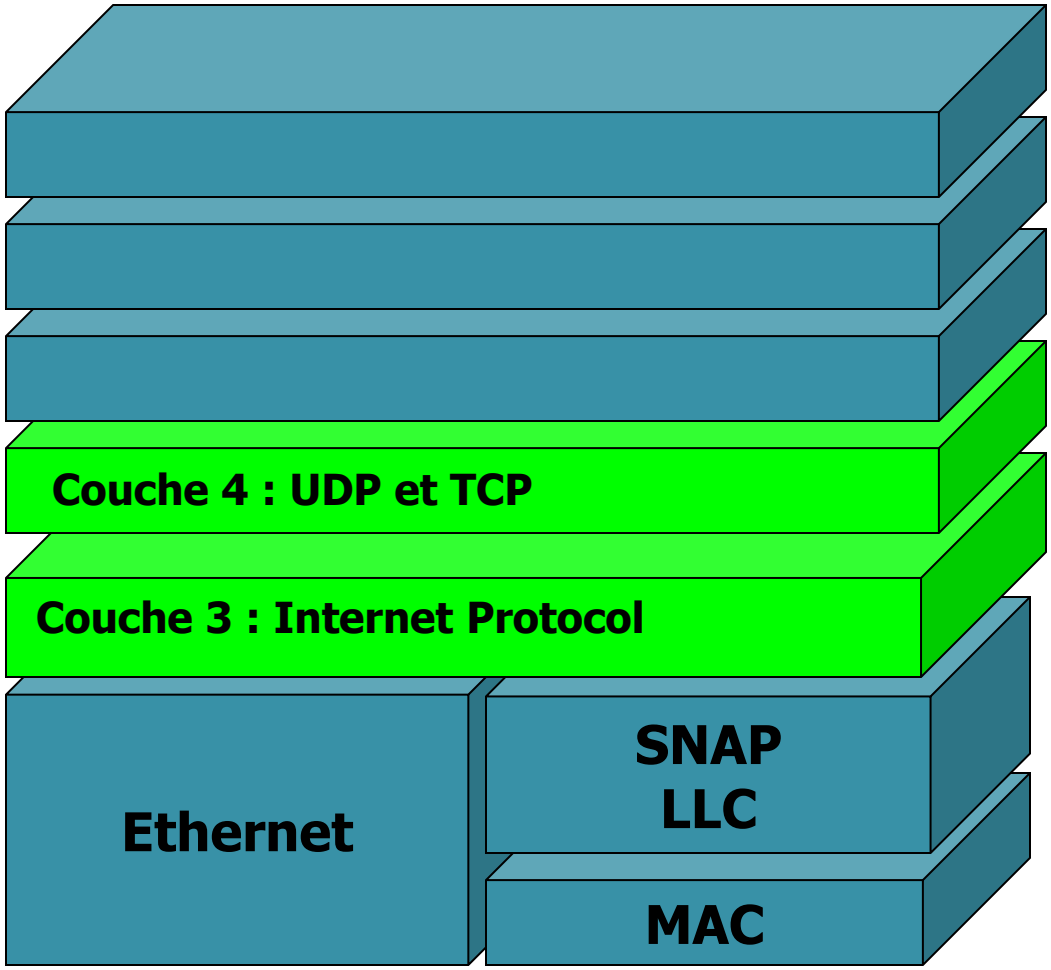
Introduction aux Réseau : UDP et TCP

- IPv6
- UDP
- TCP



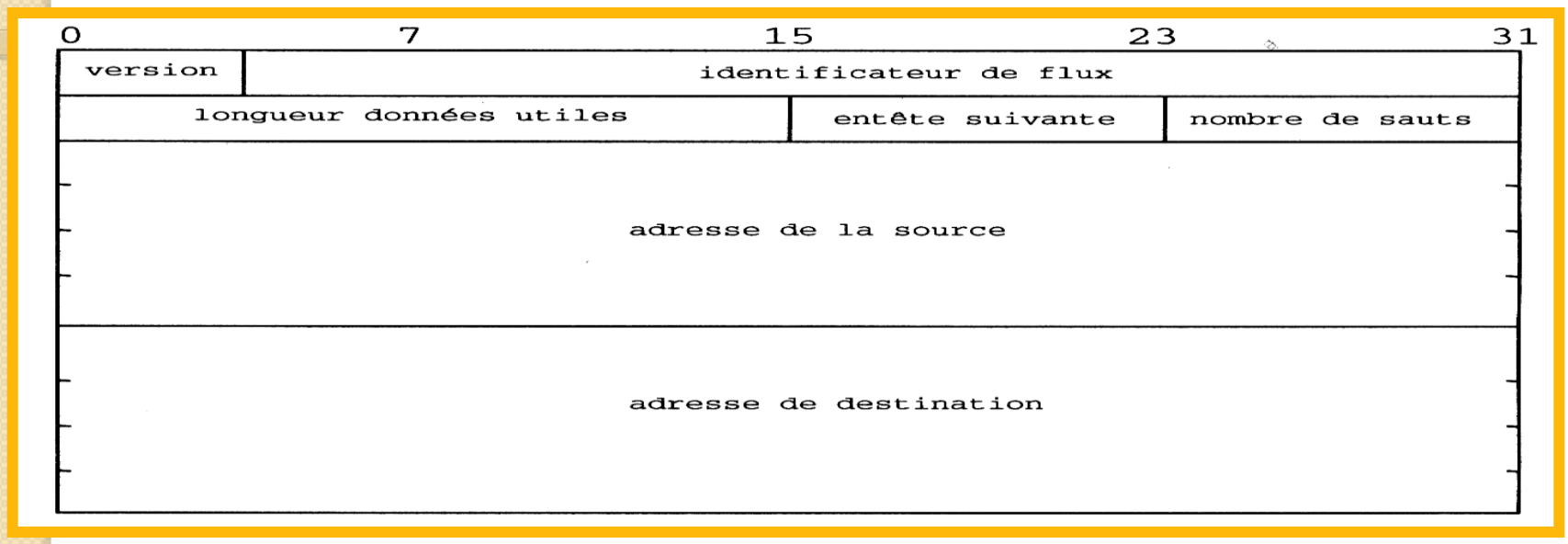
7.7 IPv6

- Nouvelle version du protocole IP (version 6)
- Objectif : augmenter le nombre de machines adressables par IPv4 (adresses 32 bits)

- Adresses IPv6 sur 128 bits (16 octets)

- D 'autres améliorations sont apparues entre IPv4 et IPv6 pour augmenter les performances du routage
  - Plus de champ checksum dans l 'entête
  - Taille des entêtes fixe
  - Options supprimées de l 'entête et transformées en nouvelles en-têtes (facilement ignorables par les routeurs intermédiaires)

7.7.1 Format d 'un paquet IPv6



7.7.2 Différents Champs

- Version
  - valeur 6
- Identificateur de Flux
  - priorité et numéro aléatoire (qualité de service)
- Longueur des données utiles (payload)
  - au-dessus de 64 Ko une option est utilisée
- En-tête suivante
  - similaire au champ protocole de IPv4
  - il peut s 'agir d 'un protocole niveau supérieur mais aussi d 'une option
- Nombre de sauts
  - similaire au champ TTL de IPv4
- Adresses source/destination

7.7.3 Adresses IPv6

type d'adresse	préfixe binaire	fraction de l'espace d'adressage
Réservé sauf Adresse IP v4	0000 0000 0000 0000 0000 0000 0000 0000	1/256
Allouée pour NSAP (CLNP)	0000 001	1/128
Allouée pour IPX	0000 010	1/128
Adresse point-à-point allouée aux opérateurs (fournisseur de service)	010	1/8
Adresses multicast locales	1111 1110	1/256
Adresses de multicast	1111 1111	1/256

