- Protocoles UDP & TCP -

Communications entre
applications via TCP/IP.

Problématique de transport

La couche transport (couche 4) du modèle OSI a pour objectif de compléter l'activité de la couche réseau en réalisant des opérations de fragmentation, de dé fragmentation et de contrôle d'erreur ; afin d'offrir une interface de programmation exploitable par les applications présentent sur la machine.

La couche transport d'Internet gère deux protocoles :

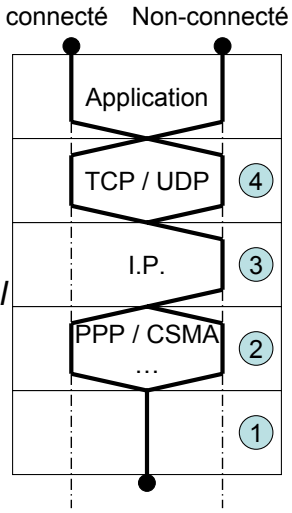
Interfaces avec les applications

Modèle UDP : *User Datagram Protocol*

- Modèle de transmission sans connexion ;
- Pas de contrôle de séquence ;
- Pas de contrôle de flux ;
- Transport sous la forme d'un paquet IP (⇔ tableau d'octet limité).

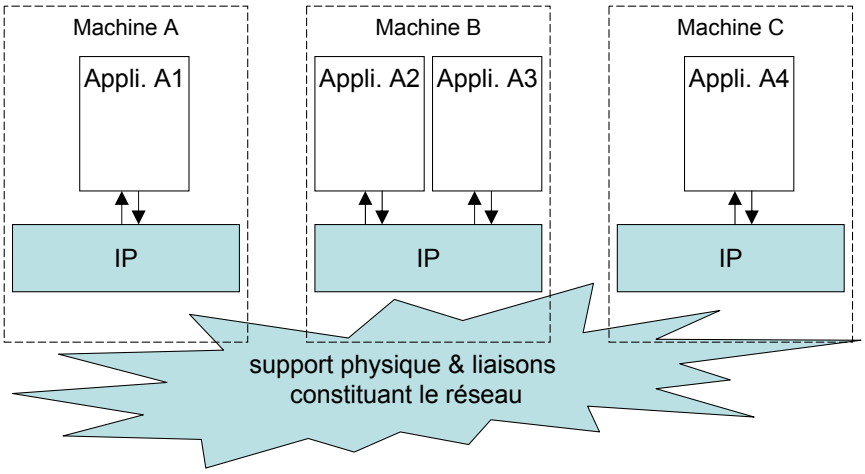
Modèle TCP : *Transmission Control Protocol*

- Modèle de transmission avec connexion ;
- Gestion de la séquence ;
- Gestion du flux ;
- Restitution sous la forme d'un flot (⇔ accès séquentiel d'un fichier).



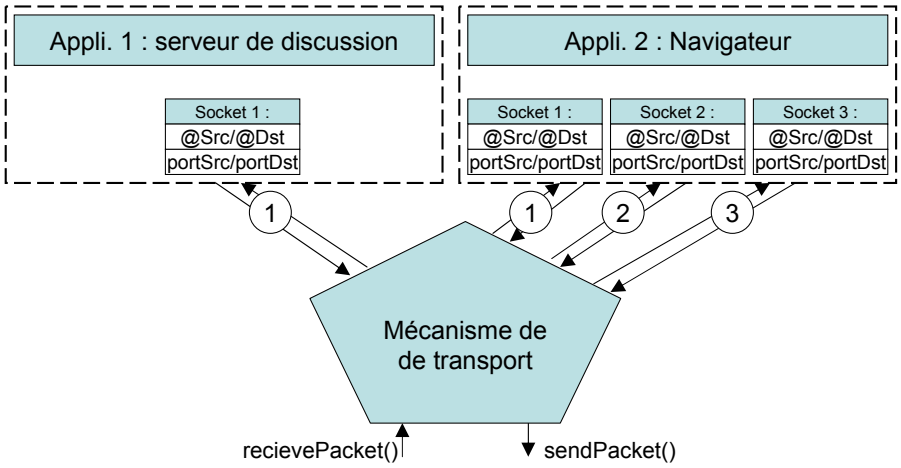
Support IP

```
void sendPacket(IPPacket p);
void receivePacket(IPPacket p);
```



Principe du transport d'informations au dessus d'IP

« routage virtuel » entre les différentes applications présentes sur la machine.



