du Trafic de Contrôle

Le trafic de contrôle est utilisé essentiellement à la fiabilisation des transferts (paquets NAK voir ACK) et au contrôle de congestion. Trois approches sont adoptées afin d'en limiter le volume.

2.3.1 Utilisation de Mécanismes de Suppression chez les Récepteurs

Une première approche consiste à limiter le nombre d'informations de contrôle qu'un récepteur peut être tenté d'émettre. Pour cela deux solutions principales existent :

- La première consiste à *transmettre en multicast les paquets NAK.* Ainsi tous les voisins ayant connu les mêmes pertes de paquets (une situation courante en cas de perte par engorgement d'un routeur) seront informés qu'une demande de retransmission. De plus pour limiter les collisions de NAKs, chaque noeud utilise un "backoff timer" dont la valeur est choisie aléatoirement. Pour garantir une meilleure scalabilité, cette valeur devra dépendre de la distance à la source. Aussi attrayante soit elle, cette solution :
  - nécessite une évaluation du RTT entre chaque source et récepteur, et
  - nécessite un service de routage multicast de type m-vers-n qui n'a peut être plus le vent en poupe[1].
- Une deuxième solution consiste à *transmettre les NAK en point-à-point vers la source mais en utilisant un "backof timer" exponentiel.* Cette solution s'est avérée être bien plus scalable et est adoptée par la plupart des propositions de multicast fiable orienté NAK.

Notons que ces solutions recherchent un *compromis* entre réagir rapidement et limiter le nombre de requêtes.

2.3.2 Utilisation du Forward Error Correction (FEC)

Les codes FEC sont des codes correcteurs d'erreurs [12]. Un paquet FEC peut ainsi se substituer à n'importe quel paquet de données qui aurait pu être perdu (plus ou moins vrai suivant le codec FEC utilisé, cf. section 4) et ainsi éviter la génération de NAK. Deux utilisations (complémentaires) peuvent en être faites :

- *par anticipation :* la source suppose qu'il y aura un certain taux de pertes et ajoute systématiquement dans le flux de données transmis des paquets FEC supplémentaires. Le problème est ici de savoir à l'avance combien de FEC ajouter. Il y a un équilibre à trouver entre overhead de transmission et capacité de recouvrement sur perte. Les conditions réseaux évoluant dynamiquement, cet équilibre devrait en théorie constamment être remis en cause.
- *par réaction :* Plutôt que de demander la retransmission d'un ou plusieurs paquets expréssement identifiés, un récepteur demande la retransmission d'un certain nombre de paquets. La source transmettra en fait un certain nombre de paquets FEC du fait de leur propriété de se substituer à n'importe quel paquet de données et donc satisfaire plusieurs récepteurs ayant subit des pertes différentes.

[1] La tendance actuelle est de se tourner vers des modèles de routage multicast à source unique, 1-vers-n, plus simples à mettre en oeuvre [3] : ainsi PIM-SSM, ou variante Single Source de PIM.

En raison de leur efficacité, toutes les approches multicast fiable utilisent du FEC selon l'une (ou les deux) modalités ci-dessus (la solution par réaction est en général systématiquement utilisée). La section 4 donne un aperçu du domaine.

2.3.3 Utilisation d'Arbres

Une dernière solution, relativement complexe à mettre en place, consiste à utiliser un arbre au sein du réseau (donc ajouter des fonctionnalités multicast fiable à des routeurs ou serveurs). Les noeuds de l'arbre permettront de supprimer des NAK redondants (car demandant la retransmission des mêmes données) issus de deux branches inférieures, ou bien d'agréger des ACK redondants lorsque tous les récepteurs en dessous ont confirmé avoir reçu les données. TRACK (section 3.3) et PGM (section 3.3.2) sont deux exemples de protocole suivant ce modèle.

2.4 Scalabilité des Retransmissions

On retrouve ici les mêmes techniques que précédemment, à savoir l'utilisation du FEC [12] et d'arbres.

2.4.1 Utilisation du FEC

Cf. section 2.3.2.

