Le routage spécifique à la source et ses applications

Matthieu Boutier

Le routage spécifique à la source et ses applications

Matthieu Boutier

boutier@pps.univ-paris-diderot.fr, Univ Paris Diderot, PPS
CNRS, IRIF, UMR 8243, Sorbonne Paris Cité, F-75205 Paris, France

Le routage spécifique à la source est une forme de routage qui consiste à prendre en compte, lors des décisions de routage, la source du paquet en plus de sa destination. Le routage spécifique à la source permet donc d'avoir plusieurs routes vers une même destination et d'en choisir une en sélectionnant une adresse source. Dans cette communication, nous décrivons les techniques de routage spécifique à la source et nous montrons quelques-unes des applications qu'il rend possible, notamment celles qui sont motivées par les travaux du groupe Homenet de l'IETF.

Mots-clefs : Routage, Routage spécifique à la source, SADR, SAD Routing, Multi-chemin, Multipath

1 Introduction

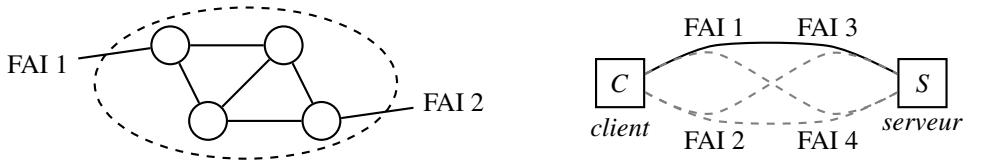
Un réseau est multihomé lorsqu'il reçoit sa connectivité à Internet de plusieurs fournisseurs d'accès (FAI). Généralement, cette pratique vise à augmenter la fiabilité du réseau, mais parfois aussi à augmenter ses performances ou diminuer les coûts. La figure 1*a* représente un réseau multihomé, où deux de ses routeurs sont connectés à deux FAI distincts.

Un réseau peut être multihomé en faisant partie du cœur de l'Internet : le réseau doit alors acquérir un préfixe d'adresses IP indépendant de tout fournisseur *(Provider Independant)*, et s'annoncer à tout le cœur par l'intermédiaire de ses FAI. Les hôtes du réseau reçoivent leur adresse IP dans ce préfixe, et le routage au sein du réseau se fait à l'aide d'un protocole de routage classique.

L'avantage de ce type de multihoming est de ne nécessiter aucun changement aux protocoles et applications existantes. La fiabilité est obtenue par la nature dynamique du protocole de routage qui, s'il détecte une panne d'un FAI, rétractera la route : les paquets passeront par un autre FAI. La diminution des coûts ou le gain de performances peuvent être obtenus en modifiant les paramètres du protocole de routage.

Malheureusement, cette technique requiert une entrée par réseau dans la table de routage globale de l'Internet ; c'est pourquoi elle n'est plus encouragée, voire prohibée (seuls des réseaux répondant à certains critères peuvent obtenir un préfixe indépendant). La plupart des réseaux reçoivent leur préfixe d'adresses IP de leur FAI : beaucoup de réseaux sont ainsi agrégés dans un seul préfixe d'adresses du FAI, le seul à apparaître dans l'Internet global.

Un réseau connecté à plusieurs FAI sans avoir de préfixe indépendant reçoit donc de chaque FAI un préfixe d'adresses dépendant du FAI. Dans ce type de multihoming, chaque hôte du réseau reçoit plusieurs adresses IP, une par fournisseur, et doit alors choisir quelle adresse utiliser lors de l'émission de paquets. Un



(a) Un réseau multihomé *(b) Chemins entre deux hôtes multihomés*

FIGURE 1: Configurations des réseaux multihomés.

paquet à destination de l'Internet doit ensuite être routé par le FAI qui est à l'origine de son adresse source : les autres FAI doivent en principe détruire le paquet.

On remarque alors que le routage offre aux couches supérieures des hôtes la possibilité de choisir entre plusieurs chemins, grâce à la sélection de la paire d'adresses source et destination. La figure 1*b* illustre ce propos, où un client a deux adresses, une fournie par FAI 1, l'autre par FAI 2, et de même le serveur a deux adresses fournies par FAI 3 et FAI 4. Le client, lorsqu'il envoie un paquet, peut choisir par quel FAI le paquet doit sortir de son réseau en sélectionnant son adresse source, et par quel FAI le paquet doit arriver dans le réseau du serveur en sélectionnant son adresse destination.

