Réseaux de Transmission de Données

3/4: Les protocoles

Olivier Fourmaux
Olivier.Fourmaux@L2ti.univ-paris13.fr

DESS AIM

Version 1.0, Septembre 2002
La couche transport
La **Couche Transport** permet de faire **communiquer directement** deux ou plusieurs entités sans avoir à se préoccuper des différents éléments de réseaux traversés :

- communications virtuelles
- de bout-en-bout (*end-to-end*)
- abstraction des technologies sous-jacentes
- services génériques attendus par les applications :
  - ✔ contrôle d’erreur
  - ✔ fiabilité
  - ✔ ordonnancement
  - ✔ contrôle de flux
  - ✔ contrôle de congestion
Couche Transport : OSI

- Application
- Presentation
- Session
- Transport
- Network
- Data link
- Physical

Interface

Application protocol
Presentation protocol
Session protocol
Transport protocol
Network protocol
Data link protocol
Physical protocol

APDU

Host A

Router

Host B
Couche Transport : TCP/IP
Couche Transport : Primitives

Interface de *programmation* (application ou développeurs)

- LISTEN
- CONNECT
- SEND
- RECEIVE
- DISCONNECT
Couche Transport : Automate de connexion

Connection request
TPDU received

Passive establishment pending

IDLE

Connect primitive executed

Active establishment pending

Established

Disconnection request
TPDU received

Passive disconnect pending

IDLE

Disconnect primitive executed

Active disconnect pending

Connection accepted
TPDU received

Disconnect primitive executed

Disconnection request
TPDU received

pictures from Tanenbaum A. S. Computer Networks 3rd edition
Couch Exchange: Establishment (call setup)
Couche Transport : Multiplexage (1)

Layer

4
3
2
1

Router lines

To router

Transport address

Network address

(a)

(b)

pictures from TANENBAUM A. S. Computer Networks 3rd edition
### Couche Transport : Multiplexage (2)

<table>
<thead>
<tr>
<th>Commande</th>
<th>Port</th>
<th>Service</th>
<th>Protocole</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unix&gt; cat /etc/services</td>
<td>time</td>
<td>37/tcp</td>
<td></td>
</tr>
<tr>
<td>tcpmux</td>
<td>1/tcp</td>
<td>whois</td>
<td>43/tcp</td>
</tr>
<tr>
<td>echo</td>
<td>7/tcp</td>
<td>domain</td>
<td>53/tcp</td>
</tr>
<tr>
<td>echo</td>
<td>7/udp</td>
<td>domain</td>
<td>53/udp</td>
</tr>
<tr>
<td>discard</td>
<td>9/tcp</td>
<td>bootps</td>
<td>67/tcp</td>
</tr>
<tr>
<td>discard</td>
<td>9/udp</td>
<td>bootps</td>
<td>67/udp</td>
</tr>
<tr>
<td>systat</td>
<td>11/tcp</td>
<td>bootpc</td>
<td>68/tcp</td>
</tr>
<tr>
<td>daytime</td>
<td>13/tcp</td>
<td>bootpc</td>
<td>68/udp</td>
</tr>
<tr>
<td>daytime</td>
<td>13/udp</td>
<td>tftp</td>
<td>69/udp</td>
</tr>
<tr>
<td>netstat</td>
<td>15/tcp</td>
<td>gopher</td>
<td>70/tcp</td>
</tr>
<tr>
<td>qotd</td>
<td>17/tcp</td>
<td>gopher</td>
<td>70/udp</td>
</tr>
<tr>
<td>chargen</td>
<td>19/tcp</td>
<td>finger</td>
<td>79/tcp</td>
</tr>
<tr>
<td>chargen</td>
<td>19/udp</td>
<td>www</td>
<td>80/tcp</td>
</tr>
<tr>
<td>ftp-data</td>
<td>20/tcp</td>
<td>www</td>
<td>80/udp</td>
</tr>
<tr>
<td>ftp</td>
<td>21/tcp</td>
<td>kerberos</td>
<td>88/tcp</td>
</tr>
<tr>
<td>ssh</td>
<td>22/tcp</td>
<td>kerberos</td>
<td>88/udp</td>
</tr>
<tr>
<td>ssh</td>
<td>22/udp</td>
<td>hostnames</td>
<td>101/tcp</td>
</tr>
<tr>
<td>telnet</td>
<td>23/tcp</td>
<td>iso-tsap</td>
<td>102/tcp</td>
</tr>
<tr>
<td>smtp</td>
<td>25/tcp</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Un protocole de la couche transport en mode non-connecté : UDP
UDP : Port source

<table>
<thead>
<tr>
<th>Source port</th>
<th>Destination port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segment length</td>
<td>Checksum</td>
</tr>
</tbody>
</table>

- multiplexage à la source (plusieurs clients vers un serveur)
- 16 bits (65K ports)
**UDP : Port destination**

<table>
<thead>
<tr>
<th>Source port</th>
<th>Destination port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segment length</td>
<td>Checksum</td>
</tr>
</tbody>
</table>

- multiplexage à la destination (plusieurs serveurs sur un hôte)
- 16 bits (65K ports)
UDP : Longueur du segment

