Gossip-Based Video Streaming: Beyond Heterogeneous Bandwidth

Arnaud Jegou

Rapport de stage : Gossip-Based Video Streaming: Beyond Heterogeneous Bandwidth

Arnaud Jégou
Encadrant : Davide Frey, Anne-Marie Kermarrec
Équipe : ASAP
Sujet : Gossip-Based Video Streaming: Beyond Heterogeneous Bandwidth

Juin 2010

Table des matières

1 Introduction

Le stage porte sur les protocoles de dissémination de données en pair à pair. Les protocoles de dissémination de données sont des protocoles dont le but est de faire parvenir un flux de données émis en temps réel à un groupe de participants. Ils ont ainsi de nombreux domaines d'application, comme la mise a jour de bases de données, la diffusion de flux *RSS* ou encore le *streaming* audio ou vidéo. C'est sur ce dernier domaine que nous nous focaliserons.

Traditionnellement, tous ces services sont fournis par un nombre limité d'entités centrales chargées de transmettre l'intégralité des données à un ensemble de participants, mais cette architecture pose plusieurs problèmes. Tout d'abord, ces services peuvent demander d'importantes capacités de calcul ou d'envoi de données (en particulier dans le cas du *streaming* de vidéo) qui peuvent avoir des coûts prohibitifs (dans [9] les auteurs estiment que le site de vidéo *youtube* dépense quotidiennement un million de dollars pour ses besoins en bande passante). De plus, le fait que la totalité de la charge repose sur un faible nombre d'entités rend le système très sensible à la défaillance d'une d'entre elles.

Ces limitations, ainsi que le fait que les utilisateurs d'*Intenet* disposent de connexions avec des capacités d'émission de données de plus en plus importante, ont conduit à l'apparition de nouveaux types de protocoles dont l'objectif est de reporter une partie de la charge de travail sur les clients du protocole afin de réduire les coûts en réduisant les besoins en bande passante de la source.

De nombreux protocoles ont ainsi vu le jour, utilisant des méthodes variées afin de répartir au mieux la charge parmis les participants. On peut les classer en trois principaux groupes selon leur manières d'organiser les participants, également nommés noeuds ou pairs :

- Arbres : Les pairs sont organisés de manière hiérarchique dans une structure en forme d'arbre.
- *Mesh* : Les pairs sont organisés de façon moins rigide et non hiérarchique.
- *Gossip* : Aucune structure fixe n'organise les pairs, ces protocoles sont très dynamiques.

Tous ces protocoles, quelque soit leur type, cherchent à reporter au maximum la charge des sources de données sur les pairs, tout en la répartissant aussi bien que possible parmis ceux-ci. Il est en effet important de répartir correctement la charge entre les pairs afin de limiter l'impact des défaillance et d'obtenir de bonnes performances (tout particulièrement lorsque le volume de données à disséminer est important). Dans ce but, la majorité des protocoles existants comportent des mécanismes afin de faire participer de manière égale tous les pairs, par exemple en fixant le nombre de voisins où de fils de chaque noeud. Mais dans les réseaux pair à pair il est fréquent que les participants disposent de capacités hétérogènes et donner à chaque pair la même tache ne permet pas d'exploiter au maximum les capacités de ceux-ci.

Ce n'est que récemment que certains protocoles ont commencé à répartir la charge parmis les pairs en fonction des capacités de chacuns. Il y a par exemple *SplitStream*[1] pour les protocoles basés sur des arbres, *GridMedia*[11] pour les *Mesh* et *HEAP* pour les protocoles de *Gossip*, qui est celui sur lequel portera le stage.

Dans le cas de *HEAP*, la répartition se fait en ajustant le nombre de pairs avec lesquels chaque noeud communique (son $fanout$) en fonction de sa bande

passante et celle des autres pairs. Cette méthode apporte un gain significatif au niveau la répartition de la charge ainsi que de la qualité de réception en comparaison d'un protocole de *Gossip* standard. Néanmoins, comme nous le verrons plus loin, tenir compte uniquement de la bande passante n'est pas suffisant pour une répartition optimale, car il y a d'autres paramètres que le $fanout$ des noeuds qui influent sur le volume de données qu'ils envoient et peuvent donc désequilibrer la répartition de la charge.

L'objectif du stage était d'améliorer $HEAP$ dans plusieurs domaines, l'un d'entre eux étant la répartition de la charge en tenant compte de ces paramètres. Une autre amélioration était de doter $HEAP$ d'un mécanisme d'estimation dynamique de la capacité des pairs, afin de lui permettre d'adapter dynamiquement la contribution de chaque pair en fonction de ses capacités réelles. En effet, $HEAP$ adaptait la participation des noeuds à partir d'une valeur donnée au protocole, qui traduit généralement mal les capacités réelles des noeuds et ne permet pas de tenir compte des variations de celles-ci au cours du temps. Une troisième amélioration consistait à rendre le protocole conscient de la topologie du réseau. En effet, un réseau comme *Internet* est loin d'être homogène, certains points du réseau sont plus proches les uns des autres et certaines zones peuvent être surchargées, ce qui a un impact significatif sur la qualité de transferts des données. Il est donc important de tenir compte de ces irrégularités afin de favoriser les échanges de données entre pairs proches et d'éviter de générer du trafic supplémentaire dans les zones déjà surchargées.

Les différentes solutions envisagées ont été implémentées et testées, tout d'abord via des simulation puis avec des expériences sur *Planetlab*, dont les résultats sont présentés dans la dernière partie de ce document.

2 Streaming en pair à pair

2.1 Types de protocoles

Différents types de protocoles de streaming en pair à pair existent, utilisant des techniques différentes pour atteindre un même objectif, à savoir utiliser de manière aussi efficace que possible les capacités des différents noeuds afin de diffuser les données à un maximum de pairs et dans des délais les plus courts possibles, tout en réduisant autant que possible les besoins en bande passante de la source.

Du point de vue de l'organisation des noeuds, il existe trois principaux types de protocoles, chacun ayant des avantages et des inconvénients.

Tree-based

Les protocoles les plus simples sont ceux basés sur des arbres. Un protocole *Tree-based* classique construit un arbre couvrant tous les participants et les données sont transmises de père en fils.

Cette architecture donne plusieurs avantages :

- Les transmissions se faisant exclusivement de père en fils, il n'y a pas de problème de routage.

– Du fait que les données prennent toujours le même chemin, les temps de transfert de la source vers un noeud sont peu variables.
– La structure peut aisément être construite de manière à s'adapter à la topologie du réseau

La rigidité de cette architecture a également des inconvénients :
– Les feuilles de l'arbre, qui constituent la majorité des noeuds, ne participent pas à la diffusion.
– Lorsque un noeud est défaillant, tous ses fils se trouvent privés de données, les arbres sont donc très sensibles aux défaillances.

Des modifications sur le fonctionnement de ces protocoles permettent de compenser ces inconvénients.
Bullet[6] construit par exemple un seul arbre couvrant, mais des données différentes sont transmises aux différents fils et ceux-ci communiquent entre eux afin de récupérer les données manquantes.
SplitStream[1] construit plusieurs arbres, chacun chargé de transférer une partie des données. Lors de la création des arbres, chaque participant est placé comme noeud interne d'un des arbres, la charge est ainsi bien répartie et les défaillances moins conséquentes.
Même si ce genre de modifications permet de diminuer l'impact des défaillances, celles-ci restent tout de même assez coûteuses.

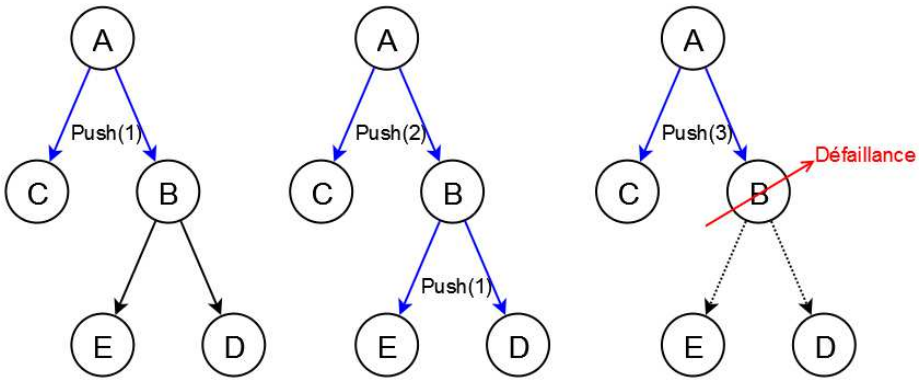


FIG. 1: Défaillance d'un noeud dans un arbre

Mesh-based

Les protocoles basés sur des *Mesh* construisent des réseaux qui sont beaucoup moins structurés que ceux construits par des protocoles basés sur des arbres. Dans ce type de protocole, chaque noeud est connecté à un certain nombre de voisins, sans qu'il y ait de hiérarchie père-fils.

Ceci a certains avantages :
– Les noeuds pouvant récupérer leurs données via plusieurs voisins, les défaillances causent moins de pertes que pour les arbres.

– La récupération d'une défaillance n'entraine généralement pas de réorganisation globale.
– Comme pour les arbres, il est possible de construire une structure tenant compte des spécificités du réseau.
– La charge est relativement bien répartie, car tous les noeuds sont susceptibles de transférer des données à leurs voisins.

Mais également quelques inconvénients :
– Les données arrivant sur un noeud pouvant avoir suivi des chemins très différents, les temps de transfert pour un même noeud peuvent être très irréguliers.
– La structure du réseau fait que le routage des données est beaucoup plus complexe que dans un arbre.

De plus, des irrégularités peuvent se créer dans le réseau ainsi créé, donnant lieu à des zones constituées de noeuds puissants et d'autres uniquement constituées de noeuds faibles.
Dans [9], chaque noeud calcule une estimation de la bande passante moyenne des participants et la compare à celle de ses voisins. Si ses voisins sont plus performants que la moyenne, il remplace celui possédant la plus grosse bande passante et inversement si les bandes passantes de ses voisins sont trop faibles. Ceci permet d'équilibrer correctement le réseau.

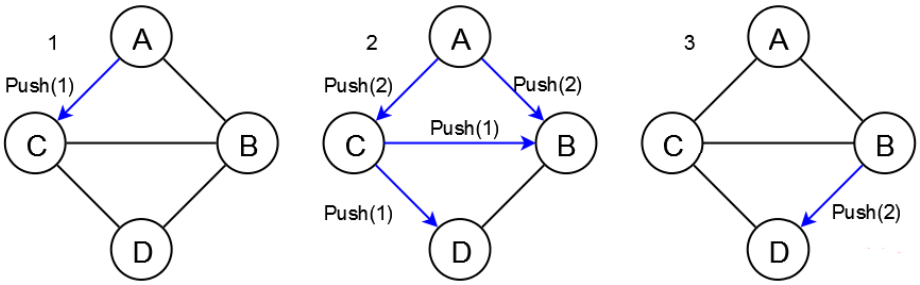


FIG. 2: Exemple de *Mesh*

Gossip

Les protocoles de *Gossip* sont assez différents des deux précédents, car ils ne construisent pas de structure rigide pour connecter leurs noeuds. Leur fonctionnement est basé sur des cycles : à chaque cycle, un noeud choisit des voisins de manière aléatoire parmi l'ensemble des noeuds participants puis échange avec eux les données reçues au cycle précédent. On peut les voir comme une forme complètement dynamique de *Mesh*.

Ce dynamisme apporte les avantages suivants :
– Les voisins étant changés très régulièrement, ces protocoles sont quasiment insensibles aux défaillances.
– Ces protocoles n'ont pas besoin de construire et d'organiser de structure globale, ils n'ont donc pas de problème de passage à l'échelle.

– Les changement fréquents de voisins permettent d'éviter une mauvaise répartition des pairs

Mais le manque de structuration a également des inconvénients :
– Les chemins suivis par les données sont très irréguliers.
– Le fait que les voisins soient changés régulièrement rend difficile la prise en compte de la topologie du réseau.
– La sélection aléatoire des voisins fait que certains pairs peuvent ne jamais se faire proposer certaines données.

Le fait que ces protocoles ne reposent sur aucune structure fait qu'ils sont facilement déployables et qu'ils ne posent pas de problème de mise à l'échelle, comme cela peut être le cas pour les protocoles structurés. De plus, le caractère aléatoire de la sélection des voisins et de l'envoi de données fait que la charge est naturellement répartie parmi les participants. Néanmoins, une répartition égale de la charge entre tous les pairs n'est pas toujours souhaitable, principalement dans le cas ou les différents pairs disposent de capacités différentes.

