RFC 6349 : Framework for TCP Throughput Testing

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 22 août 2011

Date de publication du RFC : Août 2011

https://www.bortzmeyer.org/6349.html

La mesure des performances du réseau intéresse évidemment fortement les utilisateurs (combien de temps me faudra-t-il pour télécharger justin-bieber.flac ?) et les décideurs qui cherchent à savoir, par exemple, quelle capacité offre un FAI, s'il délivre bien le service promis, ou encore s'il limite artificiellement les débits. La plupart des métriques définies rigoureusement, et des outils qui les mettent en œuvre, concernent des phénomènes de bas niveau, comme le taux de perte de paquets (RFC 7680 [1]), C'est ainsi que, lorsqu'il existe des SLA formalisés, ils portent en général sur ces phénomènes proches du réseau. Ces métriques sont utiles et relativement faciles à mesurer objectivement, mais elles sont très éloignées de l'expérience ressentie par l'utilisateur final <https://www.bortzmeyer.org/qos-ou-qoe.html>. Ce dernier voudrait plutôt savoir comment l'intégration de toutes ces métriques de bas niveau se traduit en un petit nombre de chiffres faciles à comprendre. Ce n'est pas une tâche triviale que de passer des métriques de base au ressenti utilisateur, et ce RFC est une première étape sur le chemin.

L'idée est de préciser les conditions de mesure de débit sur une connexion TCP, de bout en bout (donc tenant compte du comportement de tous les liens et équipements intermédiaires). Le débit mesuré en TCP donne une première idée de la capacité du réseau, qui est elle-même souvent la première métrique qui intéresse les utilisateurs (c'est en tout cas en général celle qui est mise en avant dans les publicités). Mais cette mesure contient de nombreux pièges et l'un des buts de ce RFC est de les lister. Au final, ce RFC ne permet pas de répondre simplement à la question de M. Toutlemonde « ça va moins ramer avec le fournisseur X qu'avec son concurrent Y ? », mais il avance quand même dans cette direction.

On grimpe donc d'une couche et on passe des mesures traditionnelles (taux de pertes, comme cité plus haut, ou bien capacité du lien au niveau 2) vers le niveau 4. Le désir de mesurer rigoureusement le débit atteignable avec TCP oblige à apporter quelques restrictions. Par exemple, le cadre de mesure décrit dans ce RFC se limite aux réseaux dont tous les aspects sont contrôlés par un fournisseur unique, avec des garanties explicites. Cela concerne donc plutôt la clientèle « entreprises ». La section 2 détaille tout ce qui n'est **pas** un objectif de ce RFC :

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc7680.txt

— Le comportement de TCP en dehors de l'équilibre (voir plus loin),
— La comparaison des mises en œuvre de TCP,
— La mesure en continu d'un réseau de production (pour cela, il vaut mieux utiliser la MIB TCP du RFC 4898).

Le but est au contraire de pouvoir mesurer, de manière objective et reproductible, les performances TCP d'un chemin donné, en fonction d'un certain nombre de réglages, bien définis. Les outils existants comme iperf ou ttcp permettent d'ajuster ces réglages.

Passons maintenant à la technique (la section 1.1 liste la terminologie employée) : TCP utilise à tout moment deux fenêtres, une pour l'envoyeur ("*congestion windows*") et une pour le récepteur ("*receive window*"). Le débit théorique est limité par la bande passante (capacité de la couche 2) et par le RTT. Leur produit, le BDP ("*Bandwidth*Delay Product*"), va déterminer la taille des tampons optimaux (le tableau en section 3.3.1 donne des valeurs numériques). Ce BDP est, en gros, le nombre d'octets qui peuvent se trouver en transit à un moment donné. Ensuite, la fenêtre de congestion sera ajustée pour tenir compte du taux de pertes. Le nombre d'octets en route à un moment donné dépendra de la fenêtre la plus petite (si on a une grande "*congestion window*" mais que l'autre partie annonce une petite "*receive window*", le débit restera faible : un émetteur TCP n'est pas censé noyer son pair sous les octets non désirés).

Toutes ces variables vont déterminer le débit effectivement atteint. Avant de tenter de mesurer celui-ci, le RFC insiste bien sur le fait que les tests de bas niveau (couches 2 et 3) doivent quand même être faits, pour déterminer que le réseau fonctionne bien. Le RFC note qu'un taux de pertes de paquets de 5 %, ou bien une gigue dans le délai d'acheminement de 150 ms, sont trop élevées pour que les mesures de débit TCP aient un sens. La méthodologie du RFC 2544 (bien que conçue à l'origine pour un environnement de laboratoire) peut être utilisée pour ces tests de bon fonctionnement.

