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

Cybersécurité des systèmes maritimes et portuaires

La couche transport

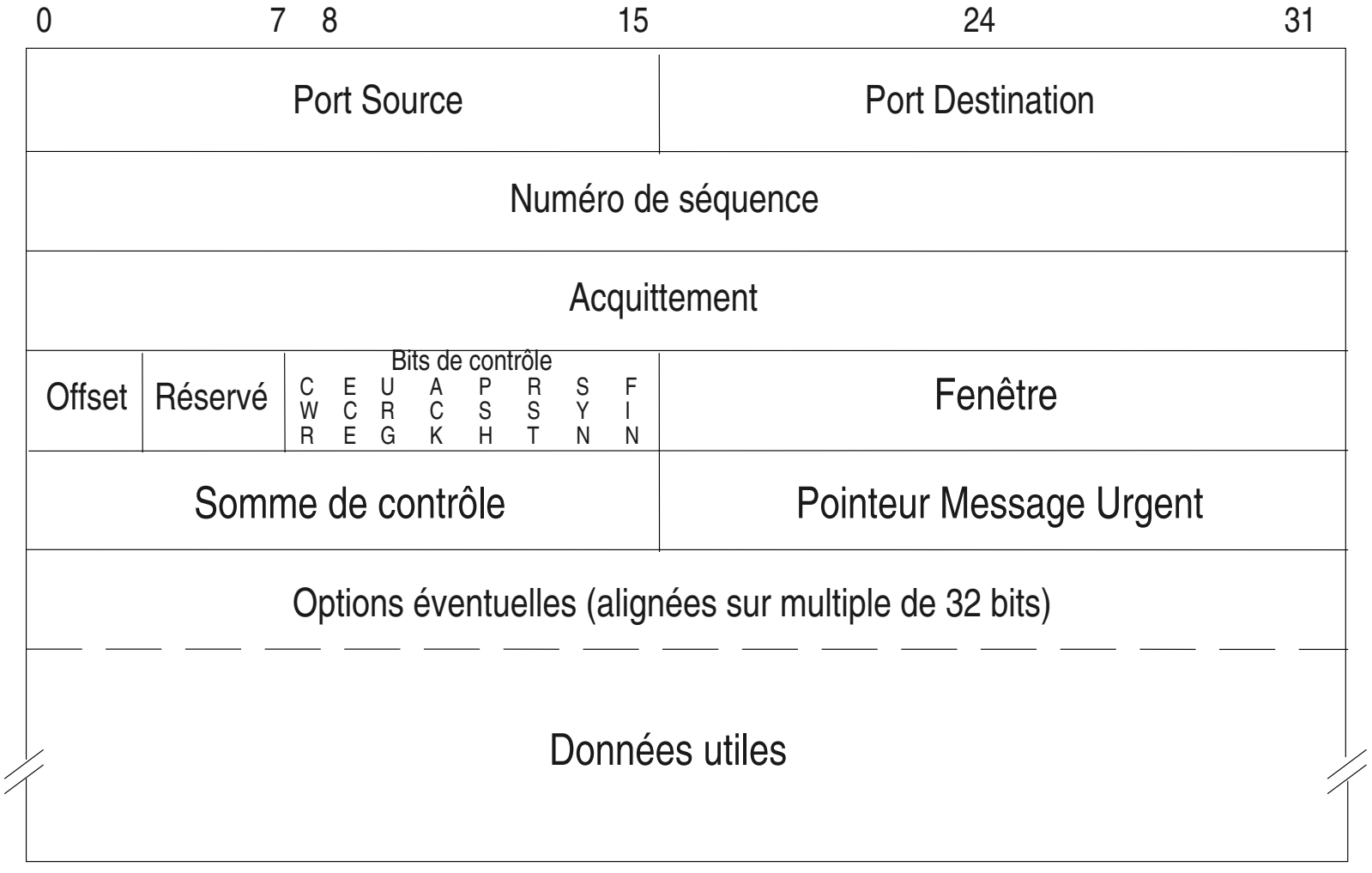
Christophe Lohr

2021

Les protocoles de transport au dessus de IP

- 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

Le segment TCP



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.