8 TCP ET UDP

- 10.1 Introduction
- 10.2 UDP : User Datagramm Protocol
- 10.3 TCP : Transmission Control Protocol

8.1 Introduction

- 10.1.1 Notion de Port
- 10.1.2 Le protocole UDP, RFC 768
- 10.1.3 Le protocole TCP, RFC 793

8.1.1 Notion de Port

- Adressage IP permet l 'identification d 'une machine du réseau IP

- Pour identifier l 'application de la machine qui doit traiter les données nous utiliserons la notion de port et de numéro de port

  - Il existe des ports bien fixés et bien connus (applications réparties connues de tous les systèmes) RFC 1700 (de 0 à 1023)
  - Les numéros > 1024, peuvent être attribués à n 'importe qu 'elle application
  - voir fichier /etc/services d 'UNIX

8.1.1 Notion de Port

- Une association entre deux applications peut alors être définies par :
  - adresse IP source
  - adresse IP destination
  - port source
  - port destination
  - type du protocole
- Les processus disposent d'une interface système leur permettant de spécifier un port ou d'y accéder (socket, ...).
- Les accès aux ports sont généralement synchrones, les opérations sur les ports sont tamponnées (files d'attente).

8.1.2 Le protocole UDP, RFC 768

- User Datagramm Protocol
- RFC 768

- permet à une application d 'envoyer des messages vers une autre application avec un minimum de fonctionnalités :
  - pas de garantie d 'arrivée
  - pas de contrôle de séquencement

8.1.3 Le protocole TCP, RFC 793

- Transmission Control Protocol
- RFC 793
- protocole de niveau 4
- transfert fiable sur connexion entre deux stations
  - contrôle des données transférées
  - reséquencement
  - contrôle de flux
  - niveau de priorités

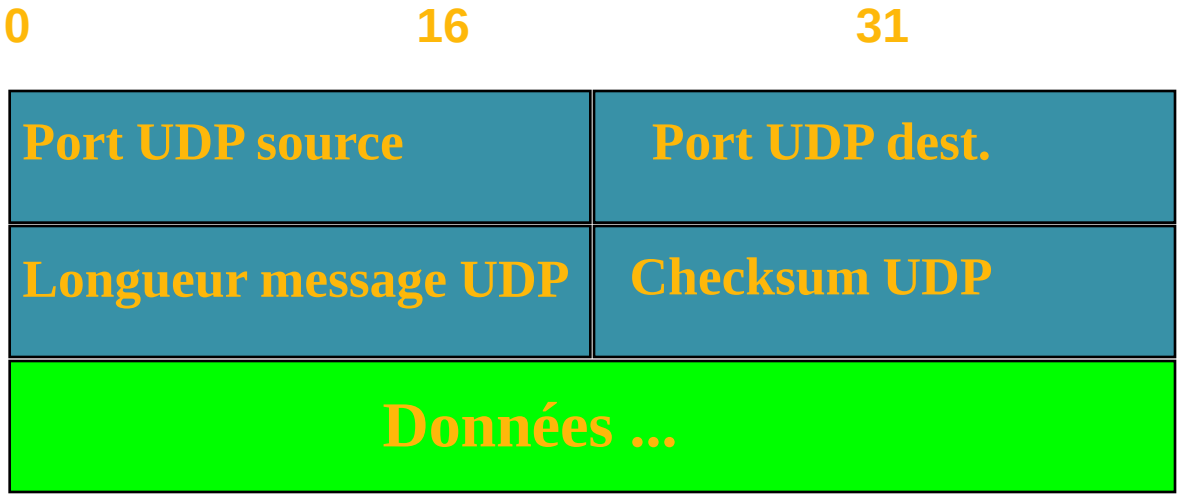
8.2 UDP : User Datagramm Protocol

- 10.2.1 UDP : format des messages
- 10.2.2 UDP : pseudo en-tête
- 10.2.3 UDP : Multiplexage
- 10.2.4 UDP : les ports standards
- 10.2.5 UDP : Exemple

8.2.1 UDP : FORMAT DES MESSAGES

- Les messages UDP sont également appelés des datagrammes UDP.
- Ils contiennent deux parties :
  - une en-tête UDP et
  - les données UDP.

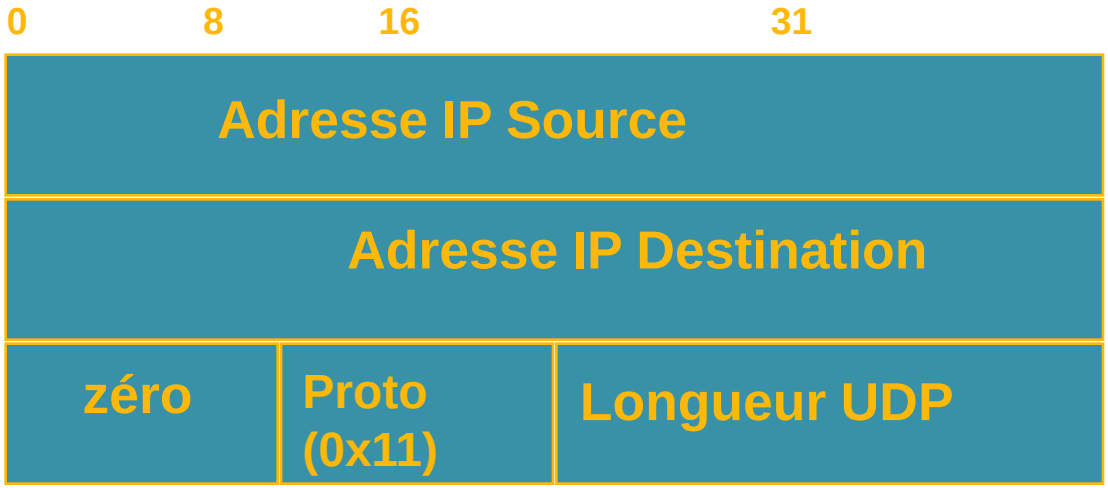
- Format des messages UDP



8.2.2 UDP : PSEUDO EN-TÊTE

- Lorsqu'il est utilisé (IPv6), le champ checksum couvre plus d'informations que celles contenue dans le datagramme UDP; En effet, le checksum est calculé avec une pseudo-en-tête non transmise dans le datagramme.

- Format de la pseudo en-tête



8.2.3 UDP : MULTIPLEXAGE

- UDP multiplexe et démultiplexe les datagrammes en sélectionnant les numéros de ports :
  - une application obtient un numéro de port de la machine locale; dès lors que l'application émet un message via ce port, le champ PORT SOURCE du datagramme UDP contient ce numéro de port,
  - une application connaît (ou obtient) un numéro de port distant afin de communiquer avec le service désiré.

- Lorsque UDP reçoit un datagramme, il vérifie que celui-ci est un des ports actuellement actifs (associé à une application) et le délivre à l'application responsable (mise en queue)
  - si ce n'est pas le cas, il émet un message ICMP port unreachable, et détruit le datagramme.

