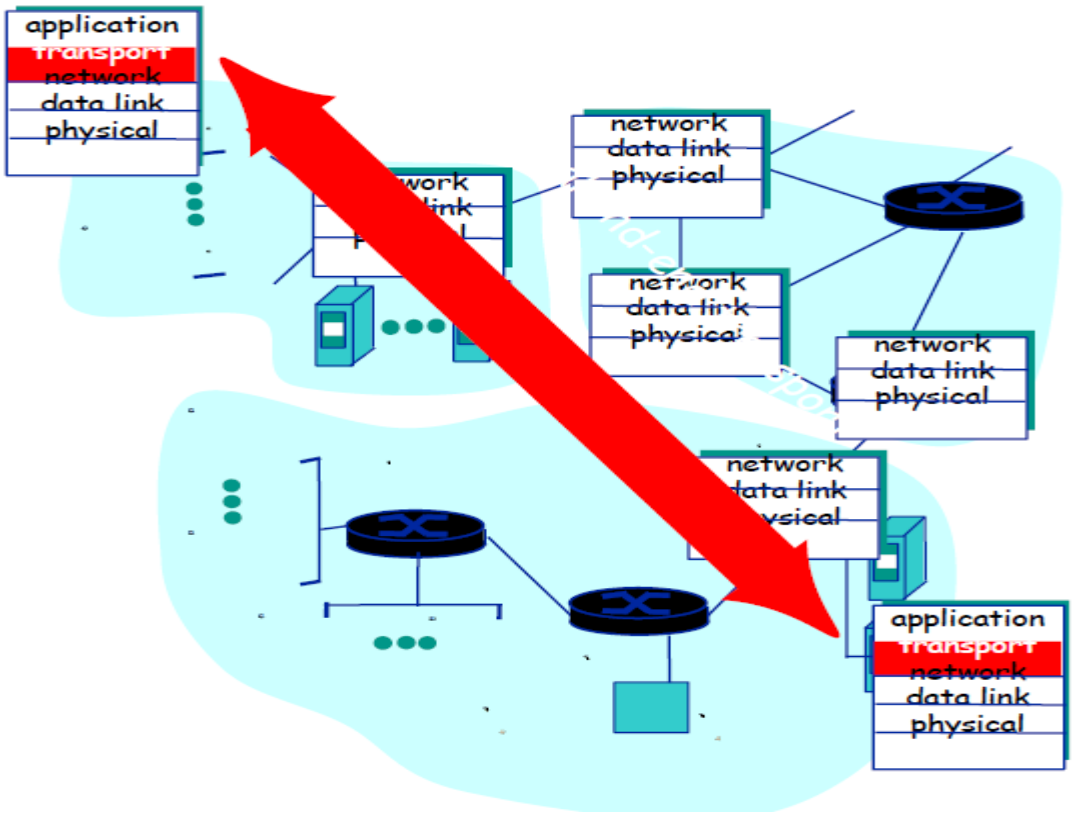
Faculté des Sciences Exactes
2ème L. Informatique

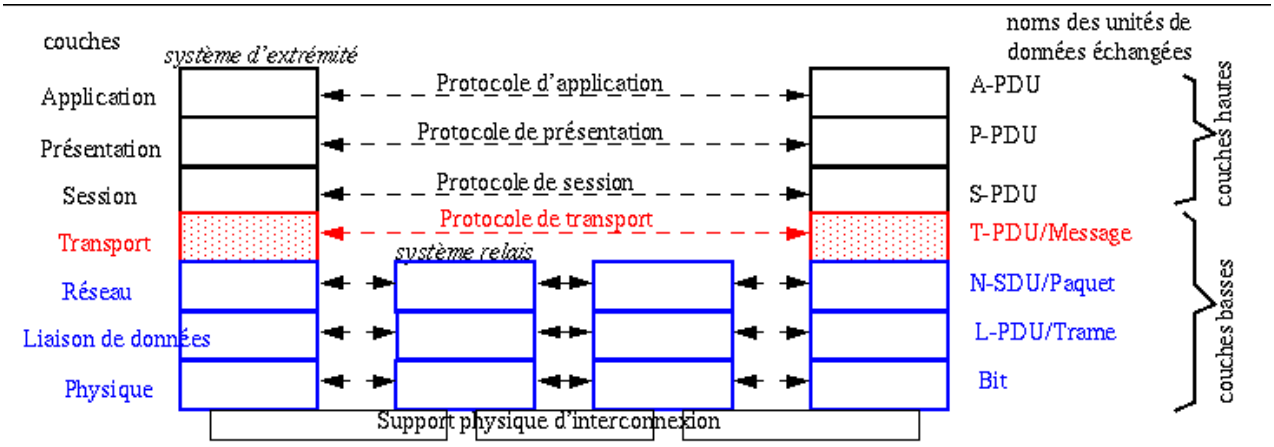
Module : Réseaux

Partie N°06 :

Fonctionnalités de la couche Transport : Communications de bout en bout ,les protocoles TCP et UDP , en mode connecté et en mode non connecté , numéros de ports, multiplexage, les sockets , segmentation, contrôle d'erreur, contrôle de flux, illustration avec Fonctionnement des protocoles UDP et TCP .



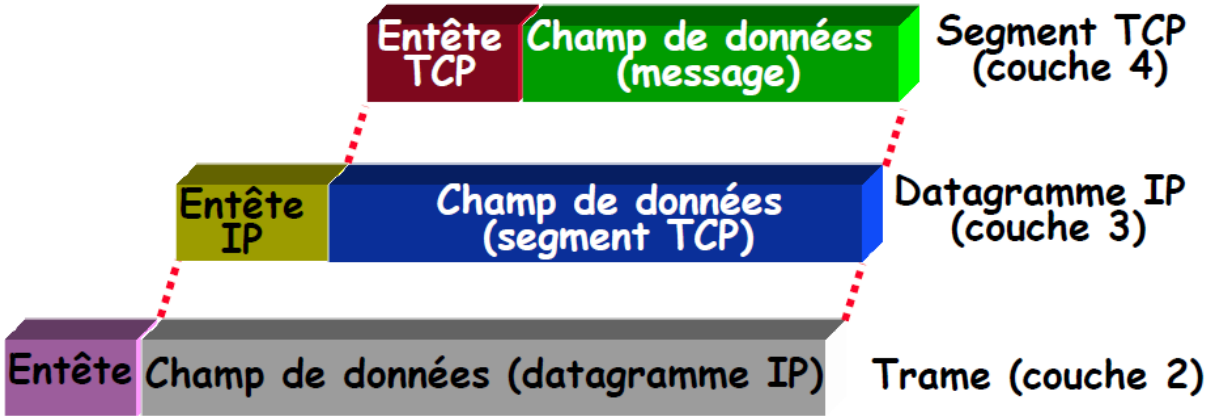
La couche **transport** constitue la quatrième couche du modèle OSI.



