IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Cybersécurité des systèmes maritimes et portuaires

La couche transport

Christophe Lohr

2021

1 Introduction

Les protocoles de transport au dessus de IP 3/44

- TCP : Transmission Control Protocol rfc793
  - Transport en mode «connecté»
  - Contrôle de flux et détection d'erreur avec retransmission
- UDP : User Datagram Protocol rfc763
  - Mode datagramme
  - Pas de contrôles, pas d'assurance de délivrance
- Nouveaux
  - DCCP Datagram Congestion Control Protocol rfc4340 2006
  - SCTP Stream Control Transmission Protocol rfc4960 2007
  - MPTCP Multipath TCP rfc6826 2013
  - QUIC A UDP-Based Multiplexed and Secure Transport draft

TCP et UDP, deux besoins extrèmes, mais relativement faciles à utiliser. De nombreux protocoles "universitaires" proposent des compromis, des services intermédiaires (p-ex. le protocole POC : Partial Order Connection), mais certains commencent à émerger sérieusement comme DCCP et SCTP.

DCCP : un genre d'UDP mais en mode connecté et avec contrôle de congestion. Pas de préservation de l'ordre des message ni de garantie de transmission des donnée, mais garantie sur les acquittement.

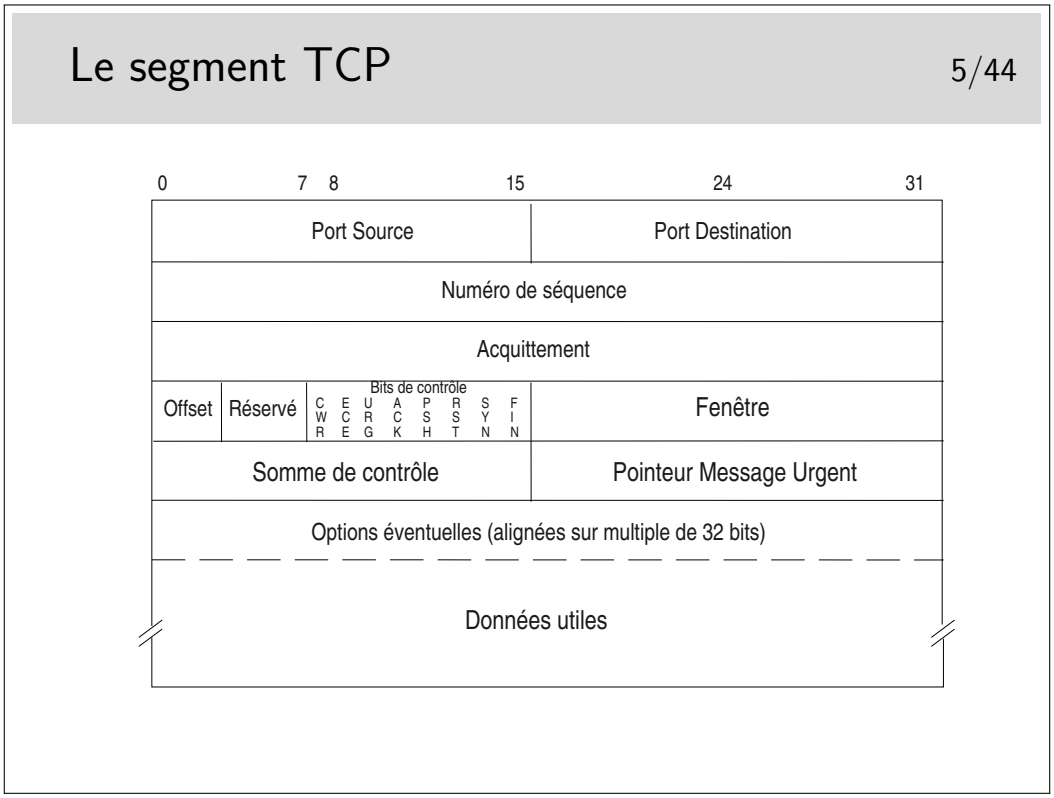
SCTP : un genre de TCP mais orienté *flux de messages* et non pas flux d'octets (flux : préserve l'ordre et guarantie la transmission) ; gère le multi-flux au sein d'une même connection, et le *multihoming* (annoncer que l'on va changer d'adresse IP pour la suite de la session).

MPTCP : compatible avec le TCP classique (programme avec des sockets TCP) ; définit des nouvelles *options* pour annoncer des sous-flux, en parallèle ou en secours, rejoindre une autre IP sans clore la session (*multihoming*), etc.

QUIC : connexion "comme TCP", mais multi-flux (plusieurs conversations dans une même connexion), chiffré de bout en bout façon TLS, et encapsulé dans de l'UDP pour pouvoir passer

les équipements legacy intermédiaires. À l'origine développé par Google vers 2012, toujours en discussion à l'IETF.

2 TCP



Ports source et destination : identifient les applications en relation, généralement l'une d'elle est serveur, son port correspond alors au numéro du protocole (exemple : 80 pour les serveurs Web)

Numéro de séquence : numéro du premier octet des données. C'est le rang du premier octet véhiculé par ce segment en comptant depuis le début de l'échange. Le numéro de début est tiré aléatoirement entre 0 et 232-1.

Acquittement : numéro du prochain octet attendu

Offset : indique la longueur de l'entête en mots de 32 bits (si égal à 5 alors pas d'option)

Bits de contrôle (CWR, ECE, Urg, Ack, Psh, Rst, Syn, Fin)

Fenêtre : permet le contrôle de flux, le récepteur indique avec ce champ combien d'octets il est prêt à recevoir (une valeur de 0 indique que ses tampons mémoire sont pleins)

Somme de contrôle : permet de savoir si le segment a été altéré ou non pendant sa transmission

Pointeur de données urgentes : indique l'emplacement de ces données dans le segment (si le bit Urg est positionné, sinon ce champ est ignoré, bien qu'il existe toujours)

Les bits de contrôle I

CWR	ECE	URG	ACK	PSH	RST	SYN	FIN

- ECE Explicit Congestion Notification-Echo (rfc 3168), l'entité TCP recevant un segment contenant ce bit à 1 doit réduire son débit
- CWR Congestion Window Reduced (rfc 3168) réponse à la réception du bit ECE pour indiquer que la requête a bien été prise en compte
- URG indique que le champ *Urgent Pointer* est significatif.
- ACK indique que le champ *Acknowledgement Number* est significatif.
- PSH fonction *PUSH*. Les données doivent être immédiatement remises à la couche supérieure.
- RST *reset*, la connexion est rompue.
- SYN synchronisation des indicateurs numéro de séquence. Segments d'initialisation de connexion.
- FIN terminaison de connexion.