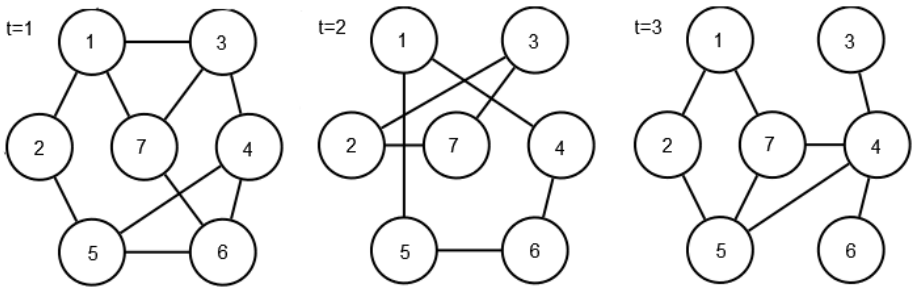


FIG. 3: Évolution des voisins dans un protocole de *Gossip*

2.2 HEAP

HEAP est un protocole de *Gossip* ayant pour principale caractéristique d'adapter la contribution de chaque pair en fonction de sa bande passante. En effet, si les protocoles de *Gossip* sont efficaces lorsque les noeuds sont homogènes, ou que les besoins de bande passante sont bien plus faibles que les capacités des différents noeuds, ils deviennent beaucoup moins efficaces si les capacitées des noeuds sont hétérogènes.

Comme pour les autres protocoles de *Gossip*, les noeuds de *HEAP* basent leurs fontionnement sur des cycles. Au cours de chacun de leurs cycles, les pairs recyclent leur ensemble de voisins en choisissant aléatoirement un nombre f de pairs dans l'ensemble des participants. Ce nombre f, appelé le $fanout$, est utilisé pour répartir la charge équitablement selon la bande passante des pairs. Un pair disposant d'une importante bande passante sélectionnera ainsi, à chaque cycle, un nombre de voisins plus important. Bien que les voisins doivent, pour bien diffuser les données et bien résister aux défaillances, être choisis aléatoirement

dans l'ensemble des pairs, un noeud ne peut en connaître la liste complète. En effet, cela demanderait une quantité de mémoire proportionelle au nombre de participants, ce qui ne passe évidemment pas à l'échelle. Néanmoins, la sélection aléatoire parmi l'ensemble des pairs est importante et pour cela les noeuds font appel à un service, le *PeerSampling*, chargé de fournir, à partir d'une liste de pairs de taille limitée et régulièrement mise à jour, les noeuds en voisins comme s'ils étaient tirés aléatoirement dans l'ensemble des participants.

Le *PeerSampling* se base lui-même sur un protocole de *Gossip*, où les informations sur leur vue partielle sont échangées entre les différents noeuds. Il intègre différents mécanismes visant à propager rapidement les informations concernant les pairs tout en assurant un minimum de fiabilité (du point de vue de la qualité des données partagées) en éliminant les informations jugées trop anciennes et pouvant donc concerner des noeuds défaillants. C'est également ce service qui est utilisé pour calculer une estimation de la bande passante moyenne des pairs.

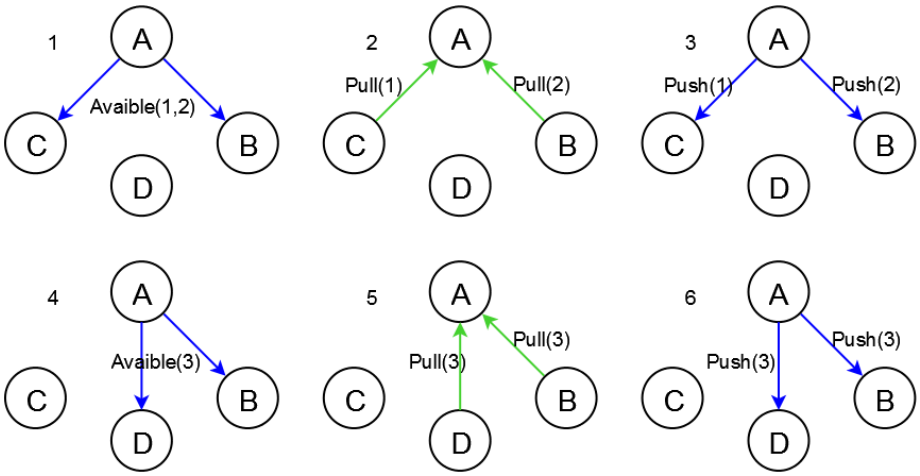


FIG. 4: Fonctionnement de *HEAP*

Une fois son nouvel ensemble de voisins sélectionné, un noeud récupère la liste de données qu'il a reçues au cours du cycle précédent et envoie des messages de propositions concernant toutes ces données à l'ensemble de ses voisins, ce qui conclut le cycle, le reste des opérations du protocole se déroulant indépendemment des cycles.

Lors de la réception d'une proposition, un noeud répond, s'il est intéressé par une partie des données, par une requête concernant les données qui l'intéressent. Afin d'éviter un gachis de bande passante, un noeud n'effectue pas de requêtes à différents pairs concernant la même donnée et si les requêtes n'aboutissent pas après un certain délai, celles-ci sont ré-émises. À la réception de requêtes, les pairs essaient immédiatement de transférer les données.

Pour répartir la charge selon la bande passante des noeuds, chaque pair calcule régulièrement la moyenne de la bande passante des autres pairs, et adapte son $fanout$ en fonction de cette moyenne. Le $fanout$ moyen est de plus fixé par l'application, avec une valeur de l'ordre de $ln(n)$, où n est le nombre de pairs,

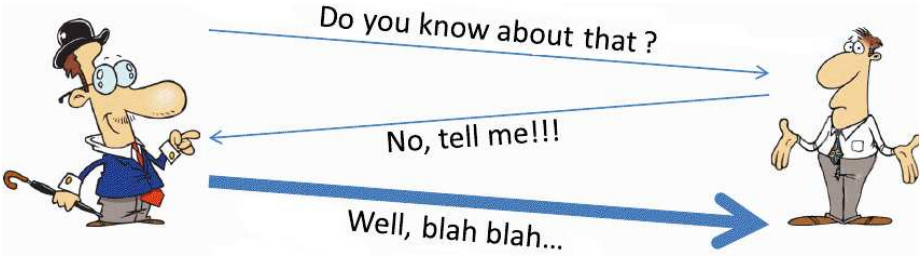


FIG. 5: Échange de données en trois phases : $Push/Pull/Push$

mais nous reviendrons plus en détail sur ces mécanismes dans la prochaine section. Les pairs possédant une importante bande passante proposent ainsi leurs données à un plus grand nombre d'autres pairs et reçoivent donc plus de requêtes. Ce comportement est important car il est très fréquent que, dans les domaines d'applications du *streaming* en pair à pair, les pairs disposent de capacité de bande passante hétérogènes. Il est important de noter que $HEAP$ ne tient compte que de la bande passante ascendante des pairs, ce qui est généralement suffisant car les bandes passantes sont souvent asymétriques, disposant d'une grosse capacité de téléchargement et d'une faible capacité d'envoi, ce qui fait qu'il est rare qu'un noeud voit sa bande passante descendante surchargée. Comme le montre le graphe 6 ci-dessous, la capacité de $HEAP$ à adapter le $fanout$ des noeuds en fonction de leur bande passante parvient à répartir la charge sur les noeuds en fonction de leur capacités de manière significative par rapport à un protocole de *Gossip* standard lorsque les bandes passantes sont hétérogènes.

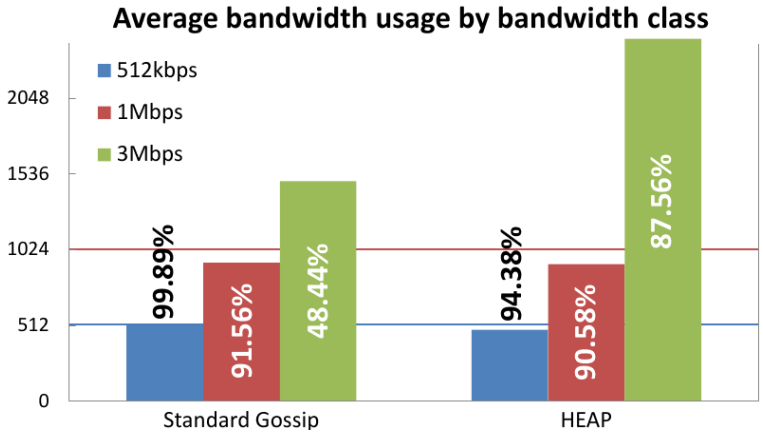


FIG. 6: Utilisation de la bande passante par $HEAP$ et un protocole de *Gossip* standard

3 Contributions

Cette section présente les différentes contribution apportées au protocole *HEAP*.

3.1 Probabilité d'acceptation

3.1.1 Répartition de la charge dans HEAP

Comme expliqué précédement, les protocoles de *Gossip* donnent des résultats assez médiocres lorsque les capacitées des noeuds sont hétérogènes, et en particulier si les besoins en bande passante sont importants. L'objectif de *HEAP* est donc de palier à ce problème en ajustant la charge de travail de chaque noeud en fonction de sa bande passante. Ainsi, si on a deux pairs A et B avec comme capacité de bande passante b_A et b_B tel que $b_A = n.b_B$, on souhaite que A travail n fois plus que B, afin que la charge soit correctement répartie. De cette manière, si les besoins en bande passante sont élevés, cela évite de surcharger B. La charge sur A doit donc être n fois la charge sur B, soit $C_A = n.C_B$, où $n = \frac{b_A}{b_B}$, on a donc :

$$C_A = C_B.\frac{b_A}{b_B} \tag{1}$$

La charge dont nous parlons ici correspond au nombre de données que chaque pair doit envoyer, mais dans un protocole de *Gossip* en trois phases, les pairs ne contrôlent pas entièrement le nombre de données qu'ils envoient, la seule variable qu'ils contrôlent étant leur $fanout$. La quantité de données que chaque pair doit transmettre à chaque cycle dépend du $fanout$, du nombre de données proposées et de la probabilité que chaque donnée soit accéptée. La quantité de données à proposer étant la même pour tous les pairs (ceux-ci reçoivent et proposent tous les mêmes données), la charge sur chaque noeud est donc dépendante uniquement du $fanout$ des noeuds et de la probabilité que leurs propositions soient acceptées. La contribution de chaque pair est donc, si cette probabilité est la même pour tout le monde, proportionelle à leur bande passante et on peut calculer le $fanout$ d'un noeud en remplaçant la charge par le $fanout$ dans l'équation précédente. On obtient :

$$f_A = f_B.\frac{b_A}{b_B} \tag{2}$$

Étendu à l'ensemble des noeuds, on a

$$f_A = \tilde{f}.\frac{b_A}{\tilde{b}} \tag{3}$$

où $\tilde{f}$ et $\tilde{b}$ sont respectivement le $fanout$ moyen et la bande passante moyenne sur l'ensemble des noeuds. Chaque noeud ajuste ainsi sa contribution en fonction des capacités de l'ensemble des noeuds.

Néanmoins, comme montré dans [4], le $fanout$ moyen des noeuds est important et ne doit pas prendre n'importe quelle valeur. En effet, des valeurs trop faibles diminuent la capacité de dissémination du protocole et fait que beaucoup de noeuds ne se voient jamais proposer certaines données. À l'inverse, des valeurs trop élevées peuvent entraîner une surcharge des noeuds, et de ce fait un

important temps de latence. Afin d'obtenir de bonnes performance, le $fanout$ moyen doit être suppérieur à $ln(n)$. Par exemple, le graphique suivant montre le résultat de plusieurs expériences menées sur un groupe de 230 noeuds. Pour chacune des expériences, un $fanout$ différent est utilisé, et les résultats montrent la proportion de noeuds recevant l'intégralité des données en fonction de la latence. On y voit que les meilleurs performances sont obtenues avec un $fanout$ entre 6 et 10, sachant que $ln(230) = 5.4$.