La couche transport gère les communications **de bout en bout** entre **processus**. **Cette couche est souvent la plus haute couche où on se préoccupe de la correction des erreurs**. C'est-à-dire que **le service de niveau transport consiste généralement en un service en mode connecté** offrant le transfert de messages ou d'octets bruts garantis sans corruption, pertes, réordonnancement, duplication. En particulier, c'est le service offert par les protocoles TCP .

Les protocoles de transport tournent dans les hôtes (Les machines), la couche transport fourni une communication logique entre processus tournant sur des hosts (machines) différents.

La couche transport : transfert de données entre processus fonde sur les services (de couche) réseau



Segment = unité de données échangées entre entités de couche transport = TPDU: Transport Protocol Data Unit

Rôle et défis du Transport

Prendre en compte le service fourni par la couche réseau :

- Pertes de paquets
- Désequencements
- Duplications

- Erreurs
- Temps de traversée imprévisibles

Prendre en compte les besoins des applications :

- Garantie de remise des méssage
- Séquencement
- Absence de duplications
- Absence de messages erronés
- Messages de long quelconque
- Synchronisation entre l'émetteur et le récepteur
- Contrôle de flux par le récepteur sur l'émetteur
- Support de plusieurs applications sur le même hôte

Protocoles de transport

Services de transport :

Livraison fiable, séquencée et point à point (**TCP**)

- Mode connecté
- Contrôle de congestion
- Contrôle de flux

Livraison non fiable ("besteffort"), non séquencée, point à point ou multicast (**UDP : User Datagram Protocol**)

Service "best effort" : les segments UDP peuvent être perdus ou désequences

- Pas de contrôle de congestion

Service en mode non connecte

- Pas d'établissement de connexion entre host émetteur et host récepteur => simplicité
- Chaque segment UDP est achemine indépendamment des autres (pour la même communication) => desequencement

Services non disponibles :

- Temps réel
- Multicast fiable

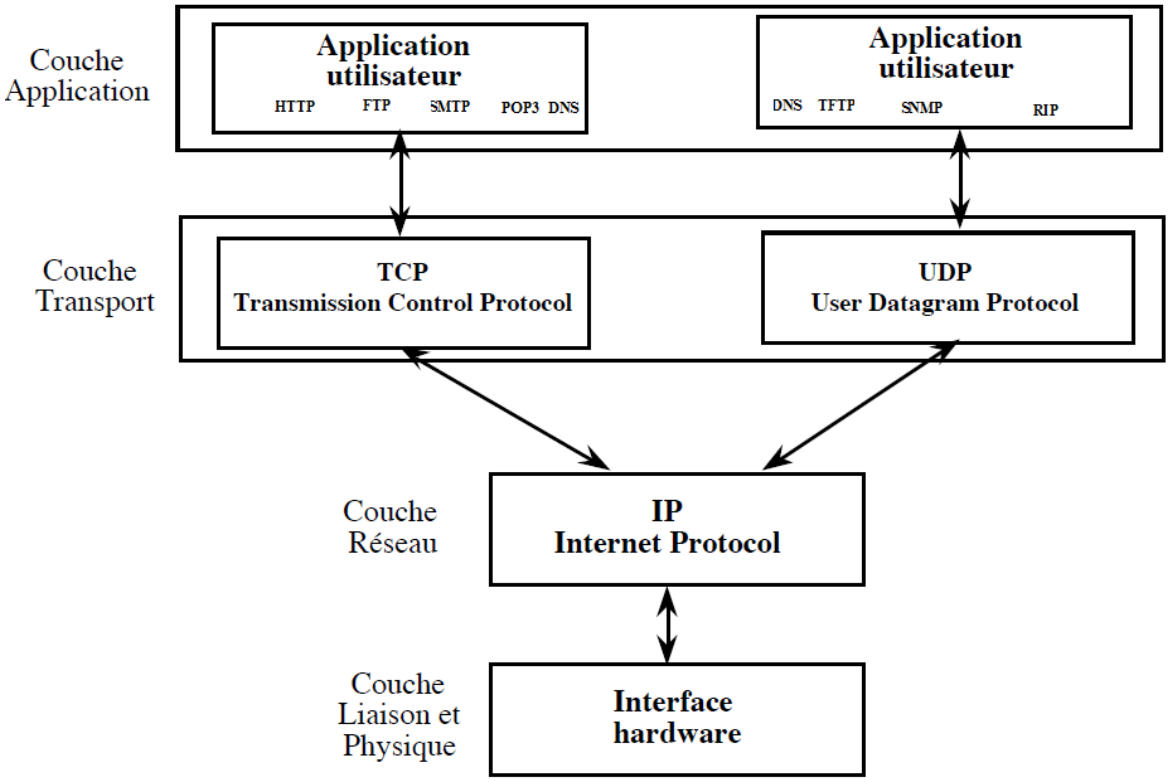
Le protocole UDP permet aux applications d'accéder directement à un service de transmission de datagrammes, tel que le service de transmission qu'offre IP.

Caractéristiques d'UDP :

- UDP possède un mécanisme permettant d'identifier les processus d'application à l'aide de numéros de port UDP.
- UDP est orienté datagrammes (sans connexion), ce qui évite les problèmes liés à l'ouverture, au maintien et à la fermeture des connexions.
- UDP est efficace pour les applications en diffusion/multidiffusion. Les applications satisfaisant à un modèle du type « interrogation-réponse » peuvent également utiliser UDP. La réponse peut être utilisée comme étant un accusé de réception positif à l'interrogation. Si une réponse n'est pas reçue dans un certain intervalle de temps, l'application envoie simplement une autre interrogation.
- UDP ne séquence pas les données. La remise conforme des données n'est pas garantie.
- UDP peut éventuellement vérifier l'intégrité des données (et des données seulement) avec un total de contrôle.
- UDP est plus rapide, plus simple et plus efficace que TCP mais il est moins robuste.

