RFC 5694 : Peer-to-peer (P2P) Architectures

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 24 novembre 2009

Date de publication du RFC : Novembre 2009

https://www.bortzmeyer.org/5694.html

Il existe un très grand nombre de systèmes pair-à-pair et leur succès est un témoignage de la capacité de l'Internet à accepter des applications complètement imprévues à l'origine. Cette variété est telle qu'il est difficile de s'y retrouver, par exemple pour d'éventuels travaux de normalisation sur les protocoles pair-à-pair. D'où ce document de l'IAB, qui définit ce qu'est un système pair-à-pair, et propose une taxonomie de ceux-ci. Ce RFC propose également des méthodes pour analyser si un système pair-à-pair est adapté à un problème donné.

Pourtant, on ne manque pas de textes sur les systèmes pair-à-pair. Il y a eu de très nombreuses publications scientifiques, plusieurs conférences uniquement consacrées à ce sujet, et un groupe de travail de l'IRTF lui est entièrement voué. Et, bien sûr, il existe de nombreuses applications pair-à-pair, certaines avec un code source disponible, et parmi lesquelles se trouvent des applications parmi les plus populaires sur l'Internet.

Mais il n'existe pas encore de compréhension commune de ce qu'est le pair-à-pair et, pire, beaucoup de gens, de bonne ou de mauvaise foi, assimilent « pair-à-pair » aux seules activités illégales de transfert de fichiers (section 1 du RFC).

Le but de de document est donc d'aider à la construction de cette compréhension commune et de montrer que le pair-à-pair est largement utilisé pour des activités tout à fait légitimes.

La section 2 du RFC s'attaque à la définition du pair-à-pair. Il existe plusieurs définitions, qui ne sont pas forcément équivalentes, d'autant plus que le marketing s'en mêle et que « pair-à-pair » est souvent un terme vendeur. En outre, la section 2 prévient aussi que certains systèmes souvent qualifiés de « pair-à-pair » ne correspondront que partiellement à cette définition et ne pourront donc pas être considérés comme complètement « pair-à-pair ». Ce n'est pas forcément un défaut : le but est de bâtir le meilleur système possible, pas de coller à la définition du « pair-à-pair ». Bien des systèmes réussis sont « mixtes », mélangeant « pair-à-pair » et « centralisé ».

La définition de ce RFC 5694 [1] est donc : « Est pair-à-pair un système où les éléments qui le composent partagent leurs ressources pour fournir le service pour lequel le système est prévu. Chaque élément du système fournit des services et en reçoit. » En théorie, cette définition impose que **tous** les éléments partagent leurs ressources mais, en pratique, il existe des systèmes où certains éléments sont des clients purs, ne fournissant aucune ressource. Le RFC considère qu'un système où les clients seraient nettement plus nombreux que les pairs ne serait pas réellement pair-à-pair. Par contre, le RFC note explicitement que le fait que certaines parties du système soient centralisées ne l'empêche pas d'être pair-à-pair. L'exemple cité est celui d'un système où il y aurait un serveur central pour enregistrer les nouveaux pairs. Un tel système serait pair-à-pair, selon la définition proposée.

On peut ensuite faire varier cette définition à l'infini, en ajoutant que les pairs doivent participer à des transactions qui ne leur bénéficient pas directement, ou que les pairs doivent pouvoir communiquer sans passage par un intermédiaire, ou que le contrôle doit être entièrement distribué (sans aucun serveur central, contrairement à ce qu'accepte la définition du RFC 5694).

Il est d'autant plus difficile de classer les systèmes pair-à-pair que beaucoup sont composites, faits de plusieurs services, pas forcément pair-à-pairs. Ainsi, si j'utilise un moteur de recherche BitTorrent accessible par le Web, comme btjunkie, pour récupérer ensuite un fichier avec le protocole BitTorrent, suis-je en pair-à-pair ou pas, le Web étant client/serveur ?