Les bits de contrôle II

Exemple : segment IP et TCP dont le bit ACK est à 1

```
 0: 0800 2074 ef05 0800 0914 18e7 0800 4500
16: 0028 8954 0000 4006 db07 c02c 4b13 c02c
32: 4b08 1770 fdbe 0162 6e86 8e21 a873 5010 /* 0001 0000 */
48: 2210 bba3 0000 023a b3a1 1829
```

Les données «urgentes» 8/44

- Données appelées parfois "hors bande" ou *Out of Band* (OOB)
- Données à traiter en priorité par la couche réceptrice
- Elles sont véhiculées dans le flux normal en suivant le chemin normal. IP n'est pas sensible à ces données, leur caractère "urgent" est significatif seulement aux extrémités
- L'arrivée de ces données a un caractère aléatoire pour les applications destinatrices
- Les applications ne lisent pas ces données dans le flux normal
- Une application devant pouvoir accepter de telles données doit avertir le système pour que celui-ci lui envoie une interruption (un signal logiciel) afin qu'elle puisse traiter en priorité la donnée. L'application doit prévoir une routine spéciale de traitement pour la lecture de ces données
- Sémantique mal définie : le RFC6093 recommande aux nouvelles application ne ne plus l'utiliser...

Le bit de contrôle RST 9/44

- Utilisé par une entité TCP connectée avec une autre entité TCP distante pour avertir d'un problème
- Une application se terminant normalement fait une fermeture sur le port TCP utilisé (souvent une *socket*), ceci se concrétise par un échange de segments avec le bit FIN positionné et la connexion est rompue
- Si l'application se termine brutalement sans fermer la connexion, l'entité TCP associé envoie vers l'autre extrémité un segment avec le bit RST positionné. L'autre extrémité est ainsi prévenue de la terminaison de la connexion

TCP est orienté connexion 10/44

- La connexion TCP :
  - Relation établie point à point entre les deux extrémités
  - Normalement transparente aux routeurs (sauf si ceux-ci mettent en œuvre du filtrage où des mécanismes de QoS)
  - Caractérisée par un contexte dans les machines d'extrémité (numéros IP local et distant, ports local et distant)

La «connexion» TCP I 11/44

- La connexion est établie par un échange de paquets initiaux : le *three way handshake*
  - Ce n'est pas une connexion au sens strict du terme, il n'y a pas de chemin virtuel établi dans le réseau, les segments TCP sont portés par IP, protocole orienté datagramme
  - La connexion TCP se traduit par un contexte mémorisé dans les machines d'extrémité (le client et le serveur), ce contexte est le quintuplet :

```
[protocole, port local, port distant, @IP locale, @IP distante]
```

La «connexion» TCP II 12/44

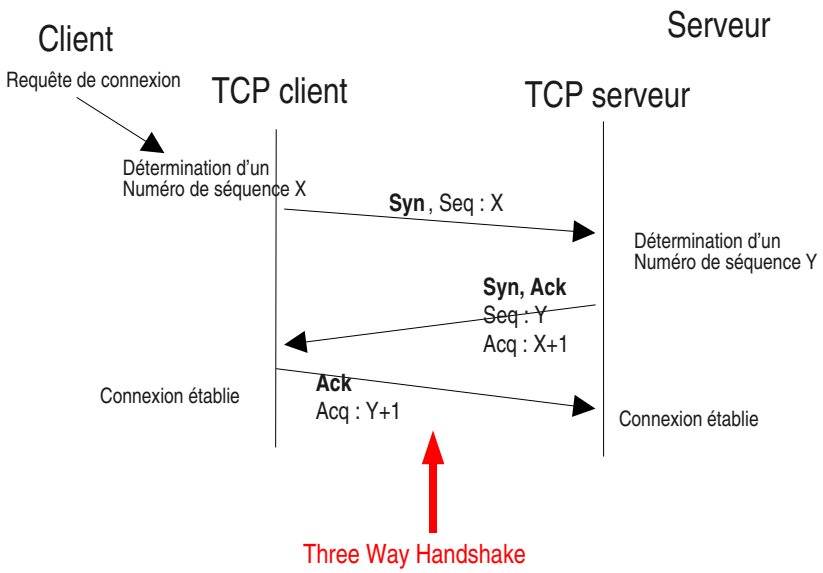
- Si les applications connectées n'ont rien à se dire, les entités TCP correspondantes ne s'envoient aucun segment, il n'y a plus alors de trafic
- Il est possible de faire en sorte que les entités s'échangent des segments d'alerte (sans données utiles puisqu'il n'y en a pas à envoyer) toutes les deux heures en positionnant une option dans TCP
- Si l'entité TCP distante ne répond pas au segment d'alerte ceci signifie que l'application associée s'est terminée sans prévenir ou que la machine s'est arrêtée. L'application locale sera prévenue lors de la prochaine lecture du port TCP

Cette option n'est pas une option au sens dans lequel ce mot sera vu plus loin. Il s'agit simplement d'un paramètre interne à l'entité TCP, positionné par l'application à l'aide d'une fonction spécifique (`setsockopt(SO_KEEPALIVE)`).