- longueur totale avec les données exprimée en **octet**
- 16 bits (64 Koctets maximum)
### UDP : Somme de contrôle du segment

- contrôle d’erreur (rudimentaire)
- nécessaire si les couche sous-jacente n’effectuent pas de contrôle
- \textit{pseudoheader}
- \[ \sum \limits_{i=1}^{16} \text{mot}_{16\text{bits}} \]
- 16 bits

<table>
<thead>
<tr>
<th>Source port</th>
<th>Destination port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segment length</td>
<td>Checksum</td>
</tr>
<tr>
<td>Protocol</td>
<td>Segment Length</td>
</tr>
<tr>
<td>IP source address</td>
<td></td>
</tr>
<tr>
<td>IP destination address</td>
<td></td>
</tr>
</tbody>
</table>

- 32 bits (4 octets)
UDP : arguments pour un transport sans connexion

Le choix d’un service transport non connecté peut être motivé par :

- pas d’établissement de connexion
- absence d’état dans les hôtes
- entête réduit
- pas de retransmission
- pas de contrôle du débit d’émission
UDP : exemples d’applications

Les applications suivantes reposent typiquement sur UDP :

- résolution de noms (DNS)
- serveur de fichiers distants (NFS)
- protocole de routage (RIP)
- administration du réseau (SNMP)
- diffusion multimédia, *streaming* (propriétaire)
- téléphonie sur Internet (propriétaire)
- visioconférence, (propriétaire)
- ...

*et toutes les applications multicast*
#include <sys/types.h>
#include <sys/socket.h>

// Create a descriptor
int socket(int domain, int type, int protocol);
// domain : PF_INET for IPv4 Internet Protocols
// type : SOCK_DGRAM Supports datagrams (connectionless,
// unreliable messages of a fixed maximum length)
// protocol : UDP (/etc/protocols)

// Bind local IP and port
int bind(int s, struct sockaddr *my_addr, socklen_t addrlen);

// Send an outgoing datagram to a destination address
int sendto(int s, const void *msg, size_t len, int flags,
            const struct sockaddr *to, socklen_t tolen);

// Receive the next incoming datagram and record is source address
int recvfrom(int s, void *buf, size_t len, int flags,
              struct sockaddr *from, socklen_t *fromlen);

// End : deallocate
int close(int s);
Un protocole de la couche transport en mode orienté connection : TCP
Mode orienté connexion

Caractéristiques associées au mode orienté connexion :

- état (mémoire)
- protocole de gestion de la connexion :
  - création
  - maintient
  - suppression
- bidirectionnel
- services avancés :
  - fiabilité
  - ordonnancement
  - contrôles d’émissions
  - ...

Olivier.Fourmaux@L2ti.univ-paris13.fr 21
Vers un service fiable

La couche transport doit assurer la fiabilité afin d’être indépendante des couches inférieures.

Mécanismes associés à la fiabilisation de la connexion :

- **ARQ** *(Automatic Repeat reQuest)*
  - acquitement positif (ACK) ou négatif (NACK)
  - détection d’erreur
  - retour d’information *(receiver feedback)*
  - retransmission

- fenêtre d’anticipation
  - protocole *stop-and-wait*
  - protocole *pipeline*
    - numéros de séquence
    - acquitement cumulatifs ou sélectifs
    - retransmissions *Go-Back-N* ou sélectives
TCP (1)

SYN
SYN+ACK
ACK

DATA
TCP (2)

Caractéristiques de base :

- connexion **bidirectionnelle** (*full duplex*)
- **point-à-point**
- ouverture en trois échanges (*three-way handshake*)
- orienté **octet**
- taille maximum de segments (bloc d’octets)
  - ✓ MSS (*Maximun Segment Size*) = taille des *data*!
  - ✓ MSS typiques : 1460, 536 et 512
- **structure**...
Datagramme TCP

<table>
<thead>
<tr>
<th>Source port</th>
<th>Destination port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence number</td>
<td></td>
</tr>
<tr>
<td>Acknowledgment number</td>
<td></td>
</tr>
<tr>
<td>Hlen</td>
<td>Rcv window size</td>
</tr>
<tr>
<td>Checksum</td>
<td>Urgent data ptr</td>
</tr>
<tr>
<td>Options</td>
<td></td>
</tr>
<tr>
<td>Message</td>
<td>(application data)</td>
</tr>
</tbody>
</table>
• identification unique de l’application **source**
• **multiplexage** à la source (plusieurs clients vers un serveur)
• 16 bits (65K ports)
**TCP : Port destination**

<table>
<thead>
<tr>
<th>Source port</th>
<th>Destination port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence number</td>
<td></td>
</tr>
<tr>
<td>Acknowledgment number</td>
<td></td>
</tr>
<tr>
<td>Hlen</td>
<td>Rcv window size</td>
</tr>
<tr>
<td>Checksum</td>
<td>Urgent data ptr</td>
</tr>
<tr>
<td>Options</td>
<td></td>
</tr>
</tbody>
</table>

- identification unique de l’application **destination**
- **multiplexage** à la destination (plusieurs serveurs sur un hôte)
- 16 bits (65K ports)
TCP : Numéro de séquence