Notre définition étant posée, le reste de la section 2 va l'appliquer à divers systèmes existants, pour voir s'ils peuvent être considérés pair-à-pair ou pas.

La section 2.1 commence ces applications avec le DNS. Les résolveurs DNS étant des clients purs, qui ne fournissent aucun service, le DNS ne peut pas être considéré comme pair-à-pair selon la définition de notre RFC.

Et SIP (RFC 3261) ? La section 2.2 en rappelle les principes, notamment le fait que les "*user agents*" sont à la fois client et serveurs. Mais un "*user agent*" ne participe qu'aux transactions qui ont un intérêt pour lui (parce qu'il appelle ou bien qu'il est appelé) et le SIP traditionnel ne peut pas être considéré comme pair-à-pair. En revanche, P2PSIP, décrit en section 2.3, est différent : les "*user agents*" ne s'enregistrent plus auprès d'un serveur mais entre eux et la fonction de rendez-vous nécessite donc la participation d'autres pairs, qui n'y tirent pas de bénéfice personnel. P2PSIP est donc bien pair-à-pair.

L'un des protocoles pair-à-pair les plus connus est BitTorrent. La section 2.4 examine son cas. Comme un pair BitTorrent, non seulement récupère des fichiers pour son usage personnel, mais en outre en envoie à d'autres, sans que cela ne présente d'interêt pour eux, BitTorrent est bien pair-à-pair. Toutefois, le monde réel est toujours plus compliqué que cela et le RFC note que, si la majorité de l'**essaim** (l'ensemble des pairs), est composée de pairs ayant récupéré tous leurs fichiers et ne faisant plus que les distribuer (des "*seeders*" dans la terminologie BitTorrent), alors, l'essaim ne fonctionne plus vraiment en pair-à-pair.

Après ces études de cas, le RFC se penche, dans sa section 3, sur les fonctions qu'il faut assurer pour qu'un système pair-à-pair fonctionne. Au minimum, et quel que soit le service fourni par ce système, il faut assurer :

— Le recrutement, c'est-à-dire le mécanisme d'authentification et d'autorisation auxquels sont soumis les nouveaux pairs,

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc5694.txt`

— La découverte des pairs : le nouveau nœud du réseau doit trouver un ou plusieurs pairs pour commencer la communication.

Dans certains systèmes pair-à-pairs (par exemple eDonkey), ces fonctions sont centralisées. Un serveur de recrutement (dit parfois "*bootstrap server*") est contacté par les nouveaux venus, et il peut, après les avoir autorisés, les mettre en contact avec les autres pairs. Ce serveur est évidemment un point vulnérable du réseau mais notre RFC 5694 ne considère pas que son existence empêche le système d'être authentiquement pair-à-pair. En effet, les fonctions considérées comme significatives sont celles qui ont un rapport direct avec le service fourni.

À noter que ces fonctions peuvent avoir été programmées une fois pour toutes dans un ensemble logiciel qui sert de base pour la construction d'un système pair-à-pair (cf. section 5.4). C'est le cas par exemple de JXTA.

Celles-ci dépendent du service et ne sont donc pas présentes systématiquement. On trouve parmi elles :

— L'indexation des données présentes dans le système.
— Le stockage des données présentes dans le système.
— Les calculs que doit faire le système.
— L'acheminement de messages d'un pair à l'autre, acheminement qui peut impliquer plus de deux pairs.

Il reste à classer les différents systèmes pair-à-pair. La section de taxonomie est la numéro 4. Le problème n'est pas simple puisqu'il existe déjà plusieurs classifications incompatibles des systèmes pair-à-pair. Pire, le vocabulaire n'est même pas identique entre les auteurs (par exemple pour des termes comme « première génération » ou « deuxième génération »). Notre RFC propose de partir de la fonction d'**indexation**. Un index peut être **centralisé**, **local** ou **distribué** (cf. RFC 4981). Dans le premier cas, une seule machine garde trace de toutes les données (Napster fonctionnait ainsi). Dans le second, chaque machine se souvient des données qui l'intéressent (les premières versions de Gnutella fonctionnaient ainsi). Avec un index distribué, les références aux données se trouvent sur plusieurs machines. Les DHT sont un bon exemple d'un tel système.

