Cours de réseaux

M1 Informatique

Faculté Jean Perrin

Plan du cours

Partie 1 : Introduction

Partie 2 : Couche physique

Partie 3 : Couche liaison des données

Partie 4 : Couche Réseau / IPv4

Partie 5 : Couche Réseau / Routage

Partie 6 : Couche Réseau / IPv4, IPv6

Partie 7 : Couche transport : TCP et UDP

Partie 8 : Couche application

Partie 9 : Couche application / Etude de protocoles

Partie 10 : Notions d'attaques et de sécurité

Partie 7
Couche transport

Vous êtes ici

Modèle OSI

7	Application
6	Présentation
5	Session
4	Transport
3	Réseau
2	Liaison
1	Physique

TCP/IP

Applications *Services Internet*
Transport (TCP)
Internet (IP)
Accès au Réseau

Où en sommes-nous ?

Question

Jusqu'à présent que savons-nous faire ?

Réponse

Etablir une **connexion logique** entre deux **machines** distantes et échanger des données entre celles-ci.

Rôles de la couche Transport

La couche transport est **responsable** de la globalité du transfert de bout en bout des données **relatives aux applications**.

Rôles de la couche transport

Permet à de nombreuses applications de communiquer sur le réseau au même moment, sur un même périphérique.

Vérifie que toutes les données ont été reçues dans le bon ordre : fiabilité.

Comment ?

Encapsulation des données d'application de manière à intégrer ses propres données.

Utilisation et mise en œuvre de mécanismes de **gestion des erreurs**.

Fonctionnalités

Pour remplir son rôle, la couche transport doit :

- Effectuer un suivi des communications individuelles entre les applications des hôtes,
- Segmenter les données et gérer chaque bloc,
- Réassembler les segments en flux de données d'application,
- Identifier les différentes applications.

Fonctionnalités

Suivi des communications individuelles

Tout hôte peut héberger plusieurs applications qui communiquent sur le réseau avec une ou plusieurs applications hébergées sur des hôtes distants.

C'est la couche transport qui gère l'ensemble de ces flux de communication entre applications.

Fonctionnalités

Segmentation de données

Préparation des données provenant de la couche application de manière à former des blocs faciles à gérer.

Cette partie inclut l'encapsulation des données de la couche application et l'ajout des en-têtes permettant de savoir à quelle communication le bloc est associé.

Fonctionnalités

Reconstitution des segments

Côté destinataire, il s'agit de réassembler les blocs de données en un flux complet et utilisable. La couche transport redirige également ces données vers l'application appropriée.

Fonctionnalités

Identification des applications

Afin de retrouver les applications cibles, la couche transport doit pouvoir identifier chaque application. Chaque processus logiciel devant accéder au réseau se voit affecter un identifiant unique, inclus dans l'en-tête de la couche transport.

Les protocoles TCP/IP appellent cet identificateur

numéro de port.

Autre fonctionnalité possible

Contrôle du flux

Gestion du flux de données entre les périphériques d'un réseau de manière à empêcher l'entrée excessive de données dans un périphérique.

-> évite les **dépassements de capacité**

Différents besoins des applications

Toutes les applications n'utilisent pas leurs données de la même façon. Elles n'ont donc pas les mêmes besoins.

Exemples :

- « Mes données doivent arriver dans un ordre précis »,
- « L'ensemble de mes données doit être disponible avant de commencer leur traitement »,
- « Perdre une certaine quantité de données n'est pas un problème pour le traitement général. »

Pour satisfaire ces besoins différents, la couche transport propose **plusieurs protocoles différents** reposant sur des règles variées.

PDU de la couche transport

L'unité traitée et considérée dans la couche transport s'appelle le **segment** (ou **message**).

En fonction des protocoles ce PDU peut prendre d'autres noms comme **datagramme** (UDP).

Segmentation des données

La couche transport reçoit directement de la donnée applicative (fonctionnelle). Cette donnée est éventuellement **découpée** puis **encapsulée** sous la forme de segment.

On appelle cette étape la **segmentation**.

A quoi sert la segmentation ?

Imaginons l'envoi sur le réseau d'une vidéo sous la forme d'un flux de communication **complet**.

Quels sont les inconvénients d'une telle solution ?

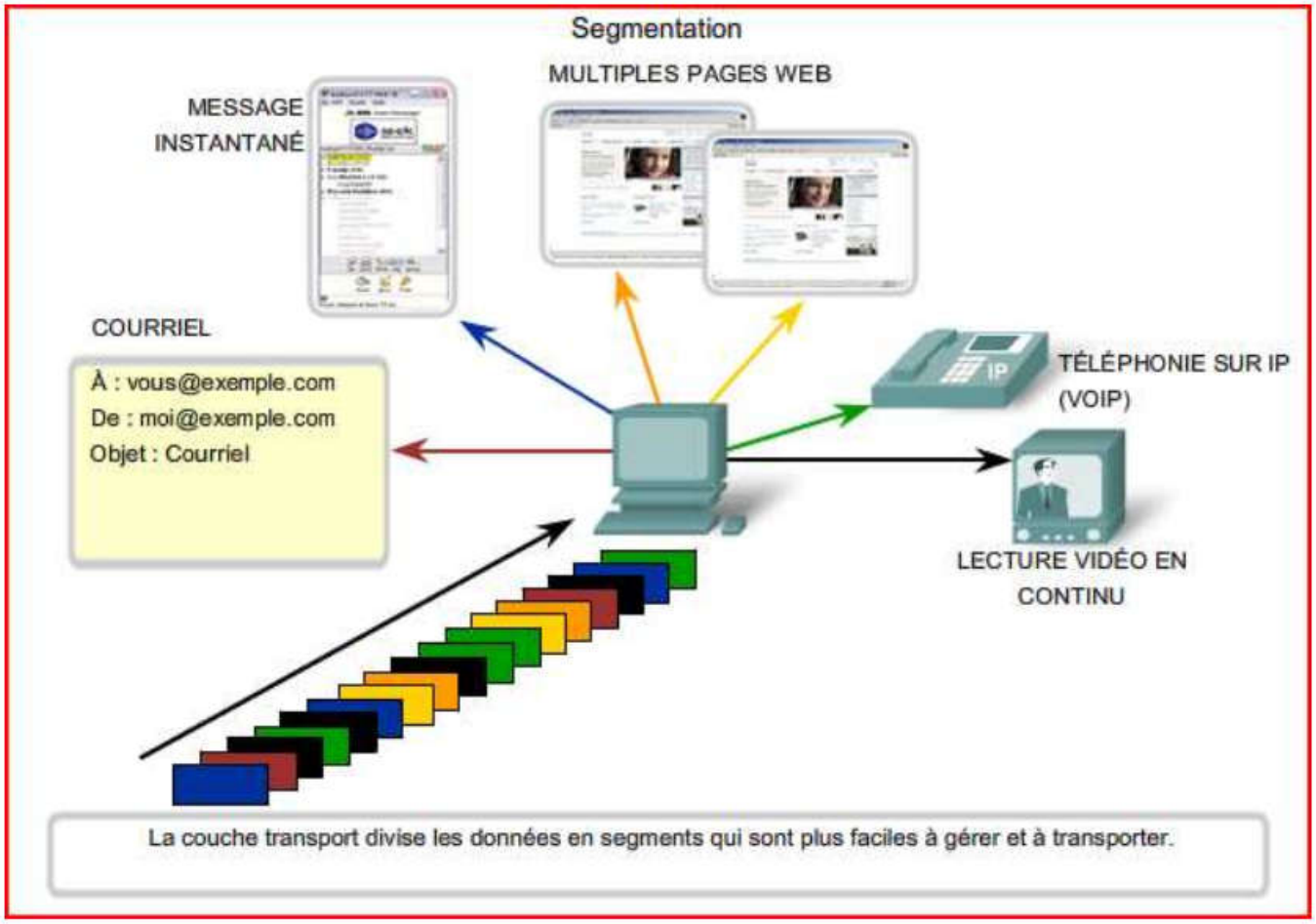
a) Empêche d'autres communications simultanées.
b) Reprise sur erreur difficile

A quoi sert la segmentation ?