- associé à chaque **octet** (et non pas à un segment)
- numérote le **premier** octet des **data**
- numérotation implicite des octets suivants
- détection des **pertes**
- **ordonnancement**
- 32 bits (boucle au bout de 4 Goctets)
TCP : Numéro d’acquittement

- indique le numéro du prochain octet attendu
- **cumulatif**, indique le premier octet non reçu (d’autres peuvent avoir été reçus avec des numéros de séquence supérieurs)
- associé à une retransmission
  - *Go-Back-N*
- **piggybacking**
- 32 bits
TCP : Longueur de l’entête

<table>
<thead>
<tr>
<th>Source port</th>
<th>Destination port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence number</td>
<td></td>
</tr>
<tr>
<td>Acknowledgment number</td>
<td></td>
</tr>
<tr>
<td>Hlen</td>
<td>Rcv window size</td>
</tr>
<tr>
<td>Checksum</td>
<td>Urgent data ptr</td>
</tr>
<tr>
<td>Options</td>
<td></td>
</tr>
</tbody>
</table>

- nombre de lignes de 32 bits dans l’entête TCP
- nécessaire car le champ option est de longueur variable (20 à 60 octets)
- valeur de 5 (pas d’options) à 15 (10 lignes d’options, soit 40 octets)
- 4 bits (valeur "0" à 15 maximum)
TCP : Indicateurs (*flags*)

<table>
<thead>
<tr>
<th>Source port</th>
<th>Destination port</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Sequence number</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Acknowledgment number</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Hlen</th>
<th>URG</th>
<th>ACK</th>
<th>PSH</th>
<th>RST</th>
<th>SYN</th>
<th>FIN</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Rcv window size</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Checksum</th>
<th>Urgent data ptr</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Options</th>
</tr>
</thead>
</table>

32 bits (4 octets)  
(min 5 lignes (min 20 octets))

Chacun sur 1 bit indique :

- **URG** : présence de données **urgentes**
- **ACK** : champ **acquittement** valide
- **PSH** : envoi **immédiat** avec vidage des tampons
- **RST** : **terminaison** brutale de la connexion
- **SYN** : synchronisation lors de l’**ouverture**
- **FIN** : **fermeture** courte
TCP : Taille de la fenêtre de réception

<table>
<thead>
<tr>
<th>Source port</th>
<th>Destination port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence number</td>
<td></td>
</tr>
<tr>
<td>Acknowledgment number</td>
<td></td>
</tr>
<tr>
<td>Hlen</td>
<td>Rcv window size</td>
</tr>
<tr>
<td>Checksum</td>
<td>Urgent data ptr</td>
</tr>
<tr>
<td>Options</td>
<td></td>
</tr>
</tbody>
</table>

- indique le nombre d’octet disponible du coté du récepteur
- dimentionne la taille de la fenêtre d’anticipation (coulissante) de l’émetteur
- **contrôle de flux**
- *piggybacking*
- 16 bits (le récepteur annonce 64 Koctets maximum)
TCP : Somme de contrôle du segment

- contrôle d’erreur (rudimentaire)
- nécessaire si les couche sous-jacente n’effectuent pas de contrôle
- *pseudoheader*
  - $\sum \text{mot}_{16\text{bits}}$
  - 16 bits
TCP : Pointeur sur les données urgentes

- permet l’envoi de données **hors bande**
- délimite des données traitées en priorité
- indique la fin des données urgentes (comprisent entre $S_n$ et $S_n + Udptr$)
- 16 bits

<table>
<thead>
<tr>
<th>Source port</th>
<th>Destination port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence number</td>
<td>Acknowledgment number</td>
</tr>
<tr>
<td>Rcv window size</td>
<td>Urgent data ptr</td>
</tr>
<tr>
<td>Checksum</td>
<td>Options</td>
</tr>
</tbody>
</table>

32 bits (4 octets)
### TCP : Options

Les options sont de la forme TLV ou *Type, Length (octets), Value* :
- **END** : fin de liste d’options (T=0)
- **NOP** : pas d’opération, bourrage (T=1)
- **MSS** : négociation du MSS (T=2, L=4, V=MSS)
- **WSIZE** : mise à l’échelle de la fenêtre par une puissance de deux (T=3, L=3, V=puissance)
- **SACK** : demande d’acquittement sélectif à l’ouverture (T=4, L=2)
- **SACK** : acquittement sélectif de *n* blocs (T=5, L=2 + 8n, 2n numéros de séquences)
- ...
TCP : Gestion de la connexion

Création de la connexion (call setup) :

- *three-way handshake*

![Diagram](image)

Deconnexion (release) :

- fermeture unilatérale (*shutdown*)
- fermeture courtoise (*gracefull release*)

*pictures from Tanenbaum A. S. Computer Networks 3rd edition*
TCP : Automate d’états finis