Le protocole UDP permet une transmission sans connexion, mais aussi sans sécurité. Pourtant de nombreuses applications reposent sur UDP :

- TFTP
- DNS
- NFS
- SNMP
- RIP



Multiplexage démultiplexage

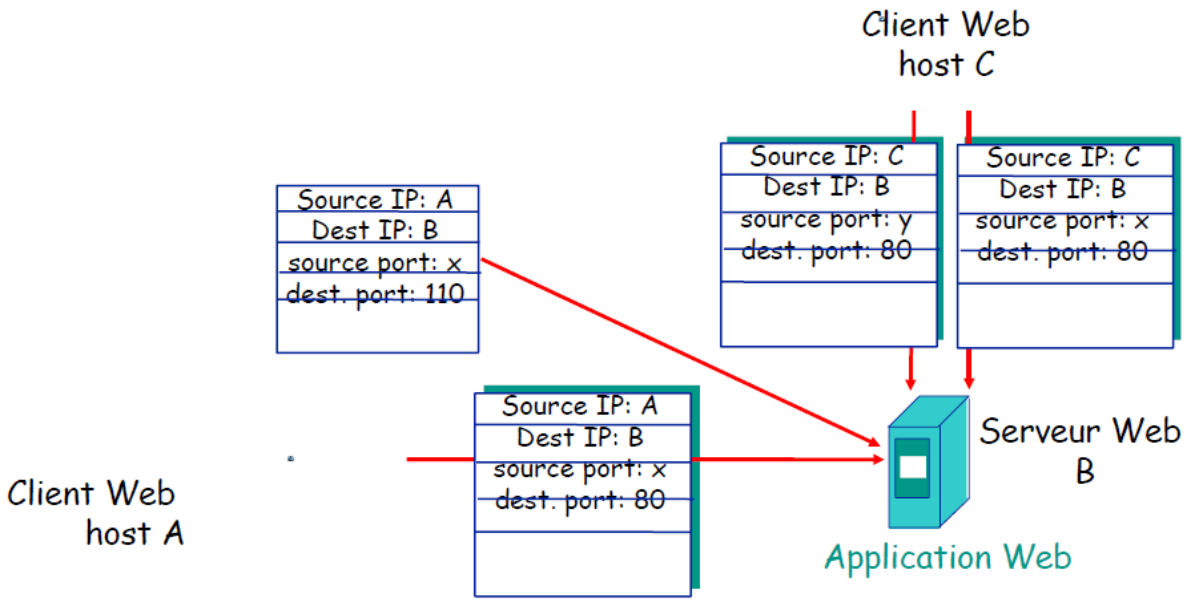
Le but :

Collecte des données envoyées par les différents processus et encapsulation dans des segments envoyés au medium.

- plusieurs applications tournent et émettent des données
- Il faut pouvoir les envoyer par le même canal
- service de **multiplexage**

Livraison des segments reçus aux bons processus

- plusieurs applications tournent simultanément sur un même host
- Il faut pouvoir les identifier de façon non ambiguë
- service de **démultiplexage**

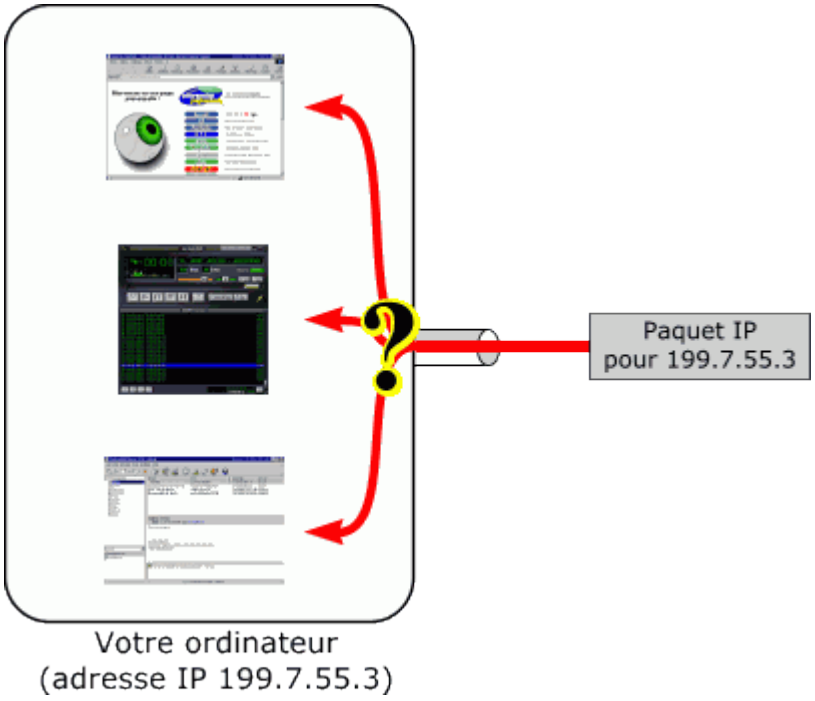


Les ports :

Avec IP, nous avons de quoi envoyer et recevoir des paquets de données d'un ordinateur à l'autre.

Imaginons maintenant que nous ayons plusieurs programmes qui fonctionnent en même temps sur le même ordinateur: un navigateur, un logiciel d'email et un logiciel pour écouter la radio sur Internet.

Si l'ordinateur reçoit un paquet IP, comment savoir à quel logiciel donner ce paquet IP ?

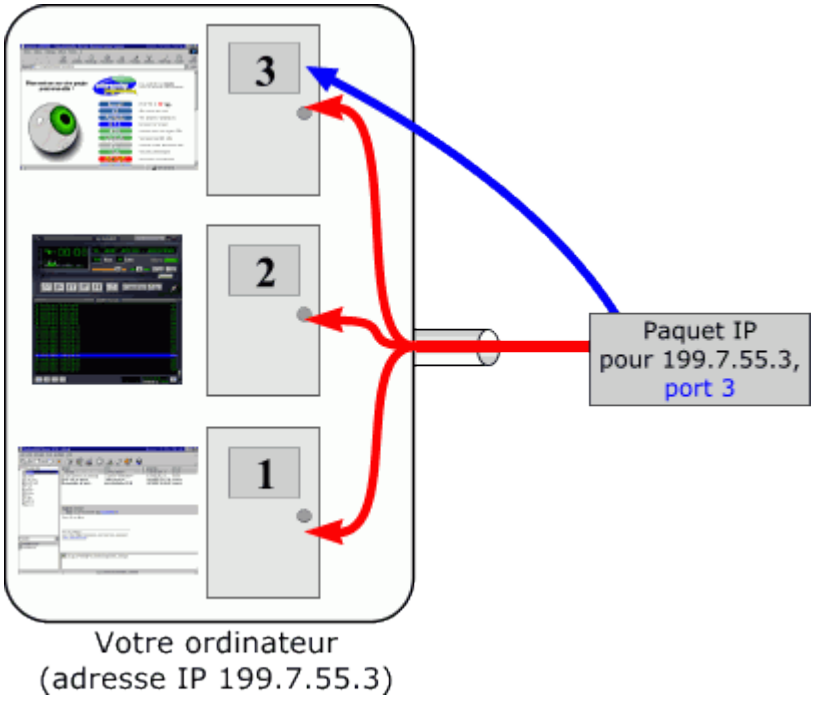


Comment savoir à quel logiciel est destiné ce paquet IP ? Le navigateur, le logiciel de radio ou le logiciel d'email ?

C'est un problème sérieux !

On pourrait attribuer un **numéro unique** à **chaque logiciel** dans l'ordinateur.

Il suffirait alors de mettre ce numéro dans chaque paquet IP pour pouvoir s'adresser à tel ou tel logiciel. On appelle ces numéros des **ports**.



De même, à une même adresse IP, on peut s'adresser à différents logiciels en précisant le numéro de port (ici: 3).