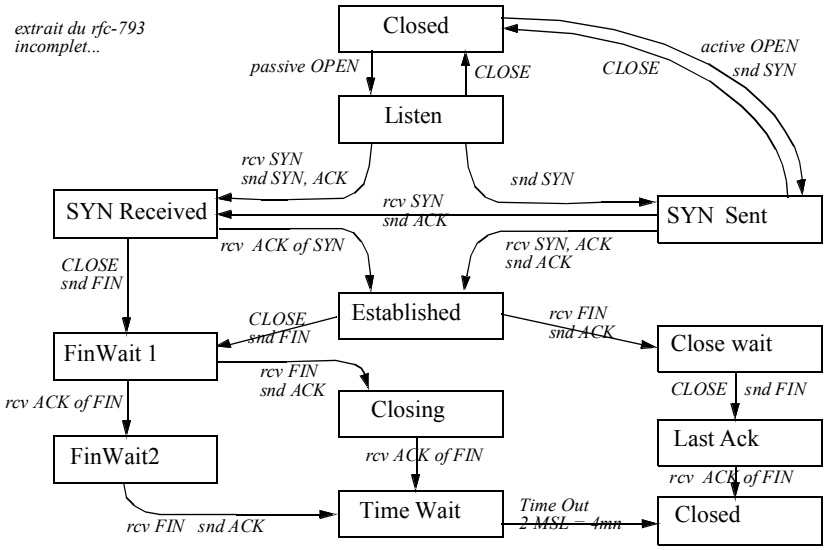
Connexions actives et passives 13/44

- Connexion passive
  - Ce n'est pas une connexion mais un point d'accès TCP ouvert par une application en fonction de serveur
  - Le contexte TCP créé attend une requête de connexion émanant d'un client
- Connexion active
  - La connexion réelle, initialisée par une application en mode client

Établissement de la connexion 14/44

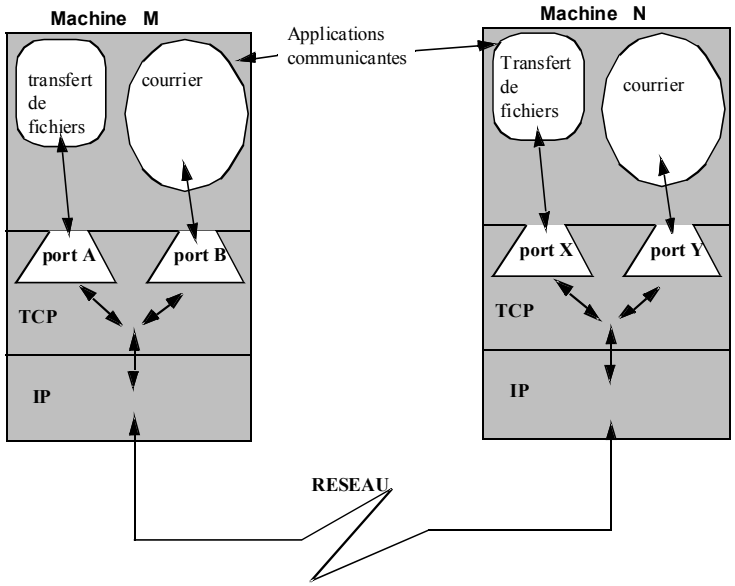


La machine d'états TCP 15/44



Ces états sont visualisables sous Windows dans une fenêtre de commande avec la commande `netstat -p tcp`. Sous Linux on fera `netstat -at`.

La notion de port 16/44



Les ports sont, en quelque sorte, les adresses des applications à l'intérieur des machines. Selon la terminologie OSI, un port est un SAP.

Quelques ports bien connus (voir aussi le fichier `/etc/services` sous Unix/Linux, ou `C:\WINNT\system32\drivers\etc\services` sous Windows)

21 : ftp (le port du canal de commande de ftp)

20 : ftp-data (le port pour la phase effective du transfert de fichiers par ftp)

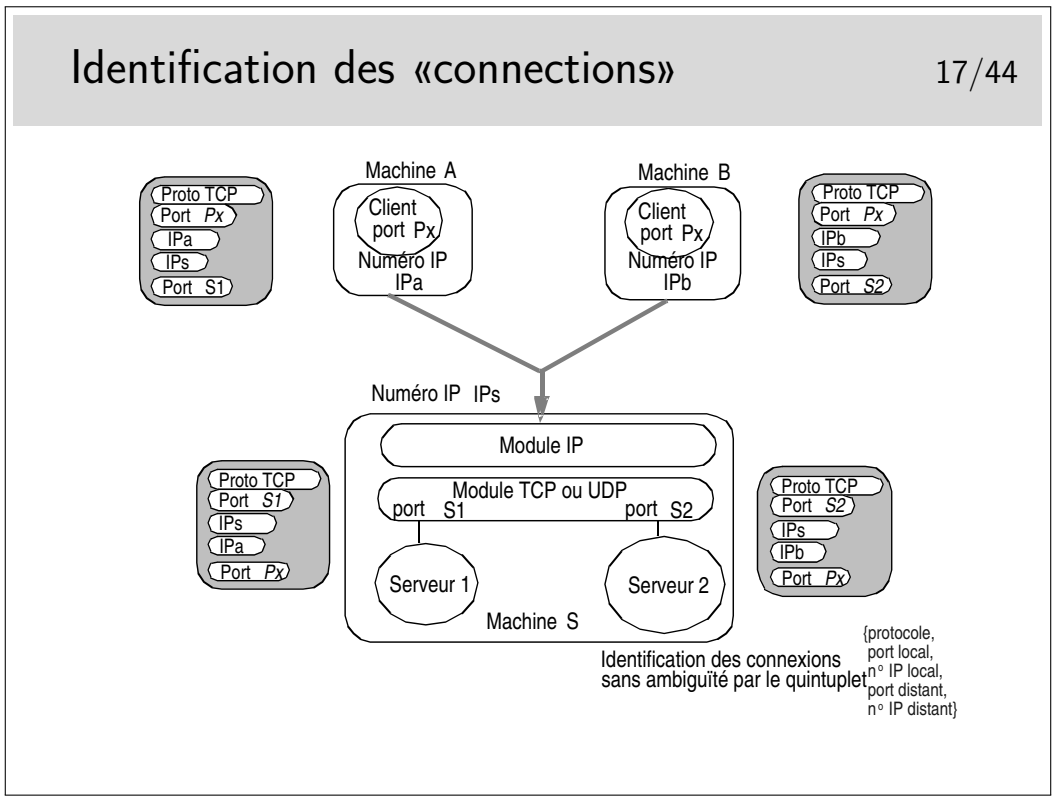
22 : ssh (les connexion à distance sécurisée par chiffrement)

23 : telnet : la même chose mais sans chiffrement

25 : le protocole d'envoie du courrier électronique Internet, SMTP

80 : le protocole http, donc le WEB

etc...