Fractionner les données en blocs plus petits permet d'entrelacer plusieurs communications différentes sur le même réseau -> **multiplexage**.

Au niveau de la couche transport, un ensemble de blocs transitant entre deux applications (source et destination) est appelé **conversation**.

A quoi sert la segmentation ?



Comment assurer la fiabilité ?

La **fiabilité** consiste à garantir que chaque bloc de données envoyé par la source arrive à destination.

3 opérations de base :

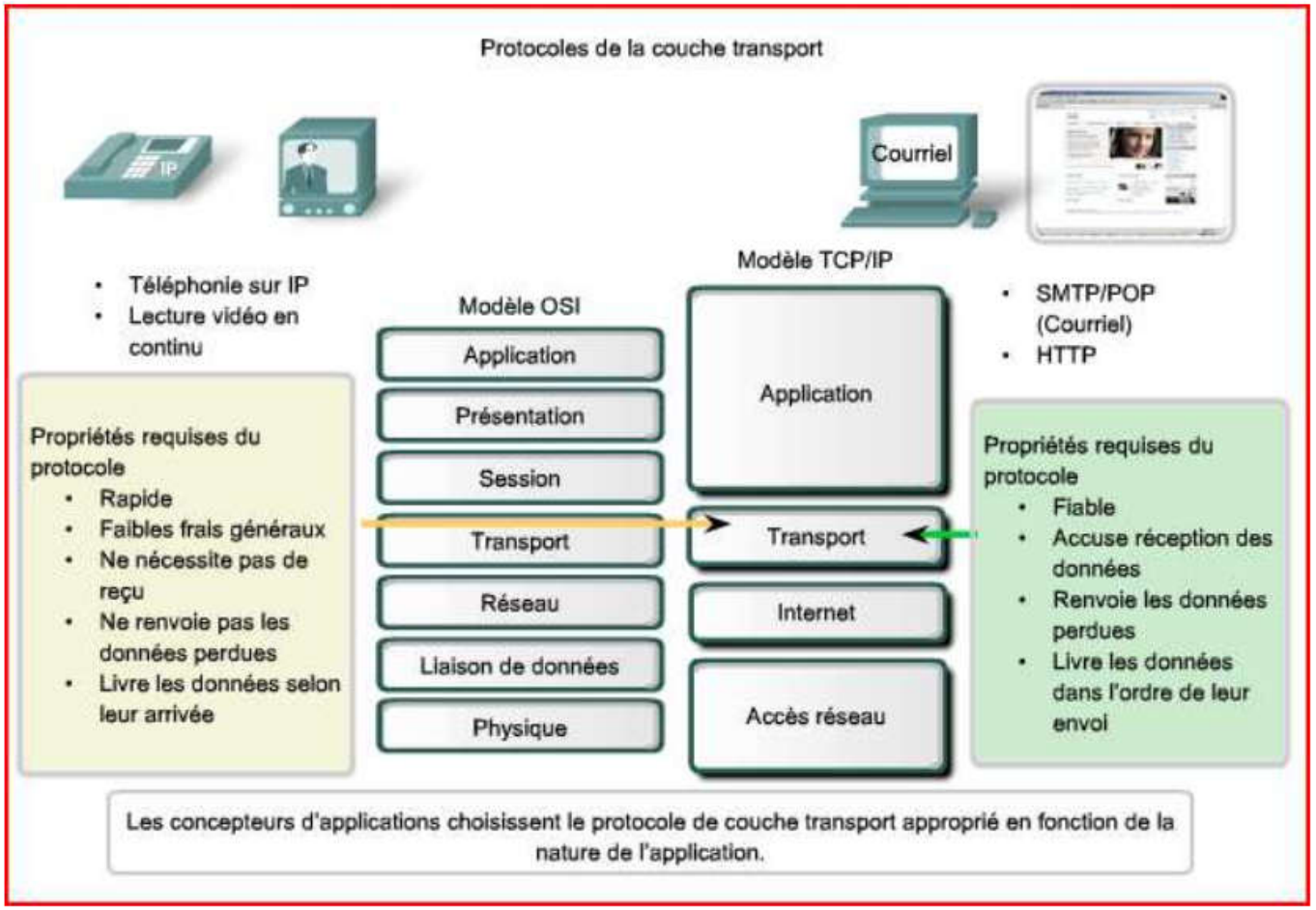
- Suivi des données transmises
- Gestion d'un accusé de réception
- Retransmission des données non acquittées

Fiabilité vs performance

Assurer la fiabilité a un coût (gestion des ACK, retransmission éventuelle…).

Certaines applications n'ont pas besoin de cette sécurité (perte de blocs possibles) mais nécessitent d'être performantes.

Fiabilité vs performance



Les protocoles de TCP/IP

TCP/IP intègre deux protocoles principaux qui gèrent les communications de la plupart des applications.

Protocole **UDP** : User Datagram Protocol

Protocole **TCP** : Transmission Control Protocol

Protocole UDP : présentation rapide

Protocole simple, sans connexion (RFC 768)

Peu de surcharge

Les blocs appelés **datagrammes** sont envoyés « au mieux ».

Exemples d'utilisation :

- Système de nom de domaine (DNS)
- Lecture en streaming
- Voix sur IP

Protocole TCP : présentation rapide

Protocole avec connexion (RFC 793)

Surcharge pour accroître les fonctionnalités :

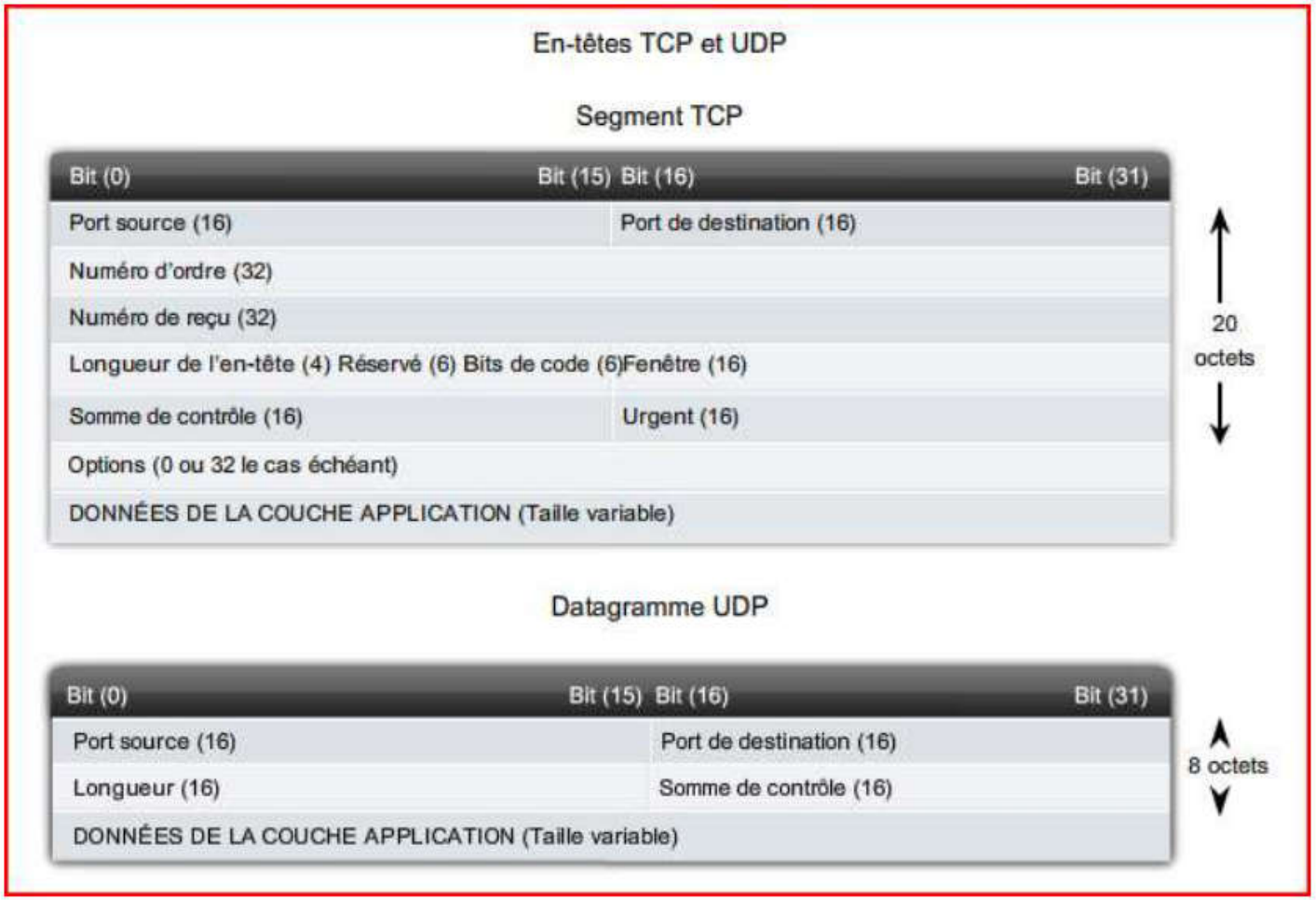
- Livraison dans l'ordre
- Acheminement fiable
- Contrôle de flux