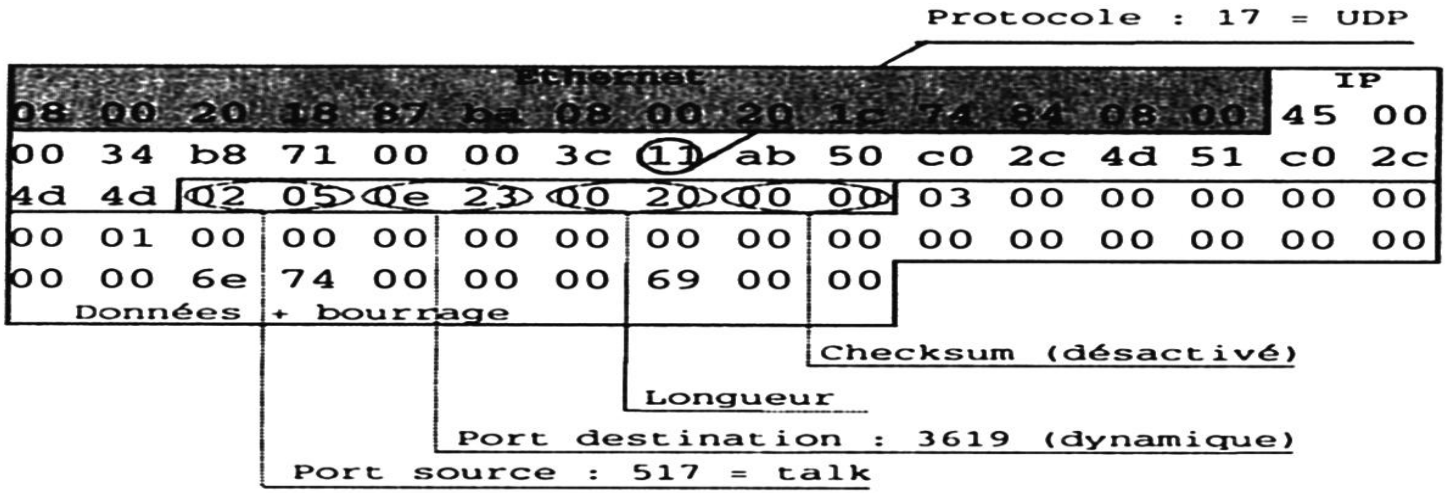
8.2.4 UDP : LES PORTS STANDARDS

Certains ports sont réservés (well-kown port assignements) :

No port	Mot-clé	Description
7	ECHO	Echo
11	USERS	Active Users
13	DAYTIME	Daytime
37	TIME	Time
42	NAMESERVER	Host Name Server
53	DOMAIN	Domain Name Server
67	BOOTPS	Boot protocol server
68	BOOTPC	Boot protocol client
69	TFTP	Trivial File transfert protocol
123	NTP	Network Time Protocol
161	SNMP	Simple Network Management prot.

D'autres numéros de port (non réservés) peuvent être assignés dynamiquement aux applications.

8.2.5 UDP : EXEMPLE



8.3 TCP : Transmission Control Protocol

- transport fiable de la technologie TCP/IP.
  - fiabilité = illusion assurée par le service
  - transferts tamponnés : découpage en segments
  - connexions bidirectionnelles et simultanées
- service en mode connecté
- garantie de non perte de messages ainsi que de l'ordonnancement

8.3.1 TCP : La connexion

- une connexion de type circuit virtuel est établie avant que les données ne soient échangées : appel + négociation + transferts
- Une connexion = une paire d'extrémités de connexion
- Une extrémité de connexion = couple (adresse IP, port)
- Exemple de connexion : ((124.32.12.1, 1034), (19.24.67.2, 21))
- Une extrémité de connexion peut être partagée par plusieurs autres extrémités de connexions (multi-instanciations)

8.3.1 TCP : La connexion

- La mise en œuvre de la connexion se fait en deux étapes :
  - une application (extrémité) effectue une ouverture passive en indiquant qu'elle accepte une connexion entrante,
  - une autre application (extrémité) effectue une ouverture active pour demander l'établissement de la connexion.

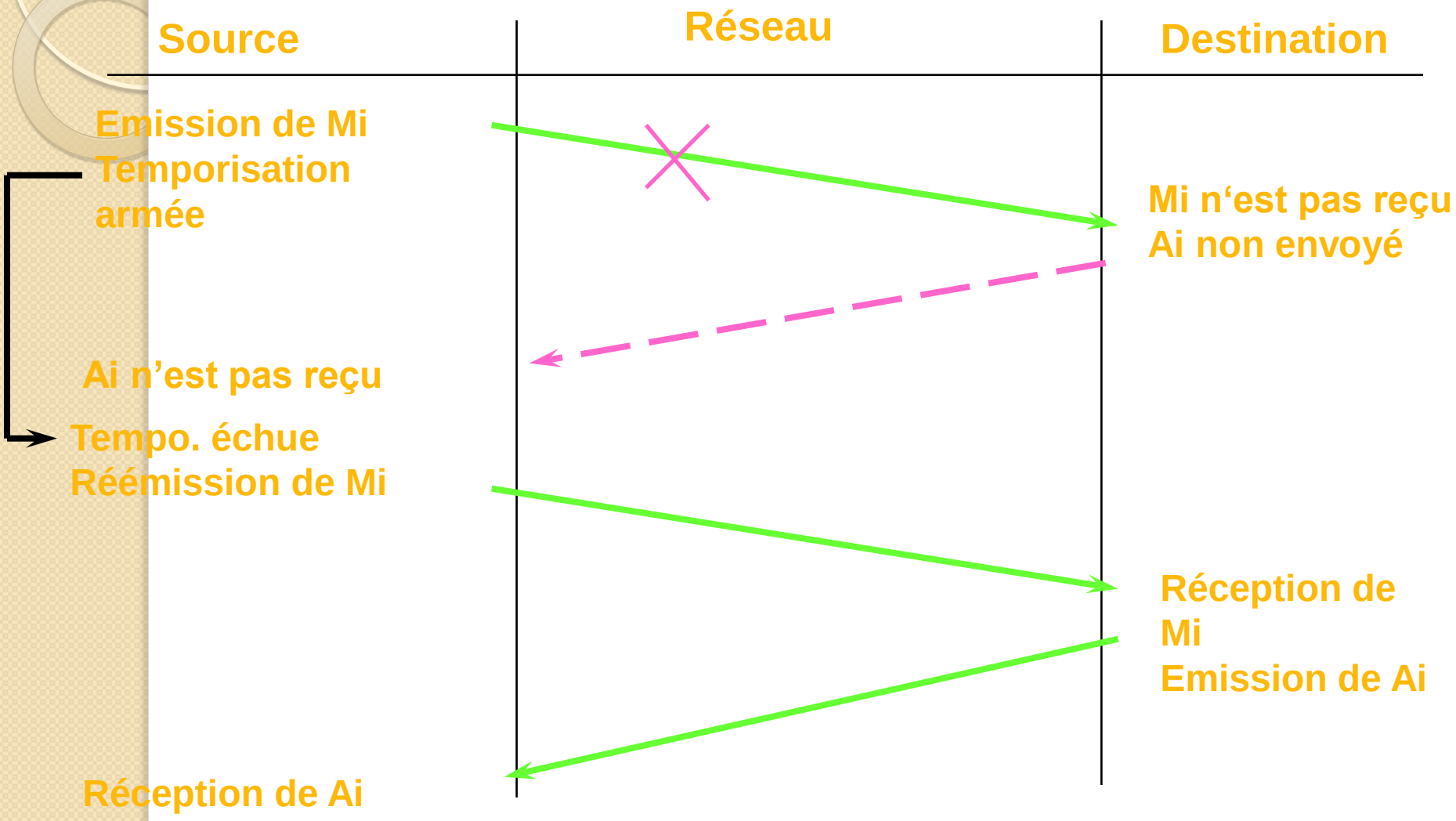
8.3.2 TCP : Segmentation

- Segmentation, contrôle de flux
  - Les données transmises à TCP constituent un flot d'octets de longueur variable.
  - TCP divise ce flot de données en segments en utilisant un mécanisme de fenêtrage.
  - Un segment est émis dans un datagramme IP.

