TCP et UDP
La couche transport

Introduction

Généralités

Services offerts par IP/NWK

- Adressage universel
- Routage et interconnexion de réseaux

} Remise des paquets au nœud de destination, quels que soient les réseaux d'origine et de destination

- Adaptation des datagrammes aux MTU des réseaux
- Durée de vie limitée pour les paquets (TTL)
- Gestions de quelques erreurs
  - En-tête pour IPv4
- Signalisation avec ICMP

Limitations d'IP

- La livraison des paquets n'est pas garantie
- Les erreurs de livraison ne sont pas signalées
- Les chemins multiples du routage IP posent également problème
  - Dé-séquencement possible des paquets
  - Duplication possible

Introduction

Généralités

Limitations d'IP

- Pas de contrôle de flux garanti
- Pas de contrôle d'erreur en dehors de l'en-tête dans IPv4, plus rien en IPv6
- Il y a une notion d'adressage de station mais comment faire pour différencier une application d'une autre ?
  - Comment faire pour qu'un paquet http soit bien remis au navigateur web et pas au client mail ?

Objectifs de la couche transport

- Aller au delà d'IP en étant capable
  - D'assurer la correction d'erreur
  - De gérer le contrôle de flux
  - De fournir un adressage des applications grâce à la notion de port de communication
  - Assurer également l'indépendance des communications entre applications

Introduction

Généralités

Au niveau de la pile de communication TCP/IP on distingue deux grandes familles de protocoles

- TCP (*Transmission Control Protocol* [RFC 793, 1122, 1323])
  - Protocole de transport fiable (mais lent) en mode connecté point-à-point assurant donc le contrôle de flux
- UDP (*User Datagram Protocol* [RFC 768])
  - Protocole de transport non fiable (mais rapide) qui permet la multidiffusion

Dans les deux cas, ces protocoles permettent de distinguer les applications au sein d'une même station hôte et garantissent l'indépendance des communication

- Ils assurent les fonctions de multiplexage/démultiplexage

Introduction

Adressage des applications

Pour gérer le fait que plusieurs applications peuvent fonctionner en parallèle, les protocoles de transport utilisent la notion de port de communication

- Les ports sont des numéros codés sur 16bits

UDP et TCP fournissent, chacun, un ensemble de ports indépendant l'un de l'autre

- Ex: sur le port 22 on peut avoir une application TCP et une application UDP (ce n'est pas le même port)

Certains numéros de ports sont réservés et correspondent à des services spécifiques

