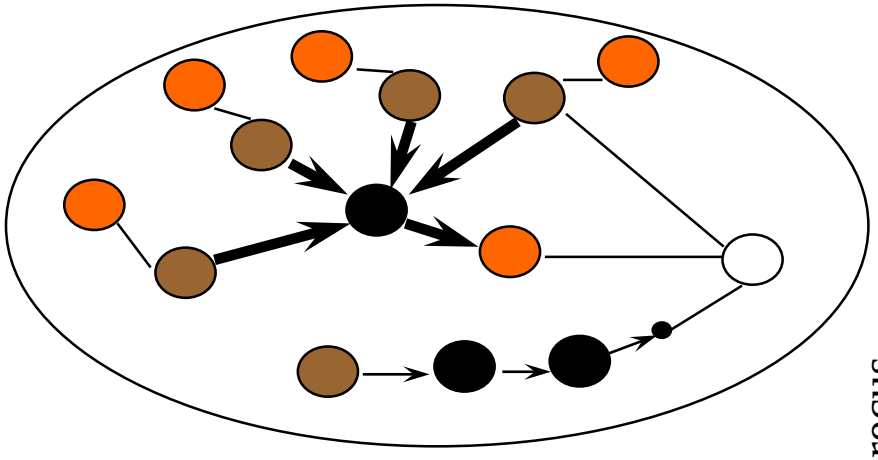
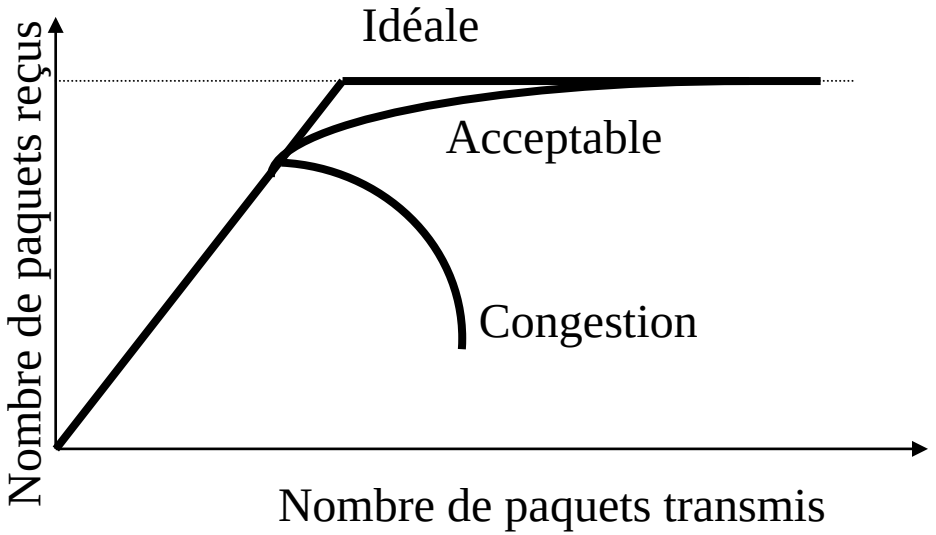
Chapitre 2
La couche réseau

6. Contrôle de congestion

Congestion : définition





- Routeur congestionné : taux de paquets en arrivée est supérieur à celui en sortie (trafic écoulé) avec files d'attentes saturées
- Un routeur congestionné finit par rejeter les paquets en arrivée

Congestion : définition

- Contrôle de congestion = assurer que le sous-réseau est capable de transporter le trafic présent

≠

- Contrôle de flux = assurer le trafic point à point entre un émetteur et un récepteur (= assurer que l'émetteur ne soit pas trop rapide vis à vis du récepteur)
- Le contrôle de flux essaie d'éviter la congestion.
- Malgré les techniques préventives la congestion peut apparaître.
- Contrôle de flux et de congestion

  → Couches : Liaison de données , Réseau et Transport

Contrôle de congestion

- Deux Approches
  - En boucle ouverte : concevoir un système qui évite, au mieux, les problèmes de congestion (prévention) :
    - Allocation de ressources aux différents flux (si mode connecté)
    - Contrôle de l'entrée du réseau (fenêtre de bout en bout ou en local, ex: CV sur X25)
    - Canalisation du trafic (Traffic Shapping): technique du seau percé (leaky bucket), seau à jeton
  - En boucle fermée : prévoir des mécanismes pour la détection de la congestion, la rétroaction et l'ajustement du trafic (guérison) :
    - Utilisation de ressources supplémentaires
    - Contrôle d'admission
    - Paquet d'engorgement
    - Délestage «load shedding»

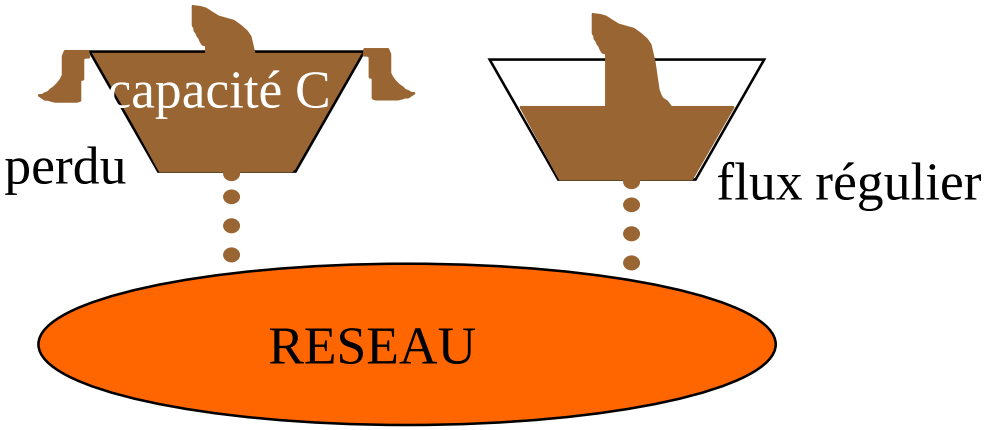
Contrôle de congestion : en boucle ouverte

- Les techniques de canalisation du trafic (« traffic shaping ») ont pour but de réguler la vitesse d'écoulement des données.
- Elles sont utilisées dans les réseaux ATM.
- Elles peuvent être adaptées aux sous-réseaux fonctionnant en mode non-connecté.
- Utilisent l'algorithme du seau percé «Leaky bucket algorithm »

Contrôle de congestion : en boucle ouverte

« Leaky bucket algorithm »

- Chaque ordinateur est relié au réseau via une interface d'accès.
- Cette interface simule le seau percé à l'aide d'une file d'attente de taille fixe.
- À chaque top d'horloge, un paquet de la file d'attente est envoyé sur le réseau, sauf si celle-ci est vide.
- Tout paquet sortant est placé dans la file d'attente, sauf si celle-ci est pleine(dans tel cas il est alors détruit).
- Ce mécanisme transforme un flux irrégulier de paquets provenant d'un processus interne à un ordinateur source en un flux régulier de paquets sur le réseau.

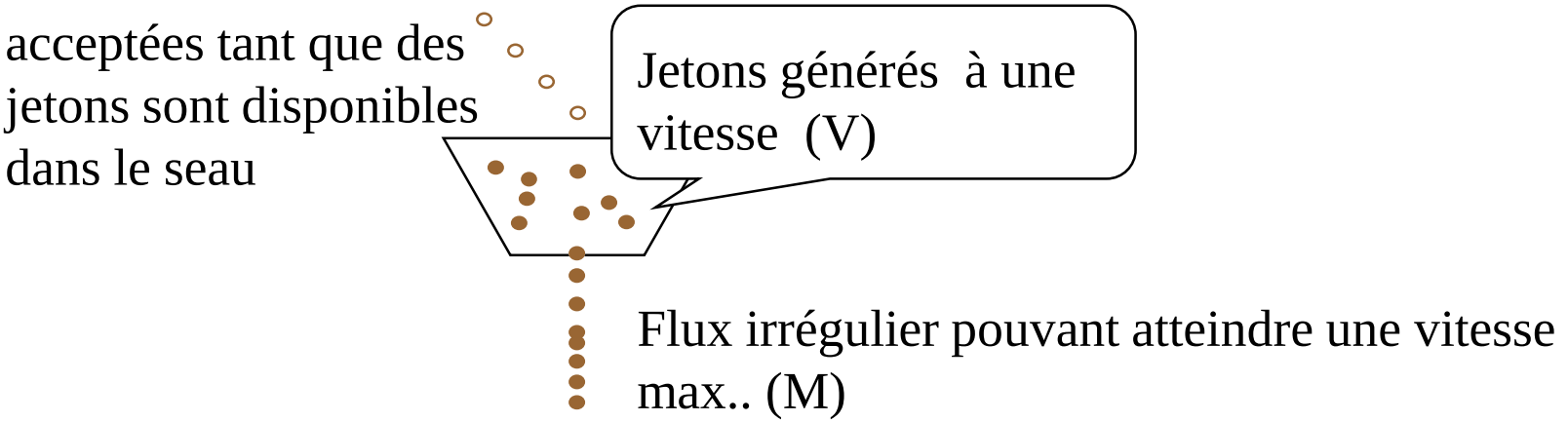


-seau : file d'attente de taille fixe
-eau : paquet ou multiple d'octets
-vitesse : paquet/sec ou octet/sec.

Contrôle de congestion : en boucle ouverte

Algorithme du seau percé à jeton

- Algorithme plus souple car il permet une augmentation provisoire du trafic
- Un jeton est engendré à chaque top d'une horloge (nombre maximum n jetons).
- Un paquet est transmis s'il reste au moins un jeton, sinon, il est rejeté (et pas détruit).
- Un jeton est détruit à chaque émission de paquet.



Contrôle de congestion : en boucle ouverte

Exemple :

- C : taille du seau (jetons): 250 kB
- ρ : Jetons générés à une vitesse de 2 MB/seconde
- M : Débit maximum en sortie est 25 MB/sec
- Le seau est plein, un trafic de 1 MB arrive
- S = taille maximale d'une rafale (burst length) en secondes

C + ρ x S = M x S (trafic max en entrée = trafic max en sortie)

→ S = C / (M - ρ) = 250 kB / (25 MB/sec - 2 MB/sec)
= 11 ms

→ Le seau peut écouler un trafic à 25 MB/sec durant 11 ms

Contrôle de congestion en boucle fermée : Paquet d'engorgement (chock packet)

- réduit le trafic (50%) vers la destination

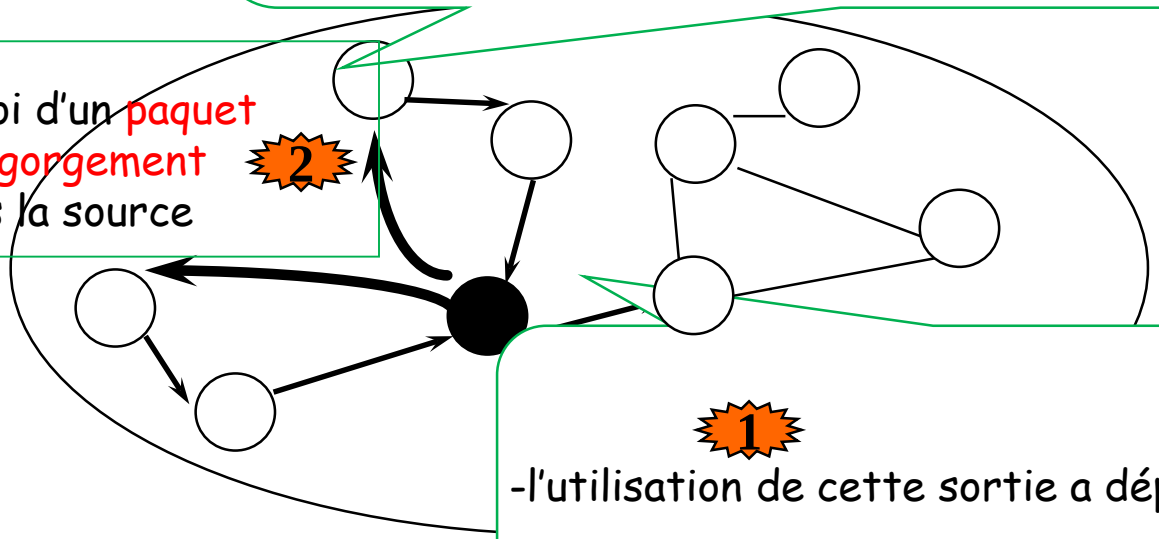
3

- ignore les paquets d'engorgement pendant une certaine durée

- si au bout d'un certain délai, aucun paquet d'engorgement n'est reçu, le trafic est augmenté par petit incrément

Envoi d'un paquet d'engorgement vers la source

2



Rétro-contrôle

1

-l'utilisation de cette sortie a dépassé un seuil

- à la sortie, le paquet est marqué

Technique des paquets d'engorgement

Contrôle de congestion : en boucle fermée

Technique des paquets d'engorgement

- Critique 1 : si la source ne collabore pas et ne réduit pas son trafic, elle peut ainsi profiter de la situation.
  Solution - algorithme du temps équitable : pour chaque sortie, les paquets sont envoyés de façon cyclique selon la source "Fair Queueing"
- Critique 2 : la taille des paquets est variable.
  Solution - algorithme du temps équitable pondéré : appliquer le même algorithme par octet "Weighted Fair Queueing"
- Critique 3 : lorsque le débit est important (exp :155Mb/s) ainsi que le temps de transit (exp : 30ms), une grande quantité de données aura été injectée dans le réseau (4,5 Mb) avant que le paquet d'engorgement n'arrive à la source.
  Solution - contrôle de l'engorgement en pas à pas - en remontant pas à pas vers la source et sur chaque routeur intermédiaire, le paquet d'engorgement a pour effet de réduire la vitesse vers la destination. Chaque routeur a ainsi besoin de réserver des mémoires tampon supplémentaires pour le trafic vers la destination. Le noeud de congestion est ainsi rapidement soulagé (« Hop by Hop chock packet").

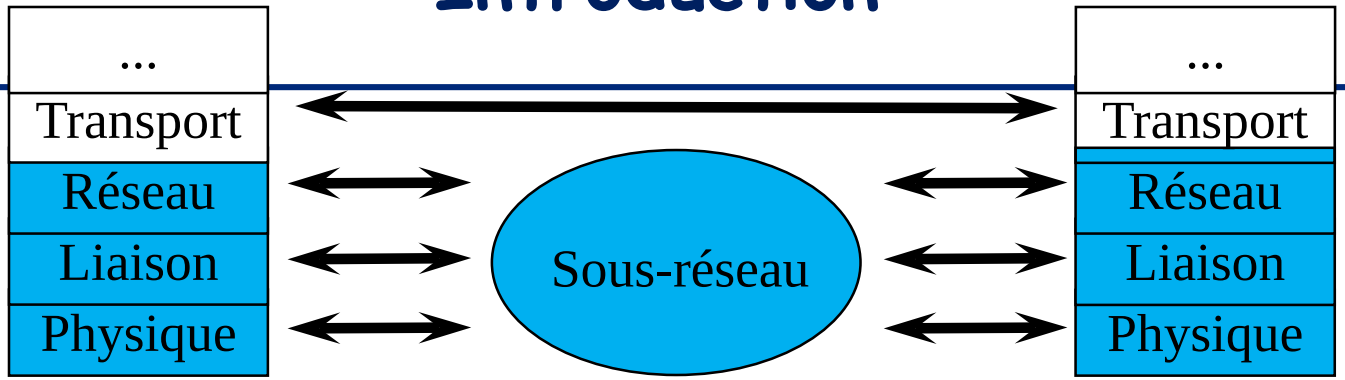
Contrôle de congestion : en boucle fermée

Le délestage «load shedding»

- Pour les méthodes de contrôle de congestion décrites précédemment, le risque de congestion n'est pas forcément écarté, le délestage consiste alors à détruire des paquets.
- Choix des paquets à détruire
  - Parfois il vaut mieux détruire les paquets les plus récents. Pour certaines applications (temps réel) c'est l'inverse.
  - Perdre une ligne de pixels d'une image est moins grave qu'un texte associé ...
- Solutions
  - les applications marquent les paquets suivant une certaine classification de priorité
  - autoriser le dépassement des limites négociées en marquant par une faible priorité le trafic en excès
  - détruire tous les fragments / cellules d'un même paquet ...

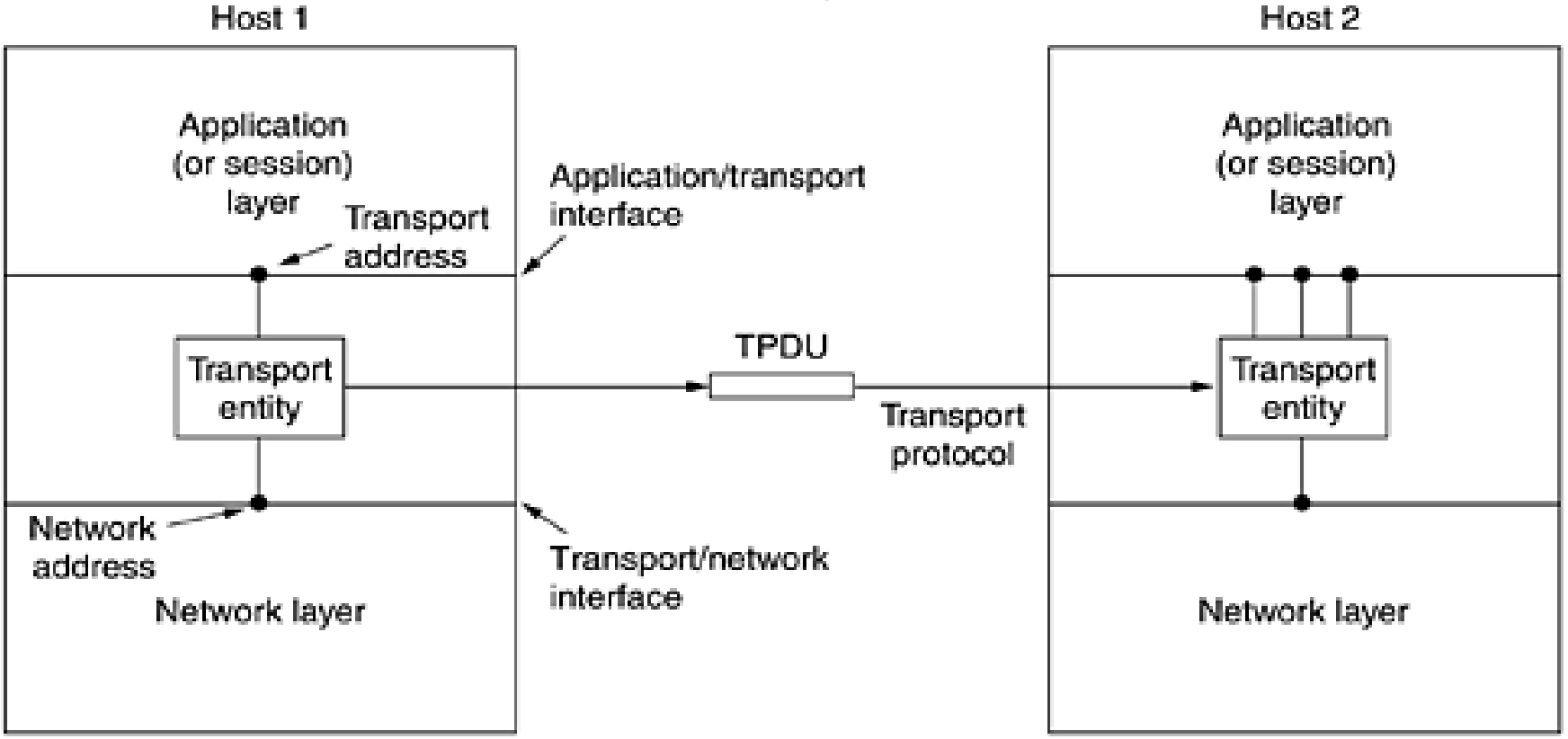
Chapitre 3
La couche transport

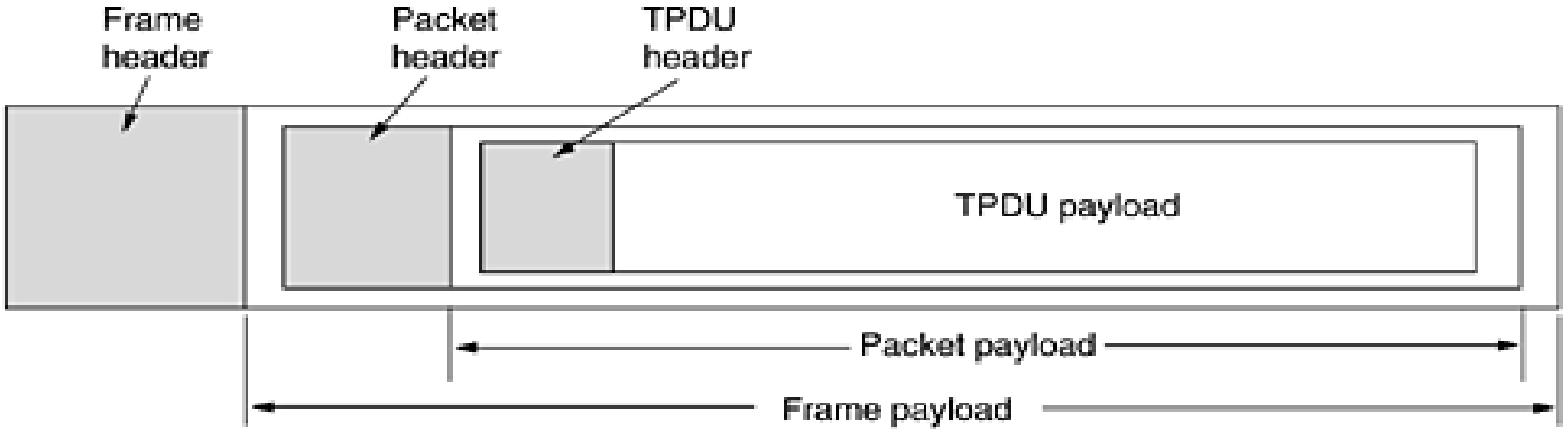
Introduction



- Gère les communications de bout en bout entre processus
- Fournir à l'utilisateur (processus de la couche application) un service de transport fiable et économique (qu'IP n'est pas en mesure d'offrir)
  - Permet de dialoguer indépendamment de la nature des sous-réseaux traversés
  - Optimise l'utilisation des services de réseau afin d'assurer à moindre coût un service approprié
  - Assure la transparence des informations transférées

Introduction





Types de services

- Mode « orienté connexion »
  - création d'une connexion préalable, maintenue pendant toute la communication et explicitement terminée,
  - négociation des paramètres de qualité de service à l'établissement,
  - communication bidirectionnelle et ordonnée,
  - contrôle de flux et de congestion

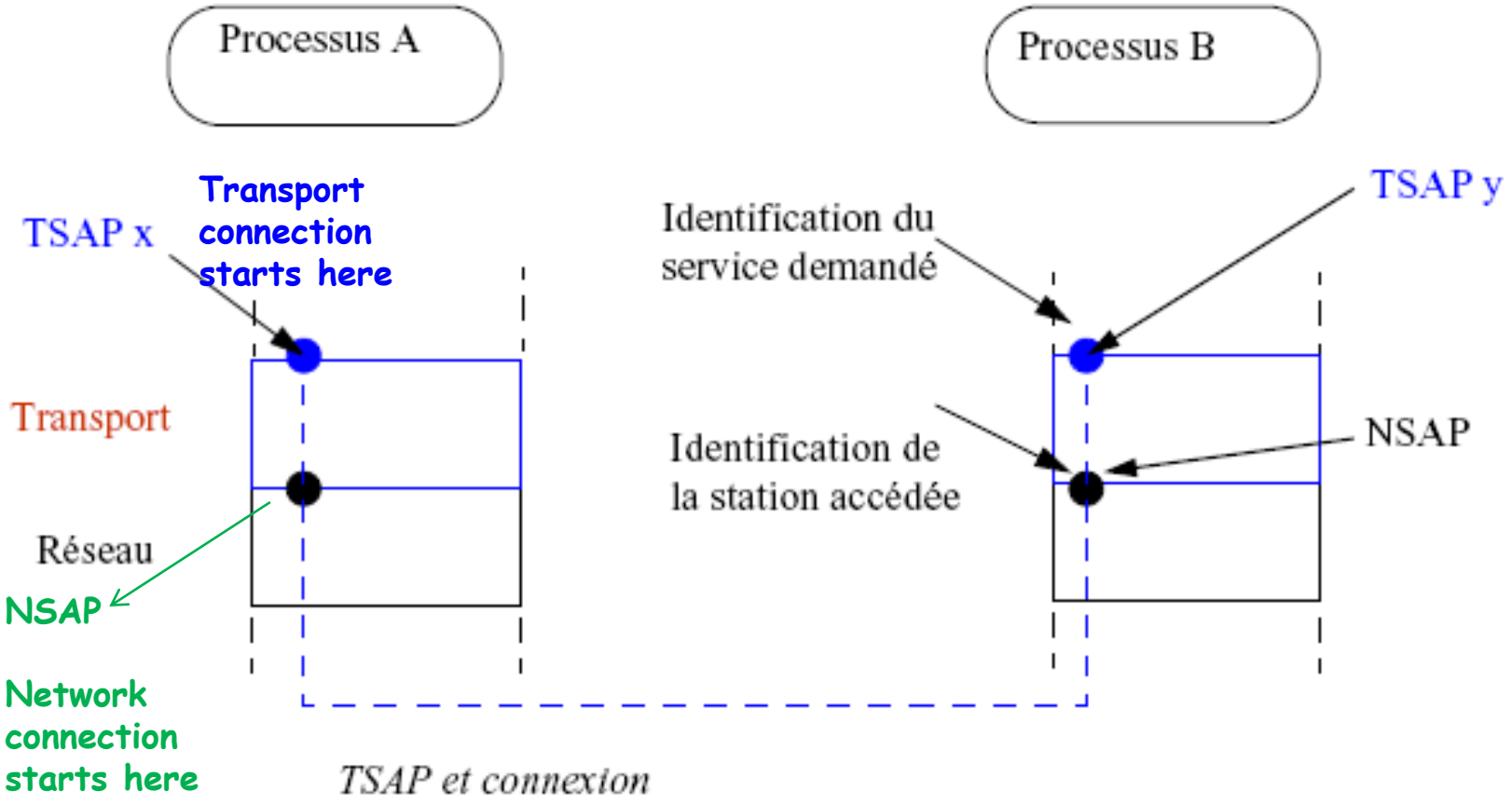
- Mode « sans connexion »
  - envoi et réception de datagrames sans garanties
  - service non fiable