8.3.3 TCP : Acquittements

- Acquittement de messages
  - Contrairement à UDP, TCP garantit l'arrivée des messages, c'est à dire qu'en cas de perte, les deux extrémités sont prévenues.
  - Ce concept repose sur les techniques d'acquittement de message : lorsqu'une source S émet un message Mi vers une destination D, S attend un acquittement Ai de D avant d'émettre le message suivant Mi+1.
  - Si l'acquittement Ai ne parvient pas à S, S considère au bout d'un certain temps que le message est perdu et reémet Mi

8.3.3 TCP : ACQUITTEMENTS

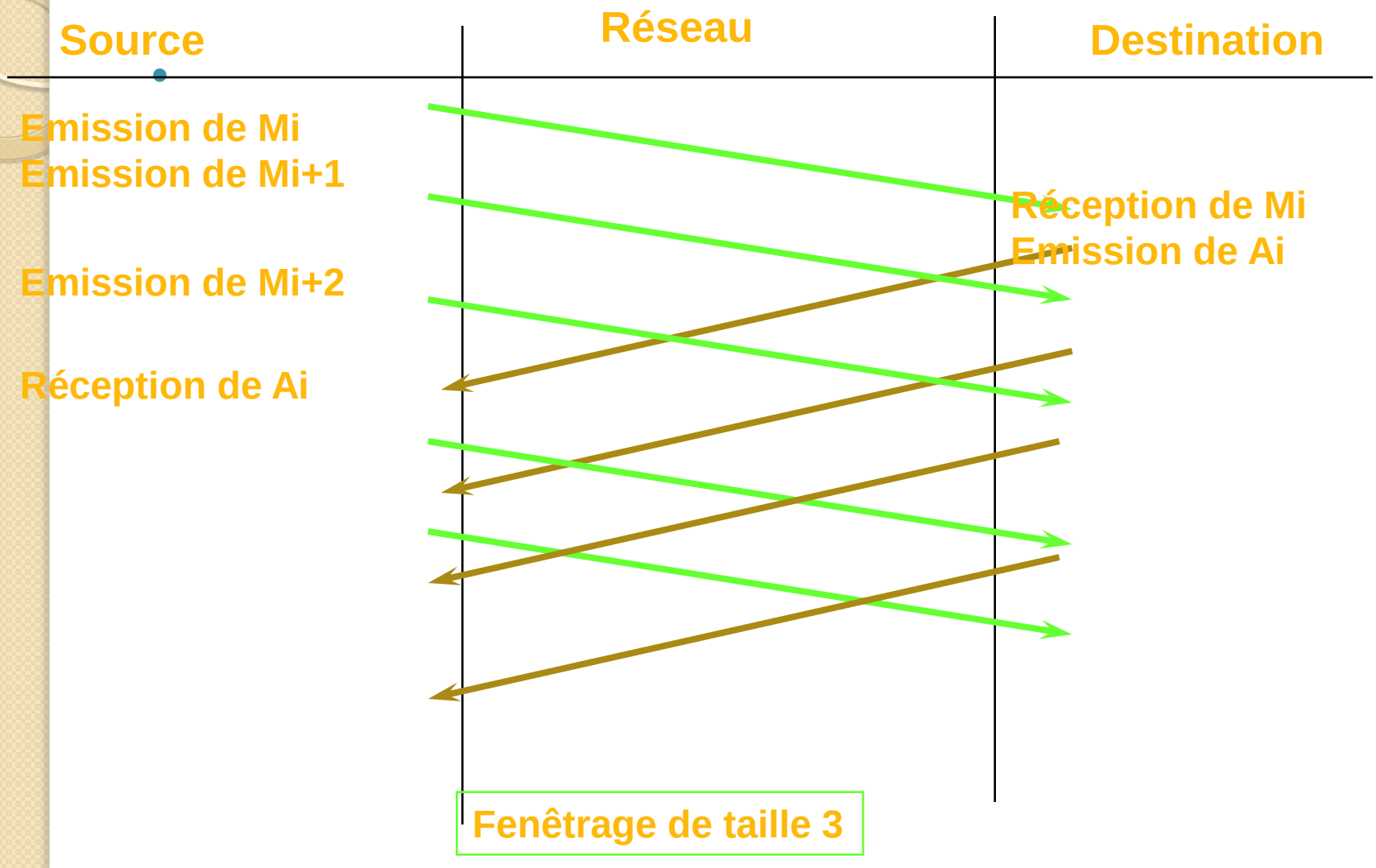


8.3.4 TCP : LE FENÊTRAGE

- La technique acquittement simple pénalise les performances puisqu'il faut attendre un acquittement avant d'émettre un nouveau message. Le fenêtrage améliore le rendement des réseaux.

- La technique du fenêtrage : une fenêtre de taille T, permet l'émission d'au plus T messages "non acquittés" avant de ne plus pouvoir émettre.

8.3.4 TCP : LE FENÊTRAGE

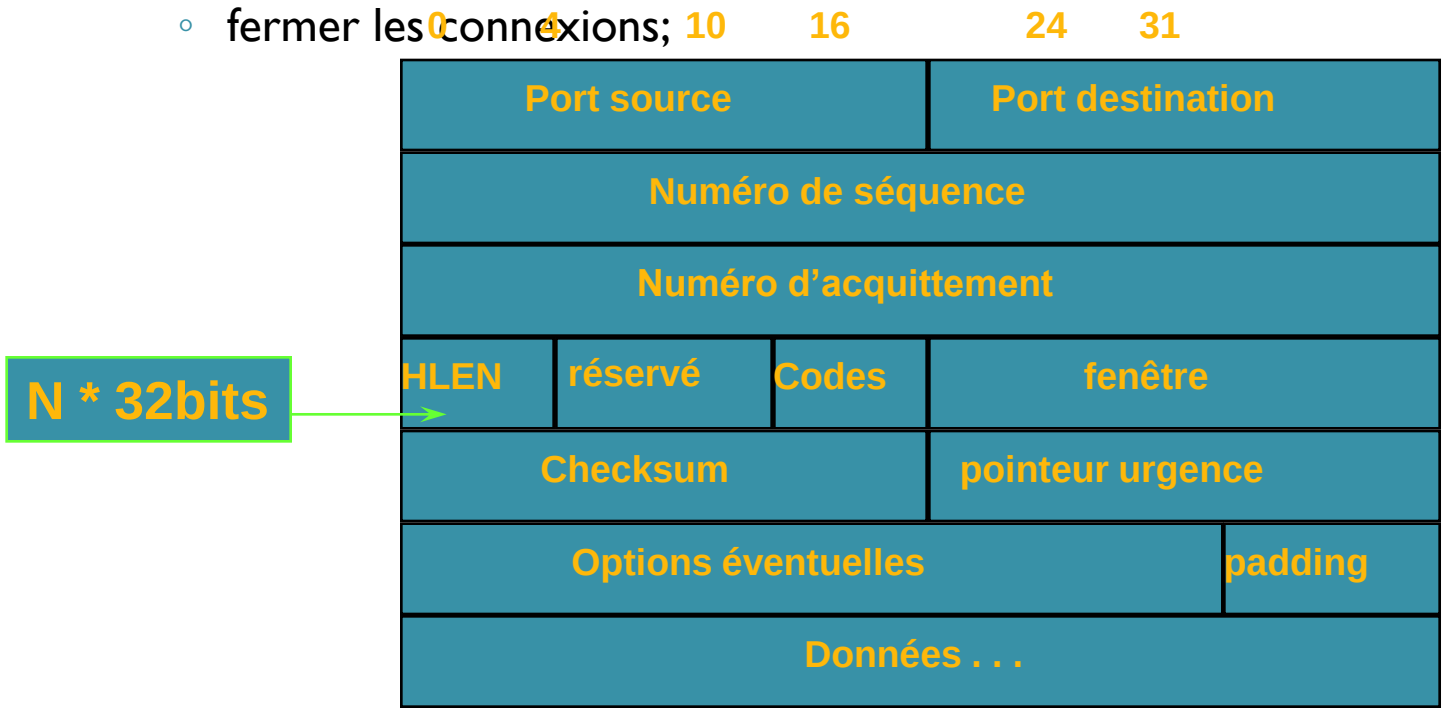


8.3.4 TCP : Technique de fenêtrage

- fenêtrage glissante permettant d'optimiser la bande passante
- permet également au destinataire de faire diminuer le débit de l'émetteur donc de gérer le contrôle de flux.
- Le mécanisme de fenêtrage mis en œuvre dans TCP opère au niveau de l'octet et non pas au niveau du segment; il repose sur :
  - la numérotation séquentielle des octets de données,
  - la gestion de trois pointeurs par fenêtrage :

