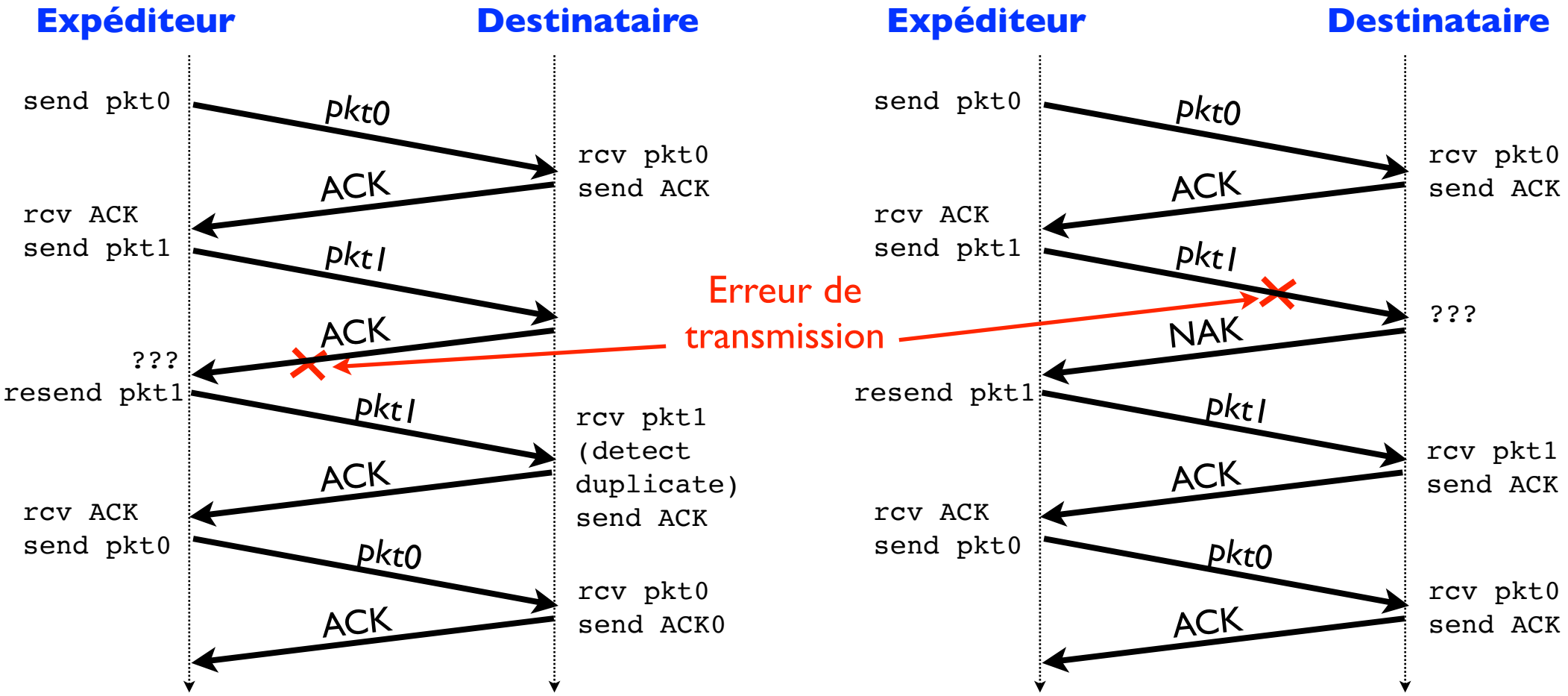
rdt2.0 : quel est le problème ?

- Si un segment de données arrive erroné 
- Mais que se passe-t-il si un ACK ou NAK arrive erroné ? 
- L'expéditeur ne sait pas si son paquet est correctement arrivé au destinataire
- Solution : simplement retransmettre le dernier paquet envoyé ?
  - Non, car comment le destinataire peut-il savoir si le message retransmis est un duplicata ou une nouvelle donnée ?
  - Note : une appli peut délivrer une succession de paquets identiques !
- Solutions possibles
  - Avertir d'une répétition, puis retransmettre
    - requiert de définir un nouveau type de paquet
  - Ajouter un numéro de séquence dans le segment et retransmettre le dernier paquet envoyé
    - solution adoptée par TCP et rdt2.1

rdt2.1 : numéro de séquence (1)

- Ajouter un numéro de séquence dans le segment
  - qui identifie chaque segment
  - à la réception d'un ACK erroné, le segment est retransmis
  - si un duplicata arrive au niveau du récepteur, le segment est supprimé
- Pour un protocole "Send and wait"
  - combien de bits sont nécessaires pour le numéro de séquence ?
  - un bit suffit pour coder le numéro de séquence
  - uniquement dans le paquet (pas dans les ACK/NAKs)

rdt2.1 : numéro de séquence (2)



