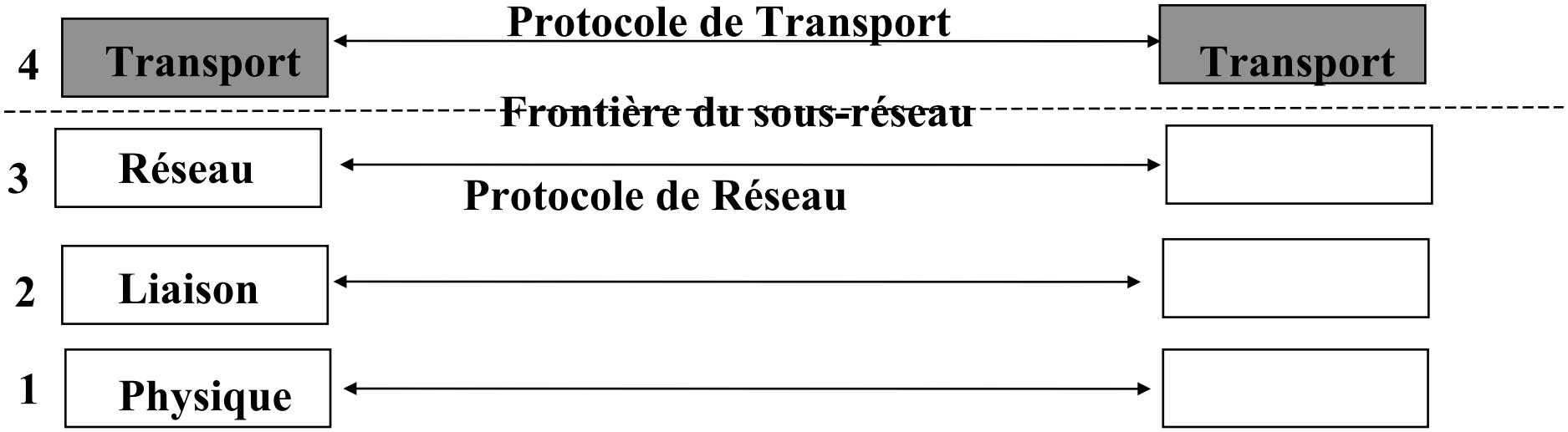
La couche transport

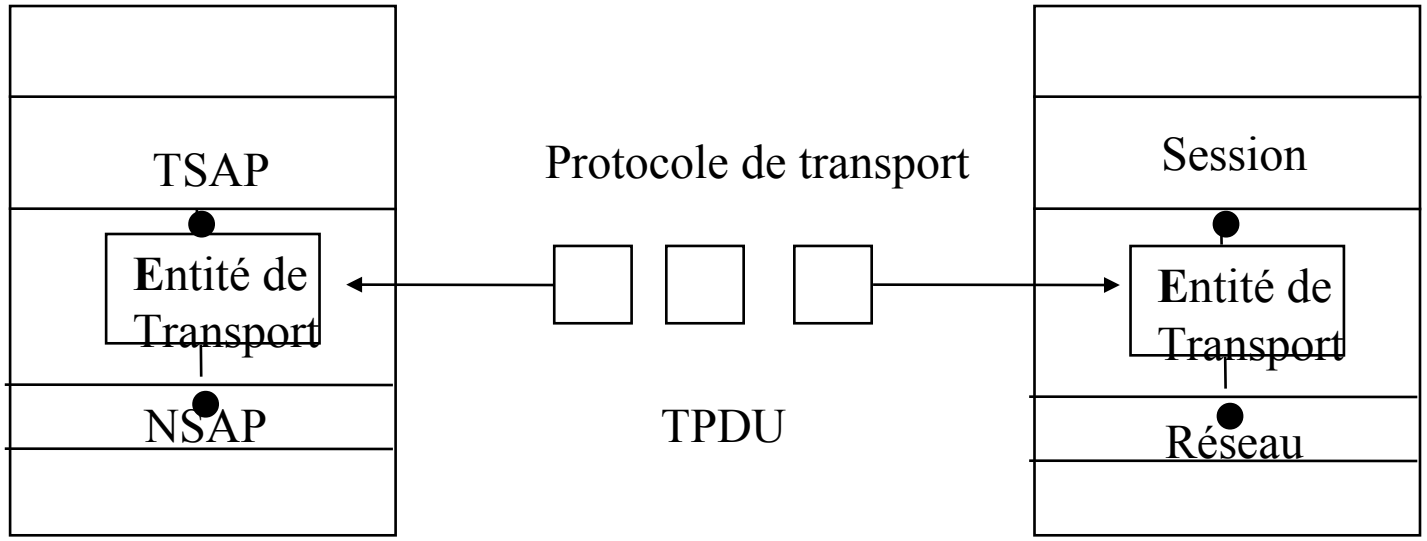


Plan du chapitre

- **Introduction**
- **Type de service**
- **Les protocoles de transport pour l'Internet UDP et TCP**
- **Gestion de la connexion TCP :**
  - **L’identification du port de connexion**
  - **Garantie de délivrance des messages**
  - **L’établissement d'une connexion**
  - **Libération d’une connexion**
  - **Contrôle de flux et mémorisation**
- **Un exemple : TCP (Transport Control Protocol)**
- **QoS**

Introduction

- **Transfert d'informations de bout en bout (pas de vision des sites intermédiaires)**
- **Mécanismes possibles : T-SAP avec fiabilité, performance, efficacité, sûreté et économie**
- **Services pour la couche application (resp. session OSI)**



Multiplexage de l'@IP : le port

En pratique le T-SAP est un numéro appelé “port” pour un service offert dans une machine.

Un couple (@IP, n° port) identifie donc de manière unique un service dans l'Internet.

Ce couple est utilisé pour définir chaque extrémité des “sockets” Unix.

NB: le nombre des numéros de ports possibles dépend de la taille du compteur (2 octets). L'OS et différents services “standards” utilisent les premiers numéros. Il est prudent d'utiliser des numéros de ports > 1024 dans vos applications …

Ex: service http sur le port 80 (tcp), DNS sur 53 (udp/tcp), time sur 37 (tcp), SSH 22 (tcp), SMTP 25 (tcp), Minecraft 25565 (tcp/udp), IPP 631 (tcp), ...

Types de service

•Orienté connexion (TCP): ex physique le réseau d'eau

•Une connexion de niveau réseau s'appelle un circuit et occupe/réserve des ressources. Le circuit peut être virtuel s'il n'est pas physiquement établi comme c'est le cas dans le logiciel TCP.

•Dans le cas de la connexion virtuelle TCP, il n'y a pas de chemin associé au circuit dans le réseau alloué à l'établissement de la connexion. Le circuit virtuel est simulé de manière logicielle en se reposant sur le service de communication non-fiable IP.

•Notez que dans IP chaque paquet est indépendant de la connexion TCP et peut etre routé différemment des autres pendant l'utilisation de la connexion virtuelle.

•A la différence des circuits physiques ou tous les paquets circulant sur un circuit empruntent le même chemin, cf téléphonie analogique, dans le cas d'une connexion virtuelle le logiciel émule le comportement d'un circuit pour la source et la destination.

•La connexion virtuelle TCP est réalisée à l'aide de logiciels et de ressources mémoires pour fournir un flux d'octets au dessus de IP.

•Orienté sans connexion (UDP): ex physique le réseau postal

•L'unité de données autonome est appelée datagramme (cf la lettre)

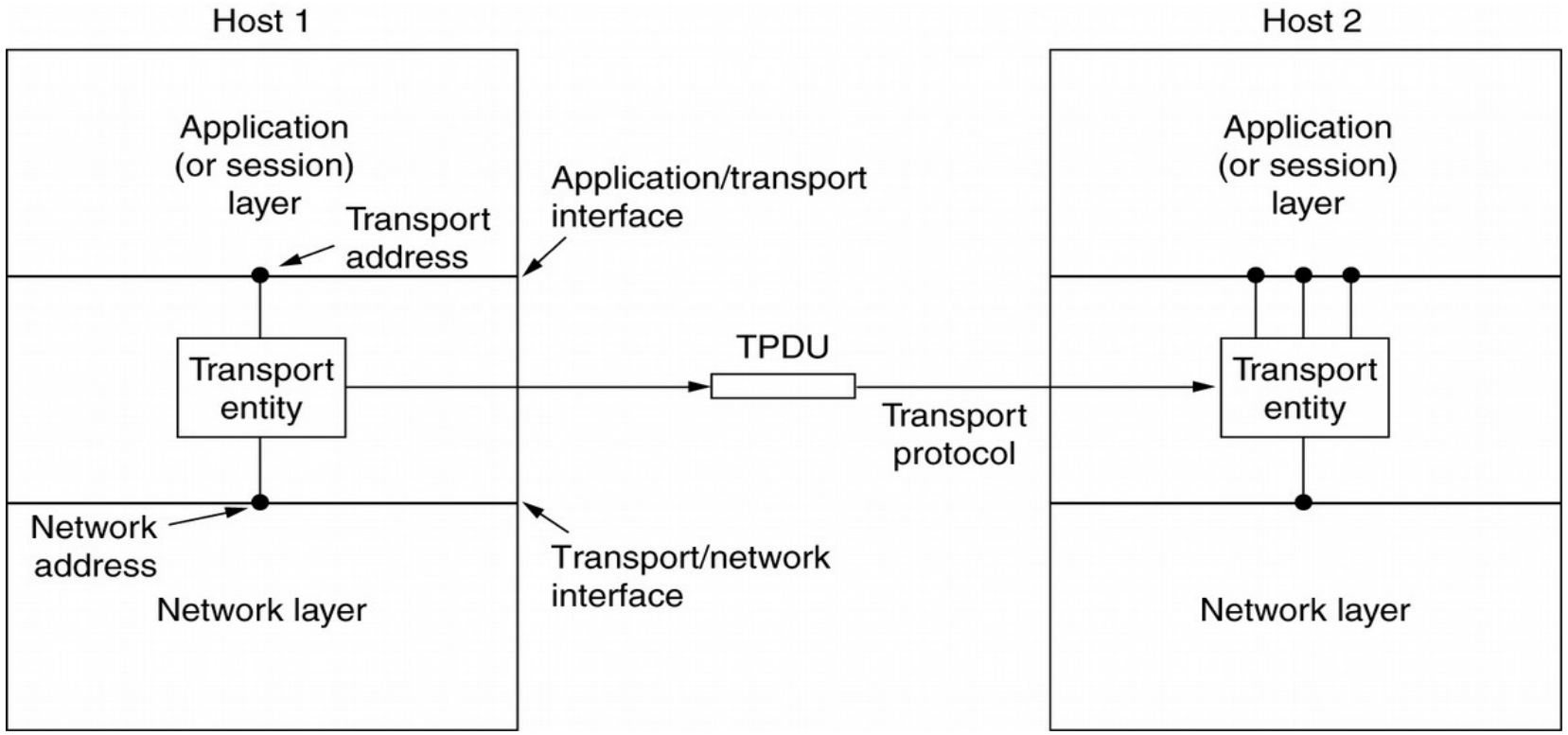
•Chaque datagramme est envoyé indépendamment des autres et est « routé » séparément

