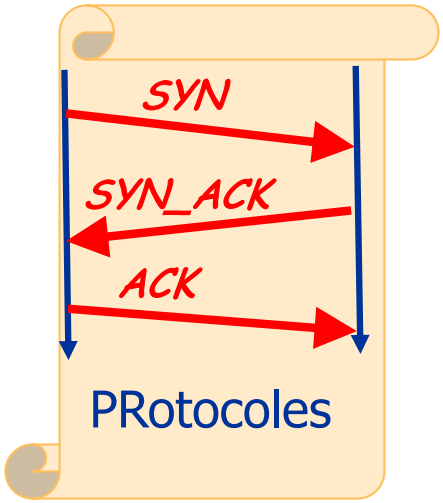
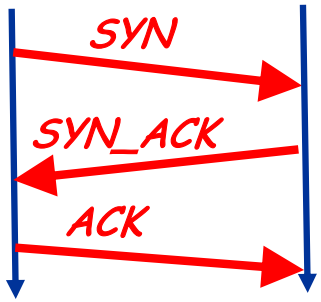
GRENOBLE INP UGA Ensimag

Chapitre Protocoles

Notion de protocole
Terminologie et concepts
Diagrammes temporels
Aperçu de TCP et UDP



PR – Protocoles

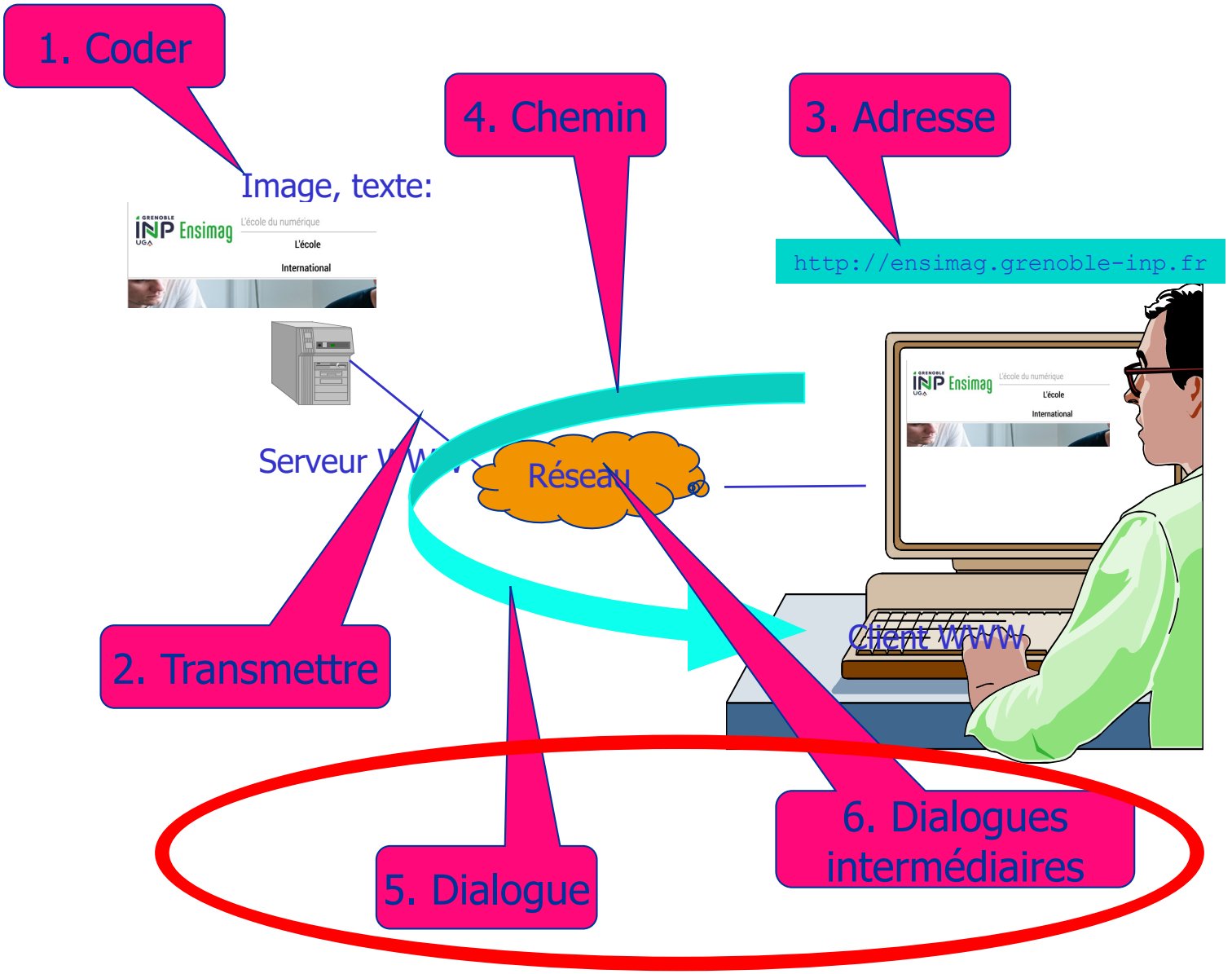


- Comment s'organise le dialogue entre machines différentes
- Vous comprendrez:
  - *Comment analyser les échanges entre deux machines*
  - (PR2) Comment deux machines peuvent acheminer sans perte des informations même si le réseau perd des messages

Plan PR1

- Notion de protocole
- Constituants d'un protocole
  - PDU, formats
  - Règles et enchaînements
- Diagrammes temporels: illustrent des enchaînements
- Lien avec l'architecture en couches
  - insertion protocole-couche
  - encapsulation
- 1er aperçu de UDP, TCP