Les blocs sont appelés **segments**.

Exemples d'utilisation :

- Transfert de fichiers
- Mails
- Navigation Web

En-têtes TCP vs UDP :



Adressage de ports

Identificateurs uniques d'**applications**.

Les en-têtes UDP et TCP intègrent deux champs :

- Port source
- Port destination

Processus serveur : numéro de port statique

Processus Client : numéro de port choisi dynamiquement pour chaque conversation

Adressage de ports

Application cliente vers Serveur

Port destination = port affecté au démon du service

Port source = généré de façon aléatoire (parmi les numéros de ports non utilisés ou réservés).

Le port source deviendra destination dans le bloc réponse.

Interface de connexion

L'ensemble *adresse IP* (couche réseau) + *numéro port* (couche transport) identifie de façon **unique** un processus précis sur une machine spécifique.

On l'appelle **interface de connexion**.

Ports spéciaux

L'IANA (Internet Assigned Numbers Authority) attribue les numéros de ports pour certaines applications.

On distingue trois « types » de ports en fonction de la plage de numéros qu'ils occupent :

- Ports réservés : 0 à 1023
- Ports inscrits : 1024 à 49151
- Ports privés ou dynamiques : 49152 à 65535

Ports spéciaux

Ports réservés

Numéros de 0 à 1023

Réservés à des services et applications « universelles » ou courantes

Exemples :

TCP	UDP	Commun
21 : FTP	69 : TFTP	53 : DNS
23 : Telnet	520 : RIP	161 : SNMP
25 : SMTP		531 : IM d'AOL
80 : HTTP		
110 : POP3		
194 : IRC		

Ports spéciaux

Ports inscrits

Numéros de 1024 à 49151

Affectés à des processus ou applications d'utilisateurs.

Applications particulières qu'un utilisateur a choisi d'installer plutôt que des applications courantes qui recevraient un port réservé

Exemples :

TCP	UDP	Commun
1863 : MSN Messenger	1812 : Authent. Radius	1433 : MS SQL
8006 : Autre HTTP	2000 : Cisco SCCP (VoIP)	2948 : WAP
8080 : Autre HTTP	5004 : RTP	

Ports spéciaux

Ports privés ou dynamiques

Numéros de 49152 à 65535

Affectés de façon dynamiques à des applications clientes lors de l'initialisation d'une connexion.

Egalement appelés **ports éphémères**

TCP en détail

Fiabilisation des conversations

A l'initialisation de la conversation, TCP établit une « **connexion** » avec la destination.

Cette connexion permet le suivi d'une session (flux de communication) entre les hôtes dans les deux directions.

Quand la destination reçoit un segment, elle envoie un reçu vers la source. La source cesse alors le suivi de ce segment.

Si la source ne reçoit pas de reçu dans un délai prédéfini, elle retransmet la donnée.

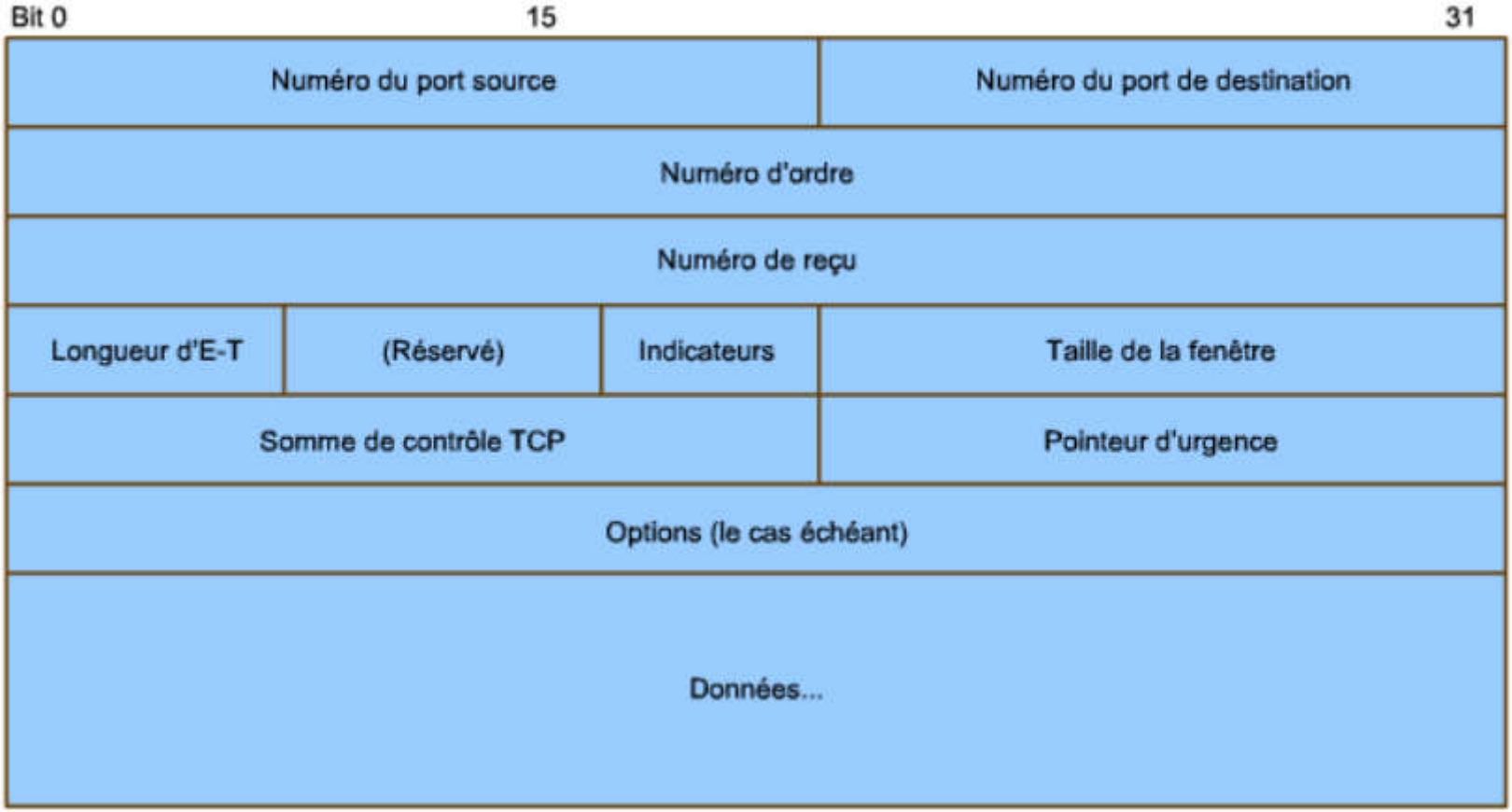
Surcharge due à la fiabilisation

Trafic réseau généré par les reçus et les retransmissions.