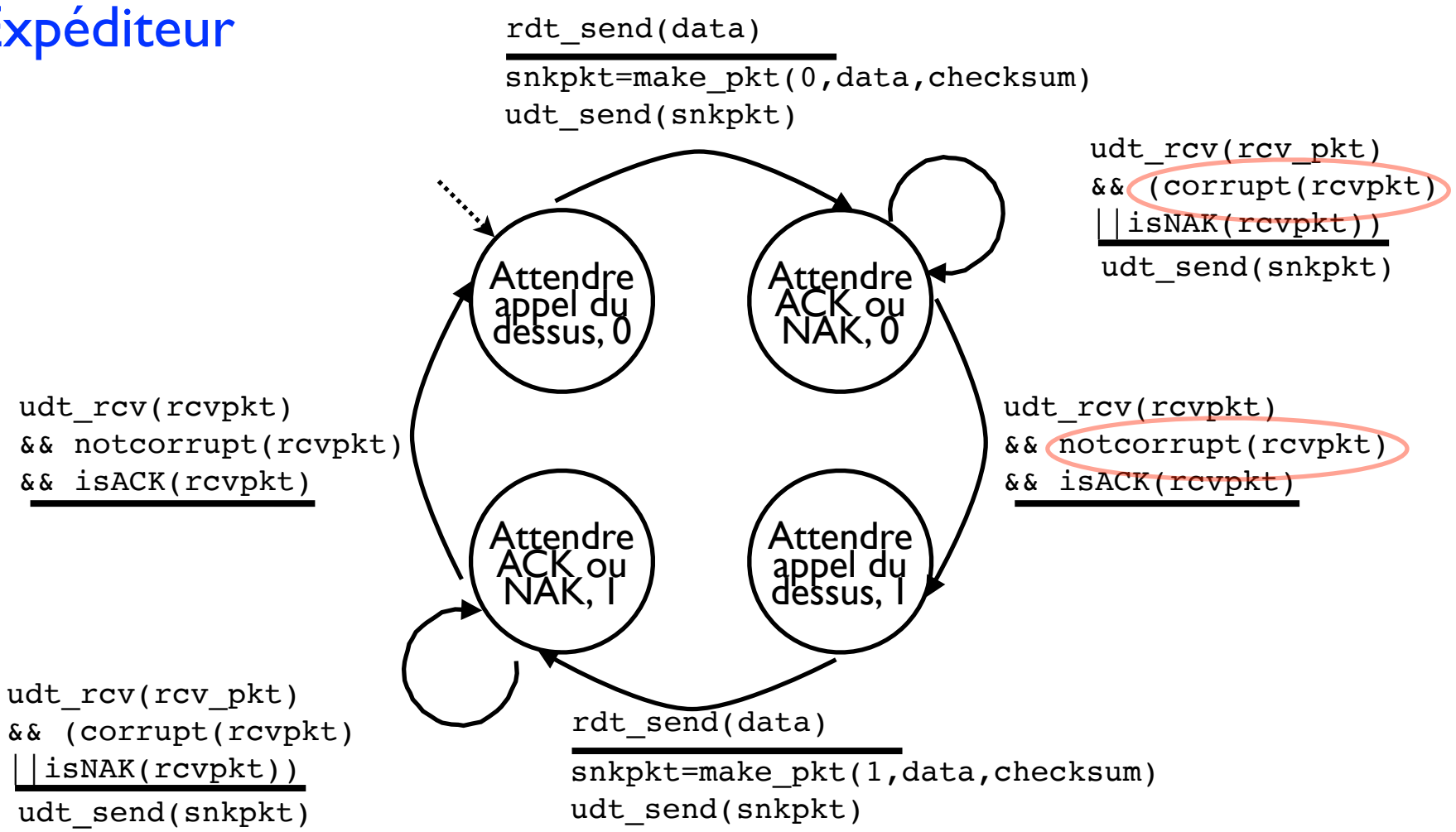
a) Erreur dans l'ACK

b) Erreur dans le segment

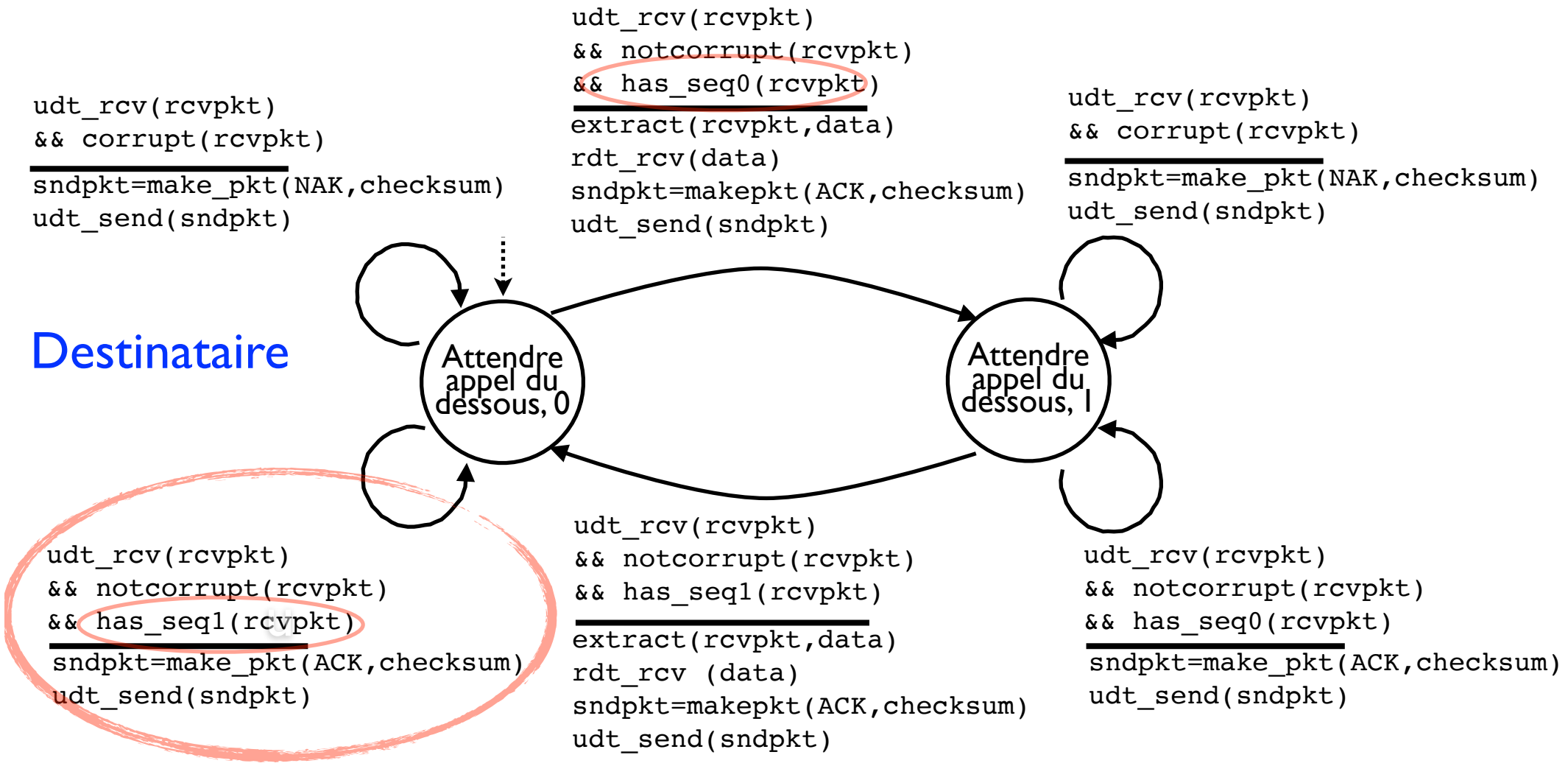
rdt2.1 : expéditeur gère les ACK/NAK erronés

- On double le nombre d'états des automates
  - pour mémoriser si le numéro de séquence du paquet courant vaut 0 ou 1

Expéditeur



rdt2.1 : destinataire gère les ACK/NAK erronés



duplicata ! (l'acquittement du paquet précédent a été erronné)

rdt2.2 : un protocole sans NAK

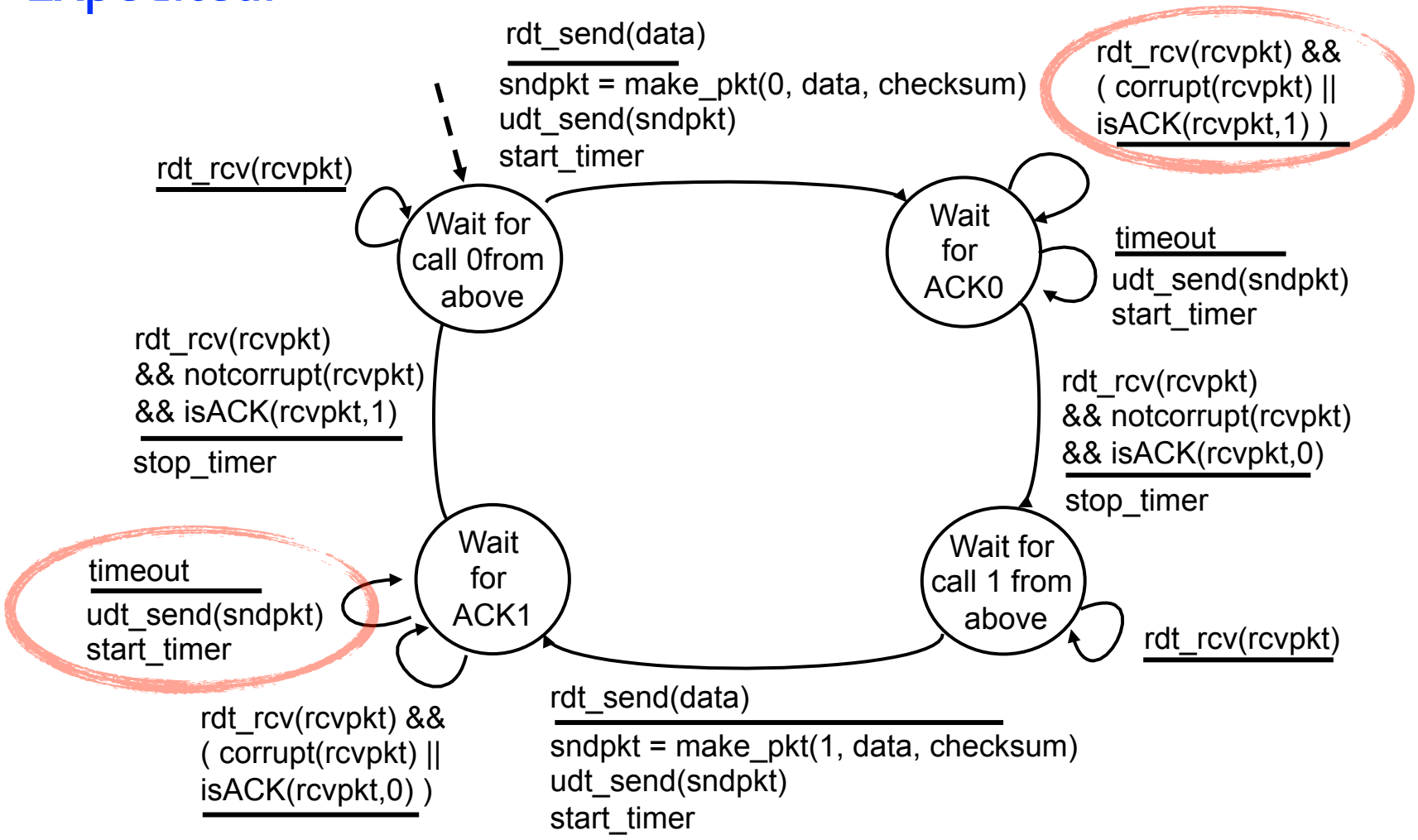
- Faire comme rdt2.1 mais uniquement avec des ACKs ?
- Au lieu d'un NAK...
  - ... le destinataire envoie un ACK associé au dernier segment correctement reçu
- Les ACKs doivent être numérotés
  - le destinataire doit explicitement inclure le #séquence du paquet dont il accuse la bonne réception
- Recevoir **2 ACK identiques ⇔ recevoir 1 NAK**
  - et donc déclenche la retransmission du paquet courant

rdt3.0 : canal avec erreurs et pertes

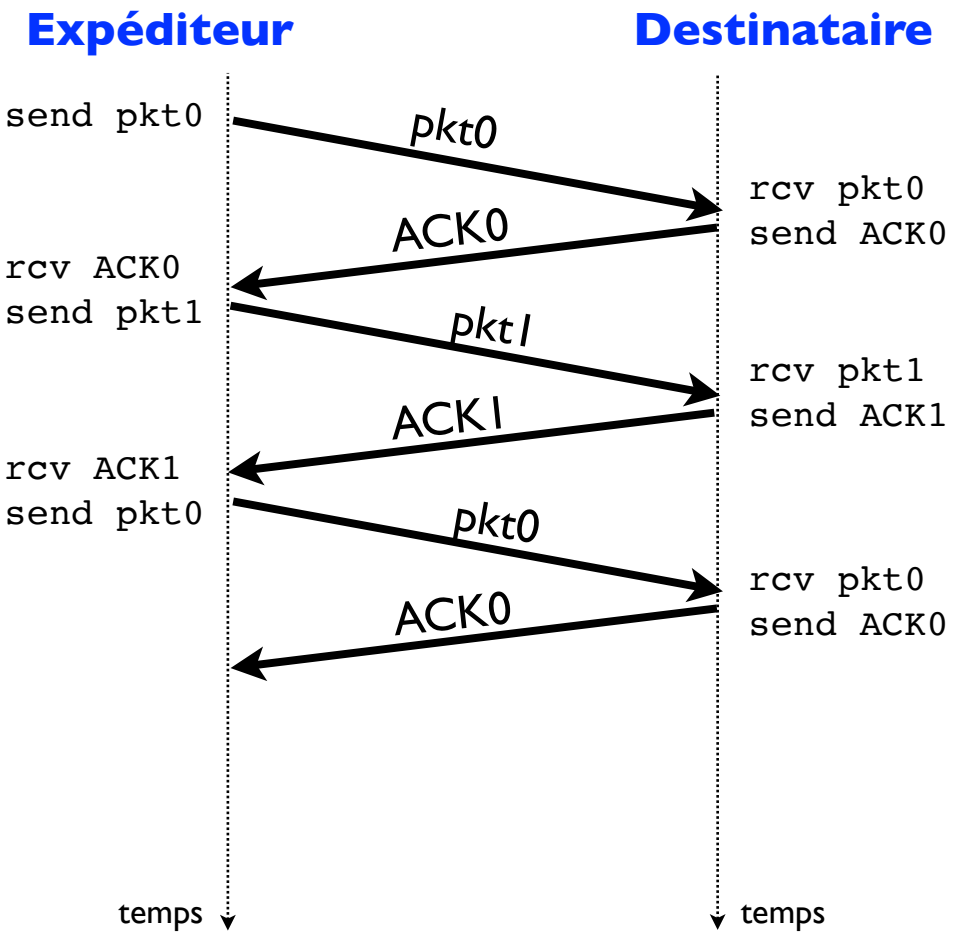
- Hypothèse : canal peut perdre des paquets (données et ACKs)
  - somme de contrôle, #séquence, ACK, retransmission : pas suffisants !
- Que faut-il de plus ?
  - Temporisateur ("timer")
  - L'expéditeur attend le retour de l'ACK
  - Si le temporisateur expire avant ("timeout") ⇒ il retransmet
  - Si le paquet ou l'ACK arrive simplement trop tard ???
    - paquets dupliqués mais #séquence pour y répondre
  - Le destinataire doit spécifier le #séquence du paquet qu'il acquitte
  - Si ACK arrive en erreur ou pas le bon ACK attendu
    - pas de réaction, on laisse le timer gérer le problème