Ainsi, *l'adresse IP* permet de s'adresser à *un ordinateur* donné, et *le numéro de port* permet de s'adresser à *un logiciel* particulier sur cet ordinateur.

Les 3 catégories des numéros de port

Il existe des milliers de ports (ceux-ci sont codés sur 16 bits, il y a donc 65536 possibilités), c'est pourquoi une assignation standard a été mise au point par l'**IANA** (*Internet Assigned Numbers Authority*), afin d'aider à la configuration des réseaux.

- **Les ports 0 à 1023** sont les «**ports reconnus**» ou réservés («**Well Known Ports**»). Ils sont, de manière générale, réservés aux processus système (démons) ou aux programmes exécutés par des utilisateurs privilégiés. Un administrateur réseau peut néanmoins lier des services aux ports de son choix.
- **Les ports 1024 à 49151** sont appelés «**ports enregistrés**» («**Registered Ports**»).
- **Les ports 49152 à 65535** sont les «**ports dynamiques et/ou privés**» («**Dynamic and/or Private Ports**»).

Voici certains des ports reconnus les plus couramment utilisés :

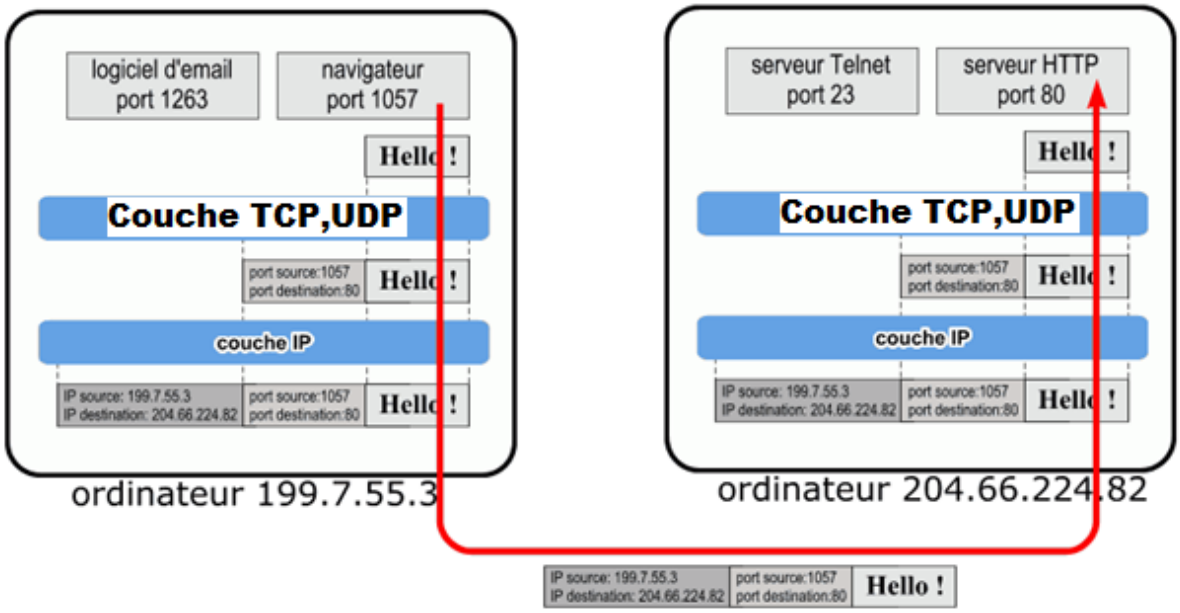
Port	Service ou Application
21	FTP
23	Telnet
25	SMTP
53	Domain Name System
63	Whois
70	Gopher
79	Finger
80	HTTP
110	POP3
119	NNTP

Ainsi, un **serveur** (un ordinateur que l'on contacte et qui propose des services tels que FTP, Telnet, ...) **possède des numéros de port fixes** auxquels l'administrateur réseau a associé des services. Ainsi, les ports d'un serveur sont généralement compris **entre 0 et 1023** (fourchette de valeurs associées à des services connus).

Du côté du **client**, **le port est choisi aléatoirement parmi ceux disponibles par le système d'exploitation**. Ainsi, les ports du client ne seront jamais compris entre 0 et 1023 car cet intervalle de valeurs représente les *ports connus*.

Avec **TCP/IP , UDP/IP** on peut envoyer des données d'une **application x** sur l'**ordinateur A** vers une **application y** sur l'**ordinateur B**.

Par exemple, votre navigateur peut envoyer un message à un serveur HTTP (un serveur Web):



Ce couple (199.7.55.3:1057, 204.66.224.82:80) est appelé un **socket**. **Un socket identifie de façon unique une communication entre deux logiciels.**

Il s'agit d'un modèle permettant la communication inter processus (IPC - *Inter Process Communication*) afin de permettre à divers processus de communiquer aussi bien sur une même machine qu'à travers un réseau TCP/IP.

La communication par socket est souvent comparée aux communications humaines. On distingue ainsi deux modes de communication :

- **Le mode connecté** (comparable à une communication téléphonique), utilisant le protocole TCP. Dans ce mode de communication, une connexion durable est établie entre les deux processus, de telle façon que ***l'adresse de destination n'est pas nécessaire à chaque envoi de données.***
- **Le mode non connecté** (analogue à une communication par courrier), utilisant le protocole UDP. Ce mode **nécessite l'adresse de destination à chaque envoi,** et aucun accusé de réception n'est donné.

TCP

On peut maintenant faire communiquer 2 logiciels situés sur des ordinateurs différents.

Mais il y a encore de petits problèmes:

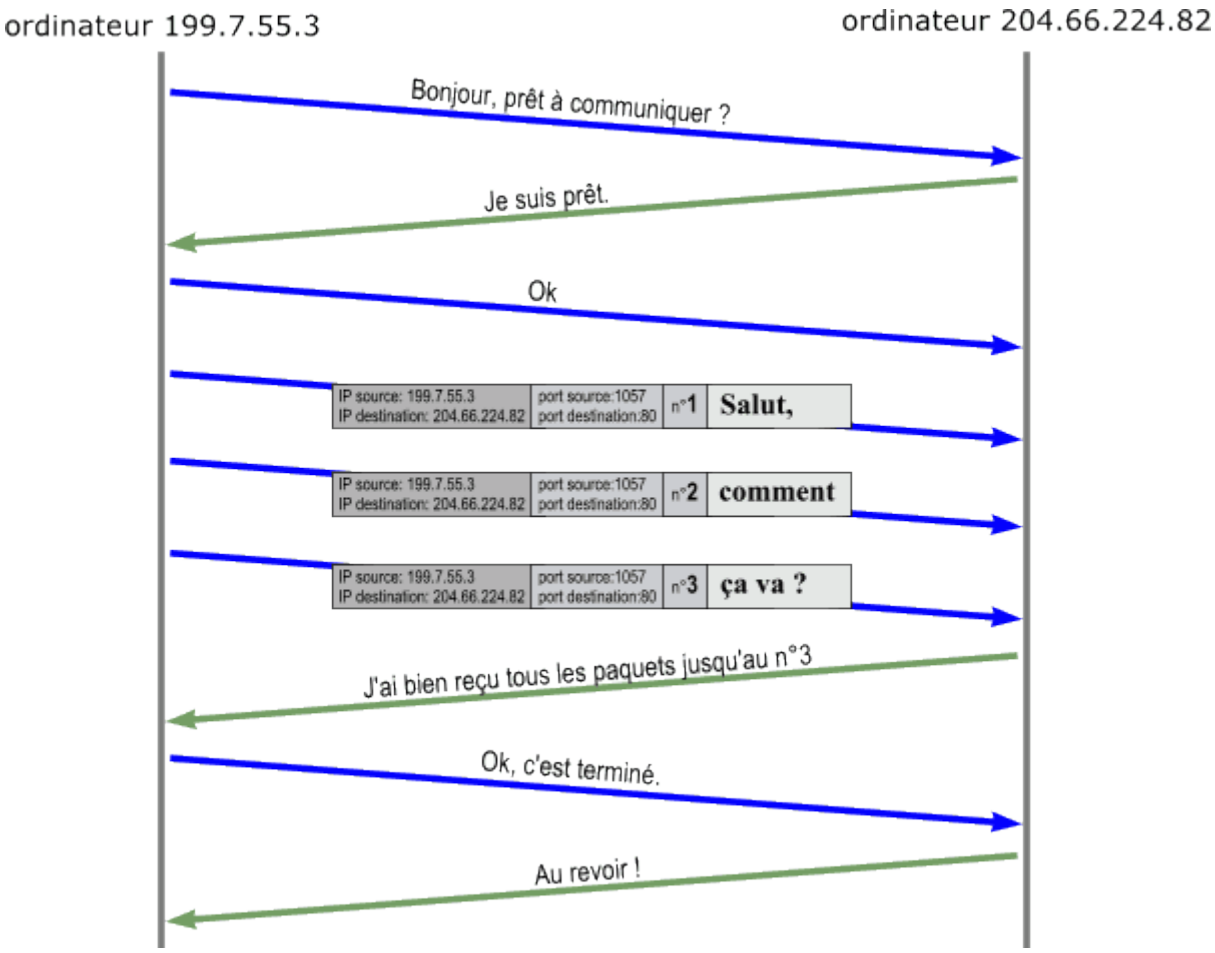
- Quand vous envoyez un paquet IP sur Internet, il passe par des dizaines d'ordinateurs. Et il arrive que des paquets IP se **perdent** ou arrivent en **double exemplaire**. Ça peut être gênant : imaginez un ordre de débit sur votre compte bancaire arrivant deux fois ou un ordre de crédit perdu !
- Même si le paquet arrive à destination, rien ne vous permet de savoir si le paquet est bien arrivé (aucun accusé de réception).
- **La taille des paquets IP est limitée** (environ 1500 octets). Comment faire pour envoyer la photo JPEG du petit dernier qui fait 62000 octets ?

C'est pour cela qu'a été conçu TCP.

Les caractéristiques principales du protocole TCP sont les suivantes :

- de faire tout ce que UDP sait faire (ports).
- de vérifier que le destinataire est prêt à recevoir les données.
- de **découper** les gros paquets de données en paquets plus petits pour que IP les accepte.
- de **numéroter** les paquets, et à la réception de **vérifier** qu'ils sont tous bien arrivés.
- de **redemander** les paquets manquants et de les **réassembler** avant de les donner aux logiciels. Des accusés de réception sont envoyés pour prévenir l'expéditeur que les données sont bien arrivées.
- permet de remettre en ordre les datagrammes en provenance du protocole IP.
- permet de vérifier le flot de données afin d'éviter une saturation du réseau.
- permet de formater les données en segments de longueur variable afin de les "remettre" au protocole IP.
- TCP permet de multiplexer les données, c'est-à-dire de faire circuler simultanément des informations provenant de sources (applications par exemple) distinctes sur une même ligne.
- TCP permet enfin l'initialisation et la fin d'une communication de manière courtoise.

Par exemple, pour envoyer le message "**Salut, comment ça va ?**", voilà ce que fait TCP (Chaque flèche représente 1 paquet IP):



A l'arrivée, sur l'ordinateur 204.66.224.82, la couche TCP reconstitue le message "**Salut, comment ça va ?**" à partir des 3 paquets IP reçus et le donne au logiciel qui est sur le port 80.