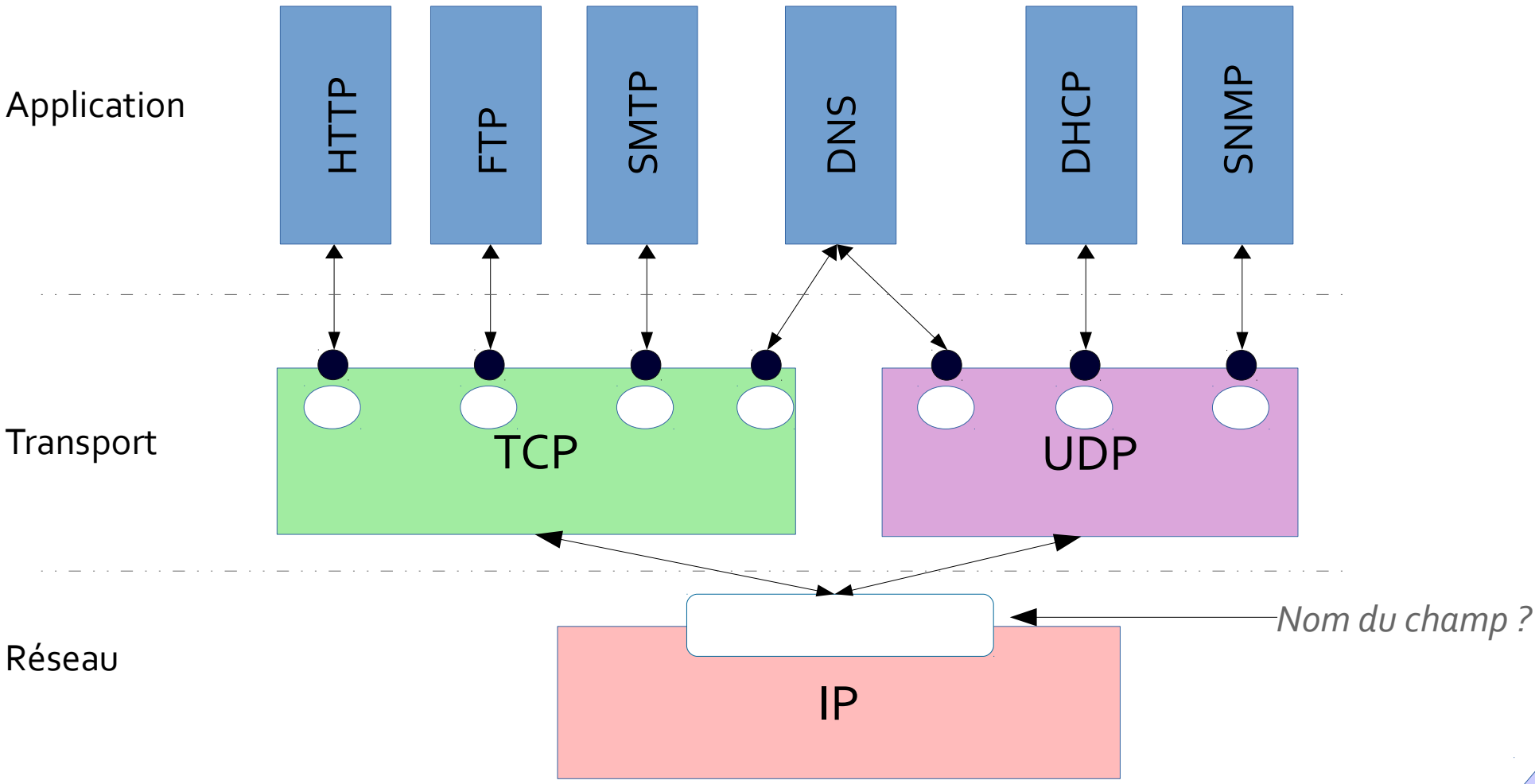
- Ex le port 22 en TCP correspond à SSH

Pour contacter une application, trois informations sont nécessaires

- L'adresse IP de la station hôte
- Le protocole de transport à utiliser
- Le numéro du port sur lequel l'application peut être contactée par le protocole de transport

Introduction

Adressage des applications



User Datagram Protocol (UDP)

Introduction d'UDP

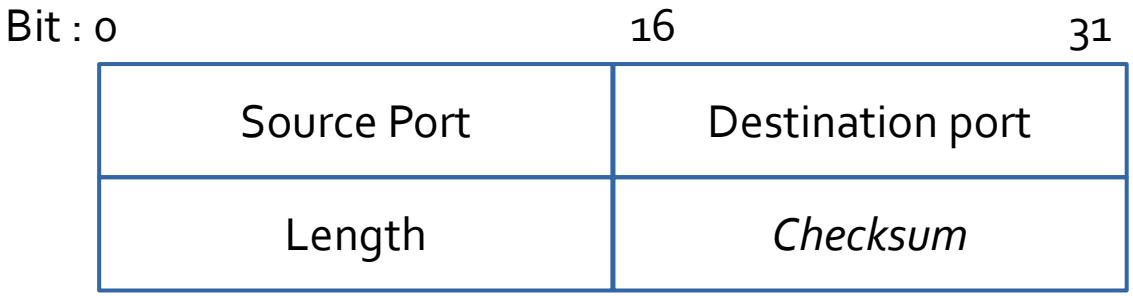
- Protocole de communication sans connexion qui n'apporte pas de fonctionnalités supplémentaires par rapport à IP
  - Il permet uniquement de désigner les numéros de ports sources et destinations par lesquels les applications distribuées vont dialoguer
  - Il assure également de manière facultative un contrôle de l'intégrité des données
- Avec UDP
  - Possibilité de pertes, de dé-séquencement et de duplication
  - Pas de contrôle de flux
  - L'ajout de fonctionnalités complémentaires est laissé aux applications
    - Usuellement il s'agit de la gestion du dé-séquencement par exemple

Avec UDP, il est possible d'exploiter la multi-diffusion IP (*multicast* et *broadcast*)

- Ex: requête DHCP pour obtenir une adresse IPv4 lors de la connexion au réseau

User Datagram Protocol (UDP)

Structure de l'en-tête UDP



- L'en-tête UDP n'a pas d'option et se compose de 8 octets (64bits)
- La taille des données est limitée à 65535 octets
- L'utilisation du *checksum* UDP est facultative mais le champ est obligatoire
  - Il prend la valeur 0 s'il n'est pas utilisé
  - Le *checksum* permet de vérifier l'intégrité de l'en-tête ET des données

User Datagram Protocol (UDP)

Structure de l'en-tête UDP

Il y a deux champs pour les ports UDP dans le paquet

- Un pour le port source
  - Indique le numéro de port sur lequel fonctionne l'application
  - 0 si pas de réponse attendue
- Un pour le port destination
  - Le numéro de port sur lequel le destinataire s'attend à recevoir des paquets
  - Si il n'y a pas d'application à ce numéro de port, un message ICMP est renvoyé et le paquet est détruit par la station destinataire

- Quelques numéros de ports usuels
  - [0, 1023] *Well Known Port Assignment*
  - 7: serveur *echo*
  - 13: serveur *daytime*
  - 53: DNS
- Pour un usage privé, privilégiez les numéros de port dynamiques (>49151)

User Datagram Protocol (UDP)

Modèle client-serveur et UDP

Dans une architecture client/serveur

- Le serveur est démarré sur un numéro de port spécifique et attend qu'un message lui soit adressé
- Le client envoie une requête à destination du serveur (adresse+udp+port)
  - Cela suppose que le client connaît l'adresse IP du serveur et le numéro de port qu'il utilise
  - Pour pouvoir recevoir une réponse du serveur, il doit également écouter sur un port local de sa station
  - Le datagramme UDP qui véhicule la requête est encapsulé dans un paquet IP avec les @IP du destinataire (serveur) et de l'émetteur (client)
- Le serveur reçoit le message du client et obtient ainsi des informations pour lui répondre
  - @IP du client, n° de port
- Tous les élément nécessaires à une communication bi-directionnelle sont maintenant disponibles