•Des datagrammes successifs peuvent emprunter des chemins différents dans le réseau

•Il peut donc y avoir déséquencement des paquets

•Le protocole UDP ajoute juste la notion de port à la couche IP

SAP Provided by Transport to the Sockets



Les services de la couche transport d'Internet UDP et TCP

→ Rappel: le niveau 3 "IP" est un service pour le niveau 4: sans connexion, non fiable, avec déséquencement possible, délai de transfert très variable, fragmentation/reassemblage suivant MTU

•Niveau 4 : deux types de services suivant les besoins de l'application à développer

- Sans connexion, aucune QoS → User Datagram Protocol (UDP)

- Avec connexion virtuelle (ordre, fiable) → Transport Control Protocol (TCP)

TCP: ouverture et fermeture de connexion, vision flux d'octets, contrôle de flux, buffering et récupération des erreurs (ack/réemission), garantie d'ordonnancement des octets, données urgentes

•TCP et UDP se sont imposés naturellement car couvrant la plupart des besoins des applications.

•TCP pensé au départ pour le transfert de fichiers et les applications où la fiabilité et l'ordre d'arrivée en séquence sont nécessaires (notion de flux). Ex terminal virtuel à distance, FTP, ...

•UDP lorsque la perte de quelques paquets n'est pas un problème, les applications "pseudo-RT" (voix, jeux, ...)

•Les normes OSI de QoS ont été définies en parallèle mais elles sont arrivées trop tard et correspondent mal aux besoins des applications informatiques utilisant l'Internet.

•NB: Différents niveaux de QoS peuvent etre implantés au niveau de application et utilisant les services du niveau transport sous-jacent.

Intérêt / inconvénient des 2 types de services

- **Avec connexion :**

+ pas de dé-séquencement, l'ordre est préservé du point de vue de l'application, notion de flux possible

+ le plus souvent fiable, faible pertes, cf téléphone analogique

+ ressources réservées au départ → garantie de qualité de service et non-congestion ultérieure

± facile de facturer / contrôler

- ressources réservées inutilement si non-utilisées

- temps d'acheminement initial plus long car temps d'établissement de la connexion au départ → problème réactivité et temps réel

Sans connexion :

+ pas de ressources réservées inutilement

+ défaillance d'un routeur → pertes seulement des paquets concernés, adaptation rapide

+ temps de traitement rapide

- congestion résolue difficilement

- qualité de service difficile à garantir

Les services du protocole UDP

Complémentaire à TCP : simplicité et multiplexage avec ports

- **Mode sans connexion (datagramme) : les messages sont indépendants les uns des autres**
- **Garanties minimales : aucun contrôle de flux ni récupération d'erreur : pas plus de garanties que IP**
- **En-tête UDP :**

Port source	**Port destination**
Longueur	**Détection d'erreur**
Données	

Les mécanismes du protocole TCP

•Transmission en tranches (segment) et re-assemblage ordonné du flux d'octet

–Rangement des octets dans des buffers : flux d'octets bi-directionnel

•Préservation de l'ordre d'émision des octets

– Numérotation implicite des octets transmis grâce à des compteurs

•Multiplexage @IP, plusieurs applications possibles : numéro de port

•Service orienté connexion : assure des transmissions fiables

–Ouverture et libération de la connexion, structures de buffering, compteurs

•Performance: notion de fenêtre glissante

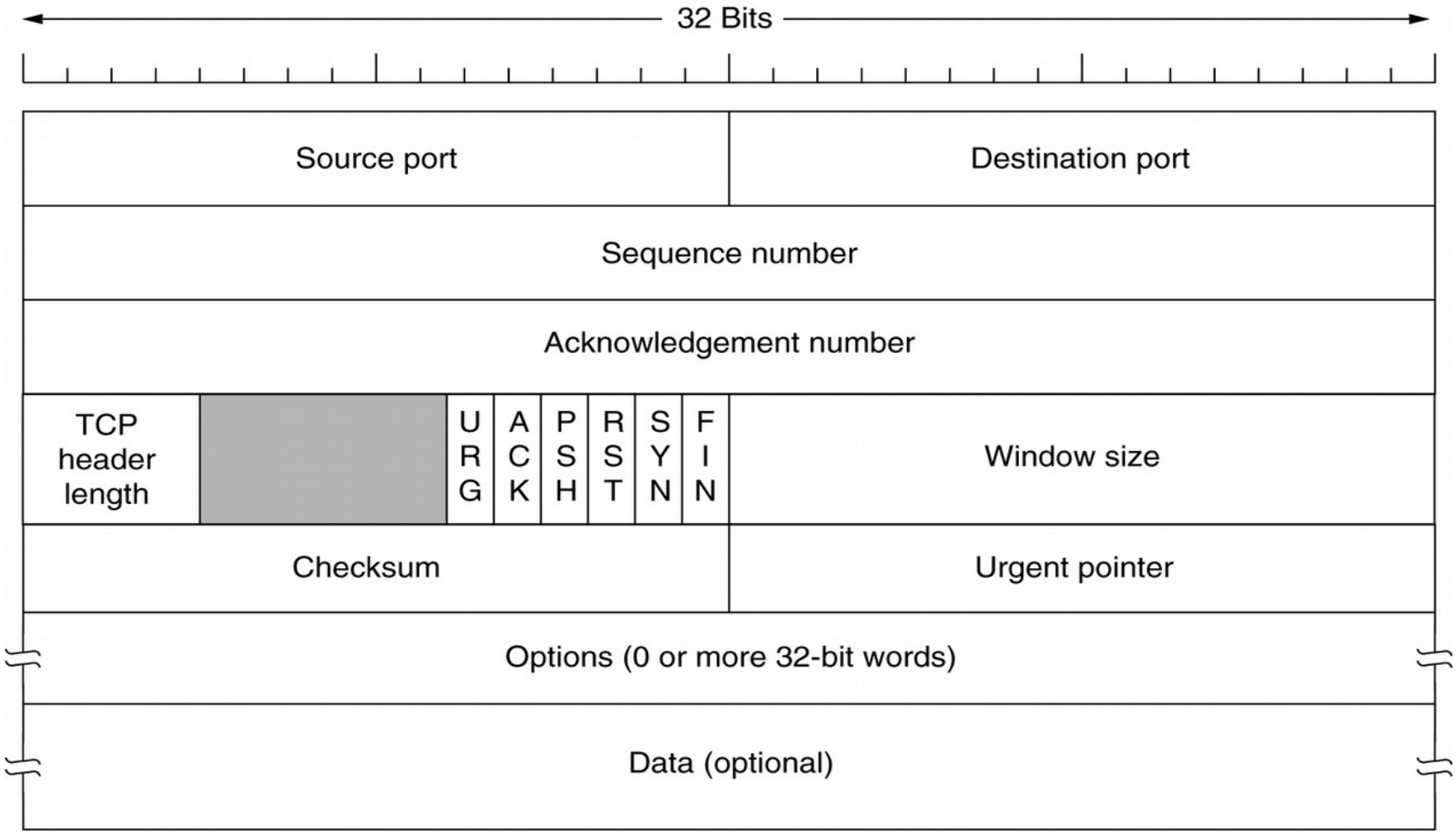
•Contrôle de flux : le récepteur contrôle l'émetteur grâce au compteur d'acquitement et à la taille de fenetre

•Contrôle de congestion global : chaque connexion du réseau limite son débit en fonction des parametres dynamiques du réseau, ajustement avec la taille de fenêtre variable

•Détection des donnees perdues et récupération des erreurs par réémission automatique, cf. compteurs

•Détection d'inactivité, reprise automatique, cf. Compteurs

The TCP Segment Header



TCP Header.

Établissement d'une connexion ?

- **Hypothèse : connexion bi directionnelle entre deux entités**
- **Deux Problèmes :**
  - **Identifier une connexion**
  - **Réaliser une connexion**

Identification de la connexion

- **Comme le niveau transport est mis en œuvre au dessus d'une couche réseau non fiable :**

  plusieurs exemplaires de la même unité de protocole en cours de transmission ?

  reconnaître et éliminer les doublons ?

  → paquet corrompu

  → perte de paquet

  → « Retard » de paquets

  Comment définir une connexion sans ambiguïté ?

  Comment associer une donnée à une connexion ?