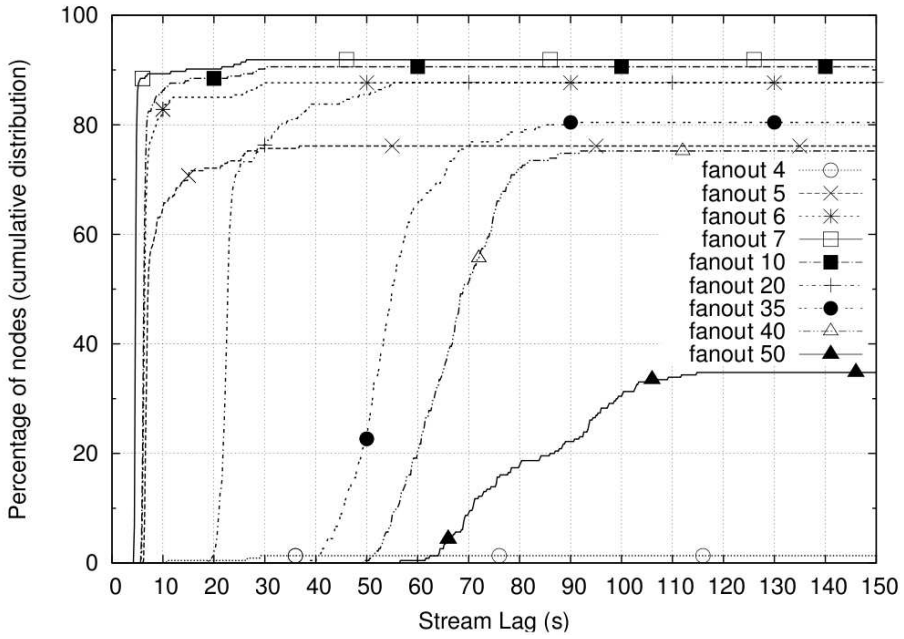


FIG. 7: Influence du $fanout$ sur la dissémination des données

Il est donc nécéssaire qu'en plus d'adapter le $fanout$ des noeuds les uns par rapport aux autres, le protocole assure également que le $fanout$ moyen soit à une valeur acceptable. Le $fanout$ moyen de chaque noeud est donc calculé en fonction sa bande passante et articulé autour d'un $fanout$ moyen $\bar{f}$ fixé par le protocole. L'équation obtenue est la suivante :

$$f_A = \bar{f}.\frac{b_A}{\tilde{b}} \tag{4}$$

où $\bar{f}$ est le $fanout$ moyen souhaité par le procotole, et calculé en fonction du nombre de pairs.

Ainsi, chaque pair envoie ses propositions à un nombre de pairs dépendant de sa bande passante, et tous sont donc supposés envoyer une quantité de donnés proportionelles à celle-ci. Mais comme dit précédemment, on ne fait varier que le nombre de propositions que l'on envoie, et pas directement le nombre de données. En effet, la quantité de données transmises est également influencée par la probabilité p que chaque proposition soit acceptée.
Tout fonctionne bien si, bien qu'elle varie selon l'âge des données, p est en moyenne identique pour tout les pairs. Mais plusieurs facteurs peuvent entraîner une variation de p parmi les pairs et ainsi déséquilibrer la répartition de la

charge. En effet, si les propositions de certains noeuds mettent plus de temps à parvenir aux autres pairs, celles-ci auront moins de chance d'être acceptées que celles provenant de pairs plus rapides. Les noeuds lents receveront donc moins de requêtes, et transmetteront ainsi un volume de données inférieur à ce qu'ils devraient au vu de leurs bandes passantes.
Plusieurs origines peuvent causer une telle lenteur, comme une fréquence de cycles de *Gossip* différents selon les pairs, le fait que la puissance de calcul d'un noeud soit très limitée (un décodeur TV intégrant le protocole par exemple), ou encore que celui-ci soit, du point de vue du réseau, éloigné des autres (par exemple, si la majorité des noeuds sont situés aux États-Unis, et quelques uns en Europe, les propositions de ces derniers mettront en moyenne plus de temps à atteindre leurs cibles que celles des noeuds situés aux États-Unis).

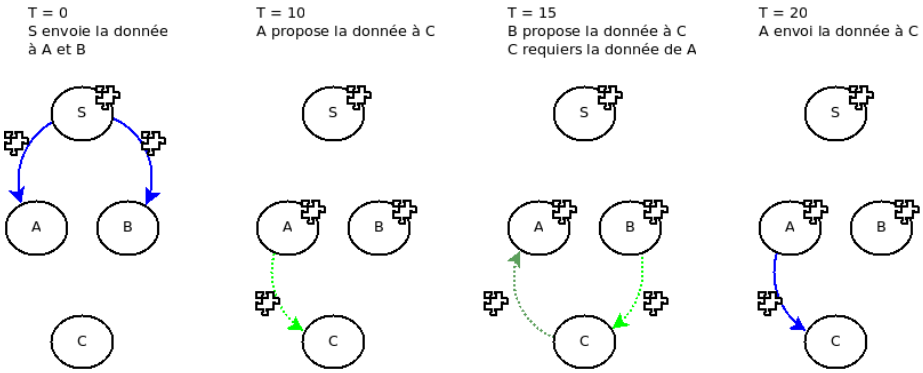


FIG. 8: Exemple d'exécution où un noeud B ne reçoit pas de requêtes car sa proposition a été envoyée trop tard

Adapter le $fanout$ uniquement en fonction de la bande passante n'est donc pas toujours suffisant si l'on souhaite bien répartir la charge et il est nécéssaire de prendre également en compte la probabilité d'être accepté par les noeuds.

3.1.2 Probabilité d'acceptation

On souhaite que le nombre moyen de pairs auxquels on envoie des données à chaque cycle soit uniquement dépendant de la bande passante, et pas de la probabilité que nos propositions soient acceptées. Ainsi, on doit ajouter de nouvelles variables dans l'équation de calcul du degré des noeuds pour obtenir :

$$f = \tilde{f}.\frac{b}{\bar{b}}.\frac{\tilde{p}}{p} \tag{5}$$

où p est la probabilité moyenne que les propositions d'un noeud soient acceptées, et $\tilde{p}$ la même probabilité mais sur l'ensemble des noeuds. En plus de mieux répartir la charge, l'utilisation de cette équation pour calculer le fanout peut également améliorer les performances brutes du protocole. En effet, lorsque le débit du contenu à diffuser est proche du débit moyen des noeuds, la bande passante de ceux-ci est proche de la saturation. Si en plus de cela certains noeuds participent moins à cause d'une probabilité d'acceptation plus faible, les noeuds

avec une probabilité élevée peuvent facilement se retrouver surchargés et donc incapables d'envoyer les données qu'on leur demande, induisant un retard dans la réception des données par les requéreurs voire la non réception de ces données.

3.1.3 Probabilité d'acceptation par hop

La probabilité d'acceptation des propositions d'un noeud ne dépend pas uniquement de la vitesse de celui-ci. En effet, la variable qui a le plus d'impact sur le fait qu'une proposition soit acceptée ou non est la proportion de pairs qui ont besoin de cette donnée. C'est pour cela qu'une donnée très récente aura toujours, quelque soit le pair, une grande probabilité d'être acceptée, alors qu'une ancienne aura toujours une probabilité faible de l'être. Ainsi, même si, en moyenne, certains noeuds ont deux fois moins de chance de voir leurs propositions acceptées que d'autres, lorsque la donnée proposée est récente et qu'elle n'est donc possédée que par un nombre très limité de pairs, l'écart de probabilité d'acceptation entre pairs lents et rapides sera très faible. Ne tenir compte que de la probabilité moyenne que nos propositions soient acceptées n'est donc pas suffisant, il faut également tenir compte de l'âge des données. Une technique simple, et relativement efficace, pour estimer l'age d'une donnée est le *hop* de celle-ci. Le hop d'un message ou d'une donnée est une valeur représentant le nombre d'intermédiaires passés entre l'envoi par la source et la récéption. Cette information n'est pas d'une précision extrême, car l'âge réel de la donnée dépend aussi beaucoup des intermédiaires par lesquels elle est passée, mais cela reste néanmoins une bonne indication.
L'idée est ici de calculer le $fanout$ à utiliser pour chaque donnée, selon le hop de celle-ci. A chaque cycle, on n'envoie donc plus une proposition à un ensemble de pairs, mais autant de propositions qu'il y a de *hop* différents dans les données à transmettre, et chaque donnée est proposée à un nombre de pairs dépendant du *hop* de celle-ci. L'équation suivante permet de calculer, en fonction de la probabilité d'acceptation par *hop* moyenne et personnelle de chaque noeud, le $fanout$ à utiliser pour chaque hop.

$$f_{hop} = \bar{f}.\frac{b}{\tilde{\bar{b}}}.\frac{\tilde{p}_{hop}}{p_{hop}} \tag{6}$$

L'utilisation de cette équation devrait apporter un meilleur contrôle de la répartition de la charge, et éviter qu'un noeud avec une faible probabilité d'acceptation moyenne se retrouve surchargé car il propose à un très grand nombre de noeuds des propositions de données récentes, ayant donc une probabilité très élevée d'être acceptée.

3.2 Estimation de bande passante

3.2.1 Intérêt

Dans tout protocole pair à pair cosommant une importante quantité de bande passante, il est important de connaitre la quantité de bande passante dont dispose chaque noeud. En effet, dans la très grande majorité des cas, les pairs ont des bandes passantes hétérogènes, et ne pas en tenir compte conduit inévitablement à des performances sous-optimales. De plus, lorsque la bande passante disponible est succeptible de varier pendant l'exécution (par exemple

lorsque plusieurs utilisateurs partagent une même connexion), la bande passante doit pouvoir être estimée dynamiquement afin de prendre en compte les changements de capacité. Les protocoles de *Gossip* ont, dans ce domaine, un avantage sur les protocoles structurés. En effet, là où les autres doivent réorganier leur structure afin de réagir aux variations de bande passante, dans les protocoles de *Gossip* seul les noeuds concernés par ces variations ont besoin de modifier leur comportement, par exemple en ajustant leur $fanout$.

3.2.2 Bande passante dans HEAP

Comme expliqué précédement, HEAP utilise une valeur représentant la quantité de bande passante des pairs pour adapter leur participation. Cette valeur peut être changée au cours de l'exécution du protocole, et HEAP adaptera le degré des noeuds en fonction des changements, mais aucune estimation n'est faite dynamiquement afin de vérifier qu'elle est correcte.

3.2.3 Solutions existantes

De nombreux travaux ont étés fait sur des méthodes d'estimation de bande passante, dont un certain nombre sont présentés dans [10], mais la majeure partie d'entre eux concerne des estimations de bande passante disponible sur un chemin entre deux points du réseau. Or, dans notre cas, les noeuds sont majoritairement des utilisateurs avec une connexion *Internet* personnelle, ayant donc une bande passante ascendante très limitée, il est donc très peu probable qu'un noeud parvienne à surcharger un chemin du réseau et la limite à la bande passante disponible viendra donc exclusivement du noeud. Malgré le fait que ces méthodes ne soient pas adaptées à notre cas, celles-ci peuvent fournir des pistes.
Dans *SLoPS* [5] par exemple, un noeud envoie des séries de paquets à un autre noeud, et celui-ci regarde le délai entre l'émission et l'arrivée de chaque paquet. Si le chemin est surchargé, ce délai augmentera car chaque donnée consécutive devra attendre un temps de plus en plus long dans la queue du lien le plus lent. Si le chemin n'est pas surchargé, le délai ne devrait pas subir de grande variation, et surtout ne devrait pas augmenter avec chaque nouveau paquet. Tant que le chemin n'est pas surchargé, le volume de données transféré est augmenté régulièrement jusqu'à trouver la limite du chemin. Même si cette technique est étudiée pour calculer la capacité maximale d'un chemin, elle fonctionnera également si la limite provient des capacités de l'émetteur, et indiquera donc les capacités d'envoi de celui-ci. Cette technique requiert néanmoins de longs envois de données vers un même pair, ce qui n'est pas très adapté à $HEAP$ où les échanges sont de très courte durée.
Dans [8], une méthode est présentée précisément pour évaluer la capacité d'envoi d'un noeud. Un noeud souhaitant estimer sa bande passante envoie des séquences de paquets à différents pairs, chacun transportant, en plus de données quelconques, le volume de données transféré jusqu'à son envoi. Les pairs recevant ces paquets peuvent en déduire de manière précise le débit de la source, en fonction des dates de récéption et de la différence de volume de données. Cette méthode est intéressante car elle pourrait bien s'adapter à $HEAP$, mais nous n'avons pas besoin d'une estimation aussi précise de la bande passante. En effet, celle-ci sert uniquement à répartir la charge entre les noeuds proportionellement

à leurs capacités, le volume d'utilisation quand à lui dépend du $fanout$ fixé par le protocole. Il n'est donc pas nécessaire d'obtenir une valeur en nombre de bytes par seconde, mais simplement une valeur qui puisse être comparée.
Les deux exemples précédents ont un point commun, elles augmentent le volume de données jusqu'à détecter qu'elles ont atteint une limite. De plus, lorsque l'objectif est de calculer la capacité d'un noeud plutôt que d'un chemin, il est préférable d'envoyer ses données à un nombre important de pairs afin de ne pas fausser les estimations à cause d'un éventuel chemin surchargé.

