RFC 6182 : Architectural Guidelines for Multipath TCP Development

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 23 mars 2011

Date de publication du RFC : Mars 2011

https://www.bortzmeyer.org/6182.html

Plusieurs efforts ont déjà été déployés pour avoir un protocole de transport qui puisse utiliser plusieurs chemins simultanés, pour le cas où deux machines disposent de plusieurs moyens de se joindre. Le projet « "*Multipath TCP*" » dont voici le premier RFC se distingue par le fait que son trafic apparaîtra comme du TCP habituel, aussi bien aux applications qu'aux équipements intermédiaires situés sur le réseau.

L'idée de base de tous ces efforts est : il arrive qu'entre deux machines A et B, il existe plusieurs chemins possibles, par exemple si A et B sont tous les deux connectés à au moins deux FAI (cf. section 1.3). Utiliser simultanément ces deux chemins permettrait d'augmenter la capacité disponible (et donc le débit effectif) et d'augmenter la résilience (en cas de panne d'un chemin pendant la connexion, on utiliserait automatiquement l'autre). TCP ne permet pas cela, puisqu'il est lié à des adresses IP source et destination qui ne changent pas. Des protocoles de couche 4 comme SCTP (RFC 4960 [1]) ont déjà exploré cette voie. Mais ils souffrent d'un double problème : il faut parfois adapter les applications, qui s'attendent à la sémantique de TCP, et il faut être sûr que le chemin complet soit de l'IP propre. Or, cela est rare dans l'Internet aujourd'hui, que des incompétents ont truffé de boîtiers intermédiaires non transparents et qui rendent le déploiement de tout nouveau protocole très difficile.

Pourtant, pouvoir mettre en commun la capacité de plusieurs chemins est tellement tentant que d'autres projets ont vu le jour. C'est ainsi que le groupe de travail mptcp <http://tools.ietf.org/wg/mptcp> de l'IETF a été créé pour mettre au point une architecture, "*Multipath TCP*" et un protocole mettant en œuvre cette architecture, MPTCP (le RFC normalisant ce protocole étant encore en cours d'élaboration).

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc4960.txt

Comme son nom l'indique, l'idée de base est de ne pas trop s'éloigner du TCP actuel, afin d'assurer une compatibilité à la fois avec les applications actuelles et avec le réseau actuel, les deux points sur lesquels SCTP a largement achoppé. SCTP est donc plus propre, plus conforme à l'architecture originale de l'Internet mais "*Multipath TCP*" est peut-être plus réaliste. (Notre RFC 6182 ne cite pas SHIM6 - RFC 5533 - mais celui-ci possède pas mal de points communs avec SCTP, notamment un très faible déploiement dans l'Internet actuel.)

La section 2 précise les buts de l'architecture "*Multipath TCP*". Du point de vue des fonctions, on voudrait :

— Augmenter le débit, par l'utilisation simultanée de davantage de ressources réseaux. Plus formellement, l'utilisation de "*Multipath TCP*" doit mener à un débit au moins égal à celui du meilleur des chemins.
— Augmenter la résistance aux pannes. La résistance totale de "*Multipath TCP*" doit être au moins aussi bonne que celle du chemin le plus fiable.

Notons que SCTP n'avait que le second but (SCTP ne permet pas actuellement d'envoyer des paquets sur différents chemins, cf. section 2.4). Naturellement, il faut en outre que le tout se fasse automatiquement : une machine MPTCP doit pouvoir trouver seule si son partenaire est également MPTCP et basculer automatiquement vers le nouveau protocole.

Mais, tout aussi importants que les fonctions sont les critères de compatibilité avec l'Internet d'aujourd'hui :