Fonctionnalités des protocoles de transport

- Principales fonctions réalisées
  - sélection d'une connexion réseau
  - la mise en relation des adresses transport et réseau
  - décision de mettre en place un multiplexage / éclatement
  - la segmentation ou la concaténation
  - le contrôle de flux
  - le contrôle d'erreur

Eléments de protocole

✓ **Adressage** : TSAP (Transport Service Access Point)

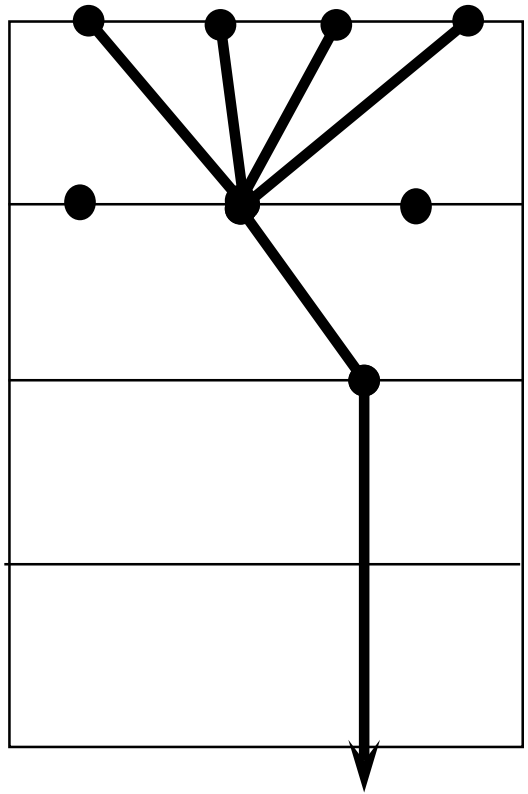


TSAP et connexion

Eléments de protocole

- **Multiplexage**
  - Multiplexage en amont: cas de sous utilisation d'une connexion réseau, taille de la fenêtre d'anticipation réduite par rapport à la capacité de la liaison
  - Multiplexage en aval (Eclatement): cas contraire

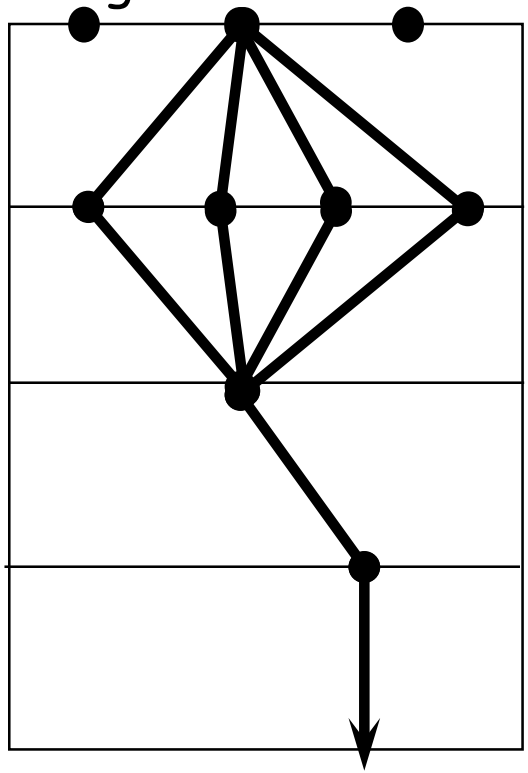
❖ Multiplexage en amont



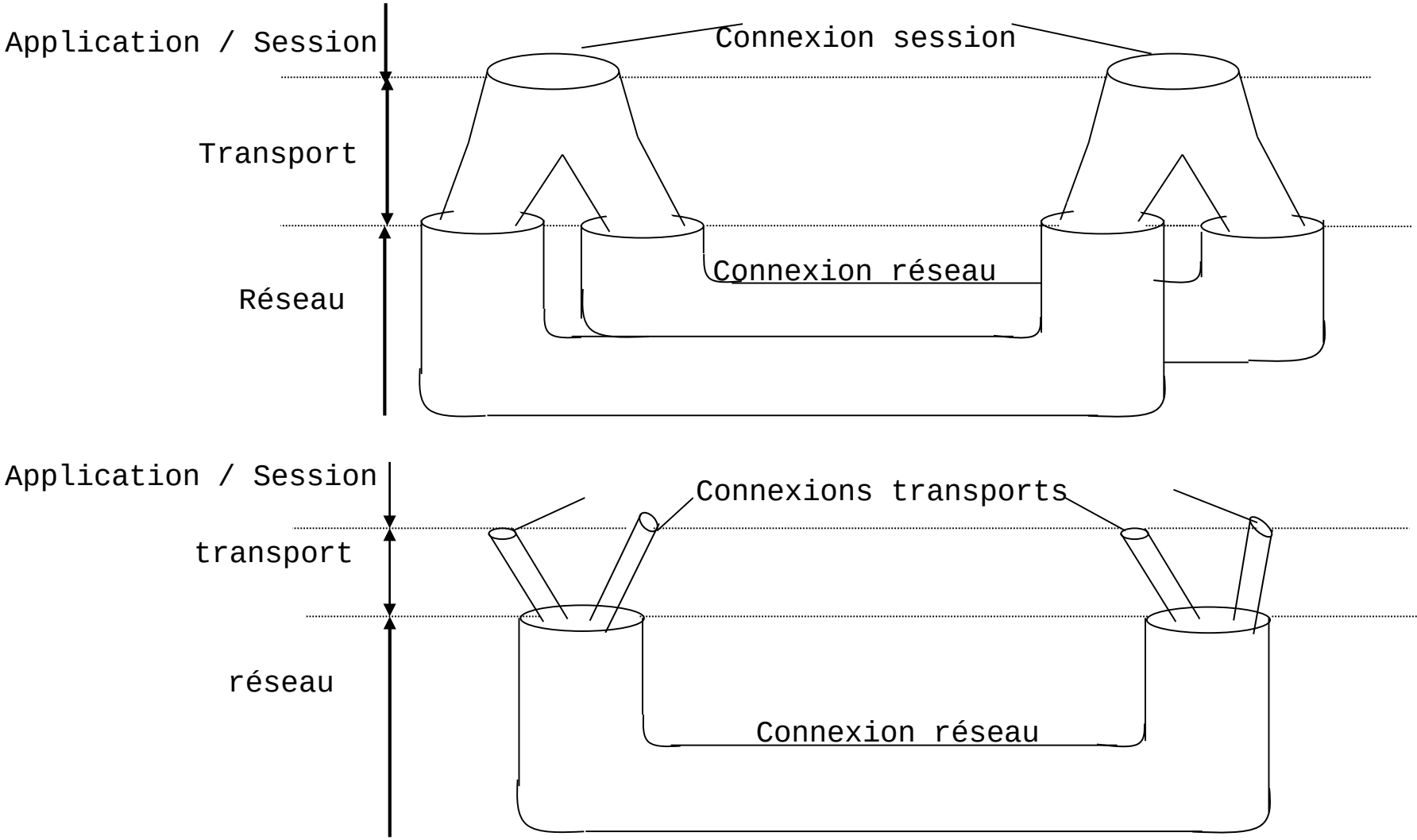
Transport

Réseau

Multiplexage en aval

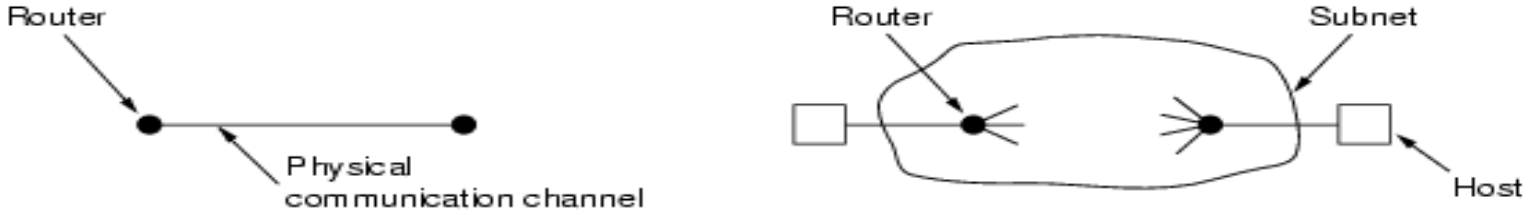


Eléments de protocole



Liaison de données vs Transport

	Liaison	**Transport**
Débit et temps de transit	Stable	Variable
séquencement, contrôle de flux	Oui Simple	Oui Plus élaboré
Besoin d'adressage	Pas nécessaire	Oui
Multiplexage	Pas nécessaire	Oui



(a) Environment of the data link layer.
(b) Environment of the transport layer.

Les protocoles de transport

- Normes relatives au modèle OSI : TP (Transport Protocol)
  - mode connecté : ISO 8072 8073,~ ITU (International Telecommunication Union) X.214 X.224
  - mode non connecté : ISO 7498 / Additif 1, ISO 8602

- Standard TCP/IP
  - mode connecté : TCP (RFC 793)
  - mode non connecté : UDP (RFC 768)

TP (Transport Protocol)

- ISO 8073 / ITU (CCITT) X214
- L'objectif est de combler l'écart existant entre les services offerts par la couche réseau et les services à offrir.
- On distingue 3 types de réseau
  - A : taux d'erreurs résiduelles et d'incidents acceptable
  - B : taux d'erreurs résiduelles acceptable mais d'incidents inacceptable
  - C : taux d'erreurs résiduelles et d'incidents inacceptable
- Pour simplifier le choix des fonctions, à mettre en œuvre, l'ISO a défini 5 classes de protocoles chacune adaptée à un type de réseau.

TP (ISO): Classes transport vs Types réseaux

	Network				
	A	B	A	B	C
	Class				
Protocol element	0	1	2	3	4
Connection estabilishment	x	x	x	x	x
Connection refusal	x	x	x	x	x
Assignment to network connection	x	x	x	x	x
Splitting long messages into TPDUs	x	x	x	x	x
Association of TPDUs with connection	x	x	x	x	x
TPDU transfer	x	x	x	x	x
Normal release	x	x	x	x	x
Treatment of protocol errors	x	x	x	x	x
Concatenation of TPDUs to the user		x	x	x	x
Error release	x		x		
TPDU numbering		x	o	x	x
Expedited data transfer		o	o	x	x
Transport Layer flow contol			o	x	x
Resyncronization after a RESET		x		x	x
Retention of TPDUs until ack		x		x	x
Reassignament after network disconnect		x		x	x
Frozen reference		x		x	x
Multiplexing			x	x	x
Use of multiple network connections					x
Retrasmission upon timeout					x
Resequencing of TPDUs					x
Inactivity timer					x
Transport layer checksum					o

x = present o = optional (blank) = absent

TP (Transport Protocol) – ISO 8073 / ITU (CCITT) X214

- Les Classes de protocole Transport
  - La classe 0 (classe de base)
    - Le protocole de classe 0 est simple. Il est prévu pour fonctionner au-dessus des services réseau de niveau A.
  - La classe 1 (classe de base avec reprise sur erreur)
    - Cette classe convient à un service réseau de niveau B (service fiable avec resynchronisation).
  - La classe 2 (classe avec multiplexage)
    - Cette classe est généralement utilisée sur un service Réseau de catégorie A.
  - La classe 3 (classe avec reprise sur erreur et multiplexage)
    - C'est l'union des classes 1 et 2 Mais le contrôle d'erreur est obligatoire
    - Cette classe est adaptée au service Réseau de catégorie B
  - La classe 4 (classe avec détection et reprise sur erreur)
    - Cette classe est prévue pour les services Réseau de niveau C (service insuffisant).

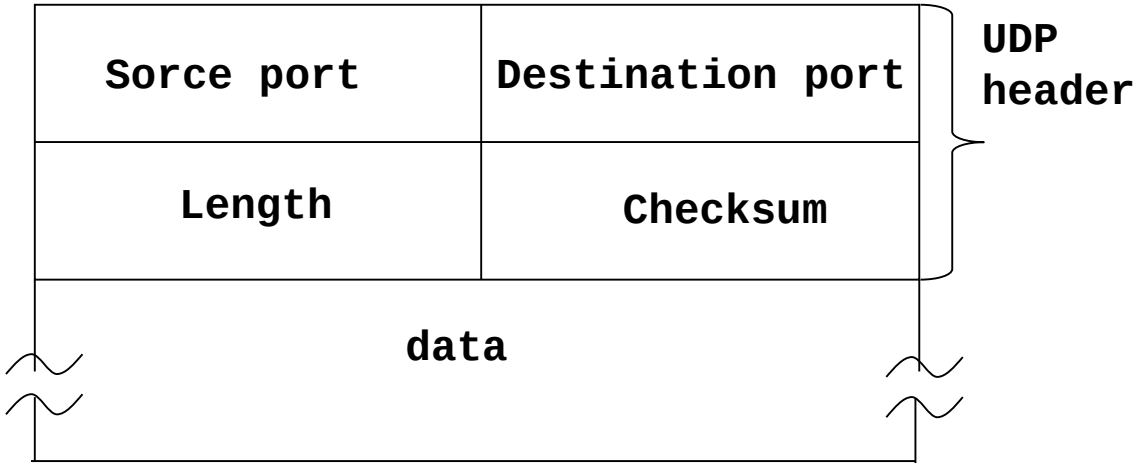
Chapitre 3
La couche transport (Suite)
Les protocoles Internet

Les protocoles de transport d'Internet

- TCP «Transmission Control Protocol» RFC 793 1122 1323
- UDP «User Datagram Protocol», RFC 768
- Service TCP
  - orienté connexion, point à point, bidirectionnel, fiable, conserve le séquencement
  - Entre TSAPs
  - TSAP: Port
    - représenté sur 16 bits;
    - Ports<256 sont prédéfinis; exemple : Telnet : 23, smtp : 25, RFC 1060, 1700
  - une connexion TCP est identifiée par le quadruplet, adresse IP et port en local et adresse IP et port distants
  - flot d'octets
    - l'entité TCP décide d'elle même du découpage du flot en segments
- Service UDP
  - datagramme, UDP ne rajoute à IP que l'information de service (ports)

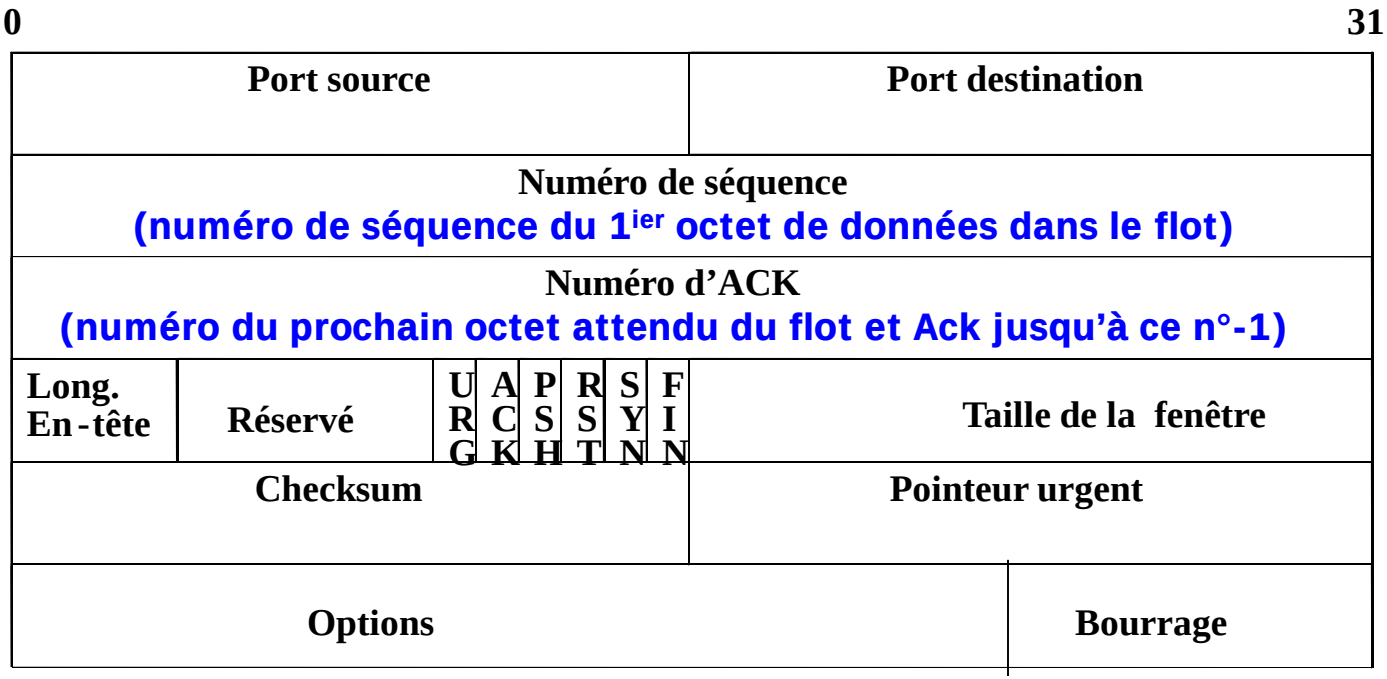
Le protocole UDP

- « User Datagram Protocol »
  - « Connectionless » ; service « end-to-end »
  - Sans contrôle de flux
  - Pas d'acquittement (ack)
  - Adressage à travers des ports
  - Détection des erreurs au niveau de l'entête (Checksum).



Le protocole TCP : Format de l'entête d'un segment

- « Transmission Control Protocol »
  - numéro de séquence par octet
  - fenêtre d'anticipation ajustable par le destinataire
  - temporisation en attente d'un ACK (contrôle de réception et retransmission)
  - Format de l'en-tête :



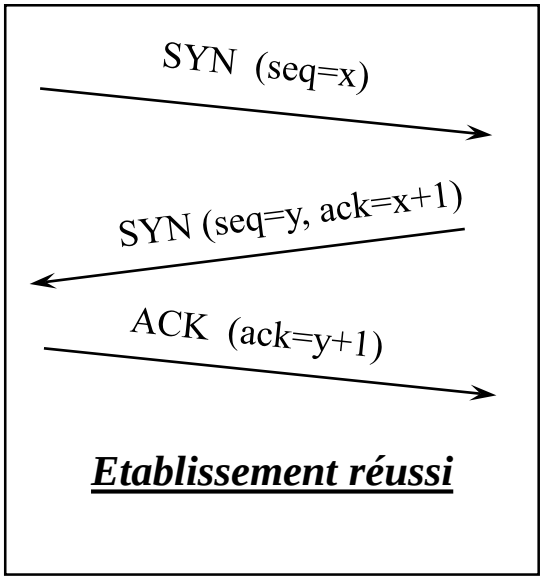
Entête d'un segment TCP (suite)

- ✓ **6 drapeaux d'un bit chacun**
  - URG est positionné à 1 si le pointeur d'urgence est en cours d'utilisation (décalage à partir du n° de séquence courant indiquant où l'on peut trouver les données urgentes)→Evite d'interrompre les messages.
  - ACK est positionné à 1 pour valider le n° d'Ack. S'il est à 0; le champ n° d'Ack est ignoré.
  - PSH (PuSH: données poussées) est positionné à 1 pour demander au destinataire de remettre les données à l'application concernée dès l'arrivée
  - RST est positionné à 1 pour réinitialiser une connexion ou refuser une tentative de connexion.
  - SYN sert à établir les connexions
    - SYN=1 et ACK=0 →Demande de connexion et le champ ack en mode superposition (piggyback) n'est pas utilisé
    - SYN=1 et ACK=1 →Demande de connexion acceptée
  - FIN est positionné à 1 pour indiquer que l'émetteur n'a plus de données à transmettre (demander la libération de la connexion). Il peut continuer à recevoir des données.

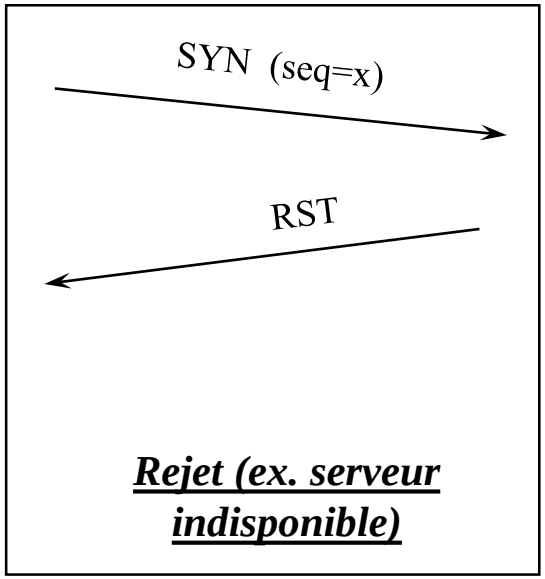
Les segments SYN et FIN ont leurs propres n° de séquence→ordre de traitement

Le protocole TCP

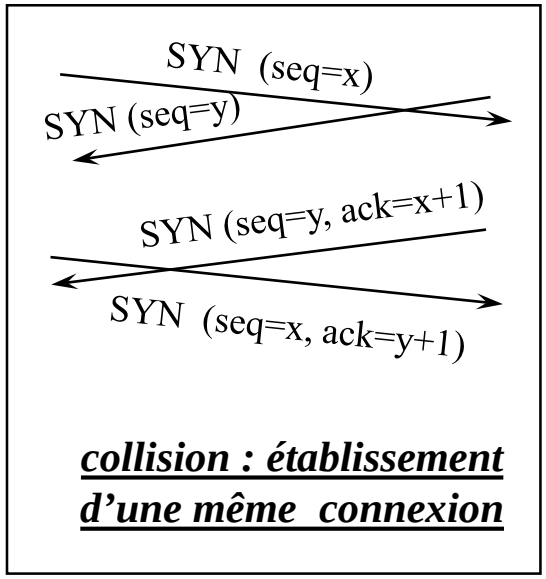
- Établissement d'une connexion (Three Way Hand Shake)



Etablissement réussi



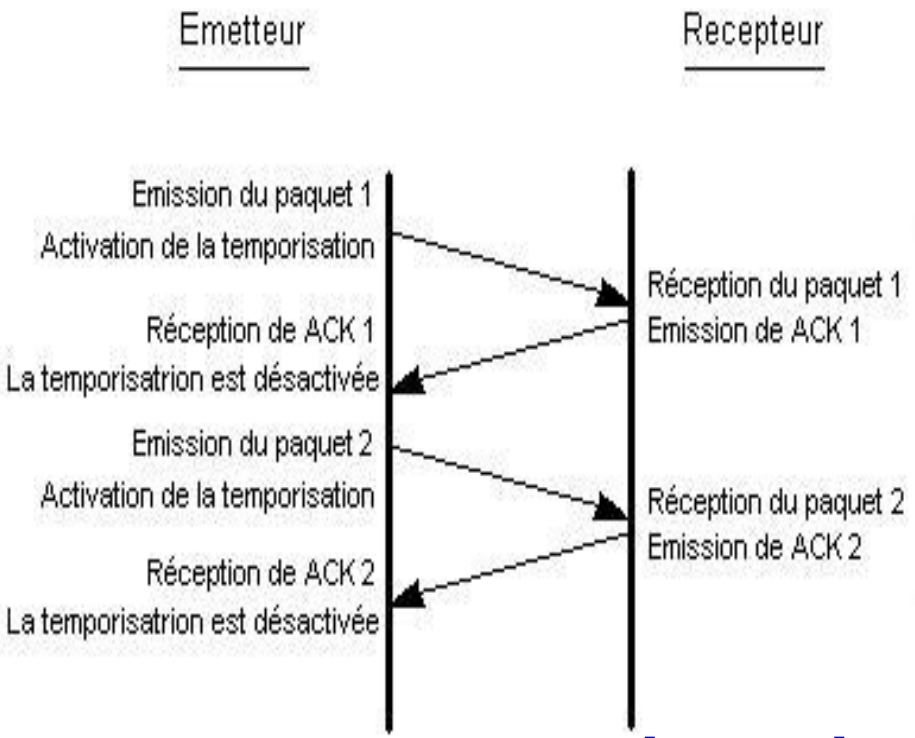
Rejet (ex. serveur indisponible)

