

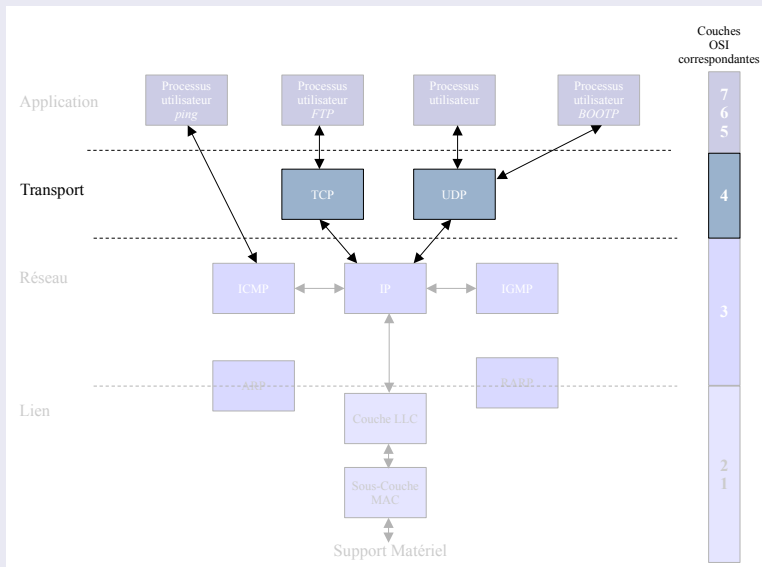
# Internet - couche transport

Nicolas Delestre et Géraldine Del Mondo

# Plan

- 1 Client - Serveur
- 2 Ports
- 3 UDP
- 4 TCP
  - Connexion
  - Déconnexion
- 5 Outils UNIX
- 6 Conclusion

# Rappels



# Client, Serveur

## Client

- Application qui prend l'initiative du lancement de la communication

## Serveur

- Application qui attend la communication
- Il peut y avoir plusieurs serveurs sur une machine :
  - Il peut y avoir l'utilisation de « super » démons (sous UNIX *inetd*, *xinetd*)

# Concept de ports

Lorsqu'une application cliente veut communiquer avec une application serveur il faut qu'elle identifie cette dernière sur l'hôte serveur : c'est le concept de **port**

- numéro compris entre 1 et 65535
- port < 1024 = port système (il faut être root pour lancer des applications serveurs qui utilisent ces ports)
- une application peut utiliser plusieurs ports (par exemple ftp)

## Sous UNIX

/etc/services

# Quelques ports

## extrait de /etc/services

```
...  
  
echo 7/tcp  
echo 7/udp  
discard 9/tcp sink null  
discard 9/udp sink null  
systat 11/tcp users  
daytime 13/tcp  
daytime 13/udp  
netstat 15/tcp  
qotd 17/tcp quote  
msp 18/tcp # message send protocol  
msp 18/udp  
chargen 19/tcp ttytst source  
chargen 19/udp ttytst source  
ftp-data 20/tcp  
ftp 21/tcp  
fsp 21/udp fspd  
ssh 22/tcp # SSH Remote Login Protocol  
ssh 22/udp  
telnet 23/tcp  
smtp 25/tcp mail  
time 37/tcp timserver  
time 37/udp timserver  
  
...
```

# Port source

- En réponse à sa requête, le serveur doit répondre sur l'un des ports du client
- Ce port est dit dynamique ou éphémère



*Cf. Réseaux et Télécoms, Servin, p266*

# UDP

- Protocole de communication orientée sans connexion
  - Peut être considéré comme de l'IP avec le concept de port
- Donc communication non fiable mais rapide
  - Application temps réel (flux vidéo, audio)
  - Service non sécurisé (DNS, SNMP, etc.)