rdt3.0 : canal avec erreurs et pertes

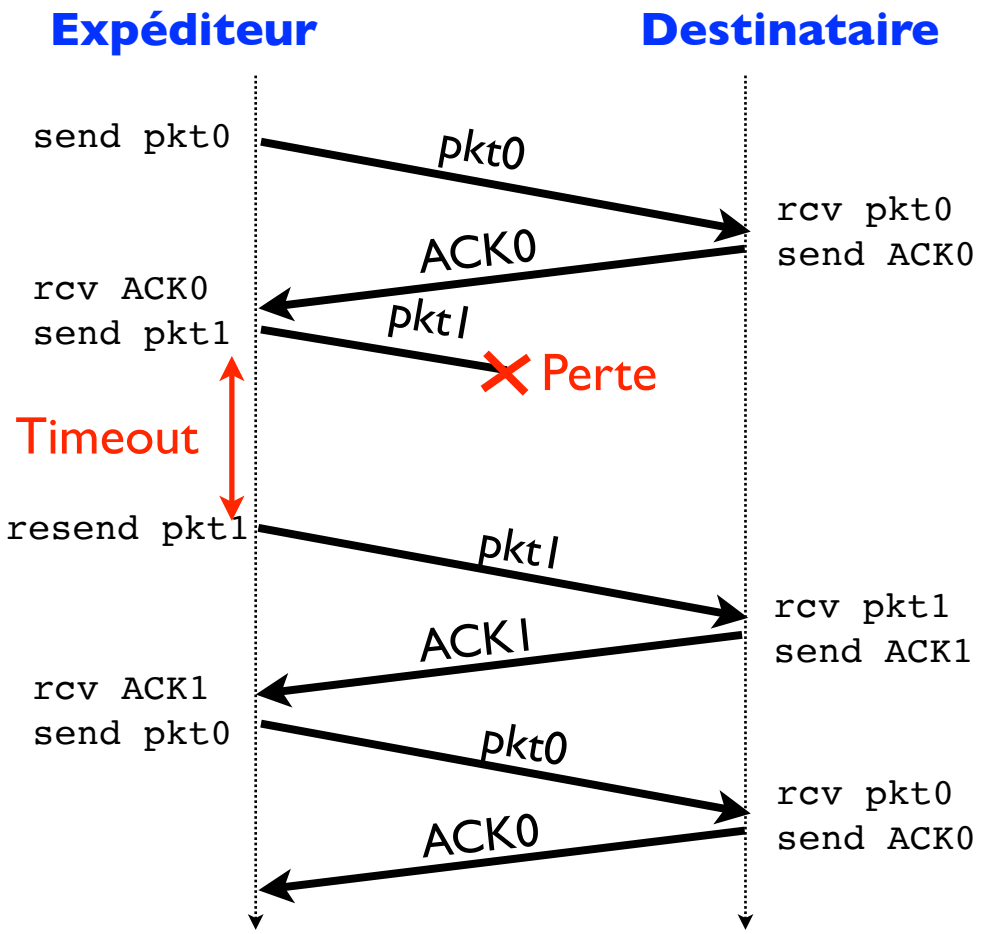
Expéditeur



rdt3.0 en action (1)

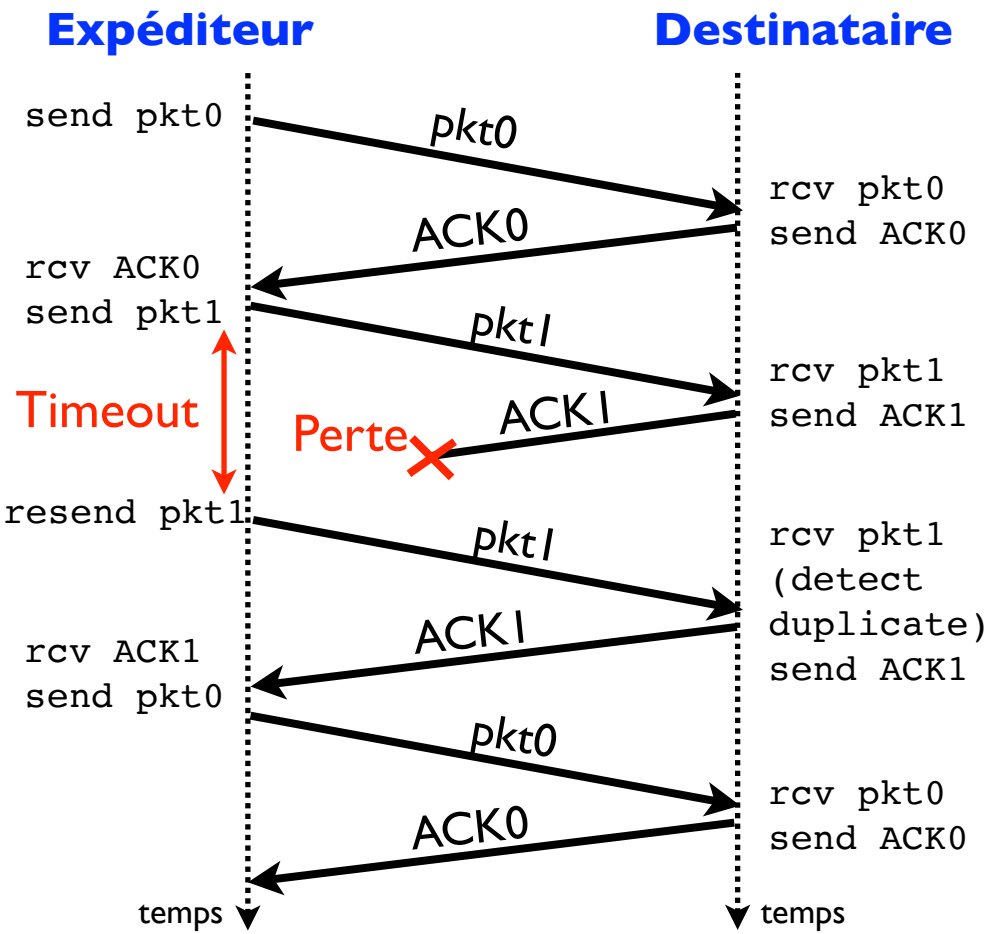


a) Sans perte

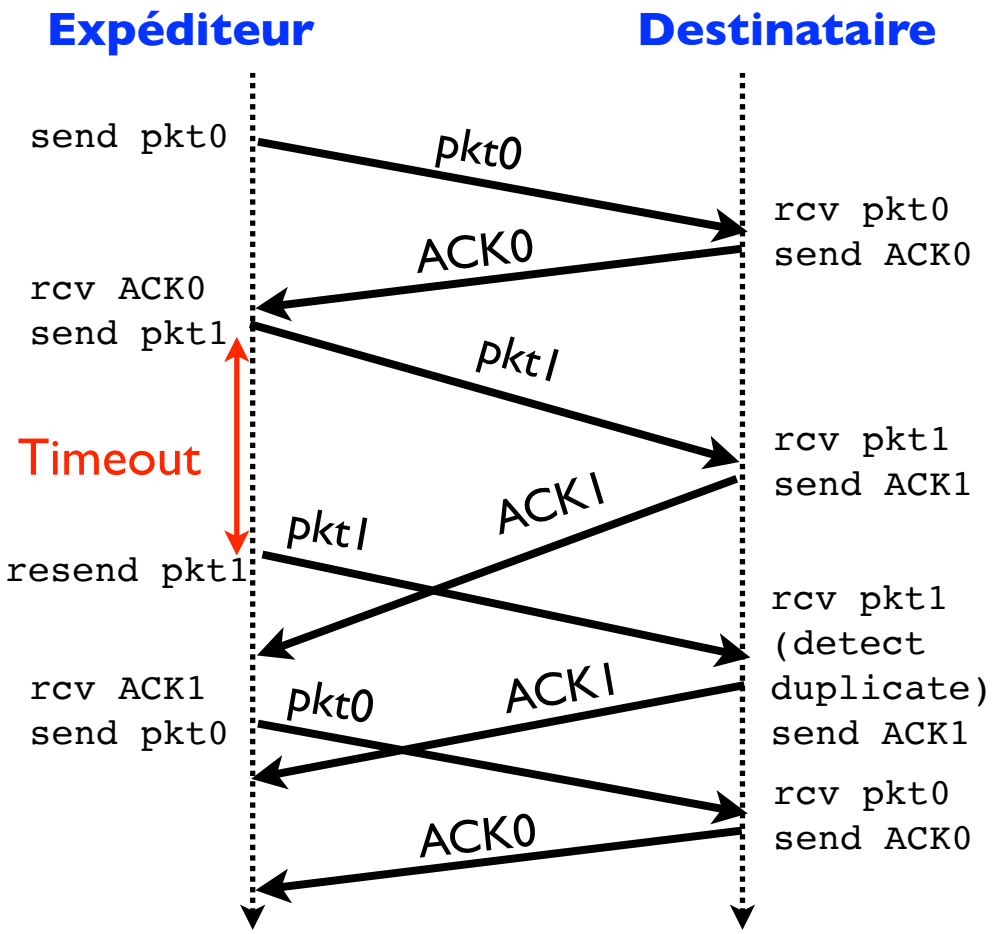


b) Paquet perdu

rdt3.0 en action (2)



c) ACK perdu



d) Expiration prématurée

Pièces maîtresse d'un protocole de transport de données fiable ("rdt")