2.4.2 Utilisation d'un Arbre de Serveurs de Retransmission

On a vu dans la section 2.3.3 l'importance d'un arbre pour limiter le nombre d'informations de contrôle que recevra la source. Une autre utilisation de cette idée d'arbre est de construire une infrastructure de serveurs pouvant répondre à des demandes de retransmission. Ces serveurs doivent posséder une capacité de stockage importante, surtout s'ils sont amenés à gérer plusieurs sessions en parallèle. Faisant office de cache, une question directe est de savoir à quel moment libérer les données contenues dans l'éventualité d'une retransmission.

2.5 Prise en Compte de l'Hétérogénéité

Au sein d'un grand groupe l'hétérogénéité est inévitable. Ceci peut avoir de forts impacts sur l'efficacité de la solution multicast fiable utilisée. Afin d'y faire face, plusieurs approches sont utilisées :

- La source peut s'adapter au rythme du récepteur le plus lent mais sans jamais descendre en deçà d'un certain seuil prédéfini. Cette solution, très courante avec les approches à "couche unique" (section 3.2 et 3.3), a des limites car les récepteurs rapides ont un rythme de réception inutilement faible.

- Les récepteurs sont éclatés en groupes indépendants et regroupant chacun des récepteurs relativement homogènes. Si cette solution prend en compte l'hétérogénéité des récepteurs, d'autres problèmes surviennent lorsque l'on doit changer de groupe suite à une évolution des conditions réseaux. En effet, dans le cas d'une transmission en mode "push", tout changement de groupe va nécessiter de se recaler sur les transmissions en cours et donc de

récupérer les paquets manquants. Comment le faire sans trop perturber le groupe ?

- La source peut utiliser des techniques de transmission en couches multiples (cf. section 3.4). Les rythmes de transmission étant progressifs sur ces couches, un récepteur rapide s'abonnera et recevra un grand nombre de couches, un récepteur très lent seulement la couche de base. Cette solution répond pleinement au problème d'hétérogénéité.

3 Les Grandes Classes de Protocoles Multicast Fiables

3.1 L'Approche "Building Block" de l'IETF

Si l'on considère la diversité des besoins ainsi que le nombre de problèmes et de solutions techniques pouvant y être apportées, il est clair qu'une solution universelle unique n'est guère envisageable. Conscient de cette situation, une approche modulaire a été adoptée par le groupe *RMT* (ou "Reliable Multicast Transport") de l'IETF :
`http://www.ietf.org/html.charters/rmt-charter.html`
Ainsi on distingue [18] :

- *les briques de base (BB, ou "Building Blocks") :* ce sont des composants de base, utilisables dans plusieurs contextes, et en particulier par les PI.
- *les instances de protocole (PI, ou "Protocol Instanciation") :* ce sont des assemblages de BB destinés à offrir un service répondant à des besoins concrets.

Fort de ce découpage logique, trois grandes classes de protocoles ont été identifiées :

- *NORM*, ou Nack Oriented Reliable Multicast,
- *TRACK*, ou Tree based Acknowledgment, et
- *ALC*, ou Asynchronous Layered Coding.

Nous allons maintenant les examiner à tour de rôle.

3.2 L'Approche NORM, ou NAK Simple

Le protocole (PI) NORM [2], ou Negative Acknowledgment Oriented Reliable Multicast, repose sur la transmission de NAK en cas de pertes. Cette approche est de ce fait simple et semble bien adaptée aux groupes de taille petite ou moyenne dans lesquels les récepteurs sont relativement homogènes. Si ce n'est pas le cas, alors le récepteur connaissant le plus de difficultés sera à l'origine d'un grand nombre de demandes de retransmissions qui viendront perturber et ralentir le flux de données. Si un contrôle de flux à la PGMCC est utilisé, alors ce dernier aura pour effet de réguler le débit d'émission pour satisfaire le récepteur le plus lent.

3.2.1 Les Briques Utilisées

NORM est construit au dessus de plusieurs briques [1] :

- la brique d'émission,
- la brique de gestion des NAK coté récepteur (gère en particulier les mécanismes de suppression),
- la brique de gestion des NAK coté émetteur,
- la brique d'estimation des RTT (utilisée par les mécanismes de suppression de NAK, de contrôle de congestion),
- la brique de contrôle de congestion (par exemple PGMCC),
- la brique d'estimation de la taille du groupe,
- la brique FEC [13], essentielle pour favoriser la scalabilité.

auxquels pourront s'ajouter une brique de sécurité, etc.