Transport Control Protocol (TCP)

Introduction

Contrairement à UDP, TCP ne manipule pas des messages mais des segments

- Un segment TCP
  - En-tête de 20 octets
  - Données
- La taille des données est définie par l'application (i.e. variable)
  - Lorsque les données sont trop petites elles sont accumulées dans un tampon
    - A l'envoi cela permet d'améliorer l'utilisation du réseau
    - A la réception cela évite de solliciter l'application lorsqu'il n'y a pas assez de données à traiter
    - Lorsqu'elles sont trop volumineuses, elles sont fractionnées
  - La taille maximum d'un segment (MSS) ne doit néanmoins pas dépasser 65535 octets (charge utile IP)
  - En pratique on adapte souvent la taille au MTU du réseau (i.e. 1500 octets)
- Chaque segment est émis dans un paquet IP

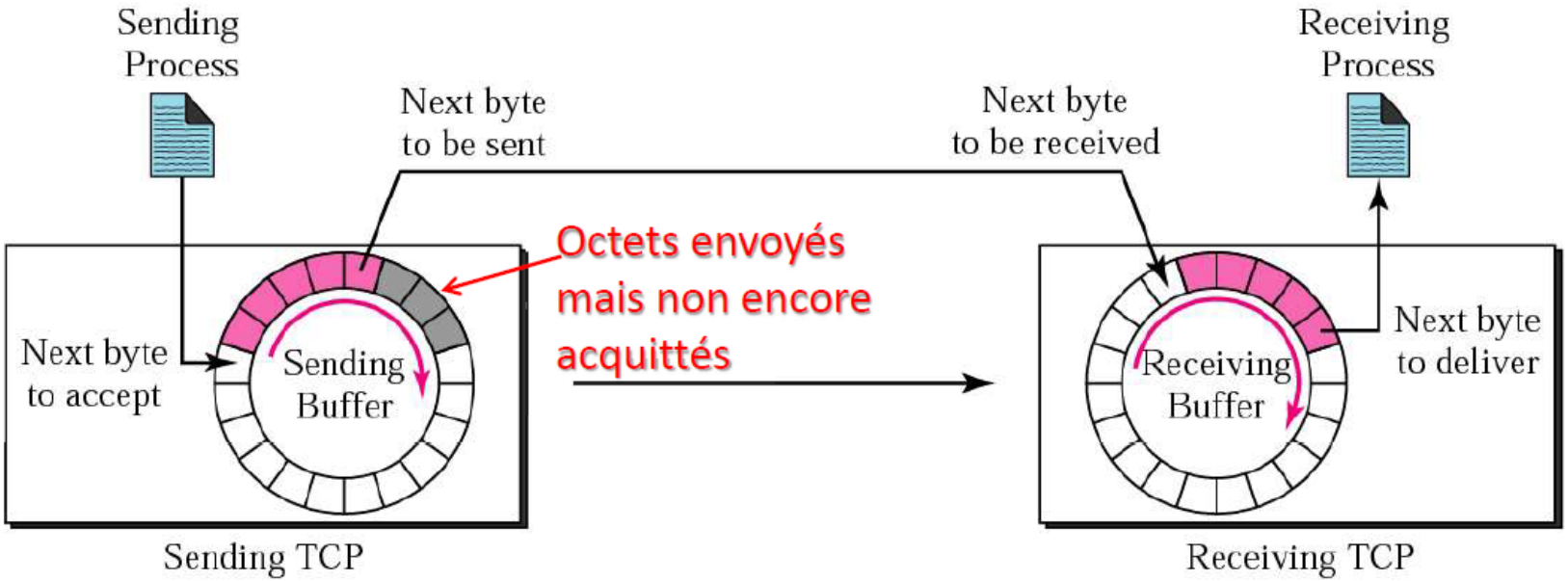
Transport Control Protocol (TCP)

Vocabulaire

- *Buffer* d'émission
  - Désigne l'espace de stockage des données côté émetteur
- *Buffer* de réception
  - Désigne l'espace de stockage des données côté récepteur
- Fenêtre de réception
  - Désigne la qté maximale de données qu'un nœud peut recevoir à un instant t
- Fenêtre d'émission
  - Désigne les données à émettre à un instant t
- Fenêtre de congestion
  - Désigne le nombre maximum de segments pouvant être émis sans attendre d'acquittement
- Taille maximum du segment (MSS)

Transport Control Protocol (TCP)