3.2.4 Solution proposée

Afin de ne pas entraîner un surcoût important (que ce soit en terme de puissance de calcul ou de nombre de messages), il est préférable de se reposer au maximum sur le traffic naturellement engendré par $HEAP$. Des données sont régulièrement envoyées à un ensemble très varié de pairs, celles-ci peuvent donc être utilisées afin de tester la bande passante des noeuds, sans être trop influencé par les zones surchargées du réseau du fait du grand nombre de pairs à qui on envoie des données. Lorsqu'un noeud souhaite tester sa bande passante, il demande, via des messages de propositions spéciaux que nous appelerons BDW_CHECK, à ce que les pairs à qui il transmet des données lui envoient des acquitements après réception de celles-ci.

Notre objectif étant d'avoir une valeur comparable, et non pas une estimation de débit précise, il n'est pas nécessaire de demander aux noeuds de calculer des valeurs comme la durée entre la réception de chaque donnée, ce qui n'est, de plus, pas toujours possible dans un système distribué.

Les acquitements transportent donc comme unique information le nombre de données requises et le nombre de données reçues, et un seul acquitement est utilisé par requête. Pour cela, après avoir reçu la proposition BDW_CHECK et répondu par une requête, un pair garde en mémoire la date et les données concernées par cette requête et envoie l'acquitement après un délai, exprimé en nombre de cycles. Du fait de l'asynchronisme des pairs, un nombre de cycles fixe et égal entre tous les pairs peut poser des problèmes, car la vitesse d'exécution d'un cycle peut fortement varier d'un pair à l'autre. Les noeuds surveillent donc le nombre de données reçues après que l'acquitement ait été envoyé, et augmente le délai si trop de données sont reçues après l'envoi de l'acquitement.

De son côté, le noeud testant sa bande passante envoie ses propositions BDW_CHECK pendant plusieurs cycles afin d'avoir un échantillon d'envoi de données représentatif (en effet, le nombre de données que l'on propose à chaque cycle dépendant du nombre de données reçues au cycle précédent, celui-ci peut être très variable et il est donc indispensable de faire les mesures sur plusieurs cycles). Une fois toutes ces propositions envoyées, le noeud attend quelques cycles (là aussi, le nombre de cycles est augmenté si trop d'acquitements arrivent en retard) afin de laisser aux autres pairs le temps de lui envoyer leurs acquitements, puis calcule le ratio données reçues / données envoyées. Le ratio ainsi calculé donne une estimation de la bande passante réelle du noeud, en fonction de l'ancienne. En effet, si l'on calcule que seules trois données sur quatre ont bien été reçues, et en supposant que les données non reçues sont due uniquement à la surcharge de la bande passante ascendante du noeud, on en déduit que la bande passante maximale du noeud n'est en réalité que 75% de ce que nous pensions. On peut donc calculer notre bande passante réelle :

$$bdw = bdw.\frac{received}{requested} \tag{7}$$

Evidemment, en utilisant le protocole UDP sur un réseau comme $Internet$, la supposition comme quoi la non réception de données est uniquement due à la tentative par le noeud source d'envoyer trop de données est fausse. En effet, une donnée peut avoir été envoyée avec succès mais être perdue en route vers sa cible, ou même ne pas avoir été reçue par la cible car la bande passante de celle-ci est surchargée. Ces pertes n'étant pas dues à la surcharge de la bande passante ascendante du noeud, elles ne doivent pas être prises en compte.
Afin de calculer une estimation du volume de pertes dûes à ces problèmes, un second test est effectué. Pour ce second test, on assigne au noeud la bande passante calculée précédement. Celle-ci est logiquement inférieure aux capacités réelles du noeud vu qu'elle prend en compte les pertes du réseau et le noeud ne devrait donc, dans ces conditions, pas avoir de difficultés à envoyer avec succès l'ensemble de ses données. Les pertes que nous observerons devraient donc être uniquement celles du réseau.

Cette méthode d'évaluation des pertes du réseau n'est pas parfaite. En effet, ces pertes sont dépendantes du volume de données transitant sur le réseau, ainsi en diminuant la quantité de données envoyées on diminue également les risques de pertes, ce qui peut biaiser l'estimation. Néanmoins, nous estimons que, compte tenu des capacités des bandes passantes ascendantes des connexions à $Internet$ standard (qui sont, dans un service de vidéo en streaming, supposées composer une large majorité des pairs), l'impact de cette diminution d'envoi de données est minime sur les pertes du réseau et peut donc être ignoré.

Nommons Q1 et Q2 le nombre de données requises pendant, respectivement, le premier et le second test, R1 et R2 le nombre de données effectivement reçues pendant, respectivement, le premier et le second test.

On a donc :
- $\frac{R1}{Q1}$: le ratio de réception standard
- $\frac{R2}{Q2}$: le ratio de succès du réseau

$\frac{R2}{Q2}$ représente donc la proportion de données acheminées avec succès par le réseau, et donc la proportion maximale de données pouvant être reçues par la cible. Le volume maximal de données pouvant être envoyées n'est donc pas $bdw.\frac{R1}{Q1}$, mais $bdw.\frac{\frac{R1}{Q1}}{\frac{R2}{Q2}}$ soit :

$$bdw.\frac{R1.Q2}{R2.Q1} \tag{8}$$

On obtient ainsi une estimation de l'éventuelle surcharge de la bande passante, qui nous permet de savoir de quel taux celle-ci doit être diminuée si elle est effectivement surchargée. Le mécanisme suivant est utilisé : Si le taux $\frac{R1Q2}{R2Q1}$ est supérieur à un certain seuil (de l'ordre de 0.9), on considère que le noeud est capable d'envoyer toutes ses données avec succès, et la bande passante est augmentée. Si le taux est inférieur à ce seuil, le noeud n'est capable d'envoyer des données qu'a hauteur de $\frac{R1Q2}{R2Q1}$ de la bande passante qui lui était atribuée au début du test, celle-ci est donc diminuée de ce taux.

Dans le cas où la bande passante n'est pas surchargée, une troisième phase permet de détecter rapidement la limite de celle-ci. Pour cela, un nouveau test est lancé pour lequel on augmente la bande passante de 10%. On obtient à la fin du test les valeurs $R3$ et $Q3$ comme pour les tests précédents, et on regarde si la perte reste la perte standard, ou si elle a augmenté en calculant le taux $\frac{R3.Q2}{R2.Q3}$. Si ce taux est toujours supérieur au seuil, la valeur de la bande passante est à nouveau augmentée, et la troisième phase est relancée. À chaque succès de la troisième phase, la valeur de la bande passante est augmentée de $10\%.succes$, où $succes$ représente le nombre de réussites consécutives de la troisième phase. Après quelques itérations, le noeud finit par atteindre la limite de sa bande passante, et met fin à la troisième phase.

La valeur que nous avons choisi d'utiliser pour représenter la bande passante est le $fanout$ maximal qu'un noeud peut supporter tout en étant capable de transmettre toutes ses données. Pendant les tests, le $fanout$ du noeud est fixé à la valeur en cours de tests, et à la fin de celui-ci le $fanout$ est fixé en fonction du $fanout$ maximal calculé, du $fanout$ maximal moyen des autres noeuds et du $fanout$ moyen fixé par le protocole.

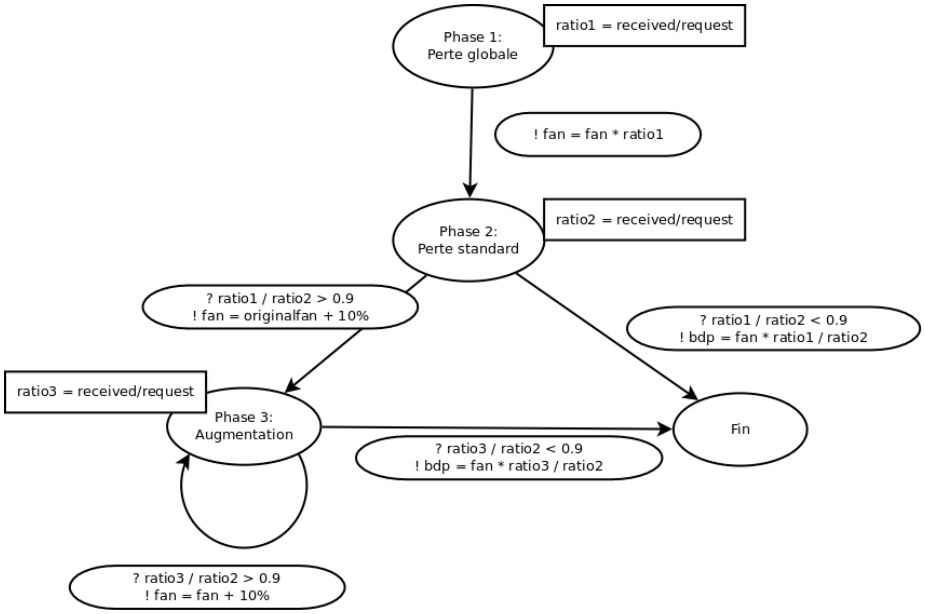


FIG. 9: Outil d'estimation de bande passante

Du fait du fonctionnement en trois phases des échanges de données (propositions, requêtes, transferts), les noeuds n'ont qu'un contrôle partiel sur la quantité de données qu'ils envoient. En effet, le volume de données à transférer dépend de trois paramètres :

- Le $fanout$
  - Nombre de pairs à qui on envoie des propositions
  - Controlé par le noeud

- Le taux d'acceptation des données
  - A l'échelle d'un cycle, dépend principalement de l'âge des données
  - Contrôle limité du noeud
- Le nombre de données
  - Nombre de données que l'on propose à chaque pair
  - Dépend uniquement des données reçues au cycle précédent
  - Aucun contrôle du noeud

Sur ces trois paramètres, un seul est entièrement contrôlé par les pairs, le taux d'acceptation et le nombre de données varient peu sur plusieurs cycles, mais peuvent varier énormément d'un cycle à l'autre. C'est la raison pour laquelle les tests sont faits sur plusieurs cycles. Néanmoins, les tests doivent être assez rapides, il n'est donc pas possible de les faire durer des centaines de cycles afin d'être sûr de disposer de bonnes valeurs moyennes. Ils sont donc effectués sur un nombre limité de cycles, avec un mécanisme vérifiant que les deux phases du test se sont déroulées dans des conditions à peu près similaires. Si aucun contrôle n'est fait, la première phase du test pourrait se dérouler avec des données majoritairement anciennes ou peu de données, et la seconde phase avec des données nombreuses et récentes. Dans ce cas, le noeud pourrait recevoir beaucoup plus de requêtes pendant la seconde phase malgré le fait que son $fanout$ ait été diminué, et le test serait ainsi faussé. Un mécanisme surveille donc le nombre de requêtes reçues pendant chaque phase du test, le nombre de requêtes traduisant bien l'impact des trois paramètres vus précédement, et compare les valeurs obtenues pendant les deux phases. Pour que le test soit valide, l'écart entre deux phases doit être de l'ordre de la modification appliquée au $fanout$. Si ce n'est pas le cas, le test est annulé et un nouveau est imédiatement relancé.

$$Request1.R1/Q1 \simeq Request2 \tag{9}$$

Un des points intéressants de cette méthode d'estimation de la capacité des pairs est qu'elle ne cherche pas à établir le débit maximal auquel ceux-ci peuvent envoyer leurs données, mais plutôt le $fanout$ maximal qu'ils sont en mesure de supporter. Ainsi, un noeud ayant une faible probabilité d'acceptation détectera un $fanout$ maximal plus important qu'un noeud disposant de la même bande passante mais ayant un taux d'acceptation plus élevé. Adapter le $fanout$ des noeuds en fonction des $fanouts$ ainsi calculés permet donc de répartir la charge entre les noeuds sans que le taux d'acceptation n'interfère.

3.3 Proximité des pairs

Dans un système pair à pair, les noeuds ne sont pas équivalents entre eux, et les connnexions qui les lient ne le sont pas non plus. En effet, certains noeuds sont liés par des connexions dont la qualité est supérieure à la moyenne, apportant une plus grande vitesse ainsi qu'une plus grande fiabilité de transmition. Cela peut être dû à plusieurs paramètres (surcharge localisée du réseau, bridage du débit par les fournisseurs, distance géographique) et peut avoir un impact assez significatif sur les performances d'un protocole. Adapter les modes de sélection des voisins afin de tenir compte de ces irrégularités est donc important, permettant à la fois d'améliorer les performances du protocole et de diminuer la charge sur le réseau (ou tout du moins de la concentrer sur des zones non surchargées).