8.3.5 TCP : SEGMENTS

- Segment : unité de transfert du protocole TCP.
  - échangés pour établir les connexions,
  - transférer les données,
  - émettre des acquittements,
  - fermer les connexions;



8.3.5.1 TCP : FORMAT DU SEGMENT

- Numéro de séquence : le numéro de séquence du premier octet (NSP) de ce segment.
- Généralement à la suite d'octets O1, O2, ..., On (données du message) est associée la suite de numéro de séquence NSP, NSP+1, ..., NSP+n.

- Il existe deux exceptions à cette règle :
  - lorsque le bit SYN (voir CODE BITS) est positionné, le NSP représente cette donnée de contrôle et par conséquent la suite NSP, NSP+1, NSP+2, ..., NSP+n+1, associe la suite de données SYN, O1, O2, ..., On.
  - lorsque le bit FIN (voir CODE BITS) est positionné, le NSP+n représente cette donnée de contrôle et par conséquent la suite NSP, NSP+1, NSP+2, ..., NSP+n, associe la suite de données O1, O2, ..., On, FIN.

8.3.5.1 TCP : FORMAT DU SEGMENT

- Numéro d'acquittement : le prochain numéro de séquence NS attendu par l'émetteur de cet acquittement. Acquitte implicitement les octets NS-1, NS-2, etc.

- Fenêtre: la quantité de données que l'émetteur de ce segment est capable de recevoir; ceci est mentionné dans chaque segment (données ou acquittement).

8.3.5.2 TCP : Format du segment, code bits

- CODE BITS : indique la nature du segment :
  - URG : le pointeur de données urgentes est valide (exemple : interrupt en remote login), les données sont émises sans délai, les données reçues sont remises sans délai.
  - SYN : utilisé à l'initialisation de la connexion pour indiquer où la numérotation séquentielle commence.
    - Syn occupe lui-même un numéro de séquence bien que ne figurant pas dans le champ de données. Le Numéro de séquence inscrit dans le datagramme (correspondant à SYN) est alors un Initial Sequence Number (ISN) produit par un générateur garantissant l'unicité de l'ISN sur le réseau (indispensable pour identifier les duplications).

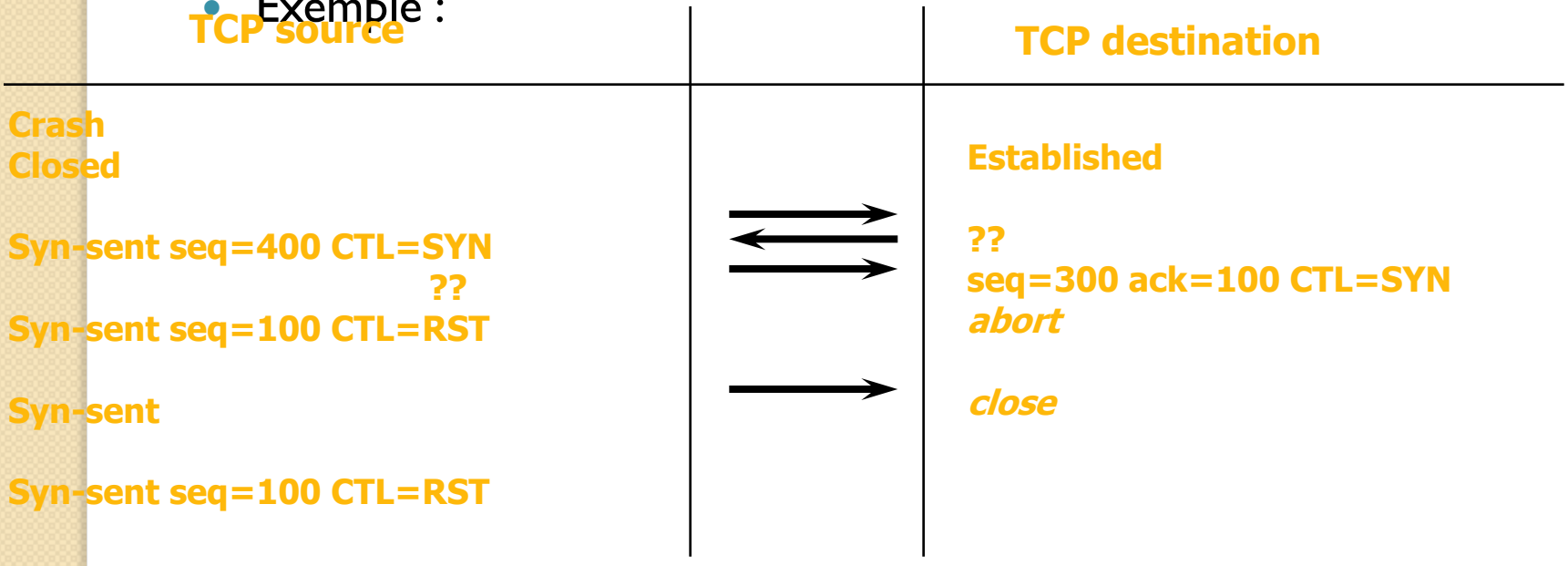
8.3.5.2 TCP : Format du segment, code bits

- FIN : utilisé lors de la libération de la connexion;
- PSH : fonction push.
  - Normalement, en émission, TCP reçoit les données depuis l'applicatif , les transforme en segments à sa guise puis transfère les segments sur le réseau;
  - un récepteur TCP décodant le bit PSH, transmet à l'application réceptrice, les données correspondantes sans attendre plus de données de l'émetteur. Exemple : émulation terminal, pour envoyer chaque caractère entré au clavier (mode caractère asynchrone).