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»

- 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

- 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

- 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 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 :

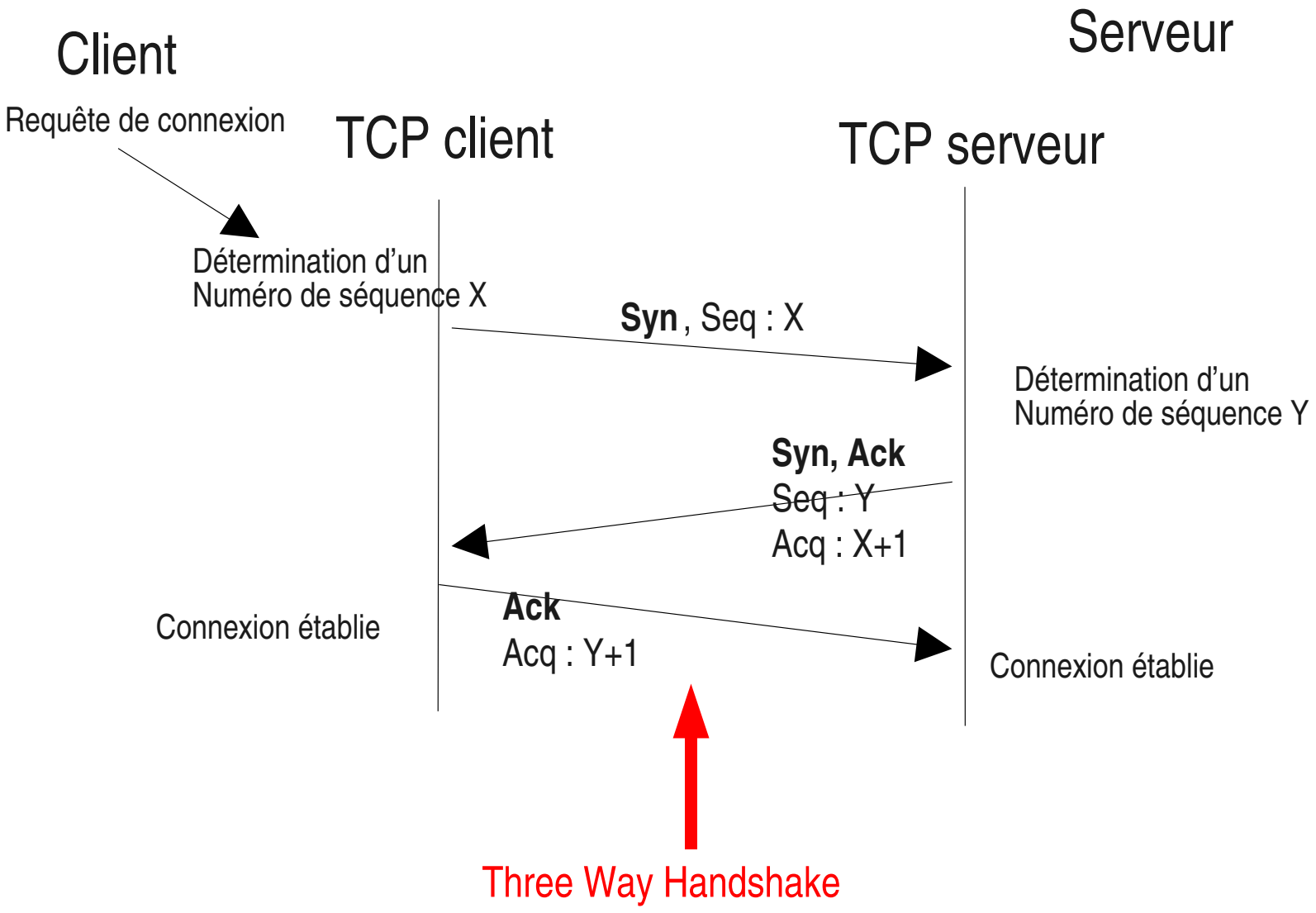
```
[protocole, port local, port distant, @IP locale, @IP distante]
```