Cette communication se place dans le contexte du multihoming à adresses dépendantes des FAI. Nous présentons deux contributions : le routage sensible à la source, et une application multichemin, mpmosh.

2 Routage spécifique à la source

Le routage spécifique à la source [BC15] est une modeste extension du routage *next-hop* où les paquets sont routés en fonction de leur adresse source en plus de leur adresse destination. Le routage spécifique à la source résout le problème du routage dans les réseaux multihomés en routant les paquets à destination de l'Internet vers le FAI fournisseur de leur adresse source. Le groupe de travail *Homenet* de l'IETF, chargé de normaliser les protocoles des réseaux domestiques, a choisi le routage spécifique à la source comme paradigme de routage.

En routage classique, les tables de routage associent un *next-hop* à un préfixe (d'adresses destination). Il arrive fréquemment que des préfixes se chevauchent : plusieurs entrées de la table de routage sont alors candidates pour router un même paquet. En présence d'une telle ambiguïté, l'entrée ayant le préfixe le plus spécifique est retenue. Cela fonctionne car les préfixes ont une structure particulière, en arbre, qui implique que l'ensemble des préfixes contenant une même adresse est totalement ordonné par l'inclusion. La notion de spécificité coïncide avec celle d'inclusion.

En routage spécifique à la source, les tables de routage associent un *next-hop* à une paire de préfixes destination et source. La relation d'inclusion (ou de spécificité) ne suffit plus à ordonner les entrées : deux entrées peuvent être ni incluses l'une dans l'autre, ni disjointes. C'est le cas si les préfixes destination et source de la première sont respectivement plus et moins spécifiques que ceux de la seconde (ou inversement).

Il est primordial que le comportement de chaque routeur du réseau face aux ambiguïtés soit exactement le même, sans quoi des boucles de routage persistantes peuvent apparaître. Les protocoles de routage doivent donc choisir un comportement. Il y a consensus au sein de la communauté pour que les paquets soient routés en suivant l'ordre lexicographique sur les paires (destination, source) : les entrées de préfixe destination plus spécifique sont préférées, et en cas d'égalité celles de préfixe source plus spécifique.

Les implémentations des protocoles de routage calculent des entrées à installer dans la table de routage des couches basses du système (FIB) des routeurs. La cohérence de ces entrées dépend de l'ordre spécifié par le protocole. Toutes les FIB implémentent nativement le routage *next-hop* avec l'ordre de spécificité, mais toutes ne supportent pas nativement le routage spécifique à la source. Il est toutefois souvent possible d'utiliser des règles d'ingénierie de trafic, qui permettent de sélectionner une table de routage en fonction d'une adresse source : le routage se fait alors avec l'ordre inverse de celui souhaîté.

Dans une publication précédente [BC15], nous décrivons une technique forçant les couches basses des systèmes à suivre l'ordre choisi par le protocole en ajoutant des routes plus spécifiques.

3 Multipath à la couche application

Nous l'avons vu, le choix d'un chemin dans des réseaux multihomés s'effectue par le choix d'une paire d'adresses source et destination. L'algorithme par défaut de sélection d'adresse [TDMC12] calcule de manière déterministe une paire d'adresses source et destination en fonction de paramètres locaux et statiques (ensemble des adresses source et destination). Nous explorons la possibilité de sélection dynamique de chemins à la couche application à l'aide de mesures bout en bout.

Travaux connexes Shim6 [NB09] est un protocole situé entre la couche réseau et la couche transport, destiné à réécrire les adresses source et destination des paquets lors de pertes de connexions. Cette opération

est complètement transparente pour les couches supérieures : les connexions TCP ne sont ainsi pas perdues s'il reste un chemin valide. Shim6 détecte la perte d'une connexion en se basant sur l'absence de trafic entrant. Tous les chemins possibles sont alors testés à l'aide de sondes, et le "meilleur" chemin est retenu pour devenir le nouveau chemin emprunté. Shim6 résout ainsi le problème de la fiabilité dans les réseaux multihomés.