Identification de la connexion

- **Solutions**
  - **L'OS mémorise les IDs de connexion déjà utilisés.**
  - **Pour reconnaître les doublons de demandes d'ouverture de connexion, l'OS conserve les références des connexions fermées pendant un temps fixé (timer).**

Problème : le nombre d'id de connexion n'est pas infini, cf. mot de 32b

reprendre à un moment les ids déjà utilisés,

ne pas recevoir des vieux paquets en transit.

→ durée de vie limitée des paquets et timer de connexion fermée, ...

Realisation de la Connexion dans TCP

- **Ouverture à trois étapes avec couple de références → HANDSHAKE**
- **L'identificateur de connexion et le numéro de début de séquence « aléatoire sur 32bits» sont confondus:**
  - **l'id sert de numéro initial (ATTENTION, par défaut wireshark affiche à partir de 0)**
  - **Les paquets suivants n'ont plus d'id de connexion mais seulement un numéro de séquence (id+nb octets déjà transmis)**
- **Connexion-Id calculé sur une horloge de période = 4 us.**
  - **Au pire Id unique pendant 2^{32}* 4 microSec = 4 heures**
  - **Problème: l'évolution des n° Seq dépend du débit de la connexion et on peut arriver à des cas d'ambiguité par rapport aux id initiaux si le transit est très très lent ou très rapide**
    - **Solution on attend « durée de vie max des segments en transit » (fixée à 120s) avant de ré-ouvrir une connexion sur la même paire de ports (cela pourrait être affiné en fonction du débit de la connexion)**
- **Quand un ordinateur plante : On attend la durée de vie maximum d'un segment. (Par défaut pour TCP: 120 s) , pour être sûr de vider hors du réseau les derniers segments en transit.**

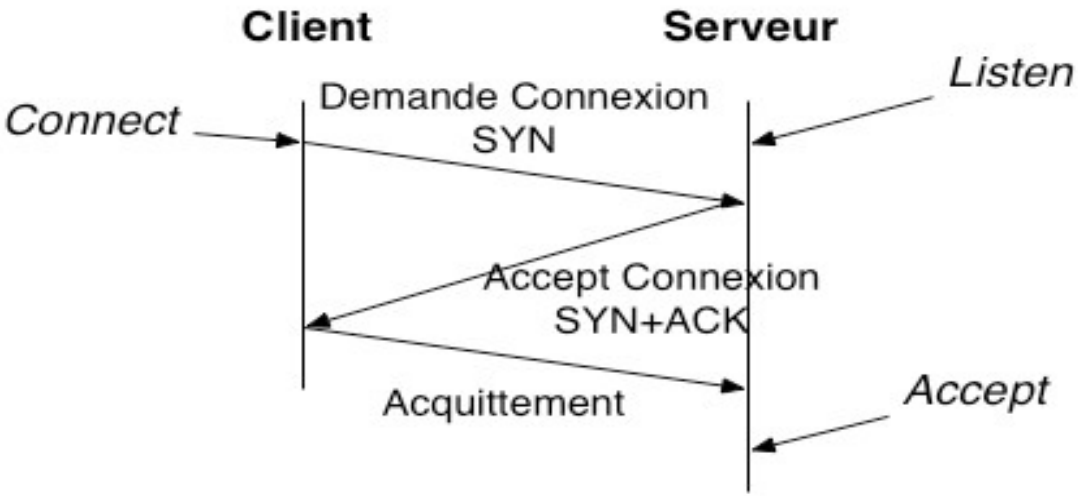
Etablissement d'une connexion TCP (1)

- **Des paquets particuliers pour ouvrir la connexion**
  - **Demande de connexion (Connection Request) : Flags SYN=1 et ACK =0**
  - **Acceptation de connexion (Connection Confirm) : Flags SYN= 1 et ACK=1**
  - **Acquittement de l'acceptation de connexion : Flags SYN= 0 et ACK=1**

Des options sont possibles :

- **Taille maximale des paquets**
- **Convention pour le contrôle de flux sur réseau haut débit ...**

- **Ouverture**
  - **Le protocole est un protocole à trois phases: « 3 ways handshake »**



Etablissement d'une connexion (2)

- **Trois paquets sont nécessaires pour garantir qu'il n'y ait pas d'ambiguité sur des demandes de connexions (doublon, timers de réémission)**
- **Le numéro de séquence initial permet de différencier des demandes de connexions dupliquées**
- **Flag Reset en cas de duplication (ou autre incohérence)**
- **Les numéros de séquence initiaux sont calculés à partir d'une horloge système de période 4 microsecondes. Cela garantit l'unicité d'un paquet d'ouverture de connexion pendant 4 heures**
- **Les No Séquences servent à numéroter aussi les octets de données**

- **Problème: l'évolution des NoSeq des données dépend du débit de la connexion.**
  - **On peut arriver à des cas d'ambiguité par rapport aux id initiaux (zone interdite par timer de 120s)**
- **Solution : on attend la durée de vie maximale d'un paquet sur Internet avant de ré-ouvrir une connexion sur la même paire de ports (2 minutes)**

Libération d'une connexion(1)

- **1ère solution:**
  - **Fermeture brutale, quand on veut fermer la connexion d'un coté , on ne se soucie pas de l'état de l'autre. Comme au téléphone, on raccroche.**
  - **Comment l'interlocuteur le sait-il ? Est-ce une panne ? Un silence ?**
- **2ème solution**
  - **Fermeture douce :**

    Problème :Il faut éviter que la rupture de la connexion ne provoque des pertes de messages. Pour cela, il faut que les deux entités de transport aient la même vision de la connexion.

    ->la connexion est gérée comme deux demi connexions unidirectionnelles

    → … →

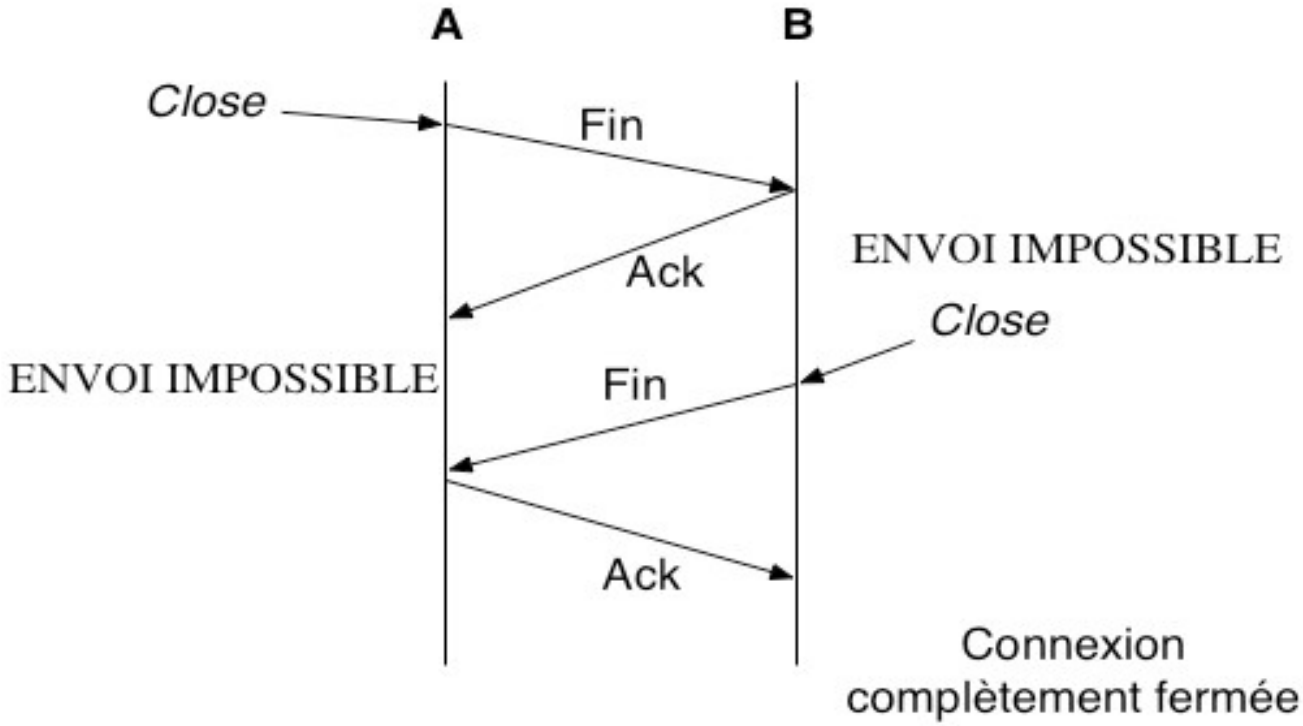
    ← … ←

  - **On peut alors fermer en émission, et continuer à recevoir, si l'autre n'a pas fini d'émettre, avant de fermer definitivement.**