Pour résumer ce qu'est une «connexion TCP» :

— c'est une relation entre deux entités TCP résidant sur des machines différentes (ou entre deux contextes de la même couche TCP d'une machine où s'exécutent les applications en communication). En terminologie OSI, dans la couche Application on utilise le mot «association» plutôt que «connexion». Il serait plus approprié ici aussi.

— il n'y a pas de connexion associée dans le réseau

— dans chaque entité TCP en relation un contexte est créé et est associé à l'application ne cause de son coté

— ce contexte comprend :

  — une machine d'état finis gérant les états de la connexion

  — un quintuplet d'identification de connexion

  — des tampons mémoire d'émission et de réception

  — diverses variables complémentaires comme, entre-autre, le compteur associé à la gestion des congestions

— C'est l'application qui demande la création de ce contexte, à l'aide de fonctions d'une API spécifique telle que les Sockets (API issue du monde Unix, BSD au départ et ensuite reprise par tous les Unix et portée sous Windows)

Les options TCP I

- Caractéristiques
  - Le champ offset de l'entête étant codé sur 4 bits (il indique le nombre de mots de 32 bits de l'entête), la longueur max de l'entête est de 15 mots de 32 bits (15dec = 1111bin). L'entête minimum étant de 5 mots de 32 bits (20 octets), la longueur du champ option est de 10 mots de 32 bits max.

Les options TCP II

- Options standards
  - mss : *maximum segment size*, permet de négocier la taille maximale des segments TCP afin d'éviter des segmentations coûteuses. cette option est utilisées au moment de la connexion. Longueur 4 octets.
  - pas d'opération (NOP) : option nulle, permet d'aligner la prochaine option sur un début de mot de 32 bits. Plusieurs NOP peuvent se suivre si besoin. (lg : 1 octet)
  - fin de liste : permet d'aligner la fin des options sur un mot de 32 bits (lg 1 octet)

Les options TCP III

- Multiplicateur du champ fenêtre (window scale)
  - valeur permettant de multiplier la fenêtre par un multiple d'une puissance de 2 (jusqu'à 2^{14}) (rfc 1323). Ceci permet d'utiliser plus efficacement la bande passante pour des sources à haut débit et/ou des communications à grande distance en permettant un flux le plus continu possible
- Horodatage des données (rfc 1323)
  - L'option contient deux valeurs : un indicateur d'heure d'émission et un acquittement. Une source peut ainsi calculer le temps d'aller et retour sur un chemin en plaçant sa valeur d'horloge dans le premier champ. Lorsque l'acquittement du segment correspondant arrivera, cette valeur sera contenue dans le second champ de cette option. Il sera alors facile de retrancher cette valeur de la valeur courante de l'horloge.

Les options TCP IV

- Négociation de l'acquittement sélectif (rfc 2018)
- Acquittement sélectif : la première option permet d'indiquer qu'une source TCP est capable d'utiliser cette fonctionnalité, la seconde met en œuvre ce type d'acquittement permettant de demander uniquement la retransmission de données perdues en évitant de retransmettre des données suivantes bien reçues

Capture d'une séquence TCP 22/44

rfc-1323 et rfc 2018 (relevé avec tcpdump lors d'un début de requête web)

```
linux1.1082 > 206.132.41.203.www: S 2047350264:2047350264(0) win 16060 <mss
1460,sackOK,timestamp 43244276>
206.132.41.203.www > linux1.1082: S 2319802134:2319802134(0) ack 2047350265 win
32120 <mss 1460, sackOK, timestamp 190973015>
linux1.1082 > 206.132.41.203.www: . ack 2319802135 win 16060 <nop,nop,timestamp
43244311 190973015>
linux1.1082 > 206.132.41.203.www: P 2047350265:2047350896(631) ack 2319802135 win
16060 <nop, nop, timestamp 43244311 190973015>
206.132.41.203.www > linux1.1082: . ack 2047350896 win 31856 <nop, nop, timestamp
190973051 43244311>)
206.132.41.203.www > linux1.1082: P 2319802135:2319803583(1448) ack 2047350896 win
31856 <nop, nop, timestamp 190973063 43244311>
linux1.1082 > 206.132.41.203.www: . ack 2319803583 win 14612 <nop,nop,timestamp
43244367 190973063>
206.132.41.203.www > linux1.1082: P 2319803583:2319805031(1448) ack 2047350896 win
31856 <nop, nop, timestamp 190973063 43244311>
```

S : bit Syn

ack : bit ack

P : bit Push

. : pas de flag (autre que ack)

sackOK : acquittements sélectifs en fonction

mss : maximum segment size

en italique : complément d'information fourni par tcpdump mais ne figurant pas réellement dans le paquet

numéro de séquence du dernier octet, correspondant à l'acquittement alors attendu. Entre parenthèses : nombre d'octets du paquet

Les trois premières trames correspondent à une ouverture de connexion (Three Way Handshake)

Le contrôle des connexions : la commande `netstat` 23/44

- La commande `netstat` avec l'option `-a` (sous Unix/linux, ou `-p tcp` sous Windows)

```
[linux30]$ netstat -atn
Connexions Internet actives (serveurs et établies)
 Proto   Recv-Q   Send-Q   Adresse locale      Adresse distante         Etat
 tcp          0        0   0.0.0.0:32768       0.0.0.0:*               LISTEN
 tcp          0        0   127.0.0.1:32769     0.0.0.0:*               LISTEN
 tcp          0        0   0.0.0.0:111         0.0.0.0:*               LISTEN
 tcp          0        0   0.0.0.0:6000        0.0.0.0:*               LISTEN
 tcp          0        0   0.0.0.0:22          0.0.0.0:*               LISTEN
 tcp          0        0   0.0.0.0:631         0.0.0.0:*               LISTEN
 tcp          0      272   192.168.100.30:22   192.168.100.18:1994   ESTABLISHED
```