- 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

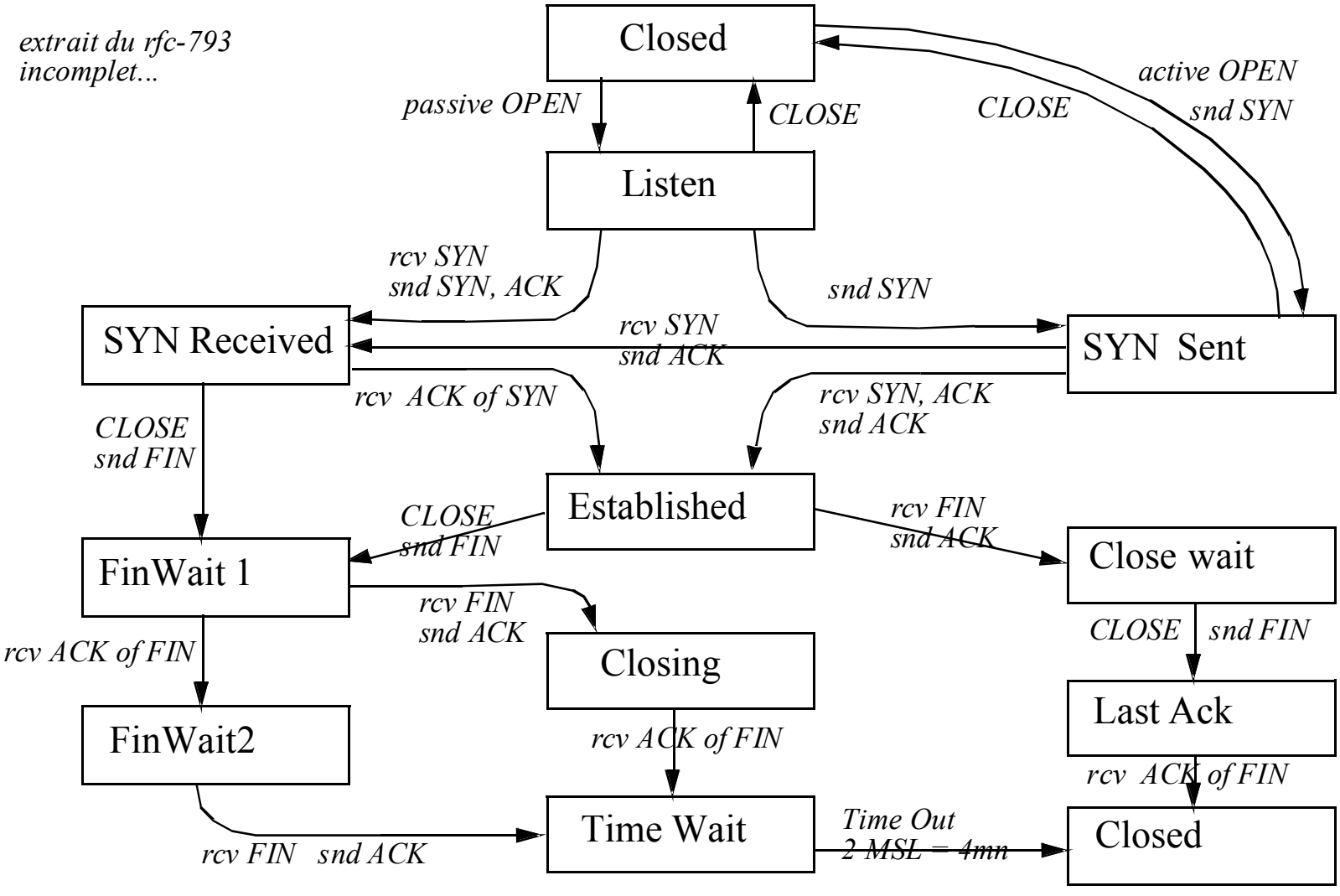
Connexions actives et passives

- 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

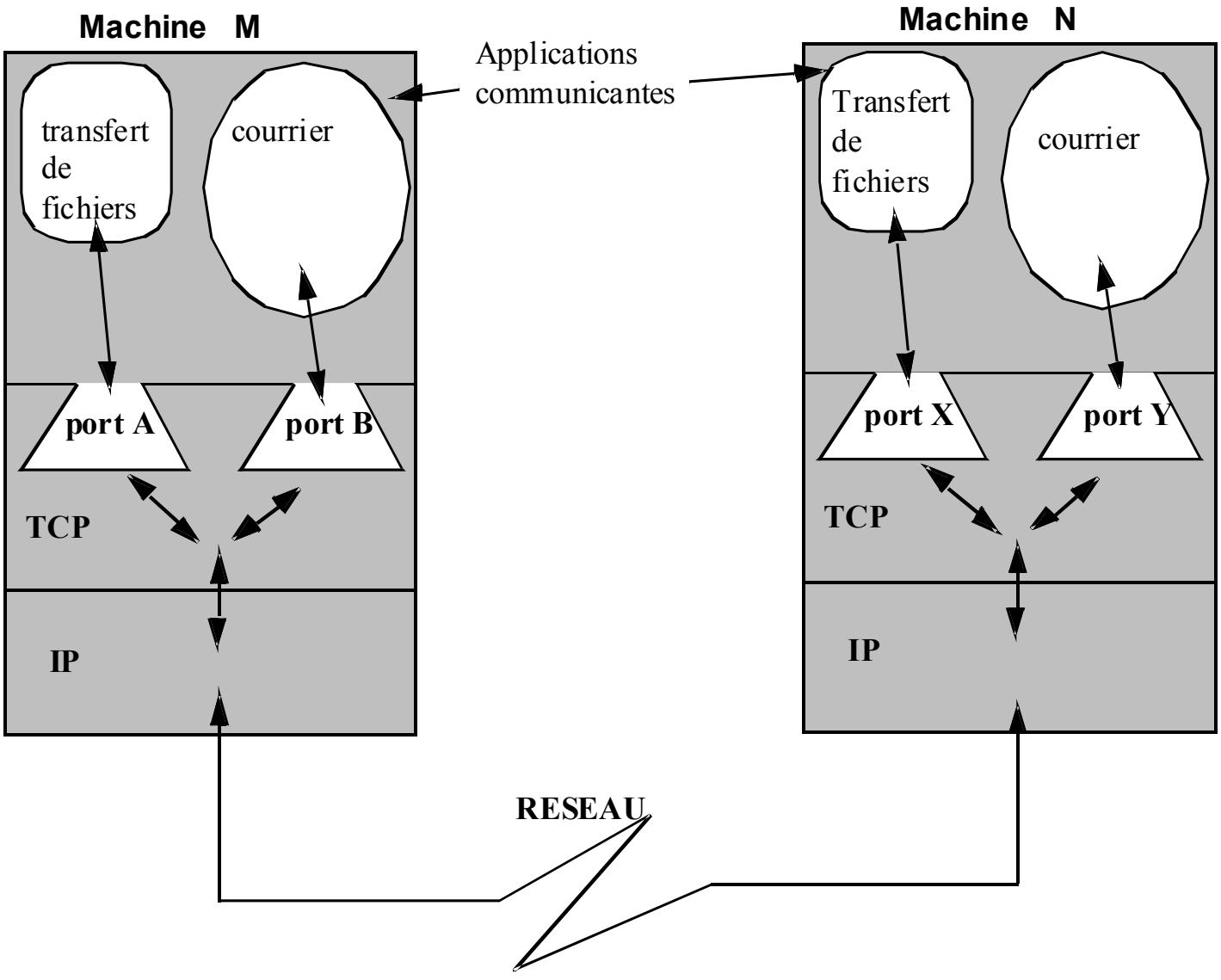