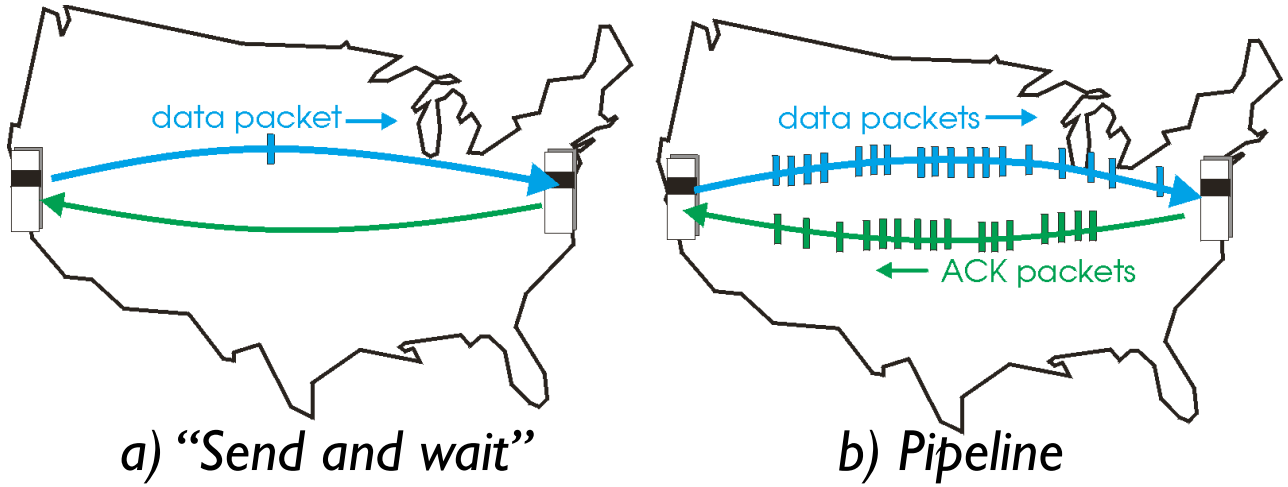
1. somme de contrôle
   - détecter les erreurs
2. accusés de réception (ACK & NAK)
   - boucle de contrôle
3. #séquence
   - détecter duplicata et erreurs sur les ACKs
4. "timer" (temporisateur)
   - pertes de paquets

Performances de rdt3.0

- rdt3.0 protocole fonctionnel mais performances ☹
  - à cause de l'approche "send and wait"
- Exemple
  - lien 1Gb/s → C, délai propagation 15ms, paquet 1Ko → L
  - U : utilisation du lien (proportion de temps à émettre) → efficacité
  - D_{utile} : débit utile
  - $t_{émission} = 8L/C = 8\mu s$
  - $U = t_{émission} / (t_{émission} + 2 \times d_{propag}) = 2.7\ 10^{-4}$
  - $D_{utile} = U \times C = 270\ 000$ b/s = 270 Kb/s
    - A comparer avec les 1 Gbps !
- Fonctionnel mais pas performant

Protocoles à anticipation (1)

- Pipeline
- Expéditeur peut transmettre plusieurs paquets à la suite sans attendre des accusés de réception
- Ce nombre de paquets = fenêtre d'anticipation



a) "Send and wait" *b) Pipeline*

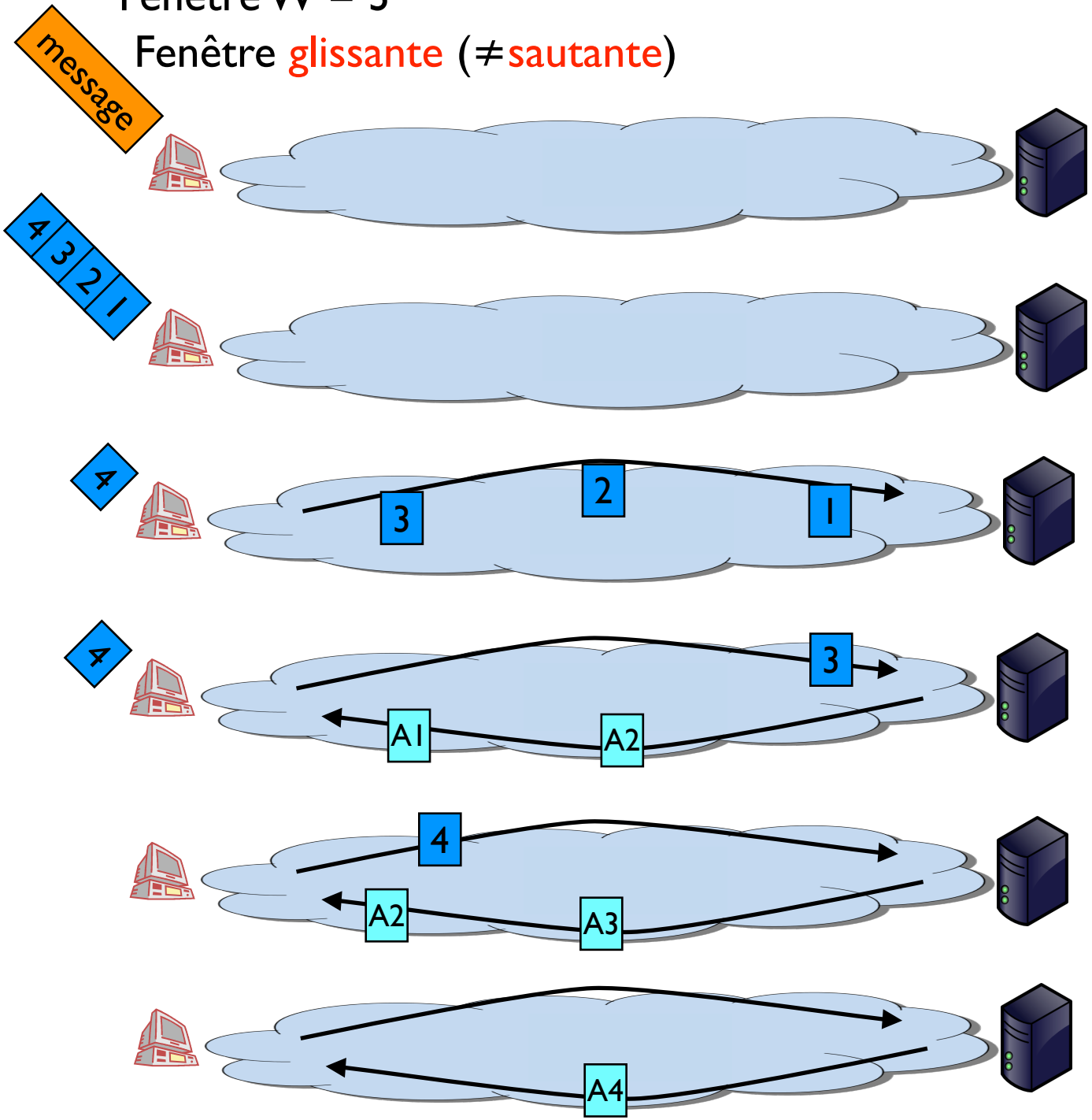
- Taille supposée de la fenêtre d'anticipation ?

1 34

Protocoles à anticipation (2)

Fenêtre W = 3

Fenêtre glissante (≠sautante)



1. message à envoyer

2. divisé en 4 segments

3. la fenêtre bloque le 4ème segment

4. les segments se "transforment" en ACK

5. l'arrivée de ACK 1 déclenche le départ du segment 4

6. le transfert prend fin

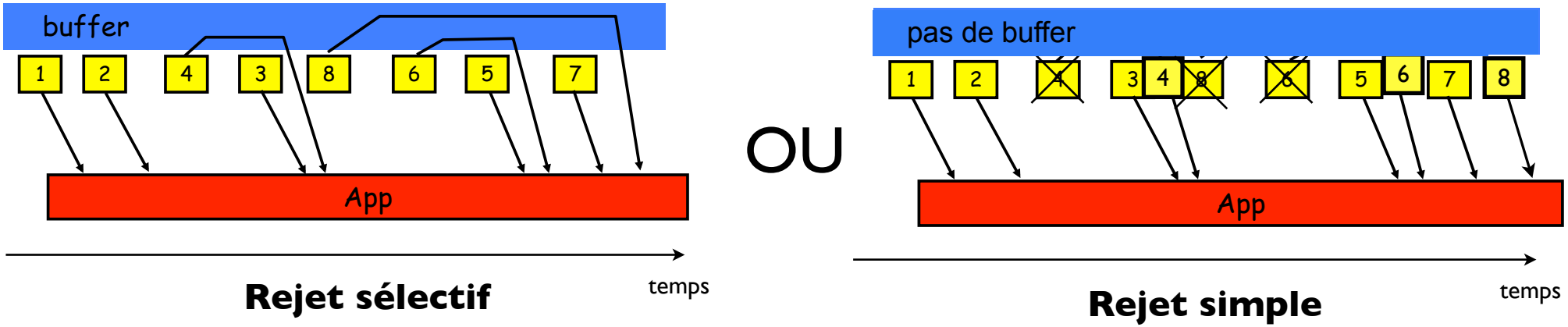
Protocoles à anticipation (3)

- Une meilleure utilisation des ressources réseaux
- Mais ...
  - Déséquencement possible des segments
  - Mémoire buffer chez l'expéditeur (et chez le destinataire ?)
  - Augmenter la gamme des #séquence

- Buffer émission : nécessaire ?
  - oui, pour sauvegarder les segments en cas de retransmission
- Buffer réception : nécessaire ?
  - non mais peut sauvegarder les segments déséquencés

Buffer de reséquencement

- Arrivées déséquencées des segments au niveau du récepteur : 1, 2, 4, 3, 8, 6, 5, 7



- Rejet Sélectif ("Selective Repeat")
  - La couche Transport maintient un buffer par connexion TCP
    - les paquets peuvent entrer déséquencés
    - et attendent jusqu'à être transmis dans l'ordre à la couche Application
- Rejet Simple ("Go-Back N")
  - La couche Transport ne maintient pas de buffer
    - Seuls les paquets bien séquencés sont acceptés

Plan

1. Services de la couche Transport
2. Multiplexage et démultiplexage
3. Transport sans connexion : UDP
4. Principes du transfert de données fiable
5. Contrôle de congestion TCP

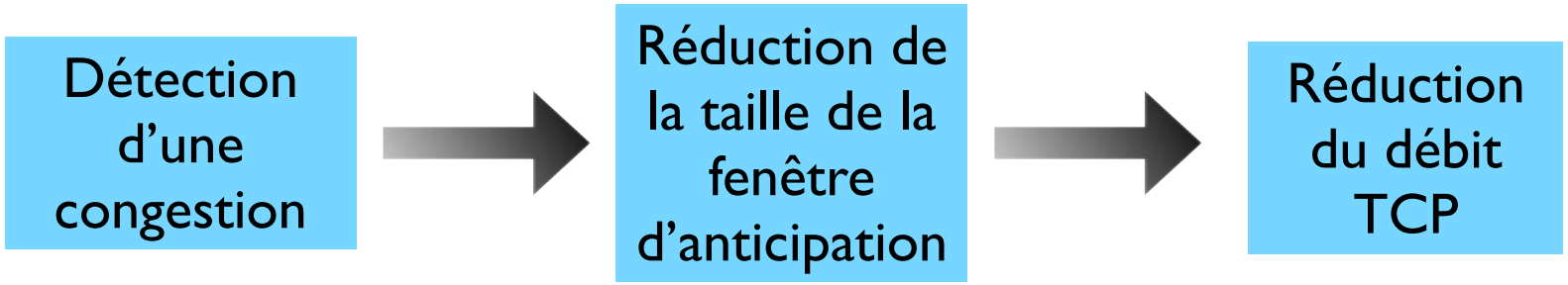
Problématique

- Ressources du réseau sont limitées
  - Capacité d'émission des liaisons
  - Capacité de traitement des nœuds
  - Capacité de stockage (buffers) des nœuds
- Lorsque le trafic soumis (charge) est trop important
  - Contention sur les ressources
  - Des files se forment dans les routeurs
  - Retards et pertes de paquets ↗
  - Phénomène de congestion
- Contrôle de congestion en 3 étapes

