RFC 793 : Transmission Control Protocol

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 29 septembre 2006. Dernière mise à jour le 3 mars 2008

Date de publication du RFC : Septembre 1981

https://www.bortzmeyer.org/793.html

Vingt-cinq ans ce mois-ci que le protocole TCP est le protocole de transport le plus répandu de l'Internet. Presque toutes les applications l'utilisent, à commencer par le Web. Il n'a finalement été remplacé par un nouveau RFC qu'en août 2022, avec le RFC 9293 [1].

Spécifier complètement un protocole de transport n'est pas trivial. Contrairement à IP, décrit dans le RFC 791, TCP a un état et chaque paquet doit donc être traité dans le contexte de cet état. Si la machine à états de TCP est relativement simple, il faut quand même 64 pages au RFC pour décrire tous les détails.

L'aventure de TCP est loin d'être finie : le passage à IPv6 ne l'affectera que peu, et, si le RFC 7414 liste de nombreux RFC qui améliorent ou corrigent la spécification initiale, notre RFC d'un quart de siècle est toujours d'actualité.

La machine à états de TCP :

Voici un paquet TCP tel que vu par Wireshark (lancé en mode texte avec `tshark -V`) lors de l'établissement d'une connexion. Le format du paquet TCP est décrit dans la section 3.1 de notre RFC. On note la notion de port (ici le port 7 pour le protocole echo du RFC 862) ; IP ne permettant que d'identifier qu'une machine, les ports servent à distinguer plusieurs processus sur une machine. On y voir le bit de contrôle SYN ("*SYNchronize*", demande d'établissement de connexion).

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc9293.txt`

```
Transmission Control Protocol, Src Port: 36682 (36682), Dst Port: 7 (7), Seq: 0, Ack: 0, Len: 0
    Source port: 36682 (36682)
    Destination port: 7 (7)
    Sequence number: 0    (relative sequence number)
    Header length: 40 bytes
    Flags: 0x0002 (SYN)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set
        ..0. .... = Urgent: Not set
        ...0 .... = Acknowledgment: Not set
        .... 0... = Push: Not set
        .... .0.. = Reset: Not set
        .... ..1. = Syn: Set
        .... ...0 = Fin: Not set
    Window size: 5840
    Checksum: 0xa57b (correct)
    Options: (20 bytes)
        Maximum segment size: 1460 bytes
        SACK permitted
        Time stamp: tsval 48358541, tsecr 0
        NOP
        Window scale: 0 (multiply by 1)
```

On note surtout l'important **numéro de séquence**. TCP gère des flux d'octets numérotés et les accusés de réception indiquent un numéro de séquence, pas un paquet. Ce numéro est affiché comme zéro par Wireshark mais c'est un numéro relatif. Dans la réalité, sur le câble, ce numéro est initialisé à une valeur tirée au hasard (il est important de la tirer au hasard pour limiter les risques qu'une session TCP soit détournée par un attaquant). tcpdump, lui, affiche la valeur absolue pour les paquets SYN, la valeur qui passe sur le câble. Il affiche une valeur relative ensuite. Si on préfère avoir tout le temps la valeur absolue, l'option `-S` est faite pour cela. Voici un exemple d'ouverture de connexion (vers le serveur Subversion de CNP3 <`https://www.bortzmeyer.org/cnp3.html`>) avec les options par défaut :

```
09:18:08.880944 IP6 2001:67c:2219:8::6:69.59834 > 2001:6a8:3080:1::4.443: Flags [S], seq 4131334602, win 144
09:18:08.908511 IP6 2001:6a8:3080:1::4.443 > 2001:67c:2219:8::6:69.59834: Flags [S.], seq 3086055825, ack 41
09:18:08.908534 IP6 2001:67c:2219:8::6:69.59834 > 2001:6a8:3080:1::4.443: Flags [.], ack 1, win 900, options
```