Établissement de la connexion



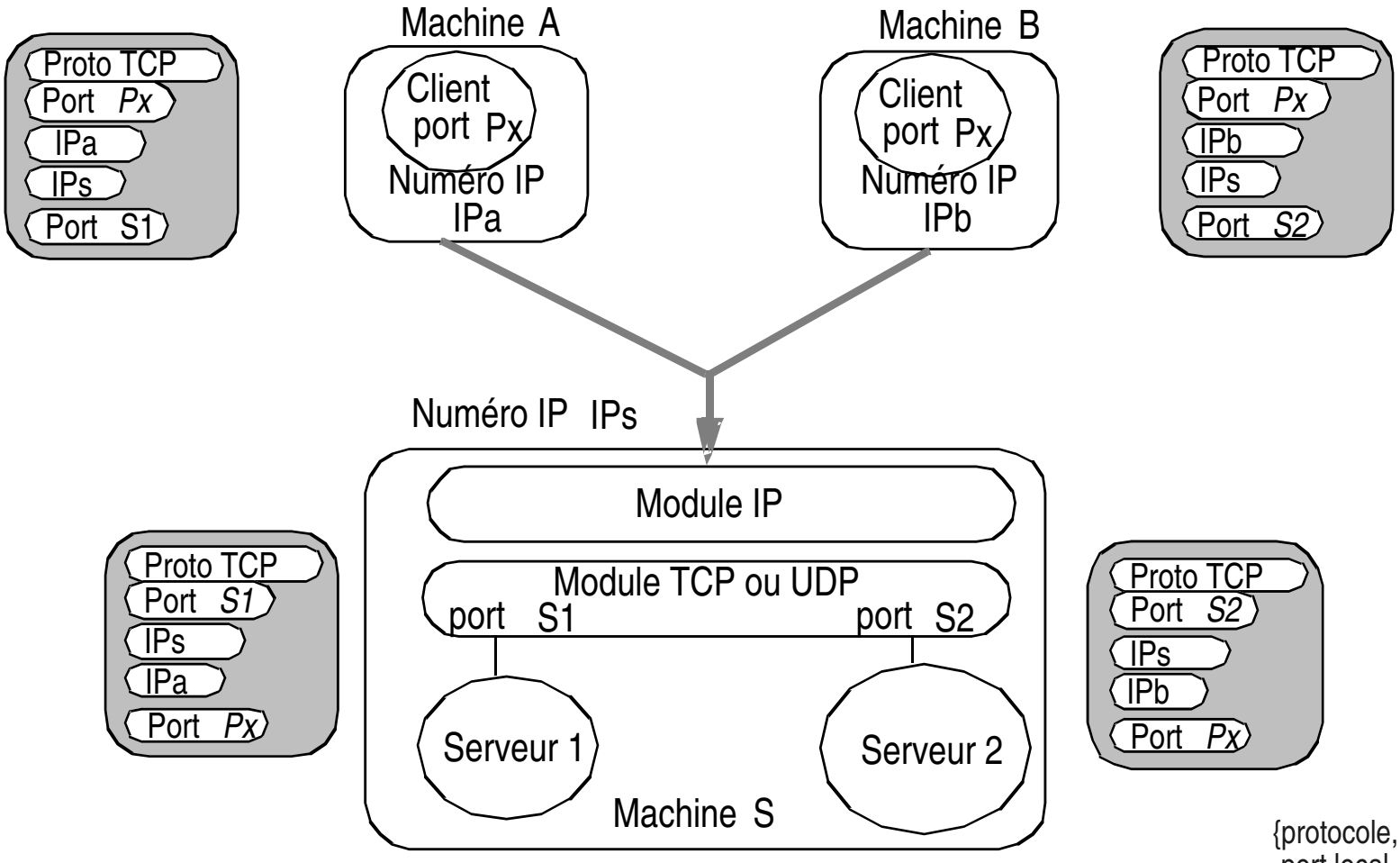
La machine d'états TCP



La notion de port



Identification des «connections»



Identification des connexions sans ambiguïté par le quintuplet {protocole, port local, n° IP local, port distant, n° IP distant}

- 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.

- 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)

- 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.