Problèmes à résoudre

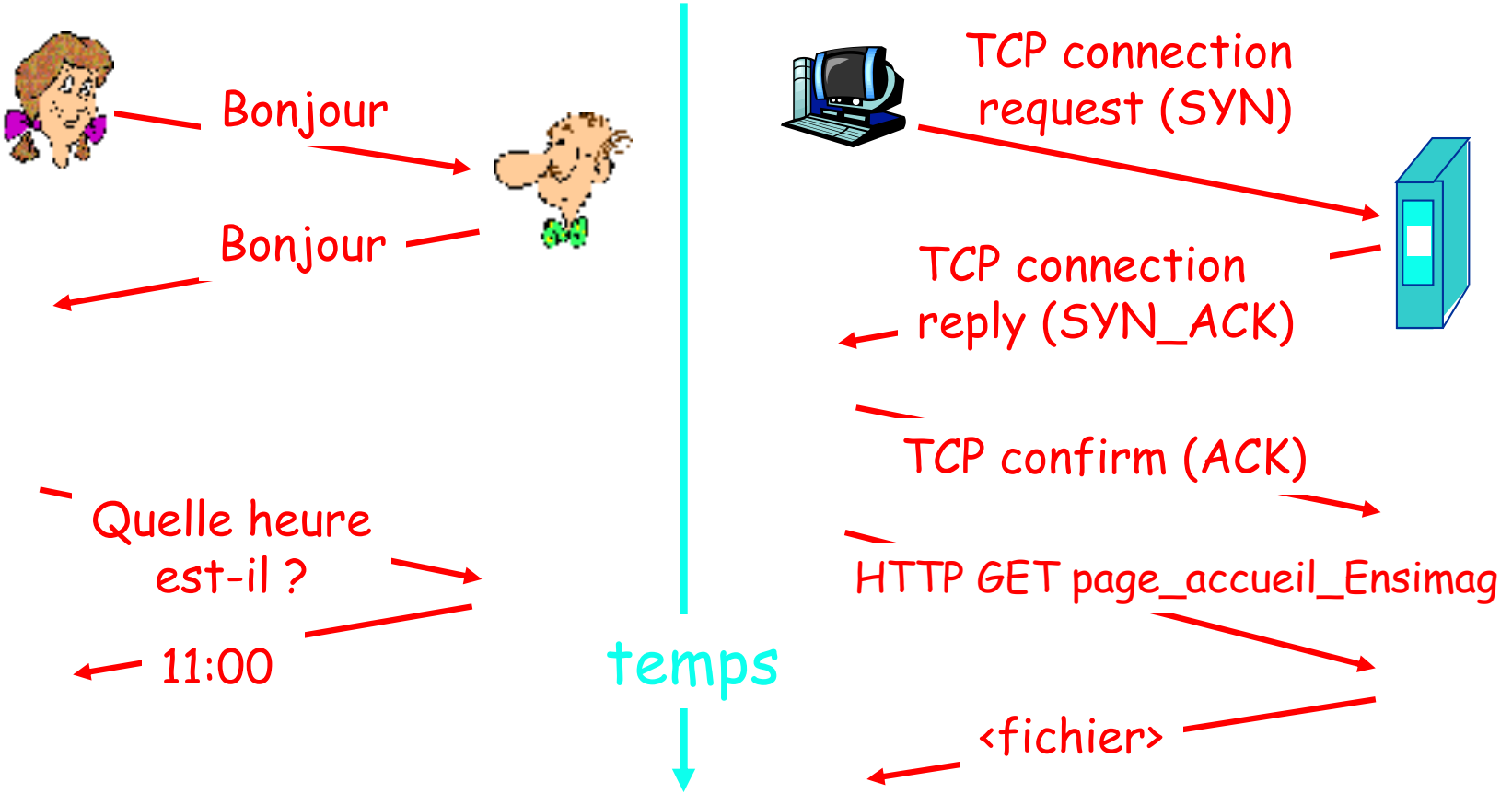


Pb5 - Dialogue entre machines

- Réseau relie des pgms informatiques
  - pas directement des humains
- Pgm répartis: en morceaux qui s'exécutent sur différentes machines
  - Application Web: morceau1=navigateur, morceau2:serveur Apache
- Machines « stupides »: besoin de protocoles précis pour se comprendre, parler le même langage, sinon se bloquent.
  - Web: navigateur et Apache se parlent HTTP

Illustration de protocoles

protocole humain & protocole réseau (entre machines)



Notion de protocole

Protocoles (sens générique)

- “comment allez-vous ?” / “how do you do”
- présentations

… génération de certains messages

… des actions spécifiques à la réception

… un ordre à respecter pour bien se comprendre

Protocoles réseau

- entre des machines
- toutes les communications sous contrôle des protocoles

Les protocoles définissent:
1.le format
2.l'ordre des messages envoyés et reçus
3.des actions à entreprendre à l'envoi et à la réception de messages

Protocole: défini par

PDU

1. Format des ~~messages~~:

successions de champs d'information (ex: @exp, heure, longueur et type du contenu...)

- codage de l'information des informations dans chaque champ

Message: PDU (Protocol Data Unit)

- Règles:
  2. d'enchaînement
  3. actions à effectuer

Exemple:

- *sur réception du PDU de type x*
  - *si le champ x.z = 0 alors répondre par un PDU de type y*
  - *sinon stocker la valeur reçue*

Exemples de PDU couche applicative: HTTP *HyperText Transfer Protocol*

Codé en caractères ASCII

- Exemple de requête

```
GET / HTTP/1.1
Host: web.ensimag.fr
```

- Exemple de réponse

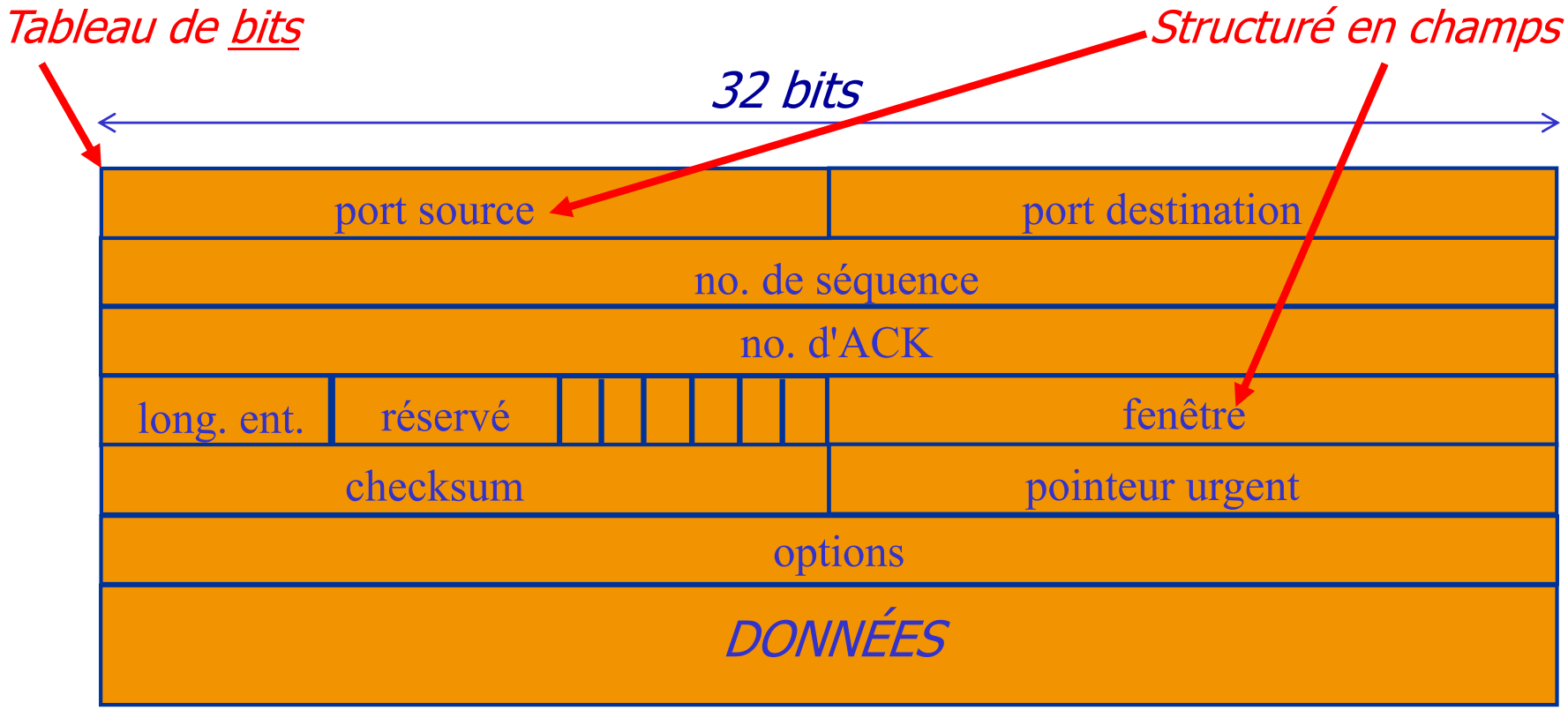
```
Date: Wed, 10 Oct 2007 17:20:14 GMT
Server: Apache/2.0.52 (Red Hat)
Last-Modified: Thu, 14 Jun 2007 11:36:53 GMT
Content-Length: 1497
Connection: close
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
<title>Extranet ensimag</title>
  etc.
```