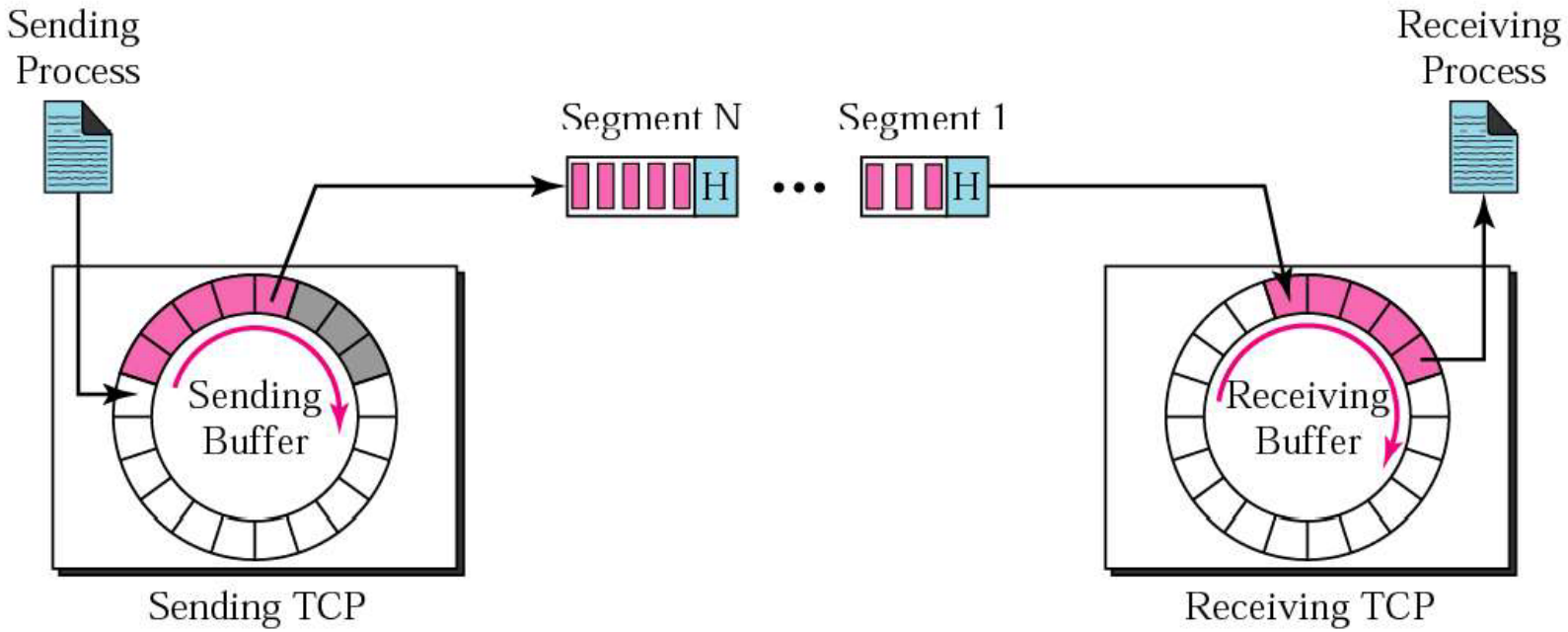
Vocabulaire



Source schéma: McGraw-Hill

Transport Control Protocol (TCP)