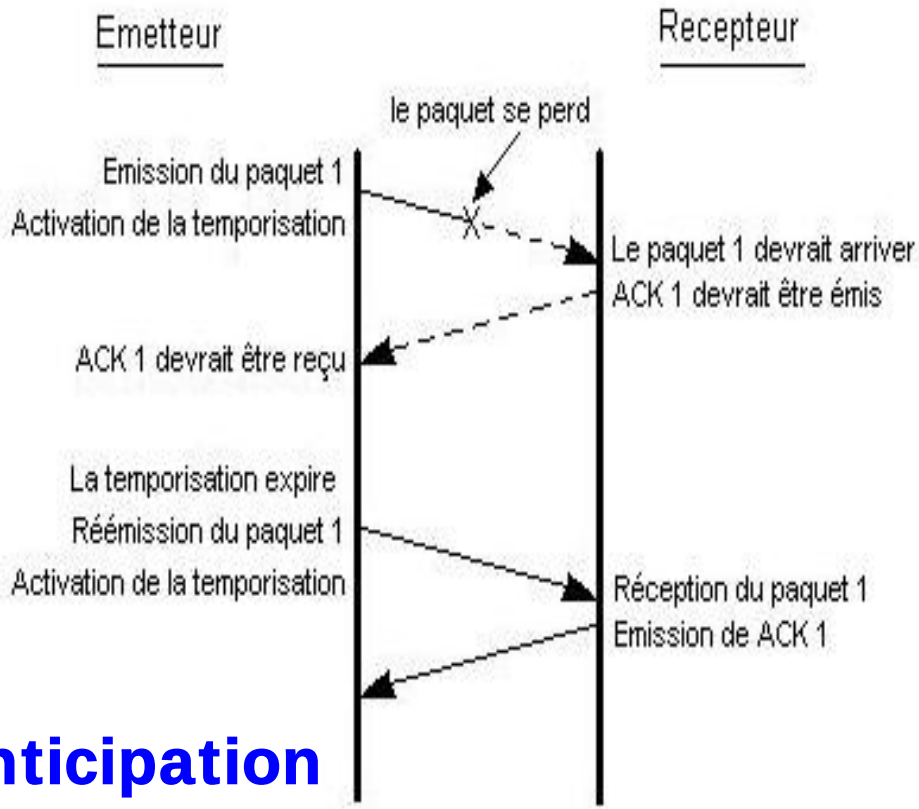


collision : établissement d'une même connexion

TCP

- Assurer la fiabilité : système d'accusé de réception
- Indiquent toujours le numéro du prochain octet attendu par le récepteur
- Les accusés de réception sont dits cumulatifs



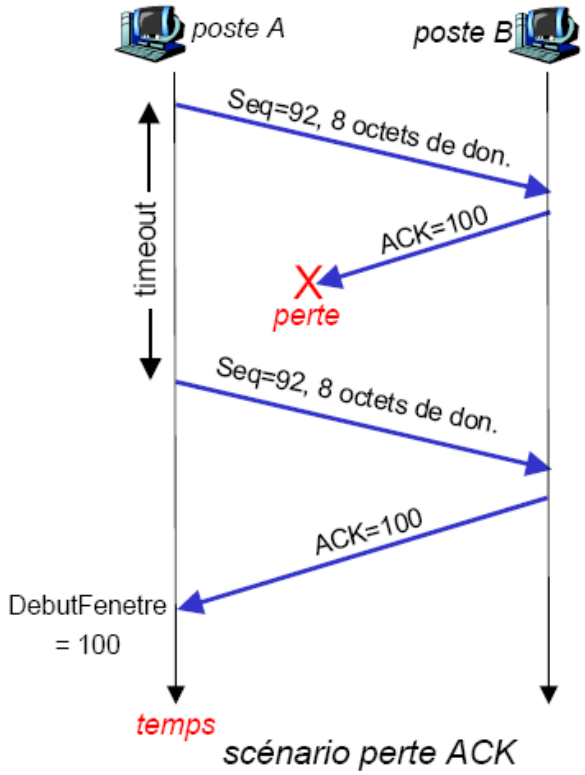


Si pas d'anticipation
→ Perte de temps

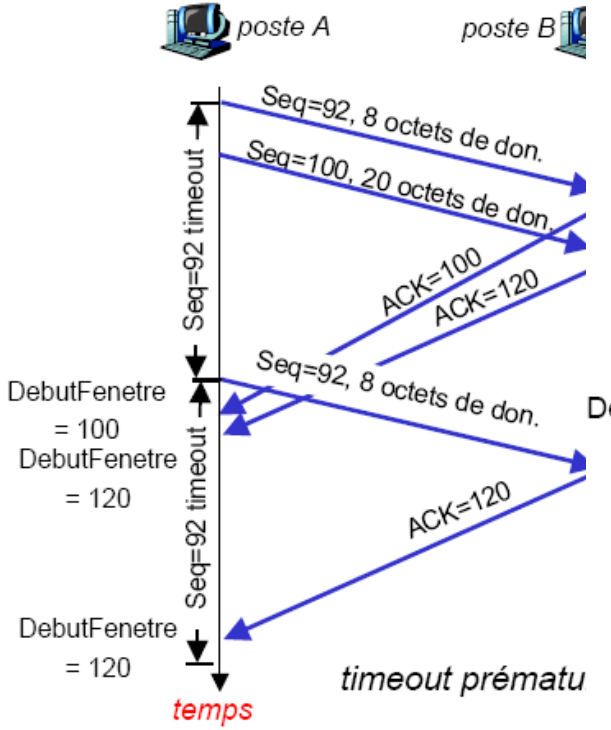
Emission sans pertes de paquets

Emission avec pertes de paquets

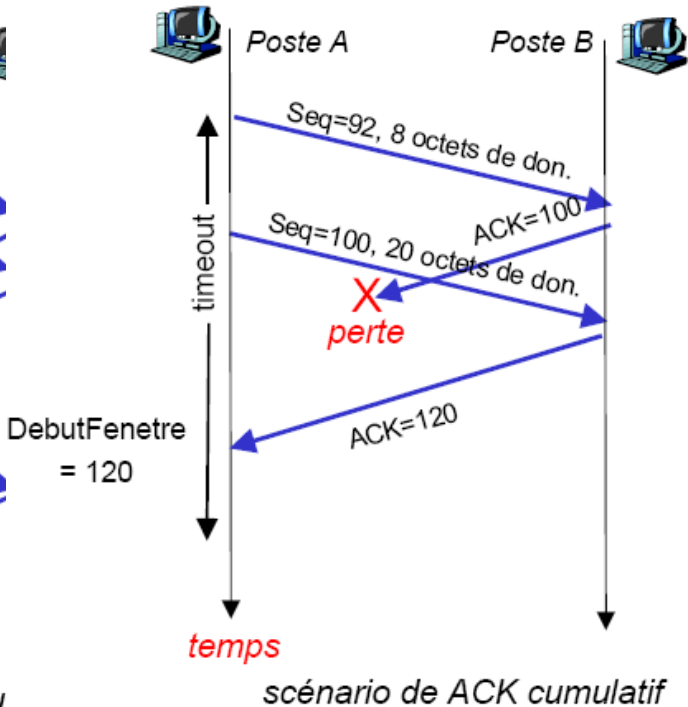
Scénarii de retransmission



scénario perte ACK



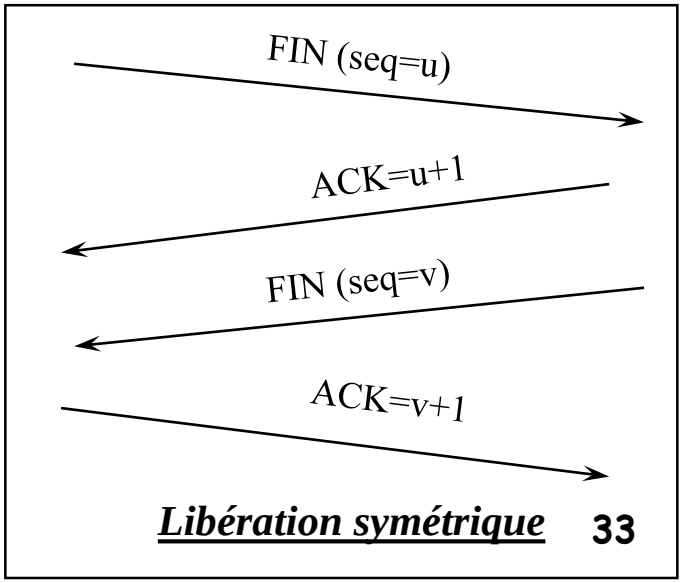
timeout prématu



scénario de ACK cumulatif

Le protocole TCP (suite)

- Libération d'une connexion
  - Symétrique
    - (connexion bidirectionnelle ~ 2 connexions à l'alternat)→ 2 (FIN+ACK)
    - primitives : close, closing, terminate
    - PDU : FIN , ACK
    - une fois sont échangées les différentes PDU nécessaires à la libération, l'entité TCP attend au moins l'équivalent de deux fois la durée de vie maximale d'un segment MSL «Maximum Segment Lifetime» (120 sec.) avant de supprimer l'enregistrement correspondant (gel du port)
    - Si l'ACK du FIN ne parvient pas, une attente équivalente est observée avant la libération de la connexion.
    - (Rq: à la suite d'une panne, une attente équivalente est également observée avant le redémarrage de la connexions).
  - Asymétrique
    - primitive : abort
    - PDU : rien
    - l'autre extrémité se rend compte lorsqu'elle tente de transmettre des données et que les temporisations expirent1



Libération symétrique

TCP - Services fournis

- Au dessus de TCP
  - Deux modes d'ouverture d'une connexion
    - Passive-open – attendre une requête de connexion, coté serveur (lancement d'une application à l'écoute d'un port et en attente de requêtes clients, les clients ne sont pas forcément connus d'avance)
    - Active-open – envoyer une requête de connexion, coté client

TCP - Services fournis

- Primitives de services de l'entité supérieure à l'entité TCP
  - **Unspecified-Passive-Open** (source-port, timeout, precedence, security)
  - **Full-Passive-Open** (source-port, destination-port, destination addr, timeout, precedence, security): Limitation à certains clients
  - **Active-Open** (source-port, destination-port, destination addr,timeout, precedence, security)
  - **Active-Open-With-Data** (source-port, destination-port, destination addr, timeout, precedence, security, data, data length,push flag, urgent flag)
  - **Send** (local connection name (structure définie par le système d'exploitation, après le succès des primitives précédentes), data, data length, push flag, urgent flag, timeout)
  - **Allocate** (local connection name, data length) /*incremental allocation for receive*/: Allocation de ressource (mémoire) au niveau de la couche transport
  - **Close** (local connection name)
  - **Abort** (local connection name)
  - **Status** (local connection name)

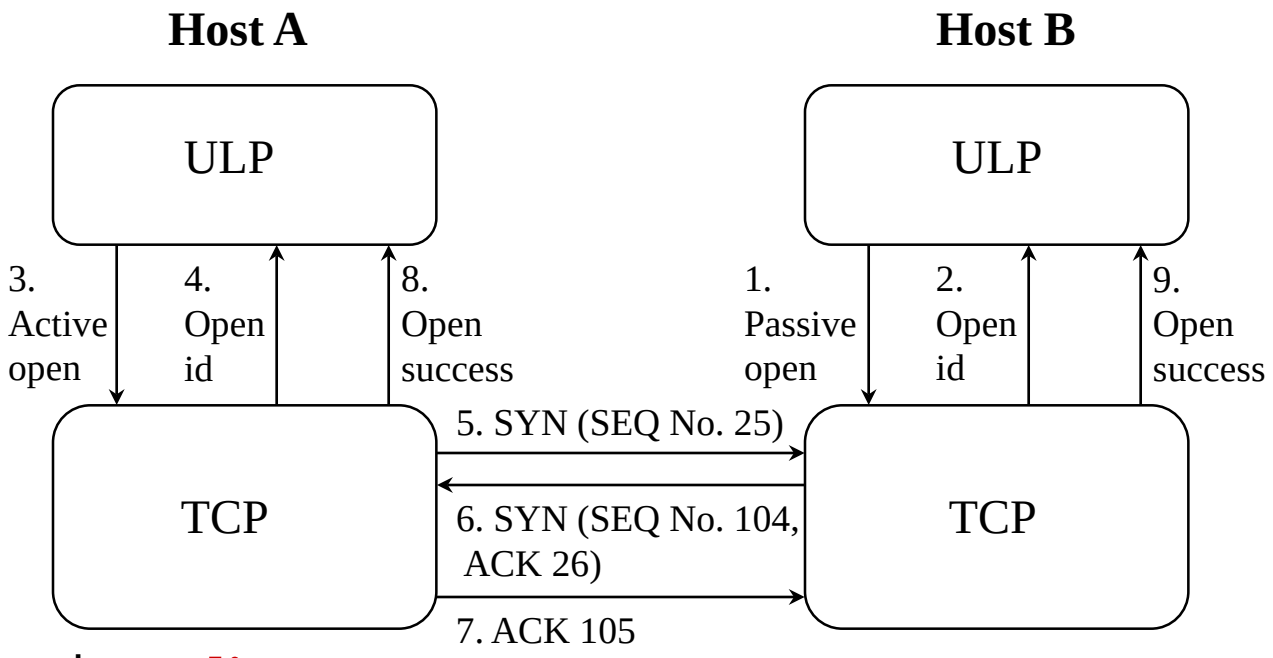
TCP - Services fournis (suite)

- De l'entité transport à l'entité supérieure
  - **Open-Id** (local connection name, source-port, destination-port, destination addr)
  - **Open-Failure** (local connection name)
  - **Open-Success** (local connection name)
  - **Deliver** (local connection name, data, data length,urgent flag)
  - **Closing** (local connection name)
  - **Terminate** (local connection name, description)
  - **Status-Response** (local connection name, source-port, destination-port, destination addr, connection state receive window, send window, amount awaiting ack, amount awaiting receipt, urgent mode, precedence, security, timeout)
  - **Error** (local connection name, error description)

TCP - Services fournis (suite)

Scénario pour l'établissement d'une connexion TCP

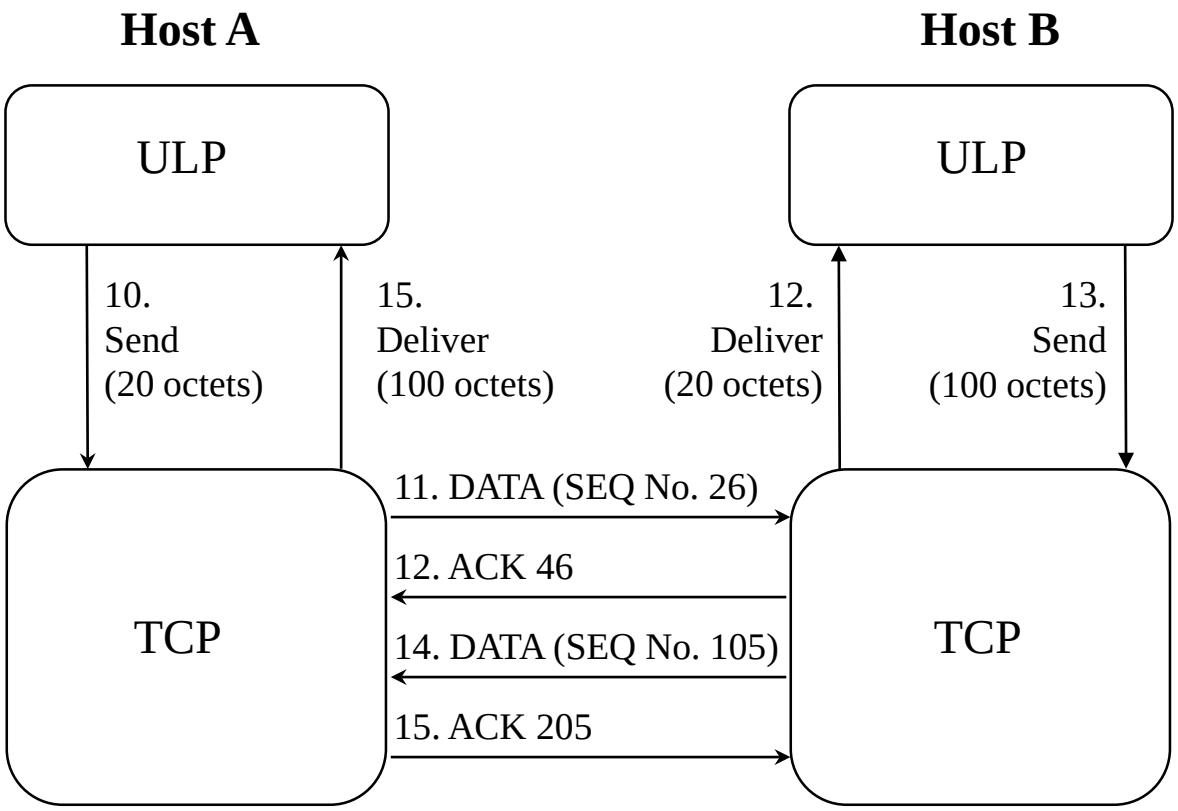
- Il faut au préalable un accord entre les deux machines (Client/Serveur)
  1. Une des machines va effectuer une « ouverture passive » (le serveur)
  2. L'autre machine va effectuer une « ouverture active » (le client)
  3. Le client demande l'établissement d'une connexion avec le serveur



ULP : Upper Layer Protocol

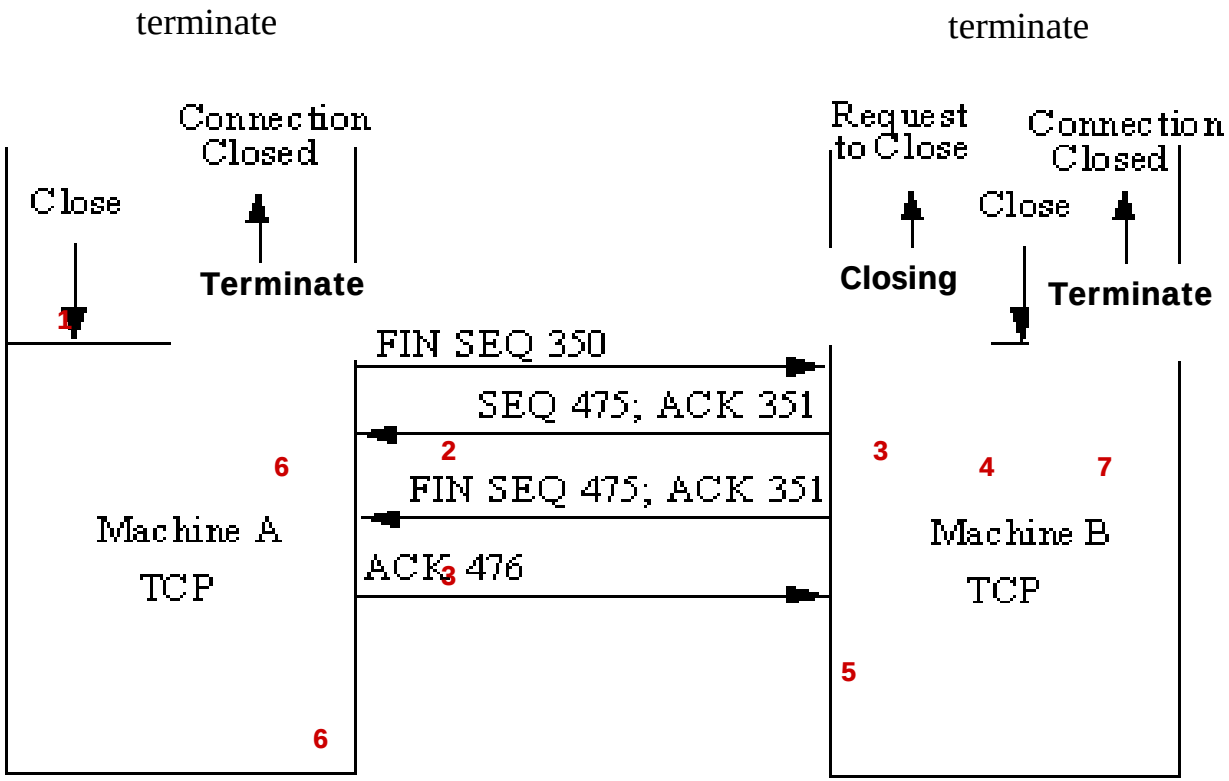
TCP - Services fournis (suite)

- Scénario pour l'échange de données



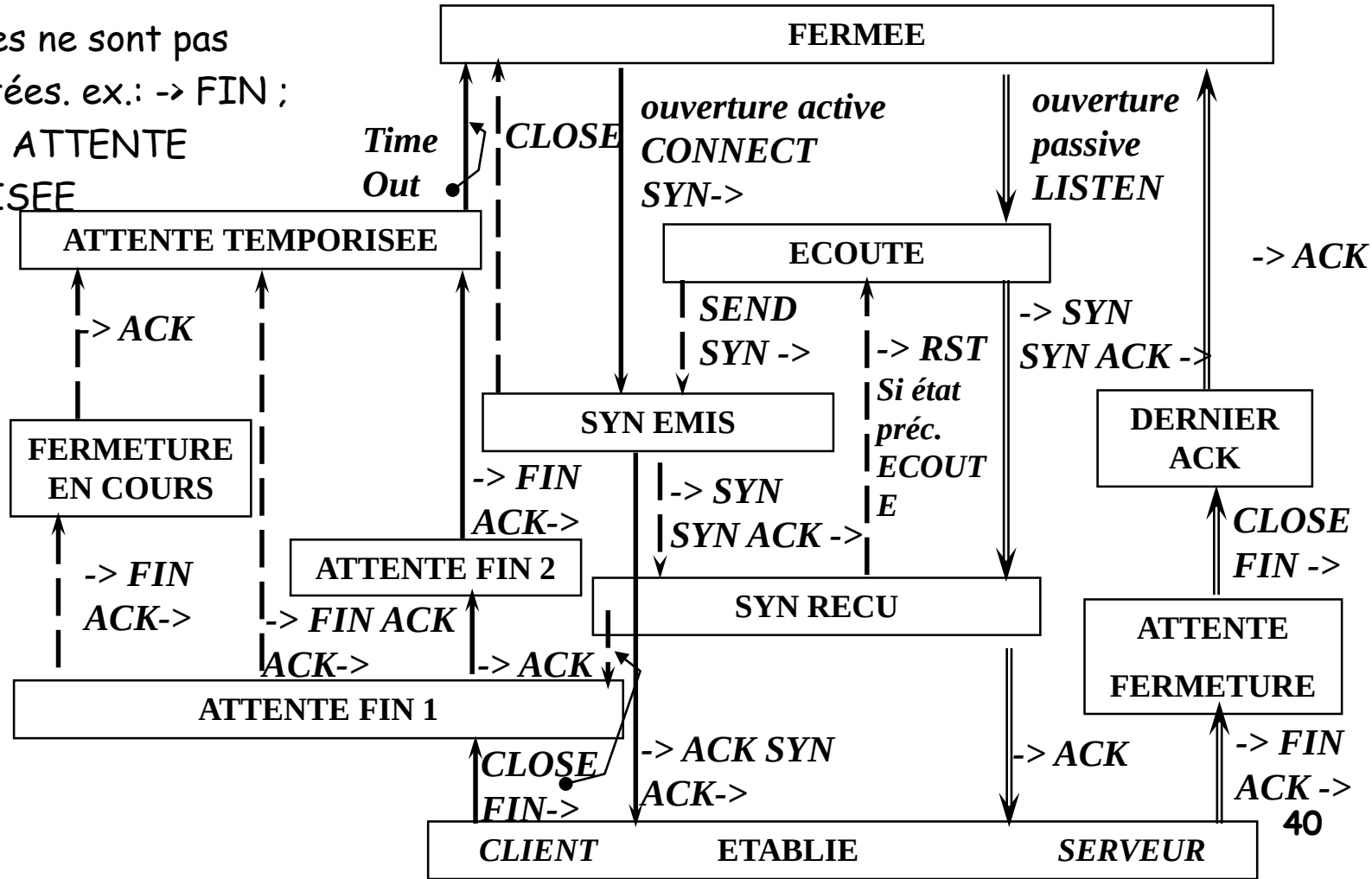
TCP - Services fournis (suite)