— Compatibilité avec les applications : l'API doit rester la même et les services rendus à l'application doivent être les mêmes (délivrance des octets dans l'ordre, et sans perte). Ainsi, on doit pouvoir passer à "*Multipath TCP*" uniquement en mettant à jour le système d'exploitation, sans réécrire les applications. Il est souhaitable que l'API soit étendue pour permettre une utilisation (optionnelle) des fonctions spécifiques de "*Multipath TCP*". Enfin, petite exception à cette compatibilité, une application qui s'attendrait à un débit très régulier peut être surprise par "*Multipath TCP*", qui aura probablement des débits et des latences moins constants. La section 6 donne des détails sur les relations entre "*Multipath TCP*" et les applications.
— Compatibilité avec le réseau : "*Multipath TCP*" devra fonctionner avec l'Internet tel qu'il est aujourd'hui. Or, comme l'illustrent très bien les figures 2 et 3 du RFC, l'architecture de l'Internet a changé, de fait. Elle était censée offrir un lien IP transparent, de façon à ce que la première couche « de bout en bout » soit la couche 4. En fait, l'Internet d'aujourd'hui est truffé de "*middleboxes*" qui ont souvent un rôle actif dans la couche 4 (terminaison de connexions, par exemple, ou bien filtrage des protocoles de couche 4 inconnus) et donc, de fait, la première couche de bout en bout aujourd'hui est la couche 7. La section 7 revient en détail sur cette interaction entre "*Multipath TCP*" et les "*middleboxes*".
— Compatibilité avec les utilisateurs existants : "*Multipath TCP*" ne doit pas « voler » de la capacité réseau aux utilisateurs du TCP normal (mais pas non plus être trop poli et les laisser tout prendre).

La deuxième exigence de compatibilité, celle avec le réseau, va donc être difficile à assurer. Mais elle est nécessaire pour le succès de "*Multipath TCP*", les innombrables "*middleboxes*" programmées avec les pieds et configurées n'importe comment étant un puissant frein au déploiement de tout protocole de transport qui ne serait pas TCP ou UDP.

"*Multipath TCP*" a également des contraintes en terme de sécurité : la section 2.3 impose qu'il n'aggrave pas la situation par rapport à TCP.

Une fois les buts exposés, comment les atteindre ? la section 3 décrit l'architecture de "*Multipath TCP*". En gros, "*Multipath TCP*" part des idées exposées dans « "*Breaking Up the Transport Logjam*" <http://www.brynosaurus.com/pub/net/logjam-abs.html> » de Ford & Iyengar : séparation de la couche transport en deux, une demi-couche haute « sémantique » (service aux applications, fonctionnant de bout en bout) et une demi-couche basse « flot et terminaison » (pas forcément de bout en bout, si des "*middleboxes*" sont présentes).

La section 4 de notre RFC descend ensuite dans le concret. On passe de l'architecture, "*Multipath TCP*", au protocole, MPTCP. Le principe de base de MPTCP est de répartir chaque connexion TCP en plusieurs « sous-connexions », chacune d'entre elles apparaissant à un observateur extérieur comme une connexion TCP complète et normale (et aura donc de bonnes chances de passer à travers les NAT et autres horreurs). Les méta-informations spécifiques à MPTCP seront transportées à part. MPTCP aura les tâches suivantes :

— Gérer les chemins : cela veut surtout dire les découvrir, essentiellement via la présence de plusieurs adresses IP sur une même machine. On pourra aussi utiliser ECMP (RFC 2992).
— Gérer les paquets : les octets transmis par l'application doivent être répartis entre les différents chemins. Cela implique la remise dans l'ordre à la destination, puisque des paquets ont pu voyager par des chemins différents.
— Gérer chaque sous-connexion : c'est relativement facile, chacune est une connexion TCP habituelle.
— Contrôler qu'il n'y a pas de congestion.

La section 5 rappelle les choix techniques importants de MPTCP et leurs raisons. Ainsi de la façon de numéroter les octets. La section 5.1 explique qu'il y avait deux façons :

— Un seul espace de numérotation, pour toute la connexion MPTCP. Cela aurait facilité le réassemblage des octets issus de chaque sous-connexion mais cela aurait rendu les sous-connexions bizarres, puisque les séquences auraient été incomplètes. Des engins comme les IDS auraient alors pu décider de couper ces connexions bizarres.
— La solution choisie a donc été d'avoir un espace de numérotation par sous-connexion, de façon à ce que chacune de celles-ci ressemble vraiment parfaitement à une connexion TCP traditionnelle. Cela implique par contre de conserver (et de transmettre au pair) la correspondance entre les numéros de séquence de la sous-connexion et ceux de la connexion complète.

Autre choix délicat, les retransmissions en cas de perte. Chaque sous-connexion étant du TCP normal, elle a un système d'accusé de réception. Mais si des données sont perdues, par exemple lors du réassemblage des sous-connexions, on ne peut pas utiliser les mécanismes TCP normaux pour demander une réémission puisque ces données ont déjà fait l'objet d'un accusé de réception. Il y a donc également un système d'accusés de réception et de réémission au niveau de la connexion MPTCP globale.