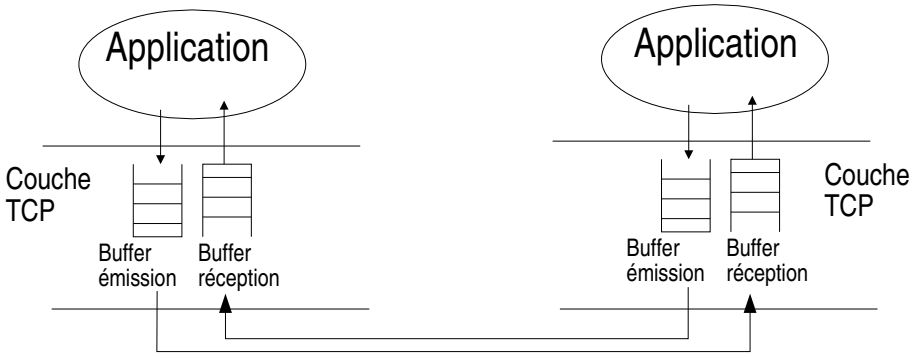
L'option `-n` de netstat permet de ne pas traduire les adresses et numéros de ports (les services). L'affichage est direct et on ne perd pas de temps en requêtes DNS pour afficher les noms des machines.

L'option `-t` restreint l'affichage au seul protocole TCP. Avec l'option `-u` on affiche les services UDP en cours (les applications serveur UDP). Il n'y a pas de connexion en UDP.

Avec la seule option `-a` on affiche toutes les connexions (ou serveurs lancées pour UDP), y compris les services n'utilisant que la communication Unix (par exemple le serveur d'affichage X-Window local et ses applications clientes).

Voir `-p tcp` ou `-p udp` sous Windows

TCP et les applications 24/44



Les données sont bufferisées et TCP les émet selon ses propres règles de contrôle de flux. Les applications n'ont pas le contrôle.

Selon la bibliothèque de développement (sockets sous Unix/Linux ou Windows par exemple) il peut être possible de dimensionner les tampons mémoire d'émission et de réception.

Le contrôle de flux TCP I 25/44

- Contrôle de flux d'extrémité à extrémité : le champ window
  - Les routeurs ne mémorisent pas "la connexion" TCP, ils ne peuvent donc pas participer au contrôle de flux. Celui-ci ne peut être réalisé que par les extrémités via le champ fenêtre.
  - Les entités TCP d'extrémités peuvent enregistrer les données reçues dans un tampon mémoire de taille limitée (8, 16, 32 Ko, ...)
  - Si l'application réceptrice ne lit pas suffisamment vite les octets s'accumulent dans le tampon mémoire de réception.

../..

Le contrôle de flux TCP II 26/44

../..
- L'entité TCP réceptrice envoie des acquittements avec une valeur de fenêtre qui diminue pour venir à 0 s'il le faut.
- 5s après avoir reçu un tel acquittement (`win=0`), l'entité émettrice teste le récepteur en lui envoyant un octet et continue ainsi en doublant l'intervalle (jusqu'à une borne de 1 min). Ce moyen permet de forcer le récepteur à renvoyer un acquittement et donner ainsi sa fenêtre. La valeur de 5s est en réalité variable selon les implémentations.
- Dès que les octets reçus par l'entité TCP réceptrice seront consommés par l'application réceptrice, la fenêtre pourra revenir à une valeur différente de 0.

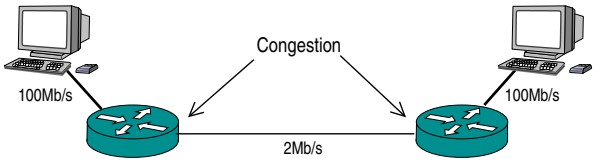
Sur Linux (noyau 2.6.xx), les paramètres liées à la gestion de la taille de la fenêtre de réception dans la pile TCP sont accessibles par la commande :

```
$ cat /proc/sys/net/ipv4/tcp_rmem
4096 87380 2076672
```

Les trois nombres donnent respectivement la valeur minimale (4096 octets), par défaut (87380 octets) et maximale (2076672 octets) de la fenêtre d'une session TCP.

Le problème de la congestion 27/44

- Il n'y a pas que les entités d'extrémités en jeu, il y a aussi le réseau. Le mécanisme de fenêtre ne permet pas d'adapter le contrôle de flux aux congestions du réseau.
- S'il y a congestion dans le réseau, des paquets peuvent être perdus, les retransmissions qui en découleront participeront au renforcement de la surcharge totale.
- Le congestion peut être due à des liens saturés mais aussi à des différences de débits entre segments de réseau.



Comment adapter TCP ?

Adaptation de TCP pour le contrôle de congestion I 28/44

- Détermination du RTT (*Round Trip Time)*
  - Avec les options spécifiques
  - Permet de «régler» un temporisateur de retransmission

Adaptation de TCP pour le contrôle de congestion II

- Le mécanisme du «*slow start*»
  - Une fenêtre de congestion est définie : `cwnd` (*congestion window*). On ne peut pas émettre plus que `cwnd` octets.
  - Juste après la connexion `cwnd` vaut 1mss (*max segment size*).
  - Un RTT plus tard, ce nombre double, et double à chaque RTT jusqu'à un seuil fixé à l'origine à 65535 (on ne peut cependant émettre que `min(cwnd, w)`, `w` étant la valeur de la fenêtre de l'entête TCP).
  - Lorsqu'une congestion a lieu, une perte se produit, le temporisateur expire et provoque une retransmission, le seuil est divisé par 2 et `cwnd` est remis à 1mss.
  - Si le seuil est dépassé, la progression de `cwnd` devient linéaire.

