RFC 7498 : Service Function Chaining Problem Statement

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 15 avril 2015

Date de publication du RFC : Avril 2015

https://www.bortzmeyer.org/7498.html

Le projet SFC <https://tools.ietf.org/wg/sfc> à l'IETF, dont voici le premier RFC, vise à mettre de l'ordre dans les innombrables traitements que suivent les flux IP dans l'Internet. Pare-feux, répartiteurs de charge, dispositifs de censure et d'espionnage, et leurs copains veulent tous appliquer des traitements aux communications et, actuellement, il n'existe pas de cadre unificateur pour ordonnancer proprement ces traitements, encore moins de normes techniques. Ce premier RFC expose le problème qu'on tente de résoudre.

Certains des traitements souhaités sont plutôt dans la couche 3, d'autres peuvent concerner les applications. La liste des traitements potentiellement effectués est très vaste, elle comprend les services de surveillance utilisant la DPI (comme les boîtes noires de Cazeneuve), le NAT, les caches, les divers optimiseurs (TCP, TLS) qui assurent certaines fonctions à la place de la machine terminale, etc. Le RFC n'en parle pas mais des fonctions de censure comme les mensonges DNS du réseau chinois <https://www.bortzmeyer.org/sichuan-pepper.html> rentrent également dans cette liste.

Certains traitements peuvent concerner une seule machine et d'autres la totalité des communications. Actuellement, ces traitements sont très couplés à la topologie du réseau. Par exemple, pour filtrer le trafic indésirable, on met un pare-feu en coupure du réseau, forçant tout le trafic à passer par le pare-feu. Si on veut effectuer un autre traitement, il faut mettre une seconde machine. Le tout devient vite très rigide, difficile à changer, et difficile à analyser dans ses effets combinés. C'est surtout dommage dans les environnements « à la demande » où on crée et détruit très vite des machines virtuelles et même des réseaux entiers.

Le terme de **SFC** ("*Service Function Chain*") a été créé pour désigner une nouvelle façon de concevoir ces services : on définit des fonctions abstraites (comme celle de pare-feu) et on a une liste ordonnée de telles fonctions, que le réseau peut appliquer, indépendamment de sa topologie. Le concept est développé plus longuement en section 3 du RFC.

Mais, d'abord, en section 2, l'exposé plus détaillé du problème. Cette section est le gros morceau de ce RFC. On l'a vu brièvement plus haut, aujourd'hui, pour assurer une fonction sur les flux réseau, il faut une topologie particulière. Par exemple, le pare-feu doit être sur le chemin et avoir deux « pattes », introduire un pare-feu là où il n'y en avait pas nécessite donc souvent de ré-architecturer son réseau, ajoutant des sous-réseaux IP qui n'étaient pas nécessaires. Ça manque sérieusement de souplesse. Idéalement, on voudrait que le réseau soit architecturé uniquement autour de la délivrance des paquets, et que les services en plus soient déployables en appuyant sur un bouton, sans refaire le réseau à chaque nouveau service.

Un autre exemple de rigidité concerne les répartiteurs de charge, si communs devant les serveurs Web. Il faut souvent configurer le répartiteur comme étant le routeur par défaut des serveurs HTTP, si bien que même le trafic non-Web (par exemple les communications de gestion en SNMP ou SSH) passe par le répartiteur, sans nécessité.

Autre conséquence de cette dépendance vis-à-vis de la topologie, cela rend difficile le changement du service de transport sous-jacent. Si on passe de MPLS à GRE, on risque de devoir changer pas mal de services, alors que, a priori, ils étaient indépendants des couches basses.

Tout cela fait que les réseaux sont plus complexes qu'ils ne le devraient. Changer un paramètre aussi simple que l'ordre dans lequel sont appliqués des services (filtrer d'abord, puis répartir le trafic après, ou bien le contraire) nécessite de changer la topologie. Résultat, les réseaux tendent à rester plutôt statiques, et à ne pas profiter à fond de la souplesse que permettrait la virtualisation (section 2.7). Et assurer la haute disponibilité devient très compliqué, puisque les services de secours doivent conserver la même topologie que le service principal (afin de recevoir les mêmes traitements).