Étape 1 : détecter une congestion

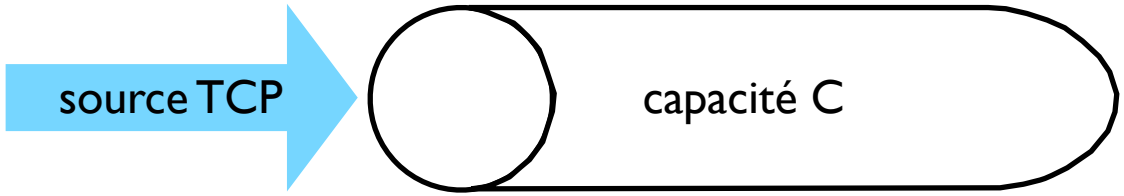
- Comment détecter une congestion ?
  - Sans assistance du réseau
  - Uniquement à partir des terminaux
  - par la perte de paquets, détectée elle-même par ?
    - “timeout”
    - 3 ACKs identiques
- Hypothèse fondamentale
  - Une congestion provoque des pertes de paquets
  - L’inverse est-il vrai ?
    - Non, les pertes ne sont pas toujours dues à une congestion

Étape 2 : comment réguler son débit ?

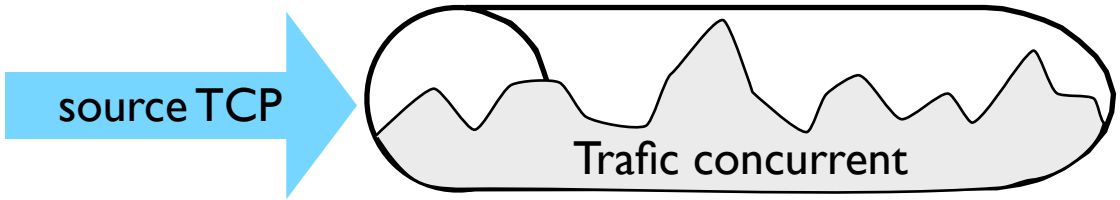


Étape 3 : à quel niveau réguler son débit ?

- À combien réguler le débit d'une source TCP ?
- Exemple 1 : un lien dédié de capacité C



  - débit souhaitable pour la source ?
  - proche de C
- Exemple 2 : un lien de capacité C partagé à plusieurs



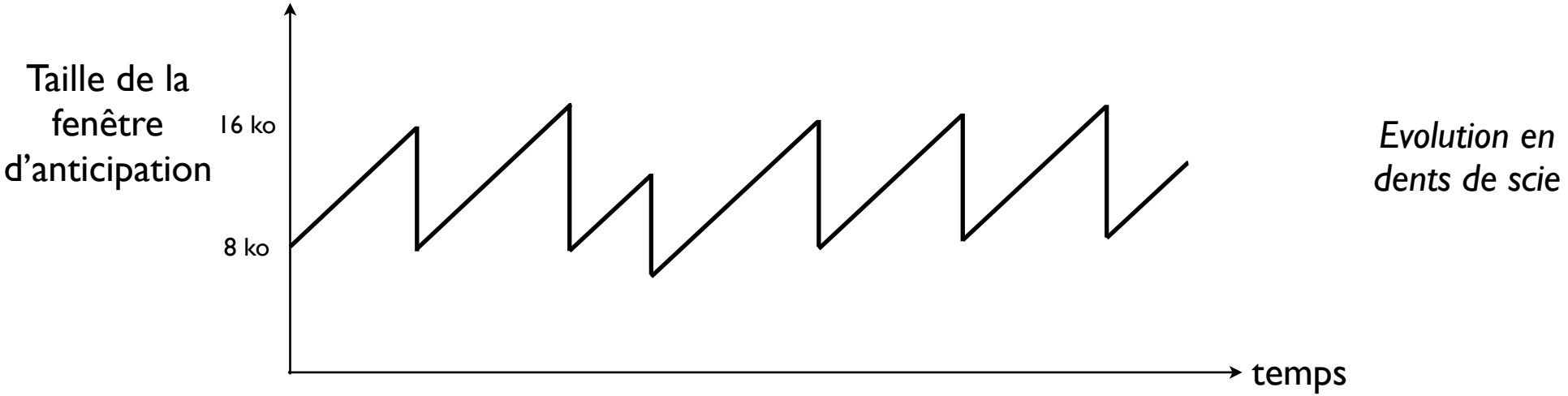
  - débit souhaitable pour la source ?
  - capacité résiduelle (capacité disponible)
- Problème : quantité inconnue et dynamique. Comment la découvrir ?

Hausse additive, baisse multiplicative

- Principe : détecter la capacité disponible sur le chemin
  - augmenter progressivement le débit de TCP en agrandissant la taille de sa fenêtre d'anticipation jusqu'à atteindre le débit max supporté
  - comment savoir qu'on a atteint le max ? une perte se produit
- Hausse additive : augmenter la taille de la fenêtre d'anticipation d'1 segment après chaque RTT
- Baisse multiplicative : diviser la taille de la fenêtre d'anticipation par 2

➡ AIMD : "Additive Increase Multiplicative Decrease"

➡ "Congestion Avoidance"



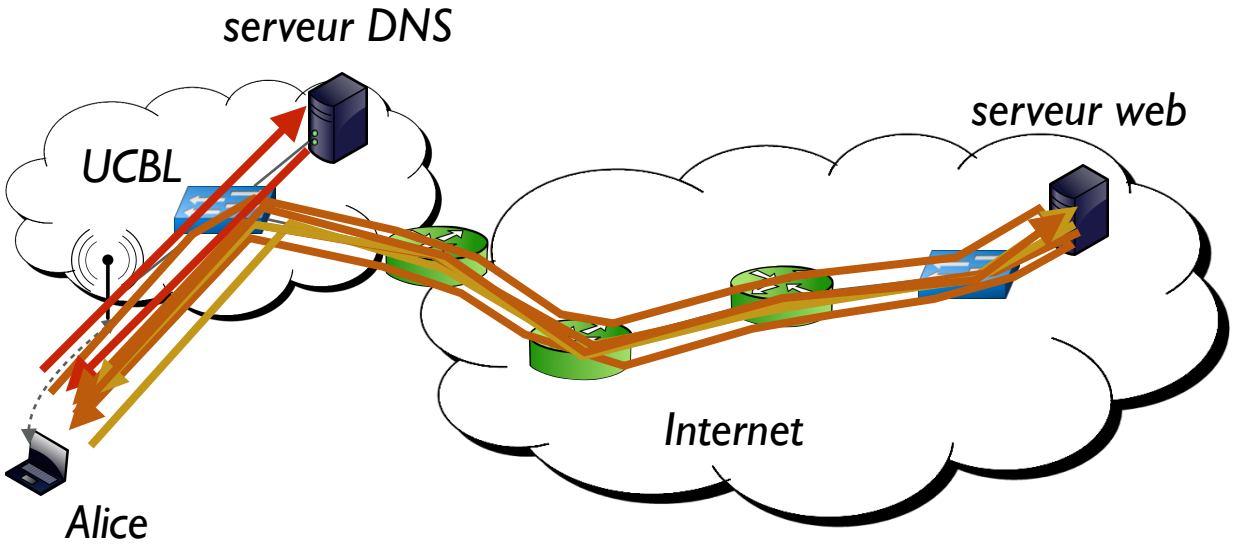
Evolution en dents de scie

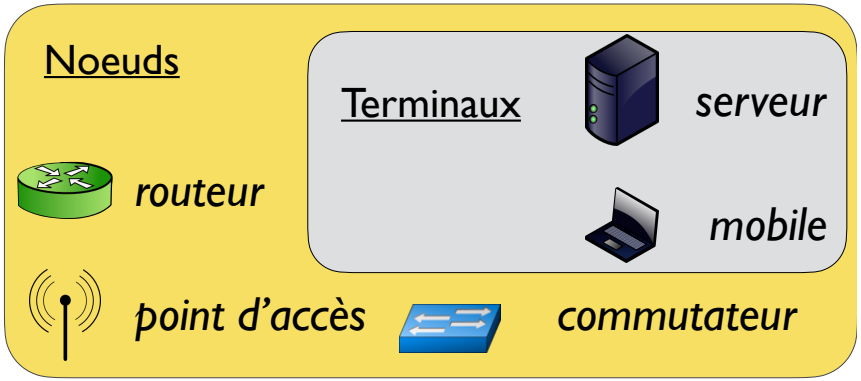
Synthèse : vue d'ensemble de TCP

[RFCs: 793, 1122, 1323, 2018, 2581]

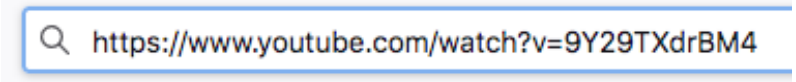
- Orienté connexion
  - échange d'information au début de la connexion ("handshaking")
  - expéditeur et destinataire fixent les paramètres du transfert
- Mode duplex
  - les données peuvent circuler dans les deux sens
- Point-à-point
  - entre un expéditeur & un destinataire
- Livraison fiable et séquencée des données
  - buffers d'émission et de réception
  - taille des segments fixée par
    - MSS :"Maximum Segment Size" (hors en-tête)
- Pipeliné
  - la fenêtre d'anticipation est dynamique

Exemple 3





Alice



Résolution DNS : Requête DNS : segment UDP - Quelle adresse IP pour www.youtube.com/watch?v=9Y29TXdrBM4 ?