Entête

Contenu

Exemple de PDU couche « basse »: TCP

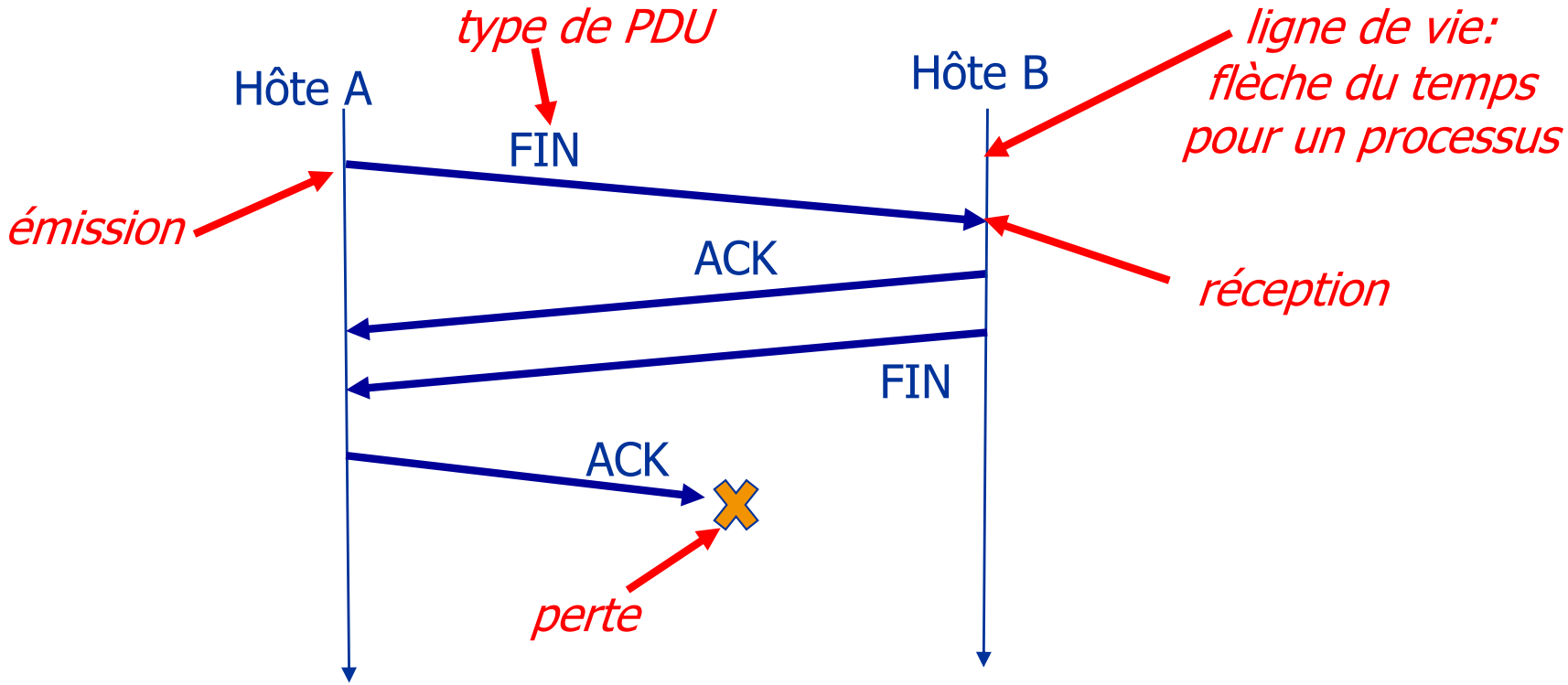


En fait: les Nx32 bits sont transmis l'un après l'autre

On « replie » les lignes en tableau pour la lisibilité

- No. de séquence
  - no. du premier octet des DONNÉES
- No. d'ACK
  - no. de l'octet attendu

Diagramme temporel: enchaînement



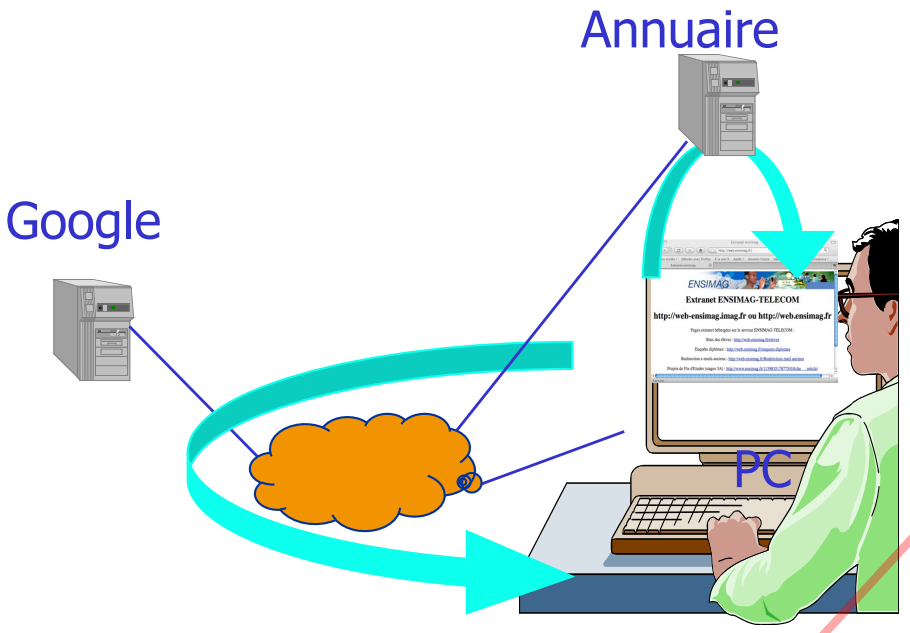
- Représente le fonctionnement dynamique des échanges

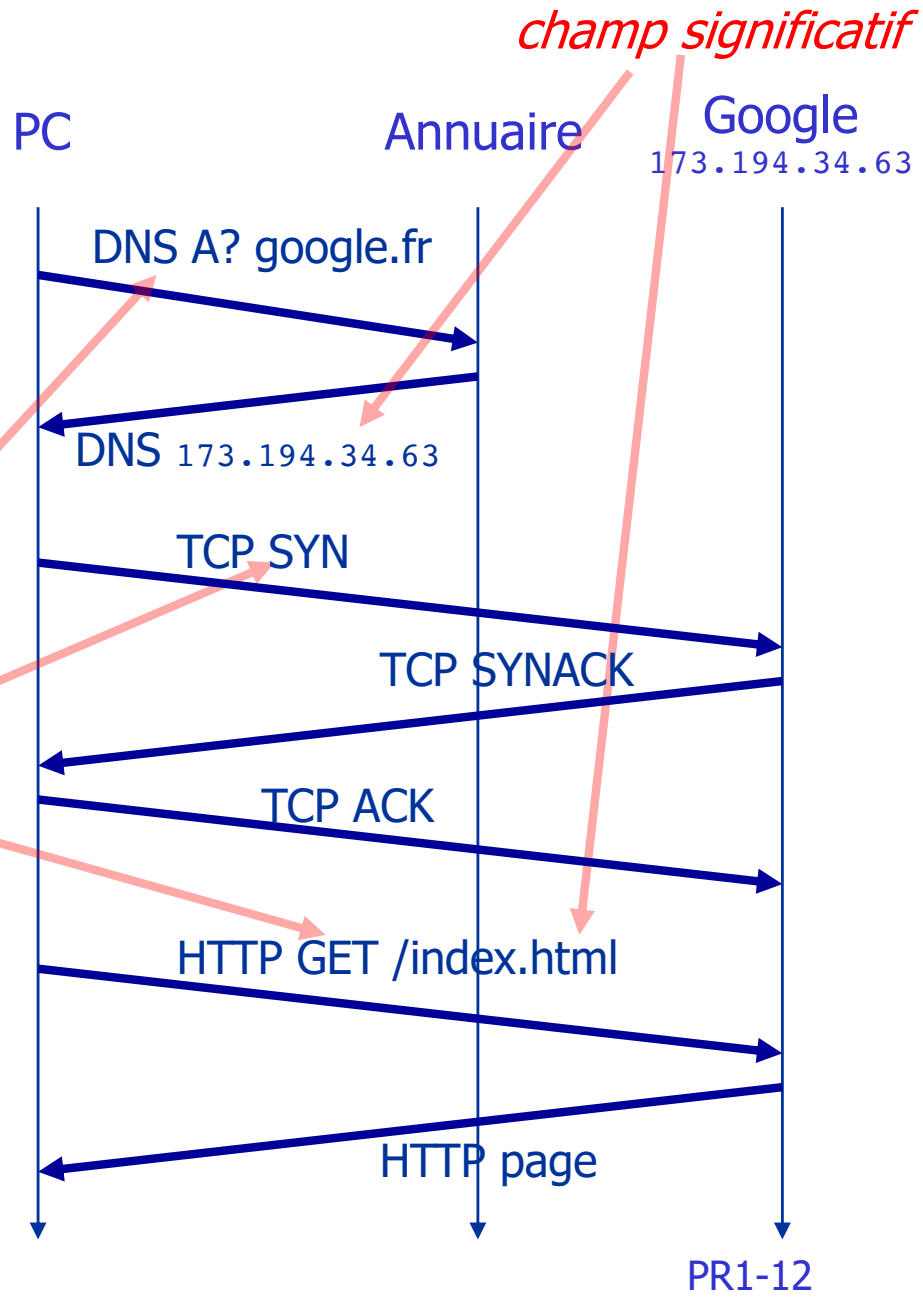
(le diagramme de couches est statique)