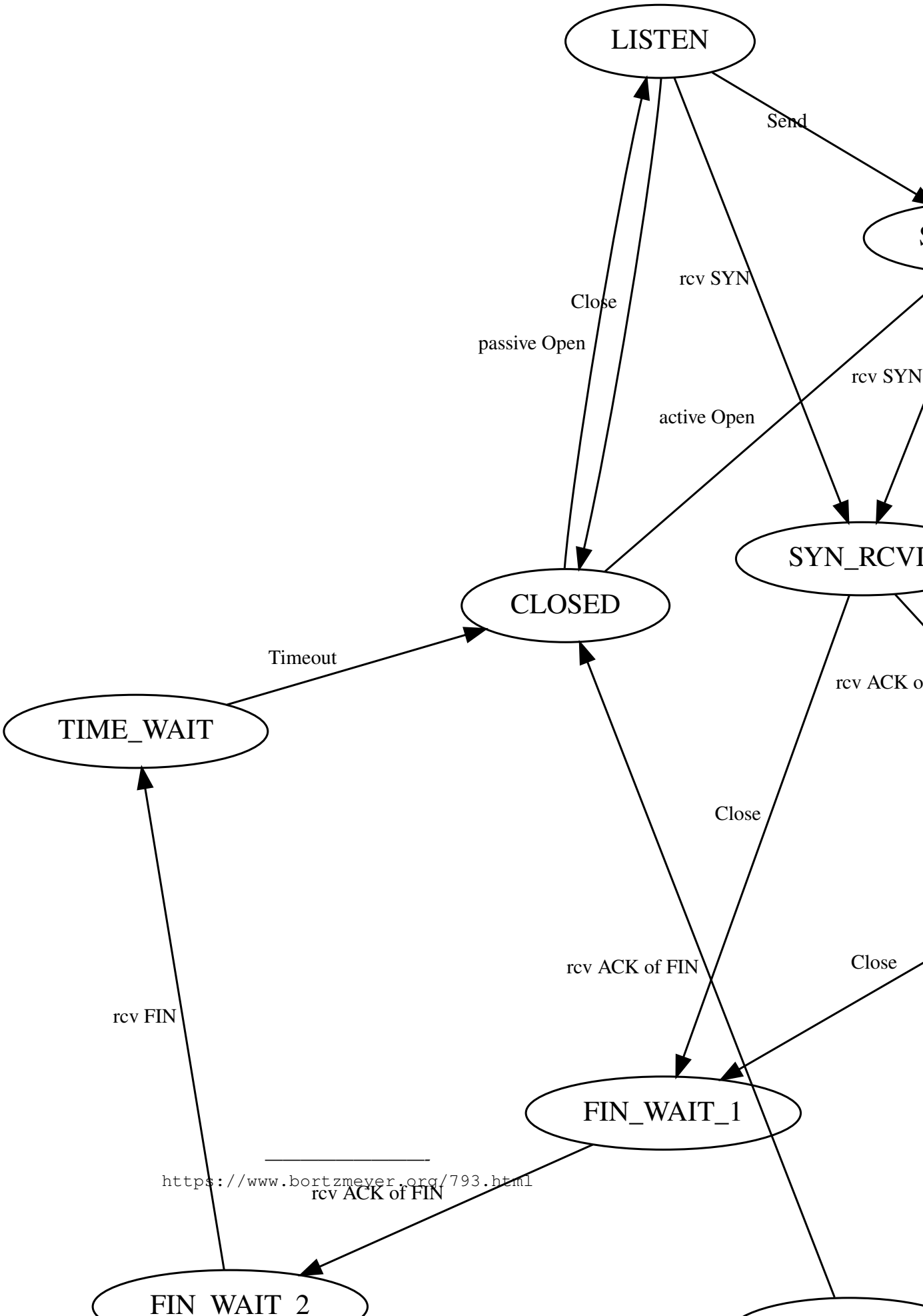
Dans le troisième paquet, l'accusé de réception de l'initiateur de la connexion, tcpdump a affiché qu'on accusait réception, et que le prochain octet attendu était le 1 (pas encore envoyé, puisque les numéros de séquence indiquent l'octet suivant). Avec `-S` :

```
09:18:08.880944 IP6 2001:67c:2219:8::6:69.59834 > 2001:6a8:3080:1::4.443: Flags [S], seq 4131334602, win 144
09:18:08.908511 IP6 2001:6a8:3080:1::4.443 > 2001:67c:2219:8::6:69.59834: Flags [S.], seq 3086055825, ack 41
09:18:08.908534 IP6 2001:67c:2219:8::6:69.59834 > 2001:6a8:3080:1::4.443: Flags [.], ack 3086055826, win 900
```

Les deux premières lignes sont inchangées, mais on voit désormais que le troisième paquet contenait en fait un accusé de réception attendant l'octet 3086055826 (puisque le numéro de séquence initial choisi par `2001:6a8:3080:1::4` était 3086055825).

À la fin de l'en-tête TCP, Wireshark montre les options, un champ de taille variable qui contient différentes valeurs, encodées selon un schéma compliqué (l'option peut faire un seul octet ou bien être composée d'un triplet Type-Longueur-Valeur). On y trouve ici la MSS (ici 1460 octets), définie dans notre RFC, et des options qui ont été ajoutées ultérieurement, comme SACK ("*Selective ACKnowledgment*", RFC 2018), ou bien "*Window scale*", RFC 7323.

On peut noter que les options de TCP, étant une famille qui s'agrandit régulièrement, posent parfois des problèmes à des coupe-feux mal conçus (probablement la majorité), qui jettent sans scrupule les paquets contenant des options qu'ils ignorent. Ainsi, il est fréquent que les petits « routeurs » installés par les FAI chez leurs clients, aient des problèmes avec certaines options TCP. À l'heure où j'écris ce texte, le "*window scaling*", pourtant normalisé il y a seize ans, ne passe pas les routeurs d'Alice/Tiscali (sur Linux, il faut mettre la variable sysctl `net.ipv4.tcp_default_win_scale` à 0). Et le SACK ne passe pas chez Noos (il faut mettre `net.inet.tcp.sack.enable` à 0 sur une FreeBSD).


LISTEN
Send
rcv SYN
Close
passive Open
rcv SYN
active Open
SYN_RCVI
CLOSED
Timeout
rcv ACK o
TIME_WAIT
Close
rcv ACK of FIN
Close
rcv FIN
FIN_WAIT_1
rcv ACK of FIN
FIN_WAIT_2