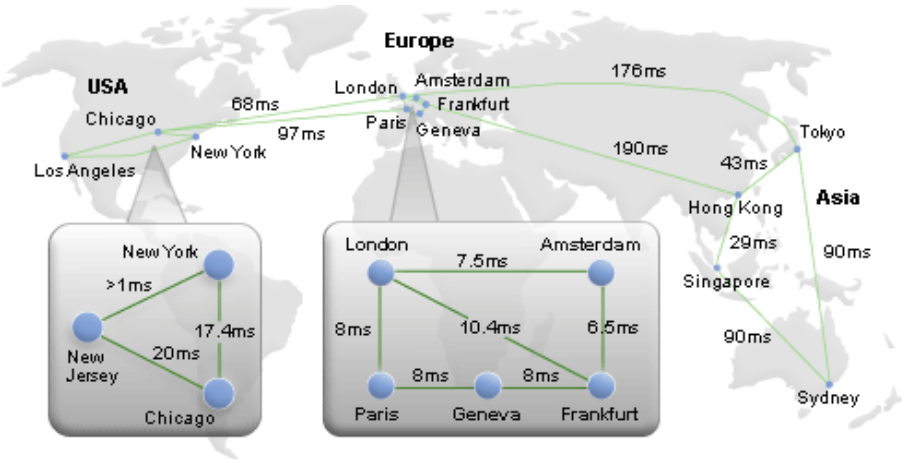


FIG. 10: Latences entres différents points du globe

Contrairement aux protocoles basés sur des *Mesh* ou des arbres, dans un protocole de *Gossip* il est difficile de construire une structure prenant en compte ces spécificités. En effet, la fiabilité de ce type de protocole vient en grande partie du fait que les noeuds changent continuellement de voisins, et restreindre trop fortement le nombre de voisins potentiels pour chaque pair pourrait rendre le protocole très sensible aux défaillances. Néanmoins, il reste possible dans un protocole où la structure est totalement dynamique de tenir compte des différentes qualités de connexions en biaisant, au moins partiellement, la méthode de sélection des voisins, et la dynamicité du protocole lui permet de s'adapter automatiquement lorsque des changements surviennent, sans avoir pour cela à reconstruire tout ou partie de la structure comme dans le cas des arbres.

3.3.1 Solutions existantes

Différentes études ont été menées sur des protocoles pair à pair d'échange ou de diffusion de données, comme dans [3], portant sur leur sensibilité aux irrégularités du réseau. Le résultat est que la majorité d'entre eux favorise les échanges au sein des *systèmes autonomes* (SA), qui sont les réseaux qui composent *Internet*, comme par exemple les réseaux des fournisseurs d'accès. Ces réseaux sont généralement connectés entre eux via un nombre limité de portes où le trafic est dense et donc la qualité de connexion plus faible qu'à l'intérieur du SA. De plus, certains fournisseurs, afin de limiter le trafic entre les SA (du fait de leur coûts), tentent de brider le débit à ces endroits, comme par exemple celui généré par le protocole bittorent [**?**]. Cette étude n'explique pas si cette préférence intra SA est dûe à une sélection des pairs dépendant de leur SA, ou si c'est une autre méthode de selection des meilleurs pairs qui aboutit à un découpage proche de celui des SA.

D'un autre point de vue, on peut chercher à estimer la qualité de la connexion entre différents noeuds, et essayer de faire communiquer entre eux les noeuds disposant des meilleures connexions. Comme expliqué précédement, de nombreuses

méthodes existent afin d'estimer la qualité des connexions, mais demandent généralement de longs échanges entre deux pairs.

CoolStreaming [7], un autre protocole de *streaming* en pair à pair, utilise une technique intéressante afin de favoriser les échanges entre pairs proches. Dans *CoolStreaming*, les données à transmettre sont divisées en *stream* et un pair souhaitant recevoir un *stream* s'abonne à un autre pair, qui lui enverra automatiquement les données de ce *stream* quand il les recevra. Chaque pair est donc abonné à différents autres pairs qui lui envoient ainsi les données dont il a besoin, et vérifie le taux de réception des données en provenance de chacun des pairs auquel il est abonné. Si l'un d'entre eux ne parvient pas à lui envoyer les données assez correctement, le pair se désabonne et trouve quelqu'un d'autre capable de lui fournir les données. En faisant ça, les pairs évitent de communiquer à la fois avec les noeuds surchargés, et avec ceux dont la connexion n'est pas suffisement bonne.

3.3.2 Solution proposée

La première solution envisagée était basée sur la supposition que, si les connexions sont de meilleurs qualité entre certains pairs, et donc plus rapide ainsi que plus fiables, cela devrait augmenter la probabilité que les propositions d'envoi de données de HEAP sur ces connexions soient accéptées.

Chaque noeud calcule donc la probabilité qu'il a d'être accépté par chacun de ses voisins, et échange prioritairement des donnés avec ceux qui ont le meilleur taux d'acceptation. Néanmoins, cette probabilité est très sensible à différents paramètres avec principalement l'âge des données proposées. Calculer cette probabilité sur un petit nombre de données ne sera donc pas représentatif, et avoir un échantillon de données suffisant serait probablement trop coûteux, d'autant plus que le nombre de pairs avec lesquels chaque noeud communique est très important. Cette solution n'a donc pas été retenue car jugée coûteuse et avec peu de chances de se montrer efficace.

Une seconde solution envisagée était d'utiliser le taux de réception de données en fonction des pairs, c'est à dire la probabilité qu'une donnée envouyée à un pair donné parvienne bien à destination. L'objectif principal est de détecter et d'éviter de communiquer via des zones surchargées du réseau, et de favoriser les échanges entre noeuds proches. Cette méthode a, contrairement à la précédente, d'importantes chances de se montrer efficace. En effet, si la qualité des connexions a un impact significatif sur la transmition des données, cela devrait se traduire en nombre de pertes de messages plus ou moins élevé selon les zones du réseau et donc être observable en surveillant le taux réussite des transferts de données.

Il est possible de calculer ce taux de réussite de deux manières :

Soit chaque pair calcule la proportion de données reçues par les différents autres pairs, soit les pairs répondent à la réception de données par un acquitement, permettant à l'envoyeur de calculer lui même le taux de réception des différents pairs.

Bien que la première solution soit plus économe, car elle ne nécessite pas d'acquitements, elle s'avère moins précise car les connnexions ne sont pas nécessairement symétriques. En effet, si deux données allant d'un point A à un point B suivront globalement le même chemin, une donnée allant de A vers B et une autre allant de B vers A peuvent suivre des chemins totalement différents,

en particulier lorsque A et B ne sont pas situés dans le même SA. La seconde solution est donc la meilleure, mais la première solution n'est pas inutile pour autant car elle permet de connaître les pairs les plus aptes à nous envoyer des données. Ainsi, lorsqu'un noeud reçoit des propositions pour les mêmes données venant de plusieurs pairs, il peut choisir le meilleur d'entre eux et ainsi augmenter la probabilité que le transfert réussisse.

Étant donné que ce sont les noeuds qui envoient des données qui ont le plus grand contrôle sur les chemins pris par les données, nous nous attarderons sur la seconde solution. Une fois les pairs triés selon le taux de transfert de données, différentes stratégies sont applicables afin d'exploiter la proximité de ces pairs.

Selection exclusive parmis les bons pairs

On peut choisir de ne proposer nos données qu'aux pairs que l'on considère comme bons afin d'éviter au maximum les pertes du réseau. Dans ce cas, il est important que cet ensemble soit de taille suffisante, car dans le cas contraire il y aurait un risque de séparer les pairs en sous-groupes ne communiquant pas entre eux. De plus, cette technique empêche, une fois la liste des bon pairs établie, d'en découvrir de nouveaux.

Sélection partielle parmis les meilleurs pairs

Il est également possible de ne sélectionner qu'une partie de nos voisins dans un sous-ensemble de bon pairs, les autres étant choisis aléatoirement parmi l'ensemble des pairs. Ainsi, il n'y a pas de risque de séparation des pairs en sous-groupes, et il est possible de découvrir de nouveaux bon voisins parmi ceux sélectionnés aléatoirement.

Exclusion des mauvais pairs

On peut également décider de simplement exclure les plus mauvais pairs afin d'éviter les zones particulièrement congestionnées du réseau. Cette méthode, bien qu'elle soit probablement la plus efficace pour éviter les zones surchargées du réseau, peut néanmoins poser problème. En effet, si un pair est situé au sein d'une zone congestionnée, il risque d'être exclus par tous les autres pairs et ainsi ne recevoir aucune donnée.

Il peut toutefois y avoir des cas, par exemple lorsque les noeuds ont tout juste la bande passante nécessaire pour transmettre les données, où l'on souhaite exclure ce type de pairs, qui seront de toute façon incapables de recevoir l'intégralité des données du fait de leurs mauvaises connexions, et ne seront pas non plus capables de fournir correctement leurs données aux autres noeuds.

Sélection probabiliste

Une dernière solution est de sélectionner, pour notre ensemble de voisins, chaque pair avec une probabilité proportionelle au taux de réussite de transferts vers celui-ci. Il n'est ainsi pas nécéssaire de séparer les pairs en sous-groupes selon la qualité des connexions, et on obtient une sélection directement dépendante de la qualité de celles-ci.

Ces différentes stratégies peuvent permettre d'adapter le comportement du protocole en fonction du contexte. Il serait ensageable d'utiliser un mécanisme étudiant l'état du réseau afin de sélectionner à tout moment la stratégie la plus adaptée.

4 Experimentations

Dans l'objectif de vérifier la validité des différentes solutions proposées précedement, nous avons implémenté celles-ci afin de les tester. Cette section présente la méthode de test ainsi que les résultats obtenus.

4.1 Méthodes

Les différentes contributions apportées au protocole ont été implémentées et testées afin de mesurer leur efficacité, puis les résultats ont été comparés à ceux du protocole original. Ces tests ont d'abord été éffectués en simulation sur des machines de la grille de calcul de l'irisa (*Igrida*), puis sur des machines de planetlab [2] dans le cadre d'une diffusion de vidéo.

Le débit de la vidéo utilisé pour les différentes simulations et expériences est de 551kbps découpée en paquets de 1316 bytes. Les paquets sont groupés par groupes de 101 paquets consécutifs auquels s'ajoutent 9 paquets d'encoding le tout constituant une fenêtre. L'interêt de ces paquets d'encoding est de permettre la reconstitution des fenêtres de données même si quelques pertes surviennent, au prix d'un volume de données à transmettre légèrement plus important, le débit total de données à transmettre étant ainsi porté à de 600kbps.

Lors de chaque expérience, les pairs se voient affecter une certaine capacité de bande passante conformément aux distributions ci-dessous, ainsi que d'autres distributions homogènes :

			Fraction of nodes		
Name	CSR	Average	2 Mbps	768 kbps	256 kbps
ref-691	1.15	691 kbps	0.1	0.5	0.4
ref-724	1.20	724 kbps	0.15	0.39	0.46
Name	CSR	Average	3 Mbps	1 Mbps	512 kbps
ms-691	1.15	691 kbps	0.05	0.1	0.85

FIG. 11: Distributions hétérogènes

Chacune de ces distributions de bande passantes sépare les pairs en trois groupes. Pour les distributions ref-691 et ref-724, la majorité des pairs disposent ainsi d'une bande passante supérieur au débit des données, alors que dans ms-691 85% des pairs ont une capacité d'émission inférieure à celui-ci.

Différentes métriques sont utilisées afin d'évaluer les résultats des simulations et des expériences. La première d'entre elles est la qualité du flux, qui représente la proportion de fenêtres de données reçues sans pertes. Pour qu'une fenêtre soit considérée comme reçue sans pertes, il faut que ses données puissent être

entièrement décodées, c'est à dire qu'au moins 101 paquets sur les 110 de la fenêtre doivent être reçus. La seconde métrique est le *streamlag*, qui correspond à la latence entre l'émission d'une donnée par la source et sa réception, puis sa lecture, par les pairs. Ces deux première métriques mises ensemble permettent de représenter la proportion de pairs recevant un certain pourcentage de données en fonction de la latence. Enfin, une troisième métrique utilisée est le pourcentage de messages n'ayant pas pu être transmis pour cause de surcharge de bande passante. Cette dernière permet d'avoir une estimation de la répartition de la charge entre les pairs. En effet, si la charge est correctement répartie, chaque pair est supposé envoyer des données proportionnellement à ses capacités, il sont donc tous sensés subir le même niveau de surcharge, et donc avoir un nombre d'abandons de message du même ordre de grandeur. A l'inverse, une très grande variation du nombre d'abandons de messages selon les noeuds est un indicateur d'une mauvaise répartition de la charge.