3.2.2 Un Vieil Exemple : SRM

Un vieil exemple de protocole de type NORM est SRM [9], ou Scalable Reliable Multicast. Il est intégré par exemple dans l'application `wb`, ou "white board", une des applications classiques du MBONE. Afin de permettre une meilleure scalabilité, les demandes de retransmission ont une portée locale, dans l'espoir qu'un membre pas trop éloigné soit en possession des paquets manquants. Ce protocole, qui a le mérite d'exister et d'être utilisé depuis longtemps, s'est avéré avoir plusieurs limites :

- il nécessite un service de routage multicast de type m-vers-n qui n'a peut être plus le vent en poupe (section 2.3.1),
- il nécessite l'évaluation de RTT entre chacun des membres ce qui s'avère relativement complexe à obtenir.

3.3 L'Approche TRACK, ou Arbre d'Assistance

Le protocole (PI) TRACK [17], ou Tree Based Acknowledgment, ajoute la notion d'arbre d'assistance au protocole NORM (sections 2.3.3 et 2.4.2). Cet arbre, qui peut être construit soit manuellement, soit automatiquement (selon une méthode qui n'est pas définie), est basé sur des noeuds (serveurs ou routeurs) qui offrent une assistance lors de la suppression de NAK, l'agrégation d'ACK (si présents), et les retransmissions. *La scalabilité est donc largement améliorée* par rapport à l'approche NORM, ceci au prix :

- d'une complexité accrue,
- de la nécessité d'identifier des noeuds d'assistance et d'y maintenir un état par groupe,
- le cas échéant d'y prévoir un espace de stockage de paquets afin de répondre aux demandes de retransmission.

3.3.1 Les Briques Utilisées

TRACK est construit au dessus des mêmes briques que NORM, auxquelles s'ajoute la brique GRA [6], ou Generic Router Assist, qui définit les fonctionnalités de l'arbre d'assistance.

3.3.2 Le Protocole PGM

PGM, ou "Pragmatic Generic Multicast", est un protocole proposé par Cisco Systems Inc, et qui a une approche similaire à TRACK/GRA (en fait PGM existait bien avant). Il est basé sur la présence de NE, ou "Network Elements", qui sont soit des routeurs multicast soit des serveurs. Ainsi un NE :

- assiste à la suppression de NAK,
- dans le cas où il s'agit d'un routeur, peut localiser les retransmissions dans le(s) sous-arbre(s) ayant constaté des pertes (pour éviter d'inonder tous les récepteurs avec des paquets qui leur sont inutiles),
- dans le cas où il s'agit d'un serveur, peut retransmettre des paquets qu'il aurait cachés.

PGM offre donc un service de diffusion fiable et scalable, qui peut être amélioré de façon incrémentale par l'ajout de nouveaux NEs. A l'inverse, dans le cas d'un groupe de taille réduite, il est également possible de se passer de NE.

3.4 L'Approche ALC, ou Transmissions en Couches

Le protocole (PI) ALC [10], ou Asynchronous Layered Coding, est radicalement différent des précédants. Nous avons ici un protocole entièrement orienté récepteur, *sans aucun retour d'information (feedback)* à la source. Pour cela ALC repose sur la notion de couches. Ainsi chaque récepteur s'abonne à un plus ou moins grand nombre de couches et reçoit les informations à son rythme propre, sans aucun impact sur les autres récepteurs. L'*hétérogénéité des récepteurs* est donc naturellement prise en compte. Cette propriété fait de ALC une solution de choix lorsqu'il s'agit de favoriser une *scalabilité maximale* et l'on peut aisément envisager des transmissions vers plusieurs millions de clients.

3.4.1 Principes de Fonctionnement

La source ALC définit n_c couches, chacune étant associée à un groupe multicast (et donc une adresse) différent. Les transmissions sur ces couches sont cumulatives et un récepteur recevant la couche i devra impérativement recevoir toutes les couches inférieures {0; i-1}. Les transmissions sur chaque couche se font à un *débit constant* qui a été prédéfini. On note r_i le débit élémentaire de la couche i.