Ensuite, notre méthodologie ne concerne que les connexions TCP ayant atteint l'état d'équilibre (section 1.2). Rappelons qu'une connexion TCP établie peut avoir trois comportements par rapport à la congestion : l'état d'équilibre où TCP essaie de remplir le tuyau au maximum, le démarrage en douceur, utilisé au début de la connexion, où TCP commence à sonder prudemment le réseau, en augmentant doucement son débit, et le troisième comportement, la récupération en cas de pertes, où TCP réduit brusquement son débit, avant de repartir plus ou moins doucement. En pratique, un certain nombre de connexions utilisées pour HTTP, par exemple, n'atteindront jamais cet équilibre, car elles durent trop peu de temps. C'est un bon exemple de la distance entre les mesures techniques et le vécu de l'utilisateur !

Dans un réseau idéal, à l'équilibre, TCP devrait avoir un débit très proche de la bande passante offerte.

La section 3 du RFC présente la méthodologie de mesure détaillée, avant que la section 4 ne décrive les métriques utilisées. La mesure impose d'utiliser un équipement capable de saturer le lien (donc, pas un vieux PC). Les mesures à plus de 100 Mb/s peuvent donc exiger une machine dédiée. La mesure exige ensuite les étapes suivantes :

— Déterminer la PMTU (en utilisant le RFC 4821, pour tenir compte des nombreux réseaux mal configurés qui bloquent ICMP) pour ensuite configurer le logiciel de test de manière à ce que les paquets soient assez petits pour éviter toute fragmentation (qui changerait drastiquement les performances). Par exemple, avec iperf (cité plus loin), l'option `--mss` permet de changer la taille des segments (des paquets TCP).
— Déterminer le RTT minimum et la bande passante offerte par les couches basses. Cela servira à optimiser des paramètres comme le tampon d'envoi. Sur un réseau de production, cette mesure doit se faire aux heures creuses, pour être sûr d'obtenir vraiment la valeur « réelle ». Il existe plusieurs façons de mesurer le RTT. Le choix dépend de la précision souhaitée. Si on se contente de la milli-seconde, les paquets ICMP echo de ping conviennent (mais c'est quand même la technique la moins fiable, surtout s'il y a un traitement différencié pour ICMP dans le réseau, ou dans

les machines de test). Si on veut se promener du côté de la micro-seconde, il faudra peut-être un dispositif de mesure spécialisé (voir section 5), et des protocoles comme celui du RFC 5357. Il y a aussi l'analyse des paquets capturés, avec un outil comme tcptrace, ou les statistiques gardées par TCP lui-même (RFC 4898). Quant à la mesure de la « bande passante » (un terme que je n'aime pas mais qui est utilisé par le RFC), elle devrait être faite suivant les règles du RFC 5136. Il ne faut pas oublier de la mesurer dans les deux directions, surtout avec les liens asymétriques comme l'ADSL.

— En enfin, faire les tests avec TCP, après avoir configuré le logiciel de test. La section 5 recommande un test d'au moins trente secondes, pour lisser les variations du réseau (avec iperf, c'est l'option `--time`, avec ipmt, c'est `-d`). Compte-tenu de la variété des mécanismes de tripotage du réseau qui existent, il peut être prudent de faire plusieurs mesures (par exemple de voir si une seule connexion obtient bien N fois ce qu'obtenait chacune des connexions d'un groupe de N connexions simultanées, cf. section 5.1).

Les métriques utilisées sont définies dans la section 4 du RFC. Elles sont au nombre de trois. La première est le **ratio du temps de transfert**, soit le rapport entre le temps nécessaire pour transférer N octets par rapport à la situation idéale (compte-tenu de la bande passante). Le calcul exact de cette situation idéale figure en section 4.1.1. Par exemple, pour une ligne T3 à 44,21 Mb/s exploitée en PPP, une MTU de 1500 octets, vus les en-têtes TCP, on arrive à 42,8 Mbps. Si on mesure un débit maximal de 40 Mbps, le ratio sera donc de 1,07 (rappelez-vous que c'est un rapport des temps de transfert, pas des débits). Pour un Ethernet 1 Gb/s, le cas idéal, qu'aucune amélioration du logiciel ne permettra de dépasser, est de 949 Mbps. L'utilisation de "*jumbo frames*" permettrait bien sûr de faire mieux.