Simulations sur Igrida

La première partie des tests a été éffectuée en simulations sur des machines de la grille de l'irisa, celles-ci demandant beaucoup moins de temps pour le déploiement et le traitement des résultats que les machines de *Planetlab*. Les simulations portaient sur 80 noeuds, chacune étant répétée plusieurs fois afin d'éviter que des cas particuliers favorisent l'un ou l'autre des protocoles en cours de comparaison.

Le simulateur utilisé intègre différents mécanismes pour simuler les irrégularités des noeuds et du réseau. Un simulateur de communications permet de limiter la bande passante, simuler des pertes de messages ainsi que d'appliquer des délais variables sur les temps de transfert des messages, afin de simuler les échanges de données via *Internet*, et un simulateur de latence sur les noeuds permet de simuler lenteur de ceux-ci.

Planetlab

Planetlab est un réseau d'ordinateurs ayant pour objectif de permettre des expériences sur les systèmes distribués. Ces machines étant réparties sur 460 sites à travers le monde, elles permettent d'effectuer ces expériences en situation réelle, sans avoir à simuler de latence ni de perte sur les transferts de messages. Néanmoins, ces machines étant principalement situées sur des sites universitaires, celles-ci disposent de connexions à très haut débit, non représentatives des connexions standard des utilisateurs d'*Internet*. Pour les besoins des expériences, un mécanisme de contrôle de bande passante est donc mis en place sur chaque noeud.

Les machines de *Planetlab* étant régulièrement surchargées, celles-ci ne sont pas toujours en mesure de participer aux expériences dans de bonnes conditions, ce qui peut conduire à d'importantes variations dans les résultats. Les expériences que nous avons faites ayant pour but d'étudier l'impact de nos modifications, la comparaison entre les performances du protocole avec et sans les contributions est plus importante que les performances réelles. Il est donc uniquement nécessaire que nous parvenions à effectuer ces expériences comparatives dans les même conditions. Afin d'être sûr que celles-ci se déroulent dans les mêmes conditions, elles sont lancées simultanément sur les mêmes machines

auxquelles sont affectées les mêmes bandes passantes.

4.2 Taux d'acceptation

La méthode d'adaptation du *fanout* des noeuds en fonction du taux d'acceptation a été implémenté et testé, en comparaison avec le protocole *HEAP* standard, en utilisant différentes distributions de bandes passantes, dont celles décrites précédement, ainsi que des bandes passantes homogènes.

Afin de calculer de manière précise ce taux, celui-ci doit être calculée sur un nombre important d'évènements, car l'âge des données proposées a un impact très important sur la probabilité que celles-ci soient acceptées. Il faut donc prendre en compte suffisamment de données si l'on veut éviter que la probabilité moyenne soit grandement affectée par l'effet de quelques données très récentes ou très anciennes. Il n'est néanmoins pas non plus bon d'utiliser un trop grand nombre de données, car si le nombre de données pris en compte est trop important, les variations du taux d'acceptation seront détéctées moins rapidement, il est donc par exemple exclu de calculer cette moyenne sur l'intégralité des données.

Dans le cadre de ces expériences, chaque pair conserve la liste des 300 dernières données qu'il a proposé ainsi que le nombre de requêtes reçues pour chacune d'entre elles. La moyenne est faite en prenant en compte les 250 plus récents de ces évènements pour ne pas fausser le calcul avec des données dont les requêtes ne nous sont pas encore parvenues. Cette donnée est ensuite propagée aux autres pairs via le *PeerSampling*, et chaque noeud récupère ainsi les informations concernant l'ensemble de ses voisins.

Simulations

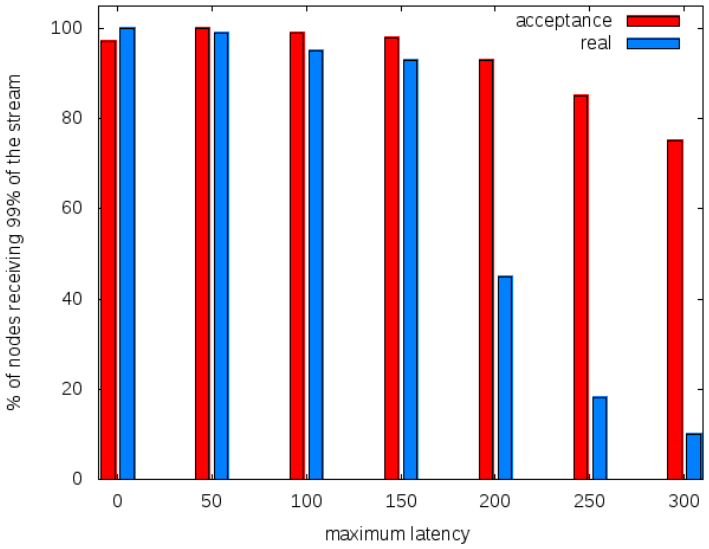
Des séries de simulations ont été effectuées afin de mettre en évidence l'effet de la latence sur les performances de *HEAP*. Pour cela, chaque pair se voit assigner en début d'expérience une latence comprise entre 0 et *lmax* millisecondes, *lmax* variant selon les expériences de 0 à 300. Ainsi, certains pairs subissant un délai supérieur aux autres, le délai entre le moment ou ils reçoivent une donnée et le moment ou celle-ci est reçue par les autres noeuds est en moyenne plus important que pour les pairs plus rapides. L'objectif de ces simulations est d'étudier l'impact de différentes valeurs de latences maximales afin de déterminer précisément les conséquences sur les performances du protocole. L'intérêt des simulations est qu'elles nous permettent de contrôler cette latence, ce qui n'est pas le cas sur *Planetlab*.

Pour les premières simulations, la distribution utilisée était une distribution homogène à 650kbps. L'intérêt principal d'utiliser une distribution homogène est que cela permet de mettre clairement en évidence les inégalités provoquées par la latence. En effet, chaque noeud disposant de la même bande passante, tous sont supposés fournir la même quantité de travail or comme nous allons le voir, ce n'est pas le cas lorsque les noeuds ont des latences différentes.

Le graphique 12 indique la proportion de pairs recevant 99% des donnéesau bout de 5000 millisecondes en fonction de la latence maximale appliquée au cours de l'expérience. Comme on peut le voir, le protocole *HEAP* standard est fortement affecté par l'augmentation de la latence, alors qu'avec le système

d'adaptation en fonction du taux d'acceptation celle-ci a un impact beaucoup plus faible.

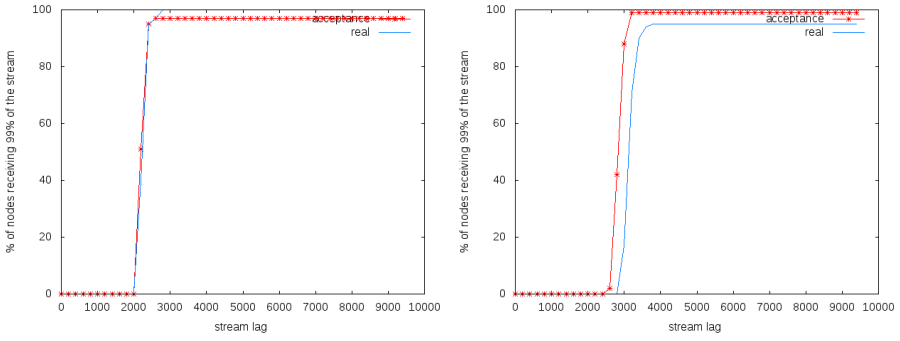
FIG. 12: Qualité de réception en fonction de la latence



Les graphiques des figures 13 et 14 indiquent la proportion cumulative de noeuds ayant reçu l'intégralité des données en fonction du *stream lag* sans latence, avec une latence maximale de 100ms, 200ms et 300ms.

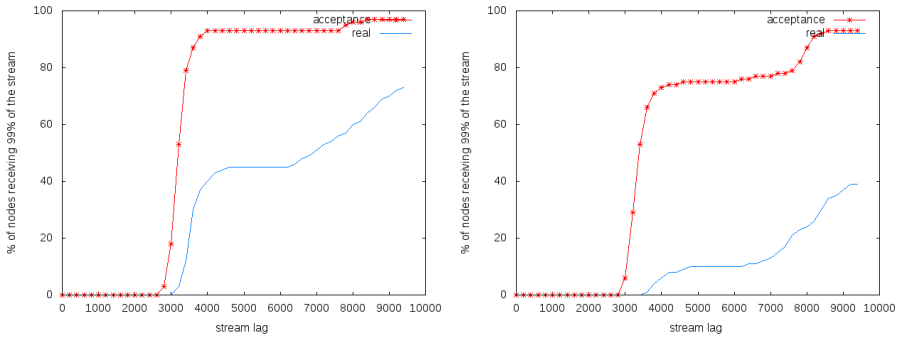
Comme on peut le voir, lorsque les pairs ne subissent aucune latence, les performances sont très bonnes et sont similaires entre *HEAP* standard et *HEAP*

FIG. 13: Homo-650 Stream Quality / Stream Lag



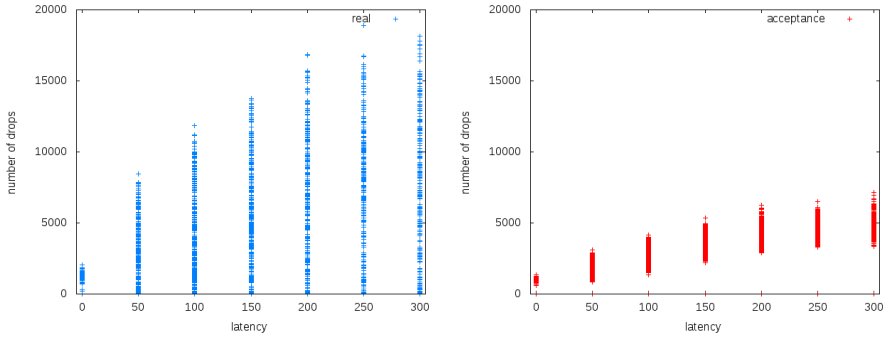
(a) 0ms (b) 100ms

FIG. 14: Homo-650 Stream Quality / Stream Lag



(a) 200ms (b) 300ms

FIG. 15: Répartition des messages jetés - homo-650

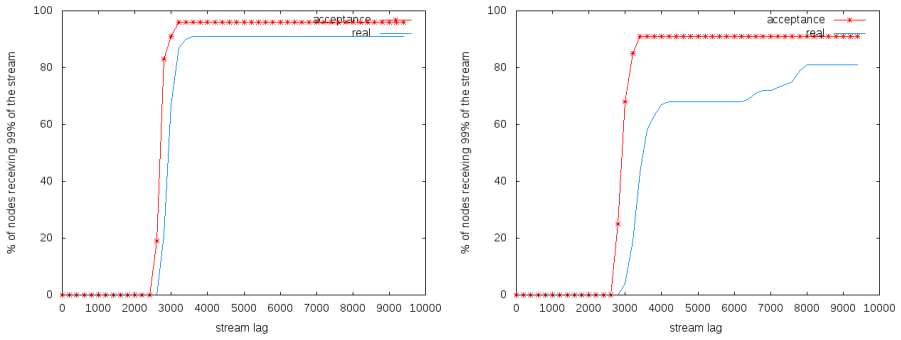


(a) *HEAP* (b) *Acceptance*

avec adaptation en fonction du taux d'acceptation. Par contre, dès que la latence augmente, les performances de *HEAP* se dégradent très fortement avec une diminution de la proportion de noeuds recevant au moins 99% du flux, et augmentation du *stream lag* pour ceux qui y parviennent. Ceci est logique, car les propositions des pairs ayant une faible latence arrivent beaucoup plus vite à leur cible que celles des pairs plus lents. Ces propositions ont donc une probabilité plus élevée d'être accéptée, ce qui fait que les pairs rapides reçoivent beaucoup plus de requêtes que les autres et ne sont pas en mesure de les satisfaire par manque de bande passante. Les pairs attendent donc longtemps avant de recevoir leurs données, ce qui explique l'augmentation du *stream lag*, et une part non négligeable ne sont jamais envoyées ce qui explique que, même avec un lag important un grand nombre de pairs ne parviennent pas à recevoir la totalité des données.