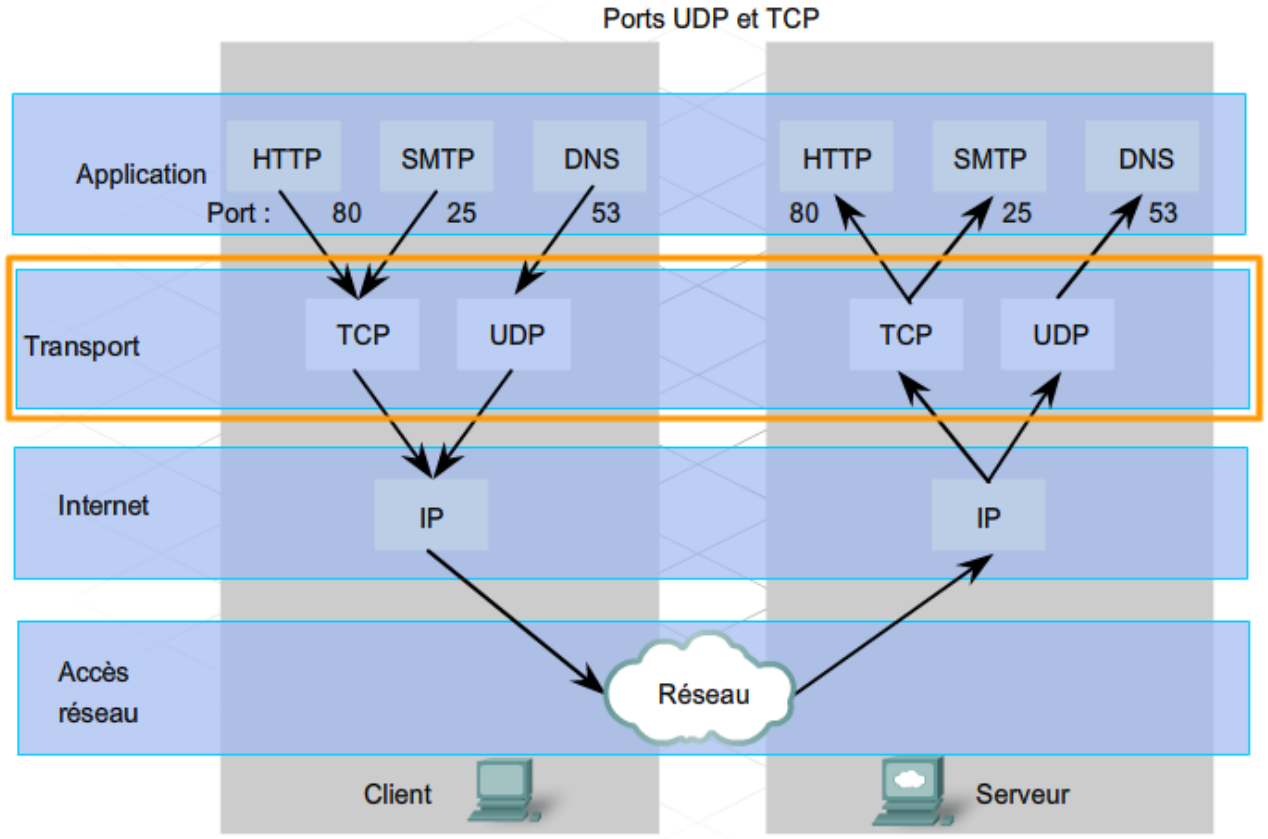
Pour conclure TCP/IP

Avec TCP/IP, on peut maintenant **communiquer de façon fiable** entre logiciels situés sur des ordinateurs différents.

TCP/IP est utilisé pour des tas de choses:

- Dans votre navigateur, le protocole **HTTP** utilise le protocole TCP/IP pour envoyer et recevoir des pages HTML, des images GIF, JPG et toutes sortes d'autres données.
- **FTP** est un protocole qui permet d'envoyer et recevoir des fichiers. Il utilise également TCP/IP.
- Votre logiciel de courrier électronique utilise les protocoles **SMTP** et **POP3** pour envoyer et recevoir des emails. SMTP et POP3 utilisent eux aussi TCP/IP.
- Votre navigateur (et d'autres logiciels) utilisent le protocole **DNS** pour trouver l'adresse IP d'un ordinateur à partir de son nom (par exemple, de trouver 216.32.74.52 à partir de 'www.yahoo.com'). Le protocole DNS utilise UDP/IP et TCP/IP en fonction de ses besoins.

Il existe ainsi des centaines de protocoles différents qui utilisent TCP/IP ou UDP/IP.

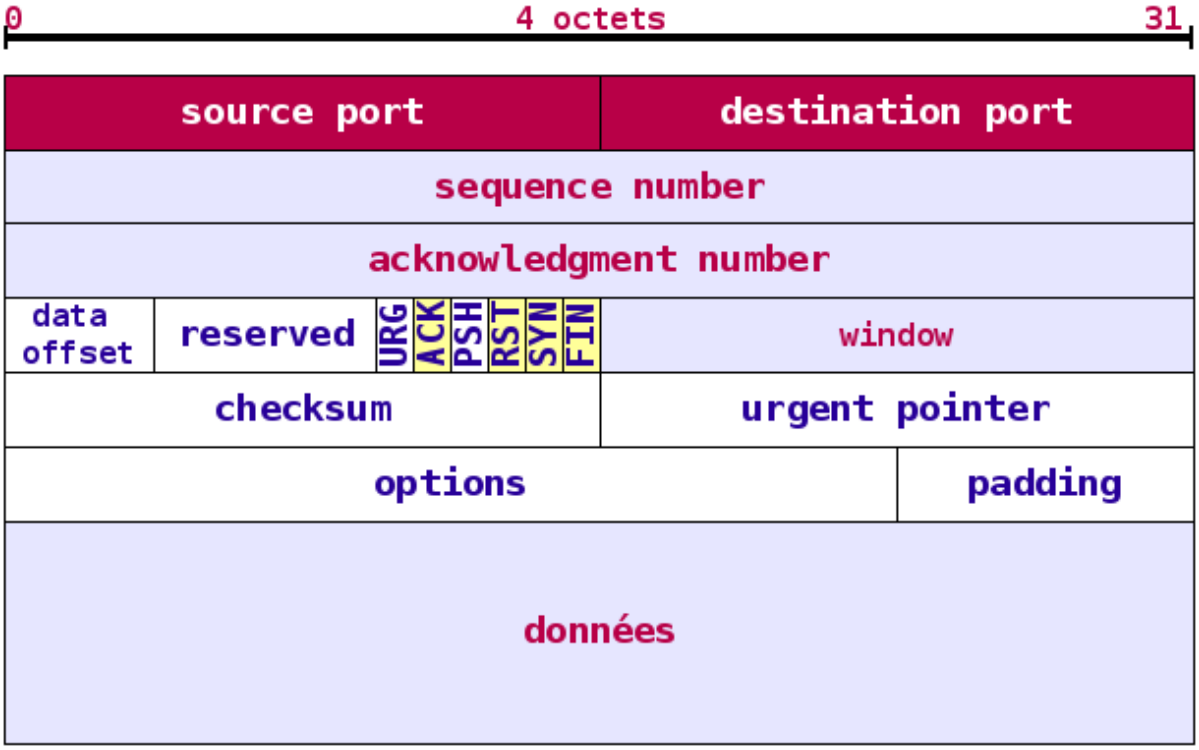


L'avantage de TCP sur UDP est que TCP permet des communications fiables. L'inconvénient est qu'il nécessite une négociation ("*Bonjour, prêt à communiquer ?*" etc.), ce qui prend du temps.

En-tête TCP

L'en tête TCP comprend cinq mots.

- Mot1 port TCP source (16 bits) ; port TCP destination (16 bits)
- Mot2 Numéro de séquence (numéro du premier octet du segment dans le flux d'émission)
- Mot3 Numéro d'accusé de réception (32 bits) (numéro du prochain octet acquitté par le récepteur)
- Mot4 Longueur de l'en-tête en mots de 32 bits (4 bits), Réservé (6 bits), Bits de code (6 bits), Fenêtre (16 bits)
- Mot5 Total de contrôle (16 bits), Pointeur d'urgence (16 bits) : pointeur des données urgentes dans le segment
- Mot6 Options (24 bits) Bourrage (8 bits) *(optionnel)*



Source Port **: 16 bits**

Numéro du port source. Ce numéro correspond au point de communication (*socket inet*) utilisé par le service de la couche application de l'émetteur.

Destination Port **: 16 bits**

Numéro du port destination. Ce numéro correspond au point de communication (*socket inet*) utilisé par le service de la couche application du destinataire.

Sequence Number **: 32 bits**

Le protocole `TCP` a besoin de garder une trace de toutes les données qu'il reçoit de la couche application de façon à être sûr qu'elles ont bien été reçues par le destinataire. De plus, le protocole doit être sûr que ces données ont été reçues dans l'ordre dans lequel elles ont été envoyées. Il doit retransmettre toute donnée perdue.