Fermeture de connexion primitive close

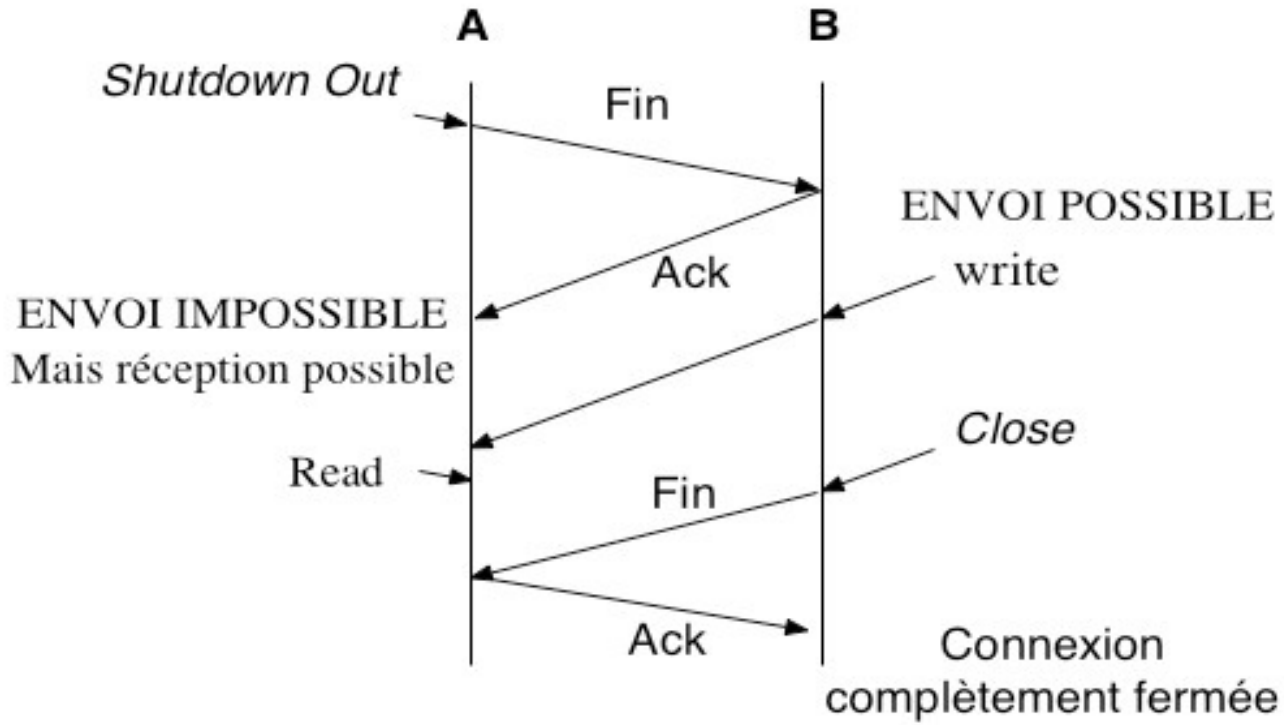
→ **fermeture par demi-connections**

- **Flag FIN pour signifier la demande de fermeture**
- **L'entité distante acquitte pour qu'il n'y ait pas d'ambiguité sur l'état de la connexion**



Fermeture de connexion primitive shutdown

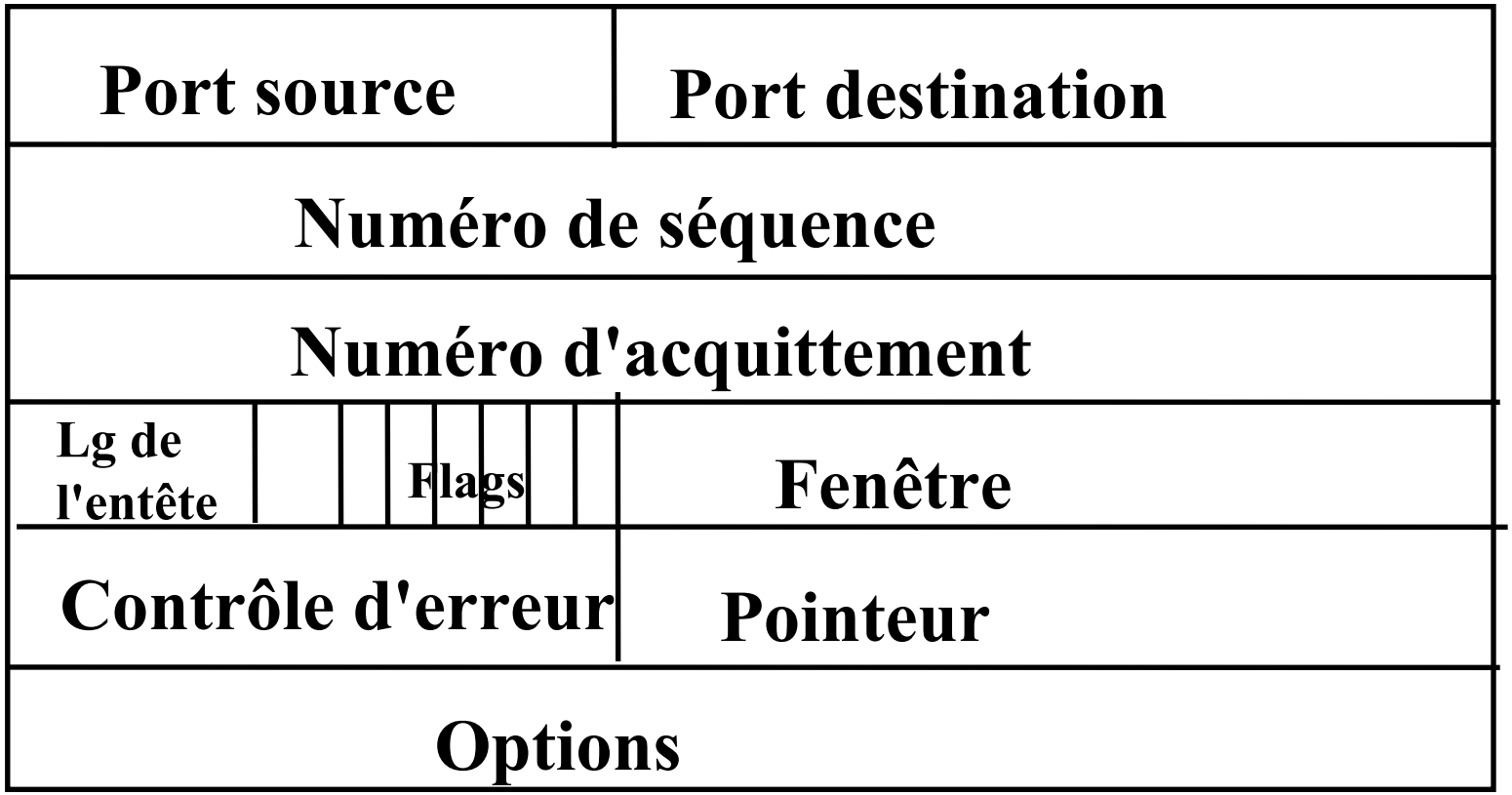
→ fermeture non brutale



Le Transport TCP : Envoi de messages sur le flux d'octets

- **Segmentation:**
  - **Flag EOM pour dernier paquet d'un message segmenté (option pour délimiter des messages dans le flux d'octets)**
- **Contrôle de flux et d'erreur:**
  - **Flux d'octets et non de message**
    - » **Champs Numéro de Séquence : Numéro du premier octet**
    - » **Bit Ack= 1: Champs Numéro d'acquittement significatif**
    - » **Champs Numéro d'Acquittement : Numéro du dernier octet acquitté**
  - **Fenêtre à anticipation variable**
    - » **Nombre d'octets pouvant être expédiés après le numéro d'acquittement**
- **(Données Urgentes**
  - **Flag URG=1, Champs pointeur indique le début des octets à traiter en priorité)**