Etablissement des sessions : échange supplémentaire de segments spécifiques.

Surcharge au niveau des hôtes pour assurer le suivi.

En-tête du segment TCP



En-tête du segment TCP

Numéro port source : session TCP sur le périphérique ayant ouvert une connexion (souvent une valeur aléatoire au moins supérieure à 1023).

Numéro port destination : identification de l'application destinataire sur le site récepteur.

En-tête du segment TCP

Numéro d'ordre : indique la position du segment dans le flux global de l'émetteur. Valeur dépendante du flag SYN. En général, il s'agit du numéro du premier octet de la partie donnée relativement au début de transmission.

Numéro de reçu : indique le prochain numéro d'ordre attendu par le destinataire.

En-tête du segment TCP

Longueur E-T (offset) : longueur de l'en-tête du segment en multiple de 32 bits. Permet d'indiquer où commence la partie data (le champ option a une longueur variable).

Indicateurs : utilisés pour la gestion de la session et le traitement des segments (6 bits)

- URG : segment à traiter de façon urgente
- ACK : segment de type accusé de réception
- PSH : forcer l'émission des données jusqu'à l'application destinataire (PUSH)
- RST : il faut réinitialiser la connexion
- SYN : demande d'établissement d'une connexion (Synchronize Sequence Number)
- FIN : mettre fin à la connexion

En-tête du segment TCP

Somme de contrôle TCP : checksum permettant de contrôler les erreurs sur l'en-tête et la donnée.

Pointeur d'urgence : utilisé uniquement avec l'indicateur URG.

Options : informations facultatives

Processus serveur TCP

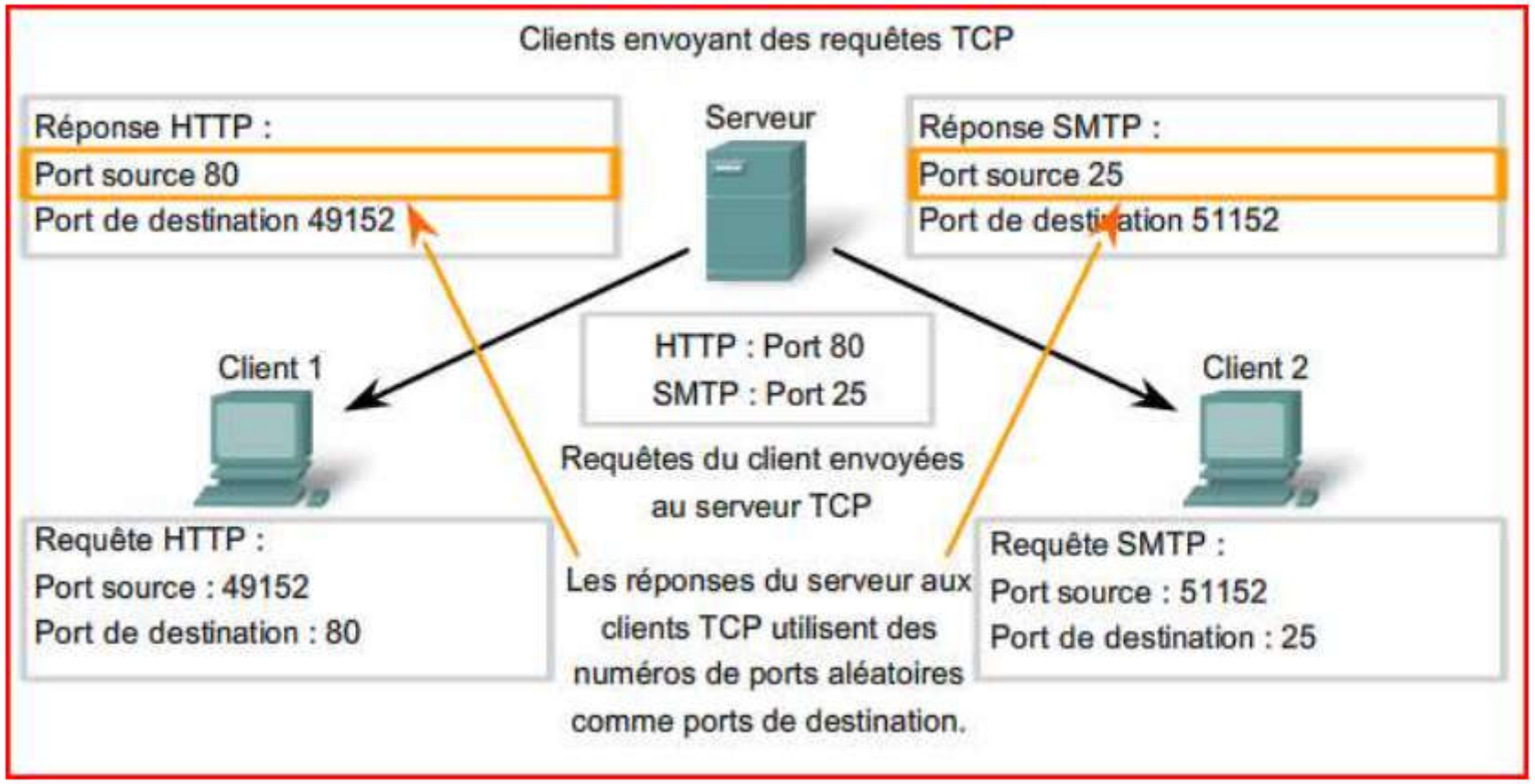
Les processus applicatifs (services) s'exécutent sur des serveurs.

Chacun est configuré avec un numéro de port (par défaut ou attribué manuellement) unique sur la machine hôte.

Lorsqu'une application active est affectée à un port, on dit que **ce port est ouvert**.

Un serveur peut fournir plusieurs services simultanément.

Processus serveur TCP



Etablissement d'une connexion TCP

C'est l'hôte jouant le rôle de client qui débute la connexion.

La connexion s'établit en **trois** étapes et permet d'établir **deux sessions unidirectionnelles** :

1) Le client envoie un segment d'initialisation.
2) Le serveur répond par un segment contenant ACK et sa propre demande d'initialisation (1ere session établie).
3) Le client répond par un l'ACK à la demande d'initialisation (2ème session établie).

La connexion est établie.

Etablissement d'une connexion TCP

Etape 1

Le client envoie un segment (SYN) contenant un numéro d'ordre initial.

Ce numéro d'ordre est appelé **ISN** : Initial Sequence Number. Il est choisi de manière aléatoire.

Par la suite, le numéro d'ordre sera incrémenté de 1 pour chaque octet de données envoyé par le client vers le serveur pour cette conversation.

Etablissement d'une connexion TCP

Etape 2

Le serveur accuse réception du segment (SYN). Il envoie un segment (ACK) dont le numéro de reçu est égal à l'ISN du client +1. **Session du client vers le serveur établie**.

Le serveur doit initier sa propre session. Le segment envoyé est donc également de type SYN et contient son propre ISN.

