RFC 8569 : Content Centric Networking (CCNx) Semantics

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 16 juillet 2019

Date de publication du RFC : Juillet 2019

https://www.bortzmeyer.org/8569.html

Certaines personnes trouvent une utilité au réseau centré sur le contenu, où adressage et nommage ne désignent que du contenu, des ressources numériques auxquelles on accède via le réseau. (Cette idée est souvent nommée ICN, pour "*Information-Centric Networking*" ou NDN pour "*Named Data Networking*".) Le groupe de recherche ICNRG <https://irtf.org/icnrg> développe des spécifications pour normaliser certains aspects de ce réseau centré sur le contenu, un projet nommmé **CCNx** (pour "*Content Centric Networking*"). Ce nouveau RFC décrit les concepts de base de CCNx. CCNx est le premier protocole conçu par le groupe ICNRG.

Précisons tout de suite : ce point de vue comme quoi le socle du réseau devrait être l'accès au contenu est très contestable. Il évoque fortement le raccourci de certains journalistes, décideurs, et opérateurs de télécommunication traditionnels comme quoi le seul usage de l'Internet fait par M. Michu serait « accéder à du contenu ». Mais j'ai déjà développé ces critiques dans un autre article <https://www.bortzmeyer.org/van-jacobson-ccn.html> il y a huit ans, donc je ne les reprendrai pas ici.

Les acteurs du réseau centré sur le contenu sont notamment :

— Le producteur ("*producer*" ou "*publisher*") qui crée du contenu. Il peut avoir une clé qui lui permet de signer ce contenu.
— Le consommateur ("*consumer*") qui veut accéder à du contenu.

(Vous noterez qu'il n'y a pas de pair à pair dans ce réseau, ce qui limite certains usages.)

Le protocole CCnx repose sur deux types de messages : `Interest`, par lequel on signale qu'on aimerait bien récupérer tel contenu, et `Content Object`, qui transporte un contenu. En général, la machine de l'utilisateur, du consommateur demandeur, enverra un `Interest` et, si tout va bien, récupérera en échange un ou plusieurs `Content Object`. Si tout va mal, on aura à la place un `InterestReturn`, qui signale un problème. Comment sont désignés les contenus ? Par des noms hiérarchiquement organisés, comme dans les URL d'aujourd'hui (section 3 du RFC). Un nom est donc composé d'une série de segments, et la correspondance entre un nom demandé et une entrée de la table de routage est toujours faite de manière exacte, bit à bit. (Pas d'insensibilité à la casse, pas de recherche floue.) Le nom est

opaque. Il n'est donc pas forcément lisible par un humain. Comme les noms sont hiérarchiques, un nom peut être exact (le nom entier) ou être un préfixe (correspondant à plusieurs noms). On dit aussi qu'un nom est complet quand il inclut un condensat du contenu (comme dans les "*magnets*" de BitTorrent). Le condensat est expliqué plus en détail dans les sections 5 et 7 du RFC. La syntaxe est décrite dans l'"*Internet Draft*" `draft-mosko-icnrg-ccnxurischeme`, qui met les noms CCNx sous forme d'URI. Par exemple, un nom possible est `ccnx:/NAME=foo/APP:0=bar`. Il n'y a pas de registre de tels noms pour l'instant. Si on veut trouver des noms, le protocole lui-même ne le permet pas, il faudra bâtir des solutions (par exemple un moteur de recherche) au-dessus de CCNx.

CCNx fonctionne en relayant les messages (aussi bien `Interest` que `Content Object`) d'une machine à l'autre. Du fait de ce modèle de relayage systématique, il faut ajouter un troisième acteur au producteur et au consommateur, le **relayeur** ("*forwarder*"), qui est toute machine intermédiaire, un peu comme un routeur sauf que le relayeur fait bien plus de choses qu'un routeur. Par exemple, contrairement au routeur IP, le relayeur a un état. Chaque demande d'objet qui passe est mémorisée par le relayeur (dans une structure de données nommée la PIT, pour "*Pending Interest Table*"), qui saura donc où renvoyer la réponse. CCNx est "*sourceless*", contrairement à IP : l'adresse source n'est pas indiquée dans la demande.

La FIB ("*Forwarding Information Base*") est la table de routage de CCNx. Si elle contient une entrée pour le contenu convoité, cette entrée indique où envoyer la requête. Sinon, la requête ne peut aboutir. Notez que ce RFC ne décrit **pas** le protocole par lequel cette FIB sera construite. Il n'existe pas encore d'OSPF ou de BGP pour CCNx.