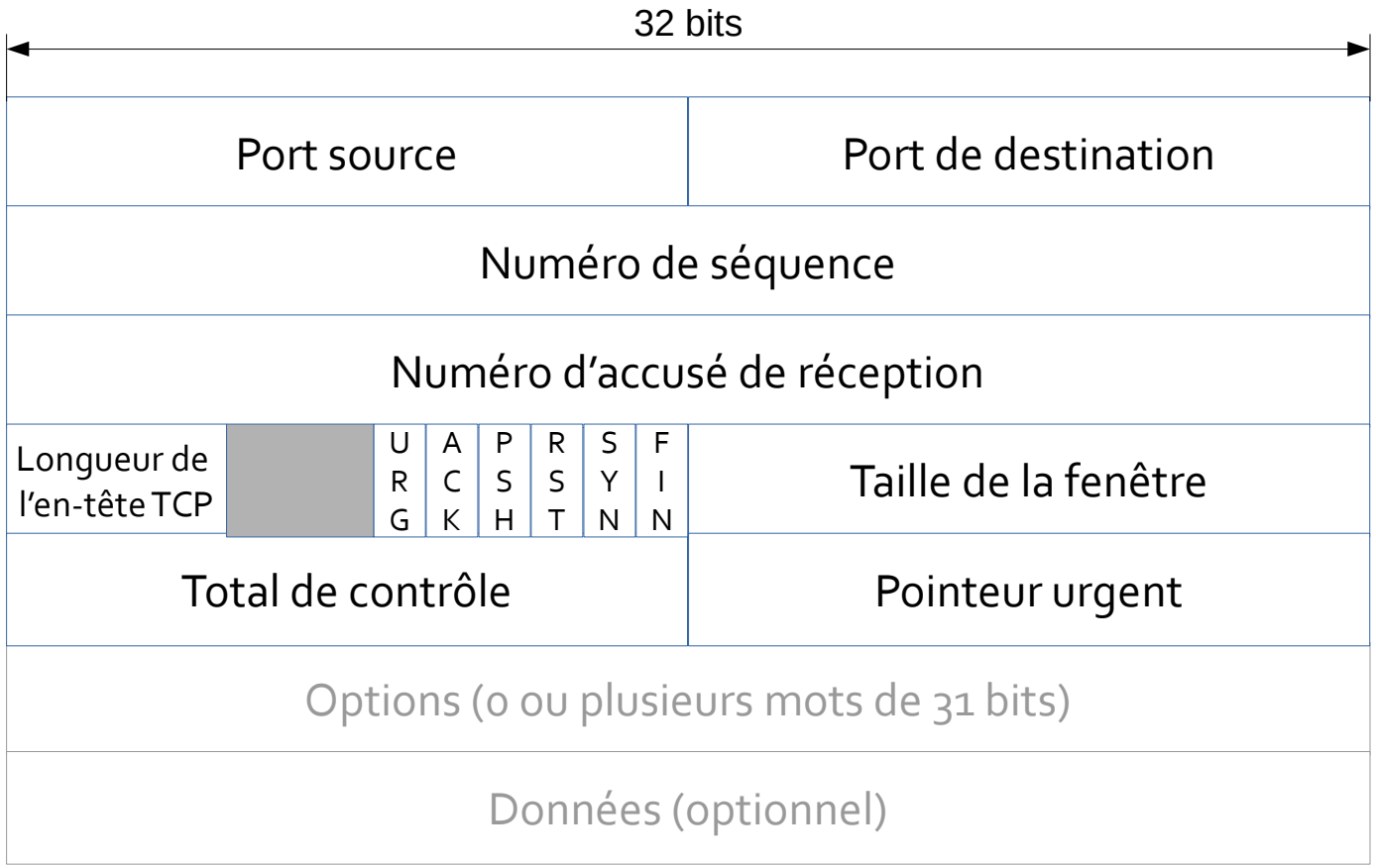
Vocabulaire



Source schéma: McGraw-Hill

Transport Control Protocol (TCP)

En-tête TCP



Transport Control Protocol (TCP)

En-tête TCP

Types

- Données
- Supervision
- Acquittement

Numéro d'acquittement

- Numéro du prochain octet attendu dans le flux
- Si ce numéro est valide, le bit ACK est positionné à « 1 »

Etablissement et fermeture de connexion

- Bits ACK, SYN et FIN

Taille de la fenêtre

- Indique combien d'octets peuvent être transmis après l'octet acquitté
  - « 0 » : signale que la destination n'est pas en mesure de traiter de nouvelles données
    - Pour la reprise, la destination envoie un message avec les mêmes numéros de séquence mais une taille de fenêtre non-nulle

Transport Control Protocol (TCP)

Services

TCP offre les services suivants

- Service de transport fiable en mode connecté
  - Étapes d'établissement, de maintenance et de libération de connexion
  - Garantie de remise des données au récepteur par un mécanisme d'ACK
  - Garantie de l'ordre des données remises (n° de séquence)
- Gestion des données sous forme de flux et non plus de message
- Support du mode full-duplex

Comme pour UDP, l'adresse d'une application utilisant TCP repose sur le triptyque (@IP, TCP, n°port)

- On parle de *Transport Service Access Point* (TSAP) pour le couple (@IP, port)
- Rq: on ne peut pas contacter directement une application TCP, il faut au préalable établir la connexion

Transport Control Protocol (TCP)

Garantie de délivrance des données

En cas de pertes les deux extrémités de la communication sont prévenues

- Utilisation d'un mécanisme d'acquittement lié à un mécanisme de réémission temporisé
  - L'acquittement n'est pas lié à des segments mais au flux d'octets
  - Si au bout d'un certains temps l'acquittement n'est pas reçu, on considère le message comme perdu et la source le retransmet
  - Le temps d'attente avant une réémission automatique est appelé RTO (*Retransmission TimeOut*)
  - Acquittement cumulatif : valide tous les octets précédents par un numéro unique
    - Pour délivrer les données sans « trous », le numéro d'acquittement sera le plus élevé d'une séquence continue
      - Ex : pour la séquence 2,3,4,6,7, l'ACK retourné porte le numéro 5

Transport Control Protocol (TCP)

Garantie de délivrance des données

- Processus
  - La source arme un *timer* à la transmission du segment o
  - Lorsqu'un ACK arrive avec un numéro d'acquittement n, la source met à jour le pointeur vers le plus ancien segment non-acquitté
    - S'il reste des segments, la source ré-arme le *timer*
  - Quand le *timer* expire, la source retransmet le plus ancien segment non-acquitté et ré-arme le *timer*

Cas particuliers

Envoi d'un segment à la fois et perte de l'ACK

=> retransmission du couple data-ack

Envoi d'une rafale de segments, réception sans pertes mais arrivée tardive des ack (après le *timeout*) à l'émetteur

=> retransmission du premier message de la rafale et réponse par un ack portant le numéro du prochain octet après la rafale