NB: dynamique (évolue / temps) <-/-> statique

- Il existe une notation normalisée un peu différente (MSC: Message Sequence Charts, norme ITU Z.120)
  - et une variante appelée Sequence Diagrams en notation UML

Diagramme temporel à 3 machines



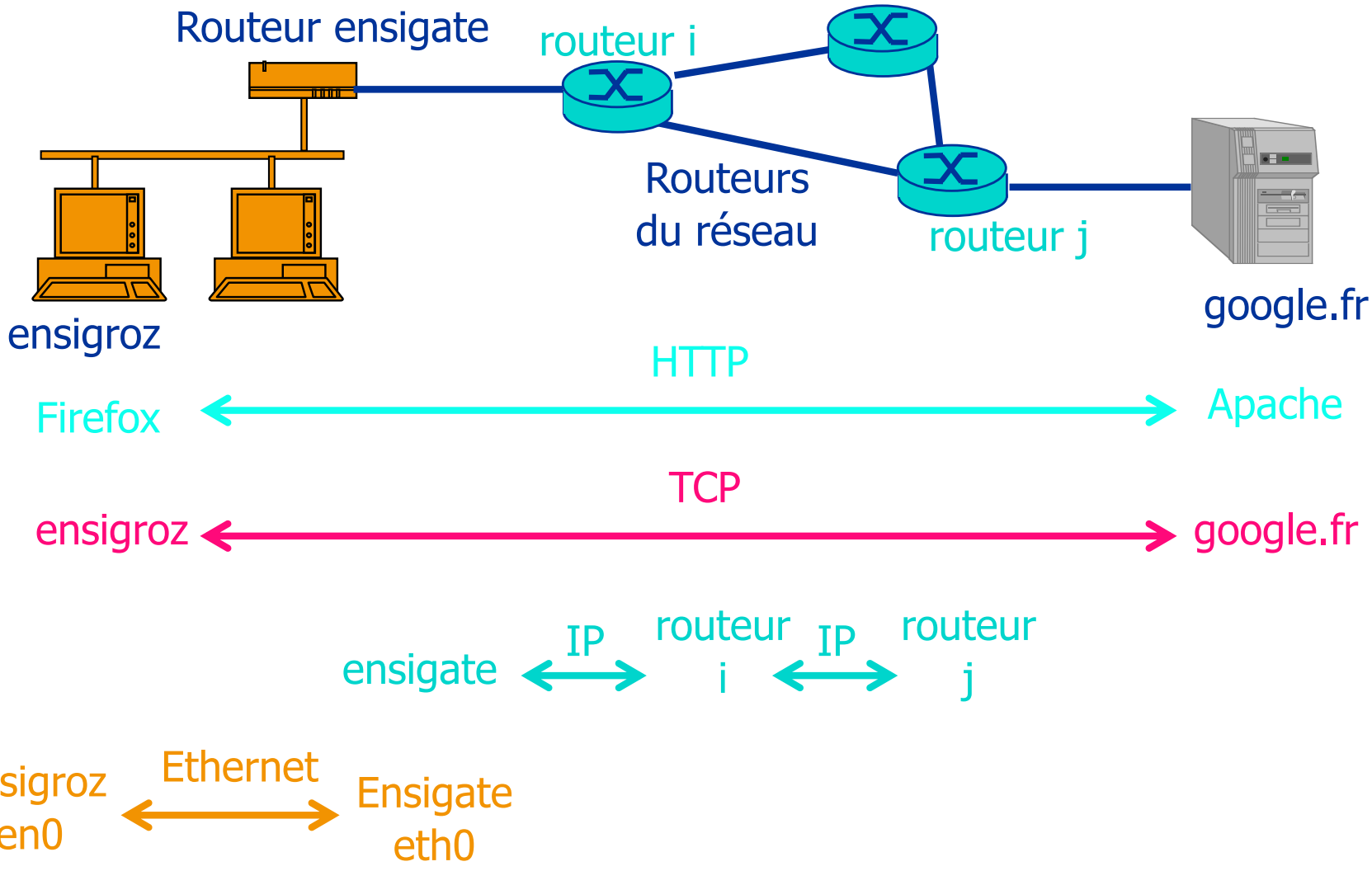


N.B. Ce diagramme fait apparaître *3 protocoles: DNS, TCP et HTTP*, utilisés successivement pour des Besoins différents.

Plan PR1

- Notion de protocole
- Constituants d'un protocole
  - PDU, formats
  - Règles et enchaînements
- Diagrammes temporels
- Lien avec l'architecture en couches
  - insertion protocole-couche
  - encapsulation
- 1er aperçu de UDP, TCP