On peut aussi classer les index entre ceux avec ou sans sémantique. Les premiers contiennent des informations sur les relations entre les données et les métadonnées mais pas les seconds. Les index avec sémantique permettent donc des recherches plus riches mais ne trouvent pas forcément toutes les données présentes.

D'autres auteurs ont classé les systèmes pair-à-pair entre systèmes **hybrides** (pas complètement pair-à-pair, par exemple parce qu'ils ont un index centralisé) et **purs**. Les systèmes purement pair-à-pair peuvent continuer à fonctionner même quand n'importe laquelle des machines qui les composent est en panne ou arrêtée.

Une autre classification est selon la structure. Les systèmes non structurés sont ceux où un nouveau pair peut se connecter à n'importe quel pair existant. Les systèmes structurés imposent au pair arrivant de ne se connecter qu'à certains pairs bien définis, en fonction de leur identité (c'est par exemple le cas des DHT, où les pairs sont en général organisés en un anneau virtuel dans l'ordre des identificateurs). Mais, là encore, la distinction n'est pas absolue et on trouve, par exemple, des systèmes non structurés où certains pairs ont des rôles particuliers.

Une autre classification qui a eu un relatif succès médiatique est par « génération ». Certains auteurs ont classé les systèmes à index centralisé dans une « première génération du pair-à-pair », ceux à index local dans la seconde génération et ceux utilisant une DHT dans la troisième. Cette classification a plusieurs problèmes, l'un des principaux étant qu'elle laisse entendre que chaque génération est supérieure à la précédente.

Aucune de ces classifications ne prend en compte les fonctions de recrutement et de découverte : un système peut donc être pair-à-pair même si un des ces fonctions, ou les deux, est effectuée par un mécanisme centralisé.

Si on développe des systèmes pair-à-pair, c'est parce qu'ils ont une utilité concrète. La section 5 décrit les principaux domaines d'application des systèmes pair-à-pair. Le plus connu est évidemment la distribution de contenu, qui fait l'objet de la section 5.1. Parfois, ce contenu est composé de fichiers illégalement distribués, par exemple parce que ledit contenu n'est pas encore monté dans le domaine public dans tous les pays. Mais le pair-à-pair va bien au delà des utilisations illégales. La distribution de contenu sur Internet se fait par de nombreux moyens, HTTP, RFC 7230 et FTP, RFC 959 étant les plus traditionnels, et personne ne prétend que HTTP est néfaste parce que du contenu illégal est parfois distribué par HTTP.

Parmi les exemples importants de distribution de contenu légal par le pair-à-pair, il y a les systèmes d'exploitation dont les images ISO sont distribués en BitTorrent (comme Debian ou NetBSD), soulageant ainsi les serveurs de l'organisation, qui n'a pas en général de grands moyens financiers pour leur fournir une capacité réseau suffisante. Il y a aussi les mises à jour de logiciel commerciaux comme World of Warcraft et bien d'autres.

Le gros avantage du pair-à-pair, pour ce genre d'applications, est le passage à l'échelle. Lorsqu'une nouvelle version de NetBSD est publiée, tout le monde la veut en même temps. Un serveur FTP ou HTTP central, même avec l'aide de dix ou vingt miroirs, s'écroulerait sous la charge, tout en restant très sous-utilisé le reste du temps. Au contraire, un protocole comme BitTorrent marche d'autant mieux qu'il y a beaucoup de téléchargeurs (les "*leechers*" dans la terminologie BitTorrent). L'effet est équivalent à celui de milliers de miroirs, sans que l'utilisateur n'aie à les choisir explicitement.