- Scénario pour la fermeture symétrique d'une connexion TCP

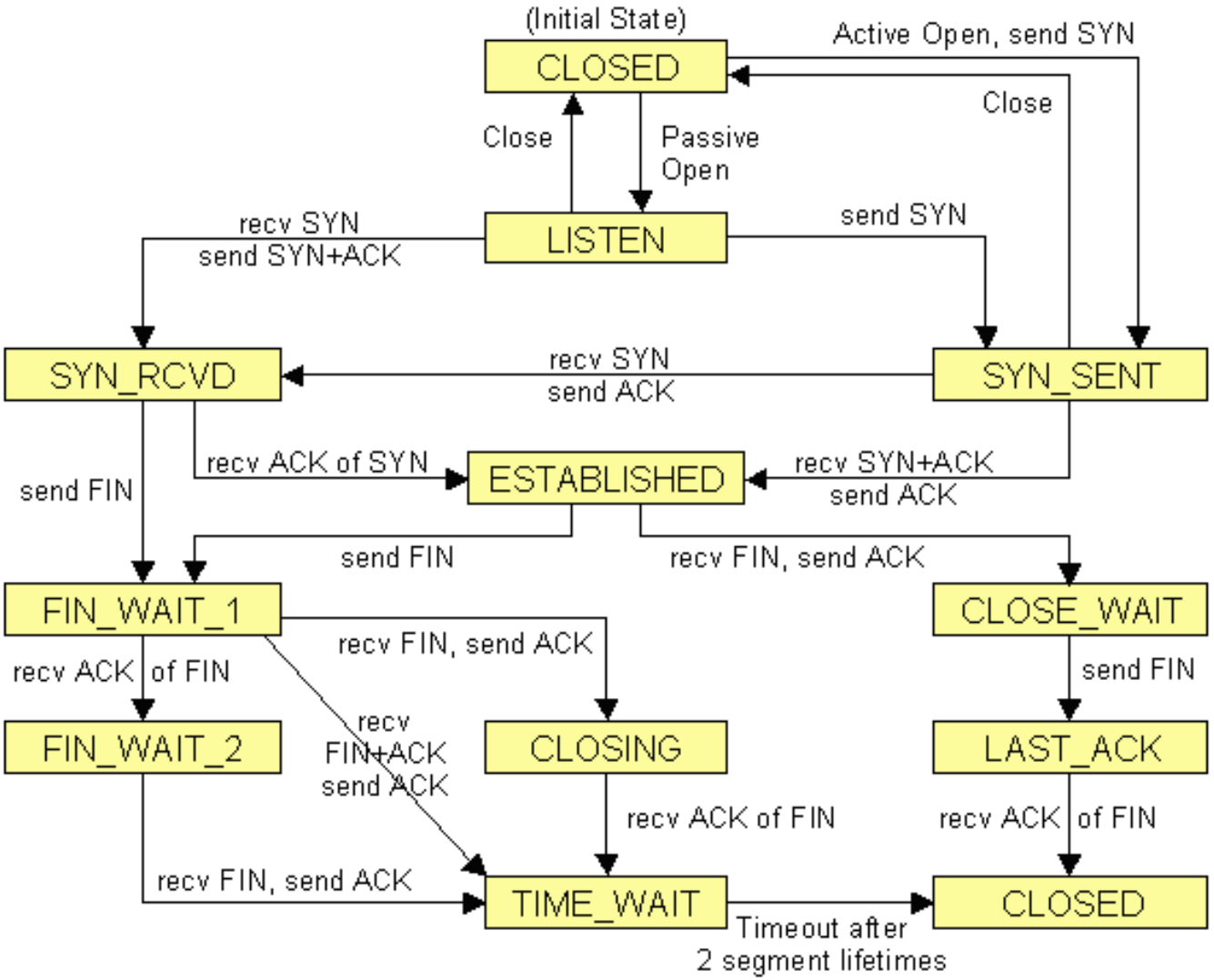


TCP -- Automate à états fini

- Les transitions liées à RST ne sont pas toutes représentées
- ATTENTE FIN 2 est temporisée et évite ainsi de rester bloqué
- Il existe un nombre de retransmissions limite, puis un test au niveau réseau, puis une attente
- Les boucles ne sont pas représentées. ex.: -> FIN ; ACK-> sur ATTENTE TEMPORISEE



TCP -- Automate à états fini

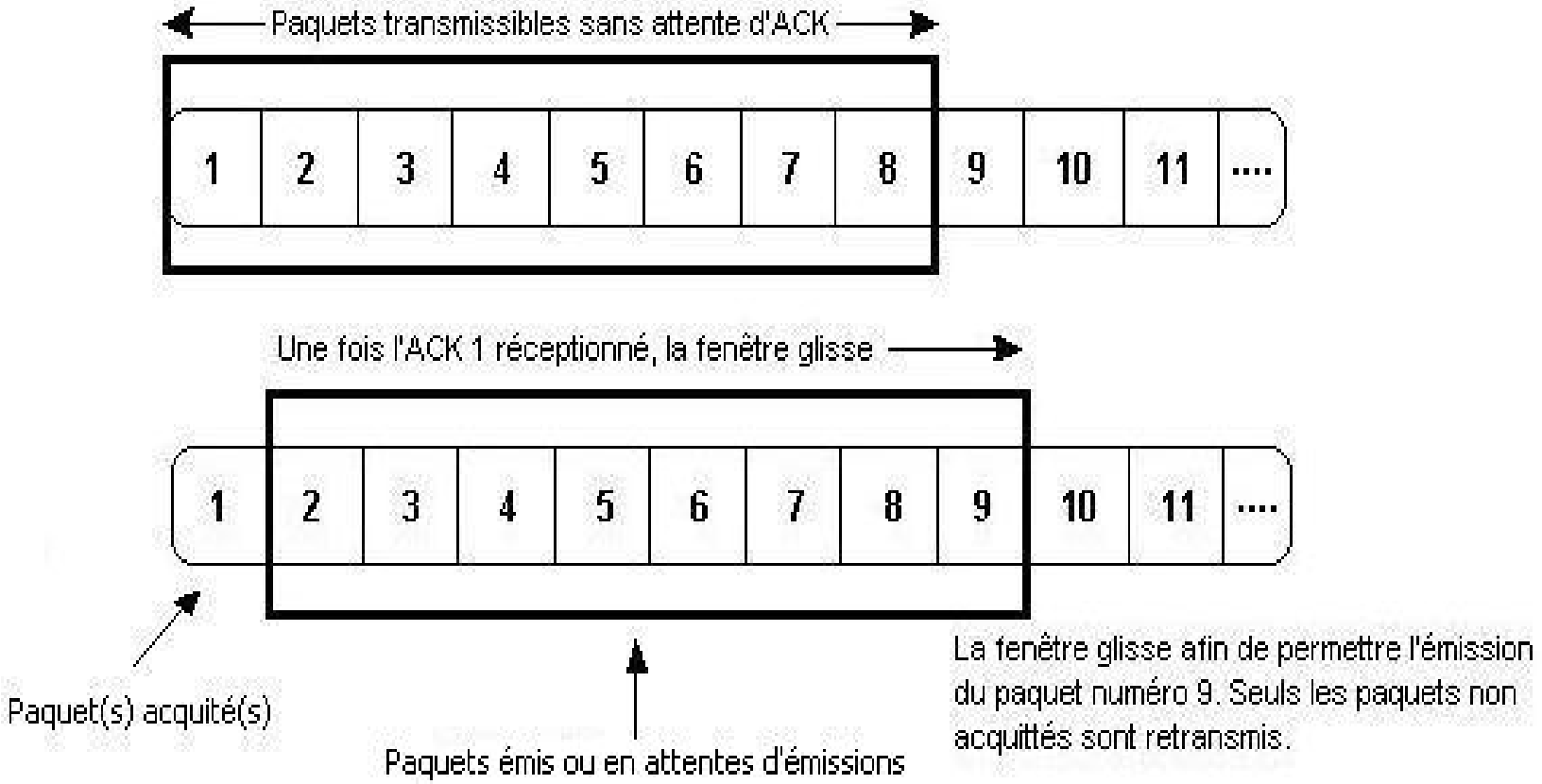


Le protocole TCP (suite)

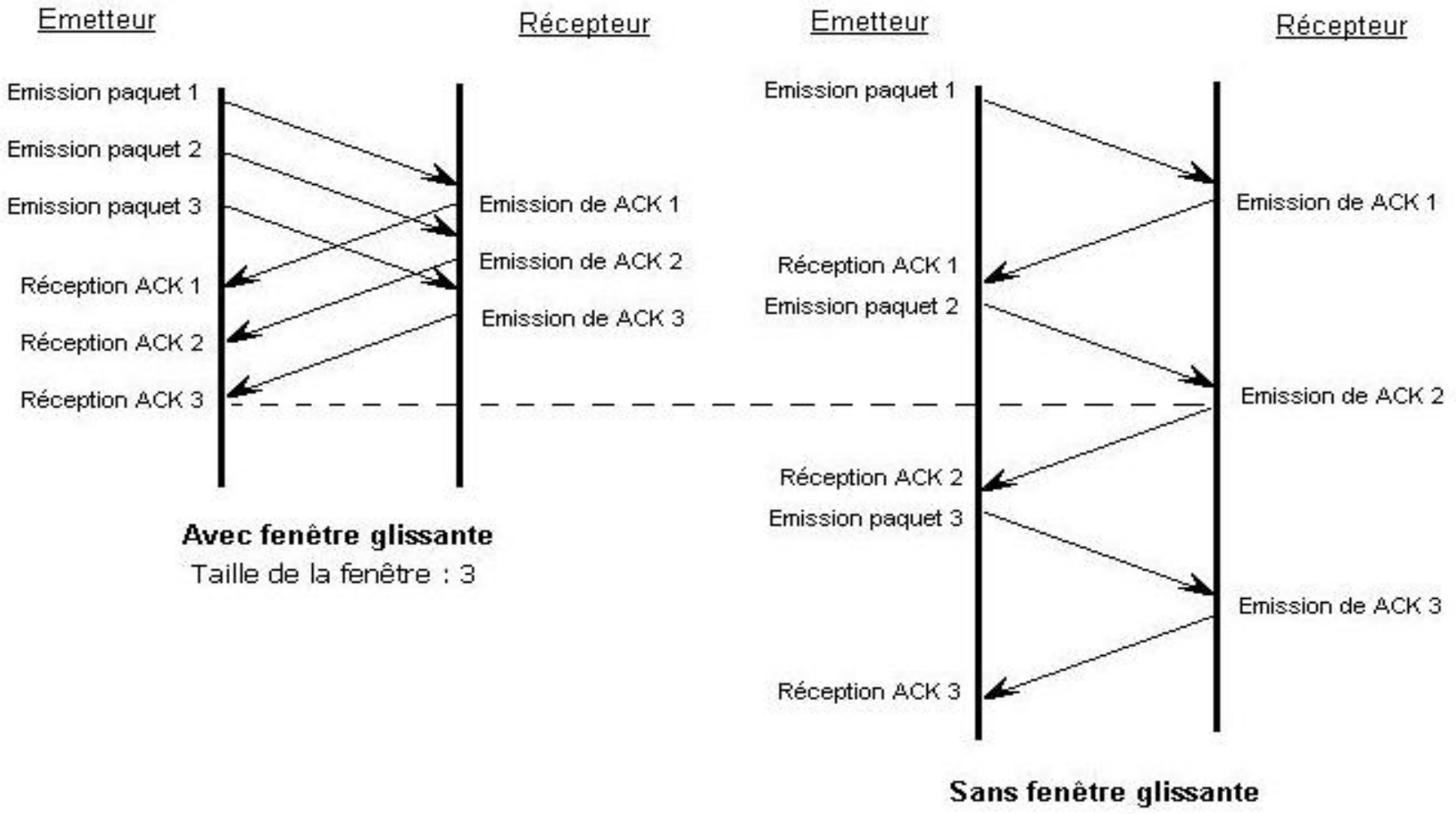
- Temporisateurs TCP
  - RT « Retransmission Timer »
  - QT « Quiet Timer » pour gel du port à sa fermeture
  - PT « Persistence Timer » : utile lorsque la fenêtre de réception est à 0, l'émetteur envoie un octet périodiquement pouvant ainsi vérifier que le récepteur n'a pas abandonné la connexion, ce dernier renvoie un ack avec la fenêtre à 0 ou +
  - A la version originale s'est rajouté : le KAT « Keep-Alive Timer » pour l'envoi périodique d'un paquet vide afin de s'assurer que l'autre extrémité est encore connectée, si aucune réponse au bout de IT « Idle Timer » la connexion est libérée. KAT entre 5 et 45 sec. IT ~ 360 sec.

TCP - La fenêtre glissante

- La fenêtre glissante du protocole opère au niveau de l'octet
- Sa taille peut varier dans le temps
- Elle permettra un contrôle de flux

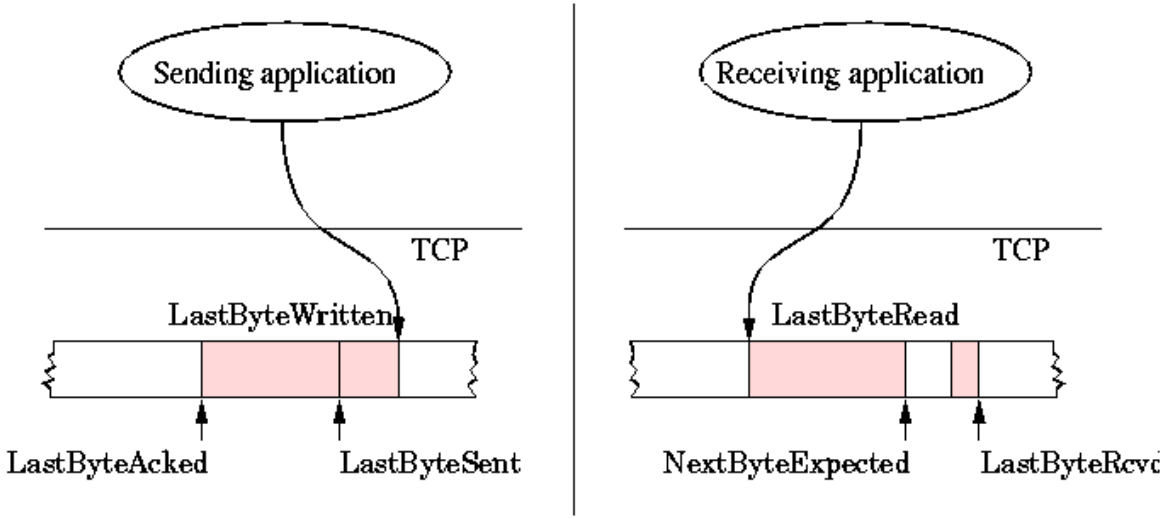


TCP - La fenêtre glissante



TCP - La fenêtre glissante

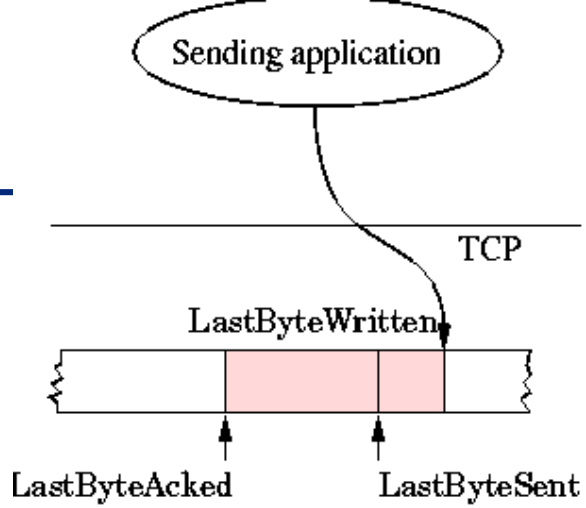
Sliding Window Revisited

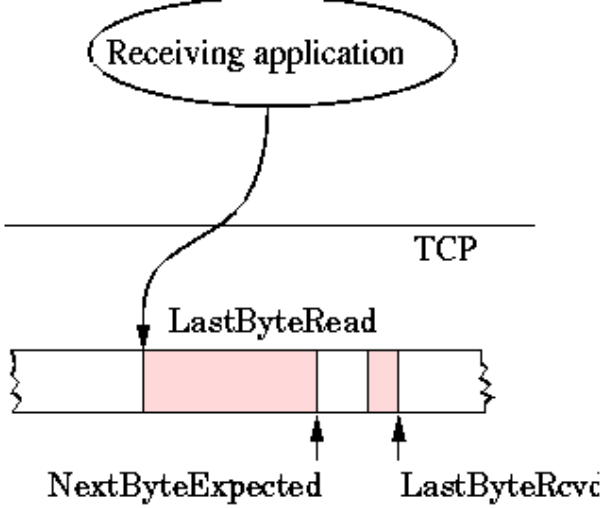


- Each byte has a sequence number
- ACKs are cumulative

- Sending side
  - `LastByteAcked` Š `LastByteSent`
  - `LastByteSent` Š `LastByteWritten`
  - bytes between `LastByteAcked` and `LastByteWritten` must be buffered
- Receiving side
  - LastByteRead < NextByteExpected
  - bytes between `NextByteRead` and `LastByteRcvd` must be buffered

TCP

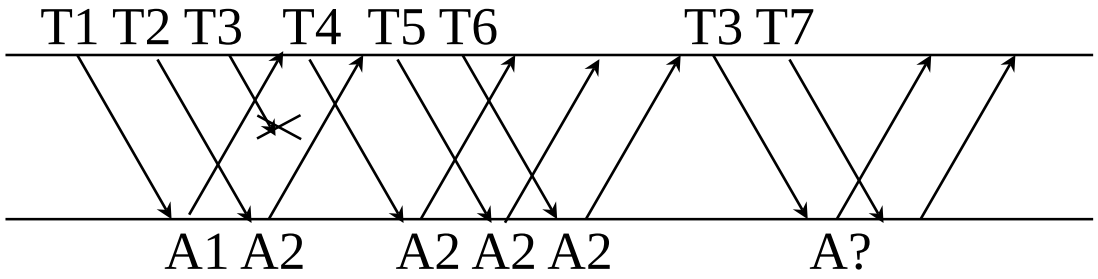




- Contrôle de flux
  - « Send buffer size »: MaxSendBuffer
  - « Receive buffer size »: MaxRcvBuffer
  - Receiving side
    - LastByteRcvd - LastByteRead < = MaxRcvBuffer
    - AdvertisedWindow = MaxRcvBuffer -(NextByteExpected -LastByteRead)
  - Sending side
    - LastByteWritten - LastByteAcked < = MaxSendBuffer
    - LastByteSent - LastByteAcked < = AdvertisedWindow
    - EffectiveWindow = AdvertisedWindow -(LastByteSent -LastByteAcked)
    - block sender if (LastByteWritten - LastByteAcked) + y > MaxSenderBuffer
  - « Always send ACK in response to arriving data segment »
  - « Persist when AdvertisedWindow = 0 »

Le protocole TCP (suite)

- La politique de transmission
  - Quand envoyer les ACK ?
  - Réponse
    - Accepter les segments dans la fenêtre, les acquitter dans l'ordre, temporiser la retransmission.
    - Utilisation du «piggybacking» : (RFC 1122) ne pas retarder l'ACK plus de 0.5 sec.
    - L'heuristique du « Fast Retransmit » : si l'émetteur reçoit 3 ACK dupliqués du même octet avant l'expiration du temporisateur alors retransmettre le segment suivant uniquement.



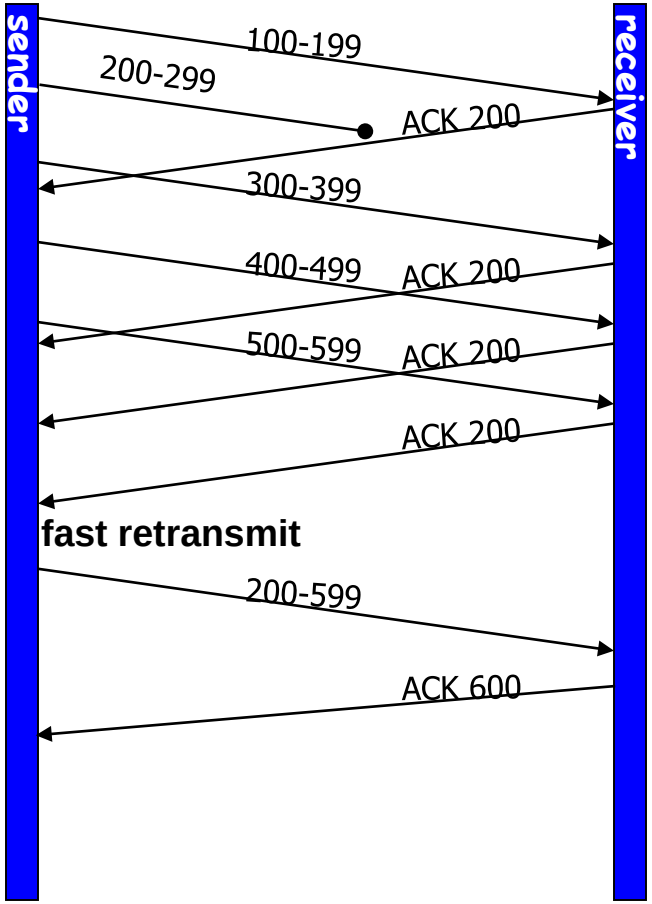
Le protocole TCP (suite)

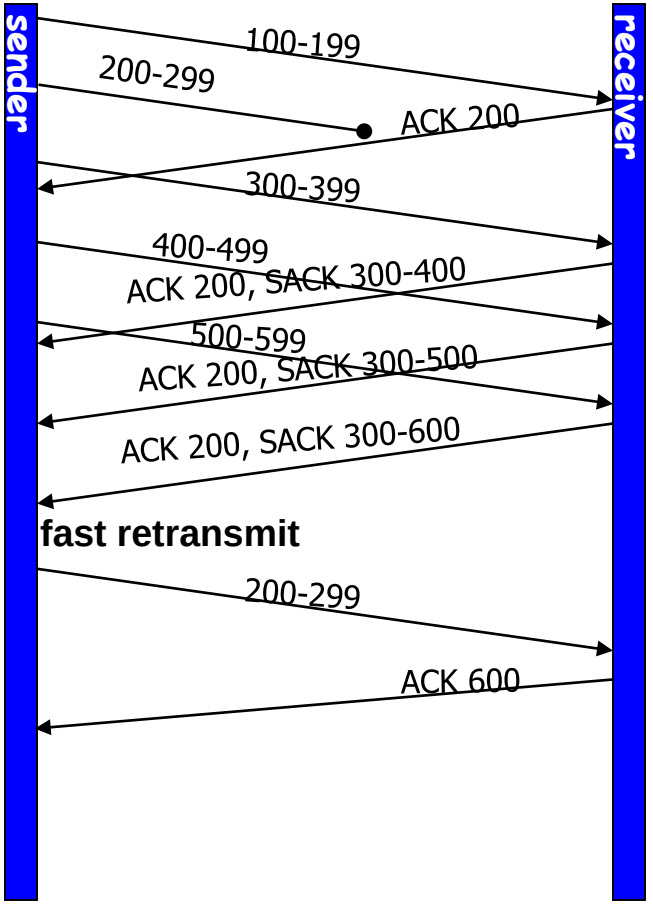
- Génération des ACK [RFC 1122, RFC 2581]

Event at Receiver	TCP Receiver action
Arrival of in-order segment with expected seq #. All data up to expected seq # already ACKed	Delayed ACK. Wait up to 500ms for next segment. If no next segment, send ACK
Arrival of in-order segment with expected seq #. One other segment has ACK pending	Immediately send single cumulative ACK, ACKing both in-order segments
Arrival of out-of-order segment higher-than-expect seq. # . Gap detected	Immediately send duplicate ACK, indicating seq. # of next expected byte
Arrival of segment that partially or completely fills gap	Immediate send ACK, provided that segment starts at lower end of gap

Le protocole TCP (suite)

- Génération des SACK "selective ACK" [RFC 2018]
  Avec SACK vs sans SACK





Rq: Bien que déconseillé, le récepteur peut par la suite rejeter des données qui ont été préalablement SACK

Le protocole TCP (suite)

–Le standard laisse libre plusieurs choix :

- Comment initialiser le temporisateur de retransmission ?
- Quand envoyer les segments de données ?
- Quand envoyer les mises-à-jour de la taille de la fenêtre ?
- Comment contrôler les problèmes de congestion ?

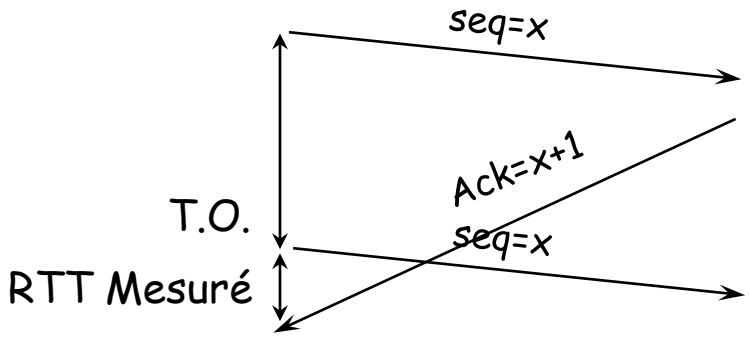
–Plusieurs algorithmes d'optimisation ont été proposés et sont retenus actuellement

- Estimation du temps de boucle
- Algorithme de Nagle
- Résolution du syndrome de la fenêtre stupide
- Contrôle de la congestion

Le protocole TCP -- Round Trip Time

- TCP prend en compte les variations de délai en utilisant un algorithme de retransmission adaptatif
- TCP calcule puis conserve l'estimation du temps de boucle moyen (RTT :Round Trip Time)
- Problème :
  - Contrairement au cas d'une liaison, le délai pour recevoir l'ACK est variable. Le sous/sur estimer aboutirait à de mauvaises performances.
- Solutions
  - A l'origine : ajuster en permanence la valeur du RTT et donc du Temporisateur.
    - RTT = a*RTT_ancien + (1-a) * NouvMesureRTT
      (a ≈ 7/8 ; entre 0.8 et 0.9)
    - Timeout = 2* RTT
  - Autres optimisations [Jacobson & Karels] : tenir compte de l'écart-type du RTT.