L'adhésion ou le retrait d'un couche sera contrôlé par l'un des protocoles de contrôle de congestion de la famille LCC (section 5.2). L'idée générale est qu'en l'absence de perte un récepteur s'abonne à une nouvelle couche et inversement en présence de pertes un récepteur se désabonne de la couche supérieure.

Le nombre de couches et leur débit définissent ainsi une plage de débits {r_0; sigma(i=0 à n_c-1)(r_i)} permettant de satisfaire simultanément plusieurs récepteurs. Notons qu'en pratique il y a de très grosses différences de performances

entre définir une échelle commençant très bas et disposant d'un grand nombre de couches (mettons 20 couches pour un débit cumulé allant de 25 kbps à 5 Mbps) et définir une échelle commençant bien plus haut mais doté d'un nombre de couches plus restreint (mettons 5 couches pour un débit de 1 Mbps à 5 Mbps). En effet, du fait de l'utilisation d'un contrôle de congestion, le temps nécessaire pour ajouter une couche progresse de façon exponentielle et ajouter une couche élevée devient de plus en plus difficile.

3.4.2 Champ d'Application

ALC est très bien adapté au mode "streaming", surtout si l'application peut se satisfaire d'un niveau de fiabilité partiel. ALC est incontournable dans le cas du mode "à la demande" auquel cas les transmissions se feront en continu, la source recommençant un nouveau cycle de transmission lorsque toutes les données d'une couche ont été transmises. Enfin ALC peut également s'envisager dans le cas de transmissions "push" lorsque le nombre de récepteurs ou leur hétérogénéité sont très importants.

En cas de streaming, la différence entre récepteurs lents et rapides se fera sur la quantité d'informations reçues, ce qui dans le cas d'une séquence vidéo se traduira par une plus ou moins bonne qualité d'image. En cas de transmission en mode "push" ou "à la demande" d'un fichier, la différence entre récepteurs lents et rapides se fera uniquement sur le temps de réception.

3.4.3 Les Briques Utilisées

ALC est construit au dessus de plusieurs briques :

- la brique LCC de contrôle de congestion (section 5.2),
- la brique FEC [13], essentielle dans le cas de transmissions en couches,
- la brique de sécurité (authentification de la source et intégrité des données) qui est fortement recommandée,
- et enfin la brique LCT (Layered Coding Transport) [11] qui constitue le coeur de ALC puisqu'elle définit le format d'en-tête principal et fait le lien avec les précédantes briques.

4 Les Codes FEC, ou Forward Error Correction

4.1 Principes Généraux

Les codes FEC employés dans le cadre des transmissions multicast fiables sont différents de ceux de la théorie de l'information. La problématique n'est pas la même : on utilise ici des canaux à perte (suite à la congestion d'un routeur bien souvent) et non à altérations, on travaille sur des paquets et non des bits, enfin on cherche à ce qu'un paquet FEC puisse se substituer à n'importe quel paquet de données perdu. Cela conduit à utiliser des codes relativement complexes, qui ne sont pas simplement une combinaison XOR des deux ou trois paquets précédants.

Le principe général est le suivant : pour k paquets de données, le codec FEC produit n-k paquets FEC. On a donc en tout n paquets, certains étant

des données, d'autres du FEC. En théorie un récepteur recevant k paquets quelconques pris parmi les n est en mesure de reconstruire les k paquets de données d'origine. En pratique, cette propriété est plus ou moins vraie (voir ci-dessous).

4.2 Classification des Codes FEC

Trois classes de codes existent [12] :

- *les codes à blocs restreints :* Le principal représentant est le code Reed-Solomon [15]. Des contraintes pratiques limitent fortement la valeur de k et n. On aura impérativement k <= n <= 256. De plus des contraintes de temps de codage et décodage limitent encore plus ces valeurs et bien souvent on utilise k=32. Dans le cas de paquets de taille 1024, le fichier sera découpé en blocs de 32 kilo-octets. Cette taille limitée des blocs diminue fortement le pouvoir de correction d'un paquet FEC (il n'a d'intérêt qu'au sein de son bloc) et donc l'efficacité de réception. En dépit de ces limites c'est le code le plus utilisé du fait de l'existence d'implémentations Open-Source [15].