Pb 6 - Dialogues via intermédiaires

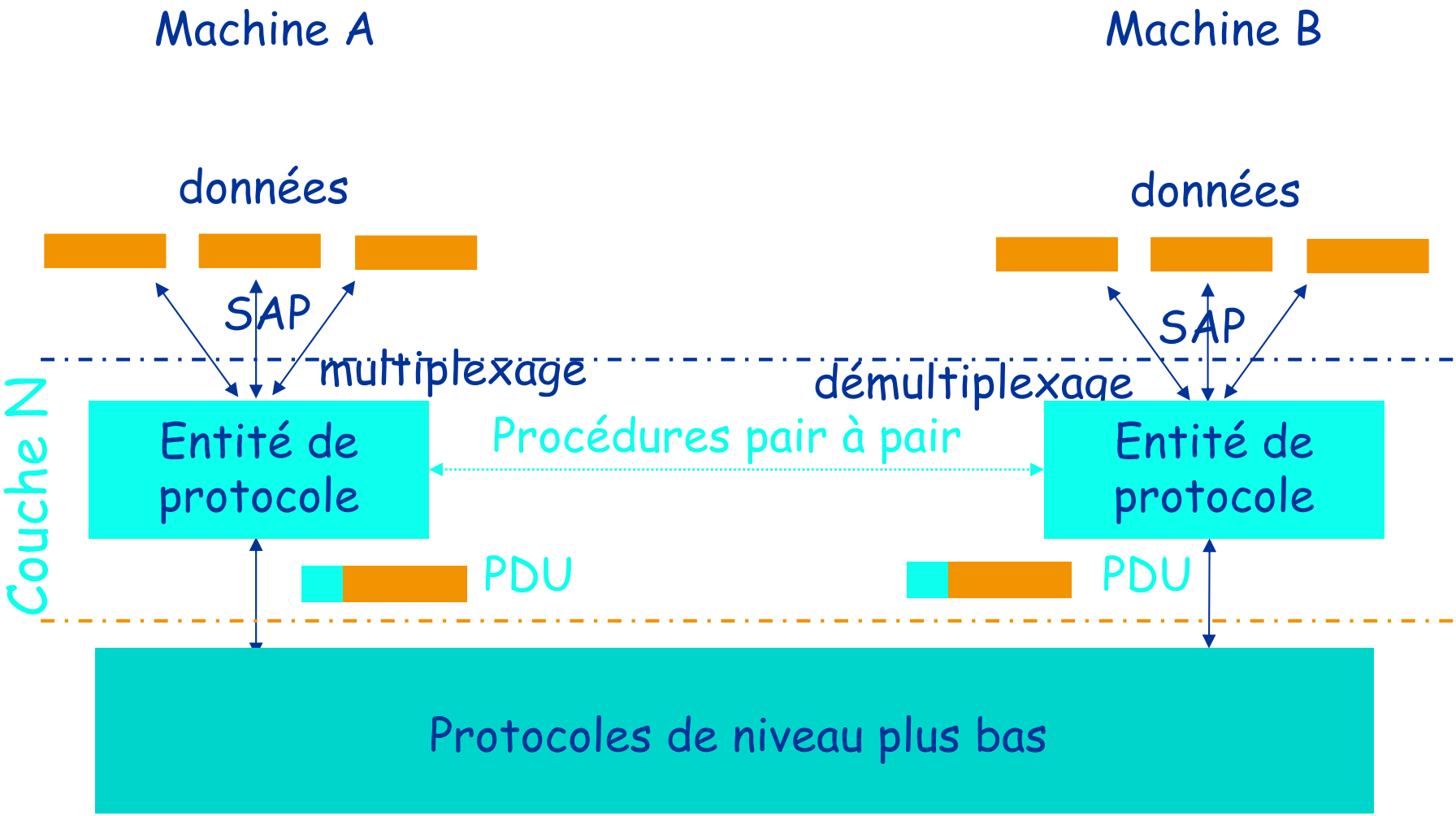


Protocoles de réseau sous-jacents

- HTTP utilise une connexion TCP (protocole gérant le transport des informations)

  TCP = « tuyau » pour communiquer entre navigateur & serveur Web
- HTTP: dialogue entre applications
  - sait dire: « envoie-moi telle page (GET) »
  - ne sait pas dire: « envoie ce GET à telle machine »
- TCP: dialogue entre processus logiciels de machines (hôtes)
- TCP s'appuie sur le protocole IP pour parler à son routeur (IP gère l'acheminement)
- IP s'appuie sur la carte Ethernet pour transférer ses octets sur le câble vers le routeur

Architecture de protocoles



Protocole: dans une couche

- Un protocole comporte
  - un format commun de données : unité de données - *PDU (Protocol Data Unit)*
    - en-tête : fonctions de contrôle
    - données opaques pour le protocole
  - un ensemble de règles de dialogue : procédures - *peer to peer procedures*

Algorithme

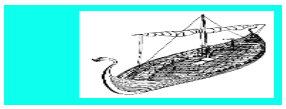
- Module logiciel: entité de protocole - *protocol entity*
  - une interface de service : *SAP (Service Access Point)* qui offre un ensemble de services (ex. `connect, send`), à laquelle on passe des SDU (*Service Data Unit*)
  - fonctions internes: construction/analyse des unités de protocoles, exécution des procédures (actions sur réception de PDU ou requête)

Programme

Encapsulation des messages

- Chaque protocole (HTTP, TCP, IP, Ethernet) réalise une partie des fonctions nécessaires
- Les messages des différents protocoles sont emboîtés comme des poupées russes: encapsulation
- Une même trame Ethernet encapsule les informations pour plusieurs protocoles à la fois
- L'outil `wireshark` observe et décortique la poupée russe qui circule sur le câble Ethernet

Encapsulation



Réponse HTTP

en-tête

données

Segment TCP

en-tête TCP

Paquet IP

en-tête IP

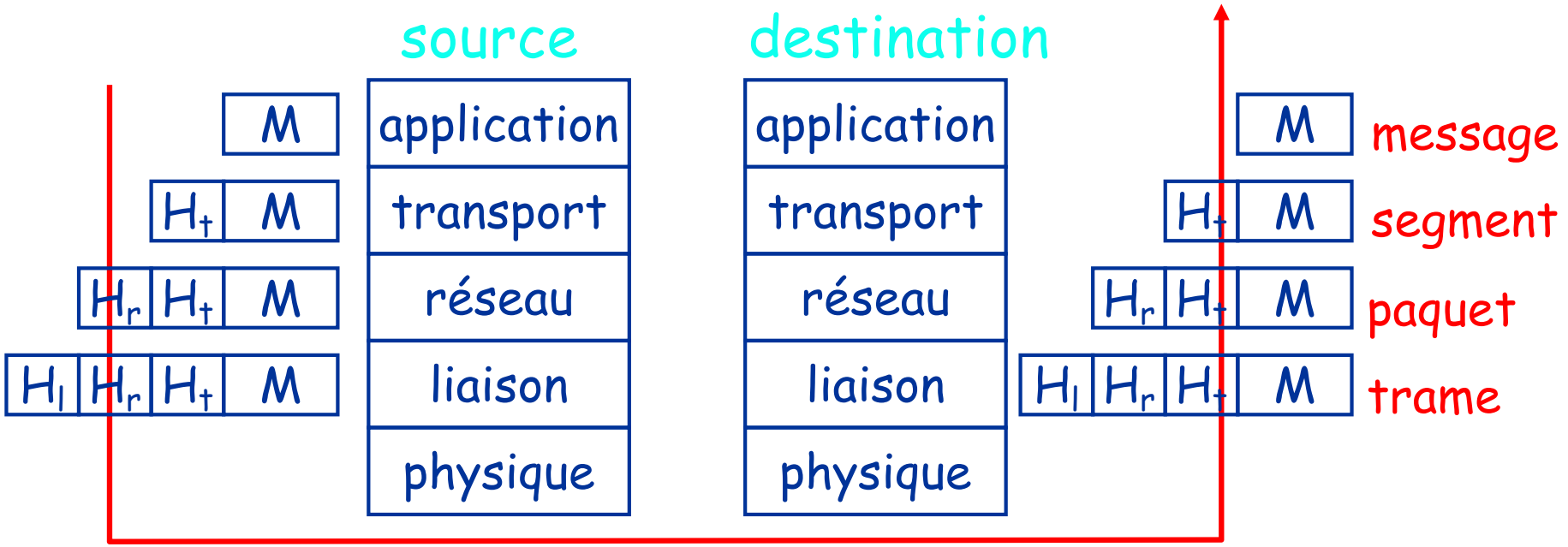
Trame Ethernet

en-tête

Couches et données

Chaque couche prend des données de la couche au dessus

- ajoute un en-tête pour créer un PDU nouveau
- passe ce PDU en tant que données à la couche au dessous



Analogie postale

- Lettre: contenu applicatif
  - Confiée à la poste (service fiable de transfert)
- Enveloppe: protocole TCP (transfert fiable)
  - Adresse expéditeur et destinataire (ex client & serveur), qualité de service demandée (affranchissement)…
- Sac de courrier: paquet IP
  - sur-emballage utilisé par la poste pour acheminer les données entre ses différents centres de manutention
- Camion postal: trame Ethernet
  - Support physique pour amener l'information à distance

Parlez-vous HTTP ?