(Start)
CONNECT/SYN
(Step 3 of the three-way handshake)
SYN + ACK/ACK
(Simultaneous open)
SYN/ SYN + ACK
(Data transfer state)
ACK/–
(Active close)
CLOSE/FIN
(Passive Close)
FIN/ACK
(Timeout)
FIN/WAIT 1
ACK/–
FIN/WAIT 2
FIN + ACK/ACK
TIME WAIT
FIN/ACK
CLOSED
(Go back to start)
CLOSE/FIN
CLOSE WAIT
LAST ACK
FIN/ACK
CLOSING
ACK/–
FIN/ACK
ESTABLISHED
SYN SENT
SEND/ SYN
CLOSE/–
SYN/ SYN + ACK
CLOSE/–
LISTEN/–
LISTEN
SYN RCVD
RST/–
ACK/–
CLOSE/FIN
CLOSE/–
CLOSED
FIN/WAIT 1
FIN/WAIT 2
SYN/RCVD
CLOSE/–
CLOSE/–
CLOSED
TIME WAIT
FIN/ ACK
(Active close)
FIN/ ACK
(FIN + ACK/ACK)
(Step 3 of the three-way handshake)
TCP: 1ère fenêtre coulissante

Sliding window:

Application writes data

Data sent and acknowledged

Data sent but not acknowledged

Data not sent but usable

Data not usable

Window size

Ack base

Send base

Ack from the receiver

Data sent to the receiver
TCP : Contrôle de flux

Application does a 2K write

Receiver's buffer

0

4K

Empty

Full

Application reads 2K

Sender is blocked

Sender may send up to 2K

2K SEQ = 0

ACK = 2048 WIN = 2048

ACK = 4096 WIN = 0

ACK = 4096 WIN = 2048

1K SEQ = 4096

2K

1K 2 K

pictures from Tanenbaum A. S. Computer Networks 3rd edition
TCP : Pertes

L’absence d’un paquet au temps voulu :
- principalement lié à la *congestion*
- rarement à une erreur de transmission

➡️ Comment évaluer une perte ?

➡️ Quelle politique de retransmission
TCP : Contrôle de congestion (1)

✓ slow start (exponential increase)
- un MSS en plus à chaque aquittement positif
- au début... jusqu’au threshold

✓ congestion avoidance (additive increase)
- un MSS en plus à chaque RTT
- du threshold... jusqu’à la perte!

✓ multiplicative decrease
- le threshold vaut la moitié du max de la fenêtre
- repart à 0 avec le slow start

Algorithmes de contrôle :
- superposition d’une seconde fenêtre coulissante
- algorithmes de contrôle du débit :

pictures from TANENBAUM A. S. Computer Networks 3rd edition
TCP : Contrôle de congestion (2)

Transmission number

Congestion window (kilobytes)

Timeout
Threshold

pictures from Tanenbaum A. S. Computer Networks 3rd edition
TCP : Temporisations

Gestion de multiples temporisations (*timers*) :

- *retransmission timer* (détecte les pertes)
  - ✓ $timeout = RTT + \beta D$ (avec $\beta$ usuel : 4)
  - ✓ $RTT = \alpha RTT + (1 - \alpha) M$ (avec $\alpha$ usuel : 2)
  - ✓ $D = \alpha D + (1 - \alpha) |RTT - M|$ (écart moyen)
  - ✓ **algorithme de Karn**
    ✐ ne pas tenir compte des packets retransmis et doubler le *timeout* à chaque échec

- *persistence timer* (évite les blocages)
  - ✓ envoi d’un acquittement avec une fenêtre à 0

- *keep alive timer* (vérifie s’il y a toujours un destinataire)

- *closing timer* (terminaison)
Implémentation

- *a trip to Nevada* :
  - **TCP Tahoe**
    - ✔ *slowstart* + *congestion avoidance* + *multiplicative decrease*
    - ✔ décrit précédemment... problème lorsque juste un segment est perdu
  - **TCP Reno**
    - ✔ idem TCP tahoe
    - ✔ + **fast retransmit** (déclenche la retransmission d’un segment après trois acquittement dupliqués, avant le *timeout*)
    - ✔ + **fast recovery** (pas de *slowstart* après un *fast retransmit*)
    - ✔ à voir : [http://www.ce.chalmers.se/~Efcela/tcp-tour.html](http://www.ce.chalmers.se/~Efcela/tcp-tour.html)
    - ✔ TCP Reno est utilisé dans la plupart des OS actuels
  - **TCP Vegas**
    - ✔ évite la congestion en la détectant **avant** les pertes
    - ✔ réduit le débit juste avant la perte en étudiant les variations des RTT
TCP : Interface socket

#include <sys/types.h>
#include <sys/socket.h>

# create a descriptor and bind local IP and port
int socket(int domain, int type, int protocol);
# domain : PF_INET for IPv4 Internet Protocols
# type : SOCK_STREAM Provides sequenced, reliable, two-way, connection-based byte streams.
# An out-of-band data transmission mechanism may be supported.
# protocol : TCP (/etc/protocols)
int bind(int s, struct sockaddr *my_addr, socklen_t addrlen);

# Server : passive queuing mode and connection acceptance
int listen(int s, int backlog);
int accept(int s, struct sockaddr *addr, socklen_t *addrlen);

# Client : active connection
int connect(int sockfd, const struct sockaddr *serv_addr, socklen_t addrlen);

# Send and receive data
int send(int s, const void *msg, size_t len, int flags);
int recv(int s, void *buf, size_t len, int flags);