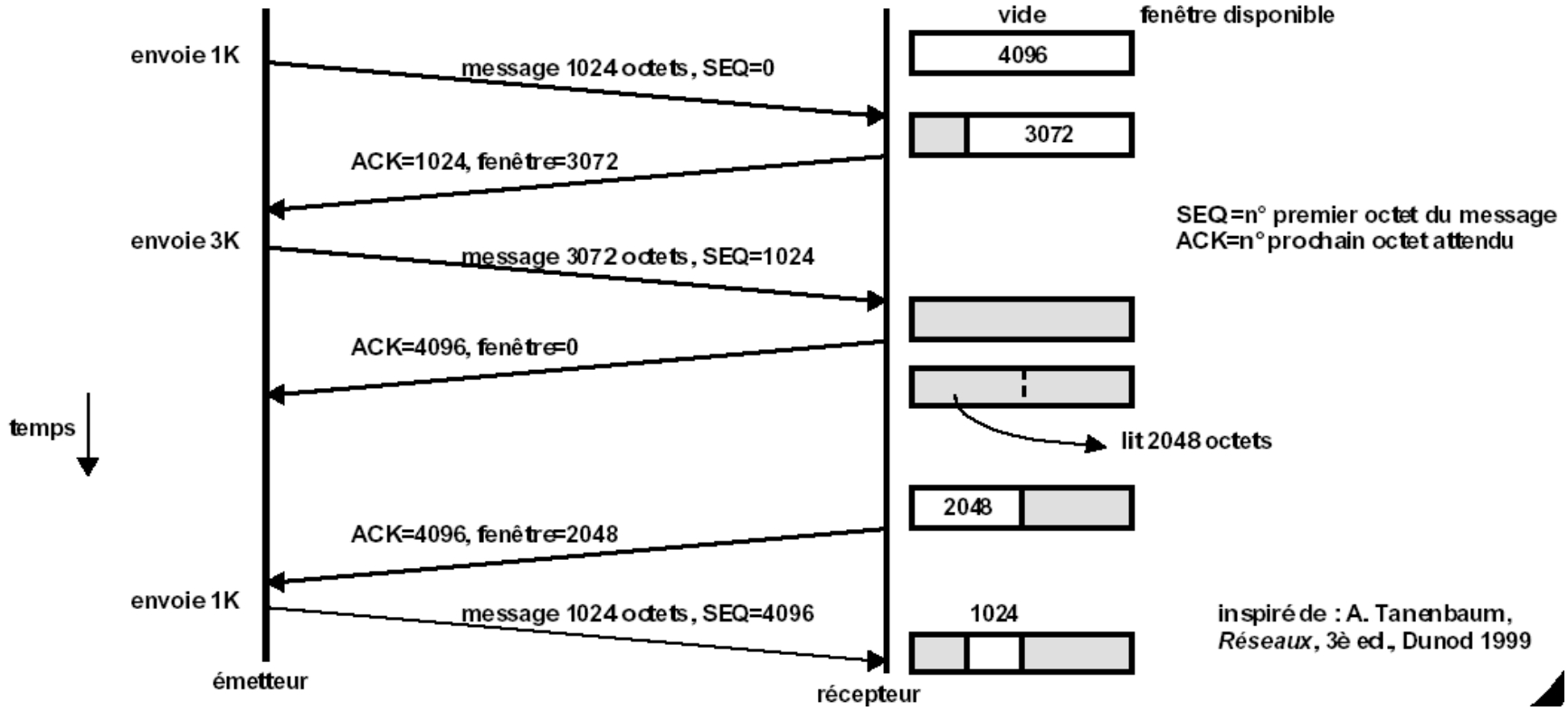
L'entête TCP



- Flags: Urgent, Ack, Eom, Rst, Syn, Fin

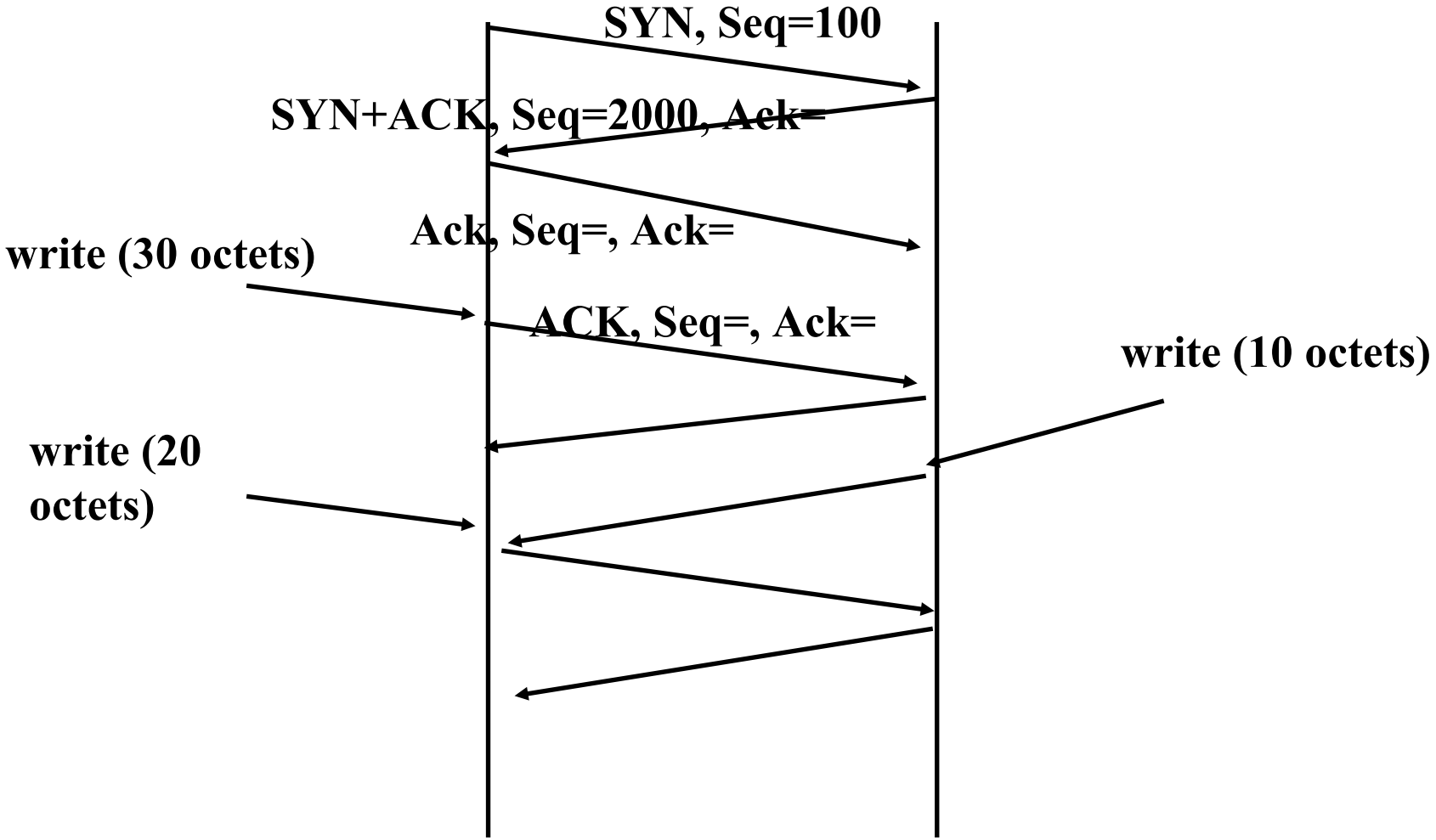
Exemple de buffer TCP
(source S.Krakowiak)

- Le récepteur accuse réception de chaque transmission en envoyant
  - le numéro du prochain octet attendu
  - la taille de la fenêtre de réception (capacité disponible pour recevoir des octets)
    - taille de fenêtre nulle = "arrêtez de m'envoyer des données"