On affecte un numéro de séquence à chaque octet de donnée pour en garder une trace lors du processus de transmission, **réception et acquittement**. Dans la pratique, ce sont des blocs d'octets qui sont gérés en utilisant les numéros de séquence de début et de fin de bloc.

Les numéros de séquence sont nécessaires à la mise en oeuvre du système de fenêtre glissante du protocole `TCP`. C'est ce système qui garantit fiabilité et contrôle de flots de données.

Acknowledgment Number **: 32 bits**

Le rôle des numéros d'acquittement est le même que celui des numéros de séquence. Simplement, chaque extrémité en communication initie son propre jeu de numéros. Ainsi chaque extrémité assure la fiabilisation et le contrôle de flux de façon autonome.

Data Offset **: 4 bits**

Nombre de mots de 32 bits contenus dans l'en-tête `TCP`. Indication du début des données. Tout en-tête `TCP`, avec ou sans options, est un multiple de mots de 32 bits.

Reserved **: 6 bits**

Champ réservé pour une utilisation ultérieure. Les 6 bits doivent être à 0.

***Control bits* : 6 bits**

Ces bits sont les indicateurs d'état qui servent à l'établissement, au maintien et à la libération des connexions `TCP`. Leur rôle est essentiel dans le fonctionnement du protocole.

- `URG` : indique que le champ *Urgent Pointer* est significatif ; ie. une partie des données du segment sont urgentes.
- `ACK` : indique que le champ *Acknowledgment field* est significatif ; ie. le segment acquitte la transmission d'un bloc de d'octets.
- `PSH` : utilisation de la fonction Push
- `RST` : indique un arrêt ou un refus de connexion.
- `SYN` : indique une demande de synchronisation de numéro de séquence ; ie. demande d'ouverture de connexion `TCP`.
- `FIN` : indique que l'émetteur n'a plus de données à transmettre ; ie. demande de libération de connexion.

***Window* : 16 bits**

- Nombre d'octets de données à partir de celui indiqué par le champ Acknowledgment.

***Checksum* : 16 bits**

- Somme de contrôle sur 16 bits de l'en-tête et des données.

***Urgent Pointer* : 16 bits**

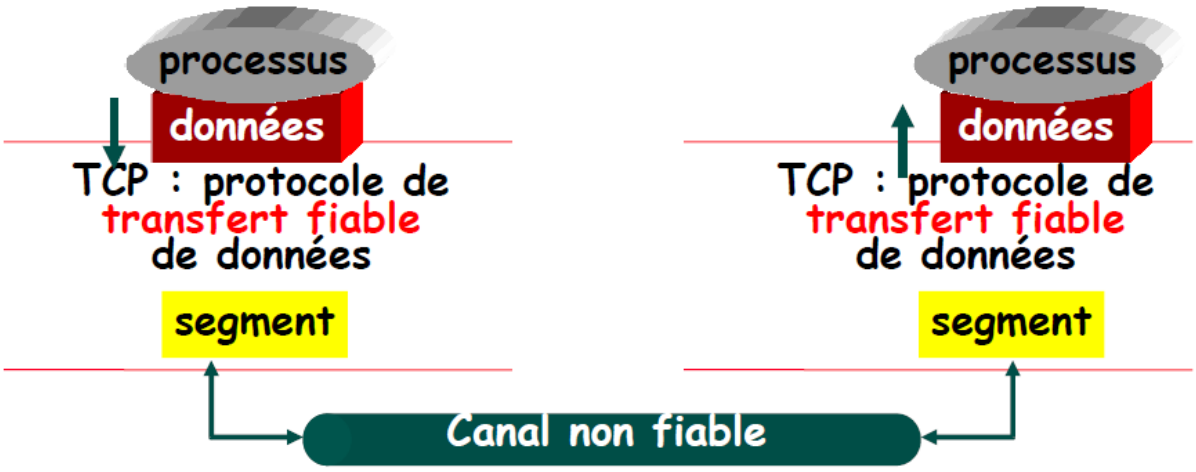
- Ce champ est interprété uniquement si le bit de contrôle `URG` est à 1. Le pointeur donne le numéro de séquence de l'octet qui suit les données urgentes.

***Options* : variable entre 0 et 44 octets**

- Il existe 2 formats d'options : un seul octet de catégorie d'option ou un octet de catégorie d'option suivi d'un octet de longueur d'option et de l'octet des données de l'option.

TCP : fiabilité du transfert de données

- **Fiabiliser le transfert de données sur un canal avec erreur et perte**



TCP : Contrôle d'erreur et Contrôle de flux

Fiabilité Mécanismes

Principe de base:

- Dire quels sont les paquets reçus et quels sont les paquets non reçus.
- On identifie ceux qui ne sont pas reçus et on les réémet.

Contrôle d'erreur

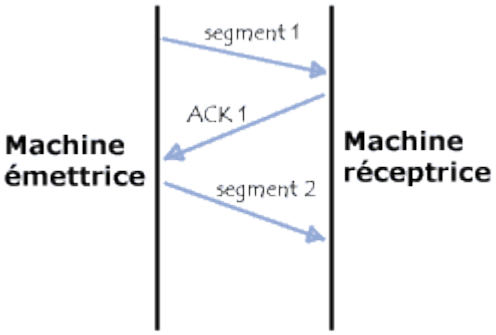
Repose sur

- **Le champ Checksum : idem a UDP**
  Permet de savoir si un bit est erroné
- **Le champ SequenceNumber**
  Permet de détecter la perte de segment (si desequencement)
- **Des acquittements**
  Permet de prévenir l'autre extrémité des pertes (sur desequencement)
- **Des retransmissions**
  Permet de remédier aux pertes de segments et aux réceptions de segments erronés
- **Un temporisateur de retransmission**

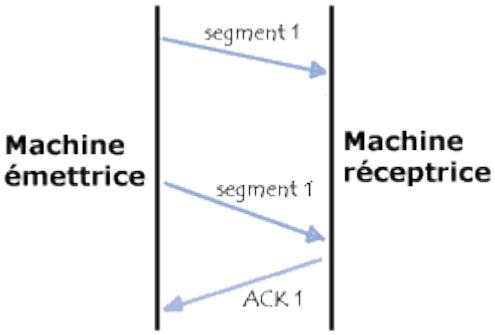
Fiabilité des transferts

Le protocole TCP permet d'assurer le transfert des données de façon fiable, bien qu'il utilise le protocole IP, qui n'intègre aucun contrôle de livraison de datagramme.

En réalité, le protocole TCP possède un système d'accusé de réception permettant au client et au serveur de s'assurer de la bonne réception mutuelle des données. Lors de l'émission d'un segment, un numéro d'ordre (appelé aussi numéro de séquence) est associé. A réception d'un segment de donnée, la machine réceptrice va retourner un segment de donnée dont le drapeau ACK est à 1 (afin de signaler qu'il s'agit d'un accusé de réception) accompagné d'un numéro d'accusé de réception égal au numéro d'ordre précédent.