# End : deallocate
int close(int s);
La couche réseau
La **Couche Réseau** achemine les paquets de la source vers les destinataires en effectuant des sauts entre les différents **nœuds intermédiaires**

- de bout-en-bout (*end-to-end*)
- connaissance de la topologie
- calcul du chemin (*routage*)
- adressage virtuel
- service unique
- abstraction des technologies sous-jacentes
  - encapsulation sur chaque technologie
  - conversion d’adresses
  - fragmentation
Couche Réseaux : OSI

1. Physical
2. Data link
3. Network
4. Transport
5. Session
6. Presentation
7. Application
Couche Réseau : Mode connecté ou non

Packets travel individually and can take different routes

End-to-end concatenated virtual circuits

pictures from Tanenbaum A. S. Computer Networks 3rd edition
Couche Réseau : Fragmentation

(a) G_1 fragments a large packet
G_2 reassembles the fragments
G_3 fragments again
G_4 reassembles again

(b) G_1 fragments a large packet
The fragments are not reassembled until the final destination (a host) is reached

pictures from Tanenbaum A. S. Computer Networks 3rd edition
Couche Réseau : Routage

Calcul du chemin

- initial (circuits virtuels)
- à chaque paquet (sans mémoire)

Décisions de routage basée :

- table de routage
  - statique
  - dynamique
    - protocole de routage
    - algorithme de routage...
Distance Vector Routing

Algorithme distribué de **Ford-Fulkerson**

![Diagram](a)

<table>
<thead>
<tr>
<th></th>
<th>To A</th>
<th>I</th>
<th>H</th>
<th>K</th>
<th>Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>24</td>
<td>20</td>
<td>21</td>
<td>8</td>
</tr>
<tr>
<td>B</td>
<td>12</td>
<td>36</td>
<td>31</td>
<td>28</td>
<td>20</td>
</tr>
<tr>
<td>C</td>
<td>25</td>
<td>18</td>
<td>19</td>
<td>36</td>
<td>28</td>
</tr>
<tr>
<td>D</td>
<td>40</td>
<td>27</td>
<td>8</td>
<td>24</td>
<td>20</td>
</tr>
<tr>
<td>E</td>
<td>14</td>
<td>7</td>
<td>30</td>
<td>22</td>
<td>17</td>
</tr>
<tr>
<td>F</td>
<td>23</td>
<td>20</td>
<td>19</td>
<td>40</td>
<td>30</td>
</tr>
<tr>
<td>G</td>
<td>18</td>
<td>31</td>
<td>6</td>
<td>31</td>
<td>18</td>
</tr>
<tr>
<td>H</td>
<td>17</td>
<td>20</td>
<td>0</td>
<td>19</td>
<td>12</td>
</tr>
<tr>
<td>I</td>
<td>21</td>
<td>0</td>
<td>14</td>
<td>22</td>
<td>10</td>
</tr>
<tr>
<td>J</td>
<td>9</td>
<td>11</td>
<td>7</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>K</td>
<td>24</td>
<td>22</td>
<td>22</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>L</td>
<td>29</td>
<td>33</td>
<td>9</td>
<td>9</td>
<td>15</td>
</tr>
</tbody>
</table>

**New estimated delay from J**

**New routing table for J**

Vectors received from J's four neighbors:

- JA delay is 8
- JI delay is 10
- JH delay is 12
- JK delay is 6

*pictures from Tanenbaum A. S. *Computer Networks 3rd edition*
Couche Réseau : Routage par état des liaisons

*Link State Routing*

Algorithme du plus court chemin de *Dijkstra*

![Diagram](image)

*pictures from Tanenbaum A. S. Computer Networks 3rd edition*
Couche Réseaux : TCP/IP

- TELNET
- FTP
- SMTP
- DNS
- TCP
- UDP
- IP
- ARPANET
- SATNET
- Packet# radio
- LAN
La couche réseau dans Internet : IPv4
IPv4

Routers

Paquets
### IPv4: Structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version (Ver)</td>
<td>32 bits (4 octets)</td>
</tr>
<tr>
<td>Header Length (Hlen)</td>
<td></td>
</tr>
<tr>
<td>Type of Service (TOS)</td>
<td></td>
</tr>
<tr>
<td>Packet Length</td>
<td></td>
</tr>
<tr>
<td>Identifier</td>
<td></td>
</tr>
<tr>
<td>DF</td>
<td></td>
</tr>
<tr>
<td>MF</td>
<td></td>
</tr>
<tr>
<td>Fragment Offset</td>
<td></td>
</tr>
<tr>
<td>Time to Live (TTL)</td>
<td></td>
</tr>
<tr>
<td>Protocol</td>
<td></td>
</tr>
<tr>
<td>Header Checksum</td>
<td></td>
</tr>
<tr>
<td>Source Address</td>
<td></td>
</tr>
<tr>
<td>Destination Address</td>
<td></td>
</tr>
<tr>
<td>Options</td>
<td></td>
</tr>
<tr>
<td>Segment</td>
<td>Segment (transport data)</td>
</tr>
</tbody>
</table>
IPv4 : Version