Le protocole TCP -- Round Trip Time



RTT«Round Trip Time»

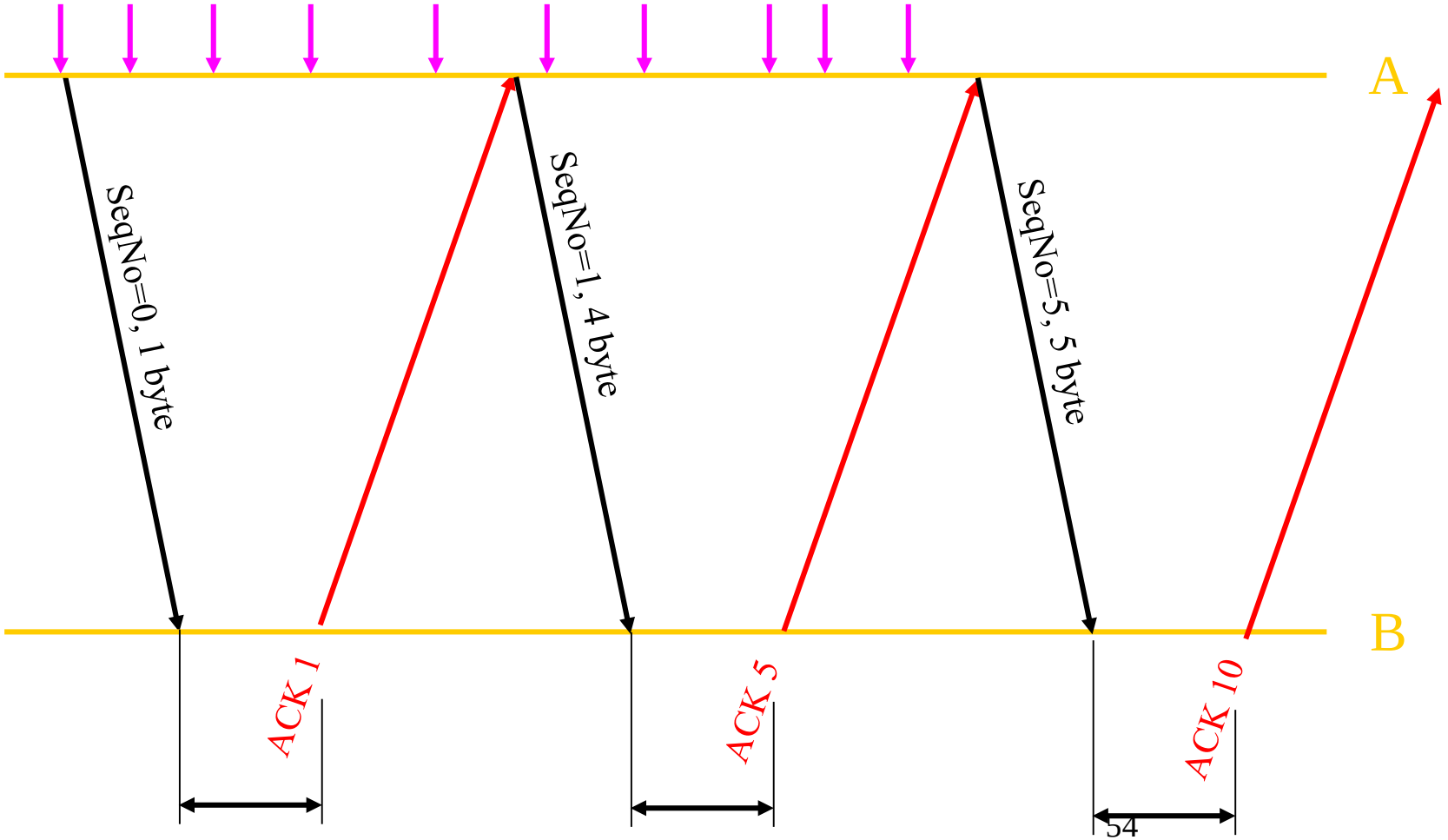
Quand le temporisateur expire, doubler le « Timeout » à chaque échec (augmente d'une façon exponentielle) jusqu 'à ce que passe le segment ou Max. 240 sec.

Le protocole TCP (suite)

- Algorithme de Nagle
  - Problème : (Tinygrams)
    - L'envoi de petits segments issus de l'application => surcoût important
  - Solution :
    - Envoyer le premier octet et accumuler le reste dans un tampon jusqu'à la réception de l'ACK (du premier octet) ou l'expiration d'un temporisateur
    - puis envoyer les octets accumulés en appliquant le même principe.
    - S'il y a assez de données pour remplir au moins la moitié du tampon, envoyer les données.
    - L'algorithme peut être désactivé par l'application

Le protocole TCP (suite)

Caractères tapés



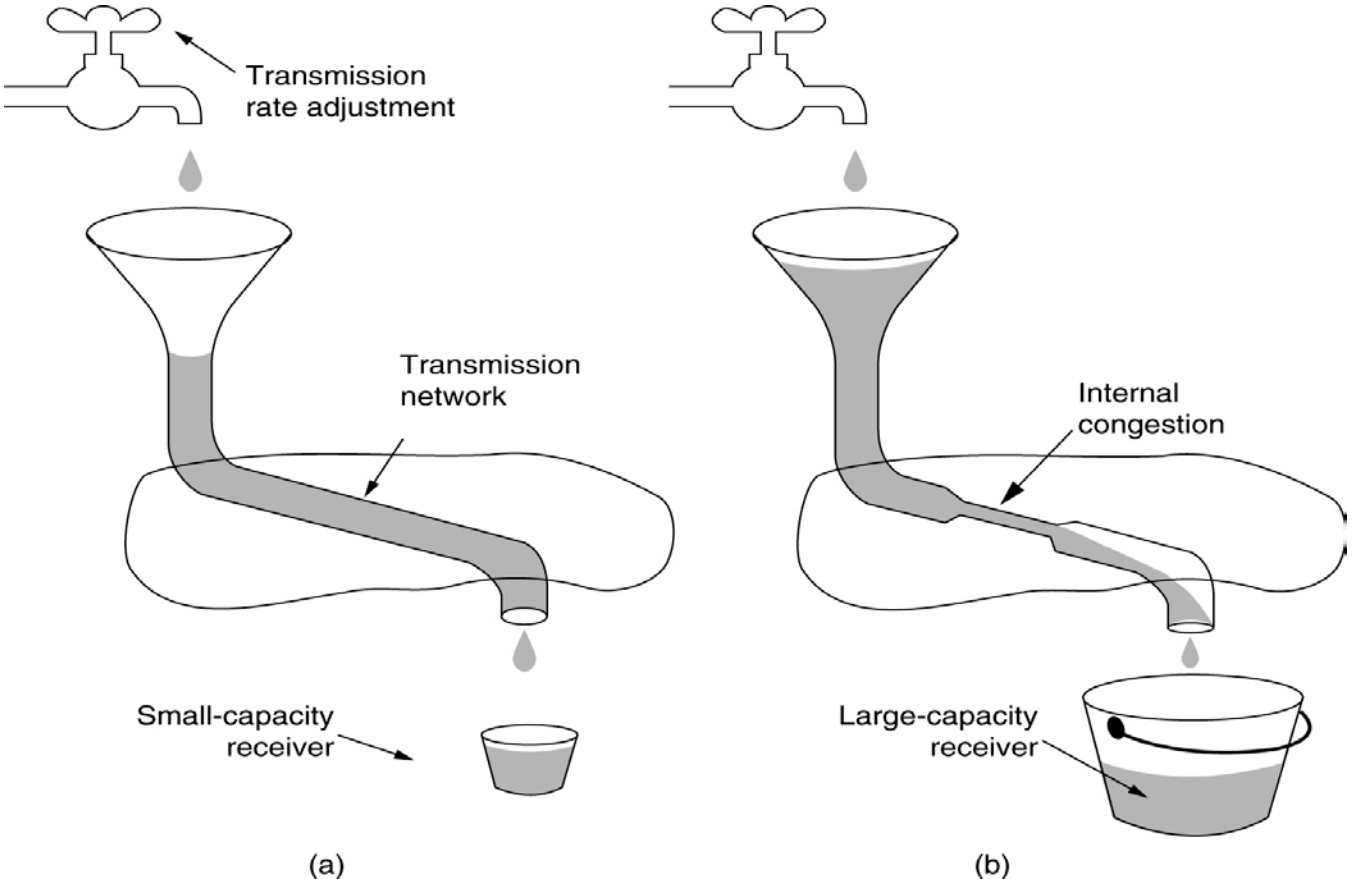
ACK délayé

Le protocole TCP (suite)

- Eviter le syndrome de la fenêtre stupide
  - Problème :
    - L'émetteur dispose d'une large quantité d'information à envoyer alors que la taille de la fenêtre d'anticipation oblige l'émetteur à envoyer de petits segments.
    - Le tampon du récepteur est rempli, ce dernier acquitte quelques octets et propose ainsi de faibles crédits
  - Solutions :
    - Le récepteur ne propose une augmentation de la fenêtre que lorsqu'il dispose d'une mémoire vide supplémentaire dont la taille est supérieure à min de (moitié de la taille du tampon du récepteur, taille maximale d'un segment).
    - Cette solution est suffisante. En plus, pour remédier au cas où le récepteur ne la met pas en œuvre, l'émetteur devrait attendre d'avoir un crédit dans la fenêtre et une taille d'un segment d'au moins la moitié de la taille estimée du tampon (=le crédit maximum proposé). Ensuite, appliquer l'algorithme de Nagle (si activé).

Le protocole TCP (suite)

- Contrôle de la congestion
  - Problème :
    - Le débit de l'émetteur est supérieur soit à la capacité du réseau soit à la capacité du récepteur ce qui provoque la perte de paquets, d'où l'apparition d'une congestion

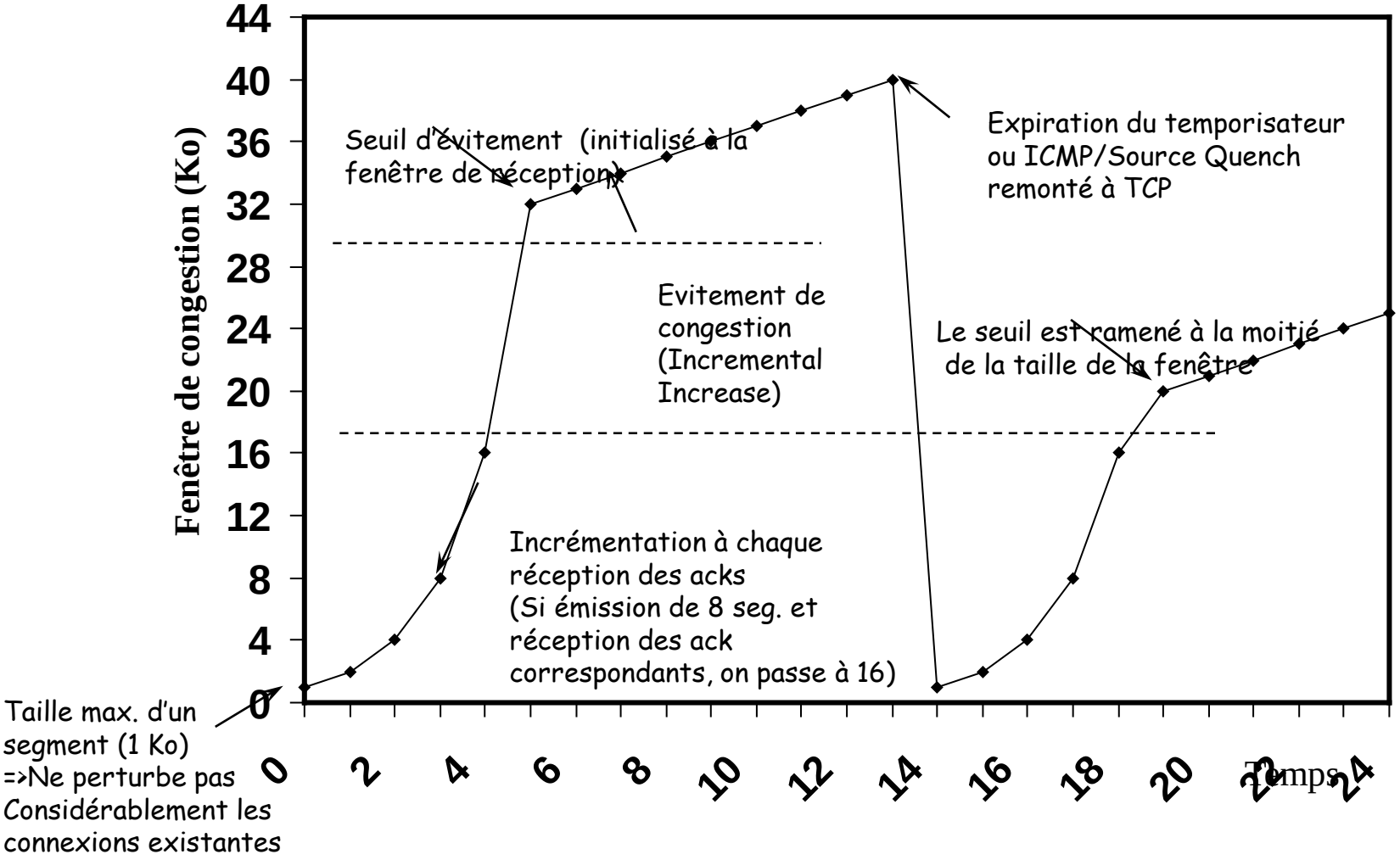


Le protocole TCP (suite)

- Contrôle de congestion
  - Solutions :
    - La détection se base sur le fait que si un paquet est perdu ceci est probablement dû à un problème de congestion plutôt qu'une erreur de transmission, le taux d'erreur sur les supports de transmission actuels est considéré faible.
    - Le récepteur peut éviter la congestion en spécifiant une taille de la fenêtre en fonction de la taille de la mémoire libre
    - Reste à résoudre la congestion dûe au réseau : algorithme du démarrage lent «Slow Start». Cet algorithme utilise une seconde fenêtre dite de congestion.
    - TCP utilise la fenêtre de congestion
      Fenêtre_autorisée = min (indication_réception, fenêtre_congestion)

Le protocole TCP (suite)

- Contrôle de la fenêtre de congestion

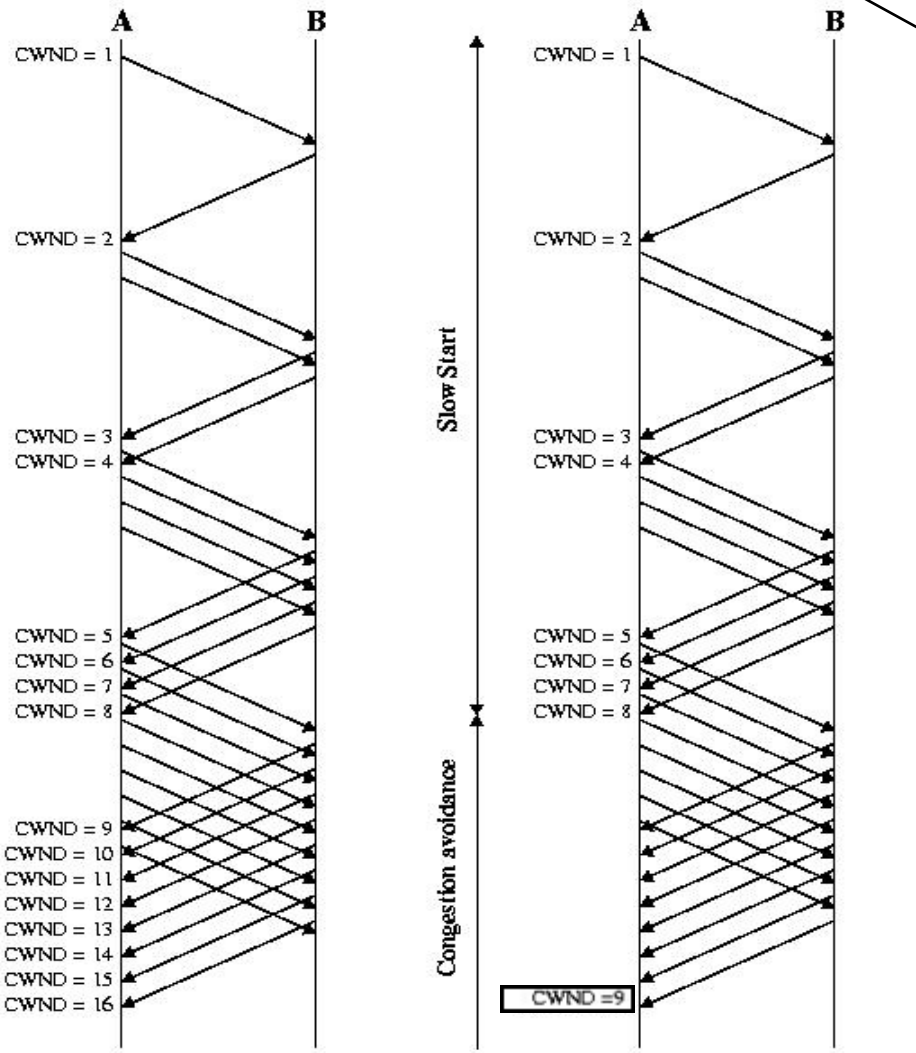


Le protocole TCP (suite)

- Effet du « Slow Start » et du « Congestion Avoidance »

Cwnd (+1 à chaque Ack reçu)

Cwnd (+1/cwnd à chaque Ack reçu)



(a) Slow start, ending with a timeout

(b) Slow start followed by congestion avoidance

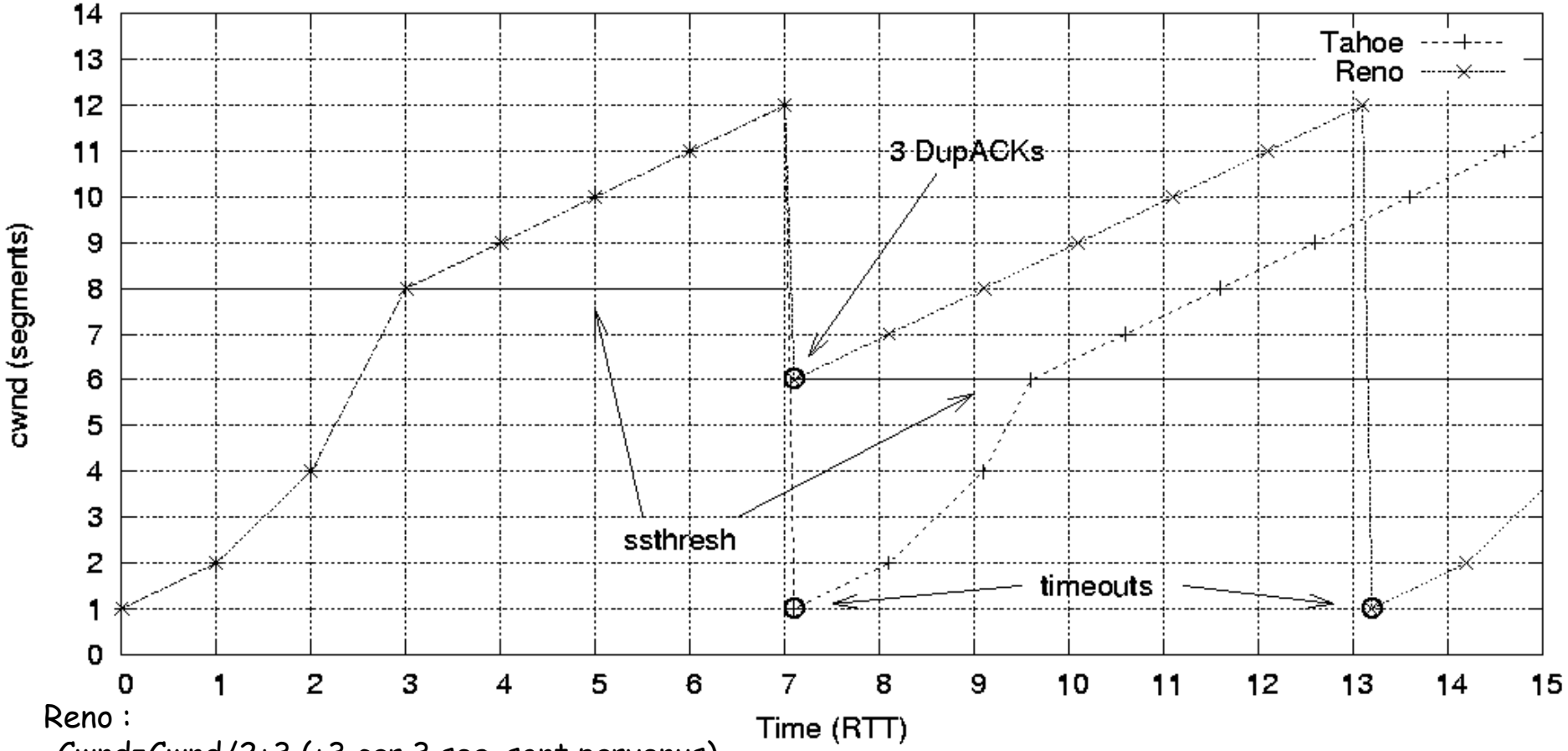
Le protocole TCP (suite)

- Tahoe
  - Phase "Slow Start" (cwnd < ssthresh)
    - Initiallement:
      - cwnd = 1*MSS (Maximum Segment Size)
      - ssthresh important.
    - Si pas de perte:
      - cwnd += 1*MSS (après chaque nouveau ACK)
      - *(ce qui donne une augmentation exponentielle de cwnd)*
    - Si perte :
      - ssthresh = max(flight size/2, 2*MSS)
      - cwnd = 1*MSS
  - Phase "Congestion Avoidance" (cwnd > ssthresh)
    - Si pas de perte:
      - augmenter cwnd *d'au plus* 1*MSS par RTT
      - cwnd += ((MSS/cwnd) * MSS) à chaque ACK (*approximation* pour augmenter cwnd de 1*MSS par RTT)
    - Si perte:
      - ssthresh = max (flight size/2, 2*MSS)
      - cwnd = 1*MSS.