Le graphique 15 représente le nombre de messages jetés, c'est à dire n'ayant pas pu être envoyés, en fonction de la latence maximale appliquée aux noeuds. Chaque point de ce graphique représente un pair et le nombre de messages qu'il a dû jeter.

FIG. 16: Latence ms-691



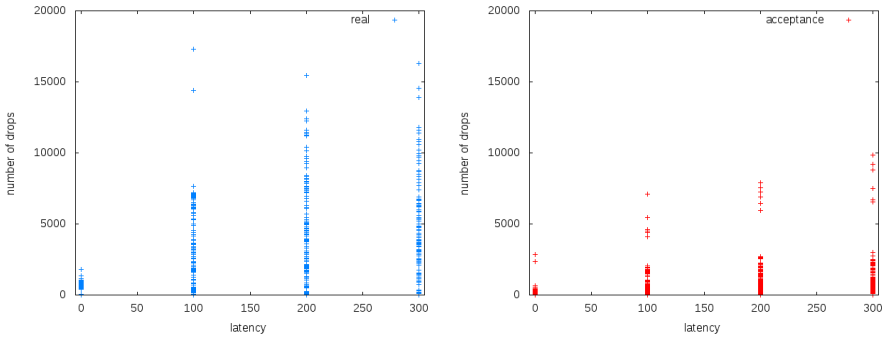
(a) $100ms$ (b) $200ms$

On remarque que lorsqu'il n'y a pas de latence, et donc que les pairs ont les mêmes chances de voir leurs propositions acceptées, le nombre de messages jetés varie très peu entre les pairs. Tous en jètent environ le même nombre, ce qui est normal car ils disposent tous des mêmes capacités. Par contre, lorsque la latence augmente, et donc que le déséquilibre entre les pairs augmentent, le nombre de messages jeté par pair devient très variable, certain n'en jetant plus aucun et d'autres, au contraire, en jètent beaucoup plus. Encore une fois, ceci est normal car les noeuds les plus lents ne recevant que peu de requêtes, ils ne sont que rarement surchargés et parviennent donc à expédier la quasi-totalité de leurs données, alors que les noeuds rapides doivent jeter de plus en plus de données avec l'augmentation de la latence. Dans le cas ou le $fanout$ des pairs est adapté en fonction de la probabilité que leurs propositions soient accéptées, on voit que, même si l'augmentation de la latence entraine un plus grand nombre d'abandons de messages, ceux-ci restent bien répartis entre les pairs, ce qui prouve que les pairs reçoivent bien des requêtes proportionellement à leurs capacités.

Les mêmes simulations ont été faites sur des distributions hétérogènes afin de voir si les mêmes problèmes se posent dans ces cas. Les figures 16 et 17 montrent la proportion de pairs recevant l'intégralité des données en fonction du *stream lag* pour des latences maximales de 100 et 300 millisecondes, ainsi que la répartition des jets de messages pour la distribution $ms-691$. Comme pour la distribution homogène, la latence dégrade assez nettement les performances de $HEAP$ et la encore le mécanisme d'adaptation au taux d'acceptation parvient a compenser la latence.

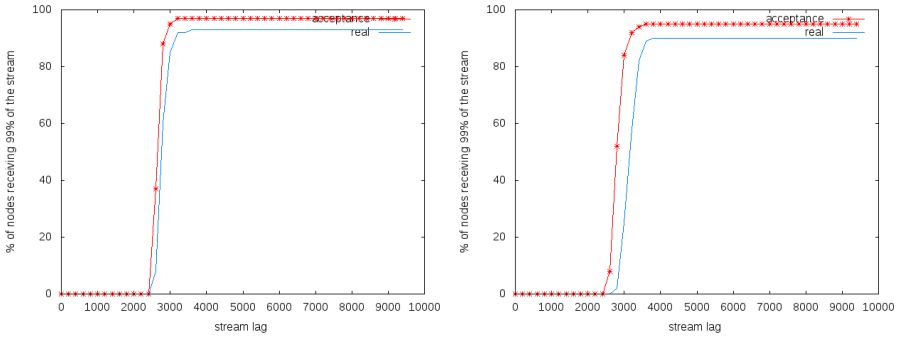
La dernière distribution utilisée pour les simulations est la distribution $ref-724$ qui, avec une moyenne de 724kbps, dispose de la plus grande capacité de transfert de données. Comme on peut le voir sur 18 et 19, cette distribution est également sensible à l'augmentation de la latence, mais de façon beaucoup moins significative que pour les distributions précédentes. Cela s'explique par le fait

FIG. 17: Répartition des messages jetés ms-691



(a) *HEAP* (b) *Acceptance*

FIG. 18: latency ref-724

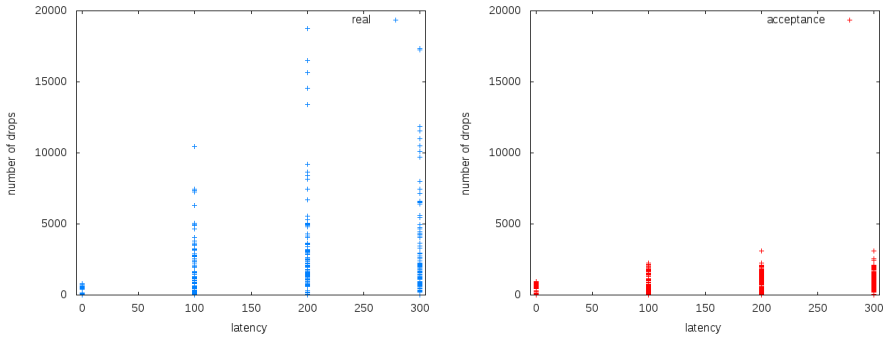


(a) $100ms$ (b) $300ms$

que les pairs disposant d'une bande passante moyenne plus élevée, ils peuvent supporter une légère surcharge de travail tout en étant capable de transférer l'intégralité de leurs données. Un déséquilibre, s'il n'est pas trop important, n'entraine donc pas de grosses pertes de performances.

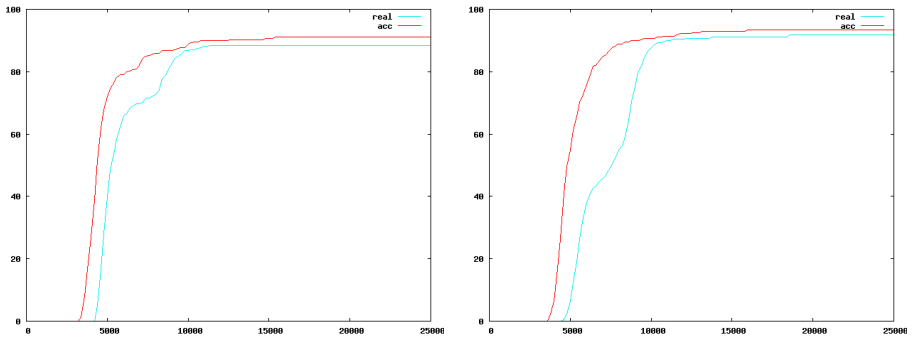
La latence a peut donc avoir un impact significatif sur les performances de *HEAP*, en entrainant une surcharge de certains noeuds qui se retrouvent dans l'impossibilité de répondre à toutes les requêtes qu'ils reçoivent pendant que d'autres ne participent presque pas aux échanges de données, tout particulièrement lorsque la bande passante moyenne des pairs est proche du débit émis par la source. Ces inégalités sont efficacement compensées par l'adaptation du $fanout$ en fonction du taux d'acceptation, ce qui se traduit à la fois par une meilleure réception des données par les pairs ainsi que par une répartition de la charge plus équitable.

FIG. 19: Répartition des drops ref-724



(a) *HEAP* (b) *Acceptance*

FIG. 20: Planetlab Stream Quality / Stream Lag

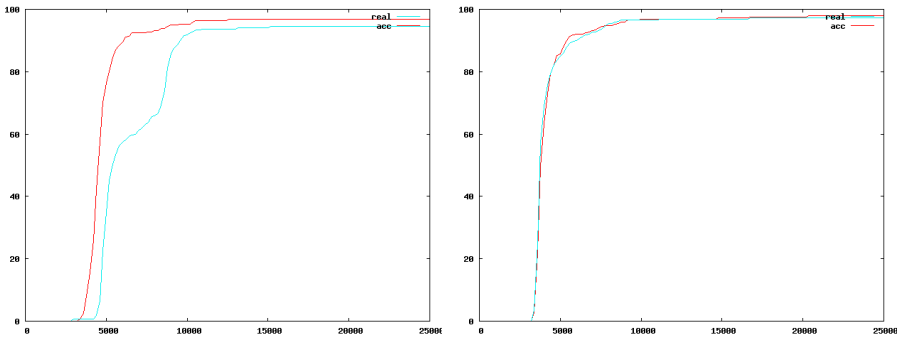


(a) homo-691 (b) ms-691

Planetlab

Des expériences similaires à celles simulées ont été faites sur *Planetlab*, afin de voir si la méthode d'adaptation en fonction du taux d'acceptation fonctionne également lorsqu'il est confronté à des pairs réellement lents, et pas uniquement lorsque cette lenteur est simulée. Les machines de *Planetlab* sont en effet fréquement surchargées à cause des nombreuses autres expériences s'exécutant au même moment, la latence ici n'est donc pas simulée, c'est la latence naturelle des différents pairs qui est "exploitée" afin de mettre en évident les problèmes de répartition de le charge. Du fait de la très grande variabilité de l'état des machines, deux expériences consécutives peuvent donner des résultats très différents. Afin de pouvoir mener des expériences comparatives sans que cette variabilité ne fausse les résultats, celles-ci sont lancées simultanément sur les mêmes machines. Toutes les expériences suivantes ont été effectuées sur un groupe de 300 machines réparties à travers le monde.

FIG. 21: Planetlab Stream Quality / Stream Lag



(a) ref-691 (b) ref-724

	ms-691	ref-691	ref-724
acceptation	6380	5311	3126
HEAP	11742	11344	5375

FIG. 22: Abandons de messages pour chaque des distributions

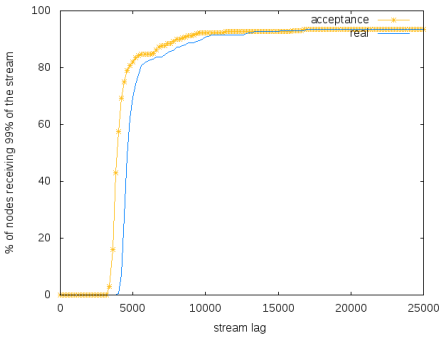
Comme on peut le voir sur 20 et 21, les résultats sont assez similaires à ceux obtenus avec les simulations. L'utilisation du taux d'acceptation apporte un gain important sur le temps nécéssaire aux pairs pour recevoir l'intégralité des données sur les distributions disposant de peu de marge de bande passante. On observe en effet une diminution de l'ordre de 15% à 20% du temps moyen que les pairs doivent attendre avant de recevoir les données émises par la source. Par contre, lorsque la marge de bande passante est plus important comme pour 22b, aucun gain n'est remarquable, ce qui s'explique comme dit précédement par le fait que les pairs sont en mesure de supporter un déséquilibre de la charge tout en parvenant à transmettre toutes leurs données.

Le nombre moyen de messages jetés pour les trois distributions hétérogènes est résumé dans le tableau 22. Ces resutats confirment les observations précédentes à savoir que le taux d'acceptation diminue significativement le nombre de messages perdus en répartissant la charge plus équitablement entre les pairs.

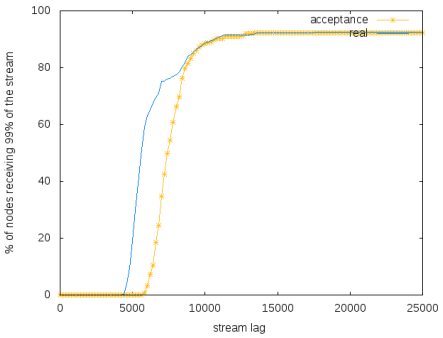
4.3 Estimation de bande passante

Afin de tester les performances du système d'estimation de bande passante, il à été exécuté en parallèle avec le protocole *HEAP* standard mais sans lui fournir d'informations sur la bande passante des pairs. L'execution à ensuite été lancée en laissant un délai au mécanisme d'adaptation de bande passante afin de lui permettre de calculer une première estimation avant que les résultats ne soient pris en compte.