- `telnet` permet d'ouvrir un tuyau (TCP) de communication avec un hôte distant (par exemple un serveur WWW, désigné par 80)

```
bash$ telnet web.ensimag.fr 80
```

- Envoi d'une requête

```
GET / HTTP/1.0
```

- Exemple de réponse

```
Date: Wed, 10 Oct 2007 17:20:14 GMT
Server: Apache/2.0.52 (Red Hat)
Last-Modified: Thu, 14 Jun 2007 11:36:53 GMT
Content-Length: 1497
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
<title>Extranet ensimag</title>
```

Plan PR1

- Notion de protocole
- Constituants d'un protocole
  - PDU, formats
  - Règles et enchaînements
- Diagrammes temporels
- Lien avec l'architecture en couches
  - insertion protocole-couche
  - encapsulation
- 1er aperçu de UDP, TCP

Plusieurs protocoles de transport

- Car les besoins des applications diffèrent: privilégier retour rapide (UDP), ou fiabilité (TCP), ou régularité (RTP pour des applications audio ou vidéo), etc (MPTCP, ...)
- Service offert différent pour chaque protocole
  - UDP: multiplexage de messages, transport rapide sans garantie (perte possible), détection erreurs (sans correction)
  - TCP: multiplexage, segmentation (transport de flux), récupération des erreurs et pertes, adaptation à la congestion du réseau...
- Service = fonctions offertes
  - QoS : Qualité de Service, caractéristiques non fonctionnelles (valeur des performances etc)

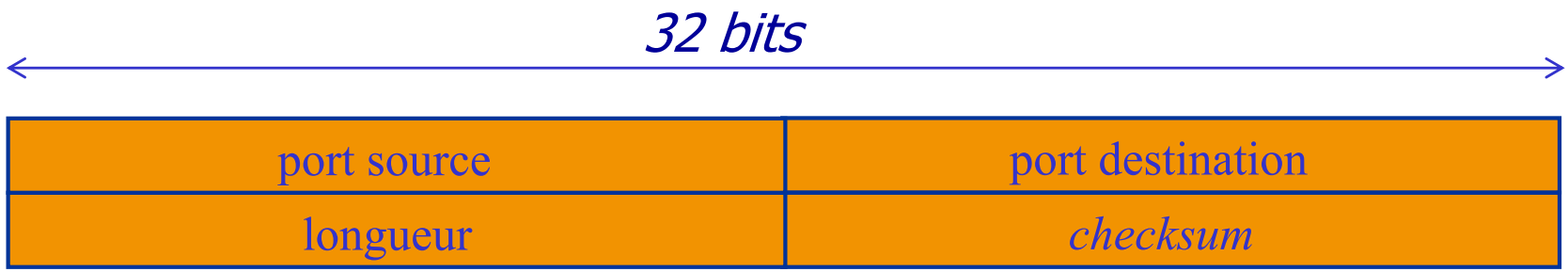
Choix du protocole de transport

- DNS (annuaire) utilise UDP
  - parce que requête et réponses courtes: 1 seul PDU
  - pas besoin de segmentation
  - réponse rapide (pas d'ouverture de connexion)
  - si perte: redemande au même serveur ou à un autre
- HTTP (web), ssh (connexion), SMTP (mél) etc. utilisent TCP
  - garantie d'acheminement sans perte de longs fichiers
  - segmentation automatique par TCP
- SIP (téléphonie sur IP) utilise RTP (Real-time Transport Protocol)

UDP *(User Datagram Protocol)*

- Simple interface à IP
  - ajoute le multiplexage
    - plusieurs communications partagent la même interface de réseau
    - applications différenciées par les numéros de port (servent d'adresses locales à cet hôte)
  - un en-tête de 8 octets :
    - port source et destination,
    - longueur (limitée à 8 Koctets)
    - *checksum* sur le paquet
- Pas de garantie d'acheminement
  - En cas de perte (routeur saturé, mauvaise transmission etc), ni l'émetteur ni le récepteur n'en sont informés

En-tête UDP



- Ports source et destination
  - identificateurs d'application
- Longueur (en octets) totale y compris les 8 octets d' en-tête
  - min 8 (stupide: payload vide)
  - max théorique 64Koctets (peut être réduit par implém: BSD 8Ko)
- *Checksum* (somme de contrôle, code détecteur d'erreur)
  - contrôle d'erreurs
  - calcul checksum: comme TCP

TCP *(Transmission Control Protocol)*

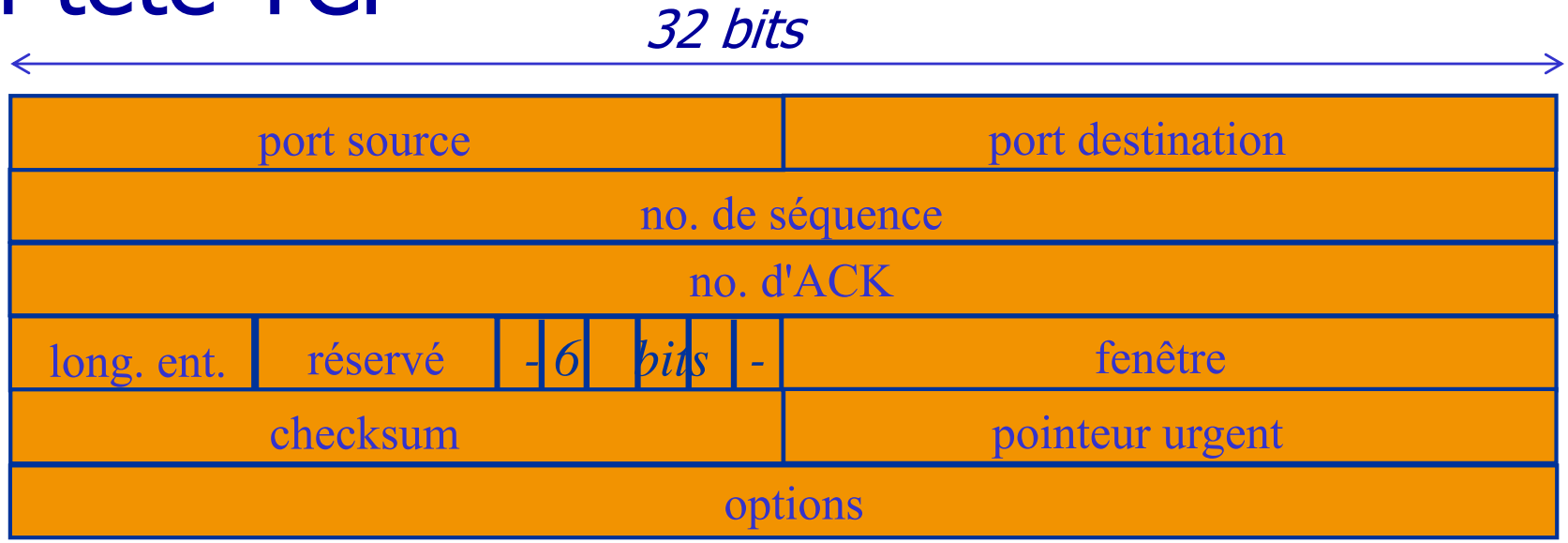
- Fonction
  - transfert d'une séquence d'octets
    - pas de marquage de messages : on numérote les octets (grain fin) et non les messages
- Unité de protocole
  - segment
- Phases
  - connexion
  - transfert
  - fermeture

TCP: Un protocole orienté connexion pour garantir la fiabilité et l'ordre des messages