Réponse DNS : *segment UDP - 172.217.171.238*

Requête / réponse http : Phase de négociation TCP

Requête HTTP : segment TCP - Envoie moi le contenu demandé

Acquittement TCP de la requête HTTP

Réponse HTTP : segment TCP - Voici le contenu demandé

Acquittement TCP de la réponse HTTP

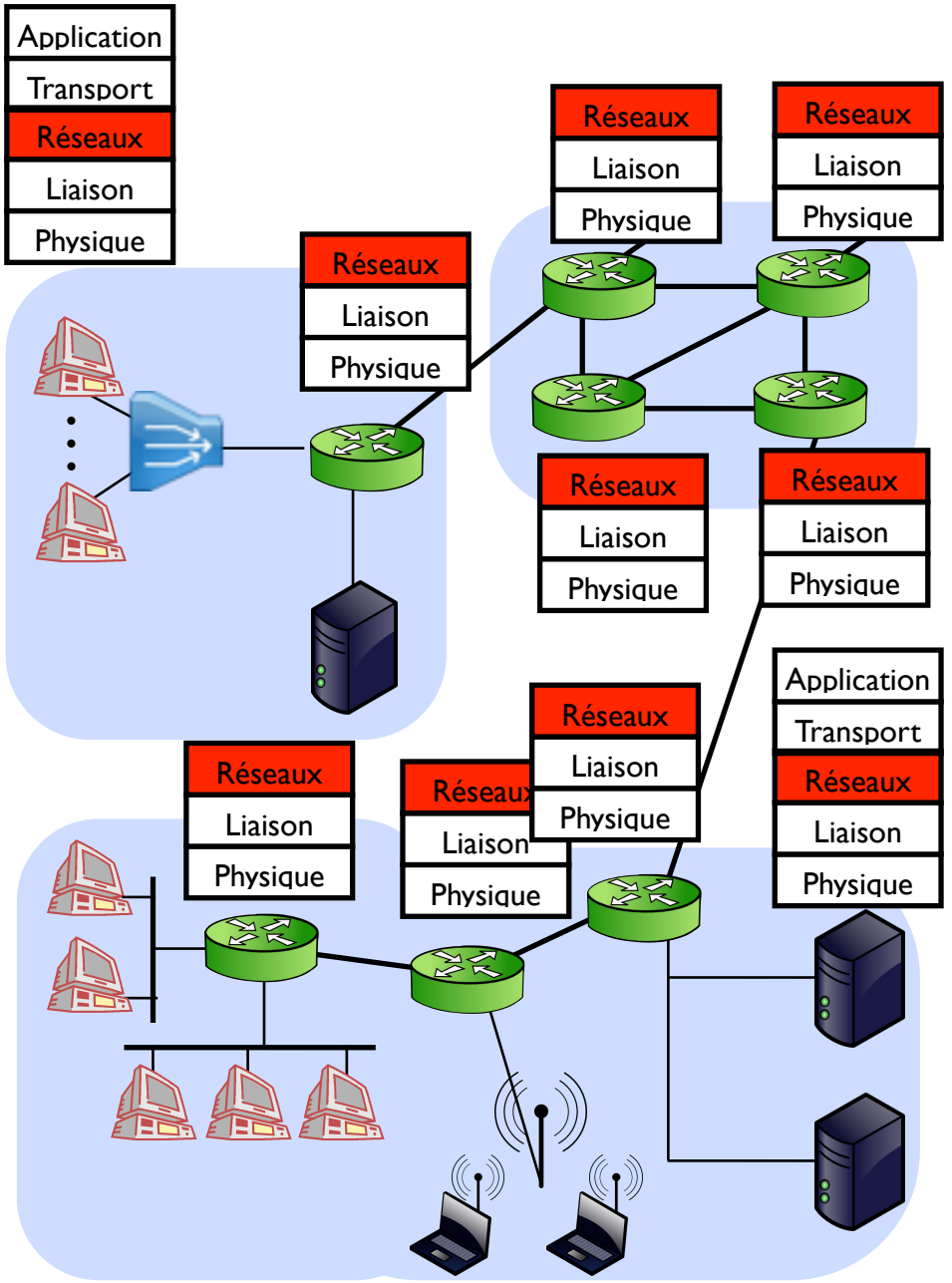
Couche Réseau

Plan

1. Introduction
2. IP : Internet Protocol
3. Tables d'acheminement
4. Algorithmes de routage
5. Le routage dans Internet

La couche Réseau

- Permet l'acheminement des segments de l'expéditeur jusqu'au destinataire
- Expéditeur
  - encapsule les segments à émettre dans des datagrammes
- Récepteur
  - transmet les segments reçus à la couche Transport
- Tous les terminaux et routeurs
  - implémentent la couche Réseau
- Les routeurs examinent les en-têtes des datagrammes



Les services essentiels de la couche Réseau

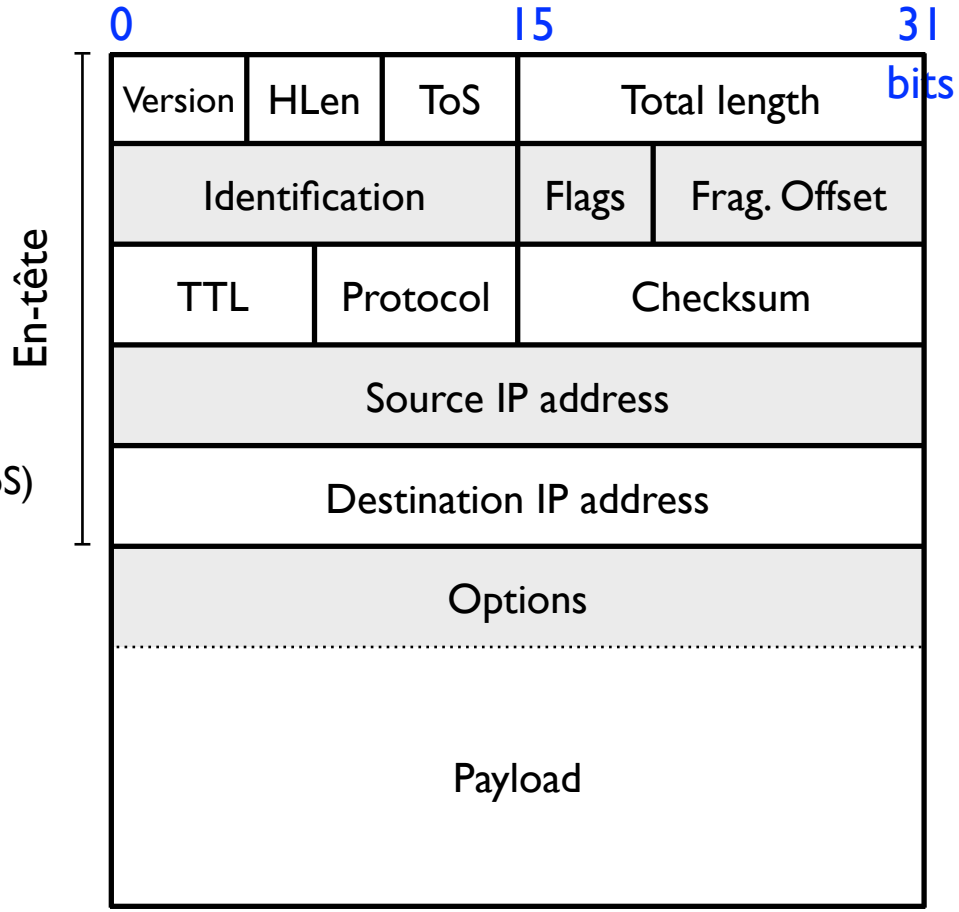
- Adressage
  - Adresse = identifiant topologique d'un noeud
    - Dépend de la position dans le réseau
    - Utile pour router
    - (En principe) unique
- Acheminement (Forwarding)
  - Commuter un paquet d'une interface à une autre
  - Interroger une table d'acheminement (table de routage)
- Routage
  - Déterminer le chemin de bout-en-bout à suivre pour un paquet depuis la source jusqu'à la destination
  - Algorithme de routage

Plan

1. Introduction
2. IP : Internet Protocol
3. Tables d'acheminement
4. Algorithmes de routage
5. Le routage dans Internet

Datagramme IPv4 (1)

- Format d'un datagramme
- En-tête de 20 octets
- Version : numéro de version d'IP
- HLen : "Header length"
  - exprimé en mots de 32 bits
  - sans option, HLen = 5 mots (5×32bits = 20 octets)
- ToS : "Type of Service"
  - indique la façon dont le paquet doit être traité (QoS)
- Total Length
  - exprimé en nombre d'octets
  - comprend l'en-tête
  - codé sur 16 bits
    - taille max d'un datagramme = ?
      - $2^{16}-1$ = 65 535 octets



Datagramme IPv4 (2)