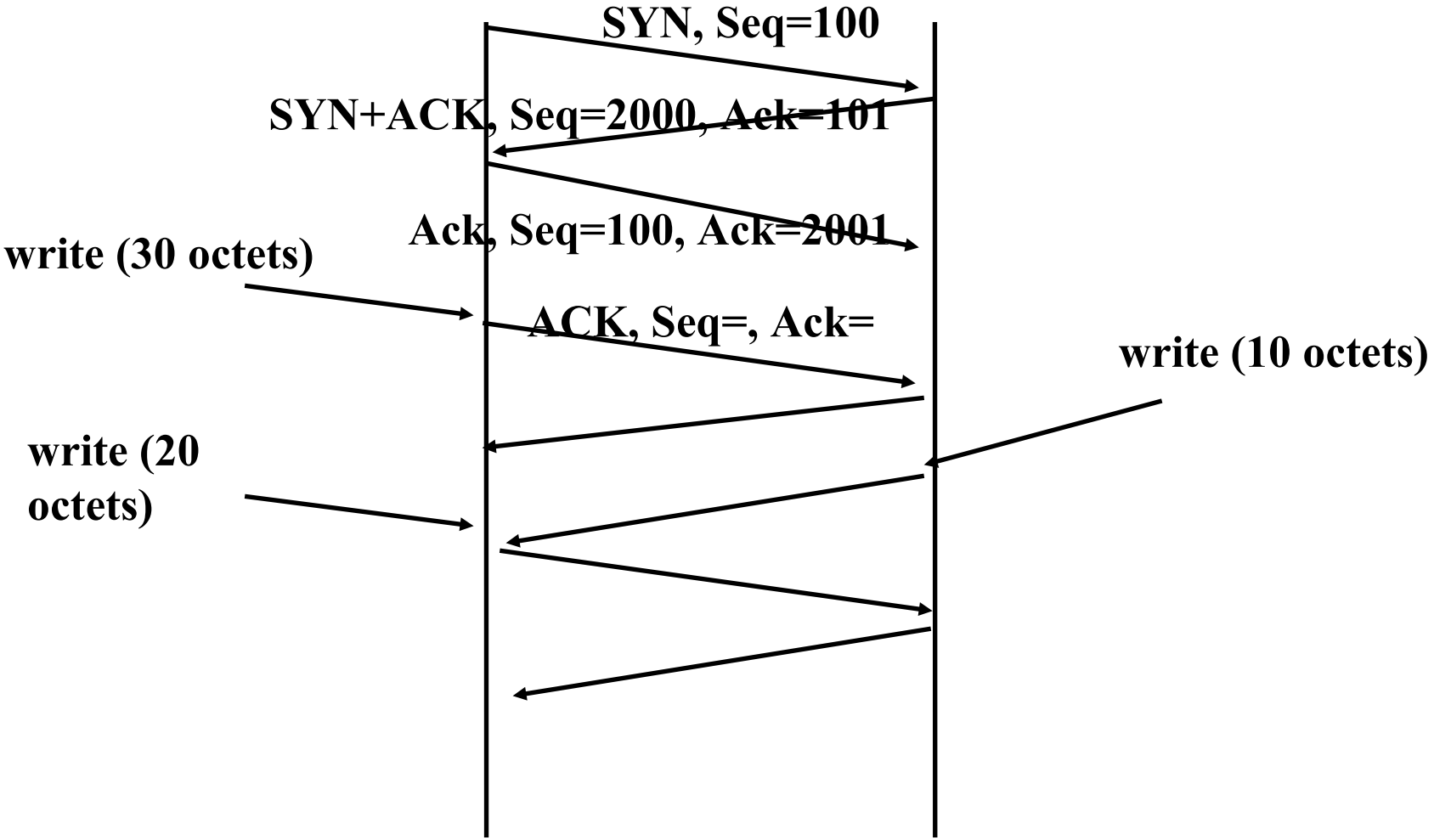


inspiré de : A. Tanenbaum, *Réseaux*, 3è ed., Dunod 1999

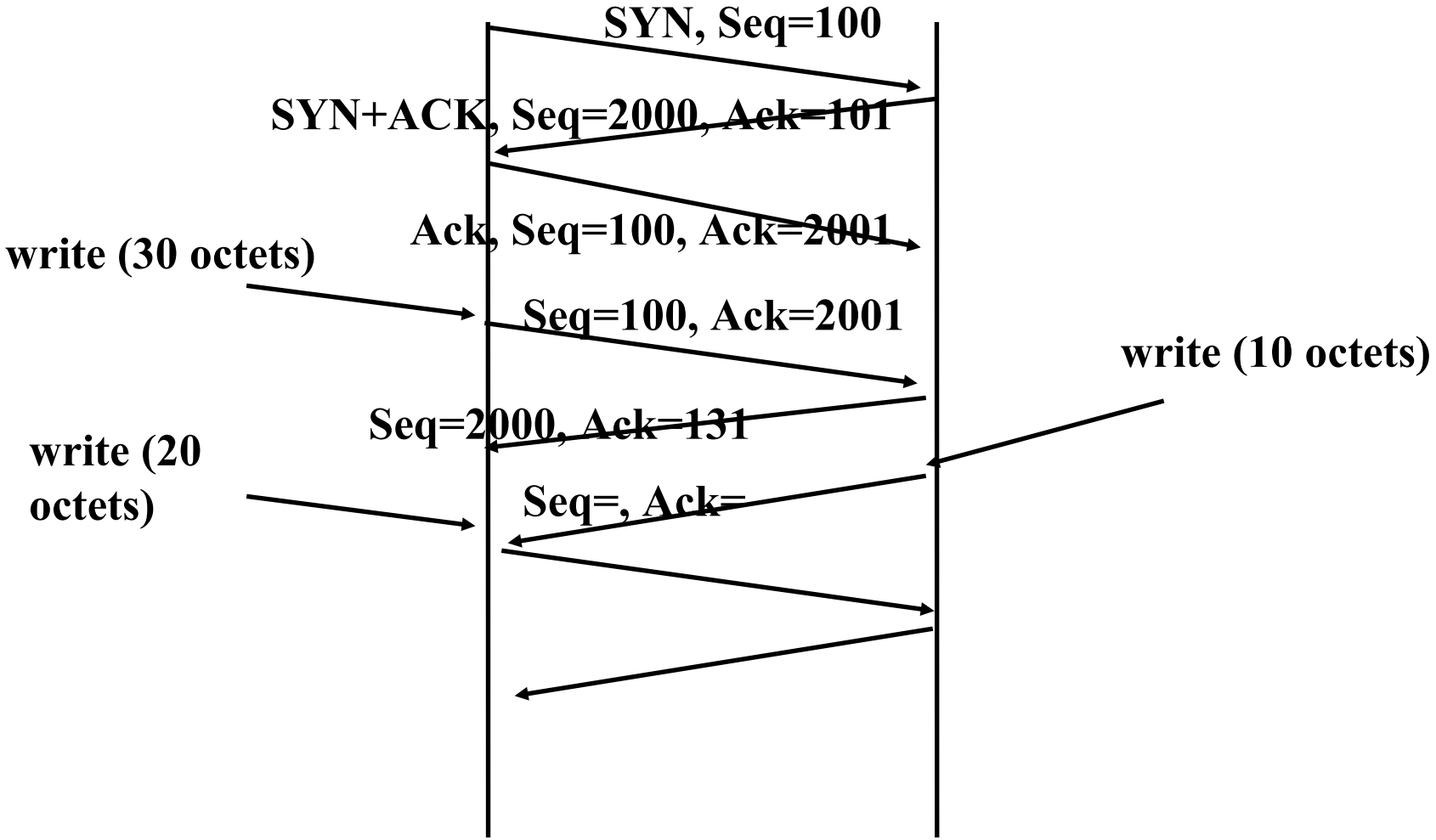
- **Exercice : Donnez les numéros de séquence et d'acquittement des paquets dans l'échange TCP suivant:**



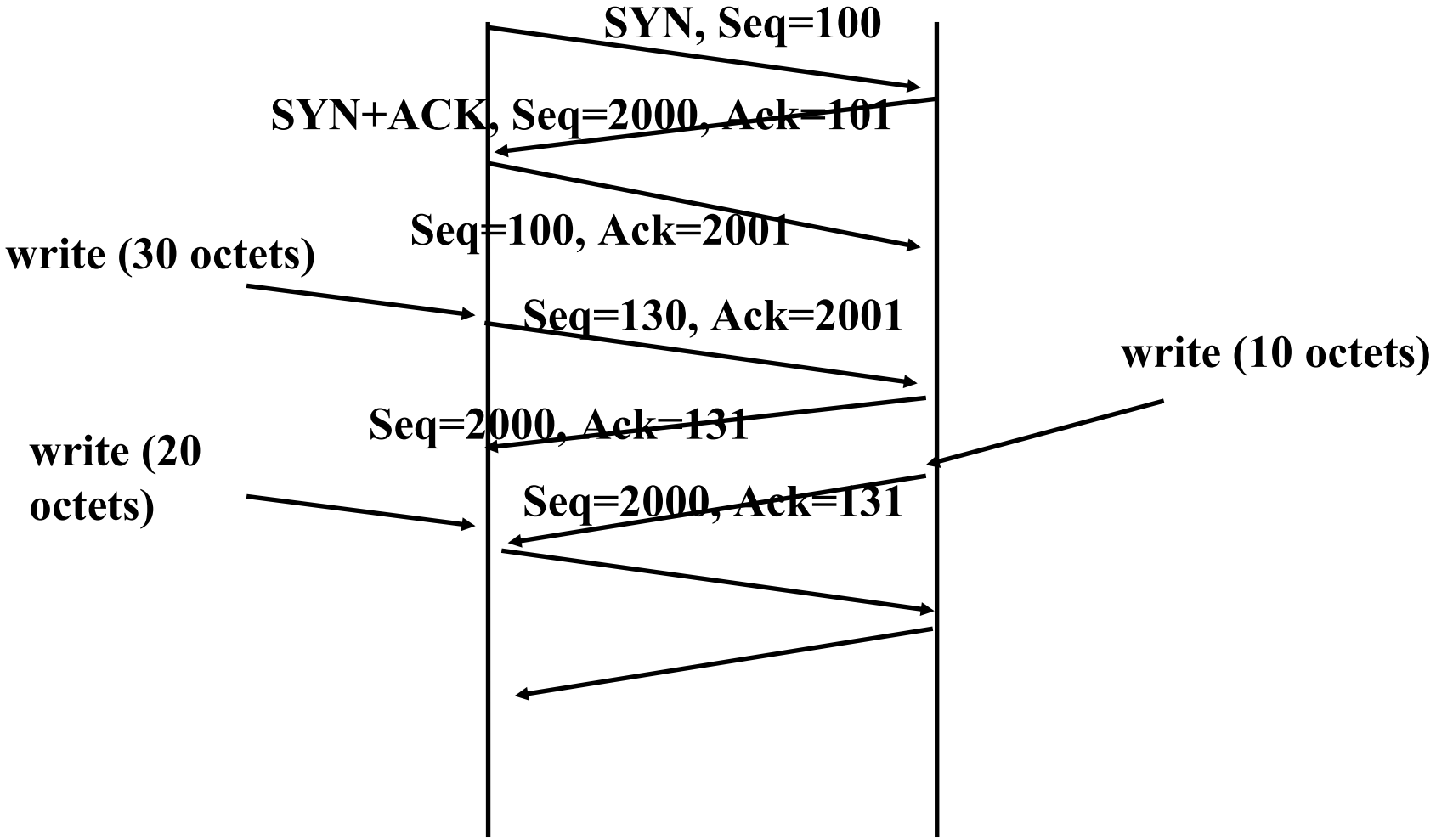
- **Exercice : Donnez les numéros de séquence et d'acquittement des paquets dans l'échange TCP suivant:**



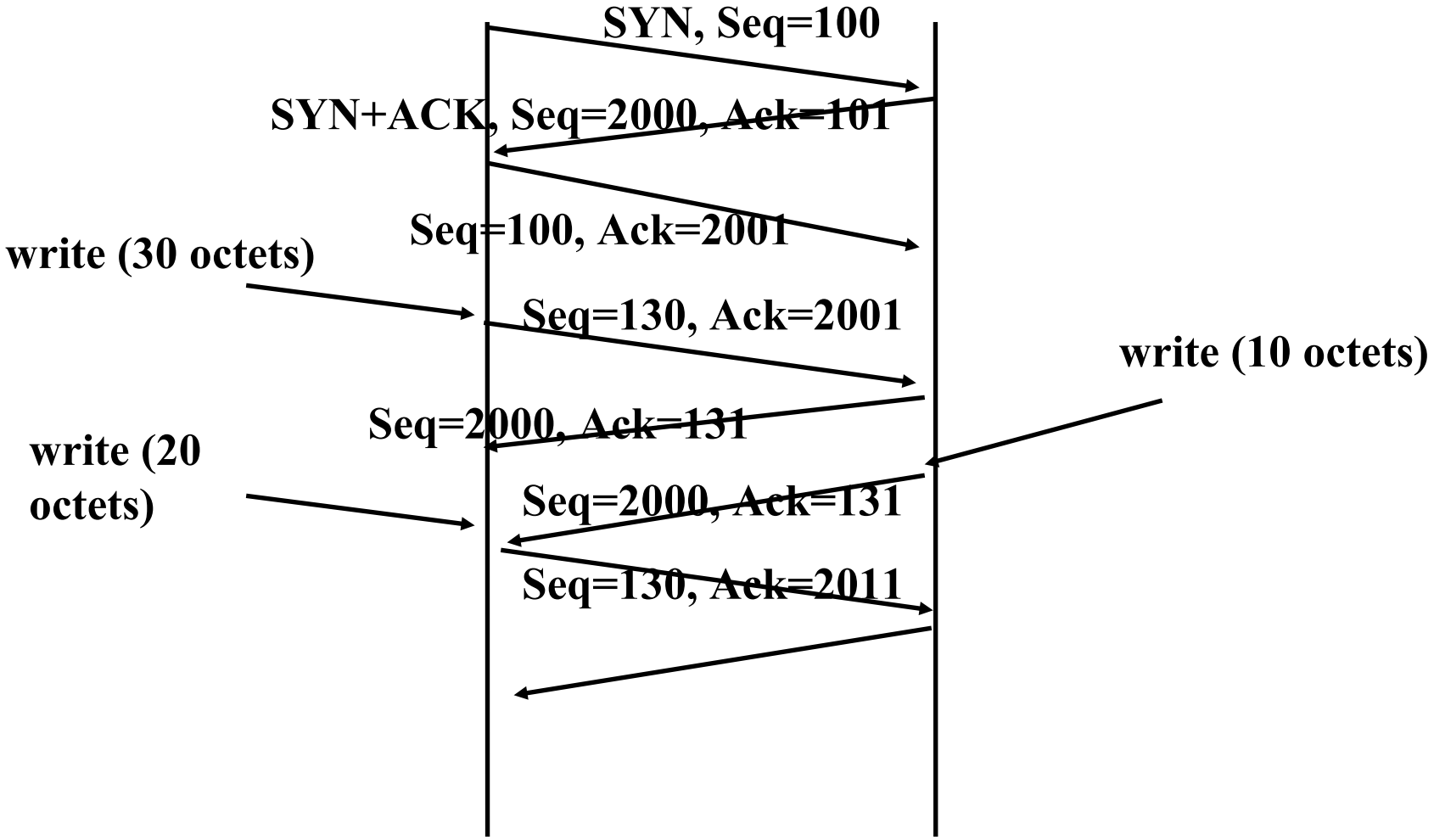
- **Exercice : Donnez les numéros de séquence et d'acquittement des paquets dans l'échange TCP suivant:**



