RFC 7069 : DECoupled Application Data Enroute (DECADE)

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 27 novembre 2013

Date de publication du RFC : Novembre 2013

https://www.bortzmeyer.org/7069.html

L'architecture traditionnelle de l'Internet est celle d'un réseau « bête », transportant des paquets sans en comprendre le contenu, avec toute l'« intelligence » concentrée aux extrémités, dans les machines terminales <https://www.bortzmeyer.org/terminal-host.html>. Un travail avait été commencé à l'IETF, dans le défunt groupe DECADE <http://tools.ietf.org/wg/decade>, pour voir s'il était possible de changer légèrement ce modèle pour certaines catégories d'applications (notamment pair-à-pair), en dotant le réseau de capacité de stockage des données (un peu comme le font, de manière non-standard, les CDN). Le projet a finalement échoué mais a publié quelques RFC, dont ce dernier document, qui décrit l'architecture d'un tel système.

Les deux autres RFC importants de DECADE étaient le RFC 6646 [1], exposé détaillé du problème, et le RFC 6392, qui fait le tour des mécanismes disponibles. (Il est très recommandé de lire au moins le RFC 6646 avant ce RFC 7069.) Finalement, le projet a été abandonné et le groupe DECADE dissous, mais, pour ne pas perdre le travail effectué, il est publié dans ce nouveau RFC, qui décrit, en termes assez généraux, l'architeecture de DECADE. Comme exemple d'usage, prenons une machine connectée en ADSL. Elle a typiquement une capacité réseau « montante » très limitée. Mettre à jour le réseau pour augmenter cette capacité (par exemple pour passer en FTTH) risque d'être coûteux. Disposer dans le réseau des dispositifs de stockage permettant à cette machine de distribuer du contenu sur l'Internet sans tuer sa capacité ADSL limitée serait peut-être plus économique. (Notez aussi que cela soulève d'intéressants problèmes politiques de contrôle sur les données servies.)

Donc, l'idée est d'avoir des serveurs DECADE dans le réseau, qui stockent les données des clients DECADE. Les données pourront être copiées entre serveurs, et récupérées par d'autres clients que ceux

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc6646.txt

qui les ont déposées. Pour cela, il faudra un protocole de gestion des ressources déposées, le DRP ("*DECADE Resource Protocol*") et un protocole d'accès aux ressources, le SDT ("*Standard Data Transfer protocol*"). Pour ce dernier, il était prévu dès le début d'utiliser un protocole standard existant comme HTTP.

Les serveurs DECADE seront fournis par des fournisseurs de stockage, qui pourront être les FAI ou bien d'autres acteurs. Ces fournisseurs décideront de l'allocation de ressources (espace de stockage, capacité réseau).

Il n'était pas prévu que les utilisateurs accèdent aux serveurs DECADE directement mais que ceux-ci fournissent un service aux applications, service masqué aux utilisateurs. La section 3 de notre RFC contient un schéma qui illustre ce principe : une application veut envoyer des données à une autre application (les deux applications parlent entre elles avec un protocole pair-à-pair quelconque). Pour cela, elle envoie les données au serveur avec le SDT (rappelez-vous que c'est un protocole standard comme HTTP) et utilise le DRP pour indiquer au serveur les conditions d'accès aux données. L'autre application utilisera SDT pour récupérer les données. Selon le niveau de confidentialité des données, il y aura aussi peut-être une étape de récupération d'un jeton que l'envoyeur transmettra au destinataire (toujours avec leur protocole pair-à-pair à eux, qui ne regarde pas DECADE), et avec lequel le destinataire pourra prouver ses droits au serveur DECADE. Un peu comme les applications « passe-plat » existants comme `<http://dl.free.fr/>` mais masqué à l'utilisateur. Entre parenthèses, ce scénario d'usage avait été réclamé dans un excellent dessin de xkcd `<http://xkcd.com/949/>`.

La section 4 décrit l'architecture du service DECADE (s'il avait été construit ; rappelez-vous que le projet a été abandonné). En gros, DECADE fournira les données ("*data plane*"), l'application pair-à-pair la signalisation ("*control plane*"). DECADE se voulait indépendant de l'application, fournissant un service général de données. Les applications auraient géré les services comme l'indexation, le moteur de recherche dans les données, etc. Des systèmes de stockage des données séparant données et signalisation existent déjà. Le RFC mentionne Google File System, ou l'extension pNFS de NFS, décrite dans la section 12 du RFC 5661.

