

RFC 1122 : Requirements for Internet Hosts - Communication Layers

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 2 Juillet 2009

Date de publication du RFC : Octobre 1989

<http://www.bortzmeyer.org/1122.html>

Compagnon du RFC 1123¹, ce document normalise ce que toute machine terminale ("*host*", par opposition à routeur) connectée à l'Internet devrait savoir. Le RFC 1123 fixait les règles des « couches hautes » (notamment la couche application), notre RFC s'attaque aux « couches basses », notamment réseau et transport).

Le résultat est un document épais (116 pages), décrivant en détail tout ce qu'IP et TCP doivent faire sur une **machine terminale** (les routeurs sont, eux, traités dans le RFC 1812). La section 1.4 note que les discussions sur ce RFC avaient duré dix-huit mois et généré trois méga-octets de courrier, ce qui était considérable à l'époque mais est moins qu'un document MS-Word d'une page d'aujourd'hui.

Ce RFC n'avait pas pour but de changer les normes existantes (comme le RFC 791 pour IPv4 ou RFC 793 pour TCP) mais de les résumer et les préciser. Comme le note la section 1, « une mise en œuvre de bonne foi des protocoles TCP/IP, faite en lisant les normes, ne doit différer de ce RFC 1122 que par des détails ».

Aujourd'hui, plusieurs de ses sections sont bien dépassées mais il reste la norme officielle, personne n'a eu le courage de mettre ce travail à jour.

Il n'est donc pas possible de résumer tout le RFC et j'ai donc sélectionné arbitrairement quelques points qui me semblaient intéressants.

La section 1.1 rappelle l'architecture de l'Internet et la section 1.1.1 revient sur la notion de **machine terminale** ("*host*" dans le RFC). La section 1.1.2 note notamment que :

¹Pour voir le RFC de numéro NNN, <http://www.ietf.org/rfc/rfcNNN.txt>, par exemple <http://www.ietf.org/rfc/rfc1123.txt>

- L'Internet est un réseau de réseaux. Une machine terminale ne se connecte donc pas réellement à l'Internet, elle se connecte à un réseau connecté à Internet. Cette connexion se fait avec les mêmes protocoles, qu'on communique avec des machines du même réseau ou bien avec des machines lointaines. (La section 1.1.3 en donne une liste partielle, du protocole IPv4 pour les couches basses aux protocoles d'application comme telnet ou SMTP.)
- Les routeurs (le RFC utilise encore souvent le vieux terme de « passerelle » - "*gateway*" - qui n'est plus utilisé aujourd'hui que pour les engins travaillant dans la couche 7) n'ont pas de mémoire, ils routent chaque paquet indépendamment des autres. Les machines terminales ont donc à faire tout le travail d'établissement et de maintien des connexions.
- Le routage, par contre, ne doit être fait que par les routeurs. Il doit y avoir une stricte séparation entre machines terminales et routeurs. (Certains systèmes d'exploitation comme les Unix BSD rou-taient autrefois automatiquement dès qu'ils étaient connectés à deux réseaux. La section 1.1.4 explique pourquoi c'est une mauvaise idée. Voir aussi la discussion à la fin de la 3.3.4.2)

La section 1.2 pose les grands principes comme le célèbre Principe de Robustesse (section 1.2.2), « Soyez tolérant pour ce que vous recevez et exigeant pour ce que vous envoyez », principe qu'on trouve dans plusieurs autres RFC. Ainsi, un programmeur doit résister à la tentation d'exploiter les cas spécifiques d'une norme, pour éviter de perturber les autres implémentations.

La section 1.2.4 détaille la configuration des machines terminales. Elle se faisait entièrement à la main à l'époque. Aujourd'hui, avec la disponibilité fréquente de DHCP (RFC 2131), il vaut mieux oublier cette section et lire le dernier document sur la configuration, le RFC 5505.

Ensuite, le RFC suit le modèle en couches (section 1.3.1), ainsi que le vocabulaire spécifique de chaque couche pour désigner une unité de transmission de données sur le réseau ("*frame*" pour la couche 2, "*packet*" ou "*datagram*" - selon que la fragmentation aie eu lieu ou pas - pour la couche 3, "*message*" - "*segment*" pour TCP - pour la couche 4).

La section 2 concerne donc la couche de même rang. C'est ici que se trouve ARP (section 2.3.2, qui insiste que l'expiration des entrées du cache ARP est obligatoire, ce qui n'était pas assez clair dans le RFC 826). C'est aussi dans cette section (en 2.3.3) que l'on rappelle que l'encapsulation normale des paquets IP sur Ethernet est celle du RFC 894, où le troisième champ de l'en-tête Ethernet est le type du protocole de couche 3 (0x800 pour IPv4) pas celle, bien plus complexe, du RFC 1042, où la longueur de la trame se trouve à cet endroit.

Évidemment, la section 3, consacrée à la couche équivalente, est bien plus longue. Elle porte aussi son âge, par exemple en consacrant une section 3.2.1.3 aux classes d'adresse, supprimées depuis (<http://www.bortzmeyer.org/fini-les-classes.html>).

Parmi les nombreux points abordés, le choix du TTL à mettre dans les paquets (section 3.2.1.7). Une machine terminale ne doit pas émettre un paquet avec un TTL déjà à zéro. La mise en œuvre d'IP doit permettre aux applications de fixer le TTL (sur une machine Posix, cela se fait normalement avec `setsockopt()`, voir « Tester quels bits de l'en-tête IP on peut changer (<http://www.bortzmeyer.org/ip-header-set.html>) »).

Mais il y a aussi des exigences du RFC qui ne sont plus du tout suivies aujourd'hui. La section 3.2.1.8 obligeait toute mise en œuvre d'IP à permettre le routage par la source alors que, pour des raisons de sécurité, quasiment plus aucun routeur ne l'accepte.