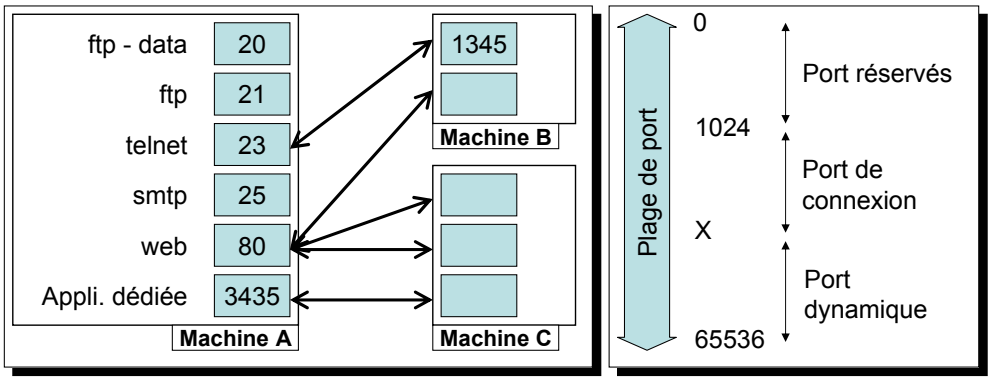
Port de connexion

- Un port identifie un service :

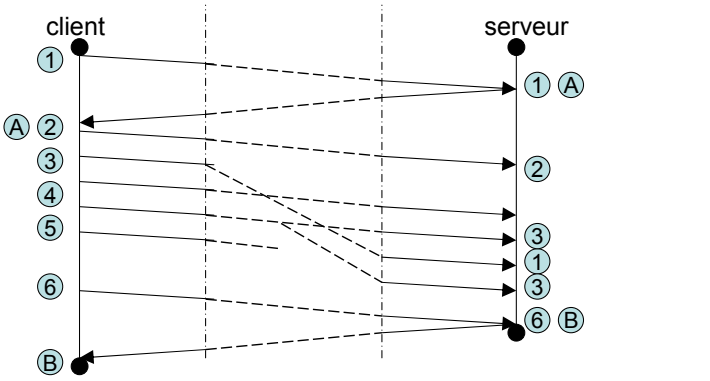
  ⇒ Port réservé / ports alloués dynamiquement.

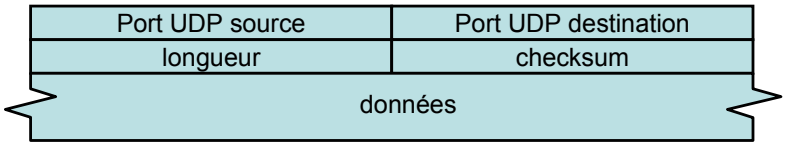
- Une liaison identifiée par 2 @IP + 2 ports :

  ⇒ port du client / port du serveur.



Fonctionnement UDP[1]





Paquet UDP

[1] UDP : *User Datagramm Protocol*

émission d'un paquet UDP
– avec Java –

```
DatagramPacket p;
DatagramSocket s;
InetAddress dst = InetAddress.getByName("brigant");
int port = 1024 ;
…
p = new DatagramPacket p(new byte[100],100, dst,
  port);
s = new DatagramSocket(/*ou port local spécifié*/);
…
s.send(p);
…
Byte [] receivedData d = p.getData();
```

Réception d'un paquet UDP – avec Java –

```
import java.net.*;
import java.io.*;
…
DatagramSocket s;
DatagramPacket p;
…
s = DatagramSocket(/*port*/ 1024);
p = DatagramPacket(/*buffer*/ new byte[512],512);
…
sock.recieve(p);
System.out.println("paquet reçu de :"+ p.getAddress()+
                   "port "+            p.getPort()+
                  "taille" +          p.getLength());
…
s.close();
```