MPTCP [RPB+12] est une extension compatible de TCP qui établit autant de sous-flots que de chemins disponibles. Il utilise ses sous-flots pour augmenter la fiabilité du réseau et répartir la charge, tout en contrôlant la congestion. Un gros avantage de MPTCP est d'être transparent à l'application, qui utilise des connexions TCP classiques : il suffit que le client et le serveur implémentent MPTCP pour que toutes les connexions TCP soient automatiquement multi-chemin.

SCTP [Ste07] est un autre protocole multi-chemin de couche transport : il n'utilise qu'un seul chemin simultanément, mais envoie des sondes périodiquement pour évaluer l'état des autres chemins. Lorsque le chemin primaire (utilisé pour les données) tombe, un autre chemin est sélectionné, assurant la fiabilité de la communication.

Couche application : mpmosh Mosh [WB12] est un shell sécurisé à distance résistant aux changements d'adresses du client, adapté aux réseaux à fort taux de pertes, fluide en cas de forte latence. Mosh utilise UDP pour son côté non connecté, non fiable et non ordonné. Ainsi, il choisit l'adresse (destination) du serveur, mais laisse le système décider de l'adresse source : si la machine n'a qu'une adresse à fois, les paquets émis prendront cette adresse, même si elle change. Mosh résiste aux pertes (et réordonnancements) en incluant dans chaque message toute l'information non acquitée : si un paquet est perdu, son contenu sera retrouvé dans le paquet suivant. Enfin, mosh dispose d'un mécanisme de prédiction, qui donne l'illusion à l'utilisateur que le serveur répond vite ; dans beaucoup de cas bien sûr, cette optimisation est impossible.

Toutefois, mosh est conçu pour des hôtes n'ayant qu'une adresse IP. Il ne survit pas à un changement d'adresse du serveur, et ne différencie pas les chemins. Il peut ainsi utiliser le moins bon chemin, voire utiliser un chemin cassé et perdre ainsi sa connectivité. Il n'est pas non plus capable de passer d'IPv4 à IPv6. Nous avons modifié mosh pour en faire une version multichemin : mpmosh. Notre version corrige ces inconvénients propres aux réseaux multihomés, et peut dans certains cas optimiser les performances de mosh en utilisant plusieurs chemins simultanément.

Découverte et évaluation des chemins Un chemin étant une paire d'adresses source et destination, le client mpmosh récupère ses adresses locales avec l'API standard du système, et demande au serveur ses adresses. Il fait ensuite le produit cartésien des adresses locales et distances, et en déduit les chemins. Seules les associations évidemment impossibles, comme par exemple une adresse source IPv4, et une adresse destination IPv6, sont écartées : nous préférons garder des chemins infaisables que manquer un chemin.

Nous évaluons les chemins à l'aide de sondes, paquets particulièrement légers ne transportant aucune donnée : une trame IPv6 contenant une sonde pèse de l'ordre de 90 octets. Des sondes sont envoyées sur chaque chemin à un intervalle dépendant de la qualité du chemin. Ainsi, un chemin inactif ou impossible ne sera testé que très rarement. Dans notre implémentation, nous envoyons une sonde toutes les 10 secondes dans ces cas, ce qui correspond à un trafic moyen d'au plus 9 octets par seconde et par chemin inactif.

Latence Mosh est une application légère et interactive : nous cherchons à optimiser la latence. Mosh implémente déjà l'algorithme de Mills pour le calcul du RTT (temps d'un aller-retour), que nous reprenons pour les sondes de mpmosh. Nous évaluons le RTT de chaque chemin indépendamment, et supposons donc que les chemins sont symétriques. Lorsqu'un chemin tombe, les paquets ne peuvent plus l'emprunter : le RTT n'est donc plus mis à jour. Pour pallier ce problème, nous estimons le temps d'inactivité du chemin, et utilisons à la place du RTT la valeur du RTT additionnée à celle du temps d'inactivité.

Nous détectons la perte d'un chemin en utilisant le calcul classique du RTO, à la manière de TCP. Le RTO est le temps au bout duquel une réponse aurait dû arriver, et est de l'ordre d'un RTT. Mosh utilisant des acquitements retardés, le RTO est adapté en conséquence. La détection d'une perte de connectivité sur un chemin est donc de l'ordre d'un RTO et de l'intervalle toléré pour les acquitement retardés.