Un autre problème lié au couplage trop fort entre les services et la topologie du réseau est celui de la sélection du trafic (section 2.8). Souvent, on ne veut appliquer les services qu'à une partie du trafic. Or, dans les architectures actuelles, cela va souvent obliger à faire passer tout le trafic par la machine qui met en œuvre le service. Ou, sinon, à mettre en œuvre des mécanismes de redirection peu pratiques. Prenons un exemple : on veut censurer certaines pages de Wikipédia mais pas toutes. Faire passer la totalité du trafic par le système de DPI qui déterminera quelles pages sont demandées serait lent, nécessiterait de grosses machines pouvant faire de la DPI à ce rythme et pourrait être considéré comme abusif par rapport au but à atteindre. Actuellement, les censeurs déploient donc souvent des architectures mixtes ("*proxy*" HTTP transparent, puis redirection du trafic des serveurs Wikipédia - et de seulement ceux-ci - vers le système de DPI, qui n'aura donc à traiter qu'une partie du trafic). C'est quand même peu élégant, et il serait donc souhaitable de découpler sélection du trafic et application du service.

Et, bien sûr, dans l'état actuel des choses, comme il n'y a aucun standardisation des services réseaux et de la manière dont ils s'insèrent, faire coexister des services réseau de plusieurs vendeurs différents est très difficile. Une standardisation, même partielle, simplifierait la vie des administrateurs réseau (le but de ce RFC n'est pas de réaliser cette standardisation, ce qui serait très prématuré, mais d'explorer le problème).

La section 3 de notre RFC, elle, décrit de manière positive ce qu'on attend du "*service chaining*", de la composition des services. Pour découpler le service de la topologie, il faudra un "*overlay*" par service. Changer la liste, ou l'ordre de la liste des services appliqués, ne nécessitera donc que de changer la connexion entre ces "*overlays*".

Le mécanisme devra fournir un moyen de classer le trafic, en fonction de critères de sélection. En pratique, les possibilités exactes du système de classification dépendront du matériel et logiciel sous-jacents, et la classification sera donc plus ou moins grossière.

Souvent, les services réseau n'ont besoin que des métadonnées pour agir, et il n'est donc pas forcément nécessaire d'envoyer aux services la totalité des flux.

Voilà, c'est très vague, comme description, je sais, mais c'est l'état actuel de la réflexion sur la composition de services (et cette section 3 a déjà suscité beaucoup de discussion dans le groupe de travail). D'autres RFC sont en préparation, plus concrets, comme le document d'architecture de SFC, le RFC 7665 [1].

Un tel système de composition de services soulève évidemment d'amusantes questions de sécurité (section 4). Certaines sont traitées par des protocoles existants (par exemple la création et la maintenance des "*overlays*", qui n'est pas un problème nouveau).

Le RFC note que certains services peuvent nécessiter des connaissances détaillées sur les données transmises, et que des techniques comme le chiffrement peuvent donc limiter ces services. En pratique, on aura moins d'informations, mais on pourra souvent se débrouiller avec. (Il reste des métadonnées non chiffrées.) D'ailleurs, à propos de chifffrement, le RFC note qu'il faudra aussi protéger le trafic SFC : les machines qui effectuent un certain service peuvent être loin de celles qui se « contentent » de faire passer les paquets et il ne faudrait pas que le trafic entre les deux soit espionné ou, pire, modifié, par un tiers (la protection est parfois assurée par le protocole d'"*overlay*"). C'est particulièrement important dans un contexte multi-clients (un grand réseau partagé par de nombreux utilisateurs).

À noter qu'Ericsson prétend avoir un brevet <http://datatracker.ietf.org/ipr/2285/> même sur la simple description du problème !

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc7665.txt