Exploitation UDP
émission – en C –

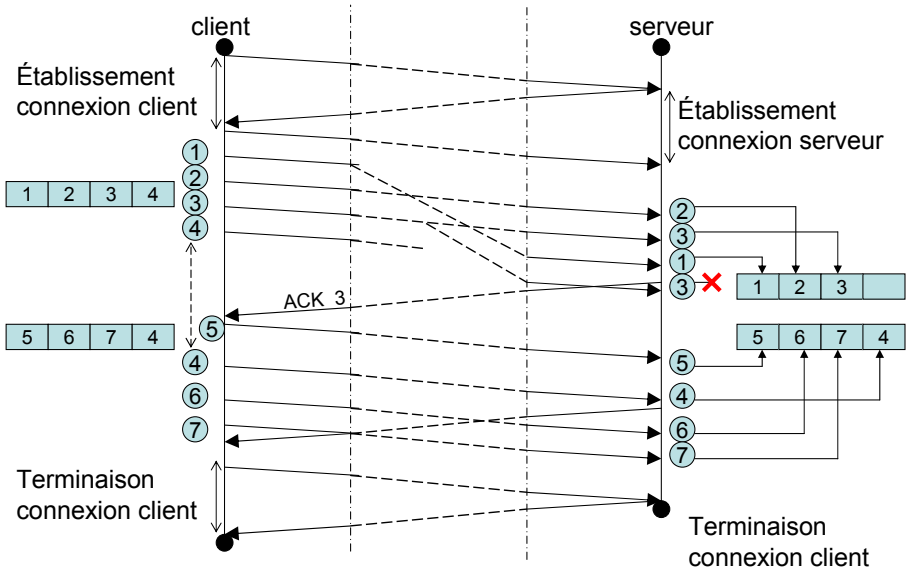
Librairies nécessaires :
<sys/socket.h>
<netinet/in.h>
<netdb.h>

```
int sock ;          // identifiant de socket
sockaddr_in name ;  // @ socket internet
hostent *hp ;       // identifiant IP
…
sock=socket( AF_INET, SOCK_DGRAM,0);     // sock négatif ⇒ erreur
…
hp = gethostbyname("brigant.lifl.fr");   //convertir @alphanu ⇒ @IP
…
memcopy((char *)hp->h_addr,(char *)&name.sin_addr,hp->h_length);
name.sin_family = AF_INET;
name.sin_port = htons(5432);
…
gethostname(buf,max_buf);          // obtenir le nom de la machine locale.
…
sendto(sock, buf, sizeof_buf, offset,(struct sockaddr*)&name,
       sizeof(name) ;              //  envoie du paquet buf vers name.
…
close(sock);                       // fermeture de la liaison.
```

Exploitation UDP
réception – en C –

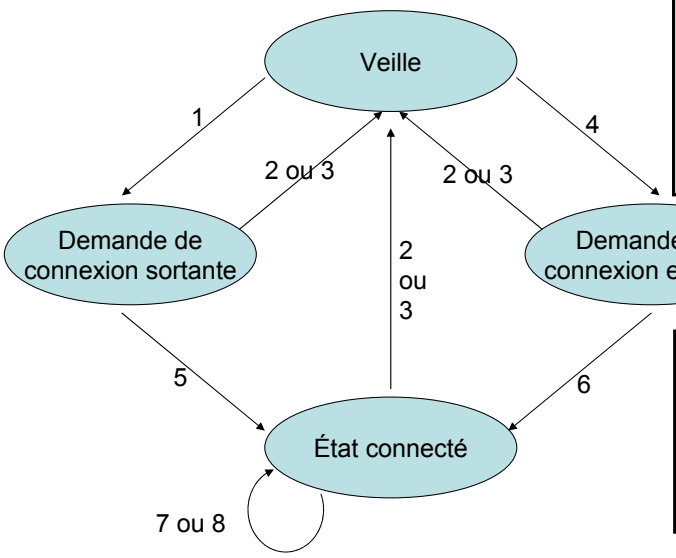
```
Struct sockaddr_in adrSrv;                  // structure pour adresseIP
int s;                                      // identifiant de socket
…
adrSrv.sin_familly = AF_INET;               // initialisation de l'adresse
adrSrv.sin_adr.s_addr = INADDR_ANY;
adrS=erv.sin_port = htons(/* numero de port*/);
…
If(s = socket(AF_INET,SOCK_DGRAM,0)) <0)  // création de la socket
// erreur de création de socket.
…
                                            // liaison entre la socket et le système
if(bind(s,(struct sockaddr *)&adrSrv, sizeof(struct sockaddr_in))<0)
// erreur de liaison de socket.
…
                                            // émission et réception de paquets
Struct sockaddr_in addressClient;
Recvfrom(/*socket*/s, /*buffer de réception*/tampon, /*taille tampon*/ MAX_T,
         /*offset*/0, /*adresse client*/ &addressClient, &taille);
```