- *les codes à blocs importants :* Ici le paramètre k peut être très grand (n est par contre souvent limité à 2k) et les temps de codage et décodage très faibles. En revanche il faudra recevoir un peu plus de k paquets pour pouvoir reconstruire les paquets de données (entre 5 et 10% supplémentaires). Le code Tornado [5] est l'exemple le plus connu.

  Du fait de ses caractéristiques, il s'agit d'un code FEC de choix. Cependant il fait l'objet de brevets et aucune implémentation OpenSource n'existe.

- *les codes extensibles :* Alors que les deux précédants codes ont un paramètre n de valeur faible par rapport à k, les codes extensibles ont la propriété de permettre de générer un nombre quasi illimité de paquets FEC. Ceci s'avère pratique avec des approches du type ALC ou l'on est confronté au problème de duplication de paquets entre les couches.

5 Les Protocoles de Contrôle de Congestion

Les transmissions multicast sont potentiellement très dangereuses car elles reposent sur UDP, un protocole qui n'offre aucun mécanisme de contrôle de congestion. Cette fonctionnalité doit donc se faire au sein du protocole multicast et fait l'objet d'une brique spécifique. Notons que bien avant que des solutions n'existent, les transmissions étaient régulées à la source à un niveau de débit fixe relativement faible (quelques dizaines de kbps bien souvent). D'autres approches adaptatives faisant usage d'un feedback de la part des récepteurs (bien souvent par le protocole RTCP associé à RTP) ont aussi été expérimentées.

L'objectif recherché est d'être *équitable avec les autres flux*, en particulier avec TCP qui tend à être vu comme la référence en terme d'équité (ainsi on parle de "TCP-fairness"), bien que d'autres définitions d'équité existent. L'utilisation de ces protocoles doit permettre une bonne adaptation aux conditions réseaux tout en *étant stable* (on veut éviter les oscillations) et en *utilisant au mieux*

les ressources réseaux disponibles (pour éviter une sous-utilisation du réseau lorsque de la bande passante est disponible).

Dans cette section nous décrivons deux grandes classes d'approches, l'une adaptée aux protocoles NORM et TRACK, la deuxième à ALC.

5.1 L'approche PGMCC

Le protocole PGMCC (PGM Congestion Control) apporte un service de contrôle de congestion (équitable avec TCP) aux transmissions n'utilisant qu'un seul flux (par exemple avec les approches NORM et TRACK). Il repose sur :

- *une approche par fenêtre de transmission qui émule la partie contrôle de congestion de TCP :* cette fenêtre évolue en fonction des acquittements (ACK) envoyés par un récepteur bien identifié, le "ACKer" qui de ce fait permet à la source de réguler son débit d'émission.

- *une procédure de sélection du "ACKer" :* son rôle est de s'assurer qu'en présence de plusieurs récepteurs, le "ACKer" est effectivement *le récepteur ayant le plus faible débit équivalent TCP*. L'identité du récepteur faisant office de "ACKer" évolue donc dynamiquement. Cette procédure est basée sur une équation qui modélise le comportement de TCP et permet à la source d'estimer le débit d'une connexion TCP point-à-point entre la source et chaque récepteur : debit = constante / (RTT * sqrt(taux_de_pertes)). Les paramètres RTT et taux de perte sont retournés par un récepteur au moyen de paquets de contrôle.

Notons que PGMCC ne s'intéresse absolument pas à la fonction de fiabilisation des transferts qui lui est totalement découplée.

5.2 L'approche LCC et les protocoles RLC, FLID-SL et FLID-DL

Les techniques LCC, ou "Layered Congestion Control", regroupent quelque protocoles de contrôle de congestion dédiés aux protocoles multicast utilisant plusieurs couches (tel ALC). Le principe général est (à l'exception de FLID-DL) le suivant :

- en l'absence de perte durant un certain intervalle de temps, un récepteur est autorisé à ajouter une nouvelle couche lors d'un certain signal transmis par la source;

- en présence d'une perte, un récepteur doit se désabonner immédiatement de la couche la plus élevée et entre alors dans une phase où tout est gelé. Ceci permet au protocole de routage multicast de supprimer la branche de l'arbre de diffusion et donc de solutionner le problème de congestion supposé.