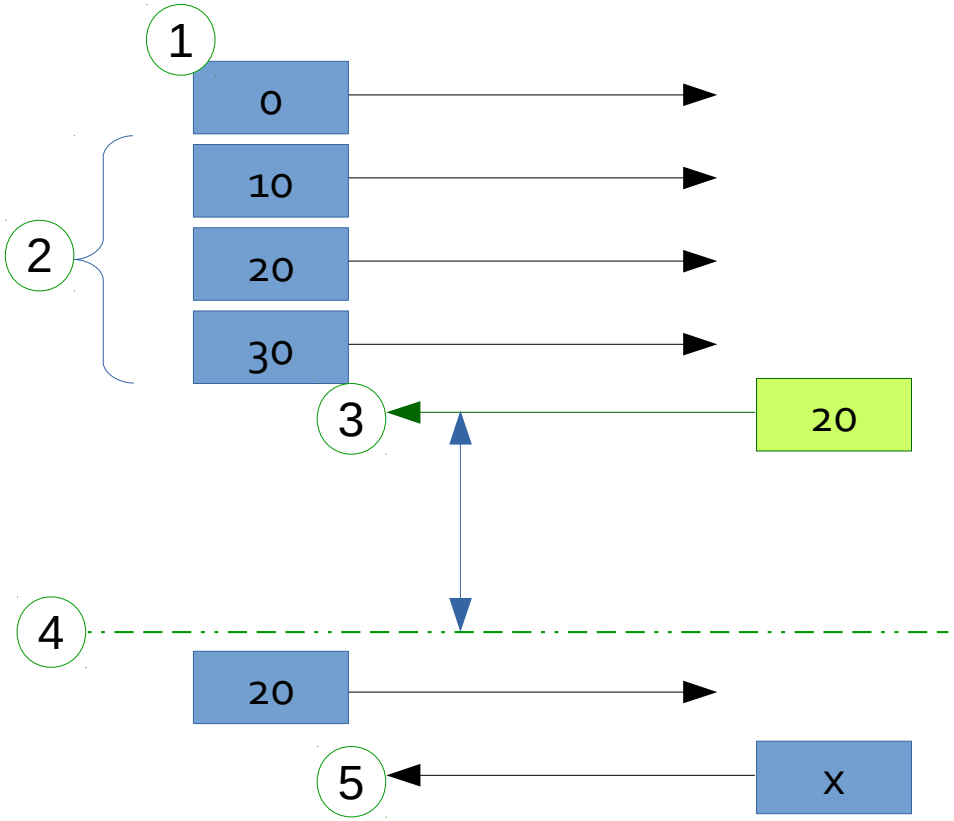
Envoi de deux segments et réception, avant expiration du délai d'attente, de l'acquittement lié au second segment

=> pas de retransmission

Transport Control Protocol (TCP)

Garantie de délivrance des données

Exemple pour un segment corrompu



1_ Activation de T
P pointe vers segment 0
2_ *Pipelining*
3_ ACK# 20 : prochain segment attendu
P pointe vers 20
Activation de T
4_ Expiration de T
Retransmission du segment 20
5_ ACK# x
x == 30 :_____
x == 40 :_____

P : pointeur vers le plus ancien segment non acquitté
T : timer pour attente de l'ack

Transport Control Protocol (TCP)

Garantie de délivrance des données

- Problèmes
  - 1 segment, 1 acquittement renvoyé immédiatement
    - Peu efficace en termes *d'overhead* si l'émetteur envoie de faibles quantités de données
  - Attendre T avant la transmission de l'ACK
    - Si des données à transmettre arrivent entre-temps, *piggy-backing*
    - T= 500 ms
  - Groupage à la transmission (algorithme de Nagle)
  - Syndrome de la fenêtre stupide
    - Profil
      - TCP présente une fenêtre de réception immense comparée à la consommation de l'application (1 octet à la fois récupéré)
      - Premier échange : un segment de taille adaptée remplit le *buffer* de réception
      - La fenêtre passe à 0
      - L'application lit 1 octet
      - TCP demande d'envoyer 1 octet (window=1)
      - La source envoie 1 octet
    - Limiter l'envoi des màj de taille de fenêtre : l'espace disponible doit être significatif

Transport Control Protocol (TCP)

Contrôle de flux

Objectif

- Ne pas étrangler le récepteur

Pour assurer le contrôle de flux, TCP met en œuvre un mécanisme de fenêtre glissante (*sliding window*)

- La taille de la fenêtre s'exprime en octets
  - Cette fenêtre correspond à l'espace disponible dans le *buffer* de réception
  - Une fenêtre de taille T autorise l'émission d'un certain nombre de segments sans attendre d'acquittement
  - La taille de la fenêtre est variable au cours d'une transmission
    - Le récepteur indique que la taille de la fenêtre peut être augmentée ou diminuée en fonction de sa charge
  - Le récepteur maintient une fenêtre glissante pour chacun de ses interlocuteurs
- A la fin de la fenêtre, le récepteur émet un acquittement indiquant l'octet qu'il attend ensuite et une fenêtre de taille « 0 »
  - Sonde fenêtre
    - Message envoyé par la source pour obliger la destination à indiquer une évolution dans la taille de la fenêtre => évitement de blocage de la source

Transport Control Protocol (TCP)

Contrôle de congestion

TCP considère que les erreurs sur un réseau sont causées par des congestions

- Congestion
  - Saturation du réseau
  - Pertes dans les files d'attente des routeurs intermédiaires
- Axes
  - Détection de congestion
  - Adaptation de la vitesse d'envoi