ICMP (RFC 792) faisant conceptuellement partie de la couche 3, il a droit à une sous-section de la section 3. Elle rappelle par exemple qu'un paquet ICMP **d'erreur** ne doit jamais être envoyé lorsque le paquet original était lui-même un paquet ICMP d'erreur, pour éviter les boucles. (On lit souvent cette

règle énoncée comme « Un paquet ICMP ne doit jamais être envoyé en réponse à un paquet ICMP », ce qui est complètement faux, voir par exemple ping avec les ICMP echo, revus en section 3.2.2.6.)

Parmi les autres sujets abordés, la détection d'un routeur mort, qui ne transmet plus les paquets. Comme les machines terminales n'ont pas (ou plus) de protocole de routage (comme, par exemple, RIP), comment savent-elles que le routeur, en panne, est devenu un trou noir ? La méthode recommandée est, formellement, une violation du modèle en couches, mais c'est la seule qui marche dans tous les cas : les couches hautes doivent informer IP qu'elles ne reçoivent plus rien. (La section 3.3.1.4 est une passionnante discussion des autres options possibles. À lire par tout ingénieur qui s'intéresse aux réseaux.)

Le "*multihoming*", la connexion d'une machine (ou d'un réseau) à plusieurs réseaux, a toujours été un problème difficile pour IP. La section 3.3.4 tente de fixer certaines règles. Par exemple, lors d'une réponse, l'adresse IP source de la réponse doit être l'adresse IP de destination où a été envoyée la question. Ou encore, une application cliente doit avoir la possibilité de choisir son adresse IP source (directive `BindAddress` de OpenSSH ou bien `tcp_outgoing_address` de Squid). Et que faire en l'absence de telles directives ? C'est l'objet de la section 3.3.4.3 qui demande de privilégier, si possible, une adresse source située dans le même réseau physique que l'adresse de destination (notez que le RFC 3684 a depuis fourni une solution plus générale à cette question de sélection d'adresse source, pour le protocole IPv6). Par exemple, une machine choisira comme adresse IP source `127.0.0.1` si elle doit contacter `localhost` et son adresse globale si elle doit contacter une autre machine.

Au sommet de la couche 3 se trouve l'interface avec la couche 4. La section 3.4 détaille les services que doit rendre cette interface. Par exemple, la couche 3 **doit** fournir un moyen de fixer (pour un paquet émis) et d'interroger (pour un paquet reçu) des paramètres comme le TTL ou le TOS (depuis rebaptisé DSCP par le RFC 2474). Cette section est rédigée sous forme d'une API virtuelle (les API de programmation réseau réelles ont suivi un autre modèle, mais fournissent les mêmes services, cf. le livre de Stevens (<http://www.bortzmeyer.org/unix-network-programming.html>)). Cette API comporte des méthodes comme `SEND(src, dst, prot, TOS, TTL, BufPTR, len, Id, DF, opt)` (envoi d'un paquet en fixant chaque paramètre).

Finalement, un tableau synthétique, en section 3.5, résume tout ce que doit implémenter le programmeur qui réalise une mise en œuvre d'IP.

La couche 4 est ensuite traitée dans la section 4. Suivant le modèle en sablier (<http://www.iab.org/documents/docs/hourglass-london-ietf.pdf>), une machine connectée à Internet a un grand choix de couches 2 et de supports physiques, une seule couche 3, IP, et à nouveau un choix en couche 4. À l'époque de notre RFC, il n'y avait qu'UDP (RFC 768) et TCP (RFC 793). Depuis, plusieurs autres ont rejoint ces deux pionniers.

Parmi les exigences pour UDP, le RFC cite une forte recommandation d'activer la somme de contrôle (section 4.1.3.4). Celle-ci est en effet facultative en UDP et plusieurs protocoles ont souffert d'avoir cru qu'ils pouvaient s'en passer (notamment NFS et le DNS pour lequel la version 1 de Zonecheck (<http://www.zonecheck.fr/>) testait que cette somme de contrôle était bien activée).

TCP est bien plus riche et fait donc l'option d'une section nettement plus longue, la 4.2. À l'époque comme aujourd'hui, il est le protocole de transport le plus utilisé.

Le RFC commence (section 4.2.2) par une discussions sur le rôle des ports, en rappelant que la seule distinction normalisée entre les ports est celle entre ceux inférieurs à 255 (services normalisés) et les autres. La distinction entre les ports privilégiés (inférieurs à 1024) et les autres n'est pas normalisée, en pratique, elle est spécifique à Unix. Le RFC ne s'oppose pas à cette distinction mais note à juste titre

qu'elle ne vaut pas grand'chose en matière de sécurité puisque rien ne dit que la machine avec qui on correspond applique les mêmes règles. En outre, depuis la publication du RFC 1122, la vaste diffusion des ordinateurs fait qu'un éventuel attaquant a probablement sa propre machine, sur laquelle il est root. La très relative protection sur laquelle comptaient rlogin (en vérifiant que le port source était 513) et le DNS (qui utilisait à une époque lointaine un port source fixe, 53), ne vaut donc plus rien.

Il y a plein d'autres détails intéressants à couvrir comme l'option PUSH (section 4.2.2.2) dont le RFC rappelle qu'elle ne fournit pas un marqueur de fin de message mais qu'elle indique juste un souhait que les données soient transmises « le plus vite possible » (TCP attend normalement un peu pour voir si d'autres données n'arrivent pas, cette optimisation, l'algorithme de Nagle est à nouveau discutée en section 4.2.3.4).

Finalement, un tableau synthétique, en section 4.2.5, résume tout ce que doit implémenter le programmeur qui réalise une mise en œuvre de TCP.