À chaque envoi de données, le chemin de plus faible RTT est utilisé, sans contrôle de stabilité. Si deux chemins ont des RTT semblables, il est possible que des oscillations apparaissent sur le choix du chemin,

ce qui n'est pas gênant dans le cas de mosh, puisque chaque paquet est indépendant, et qu'aucune retransmission ne sera demandée.

Convergence accrue Lorsque le meilleur chemin devient inactif, mpmosh peut mettre un certain temps avant de reconverger vers un autre chemin. Nous supprimons ce temps d'attente en dupliquant le prochain paquet envoyé dès qu'un chemin est détecté inactif.

Notons qu'un chemin peut être détecté inactif sur une simple perte de paquet ; c'est pourquoi la valeur du RTT associée à un chemin détecté inactif ne change pas. Lorsqu'un chemin est à nouveau reconnu actif, le temps d'inactivité est simplement mis à 0.

Pertes de paquets Mosh est naturellement résistant aux pertes de paquets, mais de forts taux de pertes peuvent nuire à l'expérience de l'utilisateur. Nous avons exploré l'idée de dupliquer les paquets sur différents chemins, pour limiter l'impact des pertes relatives à chaque chemin. Tout d'abord, nous évaluons le taux de pertes des chemins à l'aide d'une fenêtre glissante. Puis, au moment d'envoyer un paquet, les chemins disponibles sont triés par latence croissante. Le paquet est alors dupliqué sur chacun de ces chemins, jusqu'à ce que le produit de leurs taux de pertes soit proche de zéro : le paquet ne sera probablement pas perdu.

Résultats Nos expériences confirment que les optimisations de convergence accrue et de limitation de pertes de paquets fonctionnent correctement. La détection d'une perte de connectivité d'un chemin et la convergence vers un autre chemin fonctionnel se fait en un temps de l'ordre d'un RTT. La duplication des paquets peut améliorer sensiblement l'expérience utilisateur. Prenons comme exemple un parcours de fichier avec la commande `less` dans un réseau avec 4 chemins à 70% de pertes. Nous avons mesuré un temps de réponse moyen de 576 ms pour mpmosh contre 1295 ms pour mosh. Le trafic moyen pour cette même expérience était de 1,4 Ko/s pour mpmosh contre 0.6 Ko/s pour mosh.

4 Conclusion

Le routage spécifique à la source est une solution au routage dynamique dans les réseaux multihomés avec adresses dépendantes des FAI, et offre la possibilité aux couches supérieures de choisir entre différents chemins. Utiliser différents chemins à la couche application permet d'optimiser les performances de l'application.

MPmosh est un cas très particulier d'application multi-chemin que nous avons développé, où la latence est optimisée. Sa convergence est immédiate après détection de la perte de connexion, plusieurs chemins peuvent être utilisés pour réduire le taux de perte moyen. Davantage d'expérimentations sont nécessaires pour bien comprendre les besoins des applications, et de voir si la couche transport est plus adaptée ou non qu'une librairie à la couche application.

Références

[BC15] M. Boutier and J. Chroboczek. Source-specific routing. In *IFIP Networking Conference (IFIP Networking), 2015*, pages 1–9, May 2015.

[NB09] E. Nordmark and M. Bagnulo. Shim6 : Level 3 Multihoming Shim Protocol for IPv6. RFC 5533 (Proposed Standard), June 2009.

[RPB+12] Costin Raiciu, Christoph Paasch, Sébastien Barré, Alan Ford, Michio Honda, Fabien Duchene, Olivier Bonaventure, and Mark Handley. How hard can it be ? Designing and implementing a deployable multipath TCP. In *USENIX Symposium of Networked Systems Design and Implementation (NSDI'12), San Jose (CA)*, 2012.

[Ste07] R. Stewart. Stream Control Transmission Protocol. RFC 4960 (Proposed Standard), September 2007. Updated by RFCs 6096, 6335.

[TDMC12] D. Thaler, R. Draves, A. Matsumoto, and T. Chown. Default Address Selection for Internet Protocol Version 6 (IPv6). RFC 6724 (Proposed Standard), September 2012.

[WB12] Keith Winstein and Hari Balakrishnan. Mosh : An interactive remote shell for mobile clients. In *USENIX Annual Technical Conference*, pages 177–182, 2012.