Fonctionnement TCP[1]



[1] TCP : *Transport Control Protocol*

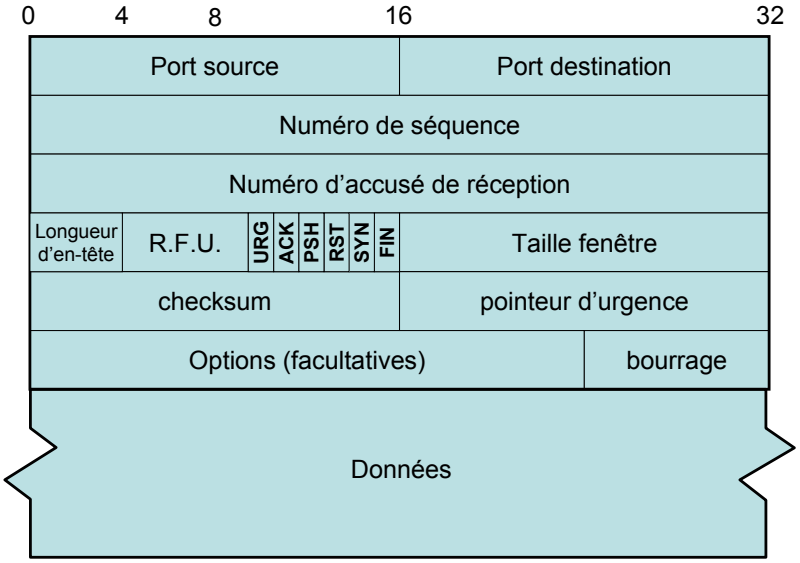
Automate de connexion de transport



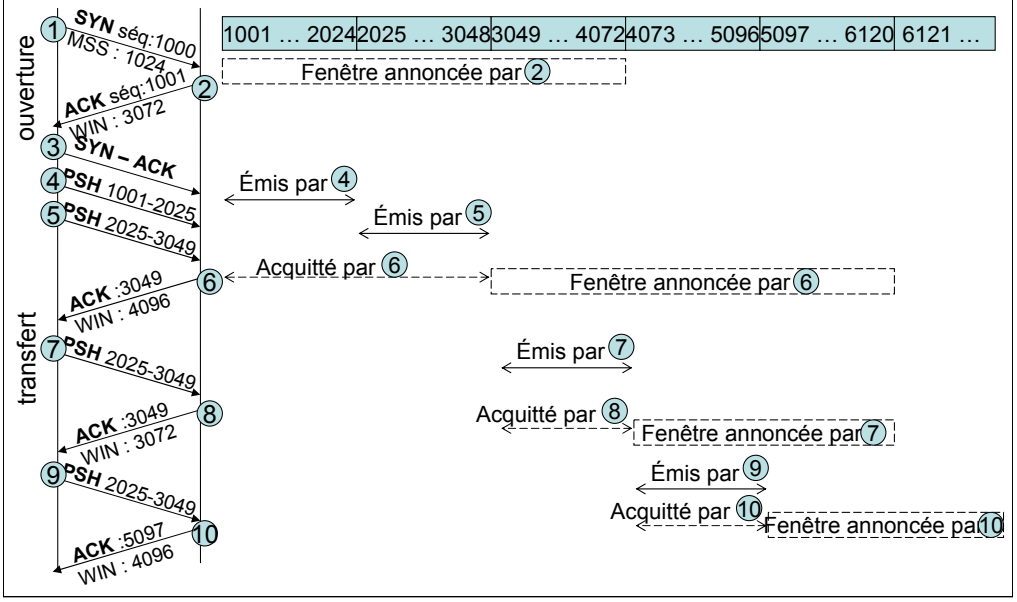
1. Requête de connexion :
en provenance de l'utilisateur
2. Indication déconnexion :
reçu par le fournisseur de service
3. Demande de déconnexion :
en provenance de l'utilisateur
4. Indication de connexion :
envoyer du fournisseur de service.