La seconde métrique est l'**efficacité de TCP**. C'est le pourcentage des octets qui n'ont **pas** été retransmis. Il vaut donc 1 en l'absence de pertes (ou de retards qui mèneraient TCP à croire à une perte de paquets).

Plus subtile, la troisième métrique est le **retard dû aux tampons**. C'est le rapport entre l'excès de RTT (par rapport au RTT minimum, celui imposé par les caractéristiques de la ligne) et le RTT minimum. Par exemple, si le RTT minimum (qui, sur les lignes à grande distance, est souvent dépendant essentiellement de la vitesse de la lumière) est de 25 ms, et qu'on observe un RTT moyen de 32 ms pendant un transfert TCP, le retard dû aux tampons est donc de 28 %.

La première métrique est celle qui est la plus directement pertinente pour l'utilisateur. Les deux autres fournissent surtout un moyen de diagnostic : si le ratio du temps de transfert est mauvais, l'examen de l'efficacité et du retard dû aux tampons nous indiquera peut-être pourquoi.

Voilà, la norme est définie. La section 5, elle, revient sur les conditions pratiques de mesure. Entre autres, elle expose les règles du choix entre une seule connexion TCP et plusieurs. Ce choix dépend des caractéristiques du réseau (si le produit bande passante * délai est de 2 Moctets, une seule connexion TCP ne pourra pas remplir le tuyau, cf. le tableau des valeurs numériques en section 5.1).

Reste l'interprétation des résultats (section 5.2). Les résultats devraient toujours être accompagnés de la valeur des paramètres indiqués plus haut (ce que font les outils existants, cf. plus loin). Le RFC demande évidemment que les valeurs des trois métriques citées plus haut soient affichées, ce qui n'est pas (encore ?) le cas avec les logiciels Unix de mesure existants. Une fois qu'on a ces valeurs, si elles ne correspondent pas aux valeurs idéales, on peut se livrer aux joies du débogage : la congestion avec des tampons de petite taille dans les routeurs fera sans doute grimper les pertes de paquets, ce qui se traduira par une mauvaise efficacité TCP. Si les routeurs ont de quoi stocker les paquets en attente, c'est le RTT qui augmentera, ce qui se verra dans le retard dû aux tampons. Mais il peut aussi y avoir ralentissement délibéré par le réseau ("*policing*"), tampons trop petits sur les machines de test (qui ont, après tout, une mémoire limitée, qui peut être trop faible pour les réseaux à haute performance d'aujourd'hui). Les machines Unix ont de tels tampons aussi bien globalement pour tout le système que par "*socket*".

On peut fixer ces derniers (la valeur idéale est le BDP, le produit bande passante*délai) dans le programme, mais aussi les laisser s'ajuster automatiquement (service "*auto-tuning*", qui existe dans Linux et FreeBSD ainsi que, depuis moins longtemps, Windows et Mac OS). Il faut également regarder du côté du mécanisme de "*window scaling*" (RFC 7323) sans lequel il est impossible d'exploiter les réseaux à très fort BDP qu'on trouve aujourd'hui. Et, bien sûr, émettre des paquets ayant la taille maximale possible sans fragmentation est aussi une bonne tactique.

Bien d'autres réglages de TCP sont possibles, comme l'option d'estampille temporelle (RFC 7323) qui permet à TCP de mieux calculer le RTT (et qui protège contre les numéros de séquence TCP qui reviennent à leur valeur initiale, un problème qui devient sérieux au delà de 100 Mb/s), les accusés de réception spécifiques (SACK) du RFC 2018, qui permettent de limiter les retransmissions, etc.