  - La réception d'un dupACKs signifie pour l'émetteur que le récepteur est toujours entrain de recevoir de nouveaux segments. Pourquoi revenir au "Slow start" après le "fast retransmit" ? => Reno

Le protocole TCP (suite)

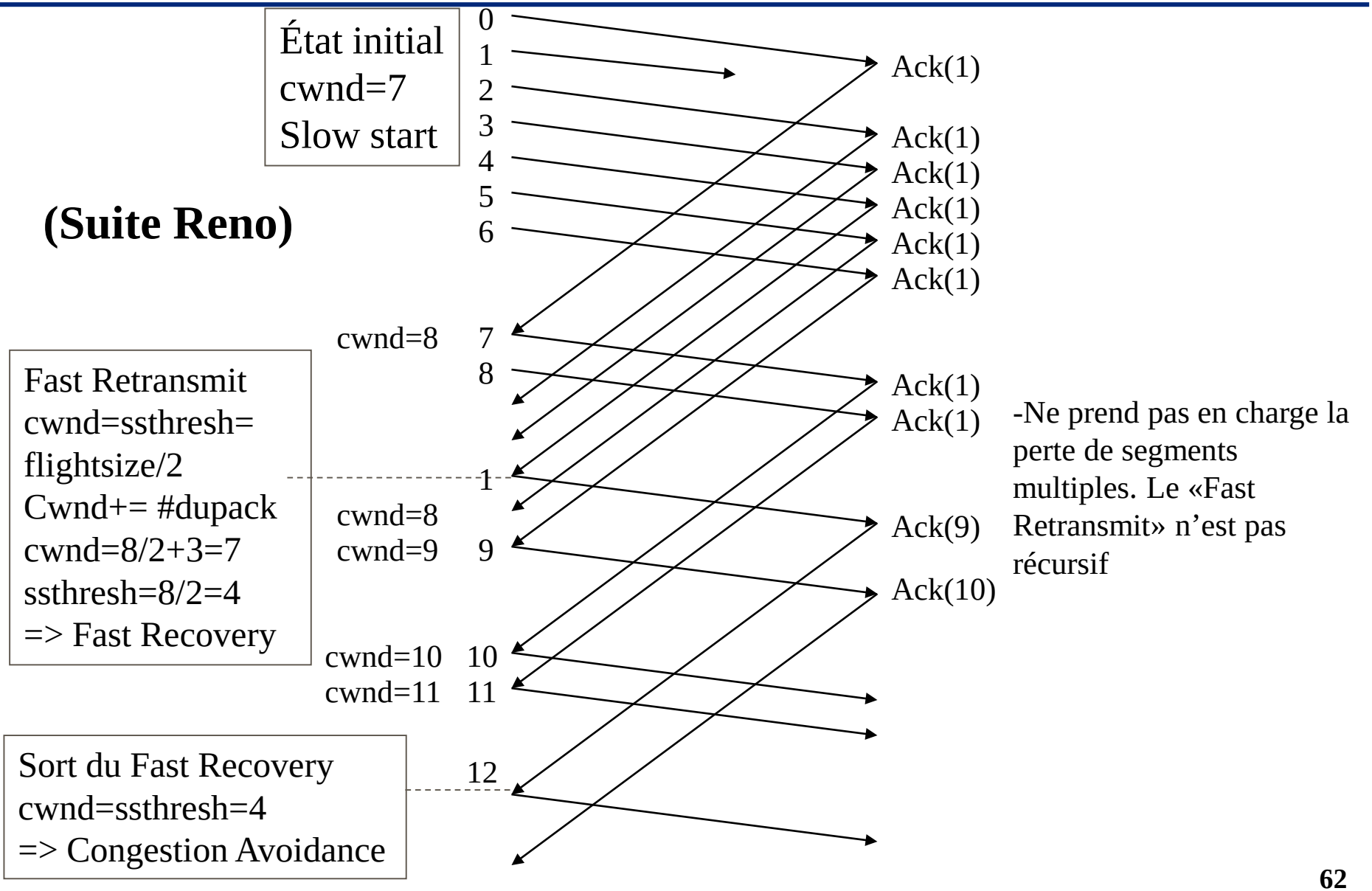
- Contrôle de la fenêtre de congestion (Reno)



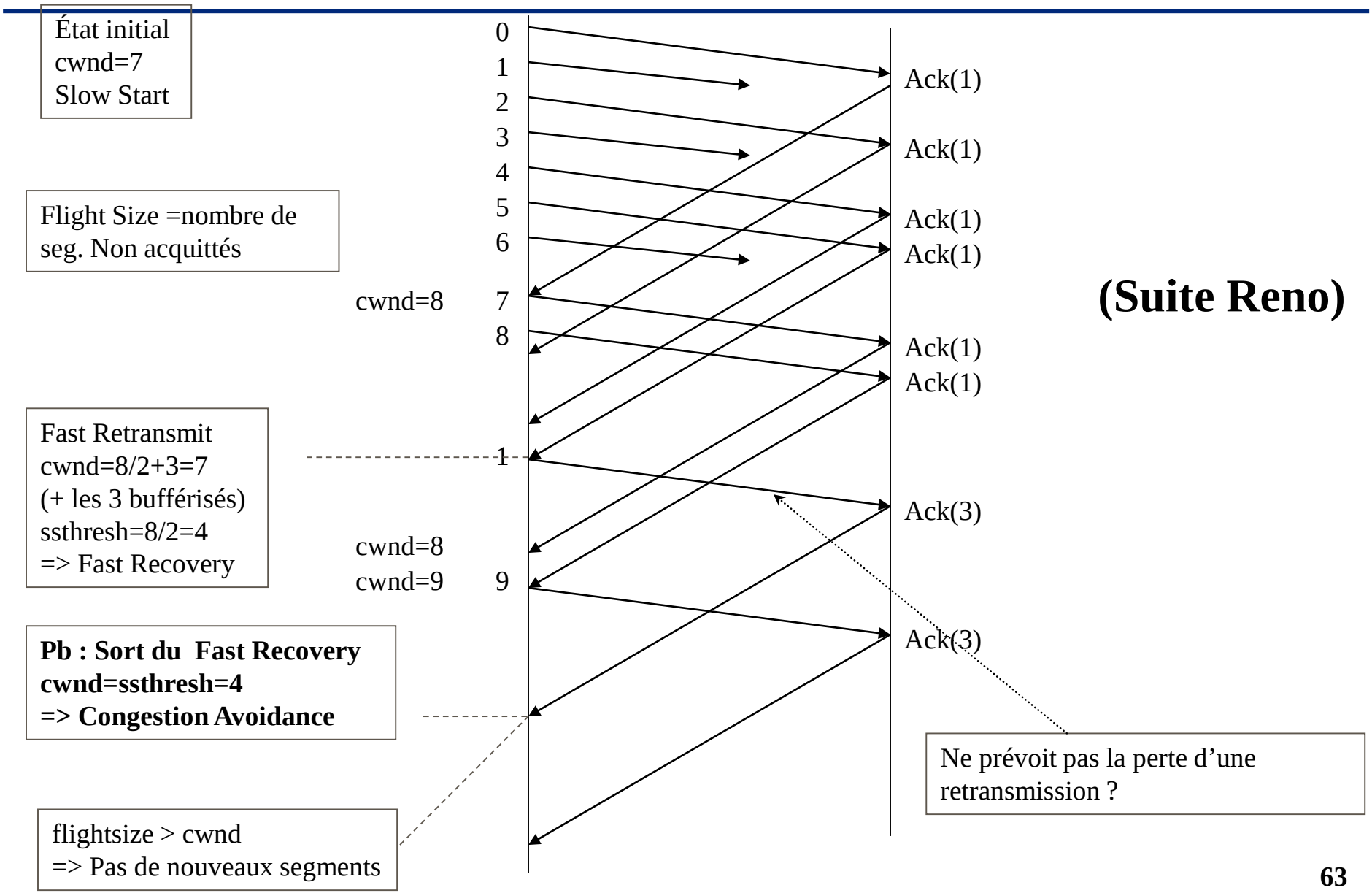
Reno :
-Cwnd=Cwnd/2+3 (+3 car 3 seg. sont parvenus)
Continuer à envoyer des seg. si autorisé par la valeur de cwnd
-Jusqu'a réception d'un nouveau ACK différent (qui acquitte tous les segments envoyés avant la phase de « Fast Recovery » si on suppose qu'il n y a pas d'autres segments qui se sont perdus)
-Pour chaque dupAck incrémenter de 1 Cwnd

Le protocole TCP (suite)

(Suite Reno)



Le protocole TCP (suite)



(Suite Reno)

Le protocole TCP (suite)

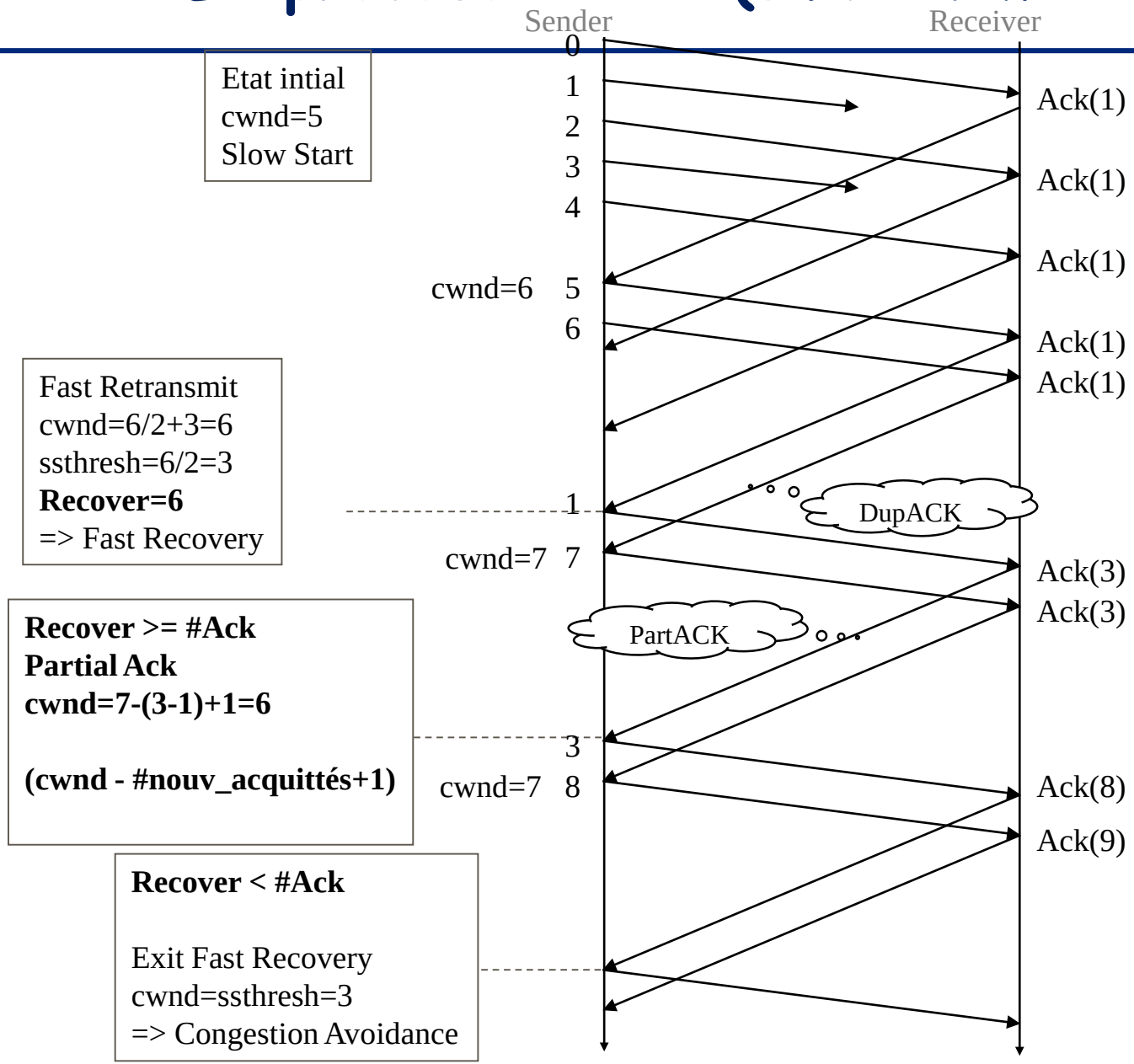
Contrôle de congestion : les solutions

- ✓ Le démarrage lent (Slow Start)
- ❑ Multiplicative decrease + Incremental Increase
  - ☞ En cas de perte d'un segment, il faut réduire la fenêtre de congestion d'un facteur multiplicatif (de moitié;)
  - ☞ Quand l'émission de trafic commence sur une nouvelle connexion ou reprend après une période de congestion, il faut commencer par une fenêtre de congestion limitée à un seul segment et incrémenter la fenêtre de congestion d'un segment à la fois, chaque fois qu'un accusé de réception est reçu.
- ❑ La phase d'évitement de congestion (congestion avoidance)
  - ☞ Objectif : ralentir le rythme d'augmentation de la fenêtre pour éviter une nouvelle congestion
  - ☞ Lorsque la fenêtre atteint 1 fois et demi sa taille initiale, la fenêtre est augmentée de 1 si tous les segments de la fenêtre ont été acquittés

Le protocole TCP (new Reno)

- L'émetteur mémorise le num. du dernier segment envoyé (Recover) avant d'entrer dans la phase « Fast Retransmint » ➔ il est ainsi possible de déterminer si un ACK (non dupliqué) couvre bien les segments avant le «Fast Retransmit »
  - Si ce non couvert , l'émetteur émet le segment perdu et reste dans le « Fast Recovery »
  - Si c'est couvert, l'émetteur termine le « Fast Recovery »
  - Appliqué aussi dans SACK

Le protocole TCP (suite new Reno)



Le protocole TCP (suite)

- TCP Vegas
  - les sources cherchent à trouver des signes qui montrent qu'une file d'attente d'un certain routeur est entrain de croitre et qu'une congestion va bientôt avoir lieu : le RTT croit, le débit s'affaiblit
  - Algorithme
    - BaseRTT = le minimum des RTTs mesurés
    - ExpectedRate = CongestionWindow / BaseRTT
    - La source calcule le débit d'envoi courant ActualRate à chaque RTT
    - Diff = ExpectedRate – ActualRate
    - Si Diff < α alors augmenter cwnd linéairement
      Sinon si Diff >β alors réduire cwnd linéairement
      sinon garder cwnd

Le protocole TCP (suite)

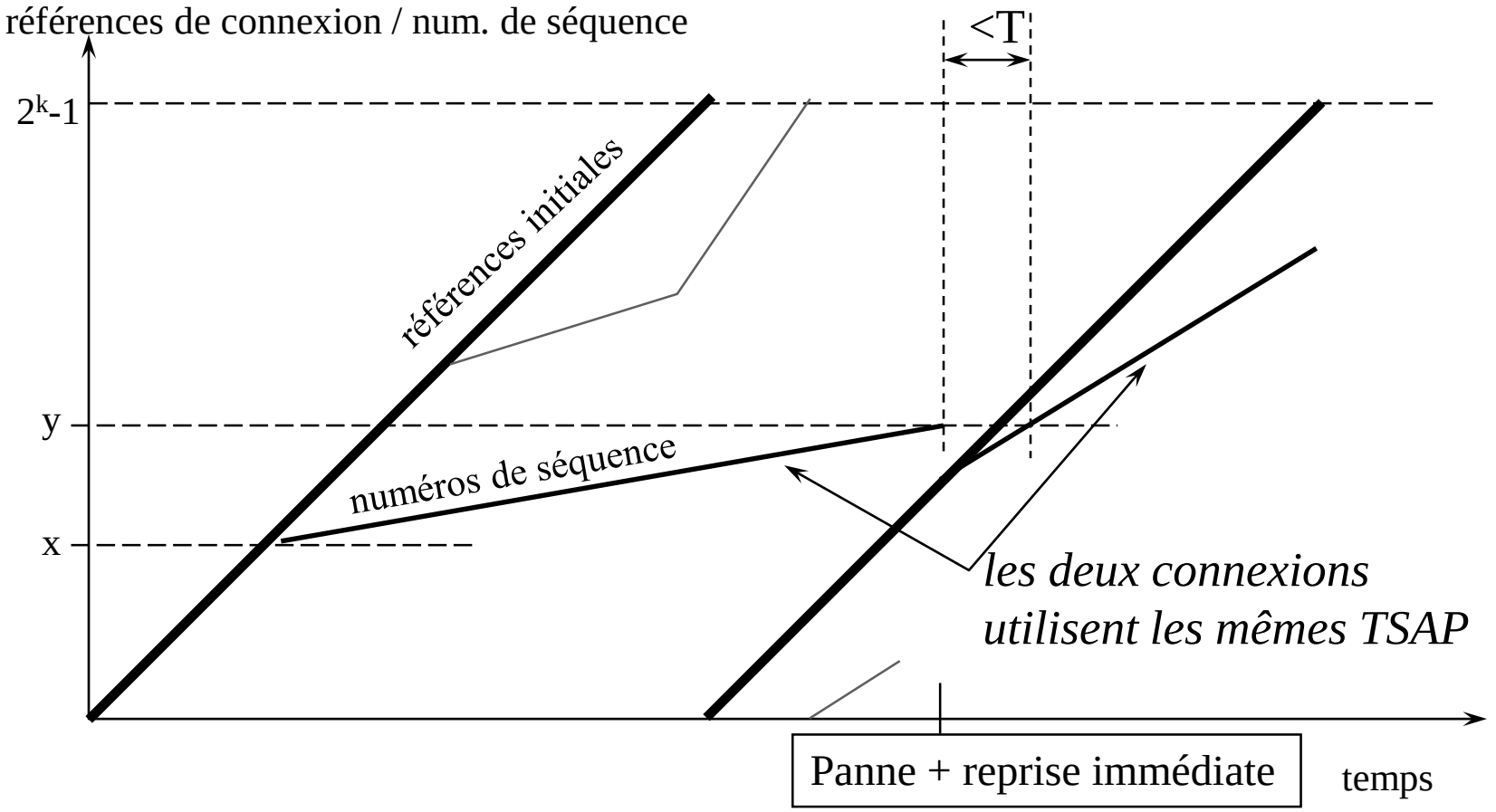
- « Active Queue Management »
  - Détecter "tôt" une congestion
  - Essayer de garder la taille des F.A. dans une marge
  - Choisir aléatoirement une IP-PDUs pour la notification de congestion
- TCP / « Explicit Congestion Notification »
  - Est un mécanisme AQM
  - TCP/IP utilise des signaux ECN
  - TCP traite les signaux ECN comme le cas de la détection d'une perte de paquet sauf que les paquets ne sont pas réellement perdus
  - Est une option négociée à l'ouverture d'une connexion
  - Un routeur intermédiaire doit aussi supporter cette option
  - Windows Vista / 2008 server, Linux … supporte ECN, mais par défaut désactivé, pour activer
    - Sous Windows: netsh interface tcp set global ecncapability=enabled
    - Sous Linux: sysctl net.ipv4.tcp_ecn=1
  - Les routeurs Cisco (à partir de 12.8(T)) et Linux peuvent être configuré pour supporter ECN en fixant la discipline de la AQM (Cisco : WRED Weighted Random early detection, Linux ; GRRED Gentle RED)

Eléments de protocole (suite)

- Principe de la méthode de Tomlinson & Sunshine
  - Soit T une durée, multiple de la durée de vie d'un paquet, au bout de laquelle un paquet et tous les accusés de réception s'y rapportant sont forcément disparus du réseau.
  - Supposons que tout hôte est équipé d'une horloge capable de continuer à fonctionner même si l'hôte tombe en panne.
  - A un paquet est associée un numéro de séquence spécifique
    - obtenu à partir d'un compteur initialisé à une valeur (référence de la connexion) qui correspond aux k bits de poids le plus faible de l'horloge ;
    - k est suffisamment grand pour que lorsque le compteur repasse à une référence déjà utilisée, tout paquet utilisant cette référence aurait disparu.

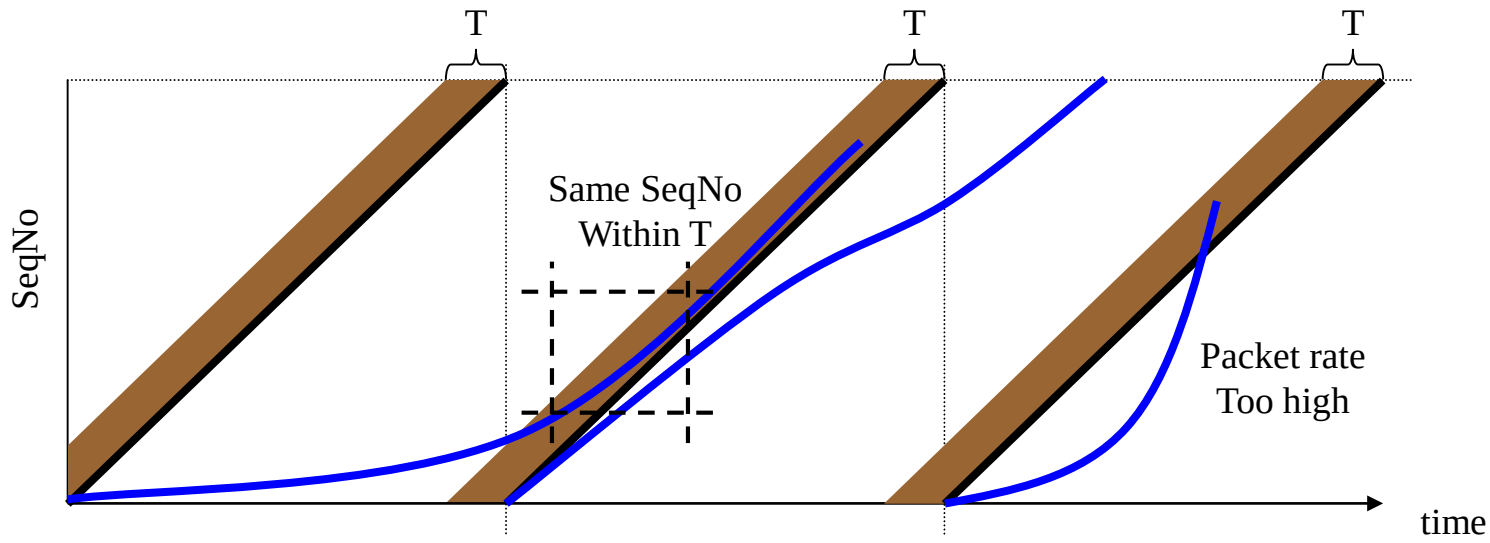
Eléments de protocole (suite)

- Problème traité par la méthode Tomlinson & Sunshine



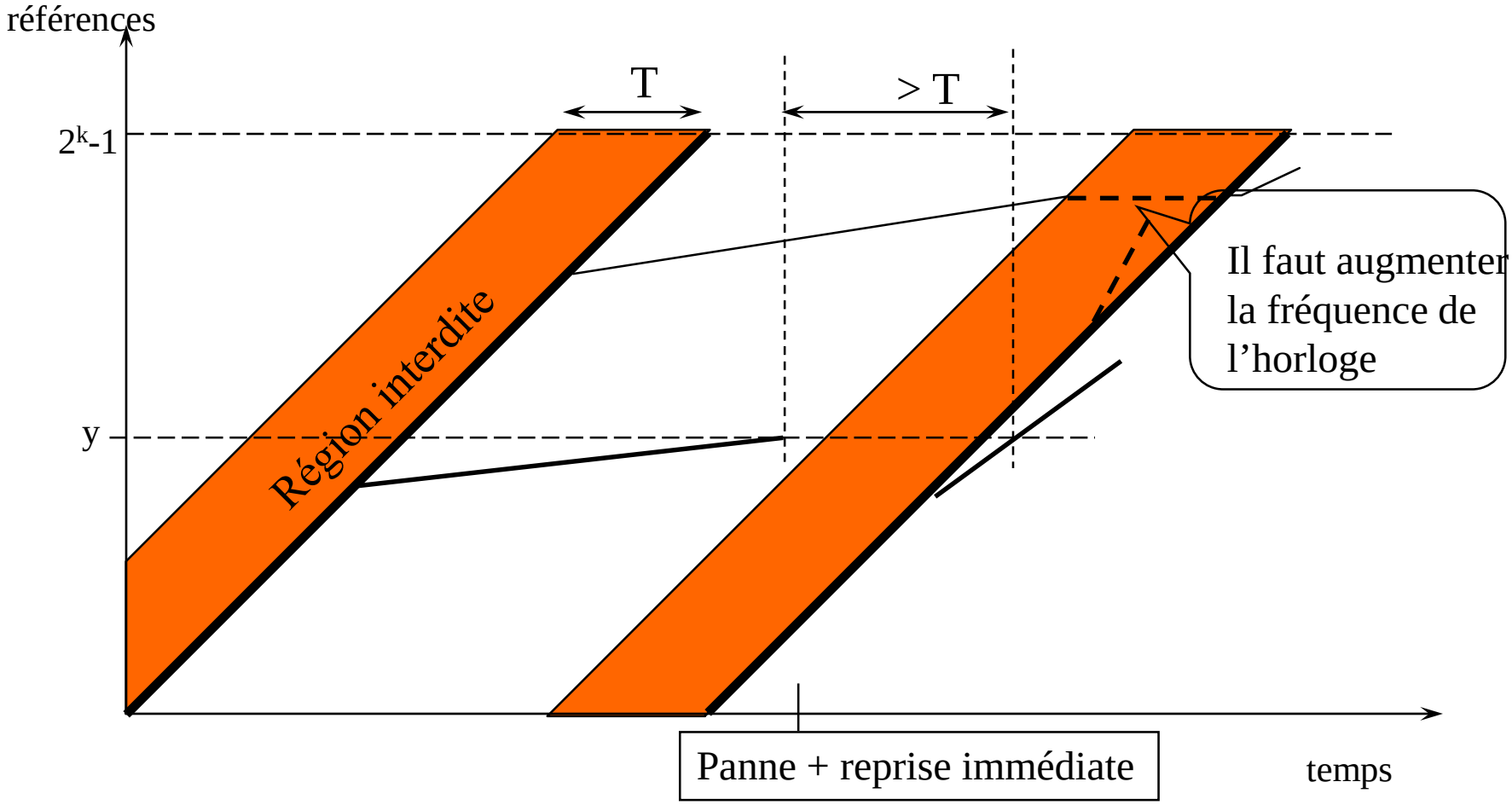
Le problème est qu'on a deux paquets différents ayant un même numéro de séquence, non espacés d'un temps supérieur à T. Solution simple mais pénalisante (quand T est important) : attendre T avant de reprendre.