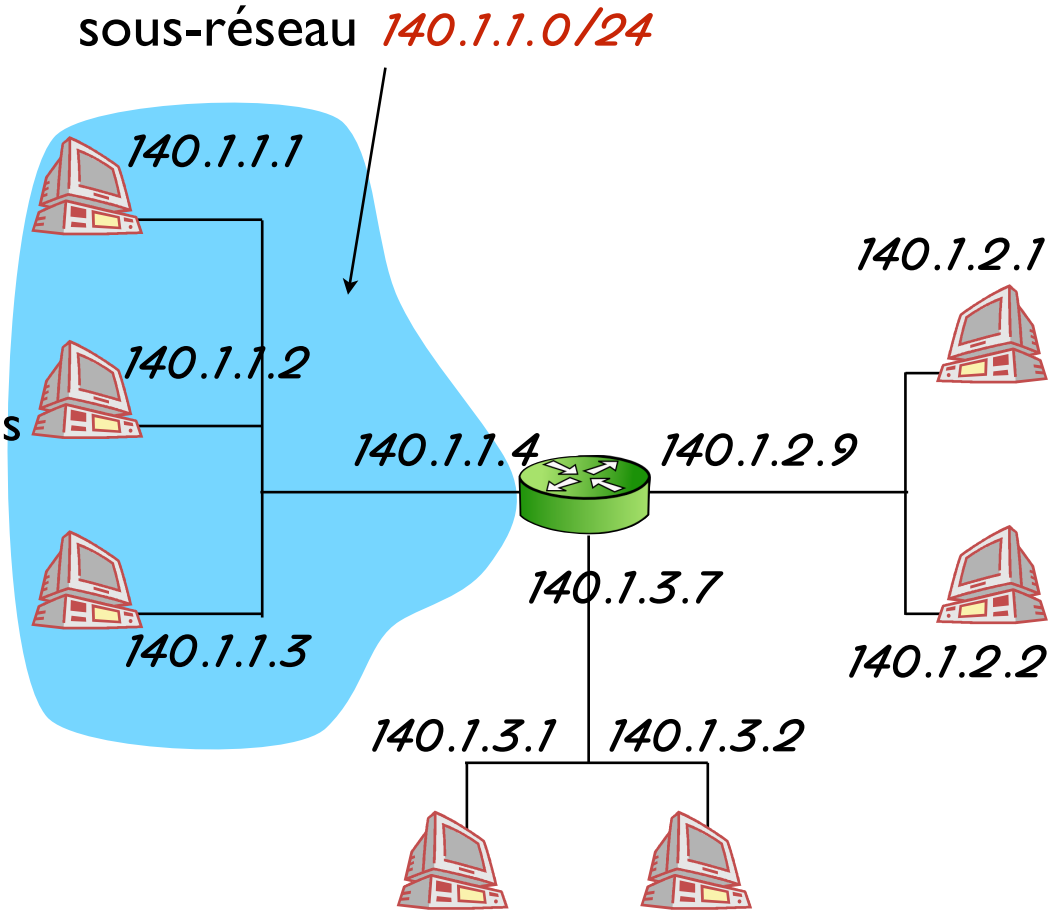
- TTL : “Time To Live”
  - valeur initiale fixée par l’émetteur
  - puis décrémentée de 1 à chaque à saut (lien de communication) traversé dans le réseau
  - si TTL = 0, le paquet est supprimé
  - Utilité ?
    - Permet d’éviter qu’un paquet tourne indéfiniment dans le réseau (par exemple en présence d’une boucle de routage)
- Protocol
  - Identifier le protocole encapsulé dans IP
    - 06 : TCP - 17 : UDP

Datagramme IPv4 (3)

- Checksum : somme de contrôle
  - détection des erreurs de transfert
  - calculé sur l'en-tête IP seulement
  - complément à 1 de la somme de l'entête
  - en cas d'erreur, le paquet est supprimé
- Source IP Address & Destination IP Address
  - codées sur 32 bits chacune
- Options : usage rare
  - sécurité, routage à la source ou estampille temporelle
- Datagramme possède des infos pour être acheminé de sa source à sa destination
  - les datagrammes d'un flux sont « autonomes » les uns par rapport aux autres

Adresses IPv4

- Adresse IP sur 32 bits
  - identifiant d'une interface réseau
- Interface réseau
  - connexion entre un noeud et un lien
  - un routeur a plusieurs interfaces
  - un terminal a une ou plusieurs interfaces
  - chaque interface a son adresse IP
- Adresse IP
  - partie réseau : bits de poids fort
  - partie hôte : bits de poids faible
  - masque : taille de la partie réseau
  - a.b.c.d/x où
    - x = masque de sous-réseau
    - x indique le nombre de bits dans la partie réseau



10001100 01001101 00000001 00000000

Partie réseau | Partie hôte

Sous-réseaux

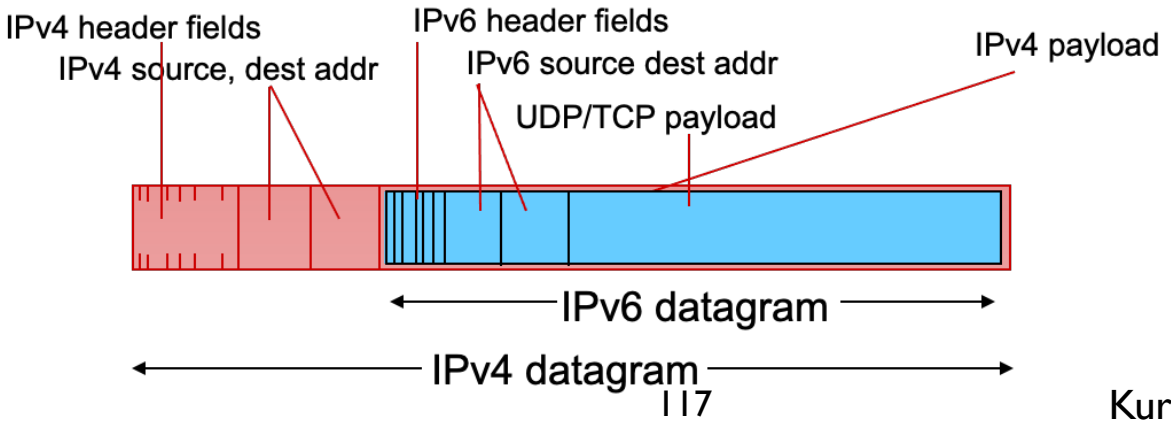
- Sous-réseaux (“Subnets”)
  - Ensemble des interfaces dont les adresses ont la même partie réseau
  - Les interfaces d’un même sous-réseau peuvent communiquer directement (sans l’intervention d’un routeur)
- Comment trouver pratiquement les sous-réseaux existants ?
  - En désactivant les interfaces des routeurs
  - Chaque nouveau réseau isolé est un sous-réseau
- Possibilité de
  - découper un réseau en plusieurs sous-réseaux
  - d’agréger des sous-réseaux

Comment obtient-on son @IP ?

- Comment un terminal obtient-il son adresse IP ?
  - "Codé en dur" par un admin système dans un fichier
    - Unix : /etc/rc.config
    - Windows : TCP/IP dans Panneau de Configurations
  - DHCP : Dynamic Host Configuration Protocol
    - Obtient son adresse dynamiquement par un serveur
    - "Plug and Play"
- Comment une organisation obtient-elle son préfixe ?
  - Elle demande à son FAI de lui allouer une portion de son espace d'adresses
- Comment un FAI reçoit-il son bloc d'adresses ?
  - ICANN : Internet Corporation for Assigned Names and Numbers
    - alloue les adresses
    - affecte les noms de domaines
    - gère le DNS
    - arbitre les conflits

IPv6

- « Nouveau » protocole IP pour faire face à la pénurie d'adresses IPv4
- Taille d'une adresse IPv6 = 128 bits
  - 32 bits pour IPv4
- Autres champs
  - Traffic class : pour faire de la différenciation de service
  - Flow label : pour identifier les datagrammes appartenant à même flux pour un traitement plus « efficace »
- Champ checksum disparaît
  - on laisse d'autres couches s'en occuper pour plus d'efficacité
- Migration IPv6
  - a démarré en 2003
  - toujours dans une phase de cohabitation entre IPv4 et IPv6
  - mécanisme de tunneling : encapsulation des datagrammes IPv6 dans des datagrammes IPv4 sur les routeurs IPv4



Kurose et al.

Plan

1. Introduction
2. IP : Internet Protocol
3. Tables d'acheminement
4. Algorithmes de routage
5. Le routage dans Internet

Table de d'acheminement (1)

- Tables de d'acheminement pour les datagrammes
  - #Entrées = #@ IP ?
  - Avec 4 milliards (2^{32}) d'@ IPv4, recherche serait trop longue
- Solution ?
  - Une entrée ≠ une adresse IP
  - Une entrée = un ensemble d'adresses IP = un sous-réseau
    - Format : @ IP / masque
    - `192.168.4.0/24` désigne les 256 (=2^8) adresses de `192.168.4.0` à `192.168.4.255`
  - Désigner toutes les adresses possibles ?
    - `0.0.0.0/0` → route par défaut

Table d'acheminement (2)

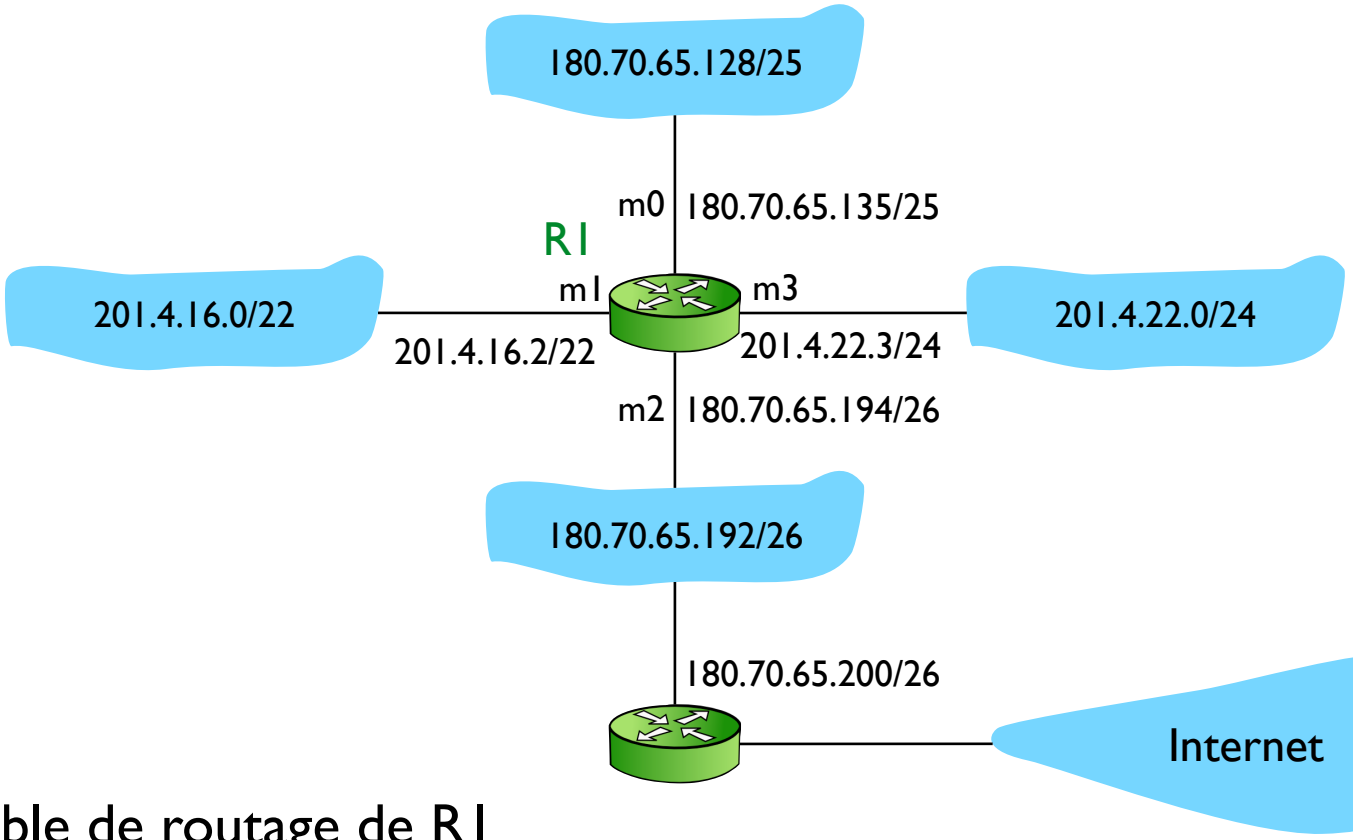


Table de routage de R1

Adresse réseau	Adresse du prochain saut	Interface de sortie
180.70.65.192/26	———	m2
180.70.65.128/25	———	m0
201.4.22.0/24	———	m3
201.4.16.0/22	———	m1
0.0.0.0/0	180.70.65.200	m2

Accessible sans routeur

Noté aussi 0.0.0.0

Table d'acheminement (3)

- Que se passe-t-il si une adresse IP vérifie plusieurs adresses de sous-réseau ?