À titre d'exemple, voici des exemples de mesure, plus ou moins compatibles avec notre RFC, dans le cas de certains outils libres. Le premier est iperf. Existant sous forme de paquetage pour divers systèmes Unix (ici, Debian), il est facile à installer. On doit le lancer côté client et côté serveur (c'est un point commun entre tous ces outils : il faut un compte sur les deux machines ; on ne peut donc pas les utiliser pour tester son débit possible avec YouTube).

```
# Serveur
% iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)

# Client
% iperf --print_mss --client test.bortzmeyer.org
------------------------------------------------------------
Client connecting to test.bortzmeyer.org, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 192.0.2.69 port 46908 connected with 203.0.113.232 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec    173 MBytes    145 Mbits/sec
[  3] MSS size 1448 bytes (MTU 1500 bytes, ethernet)
```

On peut aussi changer la taille de la fenêtre TCP. Les 145 Mb/s obtenus ci-dessus sont probablement indépassables mais, parfois, ce changement améliore les performances. Il peut aussi les dégrader, ici, on choisit une fenêtre très petite :

```
% iperf --window 1K --client test.bortzmeyer.org
...
TCP window size: 2.00 KByte (WARNING: requested 1.00 KByte)
------------------------------------------------------------
...
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  27.3 MBytes  22.9 Mbits/sec
```

Cet exemple illustre l'importance de documenter, lorsqu'on publie les résultats de la mesure, les options utilisées. Ici, on a divisé la capacité utile par sept... (Quant au message d'avertissement, il vient de l'habitude du noyau Linux de fixer la taille de la fenêtre à une valeur double de celle qui est demandée.) iperf permet théoriquement de faire fonctionner plusieurs sessions TCP simultanées (option `--parallel`) mais j'avoue n'avoir pas réussi à la faire fonctionner.

Un autre outil traditionnel est ttcp. Très ancien, il a depuis été remplacé par des versions plus récentes. Je vais utiliser nuttcp, également en paquetage dans tous les bons systèmes. Son serveur a la désagréable habitude de passer en arrière-plan immédiatement (et il ne semble pas y avoir d'option pour changer cela), et on risque donc facilement d'oublier un serveur qui tourne. Le mieux est donc de toujours le lancer avec l'option `-1` qui ne permet qu'un seul test.

```
# Serveur
% nuttcp -1

# Client
% nuttcp -v -t 203.0.113.232
nuttcp-t: v6.1.2: socket
nuttcp-t: buflen=65536, nstream=1, port=5001 tcp -> 203.0.113.232
nuttcp-t: time limit = 10.00 seconds
nuttcp-t: connect to 203.0.113.232 with mss=1448, RTT=2.173 ms
nuttcp-t: send window size = 16384, receive window size = 87380
nuttcp-t: available send window = 12288, available receive window = 65535
nuttcp-t: 166.5708 MB in 10.00 real seconds = 17056.72 KB/sec = 139.7286 Mbps
nuttcp-t: retrans = 0
nuttcp-t: 2666 I/O calls, msec/call = 3.84, calls/sec = 266.60
nuttcp-t: 0.0user 0.0sys 0:10real 0% 0i+0d 592maxrss 0+2pf 893+0csw
```

On note des chiffres proches de ceux de iperf, ce qui est rassurant. nuttcp a le même genre d'options que iperf (`-l` pour changer la taille des tampons, `-w` pour la fenêtre TCP, etc). Il peut utiliser plusieurs sessions TCP (option `-N`) mais ne semble pas permettre de fixer la MSS.

Un outil plus récent, encore un peu expérimental mais qui a bien marché sur mes machines est ipmt <http://ligforge.imag.fr/frs/?group_id=140>. On lance le serveur ainsi :

```
% tcptarget
IPv6 (and IPv4) protocol
Using port 13000
...
```

Et le client avec :

```
% ./tcpmt test2.bortzmeyer.org
IPv4 protocol

Time         Packets    Total      |    Kbit/s  Avg 10  Avg
33496.429     486       486        |    5045    5045    5045
33497.345     179       665        |    2283    3805    3805
33498.467     134       799        |    1394    2950    2950
...
```

et les résultats sont affichés en continu (utile si le réseau change).

Enfin, le traditionnel Netpipe <http://www.scl.ameslab.gov/netpipe/> ne semble plus guère maintenu et fonctionne mal sur les Unix récents.

Le RFC mentionne aussi d'autres outils comme tcptrace, outil d'analyse des sessions capturées au format pcap et qui, avec ses options `-r` et `-l`, permet de calculer des choses comme le RTT.

Un petit avertissement de sécurité, pour finir (section 6). Les mesures de ce RFC sont des mesures actives, susceptibles de perturber sérieusement le réseau. Le principe étant d'occuper **tout** le tuyau, la mesure ressemble fort à une DoS. Soyez donc prudent (RFC 4656 et RFC 5357).