Etablissement d'une connexion TCP

Etape 3

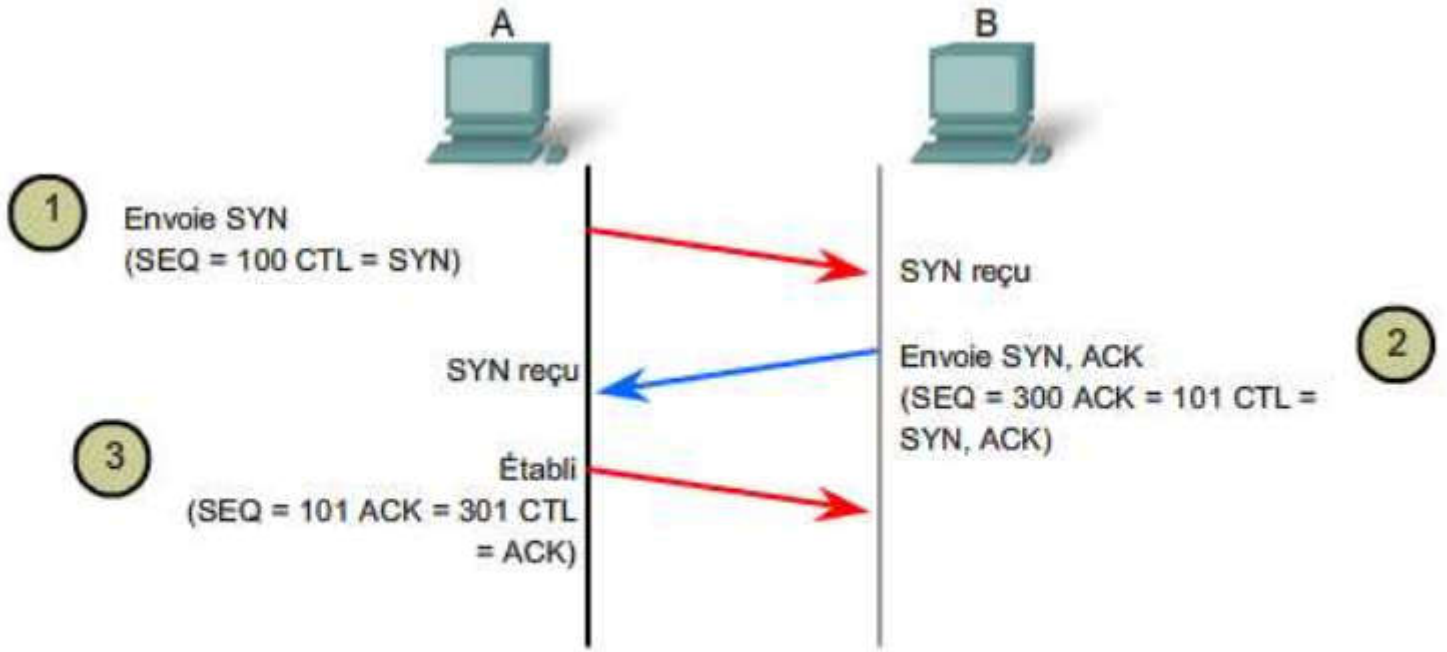
Le client envoie un reçu au serveur en réponse à la partie SYN.

Ce segment est du type ACK avec numéro de reçu égal à ISN du serveur +1. Le numéro de séquence de ce segment est égal au dernier numéro de séquence + n.

La session du serveur vers le client est établie.

La connexion complète est établie.

Etablissement d'une connexion TCP



CTL = Indique quels bits de contrôle de l'en-tête TCP sont configurés sur 1.

A envoie une réponse ACK à B.

Connexion TCP

A votre avis, pourquoi ne commence-t-on pas systématiquement la numérotation à 0 ?

Pourquoi démarre-t-on à partir d'un ISN « aléatoire » ?

Fermeture d'une connexion TCP

Utilisation de l'indicateur FIN.

Chaque session TCP unidirectionnelle est fermée en deux étapes. La connexion nécessite donc 4 étapes pour être fermée.

Le processus de fermeture peut être initié par n'importe lequel des hôtes.

Fermeture d'une connexion TCP

Etape 1

Un des hôtes (ex. A) envoie un segment (FIN).

Etape 2

Le second hôte (B) envoie un segment ACK comme accusé de réception et éventuellement ses dernières données.

Etape 3

L'hôte (B) envoie un segment (FIN).

Etape 4

L'hôte (A) envoie un segment ACK comme accusé de réception.

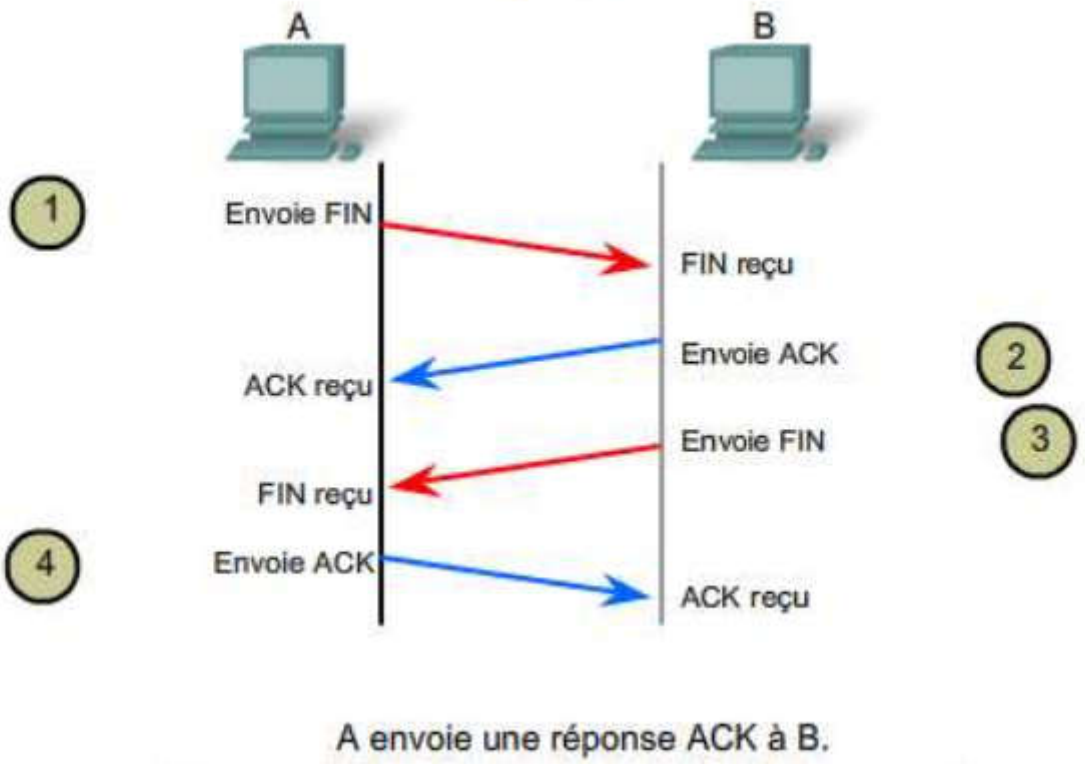
Fermeture d'une connexion TCP

Remarque :

Il est possible de fusionner les étapes 2 et 3.

Lors de la réception de FIN sur l'hôte B, si celui-ci n'a plus de données à envoyer, il peut envoyer simultanément ACK et FIN.

Fermeture d'une connexion TCP



Réassemblage des segments TCP

Lors de la transmission de données en TCP, les segments peuvent arriver à destination dans le désordre.

TCP numérote chaque segment de manière unique.

- Premier segment : ISN numéro d'ordre initial
- Segment suivant (segment N) : numéro d'ordre du segment N-1 (segment transmis juste avant) incrémenté du nombre d'octets transmis.