Adaptation de la vitesse d'envoi de l'émetteur

- Définition d'une fenêtre de congestion *cwnd*
- S'il n'y a pas ou peu de congestion, l'émetteur peut accélérer dans la limite autorisée par le contrôle de flux
  - Qté de données en circulation / temps d'aller-retour
- La vitesse dépendra toujours de la fenêtre de réception
  - La quantité de données envoyées mais non acquittées doit être inférieure ou égale au minimum entre la taille de la fenêtre de congestion et la taille de la fenêtre de réception

Transport Control Protocol (TCP)

Contrôle de congestion

Détection de congestion

- Congestion = perte de paquet = trou dans la séquence = répétition de l'ACK du prochain segment attendu (le premier segment du trou)

Phases du contrôle de congestion

- *Slow Start*
- *Congestion Avoidance*
- *Fast Recovery*
  - Recommandé mais facultatif
  - Il n'est par exemple pas présent dans TCP Tahoe au contraire de TCP Reno

Transport Control Protocol (TCP)

Contrôle de congestion

La phase de *Slow Start*

- Au démarrage de la connexion TCP, la taille de la fenêtre de congestion (cwd) est limitée à une valeur faible (classiquement 1MSS)
- Cela signifie que l'on peut envoyer uniquement 1 segment par RTO (*Retransmission TimeOut*)
- A chaque transmission correctement acquittée, la taille de fenêtre de congestion sera doublée
- L'augmentation est donc exponentielle et va s'arrêter lorsque
  - Une perte est détectée (*timeout*)
    - Dans ce cas on reprend le processus (cwnd=1) et on définit un seuil (*ssthres* = *slow start threshold*) qui vaut la moitié de la fenêtre de congestion
  - Le seuil ssthres, s'il a été fixé, est atteint
    - A partir de cet instant on passe en *Congestion Avoidance*
  - Au moins 3 acquittements dupliqués ont été reçus
    - Dans ce cas on exécute un *Fast restransmit* passe en mode *Fast Recovery* si possible

Transport Control Protocol (TCP)

Contrôle de congestion

La phase de *Congestion Avoidance*

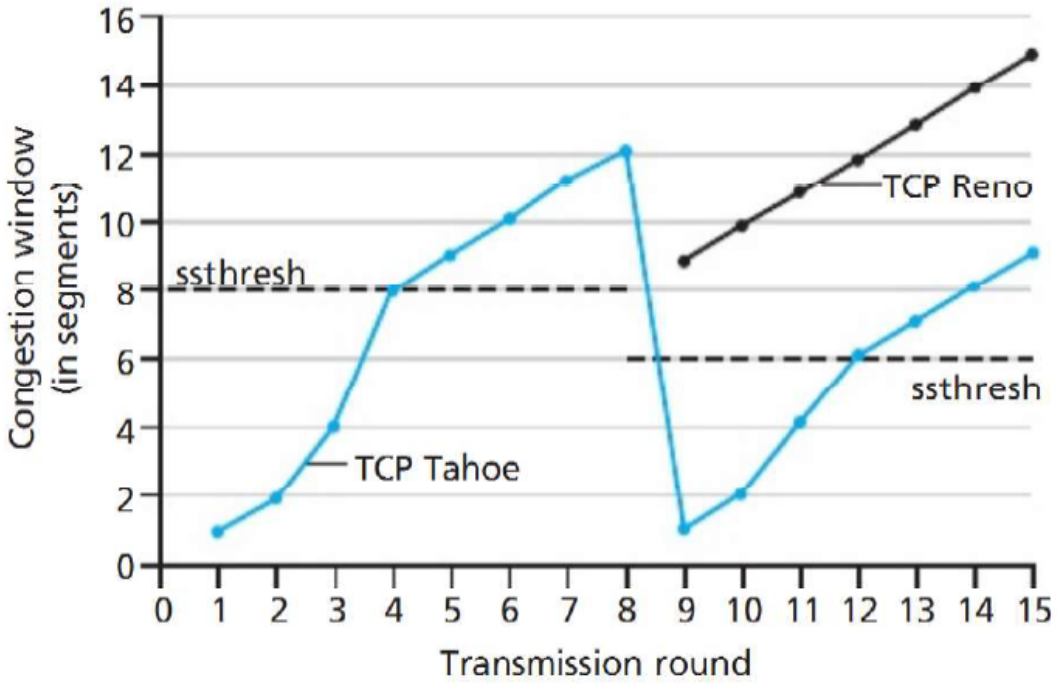
- L'objectif de cette phase est d'éviter/de retarder l'arrivée de nouvelles congestions
- Au lieu de doubler la taille de la fenêtre, on ne l'augmente plus que de 1 en 1
  - Elle s'arrête suivant les même conditions que précédemment

La phase de *Fast Recovery*

- Elle a été introduite par TCP Reno et propose de diminuer la taille de la fenêtre de congestion à la moitié de sa taille plutôt que de repartir à 1

Transport Control Protocol (TCP)