Les valeurs de RTT sont typiquement de :

10 ms entre sites intra Renater,

35 ms entre des sites intra Europe,

90 ms entre des sites situés en France et aux USA

210 ms entre des sites situés en Europe et en Asie

Adaptation de TCP pour le contrôle de congestion III

- Le mécanisme du «*fast recovery*» (entre autres) vient compléter le low start
  - Si on reçoit un même acquittement plusieurs fois, cela signifie deux choses : d'une part c'est qu'on a bien envoyé des segments et que ceux-ci ont été bien reçus (sinon on n'aurait pas reçu d'acquittement du tout) et d'autre part il doit y avoir un «trou» dans la séquence de segments transmis. Il suffit alors de ne retransmettre que le segment qui semble perdu.
  - Lorsque que plusieurs pertes successives ont lieu, ce mécanisme ne suffit plus car il ne permet que de récupérer le premier segment perdu. Il faut alors utiliser le mécanisme des acquittements sélectifs.

Comment éviter la segmentation avec TCP 31/44

- Le mécanisme du *Path MTU Discovery (rfc 1191)*
- le problème
  - la couche TCP émission connaît le MTU de l'interface IP de sortie (1500 par défaut sur Ethernet). Sur le chemin vers le récepteur un lien peut avoir un MTU inférieur (700 par exemple), le routeur amont sur ce lien devra segmenter.
- la solution
  - les routeurs sont interdits de segmentation TCP (bit D à 1). Si un paquet se présente de taille trop grande il est rejeté et le routeur qui le rejette émet un paquet ICMP vers l'émetteur. Ce paquet contient une information significative. Voir exemple :

Exemple de PATH MTU Discovery 32/44

Exemple : extrait d'un relevé avec tcpdump

- émission du premier paquet (seq = 1, taille 1448=MTU(1500)-TCPhead(20)-TCPoptions(12)-IPhead(20))

```
Em > Rec P 1:1449(1448) ack 1 win 16060 <nop,nop,timestamp 830140
827243>
```

- réception du mesage ICMP émis par un routeur intermédaire (celui qui devrait segmenter)

```
192.168.200.17 > Em: icmp: Rec unreachable - need to frag (mtu 700)
```

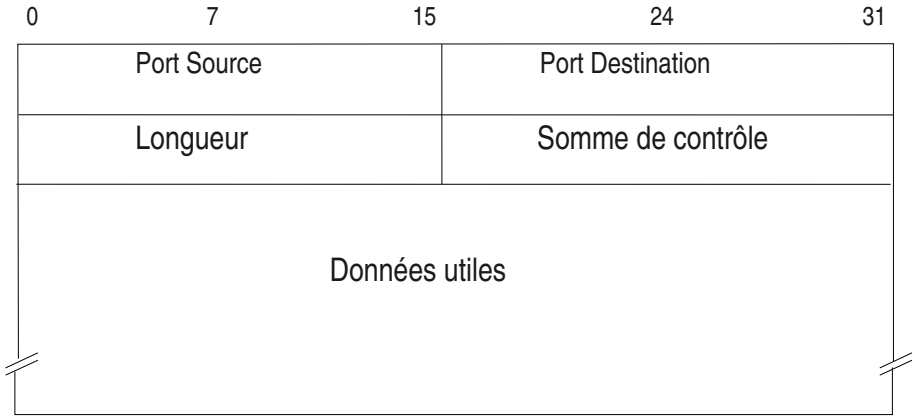
- On a bien compris... On émet des paquets de 648 octets (700-12-20-20=648)

```
Em > Rec . 1:649(648) ack 1 win 16060 <nop,nop,timestamp 830140
827243>
```

En italique : ce que tcpdump nous indique concernant la longueur du SDU véhiculé.

3 UDP

Le datagramme UDP 34/44



UDP 35/44

- Pas de connexion
- Pas de contrôle de flux
- Pas d'assurance de la remise
- Mode message : alignement des données reçues sur les données émises

UDP n'apporte que peu de choses par rapport à IP. Il garde de TCP la notion de port qui permet d'identifier les applications.

(A noter le champ longueur qui n'existe pas dans TCP)

Il n'y a pas de contrôle de flux, pas d'assurance de remise.

Il n'y a pas de «connexion».

On a cependant l'assurance que, si un datagramme arrive, il correspond très exactement à ce qui a été envoyé en ce qui concerne l'alignement des données. En d'autre termes, on peut dire

que UDP est orienté «message». Si un message est reçu, le début correspond au début envoyé, la fin à la fin...

Ce n'est pas vrai en TCP. Lorsqu'une application reçoit un buffer de données via TCP elle ne peut être certaine que cette réception correspond bien à l'émission. Certes les données sont les mêmes, elles ne sont pas alignées. TCP est orienté «flot d'octets», il assure que les octets sont bien transmis.

TCP, UDP et l'alignement des données 36/44

- TCP est orienté «flot d'octets», il ne permet pas d'assurer le «cadrage» des données reçues sur celui des données émises

  Exemple de possibilité :

- UDP est orienté «Message»

TCP garantit que les octets sont bien transmis, il ne permet pas l'alignement de la réception sur l'émission. Autrement dit, un premier «message» peut voir son début partir dans un segment TCP et arriver dans un autre. Il faut pouvoir rassembler les octets appartenant à un message donné à l'arrivée. Pour cela on peut utiliser la technique TPKT recommandée dans le RFC 1006 qui consiste à commencer les messages par un octet de version (3), un octet inutilisé (sans doute pour faire joli, ou plus sûrement pour que l'ensemble soit aligné sur 4 octets), et deux octets de longueur de message. En réception il suffit de se synchroniser sur le premier message...

UDP ne garantit pas que les messages arrivent mais s'ils sont reçus, alors, on trouve tout d'un coup (à condition toutefois que la lecture prévoit de de lire un nombre suffisant d'octets).