Cette question de la sélection du pair d'où on télécharge le fichier est centrale pour l'efficacité d'un système pair-à-pair. L'un des problèmes que pose cette sélection est que les intérêts des différentes parties peuvent être contradictoires. Les utilisateurs veulent le pair le plus rapide, les gérants du système de distribution du contenu voudraient que les pairs possédant les parties les plus rares soient sélectionnés en premier (pour augmenter la fiabilité globale de la distribution), les FAI voudraient que les pairs situés sur leur propre réseau soient privilégiés. Il n'est pas étonnant que la sélection du pair aie été tant discuté au "*IETF workshop on P2P Infrastructure*" de 2008 (dont un compte-rendu figure dans le RFC 5594). Le groupe de travail Alto <https://www.bortzmeyer.org/alto-wg.html> de l'IETF est chargé de travailler sur ce point.

La section 5.1 se termine sur un exemple original, Zebranet <http://www.princeton.edu/~mrm/zebranet.html>, un système de collecte de données sur les zèbres en liberté. Chaque animal porte un collier qui collecte des informations et les transmet aux stations qui se déplacent dans la savane. Mais il arrive souvent qu'un zèbre ne passe jamais près d'une station ! Zebranet permet alors la transmission de données de collier à collier. Chaque zèbre rassemble ainsi les données de plusieurs autres animaux, et le passage d'un seul zèbre près de la station permet de récolter toutes ces données d'un coup. Le RFC note toutefois que Zebranet n'est pas vraiment pair-à-pair puisque le collier ne fait que donner et la station que recevoir.

Une autre utilisation courante du pair-à-pair est le calcul distribué (section 5.2). Une tâche de calcul y est divisée en plusieurs sous-tâches, chacune confiée à une machine différente. Il est général bien moins coûteux d'avoir beaucoup de machines bon marché et relativement lentes plutôt qu'un seul supercalculateur. Contrairement à une grappe, les pairs sont typiquement très éloignés les uns des autres. Il existe aujourd'hui bien des systèmes de calcul distribué (comme Boinc) mais la plupart ne peuvent pas être considérés comme pair-à-pair et seraient mieux qualifiées de « maître-esclave ». En effet, en général, il existe un nœud central, qui décide des calculs à effectuer et les différentes machines lui obéissent, sans

tirer elle-mêmes profit du calcul fait (un exemple typique est SETI@home). Tous les systèmes compris sous l'appelation marketing de grille sont dans ce cas.

C'est également la même chose pour l'une des plus grosses applications du calcul distribué, les "*botnets*". Même lorsqu'ils reposent sur une technique pair-à-pair comme Storm, qui dépend d'une DHT, le zombie ne tire pas de profit des actions du "*botnet*", que ce soit l'envoi de spam ou bien la dDoS. Il ne fait qu'obéir aveuglément au C&C ("*Command and Control Center*", la ou les machines qui dirigent le "*botnet*".

La troisième grande catégorie, les applications de collaration, fait l'objet de la section 5.3. Les plus connues de ces applications sont celles de voix sur IP et de messagerie instantanée. Si la conversation elle-même se fait entre deux machines, le **rendez-vous** initial peut être réalisé en pair-à-pair, comme ce que permet P2PSIP (cf. section 2.3).

Bien, il y a donc de nombreuses applications qui utilisent le pair-à-pair. Mais est-ce à juste titre ? Dans quels domaines le pair-à-pair est-il fort et dans quels domaines vaut-il mieux s'en méfier ? C'est l'objet de la section 6 qui donne des critères d'évaluation des avantages et inconvénients du pair-à-pair. Notre RFC note qu'il n'existe pas de règle simple, la réponse à la question « Vaut-il mieux utiliser le pair-à-pair ? » dépend de beaucoup de choses. Parmi les points importants, le RFC note que les systèmes pair-à-pair sont particulièrement adaptés aux cas où il n'existe pas d'infrastructure existante et où il serait difficile de l'installer, ce qui est le cas des réseaux ad hoc.