## Segment UDP

port source	port destination
longueur	checksum
données	



# UDP

## Calcul du checksum

- Porte sur l'en-tête, les données et une pseudo en-tête (non envoyée)
  - *viole le principe de hiérarchie des protocoles*

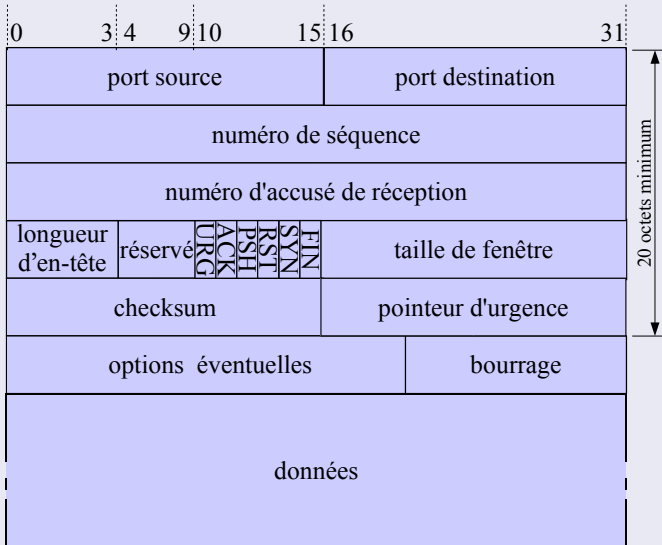
## Pseudo entête UDP

Adresse source		
Adresse destination		
0	Protocole = 17	Longueur segment UDP

# TCP

- Protocole de communication orientée connexion
- S'appuyant sur IP (réseau non fiable), TCP doit :
  - assurer la délivrance en séquence
  - contrôler la validité des données reçues
  - organiser les reprises sur erreur
  - réaliser le contrôle de flux

# Segment TCP



# Segment TCP

**Numéro de séquence** valeur aléatoire définie et acquittée par les 2 systèmes lors de la phase de connexion, ensuite il indique le rang du premier octet du segment transmis

**Numéro d'accusé de réception** accuse réception du segment en indiquant le numéro du prochain octet attendu

**Longueur d'entête** multiple de 4 octets (5 minimum)

**URG** urgent (utilisation du pointeur d'urgence)

**ACK** valide le champ numéroté de séquence acquittée

**PSH** demande au destinataire de traiter de suite le segment (pas de temporisation)

**RST** réinitialisation de la connexion

**SYN** demande de connexion

**FIN** demande de déconnexion

**taille fenêtre** combien peut-on transmettre d'octets après l'octet acquitté (contrôle de flux)

# Segment TCP

**checksum** contrôle calculée sur le segment TCP (Même méthode que l'UDP avec 6 pour type de protocole)

**pointeur d'urgence** décalage en octet à partir du numéro de séquence courant de données urgentes

**options** par exemple spécification de la charge utile TCP la plus grande

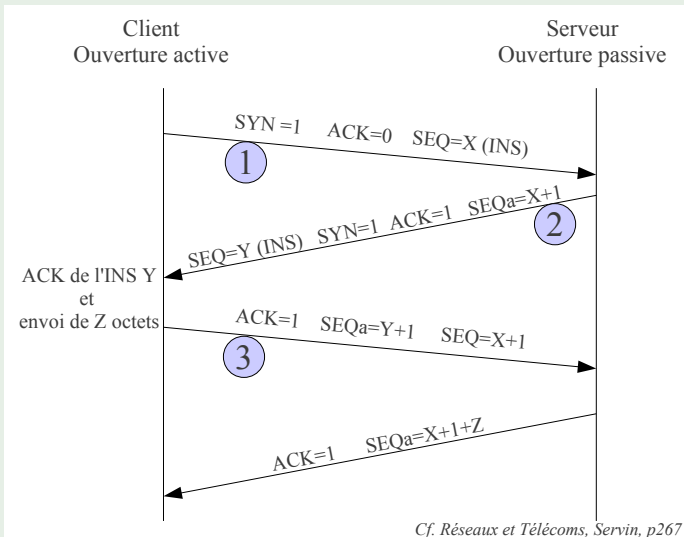
# Principe

- Puisque IP est non fiable chaque envoi d'information d'un tiers reçoit une réponse indiquant qu'il a bien reçu une information (ACK) indentifiée par le Numéro d'accusé de réception ( $SEQ_a$ )
- Comme un tiers peut être connecté à plusieurs applications, chaque connexion est identifiée au départ par un *Numéro de séquence initiale INS* qui est incrémenté par le nombre d'octets transmis