4 Protocoles applicatifs

Les protocoles applicatifs I 38/44

- Mis en œuvre par les applications
- Des API existent
  - Sockets (BSD) : le réseau est vu comme un fichier (on écrit et on lit le réseau comme un fichier), le protocole de communication est à implémenter «à la main»
  - Transport Services API : une API générique, draft IETF
  - RPC : l'API masque les aspects réseau, on appelle des procédure distantes de la même manière que des procédures locales
  - CORBA : *Common Request Broker Architecture*, extension aux applications réparties du concept «programmation objet»
  - ... multiples *middlewares* spécifiques ...

Les protocoles applicatifs II 39/44

- Codage des données
  - Protocoles «texte» : smtp, http (1.x), sip, sdp
  - Protocoles codés :
    - ASN.1/BER/PER : H323, snmp
    - XDR : NFS, NIS
- Notions de «session»
  - p.ex. protocole sip, sdp, beep, et même les *cookies* sur le web, etc.

smtp : simple mail transfert protocol (rfc-821)

http : hyper text transfert protocol (le protocole du Web, rfc-1945)

sip : session initiation protocol (un des protocoles pour la téléphonie sur IP, rfc-2543)

sdp : session description protocol (descripteur de flux multimedia, inspirateur de sip : rfc-2327)

beep : blocks extensible exchange protocol, un framework pour créer des protocoles applicatifs, incluant des mécanismes d'authentification et de gestion de session, rfc 3080 3081 3117

ASN.1 : Abstract Notation, version 1 (on espère qu'il n'y aura pas de version 2, ou alors plus simple...) : langage de spécification de types de données. Comme un langage informatique ou on ne définirait que des types et des variables de ces types.

On lui associe une syntaxe de transfert (en clair : des mécanismes de codage en ligne) tels que le BER (Basic Encoding Rules) ou le PER (Packet encoding Rules).

ASN.1/BER : utilisé pour snmp (simple network management protocol, rfc 1155 à 1158, 1212, 1213, etc)

ASN.1/PER est utilisé pour définir les données de signalisation pour la téléphonie sur IP dans l'architecture H323 (définition des PDUs H225 et H245, plusieurs centaines de pages, avec rien que du ASN.1 goûteux et savoureux).

On retrouve aussi ASN.1 dans les spécifications de la structure des certificats de sécurité X509.

JSON : JavaScript Object Notation, un format de données textuelles structurées façon Javascript (Objets). Très utilisé dans les applications Web (échanges de données entre le navigateur de l'utilisateur et le serveur, mis en forme dans une page html)

CBOR : Concise Binary Object Representation, similaire à JSON, mais avec une représentation binaire (donc plus compact).

Exemple d'échange HTTP : la requête 40/44

```
GET /index.fr.php HTTP/1.1
Host: www.enst-bretagne.fr
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:1.
Gecko/20030624 Netscape/7.1 (ax)
Accept: text/xml,application/xml,.....
Accept-Language: fr,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://www.enst-bretagne.fr/
```

Exemple d'échange HTTP : la réponse 41/44

```
HTTP/1.1 200 OK
Date: Fri, 09 Jul 2004 09:20:06 GMT
Server: Apache/1.3.22 (Unix) PHP/4.0.4pl1 mod_fastcgi/2.2.10 PHP/3.0.....
X-Powered-By: PHP/4.0.4pl1
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html
Content-Language: fr

<HTML>
<HEAD>
<TITLE>[ENST Bretagne] école formation ingénieur</TITLE>
<!-L'école nationale supérieure des télécommunications de Bretagne offre
un large choix de formation: ingénieur, mastères, télécom, thèse, DEA...->
<meta name="robots" content="noindex,nofollow">
<meta name="description" content="L'école nationale supérieure des télécommunications de Bretagne offre un
large choix de formation: ingénieur, mastères, télécom, thèse, DEA...">
<meta name="keywords" content="école ingénieur, formation ingénieur, école nationale supérieure
télécommunications, Bretagne, mastères, télécom, thèse, DEA, ENST, GET, étude télécommunication,
enseignement supérieur, technologie information, communication, TIC, concours, mines-ponts, alternance,
projets européens, recherche, laboratoire, grandes écoles, formation continue, diplôme, Brest, Rennes,
ECOLE INGENIEUR, FORMATION INGENIEUR">
<LINK rel="stylesheet" href="css/style.css">
...
```

Coté client, c'est au programmeur de l'application de préparer «à la main» les chaînes de caractères constituant les messages de requêtes du protocole. Coté serveur il faut savoir interpréter ces messages (le mot «savoir» étant traduisible en «implémentation de code»), il faut savoir aussi constituer les messages de réponse. Ces messages sont donc des suites de caractères ascii, placés dans des tampons mémoire qu'il suffit ensuite d'envoyer sur des sockets, coté émission. Coté réception il suffit de lire les sockets (comme on lirait des fichiers) et d'analyser en,suite ce qui est mémorisé dans les tampons de lecture.

Le langage est simple, il obéit à une grammaire (au sens informatique du terme) parfaitement spécifiée. Il est possible, à partir des spécifications de générer facilement des fonctions de traitement des messages. Il existe des outils adaptés aux traitements lexicaux, voire sémantiques, de telles grammaires : lex et yacc sous unix, flex et bison en logiciels libres.

Exemple d'échange smtp 42/44

(Exemple extrait du rfc-821)

```
S: MAIL FROM:<Smith@Alpha.ARPA>
R: 250 OK

S: RCPT TO:<Jones@Beta.ARPA>
R: 250 OK

S: RCPT TO:<Green@Beta.ARPA>
R: 550 No such user here

S: RCPT TO:<Brown@Beta.ARPA>
R: 250 OK

S: DATA
R: 354 Start mail input; end with <CRLF>.<CRLF>
S: Blah blah blah...
S: ...etc. etc. etc.
S: <CRLF>.<CRLF>
R: 250 OK
```

`S:` indique l'émetteur, `R:` indique le récepteur. (`S:` et `R:` ne font pas partie du protocole.)