8.3.5.2 TCP : FORMAT DU SEGMENT, CODE BITS

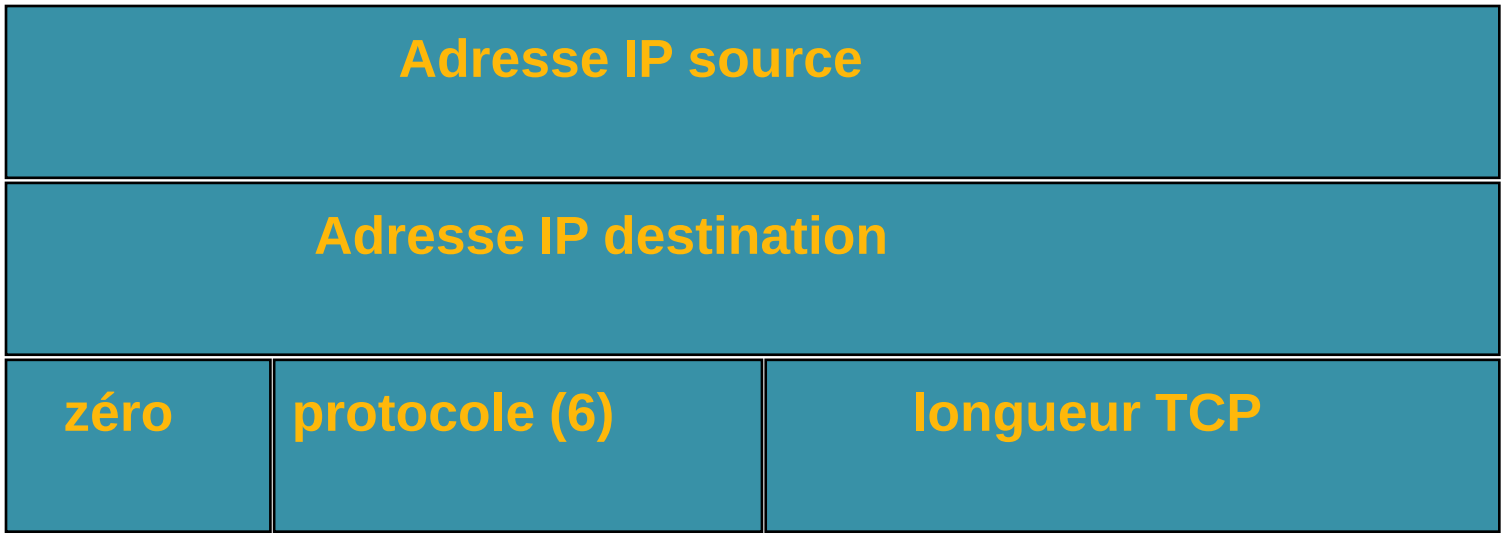
- RST : utilisé par une extrémité pour indiquer à l'autre extrémité qu'elle doit réinitialiser la connexion. Ceci est utilisé lorsque les extrémités sont désynchronisées.

- Exemple :



8.3.5.3 TCP FORMAT DU SEGMENT, CHECKSUM

- CHECKSUM : calcul du champ de contrôle : utilise une pseudo-en-tête et s'applique à la totalité du segment obtenu (PROTO =6) :



8.3.5.4 TCP : FORMAT DU HEADER

- OPTIONS
- Permet de négocier la taille maximale des segments échangés. Cette option n'est présente que dans les segments d'initialisation de connexion (avec bit SYN).
- TCP calcule une taille maximale de segment (MSS) de manière à ce que le datagramme IP résultant corresponde au MTU (Maximum Transmission Unit) du réseau. La recommandation est de 536 octets.
- La taille optimale du segment correspond au cas où le datagramme IP n'est pas fragmenté mais :
  - il n'existe pas de mécanisme pour connaître le MTU,
  - le routage peut entraîner des variations de MTU,
  - la taille optimale dépend de la taille des en-têtes (options).

8.3.6 TCP : ACQUITTEMENTS

- Acquittements et retransmissions

- Le mécanisme d'acquittement de TCP est cumulatif :

  - il indique le numéro de séquence du prochain octet attendu : tous les octets précédents cumulés sont implicitement acquittés

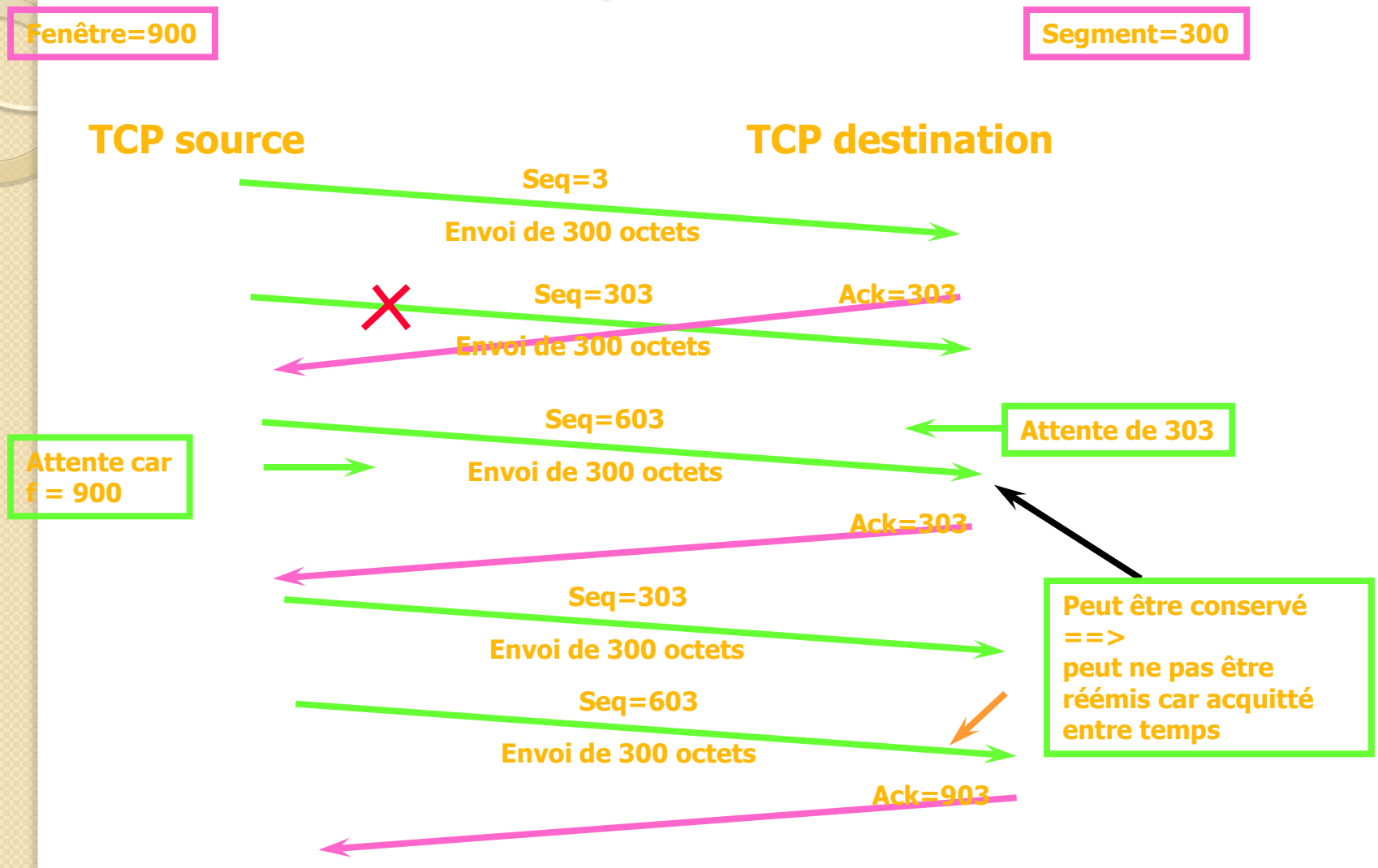
  - Si un segment a un numéro de séquence supérieur au numéro de séquence attendu (bien que dans la fenêtre), le segment est conservé mais l'acquittement référence toujours le numéro de séquence attendu(-->).

8.3.6 TCP : ACQUITTEMENTS

- Acquittements et retransmissions

- Pour tout segment émis, TCP s'attend à recevoir un acquittement
  - Si le segment n'est pas acquitté, le segment est considéré comme perdu et TCP le retransmet.
  - Or un réseau d'interconnexion offre des temps de transit variables nécessitant le réglage des temporisations;
  - TCP gère des temporisations variables pour chaque connexion en utilisant un algorithme de retransmission adaptative

8.3.6 TCP : ACQUITTEMENTS



8.3.7 TCP : RETRANSMISSIONS

- algorithme de retransmission adaptative

- enregistre la date d'émission d'un segment,
- enregistre la date de réception de l'acquittement correspondant,
- calcule l'échantillon de temps de boucle A/R écoulé,
- détermine le temps de boucle moyen RTT (Round Trip Time) :

  - RTT = (a * anc_RTT) + ((1-a) * NOU_RTT))
  - avec 0<= a < 1

  - a proche de 1 : RTT insensible aux variations brèves,
  - a proche de 0 : RTT très sensible aux variations rapides,

8.3.7 TCP : RETRANSMISSIONS

- algorithme de retransmission adaptative

- calcule la valeur du temporisateur en fonction de RTT.
- Les premières implémentations de TCP ont choisi un coefficient constant B pour déterminer cette valeur : Temporisation = B * RTT avec B >1 (généralement B=2).
- Aujourd'hui de nouvelles techniques sont appliquées pour affiner la mesure du RTT : l'algorithme de Karn.