Performances du contrôle de congestion



Source: « Computer Networking », Kurose & Ross

- Sstresh = 8 MSS
- *Threshold* atteint ? *C____on avoidance* et croissance par incréments de 1MSS
- Perte de message (ACK répété pour ancien message)
  - Tahoe : retour à slow start
  - Reno : cwnd = cwnd/2 et c___on avoidance

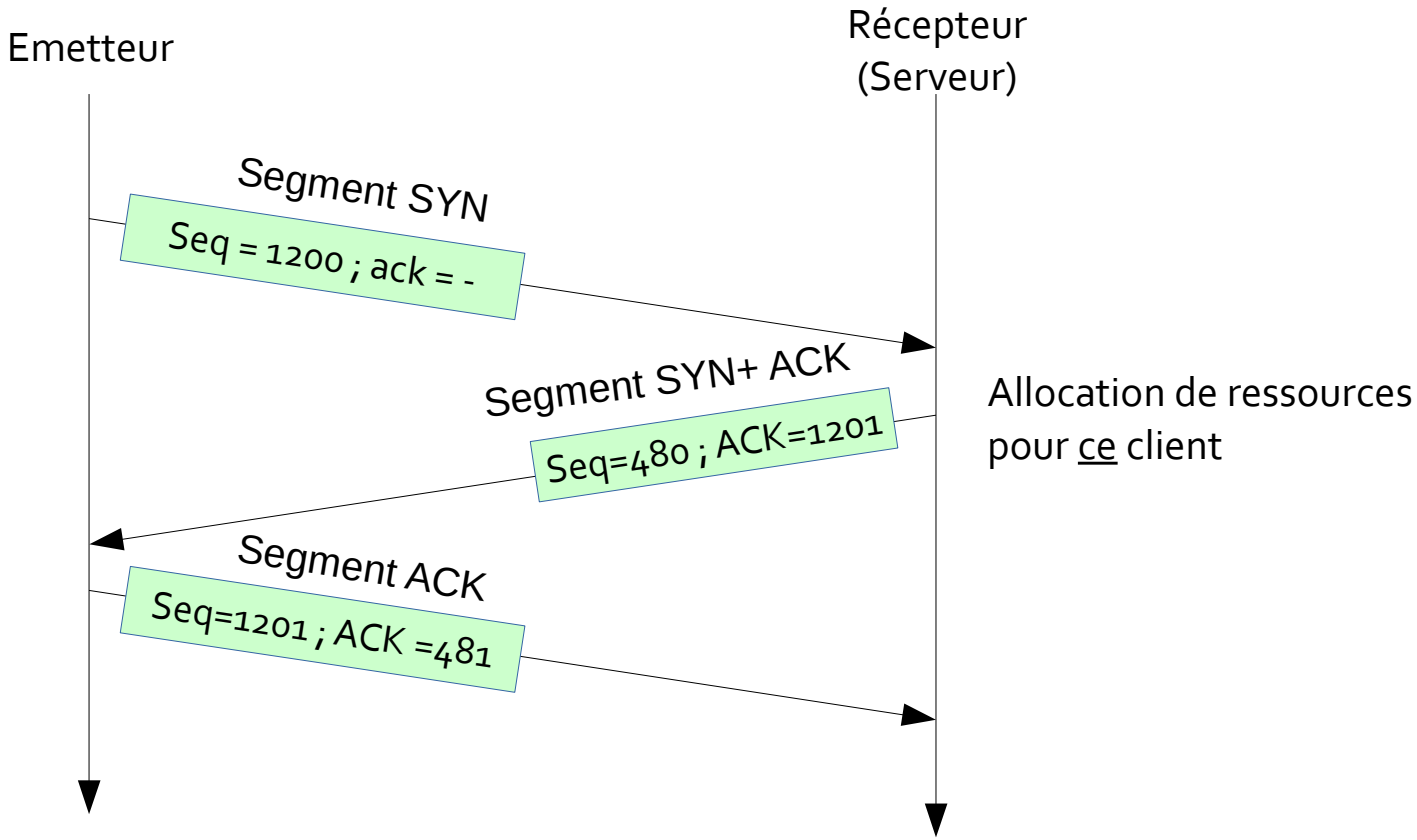
Transport Control Protocol (TCP)

Établissement de connexion TCP : *Three Way Handshake*

- Pour établir une connexion on parle de *Three Way Handshake*
  - Le serveur effectue une ouverture passive en écoutant sur un port
  - Le client effectue une ouverture active en envoyant une demande d'ouverture de connexion sur le port d'écoute du serveur (SYN)
  - Si le serveur est disponible, il acquitte la demande d'ouverture de connexion (SYNACK) et alloue les *buffers* de réception associés
  - A la réception du SYN+ACK, le client alloue ses *buffers* d'émission et envoie un ACK au serveur
    - Il est possible d'ajouter des données lors de ce dernier envoi

Transport Control Protocol (TCP)

Établissement de connexion TCP : *Three Way Handshake*



Une des premières attaque de TCP en 1996 a concerné la phase d'établissement de connexion

- Attaque en déni de service appelée SYN *flood attack*
- La défense associée s'appelle SYN *cookies*

Transport Control Protocol (TCP)

Clôture de connexion TCP : *Four Way Handshake*