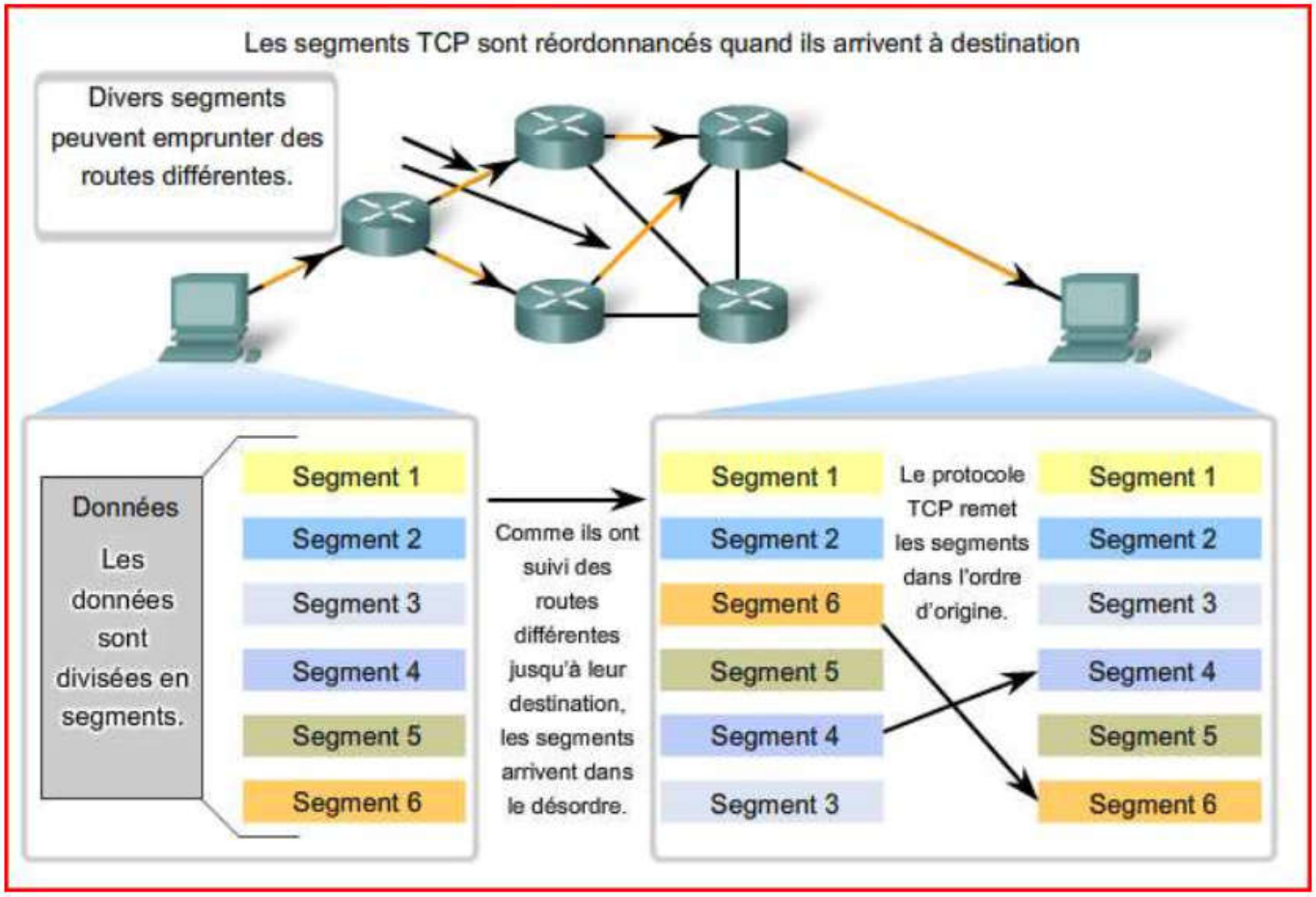
Réassemblage des segments TCP

Coté récepteur, TCP utilise ce numéro d'ordre pour réordonner l'ensemble des données d'une conversation.

Chaque segment est mis dans une mémoire tampon dans l'ordre qui convient (numéros d'ordre contigus)

Remarque : Cela permet également d'identifier les segments manquants.

Réassemblage des segments TCP



Accusé de réception

L'envoi d'accusé de réception permet d'assurer l'émetteur que chaque segment a été reçu.

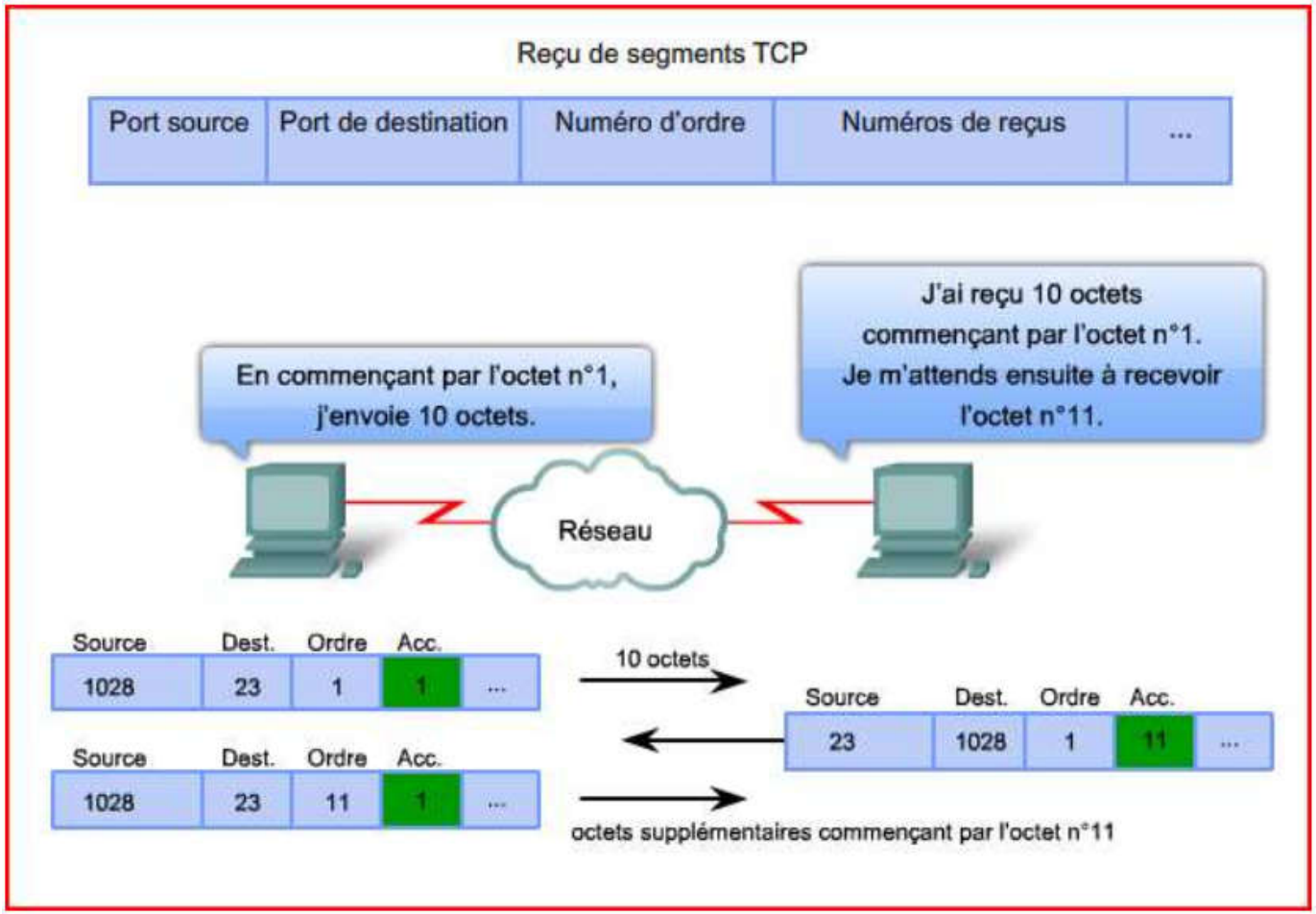
Le numéro de reçu du segment accusé de réception indique le numéro de l'octet suivant que l'on s'attend à recevoir : **reçu prévisionnel**.

Accusé de réception

Coté émetteur, l'hôte est censé envoyer un segment dont le numéro d'ordre est égal au numéro de reçu.

Ce principe suppose que l'émission N+1 ne peut être faite qu'une fois l'ACK correspondant à N reçu.

Accusé de réception



Accusé de réception avec fenêtrage

Cette attente systématique d'ack pour chaque petit segment entraine une forte surcharge.

Pour réduire le nombre d'ack, plusieurs segments peuvent être envoyés à l'avance et faire l'objet d'un reçu unique (définissant le nombre total d'octets reçus).

La quantité de données pouvant être transmise avant la réception de l'accusé est appelé **taille de fenêtre**.