8.3.7 TCP : RETRANSMISSIONS

- L'algorithme de Karn repose sur les constatations suivantes :
  - en cas de retransmission d'un segment, l'émetteur ne peut savoir si l'acquittement s'adresse au segment original ou retransmis (ambiguïté des acquittements),
  - => TCP ne doit pas mettre à jour le RTT pour les segments retransmis.

8.3.7 TCP : RETRANSMISSIONS

- L'algorithme de Karn combine les retransmissions avec l'augmentation des temporisations associées (timer backoff):
  - une valeur initiale de temporisation est calculée
  - si une retransmission est effectuée, la temporisation est augmentée (généralement le double de la précédente, jusqu'à une valeur plafond).
- Cet algorithme fonctionne bien même avec des réseaux qui perdent des paquets.

8.3.8 TCP : LA CONGESTION

- Gestion de la congestion
- TCP gère le contrôle de flux de bout en bout mais également les problèmes de congestion liés à l'interconnexion.

- La congestion correspond à la saturation de nœud(s) dans le réseau provoquant des délais d'acheminement de datagrammes jusqu'à leur pertes éventuelles.

- Les extrémité ignorent tout de la congestion sauf les délais. Habituellement, les protocoles retransmettent les segments ce qui aggrave encore le phénomène.

8.3.8 TCP : LA CONGESTION

- Dans la technologie TCP/IP, les passerelles (niveau IP) utilisent la réduction du débit de la source mais TCP participe également à la gestion de la congestion en diminuant le débit lorsque les délais s'allongent :
  - TCP maintient une fenêtre virtuelle de congestion
  - TCP applique la fenêtre d'émission suivante:
    - fenêtre_autorisée = min (fenêtre_récepteur, fenêtre_congestion).
- Dans une situation de non congestion:
  - fenêtre_récepteur = fenêtre_congestion.

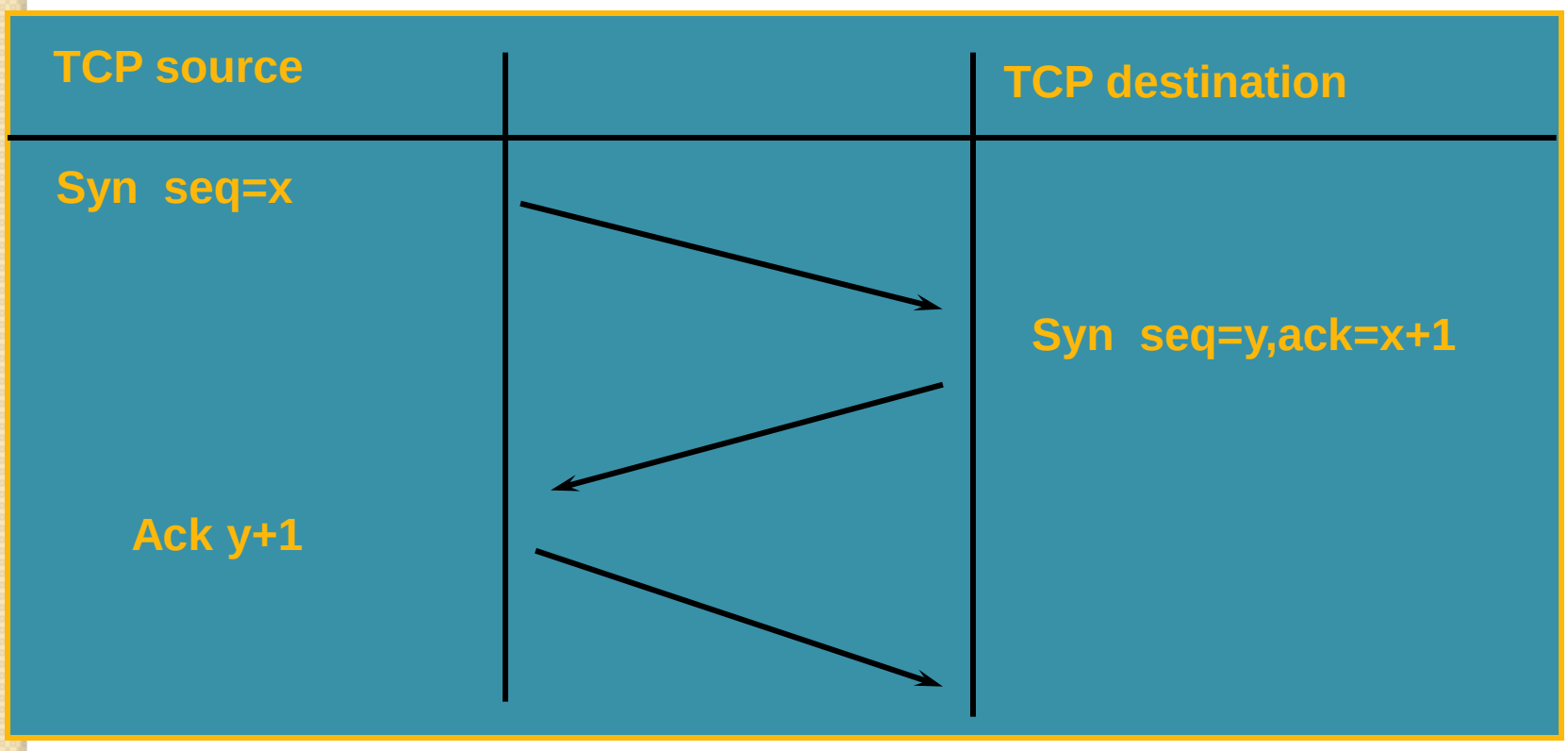
8.3.8 TCP RETRANSMISSIONS

- En cas de congestion, TCP applique une diminution dichotomique :
  - à chaque segment perdu, la fenêtre de congestion est diminuée par 2 (minimum 1 segment)
  - la temporisation de retransmission est augmentée exponentiellement.

- Si la congestion disparaît, TCP définit une fenêtre de congestion minimum égale à 1 segment et l'incrémente de 1 chaque fois qu'un acquittement est reçu; ce mécanisme permet un démarrage lent et progressif.