TCP *(Transmission Control Protocol)*

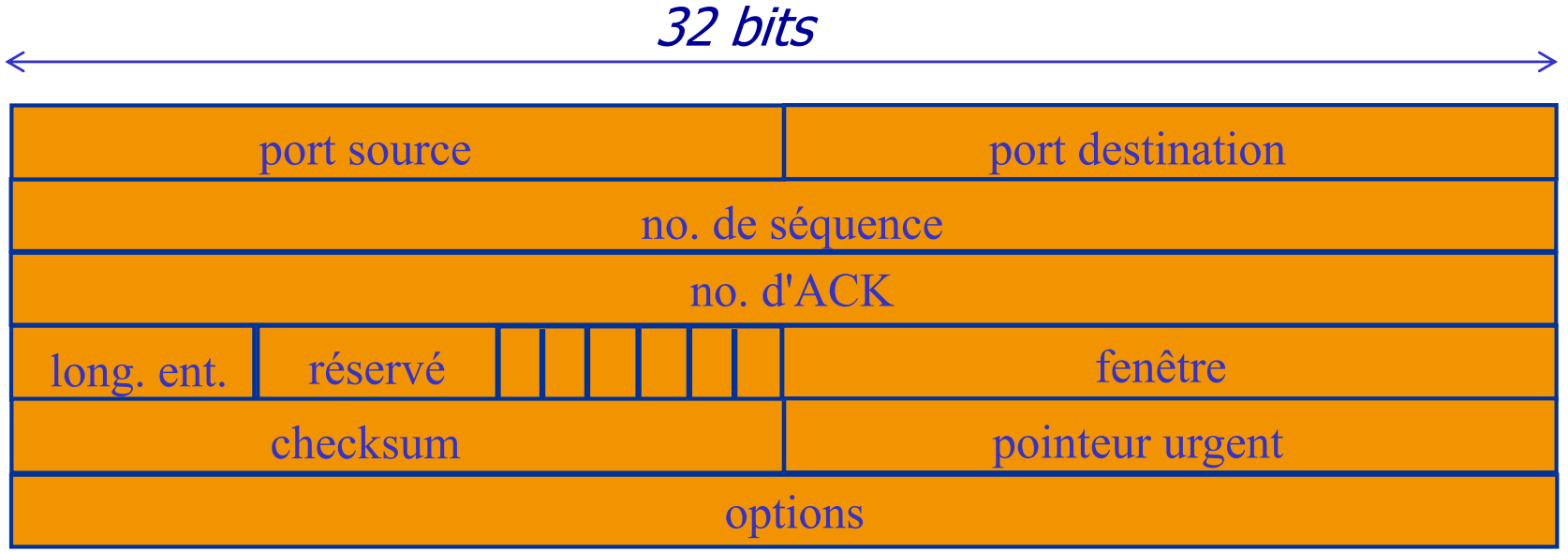
- Fiabilité
  - détection d'erreurs ou de trou dans la séquence de numéros par le récepteur
  - retransmission en cas de perte
    - heuristiques de *"retransmission rapide »*
- Contrôle de flux
  - fenêtre modulée par récepteur (crédit)
- Contrôle de congestion
  - adaptation à l'état d'occupation du réseau

En-tête TCP



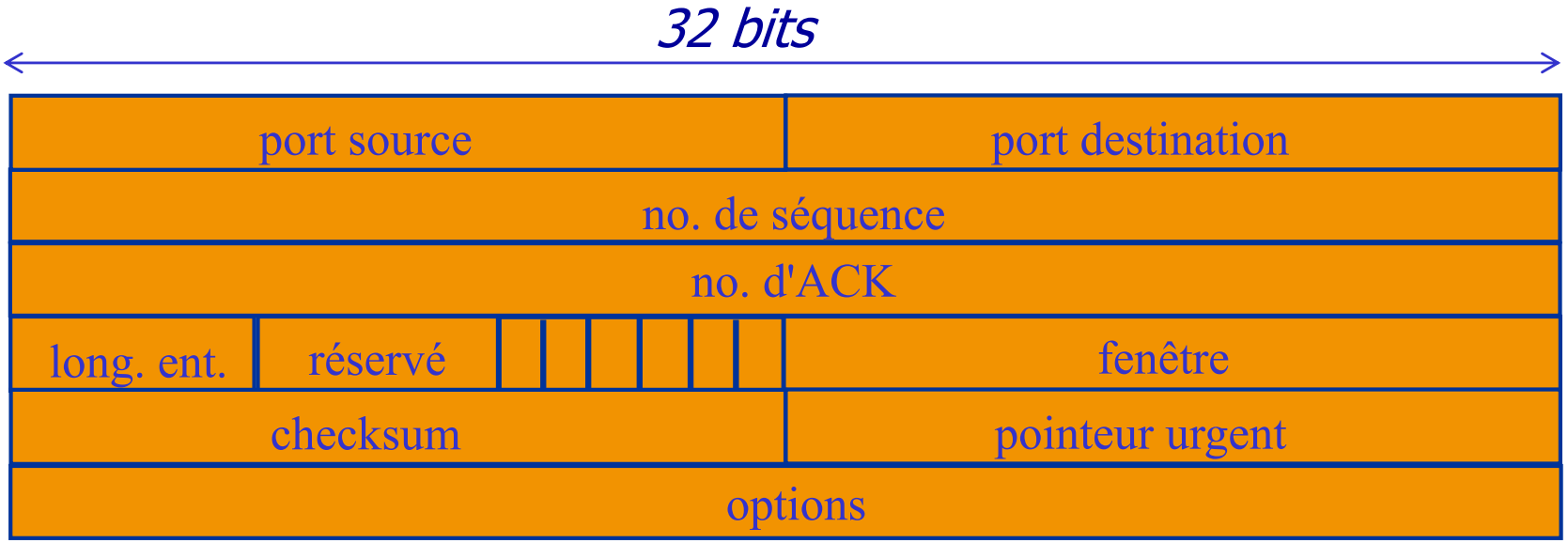
- longueur en-tête: 4 bits (nbr mots 32 bits); réservé: 6 bits
- 6 bits - *flags*: URGent, ACK, PuSH, ReSeT, SYN, FIN
  - SYN - segment de connexion, FIN - fermeture de connexion
  - ACK - no. d'ACK actif (i.e. bit ACK=1 signifie que les 32 bits au-dessus contiennent bien un No d' ACK significatif)
  - URG - pointeur urgent actif
  - PSH - force la création d'un segment et sa restitution à l'application

En-tête TCP



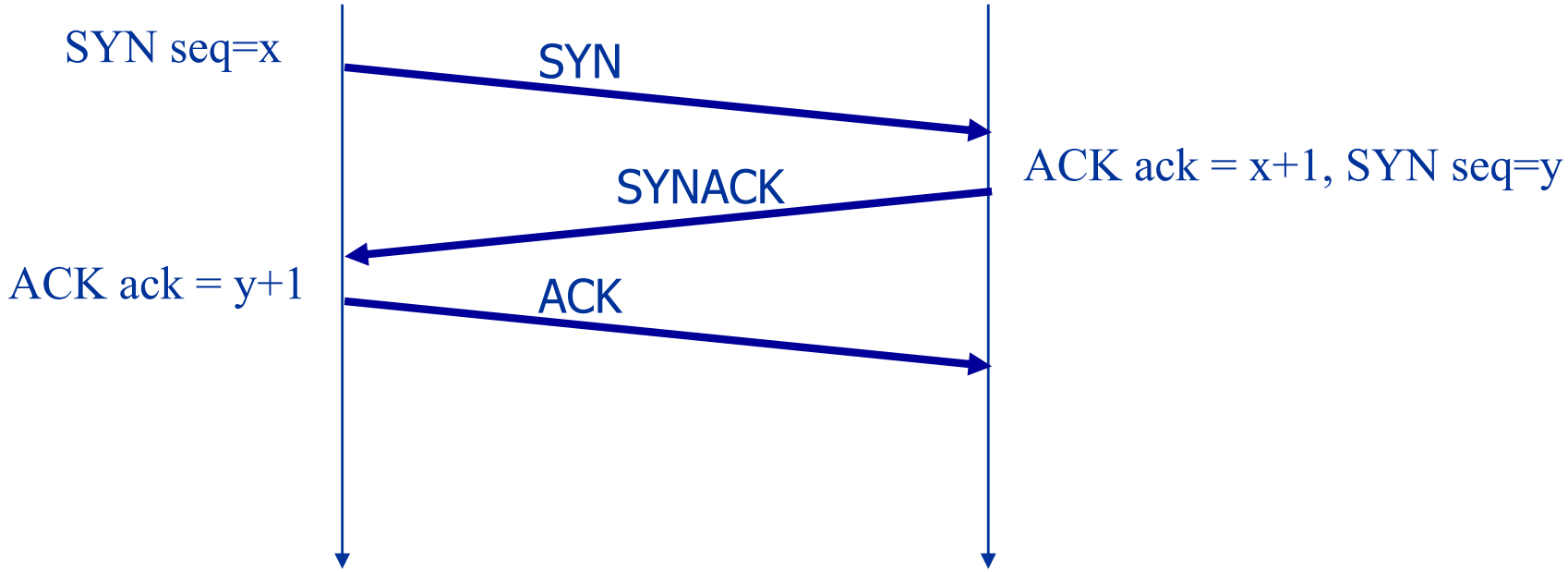
- Fenêtre annoncée
  - récepteur contrôle la fenêtre d'émission (par défaut 4 Koctets, max. 64Koctets)
- *Checksum* - contrôle d'erreurs (cf PR2)
  - sur le pseudo-en-tête: en-tête et les données TCP + adresses IP et type de protocole pris dans l'enveloppe IP + taille du pseudo en-tête)
  - complément à 1 sur somme des compléments à 1 des mots de 16 bits