Plusieurs des fonctions de MPTCP nécessitent donc un mécanisme de signalisation entre les deux pairs, par exemple au démarrage lorsqu'il faut signaler au pair TCP qu'on sait faire du MPTCP. La section 5.4 explore l'espace des choix : les options TCP (RFC 793, section 3.1) ont été retenues pour cette signalisation, de préférence à un encodage TLV dans les données. Les options sont la solution standard de TCP et la seule propre. Elle présentent toutefois une limitation, leur faible taille, et un risque : certaines "*middleboxes*" massacrent les options (section 7 et voir aussi par exemple l'excellente étude de Google "*Probing the viability of TCP extensions*" <`https://www.imperialviolet.org/binary/ecntest.pdf`>).

Et comment va t-on identifier une connexion MPTCP ? Traditionnellement, l'identificateur d'une connexion TCP était un quadruplet {adresse IP source, adresse IP destination, port source, port destination}. La connexion MPTCP, reposant sur plusieurs connexions TCP, ne peut pas utiliser un de ces identificateurs (puisqu'il risque de ne plus rien signifier si la connexion TCP associée casse). MPTCP devra donc développer un nouvel identificateur (section 5.6). Cela limitera d'ailleurs la compatibilité de MPTCP avec les anciennes applications, qui n'utiliseront comme identificateur celui de la première connexion TCP. Si celle-ci défaille ultérieurement, la connexion MPTCP entière pourra avoir un problème.

L'Internet étant ce qu'il est, tout nouveau protocole apporte un lot de nouveaux problèmes de sécurité (section 5.8). Le cas de la sécurité de "*Multipath TCP*" est traité plus en détail dans un futur RFC « "*Threat Analysis for TCP Extensions for Multi-path Operation with Multiple Addresses*" ». En attendant, les propriétés de sécurité essentielles attendues sont :

— Permettre de vérifier que les pairs qui négocient une sous-connexion TCP sont bien les mêmes que ceux qui avaient négocié la première sous-connexion de la connexion MPTCP.

— Avant d'ajouter une adresse à la liste, permettre de vérifier que le pair peut recevoir du trafic à cette adresse, pour éviter qu'un pair MPTCP ne vous fasse attaquer un tiers innocent.
— Protéger contre le rejeu.

Atteindre ces objectifs ne va pas être facile pour MPTCP : le choix d'utiliser les options TCP limite sévèrement la quantité d'information qu'on peut transmettre et le choix de permettre qu'une "*middlebox*" s'interpose oblige à compter avec le fait que certains paquets seront modifiés en transit.

Revenons aux applications (section 6). Il y aura un certain nombre de points de détail qu'elles vont devoir gérer. Par exemple, si l'application se lie explicitement à une adresse particulière de la machine, MPTCP ne doit pas être utilisé (puisque l'application a marqué une préférence pour une adresse donnée).

L'interaction avec les équipements intermédiaires fait l'objet d'une étude plus approfondie en section 7. MPTCP ne sera pas déployé dans un réseau idéal, un réseau qui ne fournirait que de la délivrance de datagrammes IP. Au contraire, il opérera au milieu d'une jungle de "*middleboxes*" toutes plus mal programmées les unes que les autres : routeurs NAT, pare-feux, IDS, PEP, etc. Toutes ont en commun d'optimiser les applications d'aujourd'hui au détriment de celles de demain. En effet, violant la transparence du réseau, elles rendent très difficiles de déployer de nouvelles techniques qui font des choses inattendues. La section 7 donne une liste détaillées des différentes catégories de "*middleboxes*" et de leur influence possible. Ansi, certains pare-feux mélangent les numéros de séquence TCP (pour protéger les systèmes bogués qui ne choisissent pas le numéro initial au hasard). Si un tel équipement se trouve sur le trajet, l'émetteur TCP ne sait même pas quel numéro de séquence sera vu par le récepteur ! Quant aux IDS, ne connaissant pas MPTCP, ils trouveront peut-être suspect cet établissement de multiples connexions. À moins qu'ils ne comprennent pas que ces connexions sont reliées, ratant ainsi certaines informations transportées. (Imaginons une attaque transmise par MPTCP, où les différentes parties de l'attaque arrivent par des chemins séparés...).

Comme indiqué, MPTCP n'est pas encore complètement spécifié. Seule l'architecture générale est choisie. Il n'est donc pas étonnant qu'on n'en trouve pas encore d'implémentations officielle, que ce soit dans Linux ou dans FreeBSD. Néanmoins, le travail est déjà en cours sur Linux à l'Université de Louvain <http://inl.info.ucl.ac.be/mptcp> qu'on peut tester selon les instructions en <http://mptcp.info.ucl.ac.be/>.