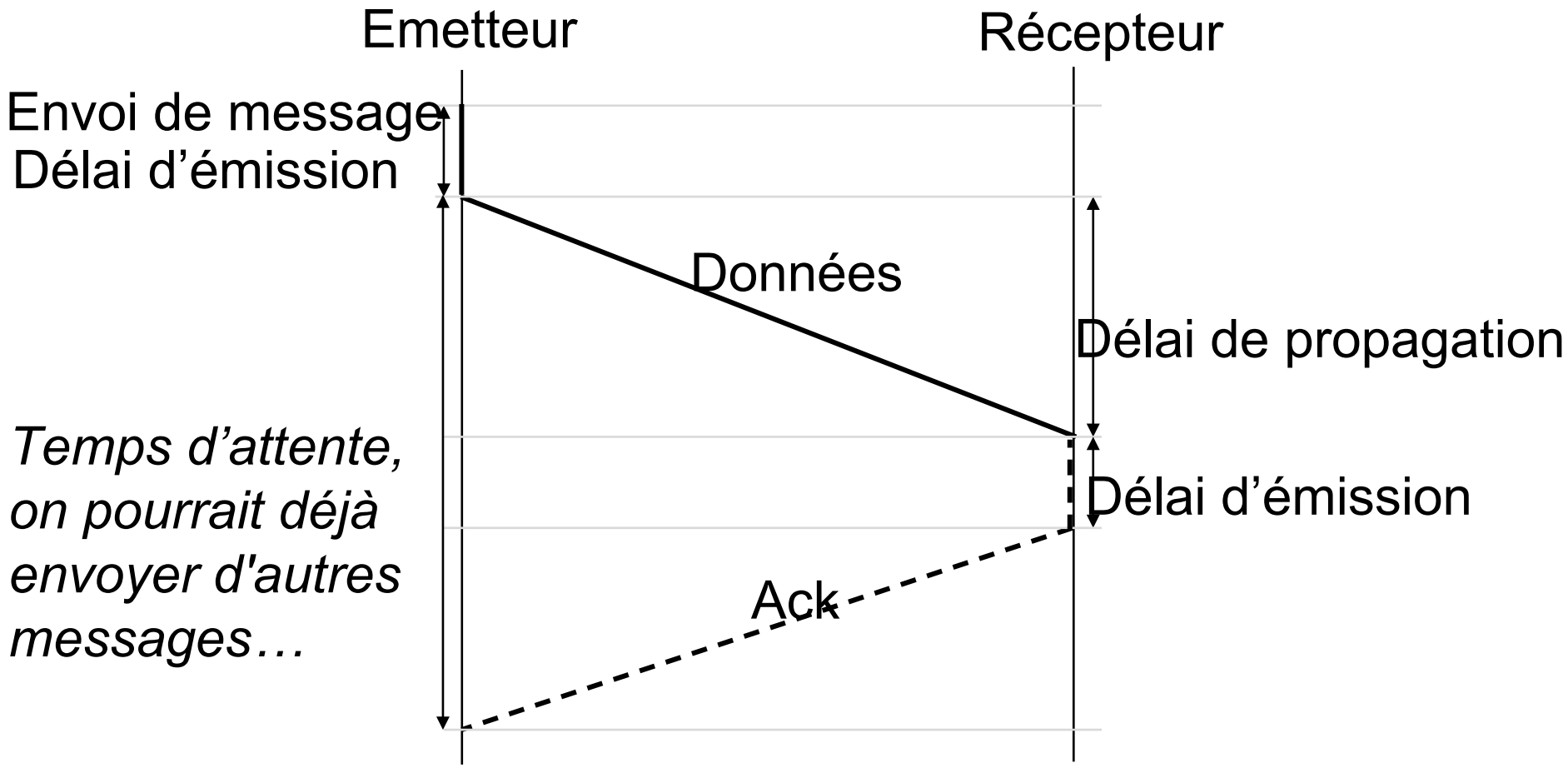
- **Exercice : Donnez les numéros de séquence et d'acquittement des paquets dans l'échange TCP suivant:**



- **Exercice : Donnez les numéros de séquence et d'acquittement des paquets dans l'échange TCP suivant:**



Performance du protocole fiable “envoyer et attendre ack”



Recommencer ...

Protocole TCP : optimisation de performance par contrôle du flux

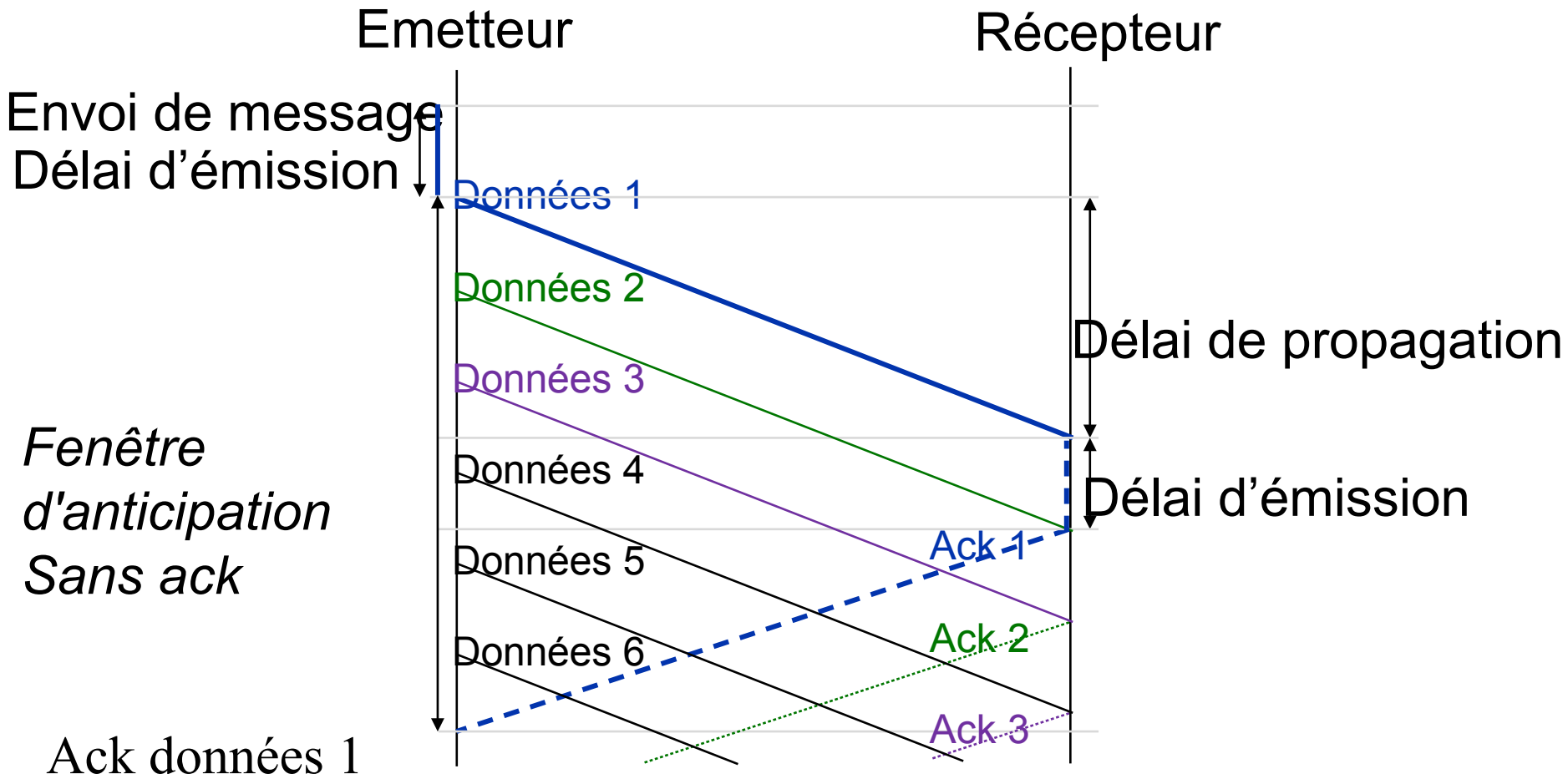
Fiabilisation via acquittement

- **Beaucoup d'attentes**

Fenêtre d'anticipation glissante gérée par le récepteur

- **Plusieurs émissions en transit sans acquittement**
- **Acquittement de plusieurs envois à la fois (cumulatif)**
- **Ré-émissions sélectives des envois perdus**
- **Buffers tampons nécessaires à l'émission comme à la réception**
- **Programme de gestion sur le recepteur**
- **Programme d'optimisation dynamique du réglage taille de fenêtre / buffers**

Protocole à fenêtre d'anticipation glissante



Protocole à fenêtre d'anticipation

- Le recepteur donne la taille de son buffer libre
- L'émetteur envoie le nombre d'octets correspondant sans ack avant de se bloquer
- Un acquittement est toujours envoyé pour chaque message reçu

La fenêtre d'émissions anticipées:→ un certain nombre de trames émises sans acquittement sont en transit

Le buffer de réception doit être supérieur ou égal à la taille de la fenêtre d'émission pour un fonctionnement optimal.