En-tête TCP



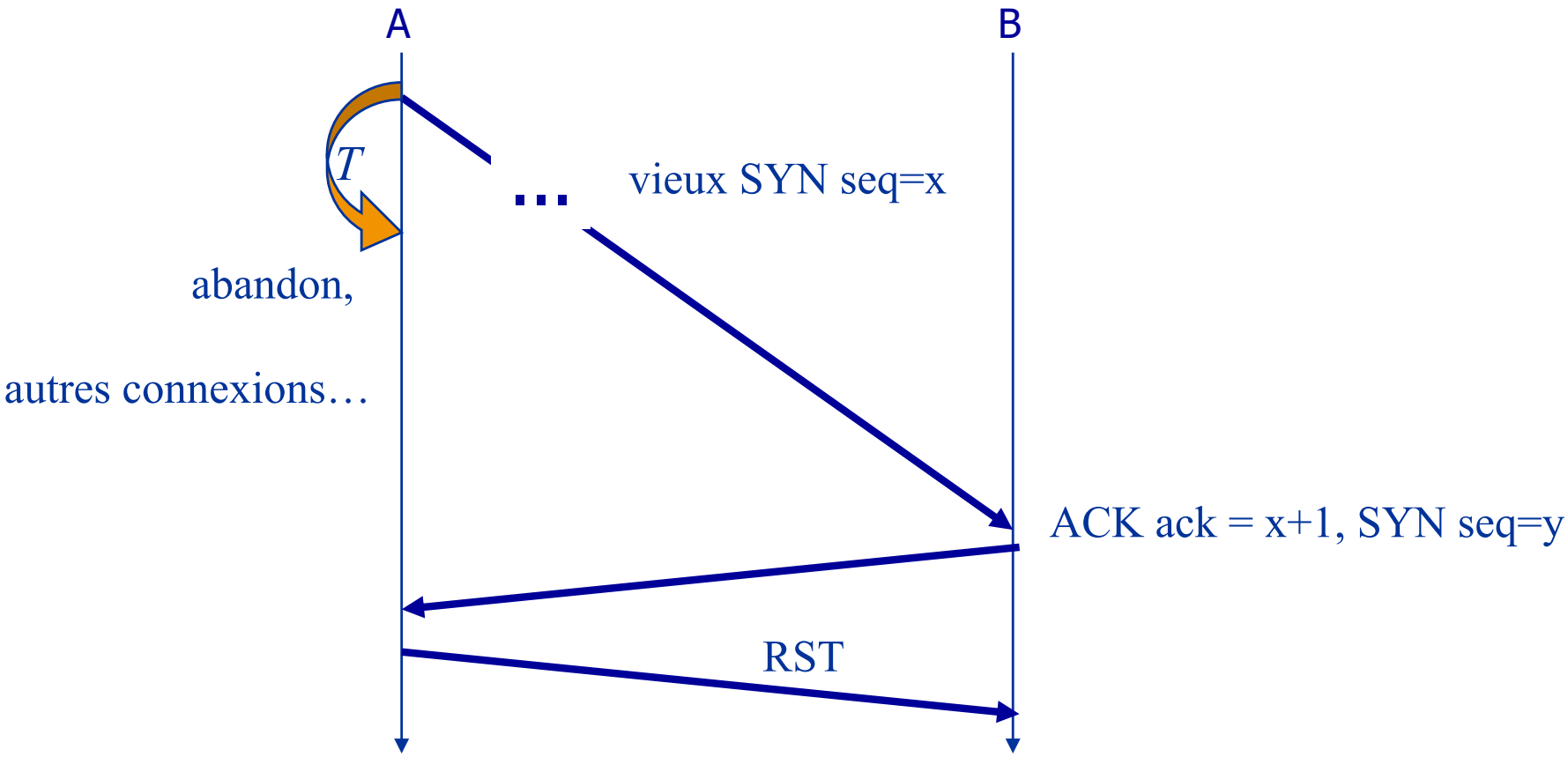
- Pointeur urgent
  - indique la fin des données urgentes
- Options (peut être vide, en-tête=20 octets)
  - MSS *(Maximal Segment Size)* (sans en-tête)
    - défaut 536 octets ; 1460

Enchaînement de PDU TCP: connexion



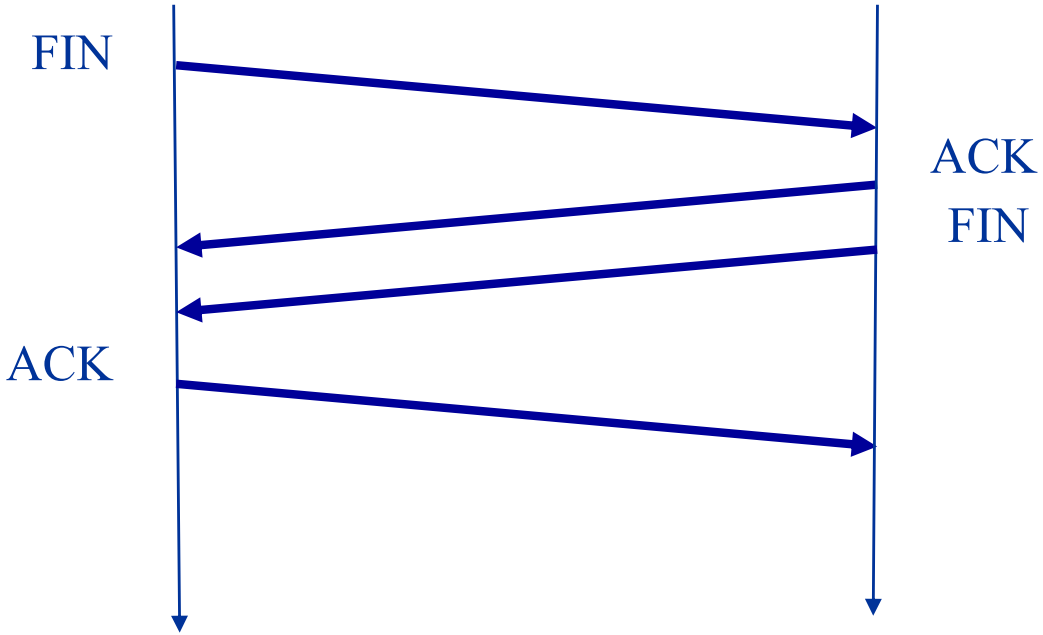
- Trois échanges *(three-way handshake)*
  - entente sur les numéros de séquences différents d'une connexion précédente
  - *x, y* : choisis en fonction de l'horloge
- Full-duplex (communication bi-directionnelle) avec 2 sens indépendants, chacun ayant sa numérotation d'octets

Exemple de connexion avortée



- Cas d'un segment retardé:
  - A détecte x périmé
  - A répond RST

Fermeture de connexion TCP



- Deux échanges *(two-way handshake)* pour chaque sens
- Chaque sens de communication libéré séparément
- Si B a encore des données à transmettre, il peut le faire indéfiniment, sinon il peut renvoyer ACK&FIN en même temps (3 messages)

Bilan PR1: notions essentielles

- Concept de protocole: algorithme réparti (règles) + format précis des messages (PDU)
- Terminologie: PDU, en-têtes, règles
- Structure d'encapsulation (associée aux couches)
- *Graphiques: diagrammes temporels*