Gestion des pertes de segments

TCP fournit plusieurs méthodes de traitement de perte de segments.

L'une d'elle consiste en un mécanisme de retransmission des segments (contenant des données) sans reçu.

Si l'hôte source n'a pas reçu l'accusé après un délai prédéfini, il retransmet le données depuis le dernier numéro de reçu connu.

Il s'agit d'un processus non spécifié par RFC.

Contrôle de l'encombrement

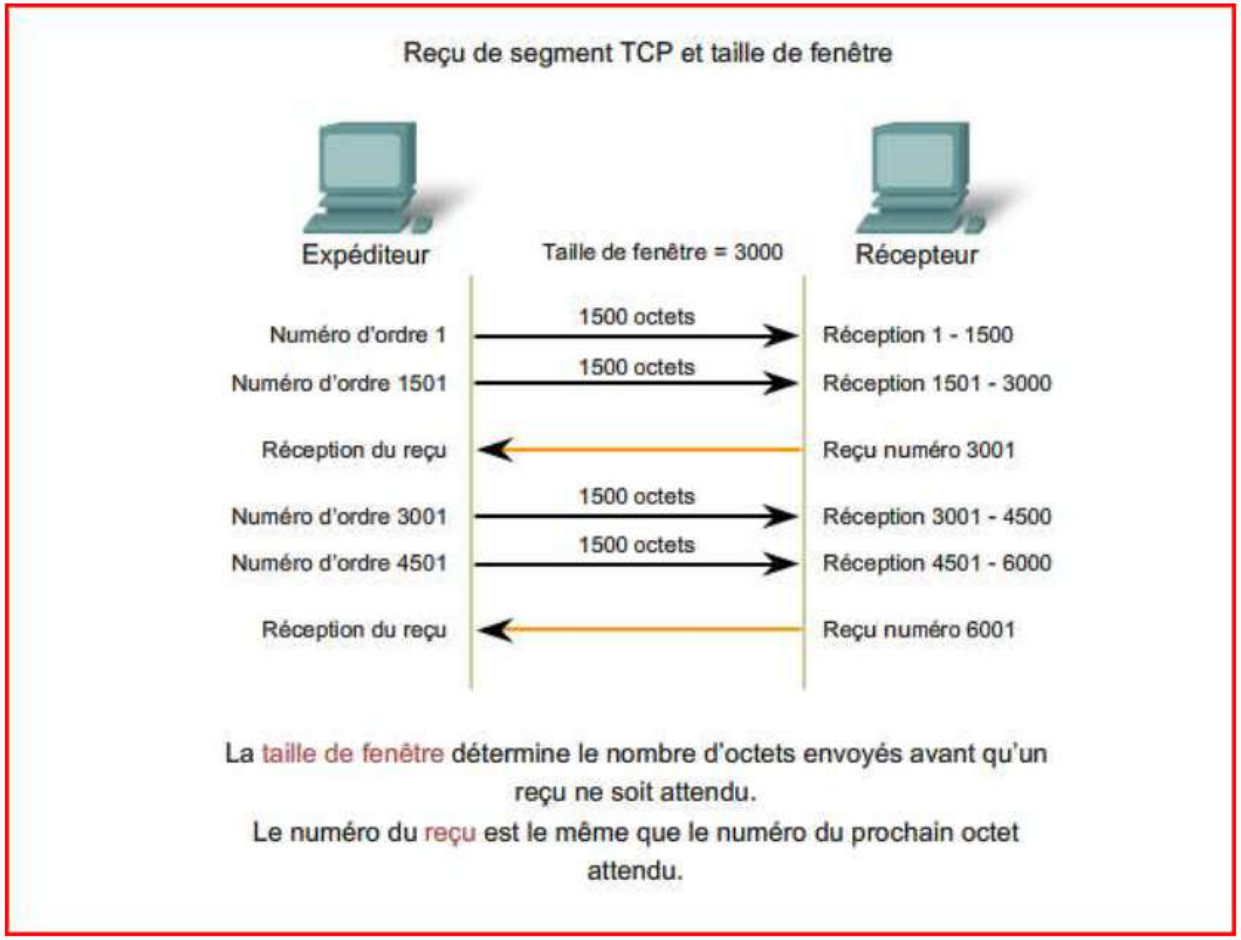
TCP inclut des mécanismes de contrôle de flux : cela contribue à la fiabilité des transmissions.

Réglage du taux effectif de flux de données.

La taille de fenêtre permet de préciser la quantité de données pouvant être transmise avant de recevoir un ack.

La taille initiale est réglée sur la valeur du flux maximum que le réseau et l'hôte de destination peuvent prendre en charge sans perte.

Contrôle de l'encombrement



Contrôle de l'encombrement

Taille de fenêtre dynamique

En cas d'encombrement, TCP peut réduire la taille des fenêtres de manière à imposer l'envoi plus fréquent de reçus.

Ceci a pour effet ralentir le taux de transmission car la source attend des reçus plus fréquent avant de réémettre.

La destination peut être à l'origine d'envoi de l'information sur le nombre d'octets qu'elle est prête à recevoir.

Au fur et à mesure de la conversation, l'émetteur augmente petit à petit la taille de la fenêtre, jusqu'à ce qu'il y ait à nouveau perte de segment.

Recherche de la taille optimale

Contrôle de l'encombrement

Mécanisme slow start

Mécanisme de base de TCP pour contrôler le flux

La taille initiale de la fenêtre est petite : débit faible

La taille de la fenêtre est doublée tant que tout va bien.

En cas de problème (perte) elle est réinitialisée.

UDP en détail

UDP : faible surcharge vs fiabilité

UDP : User Datagram Protocol

Protocole simple offrant les fonctions de base de la couche transport.

- Pas de connexion
- Pas de mécanisme de retransmission
- Pas de séquençage
- Pas de contrôle de flux

→ Surcharge réduite

Exemples d'utilisation

Il existe des protocoles importants de la couche application utilisant UDP :

- DNS : Domain Name System
- SNMP : Simple Network Management Protocol
- DHCP : Dynamic Host Configuration Protocol
- RIP : Routing Information Protocol
- TFTP : Trivial File Transfert Protocol
- Jeux en ligne

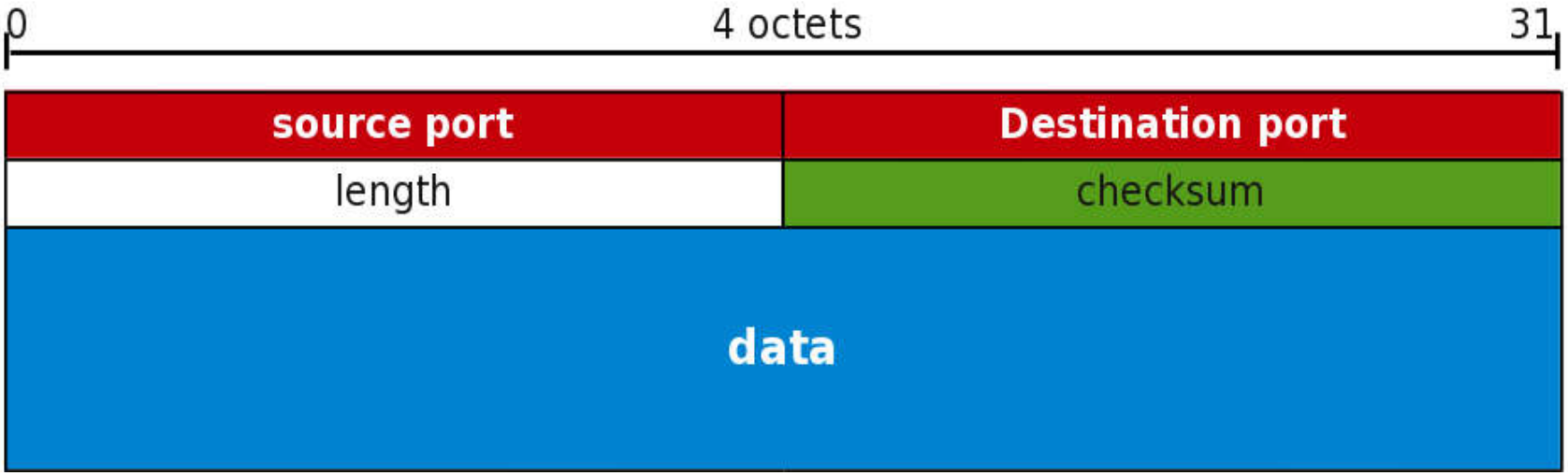
Pourquoi UDP

Certaines applications comme les jeux en ligne ou la voix sur IP peuvent tolérer la perte d'une certaine quantité de données.