- 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

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

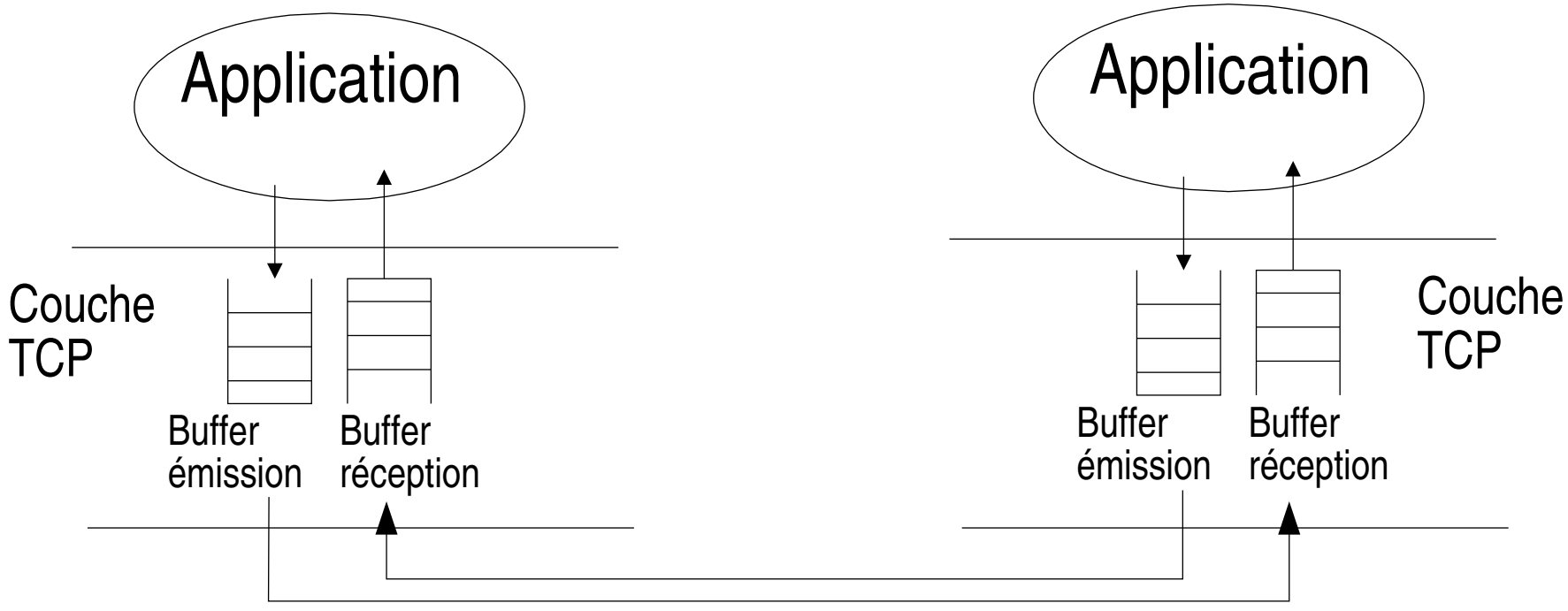
```
linux1.1082 > 206.132.41.203.www: S 2047350264:2047350264(0) win 16060 <mss
1460,sackOK,timestamp 43244276>
206.132.41.203.www > linux1.1082: S 2319802134:2319802134(0) ack 2047350265 win
32120 <mss 1460, sackOK, timestamp 190973015>
linux1.1082 > 206.132.41.203.www: . ack 2319802135 win 16060 <nop,nop,timestamp
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>
```

Le contrôle des connexions : la commande `netstat`

- 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
```

TCP et les applications



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

- 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.

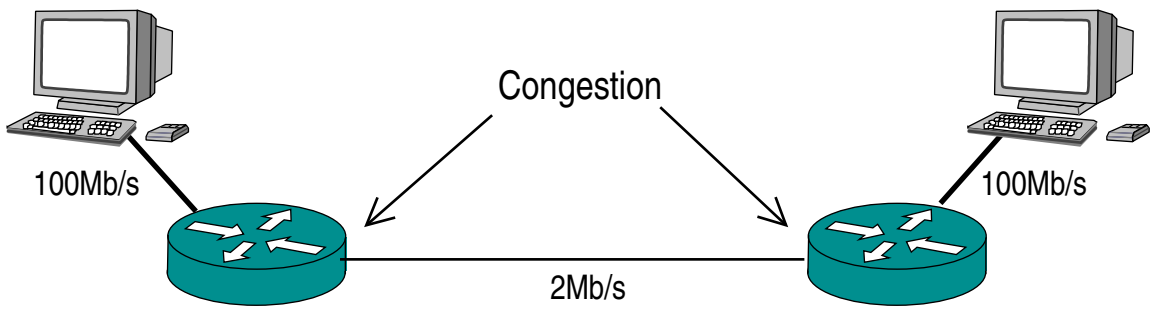
../..

../..

- 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.

Le problème de la congestion

- 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 ?

- 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.

- 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

- 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

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>
```