- IP actuel : version 4
- IP *next génération* : version 6
- 4 bits

<table>
<thead>
<tr>
<th>Ver</th>
<th>Hlen</th>
<th>TOS</th>
<th>Packet Lenght</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identifier</td>
<td>D</td>
<td>R</td>
<td>Frag. offset</td>
</tr>
<tr>
<td>TTL</td>
<td>Protocol</td>
<td>Header checksum</td>
<td></td>
</tr>
<tr>
<td>IP source address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IP destination address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
IPv4 : Longueur de l’entête

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ver</td>
<td>Version</td>
</tr>
<tr>
<td>Hlen</td>
<td>Header length</td>
</tr>
<tr>
<td>TOS</td>
<td>Type of Service</td>
</tr>
<tr>
<td>Packet Lenght</td>
<td>Lenght of the packet</td>
</tr>
<tr>
<td>Identifier</td>
<td>Identifier</td>
</tr>
<tr>
<td>Frag. offset</td>
<td>Fragment offset</td>
</tr>
<tr>
<td>TTL</td>
<td>Time to Live</td>
</tr>
<tr>
<td>Protocol</td>
<td>Protocol</td>
</tr>
<tr>
<td>Header checksum</td>
<td>Checksum</td>
</tr>
<tr>
<td>IP source address</td>
<td>Source address of the IP</td>
</tr>
<tr>
<td>IP destination address</td>
<td>Destination address of the IP</td>
</tr>
<tr>
<td>Options</td>
<td>Options</td>
</tr>
</tbody>
</table>

- nombre de lignes de 32 bits dans l’entête IP
- nécessaire car le champ option est de longueur variable (20 à 60 octets)
- valeur de 5 (pas d’options) à 15 (10 lignes d’options, soit 40 octets)
- 4 bits (valeur ”0” à 15 maximum)
### IPv4 : Type of service

**Type Of Service**

- 8 bits :
  - ✓ 3 bits de priorité (*precedence*)
    - 000 : Routine
    - 001 : Priority
    - 010 : Immediate
    - 011 : Flash
    - 100 : Flash override
    - 110 : Internetwork control
    - 111 : Network control
  - ✓ 3 bits de *service*
    - Delay
    - Throughput
    - Reliability
    - (cost)
  - • non utilisé...
  - ➤ *DiffServ Byte*
## IPv4 : Taille du paquet

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>Packet Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ver</td>
<td>Hlen</td>
<td>TOS</td>
<td></td>
</tr>
<tr>
<td>Identifier</td>
<td>D</td>
<td>M</td>
<td>E</td>
</tr>
<tr>
<td>TTL</td>
<td>Protocol</td>
<td>Header checksum</td>
<td></td>
</tr>
<tr>
<td>IP source address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IP destination address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Segment</td>
<td>(transport data)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- taille totale du paquet **avec entête**
- exprimé en octets
- taille maximum de **576 octets** minimum (512 octets de données + 64 octets de surcoût liée aux protocoles)
- 16 bits (64 Koctets maximum)
**IPv4 : Identificateur**

- défini de manière **unique** pour chaque paquet
- pour réassembler les fragments d’un **même** paquet
- habituellement, **incrément** d’un compteur pour chaque paquet successif
- 16 bits (boucle tous les 64 Kpaquets)

<table>
<thead>
<tr>
<th></th>
<th>Ver</th>
<th>Hlen</th>
<th>TOS</th>
<th>Packet Length</th>
<th>Identifier</th>
<th>DP</th>
<th>DF</th>
<th>FR</th>
<th>TTL</th>
<th>Protocol</th>
<th>Header checksum</th>
<th>IP source address</th>
<th>IP destination address</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**IPv4 : Fragmentation**

<table>
<thead>
<tr>
<th>Ver</th>
<th>Hlen</th>
<th>TOS</th>
<th>Packet Lenght</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Identifier</th>
<th></th>
<th></th>
<th>Frag. offset</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>TTL</th>
<th>Protocol</th>
<th>Header checksum</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>IP source address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IP destination address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Options</th>
</tr>
</thead>
</table>

- 1 bit réservé
- 1 bit DF : *Don’t fragment* (=1 interdit la fragmentation)
- 1 bit MF : *More fragment* (=0 pour le dernier fragment)
- 13 bits *fragment offset* en octets/8 (*shift* 3)

- exemples :
  - 0x0000 paquet entier (*offset*=0)
  - 0x2000 premier fragment (*offset*=0)
  - 0x20A0 fragment central (*offset*=1280)
  - 0x00B0 dernier fragment (*offset*=1408)
- ...
IPv4 : Fragmentation

Number of the first elementary fragment in this packet

Packet number → End of packet bit

(a)

(b)

(c)

pictures from Tanenbaum A. S. Computer Networks 3rd edition
### IPv4 : Temps de vie

**Time To Live**

- unité initiale : **seconde**
- valeur maximum fixé par l’émetteur (255, 128, 64...)
- décrément dans chaque routeur
- minimum 1 par routeur
  - nombre de **sauts**
- évite les **boucles**
- 8 bits (max 255 secondes ou sauts)