# Connexion

- À la connexion les tiers s'échangent leur INS, on a donc un protocole de connexion en 3 temps :
  - 1 Le client demande la connexion en envoyant un SYN et son INS
  - 2 Le serveur répond en accusant réception de la demande, et lui envoie aussi un SYN et son INS
  - 3 Le client accuse réception et peut éventuellement envoyer des informations

# Connexion

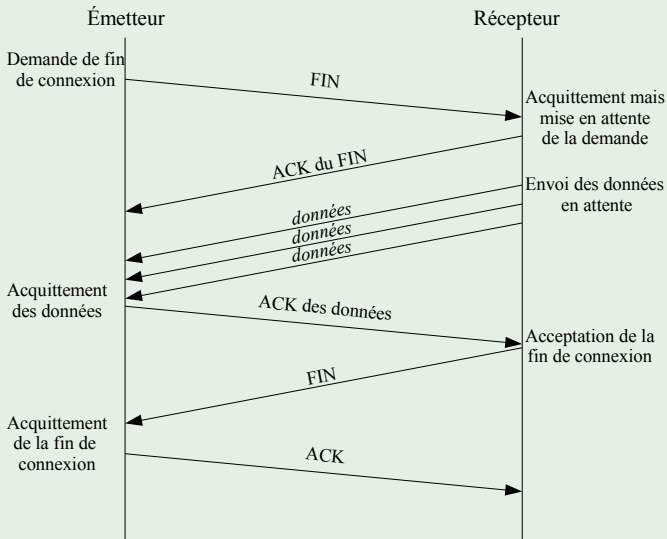




# Déconnexion

- Sachant que la communication est bi-directionnelle, le tiers qui demande la fin de connexion (il n'a donc plus d'information à envoyer) doit attendre que l'autre tiers lui envoie sa demande de déconnexion

# Déconnexion



Cf. Réseaux et Télécoms, Servin, p268

# netstat

Affiche les connexions en cours

## Exemple <sup>a</sup>

a. <http://linux-ip.net/html/tools-netstat.html>

```
[root@morgan]# netstat --inet -n
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp      0      192 192.168.98.82:22       192.168.99.35:40991    ESTABLISHED
tcp      0      0 192.168.98.82:42929   192.168.100.17:993    ESTABLISHED
tcp     96      0 127.0.0.1:40863      127.0.0.1:6010       ESTABLISHED
tcp      0      0 127.0.0.1:6010      127.0.0.1:40863     ESTABLISHED
tcp      0      0 127.0.0.1:38502     127.0.0.1:6010       ESTABLISHED
tcp      0      0 127.0.0.1:6010     127.0.0.1:38502     ESTABLISHED
tcp      0      0 192.168.98.82:53733  209.10.26.51:80      SYN_SENT
tcp      0      0 192.168.98.82:44468  192.168.100.17:993    ESTABLISHED
tcp      0      0 192.168.98.82:44320  192.168.100.17:139    TIME_WAIT
```

# tcpdump

- Permet de récupérer les flux réseaux
- Options les plus utilisées :
  - i interface
  - X avoir le contenu en ASCII et hex
  - src host suivi d'une adresse IP source
  - dst host suivi d'une adresse IP destination
  - port uniquement un port

## Exemple <sup>a</sup>

a. <http://okki666.free.fr/docmaster/articles/linux084.htm>

```
#tcpdump -x -X -s 0 src host 192.168.0.1 and dst host 212.208.225.1 and port 53 and udp
tcpdump: listening on eth0
```

```
20:30:39.189321 raven.32929 > ns1.rmcnet.fr.domain: 5314+ A? www.linux.org. (31) (DF)
0x0000  4500 003b 0000 4000 4011 ca00 c1fd d9b4  E.;...@.....
0x0010  c1fc 1303 80a1 0035 0027 213d 14c2 0100  .....5.'!=...
0x0020  0001 0000 0000 0000 0377 7777 056c 696e  .....www.lin
0x0030  7578 036f 7267 0000 0100 01  ux.org.....
```