- a l'issue de cette période de gel, le mécanisme de contrôle de congestion normal reprend.

Trois principaux protocoles existent :

- *RLC (Receiver-driven Layered Congestion Control)* [16] : Un des premiers protocoles proposés, il synchronise les ajouts/retraits de couches par le biais de signaux (ou SP, Synchronization Point) inclus dans certains paquets. RLC reste simple à mettre en oeuvre.
- *FLID-SL (Fair Layered Increase/Decrease - Static Layer)* [4] : Cette version à couches statique reste très proche de RLC.
- *FLID-DL (Dynamic Layer)* [4] : La version à couches dynamiques de FLID rompt totalement avec les deux protocoles précédants. Les récepteurs doivent, pour maintenir le même rythme de transmission, ajouter périodiquement une couche. En revanche, pour diminuer le rythme de réception aucune action n'est nécessaire.

  L'avantage majeur de cette approche est de permettre une *réaction très rapide face à des situations de congestion*, plus rapide que dans le cas de RLC et FLID-SL qui sont tributaires de la latence du protocole IGMP (plusieurs secondes). En revanche FLID-DL est bien plus complexe et génère un important trafic de signalisation au niveau des protocoles de routage multicast.

6 Quelques Implémentations OpenSource

Du fait du caractère relativement récent des documents de l'IETF, on peut distinguer deux catégories d'implémentations :

- celles qui utilisent des protocoles propriétaires, et
- celles qui suivent les standards en cours de développement.

On ne considère ici que les implémentations qui sont livrées avec le code source et distribuées selon une licence ouverte (GNU GPL, ou BSD...). On trouvera principalement dans la première catégorie :

- *RMF :* TASC Inc., Protocole à base de NAK, C++ et Java.
  `http://www.tascnets.com/newtascnets/Software/RMF/index.html`
- *SRM :* la bibliothèque SRM, utilisée jadis par l'outil "wb", est désormais disponible par le biais du consortium Open-Mash (il ne s'agit pas d'une application de transfert de fichier prête à l'emploi mais d'un toolkit).
  `http://www.openmash.org/`

On trouvera dans la deuxième catégorie :

- *JRMS :* Sun Microsystems, Protocole TRACK, Java.
  `http://www.experimentalstuff.com/Technologies/JRMS/index.html`
- *MCL :* INRIA Rhône-Alpes - Université Paris 6, Protocole ALC, C.
  `http://www.inrialpes.fr/planete/people/roca/mcl/`
- *Swarmcast :* SourceForge, Protocole ALC, Java.
  `http://sourceforge.net/projects/swarmcast/`

Nous ne mentionnons pas ici les prototypes de recherche, très nombreux, qui n'ont qu'un champ d'application très réduit et ne sont plus maintenus.

Notons que d'autres implémentations existent et sont commercialisées (Talarian, Digital Fontain, etc).

7 Conclusions

Nous avons brossé dans cet article un panorama des techniques de diffusion fiable de l'information. Nous avons vu que de nombreuses approches ont été élaborées afin de faire face aux différents besoins applicatifs. En effet il n'existe pas dans le domaine des transmissions multicast fiables de protocole universel (contrairement à TCP dans le cas "simple" des communications point-à-point). Ainsi les approches NORM, TRACK et ALC ont toutes leur domaine d'application. Les deux premières sont bien adaptées aux transmissions en mode "push" lorsque le nombre de récepteurs est raisonnable, l'approche ALC aux transmissions en mode "streaming" et "à la demande" avec un nombre quasi illimité de récepteurs.

Nous avons aussi décrit des protocoles de contrôle de congestions utilisables avec ces approches et les codes FEC qui sont incontournables lorsque l'on désire améliorer la scalabilité d'une solution.

Remerciements

Tous mes remerciements vont à Julien Labouré qui a largement contribué au projet MCL de bibliothèque multicast ainsi qu'à B. Whetten pour son tutoriel donné à NGC2000 dont je me suis largement inspiré.