<table>
<thead>
<tr>
<th>Ver</th>
<th>Hlen</th>
<th>TOS</th>
<th>Packet Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identifier</td>
<td>D</td>
<td>F</td>
<td>Frag. offset</td>
</tr>
<tr>
<td>TTL</td>
<td>Protocol</td>
<td>Header checksum</td>
<td></td>
</tr>
</tbody>
</table>

*IP source address*

*IP destination address*

*Options*
IPv4 : Protocole transporté

<table>
<thead>
<tr>
<th>Ver</th>
<th>Hlen</th>
<th>TOS</th>
<th>Packet Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Protocol</th>
<th>Header checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TTL</th>
<th>Protocol</th>
<th>IP source address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IP destination address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

- **démultiplexage vers les protocoles de la couche supérieure** :

  ```
  Unix> cat /etc/protocols
  icmp 1  # internet control message protocol
  ggp 3   # gateway-gateway protocol
  ipencap 4 # IP encapsulated in IP
  st 5    # ST datagram mode
  tcp 6   # transmission control protocol
  egp 8   # exterior gateway protocol
  udp 17  # user datagram protocol
  rdp 27  # "reliable datagram" protocol
  iso-tp4 29 # ISO Transport Protocol class 4
  xtp 36  # Xpress Transfer Protocol
  idrp 45 # Inter-Domain Routing Protocol
  rsdp 46 # Reservation Protocol
  gre 47 # General Routing Encapsulation
  ospf 89 # Open Shortest Path First IGP
  ...
  ```

- **8 bits**
IPv4 : Contrôle d’erreur dur l’entête

- contrôle d’erreur sur l’entête
- vérifie si le paquet a été bien traité
- $\sum mot_{16\text{bits}}$
- recalculé à la sortie de chaque routeur
- si faux, paquet détruit !
- 16 bits

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ver</td>
<td>Version</td>
</tr>
<tr>
<td>Hlen</td>
<td>Header length</td>
</tr>
<tr>
<td>TOS</td>
<td>Type of service</td>
</tr>
<tr>
<td>Packet Lenght</td>
<td></td>
</tr>
<tr>
<td>Identifier</td>
<td></td>
</tr>
<tr>
<td>Frag. offset</td>
<td></td>
</tr>
<tr>
<td>TTL</td>
<td>Time to live</td>
</tr>
<tr>
<td>Protocol</td>
<td></td>
</tr>
<tr>
<td>Header checksum</td>
<td></td>
</tr>
<tr>
<td>IP source address</td>
<td></td>
</tr>
<tr>
<td>IP destination address</td>
<td></td>
</tr>
<tr>
<td>Options</td>
<td></td>
</tr>
</tbody>
</table>
### IPv4 : Adresse source

- Adresse IP 32 bits
- Indique le réseau du destinataire
- Identifie l’interface du destinataire dans le réseau

<table>
<thead>
<tr>
<th>Ver</th>
<th>Hlen</th>
<th>TOS</th>
<th>Packet Length</th>
<th>Identifier</th>
<th>DP</th>
<th>Frag. offset</th>
<th>TTL</th>
<th>Protocol</th>
<th>Header checksum</th>
<th>IP source address</th>
<th>IP destination address</th>
<th>Options</th>
</tr>
</thead>
</table>
### IPv4 : Adresse destination

<table>
<thead>
<tr>
<th>Ver</th>
<th>Hlen</th>
<th>TOS</th>
<th>Packet Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>K</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- adresse IP 32 bits
- identifie l’émetteur du paquet
- permet de retourner un message à l’émetteur (ICMP, UDP...)

**Diagram:**

- **IP source address**
- **IP destination address**
- **Options**
### IPv4 : Options

<table>
<thead>
<tr>
<th>Ver</th>
<th>Hlen</th>
<th>TOS</th>
<th>Packet Length</th>
<th>32 bits (4 octets)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Identifier</th>
<th>D</th>
<th>F</th>
<th>Frag. offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTL</td>
<td>Protocol</td>
<td>Header checksum</td>
<td></td>
</tr>
</tbody>
</table>

- IP source address
- IP destination address
- Options

- système TLV identique à TCP
- analysés dans **chaque routeur**
- exemple :
  - enregistrement de la route
  - routage à la source strict
  - routage à la source lâche
  - estampilles temporelles
  - sécurité
  - ...

- 0 à 40 octets (alignés sur 32 bits)
## Adressage: Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Network</th>
<th>Host</th>
<th>Range of host addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td></td>
<td>1.0.0.0 to 127.255.255.255</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td></td>
<td>128.0.0.0 to 191.255.255.255</td>
</tr>
<tr>
<td>C</td>
<td>110</td>
<td></td>
<td>192.0.0.0 to 223.255.255.255</td>
</tr>
<tr>
<td>D</td>
<td>1110</td>
<td></td>
<td>224.0.0.0 to 239.255.255.255</td>
</tr>
<tr>
<td>E</td>
<td>11110</td>
<td></td>
<td>240.0.0.0 to 247.255.255.255</td>
</tr>
</tbody>
</table>

Multicast address: 224.0.0.0 to 239.255.255.255

Reserved for future use: 240.0.0.0 to 247.255.255.255

*Pictures from Tanenbaum A. S. *Computer Networks 3rd edition*
Adressage : Subnetting