Comme le contenu peut être récupéré après un passage par pas mal d'intermédiaires, il est crucial de vérifier son intégrité. CCNx permet plusieurs méthodes, de la signature au HMAC. Ainsi, l'intégrité dans CCNx est une protection des objets (du contenu), pas uniquement du canal comme c'est le cas avec HTTPS. CCNX permet également de signer des listes d'objets (des manifestes), la liste contenant un SHA ou un CRC des objets, ce qui permet d'assurer l'intégrité de ceux-ci.

Ces concepts avaient été décrits dans les RFC 7476 [1] et RFC 7927. Le vocabulaire est expliqué dans le RFC 8793. Maintenant, voyons les détails, sachant que le format précis des messages a été délégué à un autre RFC, le RFC 8609. La section 2 du RFC décrit précisement le protocole.

On a vu que le consommateur commençait l'échange, en envoyant un message de type `Interest`. Ce message contient le nom de l'objet qui intéresse le consommateur, et éventuellement des restrictions sur le contenu, par exemple qu'on ne veut que des objets signés, et avec tel algorithme, ou bien ayant tel condensat cryptographique de l'objet (le tuple regroupant nom et restrictions se nomme le lien, cf. section 6). Un nombre maximal de sauts restants est indiqué dans le message. Décrémenté par chaque relayeur, il sert à empêcher les boucles (lorsqu'il atteint zéro, le message est jeté). Le producteur, lui, stocke le contenu, indexé par les noms, et signale ces noms sur le réseau pour que les relayeurs peuplent leur FIB (on a vu que le protocole permettant ce signalement n'était pas encore défini, bien que plusieurs propositions existent). Enfin, le relayeur, le troisième type d'acteur, fait suivre les `Interest` dans un sens (en consultant sa FIB) et les `Content Object` en sens inverse (en consultant sa PIT).

Le relayeur a également une mémoire (un cache), qui sert notamment à accélérer l'accès au contenu le plus populaire (section 4 du RFC). Il existe des moyens de contrôler l'utilisation de cette mémoire, par exemple deux champs dans un `Content Object`, la date d'expiration, après laquelle il ne faut plus

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc7476.txt`

garder l'objet dans le cache, et la date de fin d'intérêt, après laquelle il n'est sans doute plus utile de garder l'objet (mais on peut quand même si on veut).

La validation des objets, leur vérification, est un composant crucial de CCNx. Elle est spécifiée dans la section 8 du RFC, avec ses trois catégories, la validation utilisant un simple condensat (pas forcément cryptographique), un HMAC ou bien une signature.

On a vu que le troisième type de message de CCNx, après `Interest` et `Content Object`, était `Interest Return`. Il est décrit en détail dans la section 10 de notre RFC. Notez tout de suite qu'il peut ne pas y avoir de réponse du tout, un relayeur n'étant pas forcé d'envoyer un `Interest Return` s'il ne peut acheminer un `Interest`. S'il y a un `Interest Return`, il indique l'erreur, par exemple "*No Route*" (aucune entrée dans la FIB pour ce nom), "*No Resources*" (le relayeur manque de quelque chose, par exemple de place disque pour son cache), "*Malformed Interest*" (un problème dans la demande), "*Prohibited*" (le relayeur n'a pas envie de relayer), etc.

Enfin, sur la question cruciale de la sécurité, la section 12 du RFC revient sur quelques points sensibles. Par exemple, j'ai dit plus haut que les objets pouvaient être validés par une signature numérique. Mais où trouve-t-on les clés publiques des producteurs, pour vérifier leur signature ? Eh bien ce point n'est pas actuellement traité. Notez que les relayeurs, eux, ne sont pas obligés de valider et un cache peut donc contenir des données invalides. Les RFC 7927 et RFC 7945 sont des bonnes ressources à lire sur la sécurité des réseaux centrés sur le contenu.

Il existait une version précédente du protocole CCNx, identifiée « 0.x », et décrite dans « "*Networking Named Content*" `<http://dx.doi.org/10.1145/1658939.1658941>` ». Dans 0.x, la correspondance entre le nom demandé dans un `Interest` et celui obtenu était hiérarchique : le nom demandé pouvait être un préfixe du nom obtenu. La version décrite dans ce RFC, « 1.0 », est plus restrictive ; le nom obtenu doit être exactement le nom demandé. Les fonctions de recherche ne sont pas dans CCNx, elles doivent être dans un protocole de couche supérieure, un Google ou Pirate Bay du réseau CCN. Un exemple d'un tel protocole est décrit dans l'"*Internet Draft*" `draft-mosko-icnrg-selectors`.

Et questions mise en œuvre du protocole CCNx ? Il en existe au moins deux, Community ICN `<https://wiki.fd.io/view/Cicn>`, et CCN-Lite `<http://www.ccn-lite.net/>` (cette dernière, tournant sur RIOT, visant plutôt l'Internet des objets).