5. Connexion confirmé :
en provenance du fournisseur.
6. Réponse à la connexion :
en provenance de l'utilisateur
7. Requête de donnée :
en provenance de l'utilisateur
8. Entrée de donnée :
en provenance du fournisseur.

Paquet TCP


0
4
8
16
32
Port source
Port destination
Numéro de séquence
Numéro d'accusé de réception
Longueur d'en-tête
R.F.U.
URG
ACK
PSH
RST
SYN
FIN
Taille fenêtre
checksum
pointeur d'urgence
Options (facultatives)
bourrage
Données

Fenêtre glissante de TCP



Exploitation TCP
coté client – avec Java –

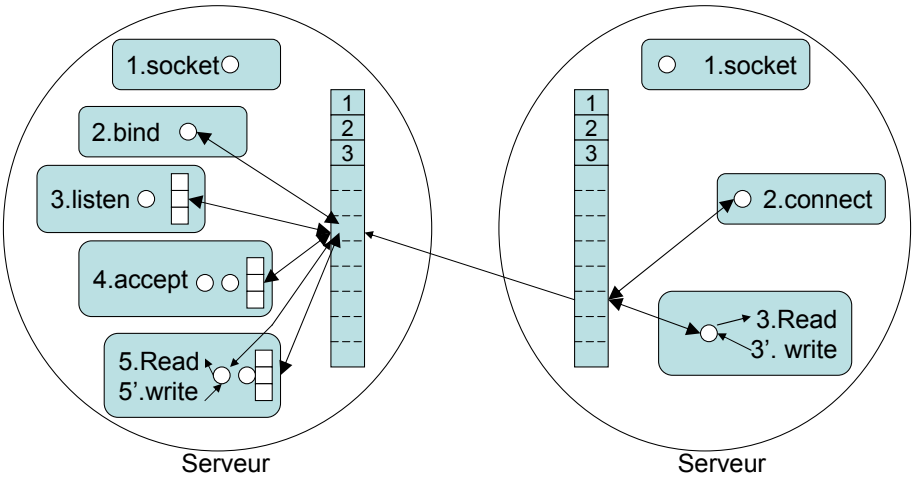
```
socket s;                                  // déclaration de la liaison s.
inputStream iS;                            // déclaration du flux d'acquisition.
outputStream oS;                           // déclaration du flux d'émission.
bufferedReader bF;                         // déclaration du support texte.
int v;
String str;
…
s = new socket("134.206.11.6",765);        // établissement de la liaison s.
iS = s.GetInputStream();                   // obtention  d'un flux d'acquisition.
oS = s.GetInputStream();                   // obtention  d'un flux d'émission.
bF = new BufferedReader(new InputStreamReader(is)); // saisie texte.
…
v=iS.read();                               // lecture d'un octet.
Os.write(v);                               // écriture d'un octet.
…
str = bF.readLine();                       // lecture d'une ligne de texte.
s.close();                                 // fermeture de la liaison.
```

Exploitation TCP
coté serveur – avec Java –

```
import java.io.*;
import java.net.*;
…
ServerSocket sose;  // serveur de socket.
Socket       sock;  // socket une fois la liaison établie.
…
sose = new ServerSocket(7654); // création d'un serveur sur 7654.
sock = sose.accept();          // attente de connexion.
// ici on dispose d'une socket identique à celle du client.
…
InputStream  in   = sock.getInputStream();
OutputStream out  = sock.getOutputStream();
idx = in.read();
…
```