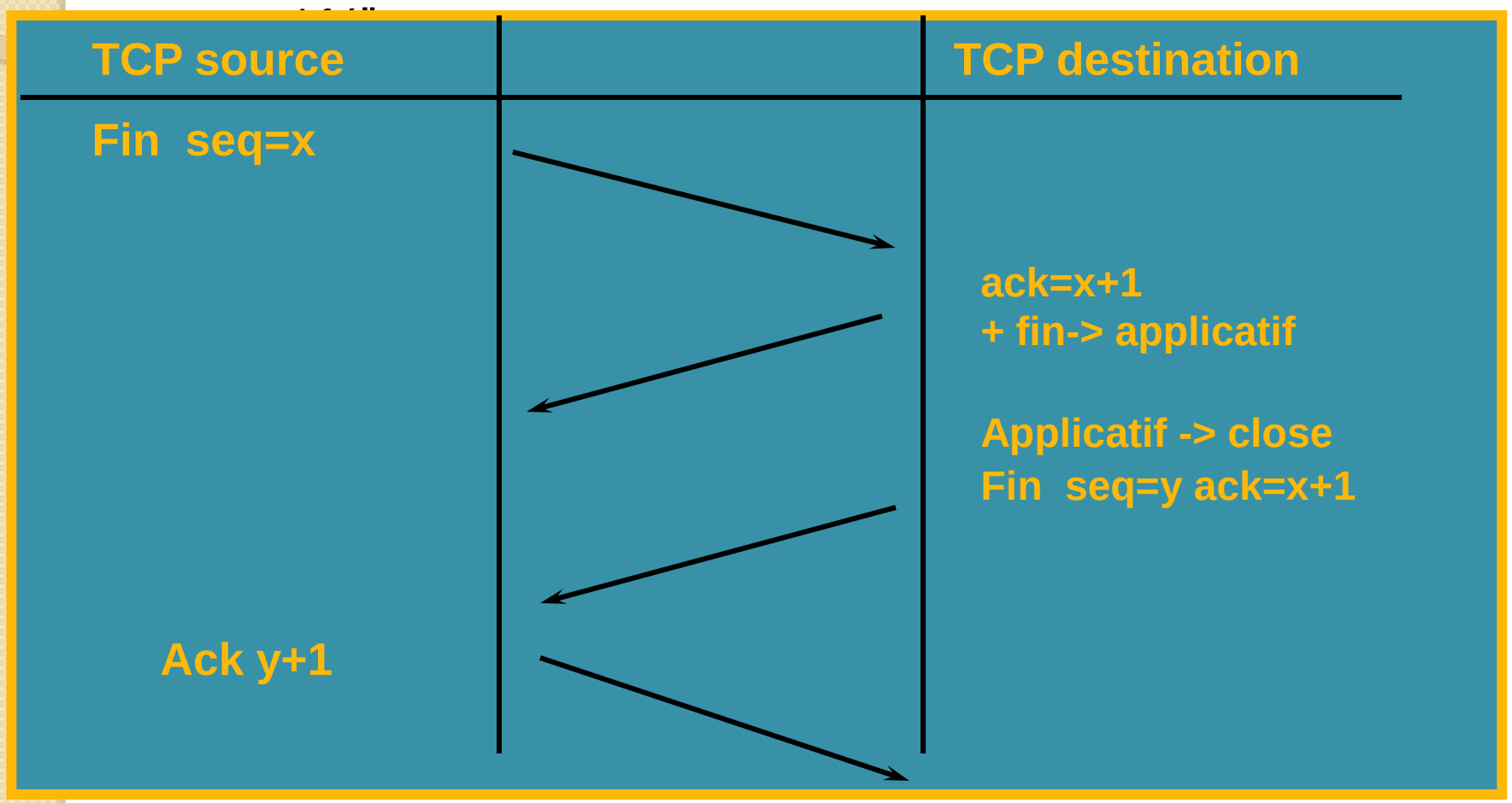
8.3.9 TCP : EXEMPLE CONNEXION

- Une connexion TCP est établie en trois temps de manière à assurer la synchronisation nécessaire entre les extrémités :

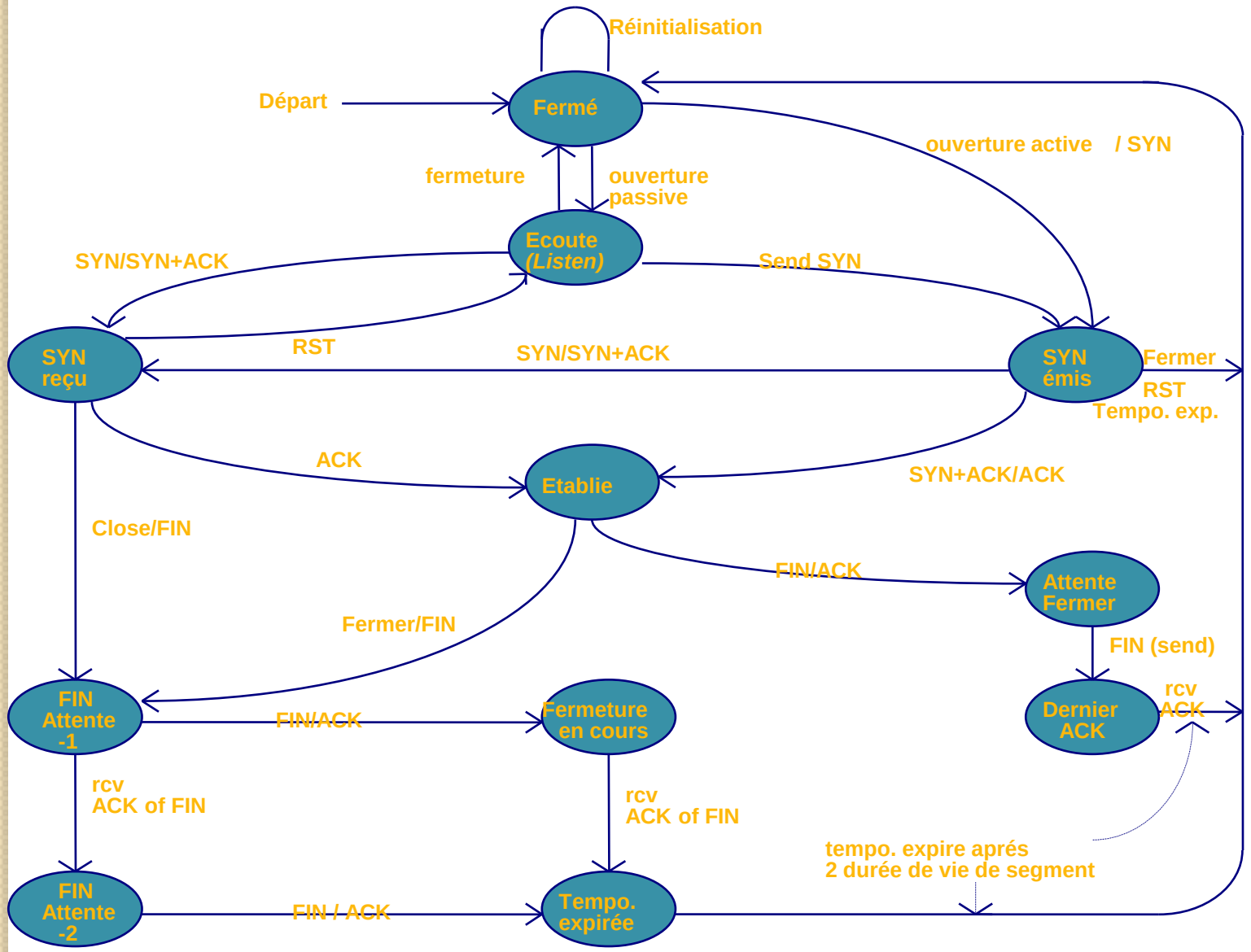


8.3.10 TCP : EXEMPLE DÉCONNEXION

- Une connexion TCP est libérée en un processus dit "trois temps



8.3.11 TCP : L'AUTOMATE



8.3.12 TCP : PORTS STANDARDS

No port		Mot-clé	Description
20	FTP-DATA	File Transfer	[Default Data]
21	FTP	File Transfer	[Control]
23	TELNET	Telnet	
25	SMTP	Simple Mail Transfer	
37	TIME	Time	
42	NAMESERVER	Host Name Server	
43	NICNAME	Who Is	
53	DOMAIN	Domain Name Server	
79	FINGER	Finger	
80	HTTP	WWW	
110	POP3	Post Office Protocol-Version 3	
111	SUNRPC	SUN Remote Procedure Call	