L'utilisation de TCP entrainerait des ralentissements (délais de détection et gestion de perte de données) préjudiciables à l'application.

Certaines applications (ex. DNS) se contentent de répéter la requête en cas de non réponse : système adapté au besoin et plus léger.

En-tête UDP


0
4 octets
31
source port
Destination port
length
checksum
data

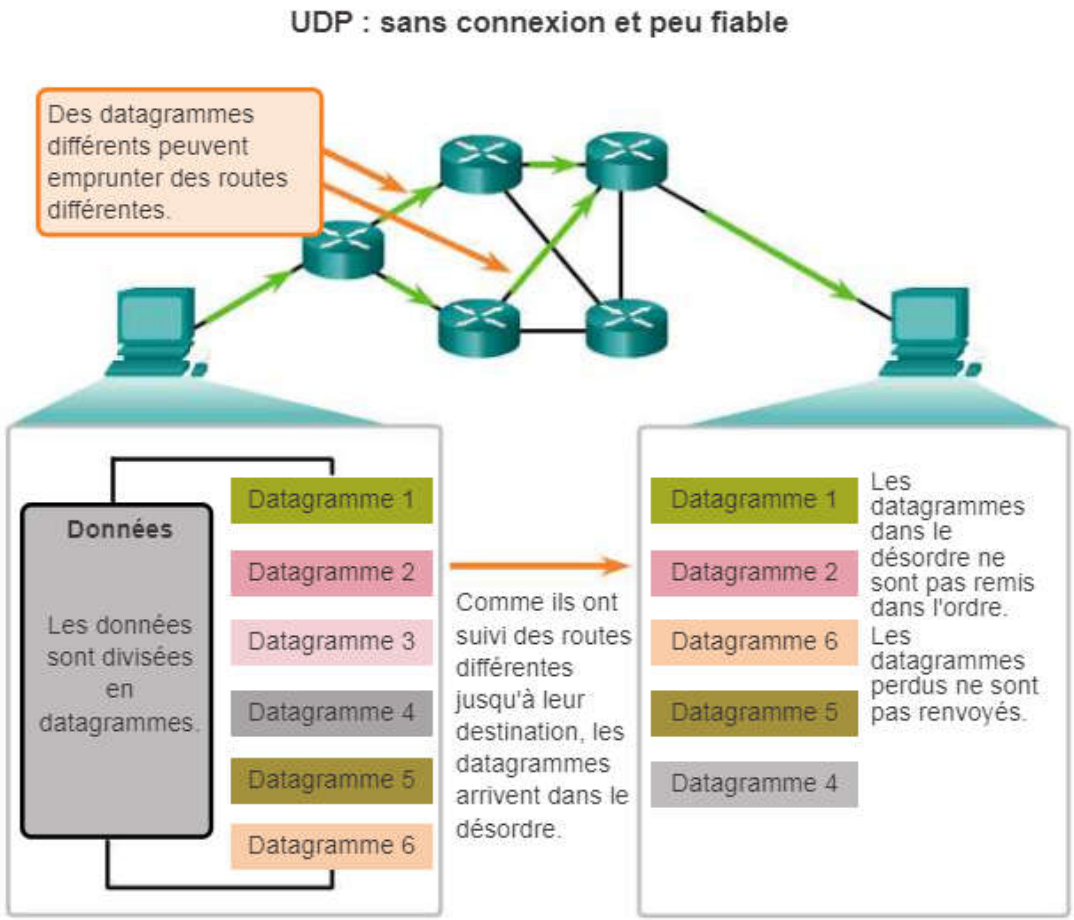
Réassemblage des datagrammes UDP

De nombreuses applications utilisant UDP envoient des petites quantités de données tenant sur un seul segment.

Si la donnée doit être découpée en plusieurs segments ou datagrammes, UDP n'a aucun moyen pour les réordonner

Ils sont réassemblés dans l'ordre d'arrivée.

Réassemblage des datagrammes UDP



Processus des clients UDP

La communication entre le client et le serveur est démarrée par une application côté client.

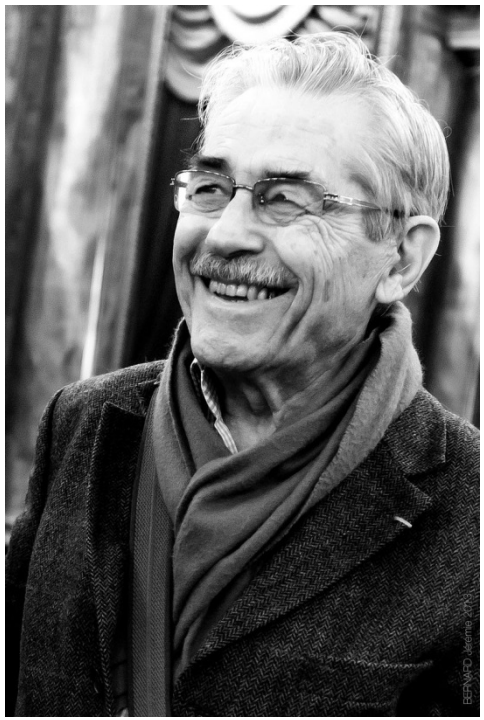
Le processus client UDP sélectionne un numéro de port aléatoire dans la plage dynamique et l'utilise comme port source.

Il identifie le port destination.

Il peut alors envoyer son datagramme : pas de session à ouvrir.

L'homme du jour

Louis Pouzin, né le 20 avril 1931 dans la Nièvre.



Ingénieur en informatique (X promotion 1950). Il travaille d'abord sur la mise au point d'un réseau téléphonique chez Bull. En 1963, il part au MIT et travaille sur le premier projet de *temps partagé*. Puis, il rentre à la Société d'Economie et de Mathématique Appliquée (SEMA -> future ATOS).

En 1970, il invente le datagramme et conçoit le premier réseau à commutation de paquets. Ses travaux ont été largement utilisés par Vint Cerf pour l'élaboration de TCP/IP.

Prix IEEE Internet (2003)

Chevalier de la légion d'honneur (2003)

En 2012 il rentre au *Temple de la renommée d'Internet* (*Internet Hall of Fame*) en tant que pionnier.

Queen Elizabeth Price for Engineering (2013) conjointement à Bob Kahn, Vint Cerf.

Conclusion

La couche transport fournit les données dont le réseau a besoin.

Les numéros de port servent à identifier les applications en communication.

Les deux protocoles couramment utilisés sont UDP et TCP : l'un apportant fiabilité et l'autre simplicité.

TCP apporte un certain niveau de fiabilité qui n'apparaissait pas dans IP.

Les développeurs d'applications choisissent le protocole de la couche transport répondant le mieux aux exigences de leur appli.

UNIVERSITÉ D'ARTOIS

L'homme de l'autre jour

Donald (Watts) Davies, né le 7 juin 1924 au Pays de Galles et mort le 28 mai 2000.

Informaticien, physicien et mathématicien, il a co-inventé avec Paul Baran la commutation de paquets. Précédemment, il a notamment travaillé à partir de 1947 au National Physical Laboratory (NPL) où Alan Turing concevait le Automatic Computing Engine (ACE).

Il a été nommé Distinguished Fellow du British Computer Society en 1975, commandeur de l'ordre de l'Empire Britannique en 1983 et fellow de la Royal Society depuis 1987.