Autre point fort du pair-à-pair, le passage à l'échelle. Avec lui, augmenter les ressources du système se fait simplement en ajoutant de nouveaux pairs. Si l'application envisagée doit pouvoir grossir jusqu'à la taille de l'Internet, le pair-à-pair est souvent une option à regarder sérieusement. (En fait, tout n'est pas toujours si simple et la fin d'OpenDHT <https://www.bortzmeyer.org/opendht-debut.html> a montré que le pair-à-pair ne se fait pas forcément tout seul.)

Si on est désireux d'avoir une application fiable (cf. RFC 4981), les systèmes pair-à-pair sont également intéressants. Si chaque pair individuel est en général considéré comme peu fiable, le système complet est conçu pour fonctionner alors même que les pairs individuels arrivent ou repartent du groupe en permanence. Il peut donc être très robuste.

Et les performances ? Il est difficile de dire si le pair-à-pair est plus rapide ou pas, cela dépend de nombreuses choses. Des « pique-assiettes » ("*free riders*"), qui profitent du système mais ne lui fournissent rien, peuvent sérieusement dégrader ces performances. D'où la recherche permanente de bons mécanismes d'incitation pour que les pairs ne se conduisent pas en parasites.

Un système pair-à-pair réel doit parfois choisir entre toutes ces caractéristiques. Le RFC cite l'exemple d'une base de données pair-à-pair : si toutes les données sont dupliquées sur tous les pairs, elle serait très robuste face aux pannes et très rapide en lecture. Mais l'arrivée d'un nouveau pair serait très lente parce qu'il devrait commencer par tout recopier. Et les écritures prendraient un temps fou.

Et les questions de sécurité ? Lorsque des tas de machines, parfois placées sous des administrations différentes, travaillent ensemble, ces problèmes de sécurité prennent une autre ampleur. La section 7 leur est consacrée. D'abord, toutes les machines du système sont-elles de confiance ? Certains systèmes pair-à-pair n'enrôlent pas n'importe qui mais uniquement des machines appartenant à la même entité. Dans ces conditions, le problème est plus facile à résoudre. Par contre, avec un système pair-à-pair ouvert à tout l'Internet, que n'importe quelle machine peut joindre quand elle veut, le problème devient un défi...

Dans tous les cas, la liste des problèmes de sécurité potentiels posés par les systèmes pair-à-pair est longue. Par exemple :

— Pour les systèmes qui ont un composant centralisé, une DoS réussie contre ce composant stoppe tout service,
— Pour les (nombreux) systèmes pair-à-pair où les messages ne sont pas transmis directement mais routés d'un pair à un autre jusqu'à la destination finale (cas de DHT comme Chord par exemple), les pairs intermédiaires ont des capacités de nuisance très importantes. Une attaque Sybil peut aboutir à ce qu'une grande partie des intermédiaires ne soient pas dignes de confiance,
— Si le protocole utilisé ne chiffre pas les messages, ceux-ci peuvent être écoutés par les routeurs IP mais aussi par les pairs intermédiaires (si on chiffre, il faut quand même laisser ceux-ci déchiffrer ce qui concerne le routage),
— Pour les systèmes pair-à-pair qui permettent le stockage des données, comme les DHT, un attaquant peut essayer d'épuiser les ressources. Un quota peut donc être nécessaire (ou une politique de nettoyage, comme le faisait OpenDHT <https://www.bortzmeyer.org/opendht-debut.html>),
— Si l'attaquant ne réussit pas à faire enrôler des machines à lui pour s'insérer dans le système, il peut, dans certains cas, subvertir le routage pour arriver à recevoir quand même les messages (attaque Eclipse, voir « *"Eclipse Attacks on Overlay Networks : Threats and Defenses"* <http://www.cs.rice.edu/~atuls/papers/eclipse-infocom06.pdf>).