Applications TCP/IP en C

Schéma de principe



Exploitation TCP
coté client – en C –

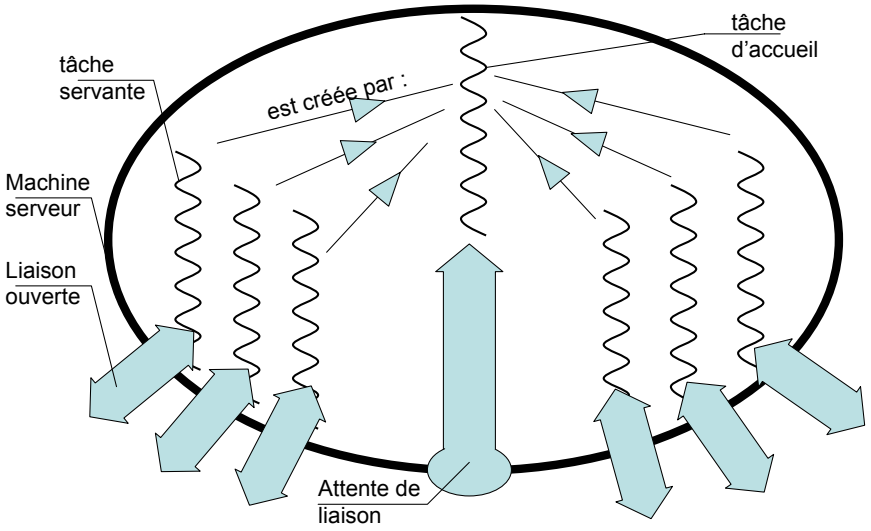
```
Struct sockaddr_in sadr,adr;
int s,s2,taille;
…
s = socket(PF_INET, SOCK_STREAM, 0);
…
// Initialiser sadr
sadr.sin_familly = AF_INET;
sadr.sin_addr = getByName("brigant.lifl.fr");
sadr.sin_port = htons(7654);
…
if(connect(s,(struct sockaddr *)&sadr,sizeof(sadr))<0)
   // erreur à la connexion ;
…
taille = read(s2,tampon,MAX_TAMPON) ;    // valeur négative ⇒ erreur
taille = write(s2,tampon,tailleA) ;      // taille != tailleA ⇒ erreur.
…
```

Exploitation TCP
coté serveur – en C –

```
Struct sockaddr_in sadr,adr;
int s,s2,taille;
…
s = socket(PF_INET, SOCK_STREAM, 0);
…
// Initialiser sadr
…
bind(df,(struct sockaddr *)&sadr,sizeof(sadr));
…
listen(df,MAX_CONNEXIONS); // initialisation de l'écoute.
…
s2 = accept(df,(struct sockaddr *)&cadr,&taille); // attente de
    connexion extérieure (S2 négatif ⇒ erreur).
…
taille = read(s2,tampon,MAX_TAMPON) ;    // valeur négative ⇒ erreur
taille = write(s2,tampon,tailleA) ;      // taille != tailleA ⇒ erreur.
```

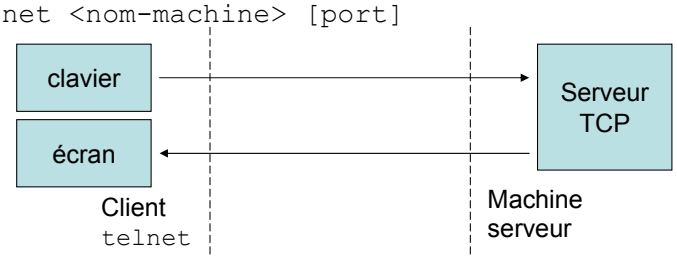
Architecture d'un serveur réseau

Gérer des connexions multiples.



Outil socket générique

```
>telnet <nom-machine>[:port]
>telnet <nom-machine> [port]
```




```
/etc/services
nom          port/type   protocole  # commentaire
…
ftp-data      21 / TCP
…
www           80 / TCP   http       # World Wide Web protocol
```