Une chose importante avec DECADE est que ce système ne prévoit pas de modifier le contenu déposé dans le stockage en réseau. Les objets écrits sont immuables (section 4.2). Ce principe permet de simplifier considérablement le système, puisqu'il n'y a plus de problème de synchronisation entre les différents serveurs de stockage. Si on veut vraiment modifier une ressource mise en ligne, il faut détruire l'ancienne et en mettre une nouvelle. Ce sera à l'application, pas à DECADE, de changer la correspondance entre le nom de la ressource et le nouvel objet qui, pour DECADE, sera un objet différent, sans lien avec le premier. Chaque application aura en effet son propre système de nommage. On peut comparer cela avec le nommage dans un système de fichiers (comme `/home/stephane/Downloads/galaxy.avi`) contre celui de plus bas niveau utilisé par le système de fichiers pour parler aux contrôleurs de disque.

Puisqu'on a commencé à parler d'identité (« le même objet ») et donc d'identificateurs, quels sont les identificateurs dans DECADE ? Chaque objet a un identificateur unique. Si un objet est répliqué sur plusieurs serveurs, toutes ces copies sont toujours désignées par le même identificateur. DECADE n'envisageait pas d'imposer un mécanisme unique de nommage mais le RFC suggère d'utiliser des condensats du contenu, comme dans le RFC 6920 (voir aussi la section 6.1). Là encore, l'immuabilité des objets stockés est essentielle pour que ce nommage par le contenu fonctionne.

La section 5 est plus concrète, décrivant certains des composants du système. Par exemple, elle couvre les jetons d'authentification, dont l'utilisation était prévue pour que le déposant d'un fichier puisse distribuer des droits d'accès à des lecteurs (voir aussi la section 6.2.1). Elle parle aussi de l'importance d'avoir un mécanisme de découverte (non spécifié : rappelez-vous que le travail sur DECADE avait été interrompu en route) pour localiser le serveur DECADE approprié.

La section 6, elle, est consacrée aux protocoles DRP (protocole de contrôle) et SDT (protocole de transport des données). Elle est également assez abstraite : l'idée originale est que des protocoles concrets seraient choisis ultérieurement. Ces protocoles ne sont pas utilisés seulement entre un client DECADE et un serveur DECADE mais aussi entre serveurs, notamment pour la réplication. Les données stockées dans DECADE peuvent en effet être automatiquement copiées de serveur à serveur pour augmenter les performances lors de la récupération (section 6.4). Une évaluation des protocoles possibles pour remplir ces rôles figure dans l'annexe A.

HTTP (RFC 7230) est évidemment le premier protocole considéré. C'est un candidat presque idéal pour le rôle de SDT mais il peut aussi servir de DRP. HTTPS fournit des mécanismes de sécurité bien compris et largement déployés. HTTP permet d'assurer les deux fonctions du SDT, en lecture mais aussi en écriture (avec les méthodes `PUT` et `POST`). Par contre, il lui manque quelques fonctions, comme des ACL ou comme des mécanismes de transmission de politique d'accès au serveur. Ceci dit, ces manques pourraient être comblés en ajoutant quelques en-têtes dans les requêtes et réponses HTTP. C'est ce que fait Google Storage <`https://developers.google.com/storage/docs/concepts-techniques`>. Ce dernier utilise OAuth (RFC 6749) pour les délégations d'accès.

Un autre candidat est CDMI. Il est basé sur HTTP mais avec de nombreux enrichissements. Lui dispose d'ACL aussi riches qu'on le veut. Et, question contrôle des données, il permet de spécifier des choses comme le nombre de copies à générer, leur placement géographique, la durée de conservation, etc. CDMI parait donc plus proche des besoins de DECADE.

Et la sécurité d'un système comme DECADE ? La section 8 la décrit en détail, en s'appuyant sur la section 5 du RFC 6646. Il y a évidemment le risque d'attaques par déni de service, par exemple si un client méchant envoie une quantité astronomique de données à stocker.

Pour l'accès aux données, DECADE comptait sur un mécanisme de délégation : on n'autorise pas le client, mais on distribue des jetons dont la possession suffit pour l'accès. Les problèmes de sécurité liés à ces mécanismes de délégation sont traités dans la section 10 du RFC 6749.

Un autre type d'attaque est dirigé contre le client : arriver à lui faire télécharger des données qui ne sont pas celles qu'il espérait. Ce genre d'attaques est courant dans le monde du téléchargement pair-à-pair où un fichier nommé `hot-naked-women-at-the-beach.avi` peut en fait contenir un discours d'un télévangéliste. S'il est difficile de se prémunir contre un nom trompeur, en revanche, DECADE peut sécuriser la liaison entre un nom et un objet. Ainsi, des adresses basées sur un condensat du contenu sont auto-validantes : après le téléchargement, on recalcule le condensat et on vérifie que le contenu est bien celui demandé. Cela ne règle qu'une partie du problème : la publicité mensongère, par exemple pour tromper un moteur de recherche, reste possible.