Eléments de protocole (suite)



Eléments de protocole (suite)

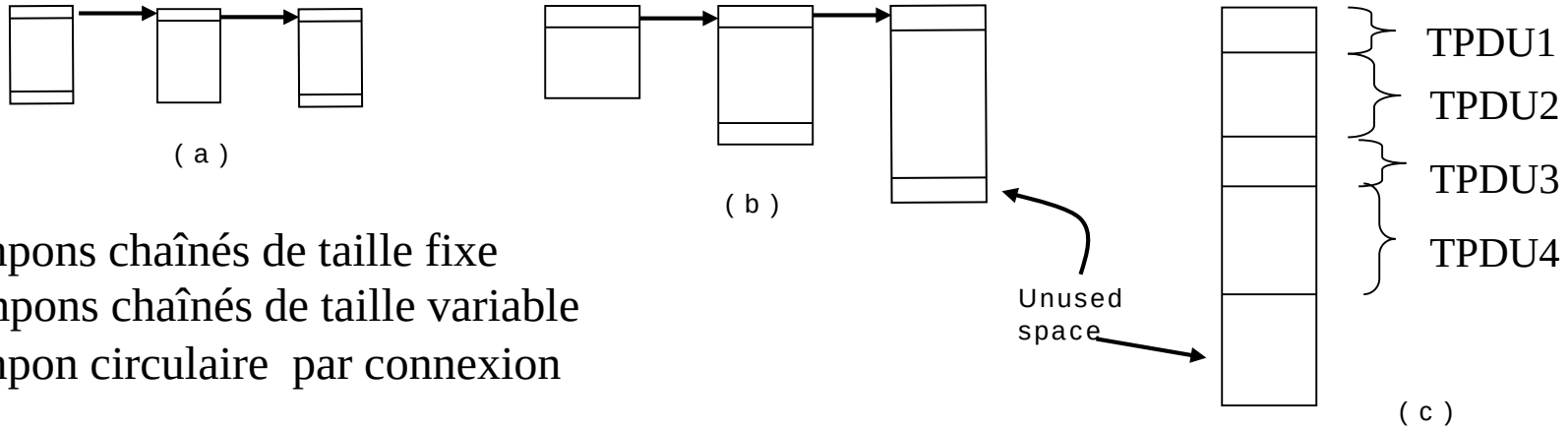
- Principe de la méthode Tomlinson



Cette méthode résout le problème des TPDU de données duplicata

Eléments de protocole (suite)

- Contrôle de flux et gestion de la mémoire tampon
  - –Etant donné que le nombre de connexions de transport peut être important, le stockage des TPDUs utilise des techniques différentes de ceux de la couche liaison : besoin d'ajuster dynamiquement la taille de la mémoire.
    - ➢ gestion de la mémoire tampon

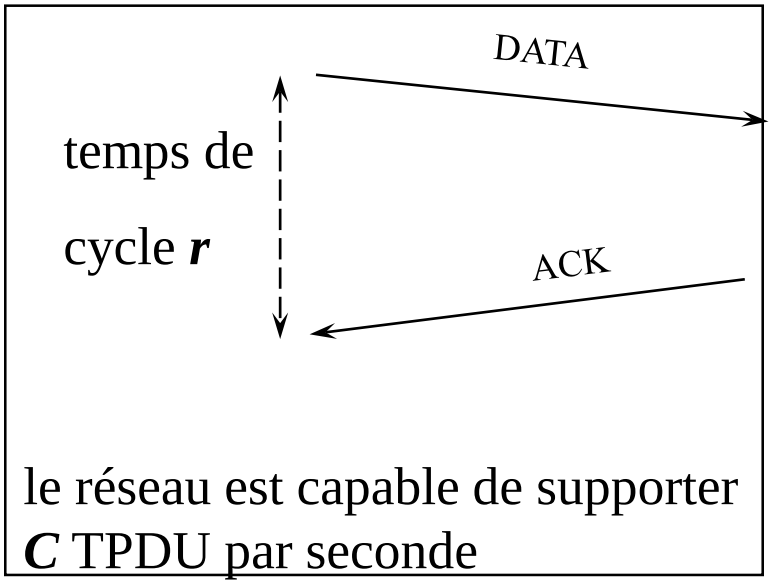


(a)Tampons chaînés de taille fixe
(b)Tampons chaînés de taille variable
(c)Tampon circulaire par connexion

- -

Eléments de protocole (suite)

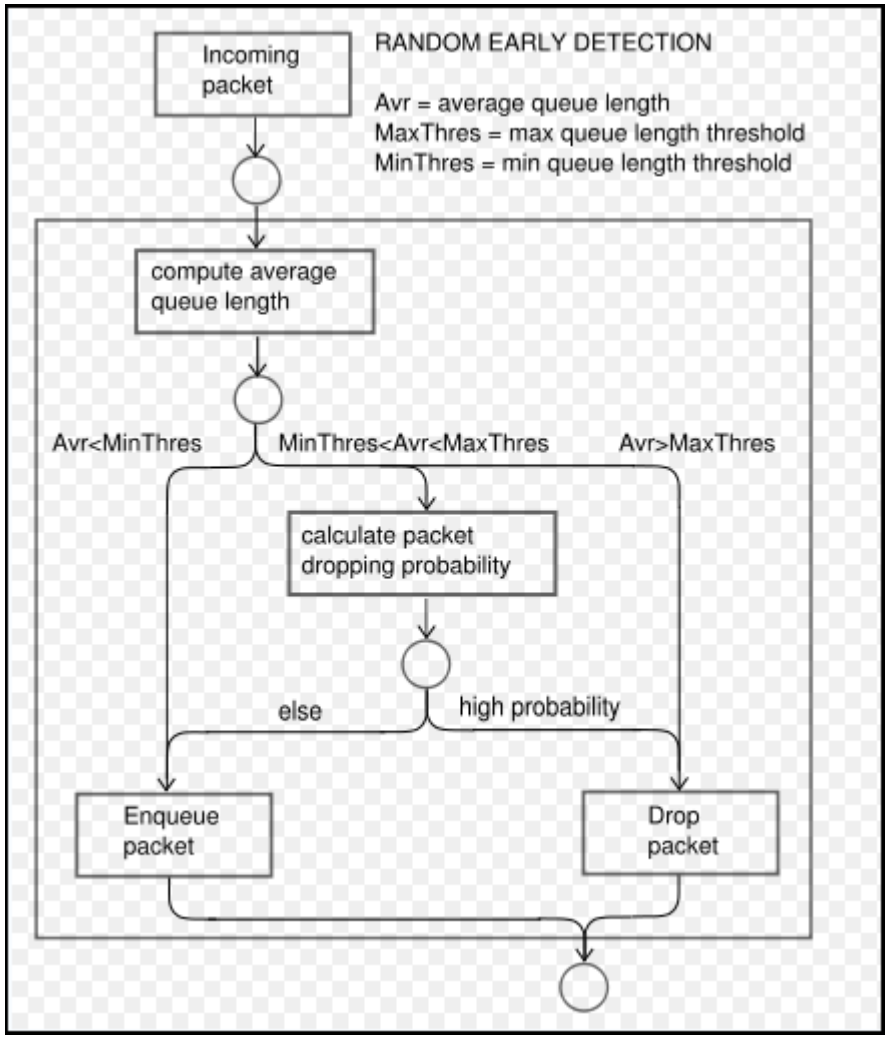
–ajuster dynamiquement la taille de la fenêtre d'anticipation en fonction de la capacité de transmission (débit) du réseau [Belsnes 1975]

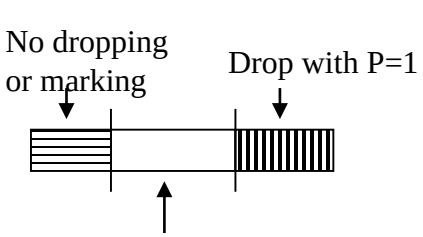


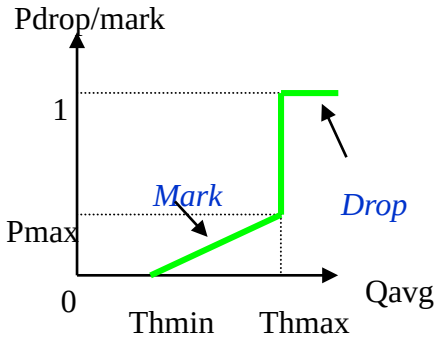
taille de la fenêtre d'anticipation = ***Cr***

C et r sont constamment réévalués

Le protocole TCP (suite)







Average Queue Length

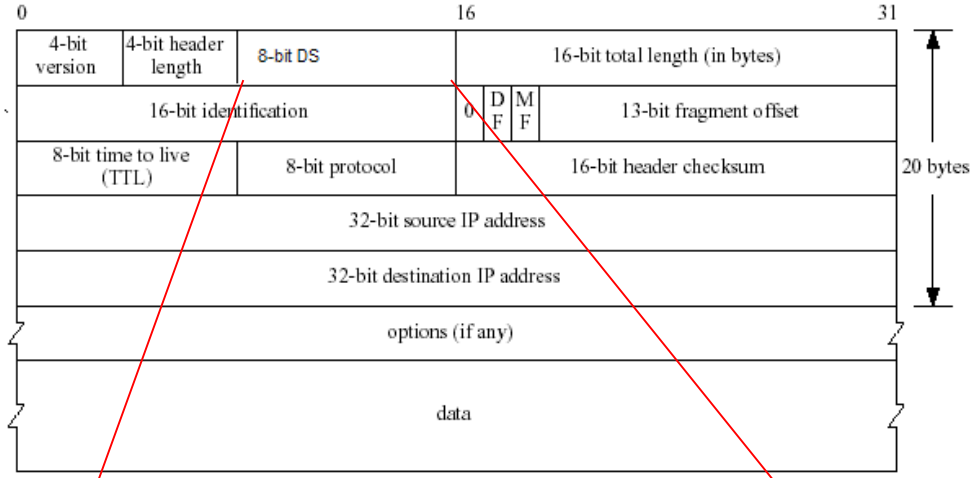
Drop Probability P

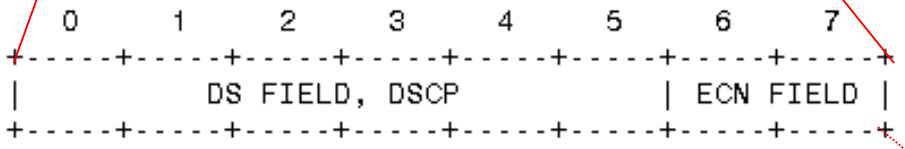
–WRED différente files peuvent avoir différents seuils

–GRED la transition probabiliste à toujours rejeter est progressive (pas « soudaine »)

Le protocole TCP (suite)

- Entête IP



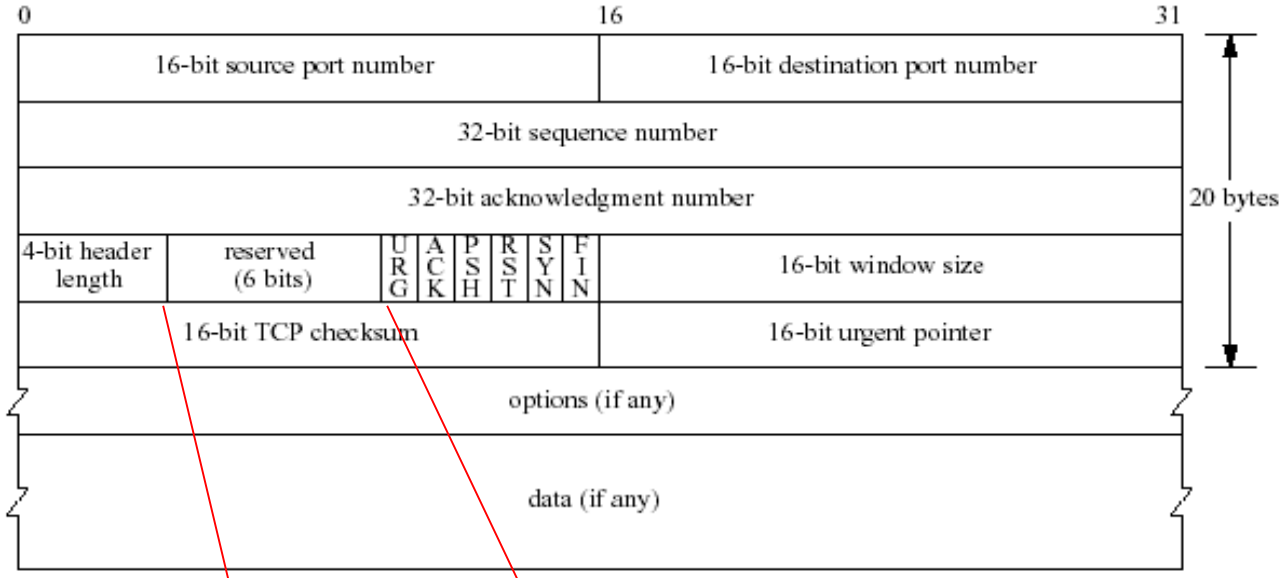


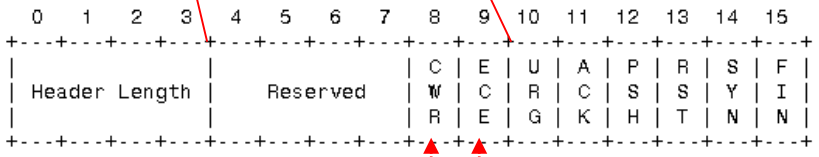
DSCP: differentiated services codepoint
ECN: Explicit Congestion Notification

2 bits => 4 ECN Codepoints

Value	Name
00	Not-ECT (Not ECN Capable Transport)
10	(ECN Capable Transport (0))
01	ECT(1) (ECN Capable Transport(1))
11	CE (Congestion Experienced)

Le protocole TCP (suite)

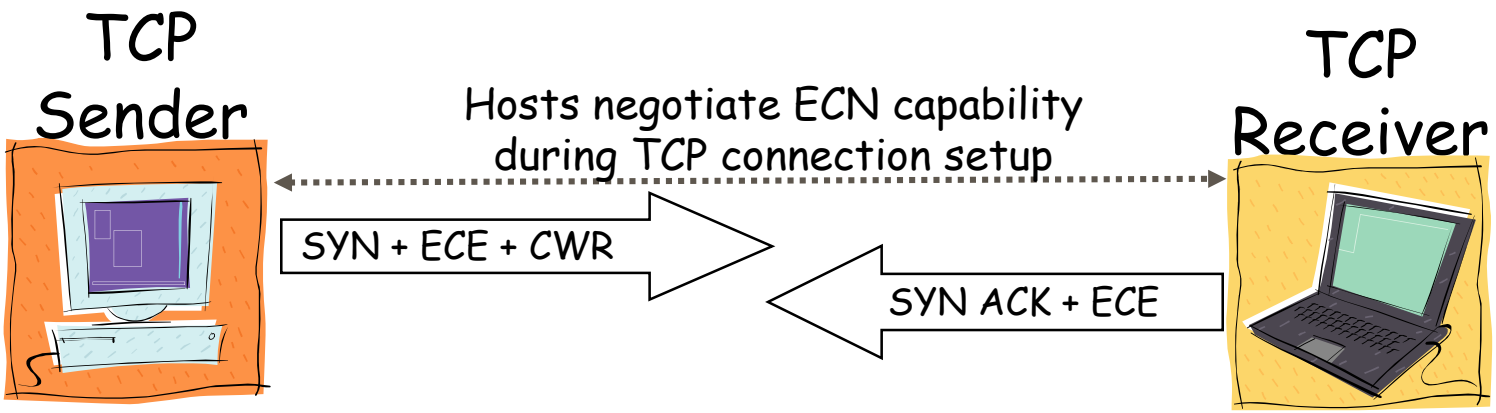




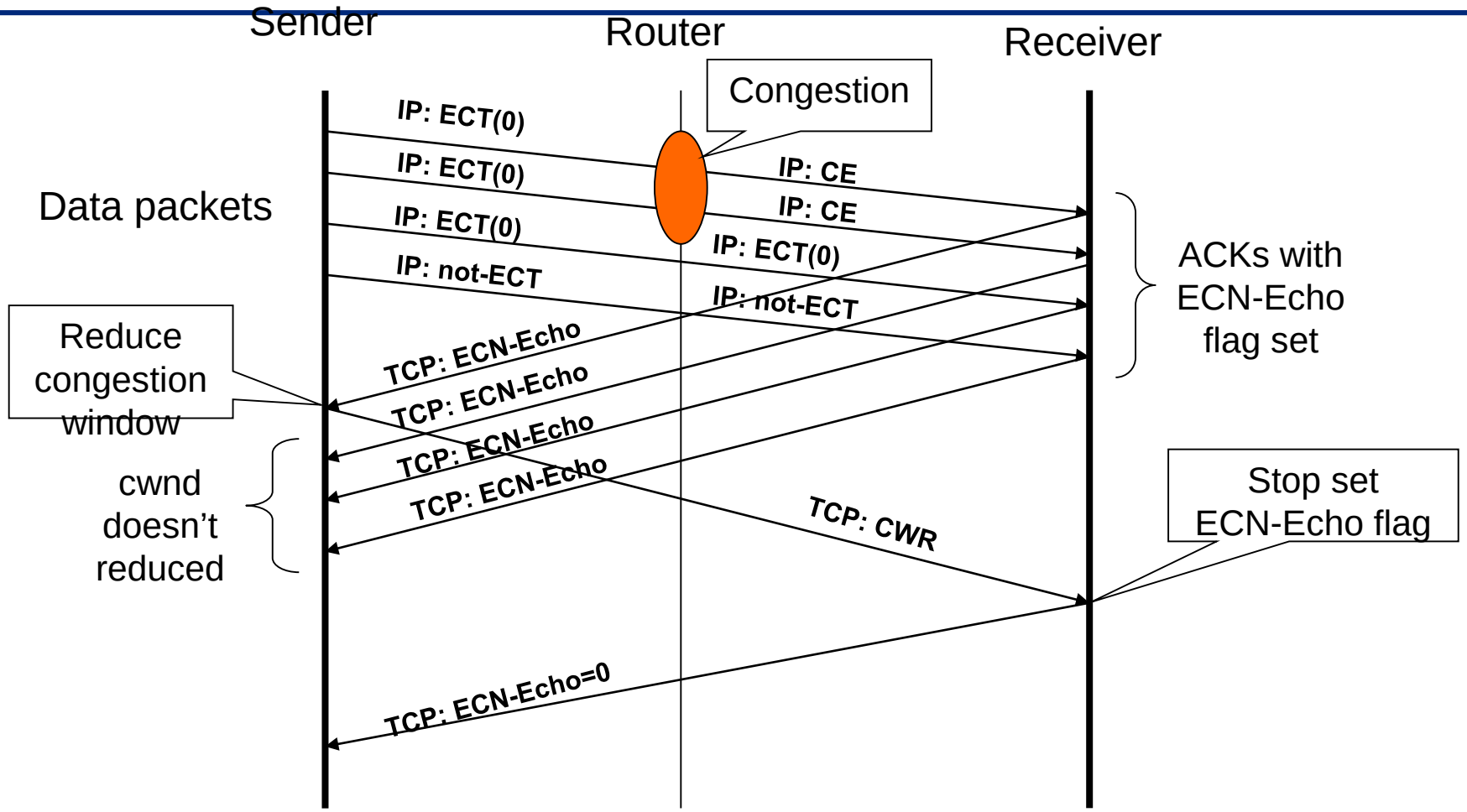
ECE flag - ECN-Echo flag

CWR flag - Congestion Window Reduced flag

Le protocole TCP (suite)



Le protocole TCP (suite)

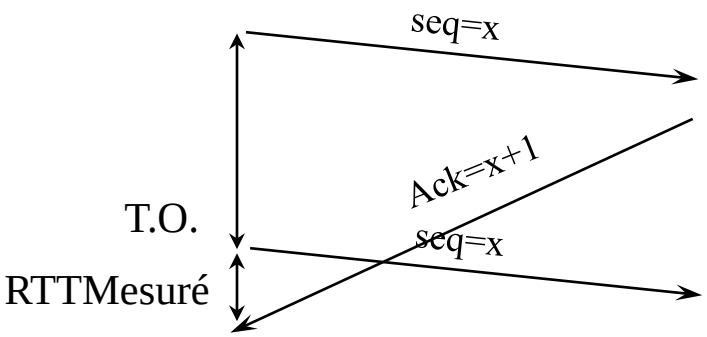


Le protocole TCP (suite)

- Les valeurs 7/8, 3/4 et 4 ont été fixées de façon empirique, ils ont en plus l'avantage de faciliter les calculs (décalage de bits)
- Dépend de la granularité de l'horloge (0,5 sec sur des machines UNIX)

•Karn & Partridge

- Problème



- Solution

Si retransmission

- Ne pas mesurer le RTT mais

- doubler le RTO à chaque échec (**exponential back-off**) jusqu 'à ce que passe le segment ou Max. 240 sec.

Continuer à utiliser pour le RTO le back-off jusqu'à ce qu'un segment non retransmis soit acquittée

Le protocole TCP (suite)

Measure	RFC 1122	TCP Tahoe	TCP Reno
RTT Variance Estimation	✓	✓	✓
Exponential RTO Backoff	✓	✓	✓
Karn's Algorithm	✓	✓	✓
Slow Start	✓	✓	✓
Dynamic Window Sizing on Congestion	✓	✓	✓
Fast Retransmit		✓	✓
Fast Recovery			✓

Le protocole TCP (suite)

- « Time Until Wrap Around »
  - Num. de séquence sur 32 bits
  - Bandwidth « Time Until Wrap Around »

Bandwidth	« Time Until Wrap Around »
T1 (1.5 Mbps)	6.4 heures
Ethernet (10 Mbps)	57 minutes
T3 (45 Mbps)	13 minutes
FDDI (100 Mbps)	6 minutes
STS-3 (155 Mbps)	4 minutes
STS-12 (622 Mbps)	55 secondes
STS-24 (1.2 Gbps)	28 secondes

  - Solution : PAWS « Protect against wrapped sequence numbers » RFC 1323) utilisé dans tcp_input()
    - Option TCP : rajout d'un TimeStamp à 32 bits avec une fréquence 1/1ms (24 jours pour changer le bit de signe)
    - Rejeter nouveau segment si un segment est plus ancien que le dernier enregistré avec un numéro de séqu. supérieur à celui enregistré (TimeStamp du seg. < à celui du dernier enregistré) et (num. seq du seg. > à celui enregistré)
    - Risque d'erreur si connexion au repos pendant plus de 24 jours.

Le protocole TCP (suite)

- Utiliser toute la bande passante
  - AdvertisedWindow sur 16 bits
  - Bandwidth — Delay (~10 ms) x Bandwidth

Bandwidth	Delay (~10 ms) x Bandwidth
T1 (1.5 Mbps)	18KB
Ethernet (10 Mbps)	122KB
T3 (45 Mbps)	549KB
FDDI (100 Mbps)	1.2MB
STS-3 (155 Mbps)	1.8MB
STS-12 (622 Mbps)	7.4MB
STS-24 (1.2 Gbps)	14.8MB

  - Solution : option «TCP window scale option » par décalage des bits du num. de séq., jusqu'à 1Go (shift.cnt)

Le protocole TCP (suite)

- S: "Maximum Normalized Throughput"

  S= 1 si W > RD / 4

  et S= 4W/RD si W < RD / 4

  avec

  W = taille de la fenêtre (en octets)

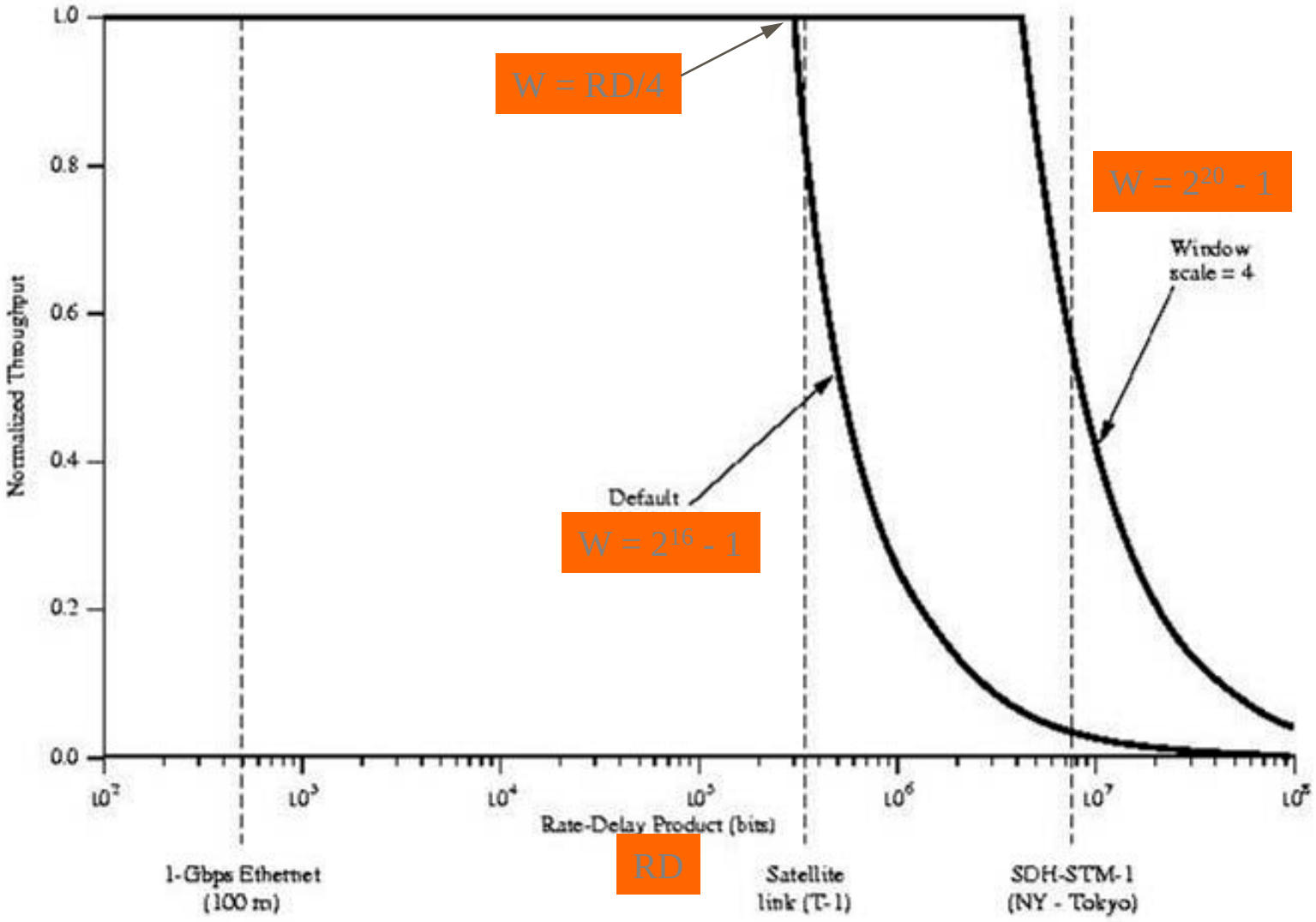
  R = vitesse de transmission (bps) à une source TCP

  D = délai de propagation (2D = RTT)

  RD (bits) est connu comme "rate-delay product".