Taille de l’identifiant de réseau (*netid*) par défaut :

- classe A : /8 (255.0.0.0)
- classe B : /16 (255.255.0.0)
- classe C : /24 (255.255.255.0)

Subdivision possible :

- 10.1.2.0 *netmask* 255.255.255.0 (notation par *masque* de réseau)
- 132.77.12.0/22 (notation par *préfixe*)

---

**Diagram:**

- 32 Bits

<table>
<thead>
<tr>
<th>Subnet mask</th>
<th>Network</th>
<th>Subnet</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
</tbody>
</table>

*Pictures from TANENBAUM A. S., Computer Networks 3rd edition*
## Adressage : Adresses particulières

### Adresses particulières :

<table>
<thead>
<tr>
<th>Network</th>
<th>1111</th>
<th>...</th>
<th>1111</th>
<th>127</th>
<th>(Anything)</th>
<th>Host</th>
<th>00</th>
<th>...</th>
<th>00</th>
<th>00000000000000000000000000000000</th>
</tr>
</thead>
<tbody>
<tr>
<td>This host</td>
<td>A host on this network</td>
<td>Broadcast on the local network</td>
<td>Broadcast on a distant network</td>
<td>Loopback</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Adresses non routées :

- 10.0.0.0/8
- 172.16.0.0/12
- 192.168.0.0/16
Adressage : affectation

192.33.1.0

56.0.0.0

10.0.0.0

140.18.0.0

140.18.1.0

140.18.2.0

140.18.39.0

140.18.11.0

140.18.21.0

140.18.22.0

140.18.45.0

56.17.45.0

56.1.0.0

56.1.3.0

56.1.4.0

56.1.5.0

56.1.6.0

56.1.7.0

56.2.0.0

56.2.2.0

56.2.8.0

56.17.8.0

56.2.21.0
Adressage : Superneting

CIDR (Classless InterDomain Routing)

Agrégation d’adresses :
• les adresses :
  ✓ 192.33.16.0/24
  ✓ 192.33.17.0/24
  ✓ 192.33.18.0/24
  ✓ 192.33.19.0/24
• peuvent entre regroupées en :
  ✓ notation par **préfixe** : 192.33.16.0/22
  ✓ notation par **masque** : 192.33.16.0 netmask 255.255.252.0
IPv4 : Routage

Table de routage : *longuest préfix match*

Unix> route -n

Kernel IP routing table

<table>
<thead>
<tr>
<th>Destination</th>
<th>Gateway</th>
<th>Genmask</th>
<th>Flags</th>
<th>Metric</th>
<th>Ref</th>
<th>Use</th>
<th>Iface</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.33.182.0</td>
<td>0.0.0.0</td>
<td>255.255.255.0</td>
<td>U</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>eth0</td>
</tr>
<tr>
<td>10.0.0.0</td>
<td>0.0.0.0</td>
<td>255.0.0.0</td>
<td>U</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>atm0</td>
</tr>
<tr>
<td>154.18.2.0</td>
<td>0.0.0.0</td>
<td>255.255.255.0</td>
<td>U</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>eth1</td>
</tr>
<tr>
<td>132.77.0.0</td>
<td>154.18.2.254</td>
<td>255.255.0.0</td>
<td>UG</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>eth1</td>
</tr>
<tr>
<td>default</td>
<td>192.33.182.254</td>
<td>0.0.0.0</td>
<td>UG</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>eth0</td>
</tr>
</tbody>
</table>

Algorithme de routage :

- routage interne (IGRP) dans un système autonome (AS)
  - vecteur de distance (RIP...)
  - état des liaisons (OSPF, EIGRP...)

- routage externe (EGRP) entre AS
  - vecteur de chemin (BGP...)

Olivier.Fourmaux@L2ti.univ-paris13.fr 76
**IPv4 : ICMP**

*Internet Control Message Protocol* (RFC792)

- encapsulé dans un paquet IP (mais appartient à la couche 3)
- test et diagnostique du réseau :

<table>
<thead>
<tr>
<th>ICMP Type</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>echo reply</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>destination network unreachable</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>destination host unreachable</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>destination protocol unreachable</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>destination port unreachable</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>destination network unknown</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>destination host unknown</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>source quench</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>echo request</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>router advertisement</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>router discovery</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>TTL expired</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>reassembly time exceeded</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>IP header bad</td>
</tr>
</tbody>
</table>
IPv4 : Blocage ou modification des adresses

Network Address Translation (NAT)

Firewall...
Couche IPv4 : Internetworking

Leased lines to Asia

US backbone

Leased transatlantic line

European backbone

Regional network

IP token bus LAN

A

IP token ring LAN

C

Tunnel

SNA network

IP Router

National network

Host B

IP Ethernet LAN

1

D

2

pictures from Tanenbaum A. S. Computer Networks 3rd edition
Fin

Document réalisé avec \LaTeX.
Généré le 4 octobre 2002.
Classe de document foils.
Dessins réalisés avec xfig.

Olivier Fourmaux, Olivier.Fourmaux@L2ti.univ-paris13.fr
http://www-L2ti.univ-paris13.fr/~fourmaux