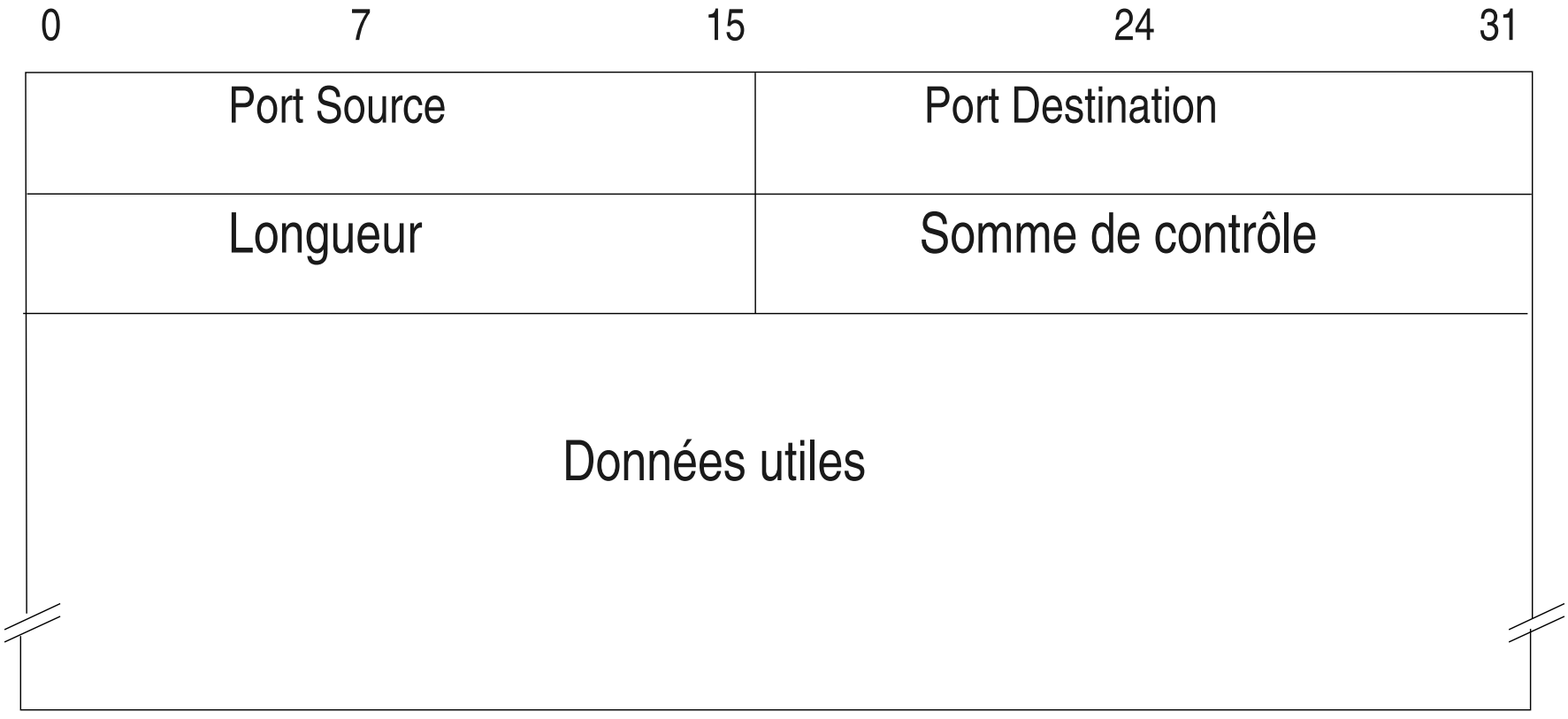
- réception du mesage ICMP émis par un routeur intermédaire (celui qui devrait segmenter)

```
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>
```

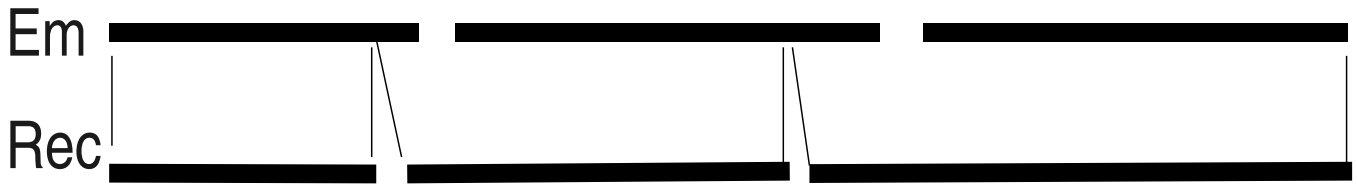
Le datagramme UDP



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

- 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»

- 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 ...

- 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.

Exemple d'échange HTTP : la requête


```
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


```
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">
...
```

Exemple d'échange smtp

(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
```

Fonctionnement du courrier électronique I

- 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*)

Fonctionnement du courrier électronique II