`MAIL FROM:`, `RCPT TO:`, `DATA` sont des entêtes du protocole, de même que les séquences `<CRLF>` qui signifient simplement *retour à la ligne* pour `CR` (Carriage Return) et *saute de ligne* pour `LF` (Line Feed).

On peut noter que le cacatère `.` (point) est aussi partie intégrante du protocole.

Voici donc le protocole de base du courrier Internet si utilisé aujourd'hui. Ses spécification datent d'août 1982 !

Que pensez-vous des aspects sécurité d'un tel protocole ?

Fonctionnement du courrier électronique I 43/44

- Les utilisateurs ont une adresse de format standard
  - Par ex. :
    - `nom_utilisateur@nom-du-serveur.domaine`
  - ou plus concis :
    - `nom_utilisateur@domaine`
- Les outils mis en œuvre sont
  - Le client (MUA : *Mail User Agent* en langage OSI) : netscape, outlook, lotusNotes, etc...
    - Parle SMTP lors de l'envoi
    - Parle POP ou IMAP (sécurisé ou non) pour recevoir
  - Les serveurs (MTA : *Message Transfert Agent*) : ISS, sendmail, Postfix, etc...
    - Intermédiaires et final
  - Le DNS : requêtes MX (*Mail Exchanger*)

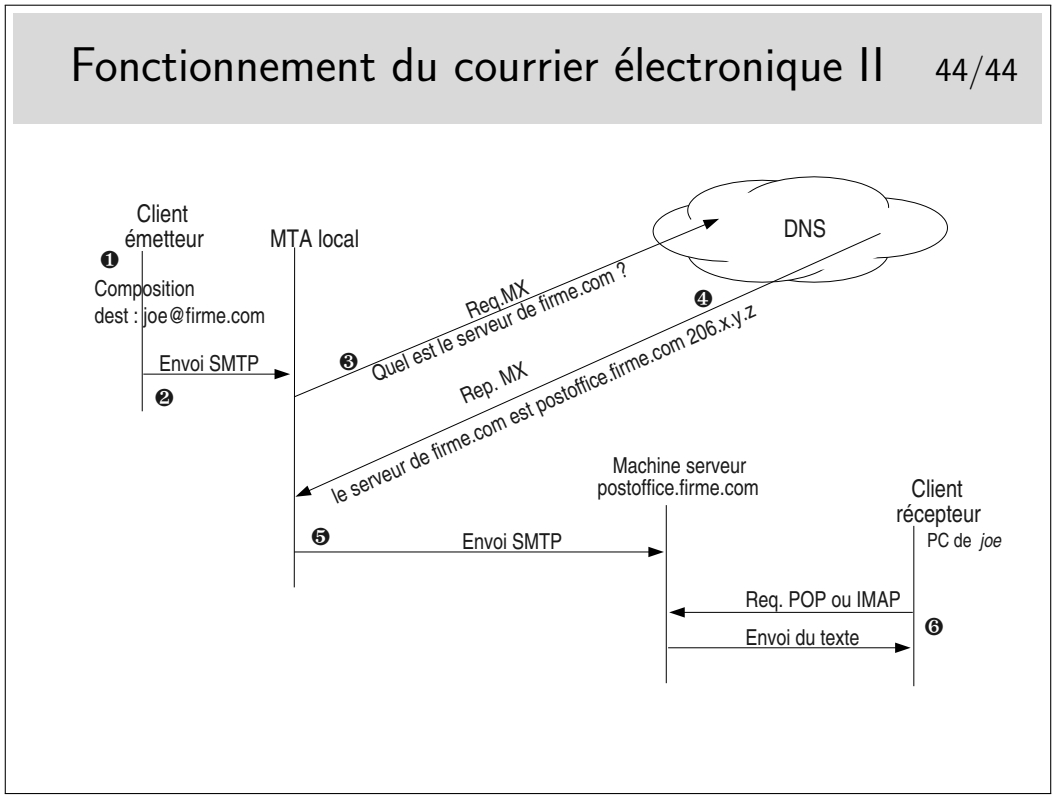
POP : Post Office Protocol

IMAP : Internet Message Access Protocol

Les protocoles POP et IMAP sont par défaut sans chiffrement et nécessitent l'échange de mots de passe utilisateur. Ces échanges se font par défaut en clair pour POP. IMAP supporte différents mécanismes d'authentification, notemment un système de challange dans lequel le mot de passe est chiffré (pour autant, les messages ne le sont pas...) Les outils clients ainsi que les serveurs peuvent être configurés pour fonctionner avec chiffrement (POPS ou IMAPS : POP/IMAP + SSL).

POP permet de télécharger sur son poste client les courriers reçus sur le serveur final. Les messages sont totalement transférés. Ils peuvent cependant rester sur le serveur.

IMAP permet de ne charger que les entêtes, il permet aussi de rechercher les messages par mot clé, les messages peuvent rester stockés sur le serveur, et de gérer des sous-répertoires (*folders*) sur le serveur.



Il peut y avoir plus de machines intermédiaires que ne le montre la figure ci-dessus. En particulier, coté réception, la machine serveur (appelée ici `postoffice.firme.com`) peut renvoyer vers un serveur interne déservant le département de l'utilisateur joe, ce denier paramètre alors son client pour qu'il se connecte par POP ou IMAP sur ce serveur particulier.

Il peut y avoir plusieurs serveurs d'arrivée, classés selon des priorités dans le DNS.

Il peut être intéressant de regarder l'entête complète des messages que l'on reçoit pour avoir la liste des MTA traversées par ces messages. Cette curiosité étant encore plus intéressante en cas de problèmes d'acheminement.

Les MTAs peuvent filtrer les messages, au départ comme à l'arrivée. Ils peuvent être munis de détecteurs de SPAMs et de virus.

Vous pouvez tester le DNS avec des requêtes MX de la manière suivante (sous Windows) :

```
c:\nslookup
> set type=MX
> microsoft.com
```

Ou sous Unix : `host -t MX microsoft.com` (la commande `nslookup`, bien qu'encore présente, est maintenant considérée comme rendue obsolète par la commande `host`).