  - Remarque : TCP peut émettre au max. 2RD bits soit RD/4 octets

Influence du "Window Scale Parameter"



Les protocoles de transport d'Internet

- Services / Caractéristiques de TCP
  - Orienté connexion, point à point, bidirectionnel, fiable
  - Flot d'octets, l'ordre d'envoi est conservé
    - l'entité TCP décide d'elle même du découpage du flot en segments
    - utiliser le bit PUSH pour forcer l'envoi de données
    - le bit URGENT provoque une interruption chez l'application destination pour traiter les données « urgentes »
  - Evitement de congestion

- Services / Caractéristiques de UDP
  - datagramme
  - Simple avec le moins de surcoût : essetiellement UDP rajoute à IP l'information de service (port)

Le protocole SCTP

- Pour remédier aux limites
  - de TCP
    - Problème du HOL (Head Of Line blocking)
    - L'incapacité de supporter le multihoming
    - Problème de l'attaque SYN flooding (DoS)
  - et de UDP
    - Non fiable
    - Aucun contrôle de flux/congestion
- Stratégies de controle de flux et de Congestion similaire à TCP
  - Slow Start et Congestion Avoidance
  - Selective Acknowledgement et Fast Retransmit
  - Légères différences pour supporter le multistreaming et le multihoming
-

Les protocoles de transport d'Internet

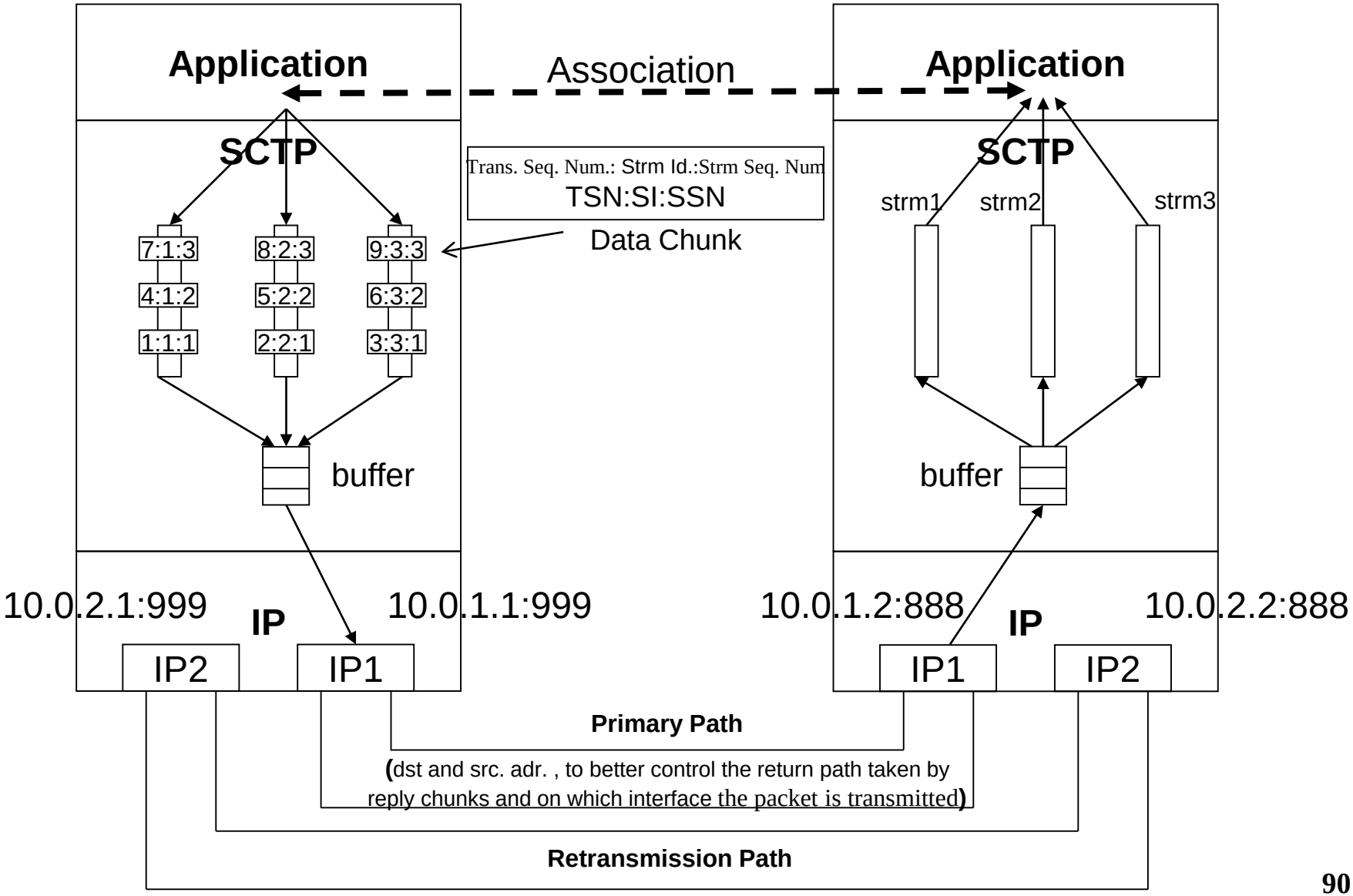
- Services / Caractéristiques de SCTP
  - orienté connexion, point à point, bidirectionnel
  - fiable, grâce à un mécanisme d'acquittement indépendant du séquencement (acquittement selective)
  - Une meilleure tolérance aux pannes des réseaux (multi-homing)
  - orienté message (et non octet)
  - livraison
    - en séquence (ordonnée) d'un flot (stream) de messages, avec la possibilité de définir plusieurs streams (multistreaming) + fragmentation/stream
    - possibilité de livrer des messages de manière non ordonnée
  - supporte le multi-homing : une extrémité logique (endpoint) associée à plusieurs adresses IP
  - Mécanismes de sécurité : contre DoS(SYN-flood) et Masquerade
  - Réalise aussi l'évitement de congestion

Comparaison

Services/Features	SCTP	TCP	UDP
Full-duplex data transmission	yes	yes	yes
Connection-oriented	yes	yes	no
Reliable data transfer	yes	yes	no
Partially reliable data transfer	optional	no	no
Ordered data delivery	yes	yes	no
Unordered data delivery	yes	no	yes
Flow and congestion control	yes	yes	no
Explicit congestion notification support	yes	yes	no
Selective acks	yes	optional	no
Preservation of message boundaries	yes	no	yes
Path maximum transmission unit discovery	yes	yes	no
Application data fragmentation/bundling	yes	yes	no
Multistreaming	yes	no	no
Multihoming	yes	no	no
Protection against SYN flooding attack	yes	no	n/a
Half-closed connections	no	yes	n/a

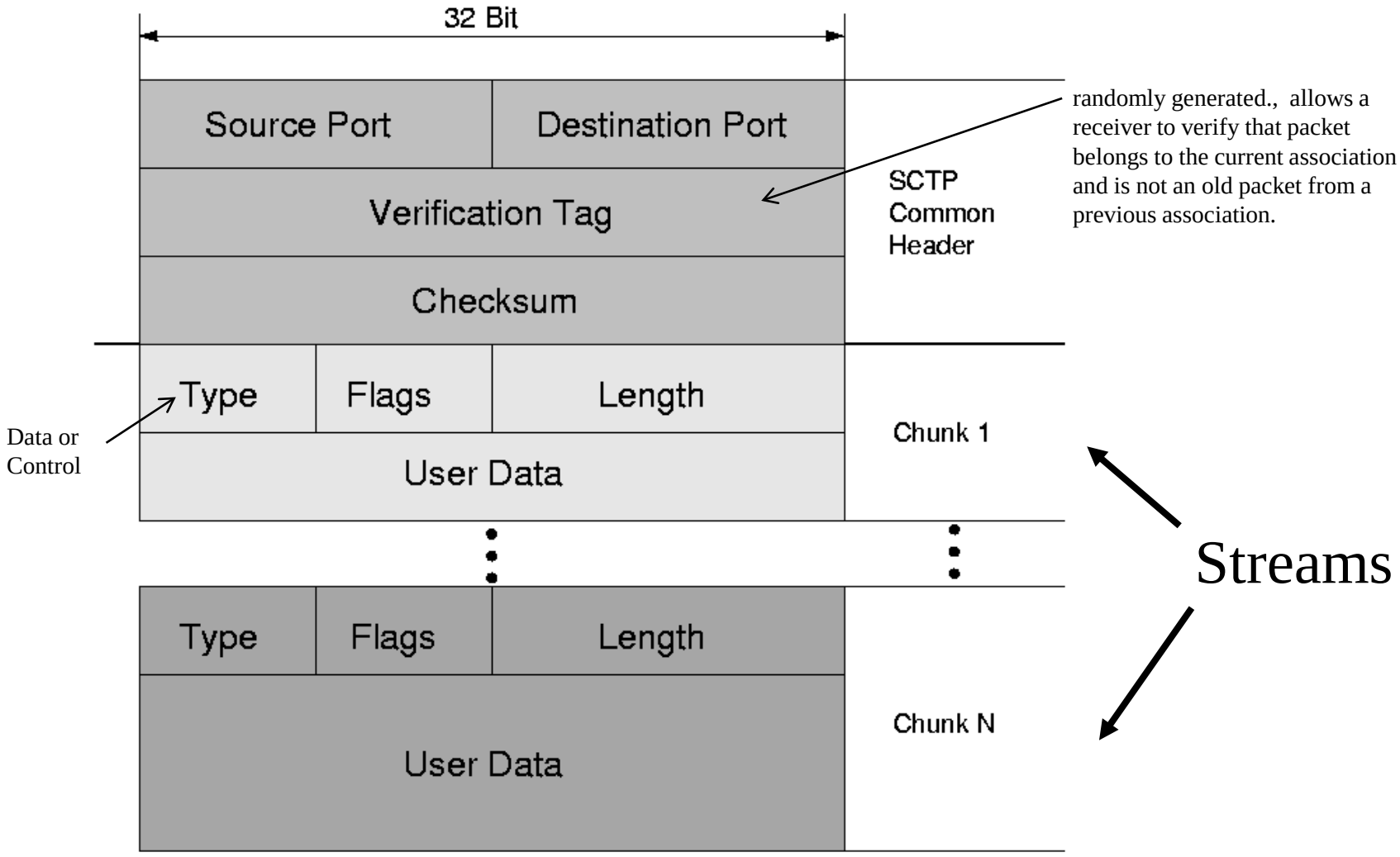
Le protocole SCTP

(



Le protocole SCTP

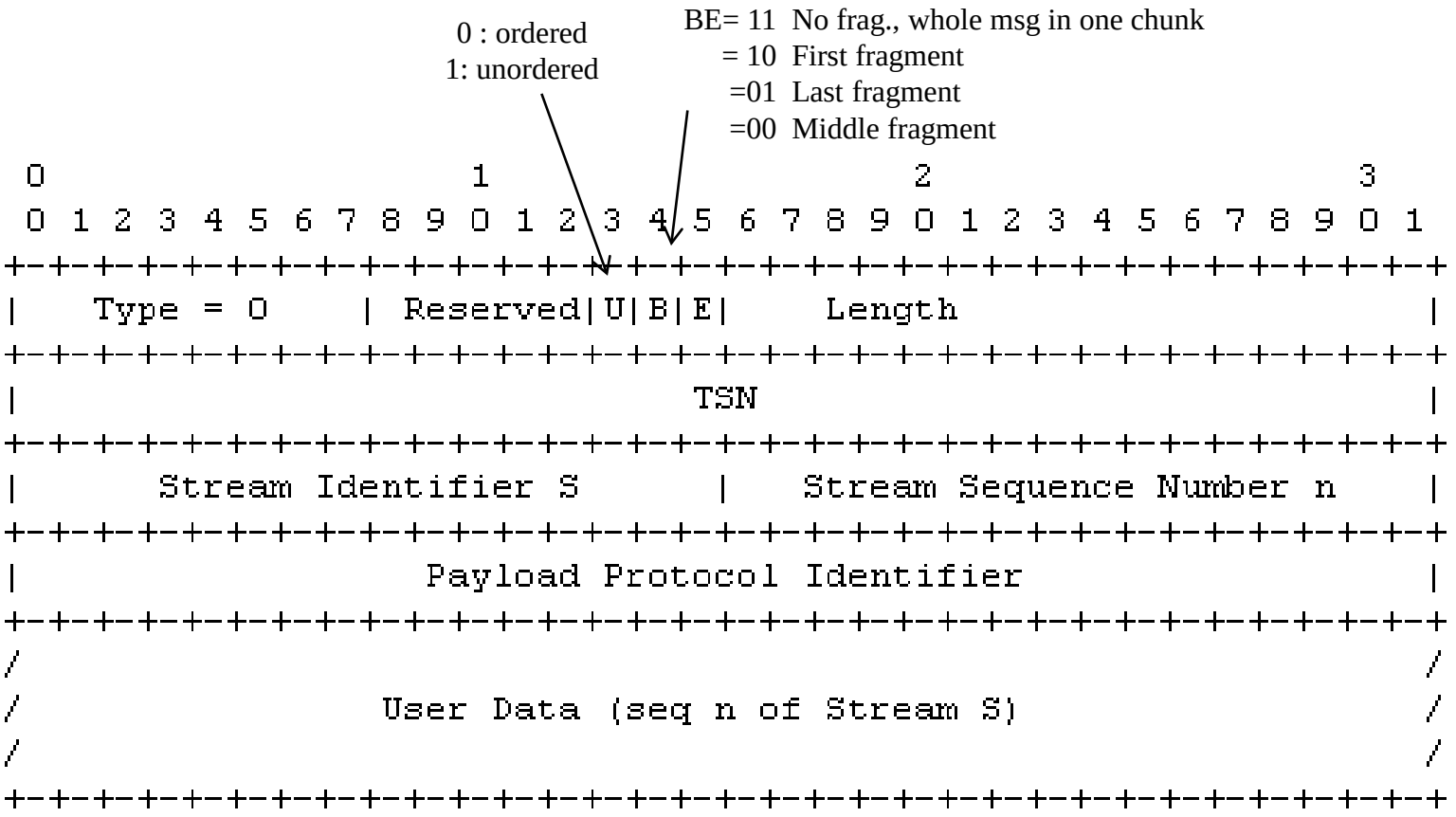
Format d'une SCTP-PDU



Le protocole SCTP

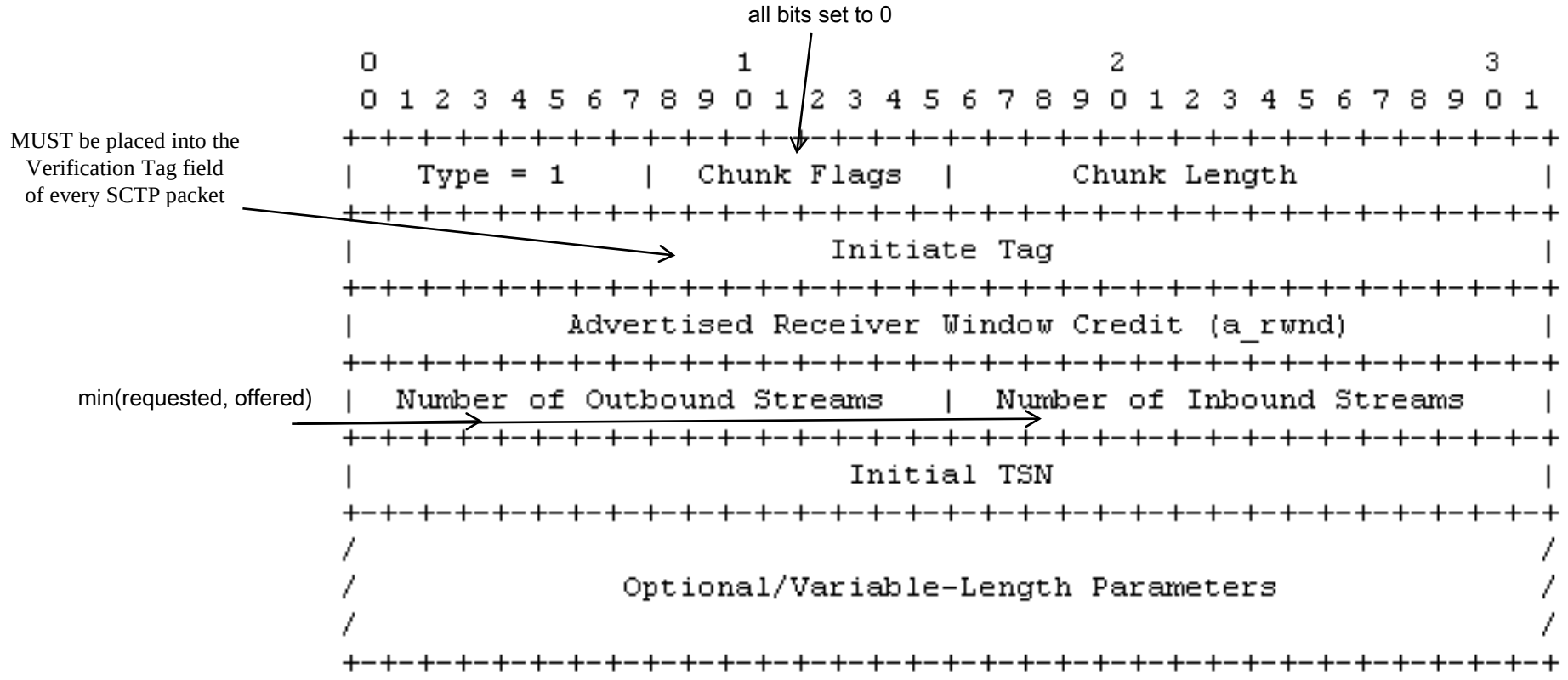
ID Value	Chunk Type
0	- Payload Data (DATA)
1	- Initiation (INIT)
2	- Initiation Acknowledgement (INIT ACK)
3	- Selective Acknowledgement (SACK)
4	- Heartbeat Request (HEARTBEAT)
5	- Heartbeat Acknowledgement (HEARTBEAT ACK)
6	- Abort (ABORT)
7	- Shutdown (SHUTDOWN)
8	- Shutdown Acknowledgement (SHUTDOWN ACK)
9	- Operation Error (ERROR)
10	- State Cookie (COOKIE ECHO)
11	- Cookie Acknowledgement (COOKIE ACK)
12	- Reserved for Explicit Congestion Notification Echo (ECNE)
13	- Reserved for Congestion Window Reduced (CWR)
14	- Shutdown Complete (SHUTDOWN COMPLETE)
15 to 62	- reserved by IETF
63	- IETF-defined Chunk Extensions
64 to 126	- reserved by IETF
127	- IETF-defined Chunk Extensions
128 to 190	- reserved by IETF
191	- IETF-defined Chunk Extensions
192 to 254	- reserved by IETF
255	- IETF-defined Chunk Extensions

Le protocole SCTP



A DATA chunk cannot carry data belonging to more than one message,
but a message can be split into several chunks.

Le protocole SCTP

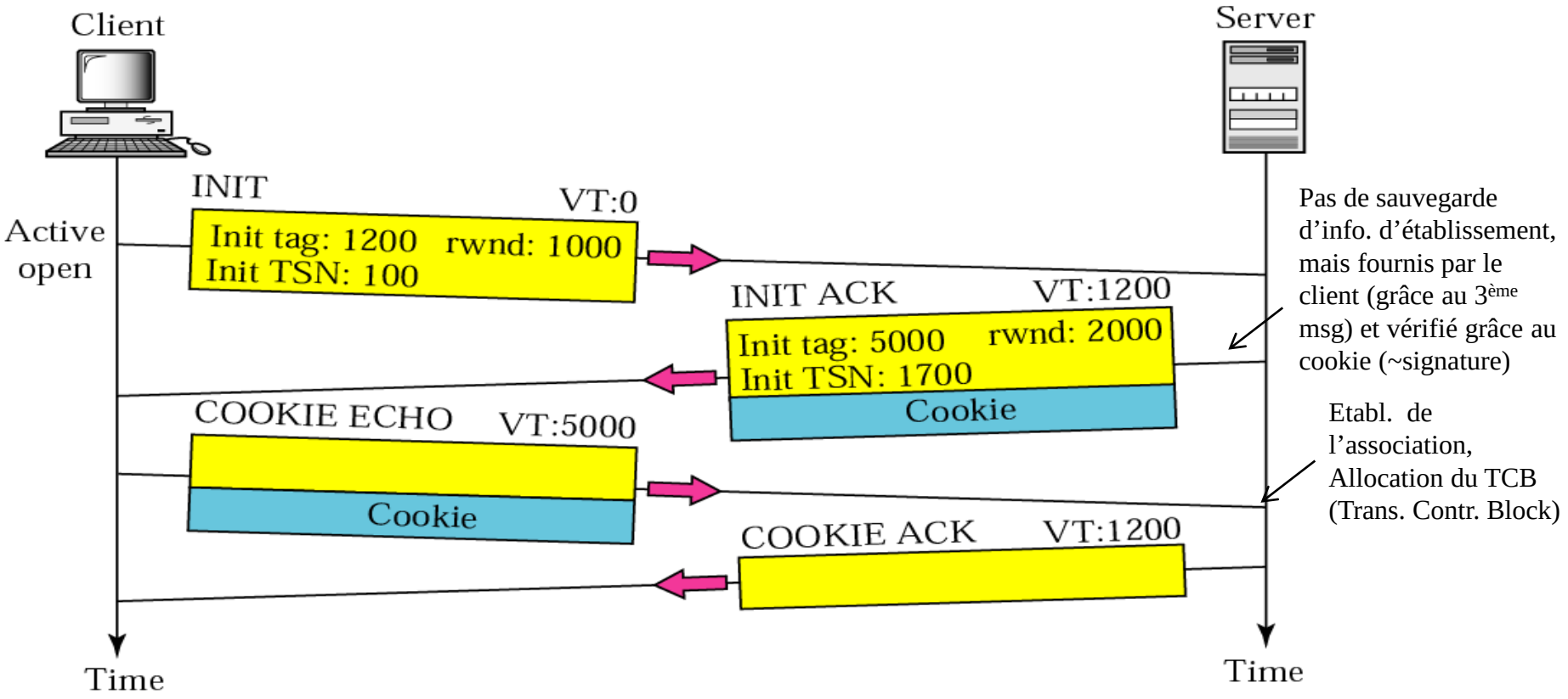


- INIT ACK is formatted similarly to the INIT chunk

Le protocole SCTP

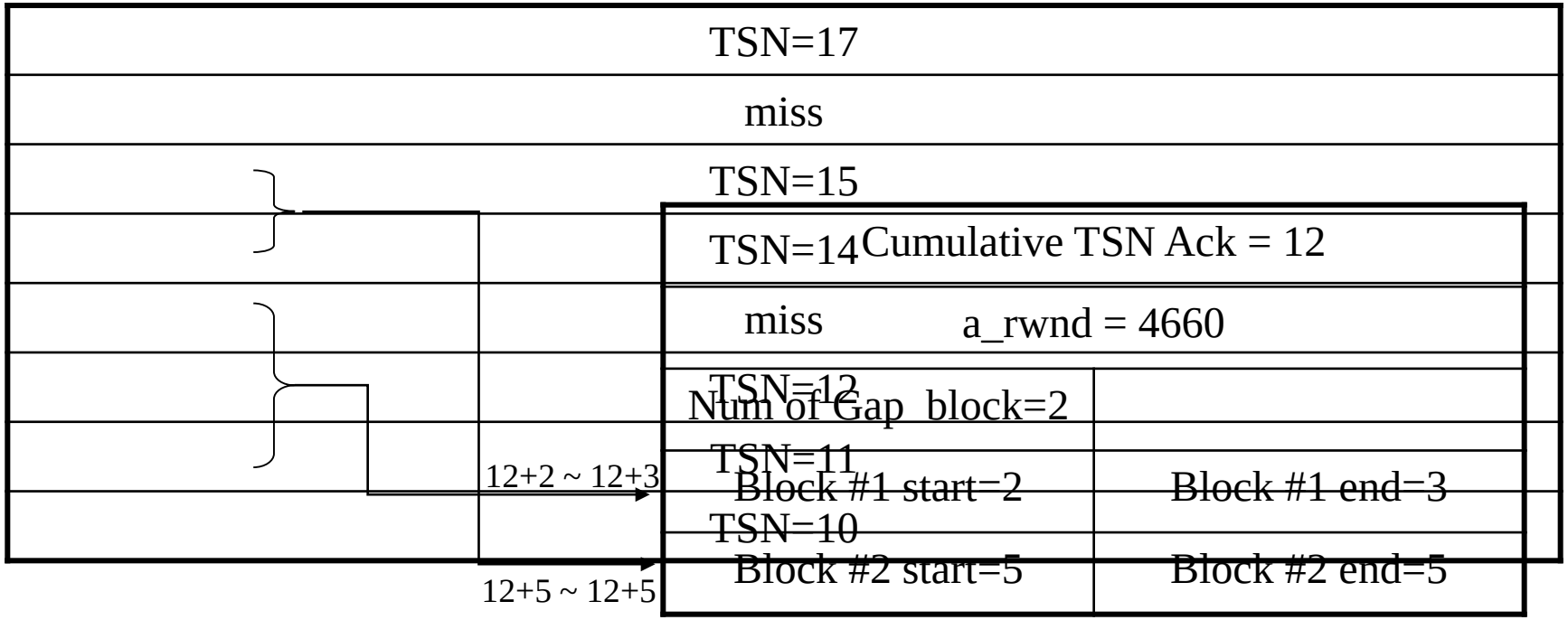
- Four-way handshaking

  Utile contre les attaques de type SYN flooding (DoS)



Le protocole SCTP

- Selective Acknowledgement (SACK)



Fin du chapitre 3