De plus, grâce à une minuterie déclenchée dès réception d'un segment au niveau de la machine émettrice, le segment est réexpédié dès que le temps imparti est écoulé, car dans ce cas la machine émettrice considère que le segment est perdu...



Toutefois, si le segment n'est pas perdu et qu'il arrive tout de même à destination, la machine réceptrice saura grâce au numéro d'ordre qu'il s'agit d'un doublon et ne conservera que le dernier segment arrivé à destination...

Etablissement d'une connexion

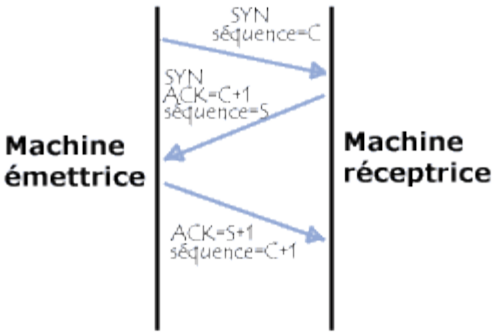
Etant donné que ce processus de communication, qui se fait grâce à une émission de données et d'un accusé de réception, est basé sur un numéro d'ordre (appelé généralement numéro de séquence), il faut que les machines émettrices et réceptrices (client et serveur) connaîssent le numéro d'ordre initial de l'autre machine.
L'établissement de la connexion entre deux applications se fait souvent selon le schéma suivant :

- Les ports TCP doivent être ouverts
- L'application sur le serveur est passive, c'est-à-dire que l'application est à l'écoute, en attente d'une connexion
- L'application sur le client fait une requête de connexion sur le serveur dont l'application est en ouverture passive. L'application du client est dite "en ouverture active"

Les deux machines doivent donc synchroniser leurs séquences grâce à un mécanisme communément appelé three ways handshake (poignée de main en trois temps), que l'on retrouve aussi lors de la clôture de session.

Ce dialogue permet d'initier la communication, il se déroule en trois temps, comme sa dénomination l'indique :

- Dans un premier temps la machine émettrice (le client) transmet un segment dont le drapeau SYN est à 1 (pour signaler qu'il s'agit d'un segment de synchronisation), avec un numéro d'ordre N, que l'on appelle numéro d'ordre initial du client
- Dans un second temps la machine réceptrice (le serveur) reçoit le segment initial provenant du client, puis lui envoie un accusé de réception, c'est-à-dire un segment dont le drapeau ACK est à 1 et le drapeau SYN est à 1 (car il s'agit là encore d'une synchronisation). Ce segment contient le numéro d'ordre de cette machine (du serveur) qui est le numéro d'ordre initial du client. Le champ le plus important de ce segment est le champ accusé de réception qui contient le numéro d'ordre initial du client, incrémenté de 1
- Enfin, le client transmet au serveur un accusé de réception, c'est-à-dire un segment dont le drapeau ACK est à 1, dont le drapeau SYN est à zéro (il ne s'agit plus d'un segment de synchronisation). Son numéro d'ordre est incrémenté et le numéro d'accusé de réception représente le numéro d'ordre initial du serveur incrémenté de 1

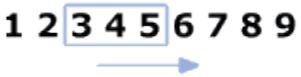


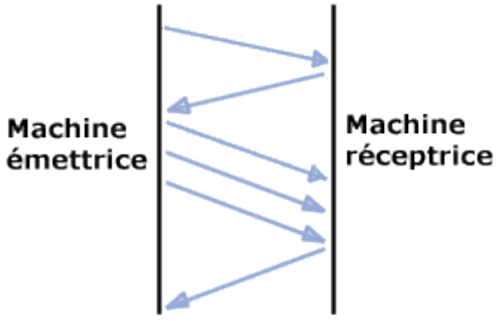
- Suite à cette séquence comportant trois échanges les deux machines sont synchronisées et la communication peut commencer!
- Il existe une technique de piratage, appelée spoofing IP, permettant de corrompre cette relation d'approbation à des fins malicieuses !

Méthode de la fenêtre glissante

Dans de nombreux cas, il est possible de limiter le nombre d'accusés de réception, afin de désengorger le réseau, en fixant un nombre de séquence au bout duquel un accusé de réception est nécessaire. Ce nombre est en fait stocké dans le champ fenêtre de l'en-tête TCP/IP.

On appelle effectivement cette méthode "méthode de la fenêtre glissante" car on définit en quelque sorte une fourchette de séquences n'ayant pas besoin d'accusé de réception, et celle-ci se déplace au fur et à mesure que les accusés de réception sont reçus.





De plus, la taille de cette fenêtre n'est pas fixe. En effet, le serveur peut inclure dans ses accusés de réception en stockant dans le champ fenêtre la taille de la fenêtre qui lui semble la plus adaptée. Ainsi, lorsque l'accusé de réception indique une demande d'augmentation de la fenêtre, le client va déplacer le bord droit de la fenêtre.

Par contre, dans le cas d'une diminution, le client ne va pas déplacer le bord droit de la fenêtre vers la gauche mais attendre que le bord gauche avance (avec l'arrivée des accusés de réception).

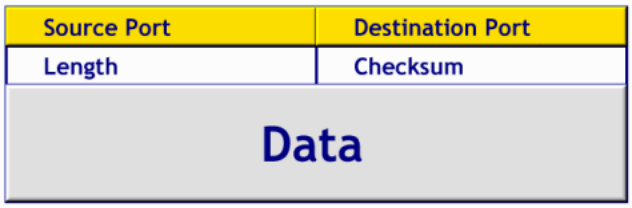
Fin d'une connexion

Le client peut demander à mettre fin à une connexion au même titre que le serveur. La fin de la connexion se fait de la manière suivante :

- Une des machines envoie un segment avec le drapeau FIN à 1, et l'application se met en état d'attente de fin, c'est-à-dire qu'elle finit de recevoir le segment en cours et ignore les suivants
- Après réception de ce segment, l'autre machine envoie un accusé de réception avec le drapeau FIN à 1 et continue d'expédier les segments en cours. Suite à cela la machine informe l'application qu'un segment FIN a été reçu, puis envoie un segment FIN à l'autre machine, ce qui clôture la connexion...

EN-TETE UDP

Le protocole UDP est apparu avec le développement des réseaux locaux dont la fiabilité permet de s'affranchir des fonctions de contrôle. C'est un protocole minimum sans garantie de délivrance des messages et sans séquencement.



***Source Port* : 16 bits**

Numéro du port source. Ce champ est optionnel.

***Destination Port* : 16 bits**

Numéro du port destination.

***Length* : 16 bits**

Longueur en octets du datagramme UDP incluant l'en-tête et les données.

***Checksum* : 16 bits**

Somme de contrôle sur 16 bits de l'en-tête et des données.