- On peut perdre des paquets → donc l'émetteur mémorise les paquets expédiées jusqu'à ce qu'ils aient été acquittées.
- Le temps de transmition est variable → taille de la fenêtre, timers de réémision, ack cumulatifs, ...

TCP : stratégie de maximisation du flux global et de contrôle de congestion par les emetteurs

- **Réglage de la taille de la fenètre d'émission anticipee**

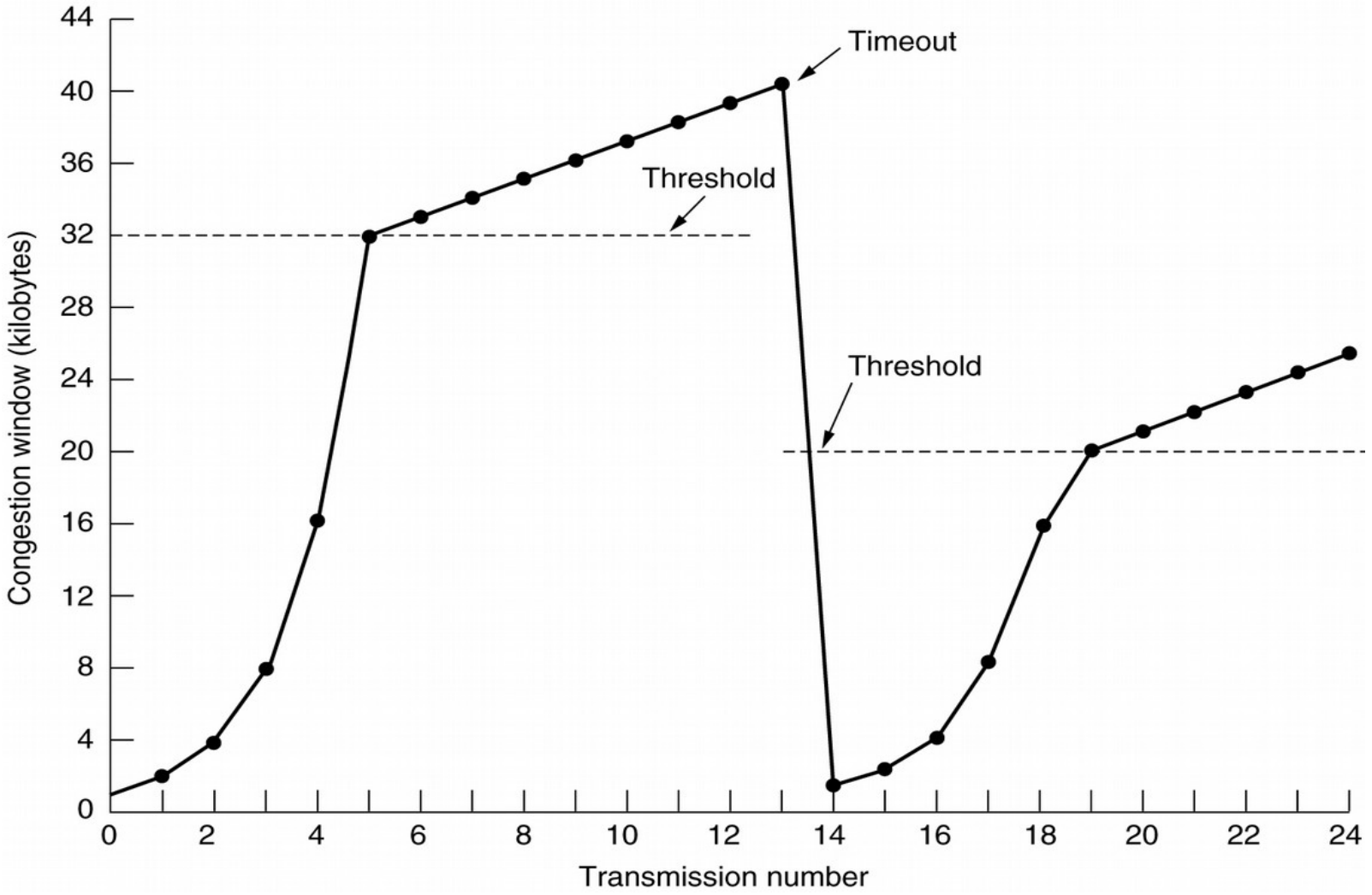
  F=min(congestion_size, ack_window_size)

  congestion_size gérée par la source

  ack_window_size géré par la destination
- **Congestion_size : débute très petit, progresse vite (x2) jusqu'à taille_équilibre et tente d'y rester le plus longtemps possible en progressant alors très doucement.**
- **Si problème retour au départ avec**

  taille_équilibre = congestion_size/2

TCP Congestion Control



An example of the Internet congestion algorithm.

Fonctionnalités des protocoles Transport d'Internet: client/serveur

. Protocoles client/serveur:

- Le serveur se met en attente de demandes

- Le client initie le dialogue par une demande

•Le Problème est de pouvoir faire communiquer 2 processus sur des hôtes différents

•Interface "sockets"

→ 1 service : 1 adresse + 1 numéro de port fixé = un point d'entrée prédéfini sur une machine donnée

=> dans le systeme un processus peut être associé à un port

- Côté serveur: ports réservés pour applications standards

cf. Fichier /etc/services; Exemple: HTTP port 80 en TCP

- Côté client: ports alloués dynamiquement

•Multiplexage des applications:

- Adresses Internet source et destination, numéros de ports source et destination

- Grâce aux entêtes réseau (IP) et transport (TCP ou UDP)

Notions de Qualité de service (QoS) : Paramètres selon Norme OSI

- **Des besoins de qualité variables**
  - **Ex1: transfert de fichier : taux d'erreur nul, débit non primordial, temps de transit non plus**
  - **Ex2: téléphone: taux d'erreur peut être non nul, débit et temps de transit minimum (<0,25s)**
- **Temps d'établissement d'une connexion**
  - **Durée s'écoulant entre l'instant où la demande est émise par l'utilisateur et celui à partir duquel où la confirmation est reçue.**
- **Probabilité d'échec d'établissement**
  - **Probabilité que la connexion ne puisse s'établir dans un délai maximum défini.**
- **Débit de la connexion**
  - **Mesure du nombre d'octets utiles pouvant être transférés en une seconde sur la liaison. Le débit est mesuré séparément dans les deux sens.**
- **Temps de transit**
  - **Temps s'écoulant entre l'instant où un message est émis et l'instant où il est reçu par l'entité de transport du destinataire.**
- **Taux d'erreur résiduel**
  - **Quotient entre le nombre de messages perdus ou mal transmis et le nombre total de messages émis au cours d'une période de temps.**

Qualité de service : Paramètres (suite)

- **Probabilité d'incident de transfert**
  - **Fraction de temps durant laquelle la qualité de service n'a pas été respectée.**
- **Temps de déconnexion**
  - **Durée s'écoulant entre la demande de déconnexion émise par un utilisateur et la déconnexion effective de l'usager distant.**
- **Probabilité d'erreur de déconnexion**
  - **Taux de demandes de déconnexion non exécutées pendant le temps maximum qui était défini.**
- **Protection**
  - **Possibilité de résister aux intrusions d'un tiers sur la connexion.**
- **Priorité**
  - **Permet aux utilisateurs de privilégier certaines connexions.**
- **Résiliation**
  - **Liberté laissée à la couche transport de fermer une connexion suite à un engorgement ou à des problèmes internes.**

Qualité de service : mécanisme

- **À l'ouverture de la connexion deux valeurs de chacun des paramètres sont transmises à l'entité transport :**
  - **La valeur souhaitée**
  - **La valeur minimum acceptée**
- **L'entité de transport locale regarde si elle peut au moins satisfaire les valeurs minimales demandées.**
  - **si oui, elle transmet à l'entité distante les valeurs définissant la QOS qu'il peut satisfaire.**
  - **si non, elle retourne une erreur à la couche application.**
- **L'entité de transport distante opère de la même façon sur les valeurs reçues.**
  - **Si une ou plusieurs valeurs ne peuvent être garanties, la demande est rejetée.**
  - **Si non la demande est acceptée et l'utilisateur reçoit les valeurs accordées.**
- **Les paramètres définissant la QOS ne sont pas normalisés**

QoS dans Internet est limité

- **Application temps-réel : la QoS de TCP est insuffisante**
- **Niveau Réseau, protocoles spécialisés**

– **RSVP: Ressource Reservation Protocol**

- Permet de réserver dans les routeurs des ressources permettant de garantir, un débit, le délai de traversée...

- C'est le destinataire qui est à l'origine des réservations en fonction de la QoS qu'il espère

- Nécessite des échanges de paquets de signalisation

– **Différentiation de service (DiffServ):**

- Définition de classes de services de qualités différentes

- Marquage des paquets et traitement prioritaire dans les routeurs

–**Ces techniques sont peu utilisés à grande échelle car difficiles à mettre en oeuvre**

Exemple de besoin en QoS

- **Niveau Tansport: QoS inutile pour beaucoup d'application "temps-réel"**
  - **Exemple: le téléphone**
  - **Les paquets perdus et réemis arrivent trop tard**
  - **Un taux d'erreur non nul est acceptable**
  - **Utilisation de UDP**

- **Une couche supplémentaire entre UDP et l'application: RTP (Real Time Protocol) normalisé**

→ Numérotation des paquets, estampillage temporel, rapports du récepteur à l'émetteur pour signaler: la QoS courante (délai de transit, taux d 'erreur, débit…)

→ Adaptation par l'application (changement de taux de compression, correction à l 'arrivée, tampon d 'amortissement à l 'arrivée …)