FIG. 23: Stream Quality / Stream Lag

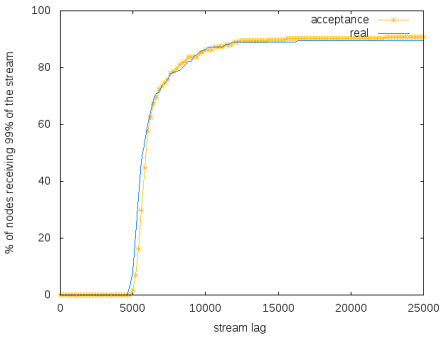


(a) ref-691

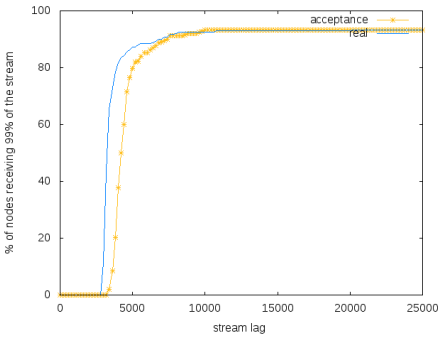


(b) ref-724

FIG. 24: Stream Quality / Stream Lag



(a) ref-691



(b) ref-724

Le système suit le comportement décrit plus tôt, la durée de chaque phase d'un test est fixée à 30 cycles, suivit d'une attente de 30 cycles durant laquelle les résultats sont collectés.

Les figures 23 et 24 montrent la qualité de réception en fonction du *stream lag* pour $HEAP$ et $HEAP$ avec estimation de la bande passante. On voit que les deux protocoles ont des performance assez similaires, voir même un gain de performances dans pour la distribution homogène. Ce gain de performance provient du fait que, en calculant le $fanout$ maximal que chaque noeud peut supporter, l'estimation de bande passante prend également en compte les inégalités d'acceptation. En effet, si deux pairs disposent de la même bande passante, mais que l'un des deux à un taux d'acceptation plus faible, il pourra supporter un $fanout$ plus important. L'équlibre entre ces deux pairs ne se fera donc pas en fonction de leur bande passante, mais en fonction du $fanout$ maximal qu'ils puissent supporter, et donc proportionnelement à leurs capacités réelles.

Par contre dans le cas des distributions hétérogènes $ms - 691$ et $ref - 724$, la latence est augmentée. Cela peut être dû au fait que l'estimation n'est pas

suffisement précise, en particulier lorsque les pairs ont des bandes passantes très faibles où très élevées entraînant de ce fait un léger désequilibre de la répartition de la charge. Ces deux distributions sont en effet celles dont la répartition est la plus extrème avec, pour $ms - 691$ 85% des pairs dont la bande passante est inférieure au flux de données émis par la source, et 15% dont la bande passante est très importante. De même pour $ref - 724$ où près de la moitié des pairs disposent de moins de la moitié du débit du flux de données.

Un autre facteur pouvant expliquer des performances sub-optimales est la perturbation induite par le test en lui même. En effet lors du test un pair commence par réduire sa bande passante, réduisant du même coup sa participation, puis l'augmente afin de trouver une limite supérieur qui, lorsqu'elle est atteinte, entraine la réception d'un trop grand nombre de requêtes auxquelles le noeud ne peut pas répondre. De plus, de nombreux paramètres entrent en jeu lors de cette estimation, entre autres la durée des tests et leur fréquence, et nous n'avons pas encore pû, par manque de temps, tester correctement ces différents paramètres afin de déterminer leur impact.

Toutefois comme on peut le voir, l'estimation de bande passante arrive tout de même à un niveau presque similaire à celui de $HEAP$ alors que celle-ci ne dispose d'aucunes informations au début de l'exécution. En particulier, elle améliore les performances dans le cas de la distribution homogène et atteint des performances équivalentes pour $ref - 691$. Les prochains travaux consisterons à étudier l'effet des différents paramètres afin d'améliorer la précision de l'estimation et de diminuer le coût de celle-ci.

4.4 Proximité des pairs

La première solution ayant été envisagée, qui consistait à sélectioner les pairs avec lesquels nous avons le plus haut taux d'acceptation, à été implémentée et testée, mais malgres différentes tentatives, elle n'a pas montré de signes d'amélioration du protocole. Ceci peut s'expliquer par le fait que l'effet sur le taux d'acceptation d'une meilleure connexion n'est probablement pas significatif en comparaison des autres facteurs entrant en jeu, comme le nombre et la vitesse des intermédiaires parcourus par la donnée avant de nous parvenir. Il serait donc nécessaire d'effectuer des mesures pendant un interval de temps très important afin d'avoir une chance d'effacer la variation tu taux d'acceptation dûe aux données proposées et ainsi savoir réellement les pairs avec qui nous avons un taux d'acceptation supérieur à la moyenne, or cela n'est pas compatible avec le fonctionnement de $HEAP$. Cette idée a donc été abandonnée au profit de la seconde solution, plus prometteuse à nos yeux.

Cette seconde solution, consistant à étudier le taux de réussite des transferts de données afin de détecter les voisins les plus proches, n'a pour le moment pas pu être testée par manque de temps, l'implémentation étant en cours. Celle-ci étant déjà bien avancée, nous esperons pouvoir tester l'efficacité de cette solution dans les prochaines semaines, et ainsi évaluer les performances des différentes stratégies proposées.

5 Travaux futurs

Comme expliqué précédemment, les différentes solutions proposées pour améliorer *HEAP* n'ont pas toutes pu être entièrement testées. En effet, le système d'adaptation de bande passante, bien qu'il fonctionne, n'a pas encore subi tous les tests nécéssaires. Il est tout d'abord important de mesurer l'impact de celui-ci sur le protocole, tant au niveau du volume de données supplémentaires transmises que sur les performances. Ce mécanisme est de plus contrôlé par de nombreux paramètres comme la durée des tests ou le seuil à partir duquel on considère que toutes les données ont été transmises et de nombreuses configurations sont donc possibles, certaines d'entre elles potentiellement plus performantes que d'autres dans certaines situations. Il serait également interessant d'améliorer ce mécanisme en lui permettant de changer dynamiquement sa configuration. Par exemple, lorsque les différents tests succéssifs indiquent de grandes variations de capacité, il est probablement intéressant de lancer des tests plus fréquement, même si ceux-ci sont moins précis, afin de s'adapter rapidement aux variations. À l'inverse, lorsque les capacités varient très peu, il peut être préferable de faire moins de tests mais de les faire durer plus longtemps dans le but d'obtenir une meilleure estimation.

Du côté de la topologie du réseau, l'implémentation n'est pas encore terminée et nous devons, en premier lieu, étudier la validité de la métrique utilisée, à savoir le taux de réception des données par les différents pairs. Si celle si est valide, il serait également intérêssant de mettre en corrélation ses résultats avec des métriques connues (distance géographique, latence, systèmes autonomes...). S'il s'avère que cette métrique est effectivement représentative de la topologie du réseau, il nous faudra étudier l'effet des différentes stratégies proposées et voir comment elles se comportent dans différents contextes.

6 Conclusion

L'objectif de ce stage était d'améliorer le protocole *HEAP* sur plusieurs points afin de le rendre plus efficace dans différents domaines.

La première contribution a été d'améliorer la méthode de répartition de la charge entre les pairs utilisée par *HEAP*. En effet comme nous l'avons vu, une bonne répartition en fonction des capacités des différents pairs est primordiale si l'on souhaite obtenir de bonnes performances. Or le mécanisme de répartition de *HEAP* pouvait être faussé lorsque certains noeuds réagissent plus vite que d'autres, induisant ainsi un biais dans la répartition de la charge et une surcharge de certains pairs. Nous avons donc amélioré de mécanisme en lui faisant prendre en compte ces inégalités afin de les compenser et obtenir une meilleure répartition de la charge, ce qui apporte un gain de performances significatif dans certaines configurations.

La seconde contribution fut d'intégrer à *HEAP* un système d'estimation de bande passante afin de lui permettre de calculer lui même les capacités des différents noeuds, lui permettant ainsi d'adapter la contribution de chacun dynamiquement. Il est en effet fréquent dans les systèmes pair à pair que les capacités des différents participants varient au cours du temps, et dans ce cas se fier à une valeur fixe représentant leur bande passante pour répartir la charge n'est pas satisfaisant. Il est donc important de disposer d'un mécanisme permettant de

calculer dynamiquement la capacité des noeuds et le choix a été fait d'exprimer cette capacité en terme de $fanout$ maximal. Ainsi, les noeuds ne cherchent pas à calculer leur débit en nombre de bits par seconde, mais calculent à la place le $fanout$ le plus élevé qu'ils puissent supporter tout en étant capable d'expédier l'intégralité des données qui leurs sont demandées. Cette manière de procéder a pour avantage de permettre la répartition de la charge non pas en fonction de la bande passante des noeuds, mais en fonction de leurs capacités réelles. En effet, cette valeur représentant le $fanout$ qui leur permet d'envoyer des données au maximum de leurs capacités, et le $fanout$ réellement utilisé étant une portion de cette valeur, les pairs participent à hauteur de leurs capacités réelles. Ainsi avec ce mécanisme, si par exemple la bande passante ou la réactivité d'un pair varie au cours de l'exécution, celui-ci adaptera automatiquement sa contribution à hauteur de ses capacités.

Enfin la dernière contribution a été de proposer une méthode pour rendre $HEAP$ réactif à la topologie du réseau. L'implémentation de ce dernier mécanisme étant toujours en cours, nous n'avons pu vérifier sa validité mais nous avons néanmoins définit la métrique à utiliser, à savoir le taux de réussite des transferts des messages à destination des différents pairs, ainsi que plusieurs stratégies afin d'exploiter cette métrique en fonction des différentes situations pouvant survenir.

Références

[1] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, and Atul Singh. Splitstream : high-bandwidth multicast in cooperative environments. In *SOSP '03 : Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 298–313, New York, NY, USA, 2003. ACM.

[2] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. PlanetLab : An Overlay Testbed for Broad-Coverage Services. *ACM SIGCOMM Computer Communication Review*, 33(3) :00–00, July 2003.

[3] Delia Ciullo, Maria Antonieta Garcia, Akos Horvath, Emilio Leonardi, Marco Mellia, Dario Rossi, Miklos Telek, and Paolo Veglia. Network awareness of p2p live streaming applications. *Parallel and Distributed Processing Symposium, International*, 0 :1–7, 2009.

[4] Davide Frey, Rachid Guerraoui, Anne-Marie Kermarrec, Maxime Monod, and Vivien Quéma. Stretching Gossip with Live Streaming. In *Proceedings of the 39th IFIP/IEEE International Conference on Dependable Systems and Networks (DSN)*, 2009.

[5] Manish Jain and Constantinos Dovrolis. End-to-end available bandwidth : measurement methodology, dynamics, and relation with tcp throughput. *IEEE/ACM Trans. Netw.*, 11(4) :537–549, 2003.

[6] Dejan Kostic, Adolfo Rodriguez, Jeannie Albrecht, and Amin Vahdat. Bullet : High bandwidth data dissemination using an overlay mesh. In *In SOSP*, 2003.

[7] B. Li, Y. Qu, Y. Keung, S. Xie, C. Lin, J. Liu, and X. Zhang. Inside the new coolstreaming : Principles, measurements and performance implications. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, 2008.

[8] Mugurel Ionut Andreica and Nicolae Tapus. Efficient Upload Bandwidth Estimation and Communication Resource Allocation Techniques. In *Recent Advances in Signals & Systems [Proceedings of the 9th WSEAS International Conference on Multimedia, Internet & Video Technologies (MIV)] (ISBN : 978-960-474-114-4 / ISSN : 1790-5109) Recent Advances in Signals & Systems [Proceedings of the 9th WSEAS International Conference on Multimedia, Internet & Video Technologies (MIV)] (ISBN : 978-960-474-114-4 / ISSN : 1790-5109)*, pages 186–191, Budapest Hongrie, 09 2009. Some of the communication optimization problems discussed in the paper were used as tasks in algorithmic contests or as teaching materials.

[9] Fabio Picconi and Laurent Massoulié. Is there a future for mesh-based live video streaming ? In *P2P '08 : Proceedings of the 2008 Eighth International Conference on Peer-to-Peer Computing*, pages 289–298, Washington, DC, USA, 2008. IEEE Computer Society.

[10] R. S. Prasad, M. Murray, C. Dovrolis, and K. Claffy. Bandwidth estimation : Metrics, measurement techniques, and tools. *IEEE Network*, 17 :27–35, 2003.

[11] Meng Zhang, Qian Zhang, Lifeng Sun, and Shiqiang Yang. Understanding the power of pull-based streaming protocol : Can we do better ? *IEEE Journal on Selected Areas in Communications*, 25(9) :1678–1694, 2007.