References

[1] B. Adamson, C. Bormann, M. Handley, and J. Macker. *NACK-Oriented Reliable Multicast (NORM) Protocol Building Blocks*, July 2001. Work in Progress: <draft-ietf-rmt-bb-norm-02.txt>.

[2] B. Adamson, C. Bormann, M. Handley, and J. Macker. *NACK-Oriented Reliable Multicast Protocol (NORM)*, July 2001. Work in Progress: <draft-ietf-rmt-pi-norm-02.txt>.

[3] S. Bhattacharyya, C. Diot, L. Giuliano, R. Rockell, J. Meylor, D. Meyer, G. Shepherd, and B. Haberman. *An Overview of Source-Specific Multicast(SSM) Deployment*, August 2001. Work in Progress <draft-ietf-ssm-overview-01.txt>.

[4] J. Byers, M. Frumin, G. Horn, M. Luby, M. Mitzenmacher, A. Roetter, and W. Shaver. Flid-dl: Congestion control for layered multicast. In *2nd Workshop on Networked Group Communication (NGC2000)*, November 2000.

[5] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A digital fountain approach to reliable distribution of bulk data. In *ACM SIGCOMM'98*, August 1998.

[6] K. Calvert, C. Papadopoulos, T. Speakman, D. Towsley, and S. Yelamanchi. *Generic Router Assist - Functional Specification*, July 2001. Work in Progress: <draft-ietf-rmt-gra-fspec-00.txt>.

[7] C. Diot, B. Levine, B. Lyles, H. Kassem, and D. Balensiefen. Deployment issues for the ip multicast service and architecture. *IEEE Network*, pages 78–88, January 2000.

[8] Unité Réseaux du CNRS. *La Page de l'équipe FMbone*, 2001. URL: http://www.urec.cnrs.fr/fmbone/.

[9] S. Floyd, V. Jacobson, S. McCanne, C. Liu, and L. Zhang. A reliable multicast framework for light-weight sessions and application level framing. In *IEEE SIGCOMM'95*, 1995.

[10] M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, and J. Crowcroft. *Asynchronous Layered Coding (ALC): a massively scalable content delivery transport*, October 2001. Work in Progress: <draft-ietf-rmt-pi-alc-03.txt>.

[11] M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, M. Handley, and J. Crowcroft. *Layered Coding Transport (LCT): a massively scalable content delivery transport*, October 2001. Work in Progress: <draft-ietf-rmt-bb-lct-02.txt>.

[12] M. Luby, L. Vicisano, J. Gemmell, L. Rizzo, M. Handley, and J. Crowcroft. *The use of Forward Error Correction in reliable multicast*, October 2000. Work in Progress: <draft-ietf-rmt-info-fec-01.txt>.

[13] M. Luby, L. Vicisano, J. Gemmell, L. Rizzo, M. Handley, and J. Crowcroft. *Reliable Multicast Transport Building Block: Forward Error Correction codes*, October 2001. Work in Progress: <draft-ietf-rmt-bb-fec-04.txt>.

[14] A. Mankin, A. Romanow, S. Bradner, and V. Paxson. *IETF Criteria for Evaluating Reliable Multicast Transport and Application Protocols*, June 1998. Request for Comments: RFC2357.

[15] L. Rizzo and L. Vicisano. Effective erasure codes for reliable computer communication protocols. *ACM Computer Communication Review*, 27(2), April 1997.

[16] L. Vicisano, L. Rizzo, and J. Crowcroft. Tcp-like congestion control for layered multicast data transfer. In *IEEE INFOCOM'98*, February 1998.

[17] B. Whetten, Dah Ming Chiu, M. Kadansky, and G. Taskale. *Reliable Multicast Transport Building Block for TRACK*, March 2001. Work in Progress: <draft-ietf-rmt-bb-track-01.txt>.

[18] B. Whetten, L. Vicisano, R. Kermode, M. Handley, S. Floyd, and M.Luby. *Reliable Multicast Transport Building Blocks for One-to-Many Bulk-Data Transfer*, March 2000. Request for Comments, RFC 3048.