# Wireshark

Filter:  Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	7e:a9:d4:ff:2a:55	Broadcast	ARP	42	Who has 10.0.0.2? Tell 10.0.0.1
2	0.000028	86:ae:d8:02:2c:56	7e:a9:d4:ff:2a:55	ARP	42	10.0.0.2 is at 86:ae:d8:02:2c:56
3	0.000166	10.0.0.1	10.0.0.2	TCP	74	33378 > telnet [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=2787 TSecr=0 WS=2
4	0.000427	10.0.0.2	10.0.0.1	TCP	74	telnet > 33378 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=2770 TSecr=2787 WS=2
5	0.000544	10.0.0.1	10.0.0.2	TCP	66	33378 > telnet [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSval=2791 TSecr=2770
6	0.004417	10.0.0.1	10.0.0.2	TELNET	90	Telnet Data ...
7	0.004733	10.0.0.2	10.0.0.1	TCP	66	telnet > 33378 [ACK] Seq=1 Ack=25 Win=5792 Len=0 TSval=2770 TSecr=2791
8	0.103231	10.0.0.2	10.0.0.1	TELNET	78	Telnet Data ...
9	0.103454	10.0.0.1	10.0.0.2	TCP	66	33378 > telnet [ACK] Seq=25 Ack=13 Win=5840 Len=0 TSval=2800 TSecr=2770
10	0.103705	10.0.0.2	10.0.0.1	TELNET	81	Telnet Data ...
11	0.103712	10.0.0.1	10.0.0.2	TELNET	69	Telnet Data ...
12	0.114437	10.0.0.2	10.0.0.1	TCP	66	telnet > 33378 [ACK] Seq=28 Ack=28 Win=5792 Len=0 TSval=2783 TSecr=2800
13	0.114589	10.0.0.1	10.0.0.2	TELNET	75	Telnet Data ...

▶ Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits)

▶ Ethernet II, Src: 7e:a9:d4:ff:2a:55 (7e:a9:d4:ff:2a:55), Dst: Broadcast (ff:ff:ff:ff:ff:ff)

▶ Address Resolution Protocol (request)

```

0000 ff ff ff ff ff ff 7e a9 d4 ff 2a 55 08 06 00 01 .....U....
0010 08 00 06 04 00 01 7e a9 d4 ff 2a 55 0a 00 00 01 .....U....
0020 00 00 00 00 00 00 0a 00 00 02 .....

```

# nmap, nmapfe

- Permet de scanner des ports (et d'identifier l'OS de la machine et les applications serveurs)

Starting nmap 3.49 ( <http://www.insecure.org/nmap/> ) at 2003-12-19 14:28 PST  
 Interesting ports on www.insecure.org (205.217.153.53):  
 (The 1212 ports scanned but not shown below are in state: filtered)  
**PORT** **STATE** **SERVICE** **VERSION**  
 22/tcp open **ssh** OpenSSH 3.1p1 (protocol 1.99)  
 25/tcp open **smtp** qmail **smtpd**  
 53/tcp open **domain** ISC Bind 9.2.1  
 80/tcp open **http** Apache **httpd** 2.0.39 ((Unix) mod\_perl/1.99\_07-dev Perl/v5.6.1)  
 113/tcp closed **auth**  
 Device type: general purpose  
 Running: **Linux** 2.4\_X12.5.X  
 OS details: **Linux** Kernel 2.4.0 - 2.5.20  
 Uptime 212,119 days (since Wed May 21 12:38:26 2003)  
 Nmap run completed -- 1 IP address (1 host up) scanned in 33,792 seconds

Command: `nmap -sS -sV -O -F -PI -T4 www.insecure.org`

[http://www.insecure.org/nmap/nmap\\_relatedprojects.html](http://www.insecure.org/nmap/nmap_relatedprojects.html)

# Conclusion : couche Transport

→ Un émetteur, un récepteur, une transmission ...

- Assurer la transmission de bout en bout
- Assurer la qualité de service
- Assurer le contrôle de flux

- Les concepts de :
  - Ports
  - Protocole sans connexion : UDP
  - Protocole orienté connexion : TCP
  - Contrôle de flux
  - Bufferisation

# Références

- [1] P. Nicolas.  
Cours de réseaux de la maîtrise de l'université d'angers.  
<http://www.info.univ-angers.fr/pub/pn>, 2004.
- [2] C. Servin.  
*Réseaux et Télécoms*.  
Dunod, Sciences SUP, 2003.  
ISBN 2-10-007986-7.
- [3] A. Tanenbaum.  
*Réseaux 4<sup>ème</sup> édition*.  
Pearson Education, 2003.  
ISBN 2-7440-7001-7.