Adresse réseau	Plage d'adresses	Adresse du prochain saut	Interface de sortie
192.168.0.0/28	192.168.0.0 à 192.168.0.15	———	0
192.168.0.0/24	192.168.0.0 à 192.168.0.255	———	1
192.168.0.0/22	192.168.0.0 à 192.168.3.255	———	2
0.0.0.0/0	0.0.0.0 à 255.255.255.255	192.168.0.1	0

- Exemple : 192.168.0.7
  - on choisit la première qui apparaît dans la table de routage ?
  - on tire au hasard ?
  - on choisit l'entrée la plus récente dans la table d'acheminement ?

Table d'acheminement (4)

- Si plusieurs choix dans la table sont possibles, on choisit la plus spécifique
  - Algorithme du Plus Long Préfixe Partagé
  - Il suffit donc de parcourir les entrées de la table par les préfixes les plus longs

Table d'acheminement (5)

- Donc, seuls les préfixes suffisent dans les tables

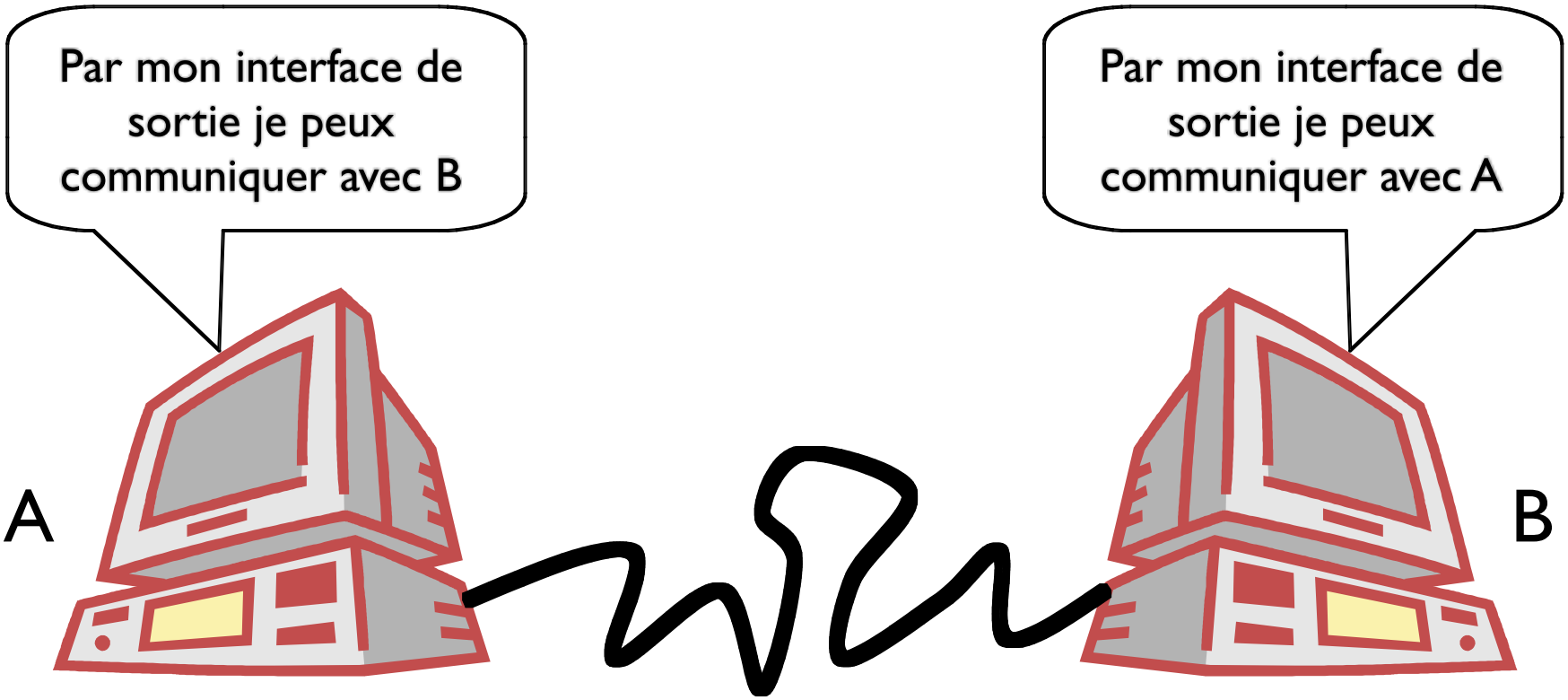
Adresse réseau	Préfixe des adresses réseaux	Adresse du prochain saut	Interface de sortie
192.168.0.0/28	11000000 10101000 00000000 0000	———	0
192.168.0.0/24	11000000 10101000 00000000	———	1
192.168.0.0/22	11000000 10101000 000000	———	2
0.0.0.0/0	-	192.168.0.1	0

- Exemples, quelles interfaces de sortie pour ?
  - Adresse destination : 192.168.0.17
    11000000 10101000 00000000 00010001 → Interface : 1
  - Adresse destination : 192.168.0.7
    11000000 10101000 00000000 00000111 → Interface : 0
  - Adresse destination : 192.168.4.17
    11000000 10101000 00000100 00010001 → Interface : 0

Plan

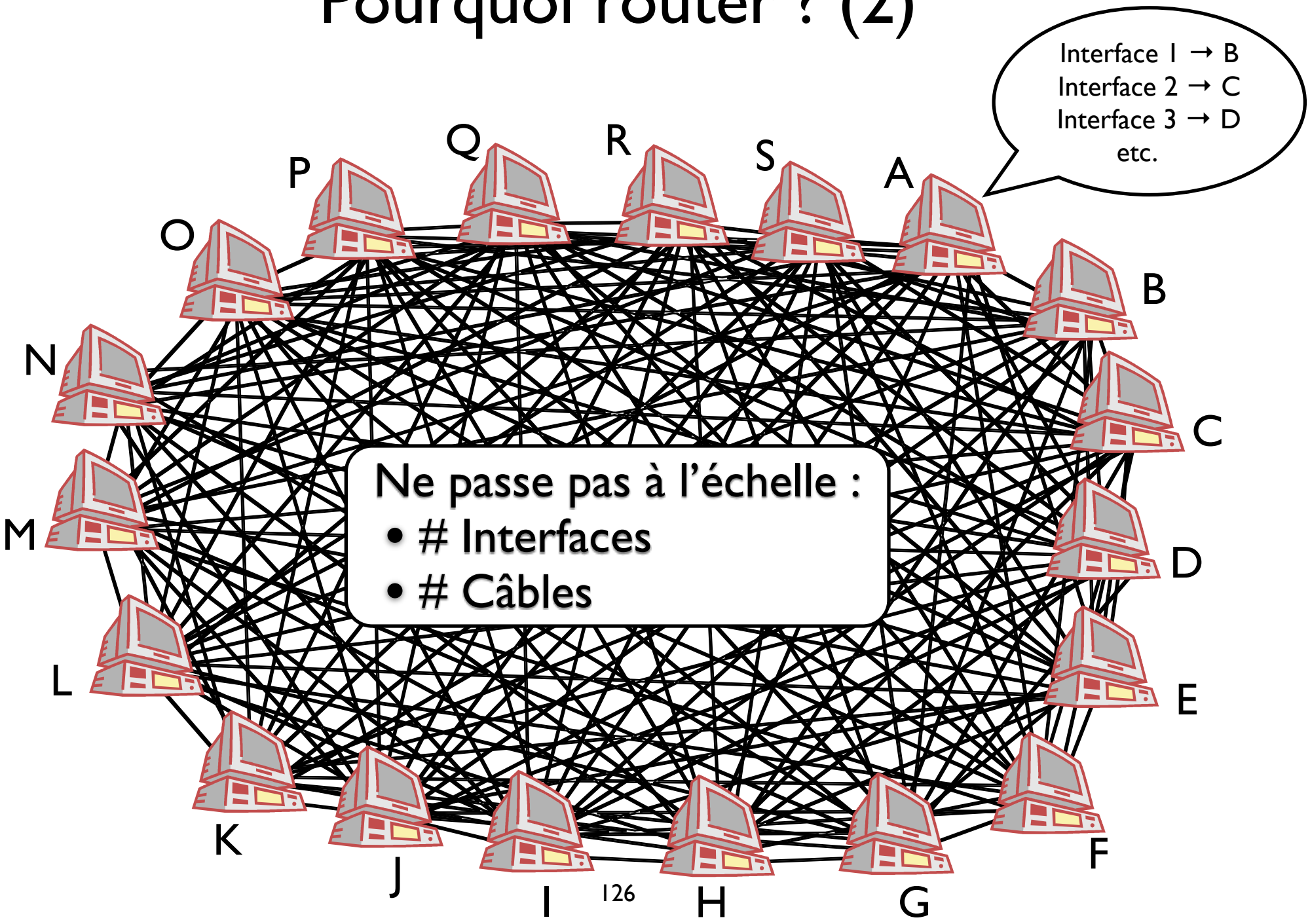
1. Introduction
2. IP : Internet Protocol
3. Tables d'acheminement
4. Algorithmes de routage
5. Le routage dans Internet

Pourquoi router ? (1)

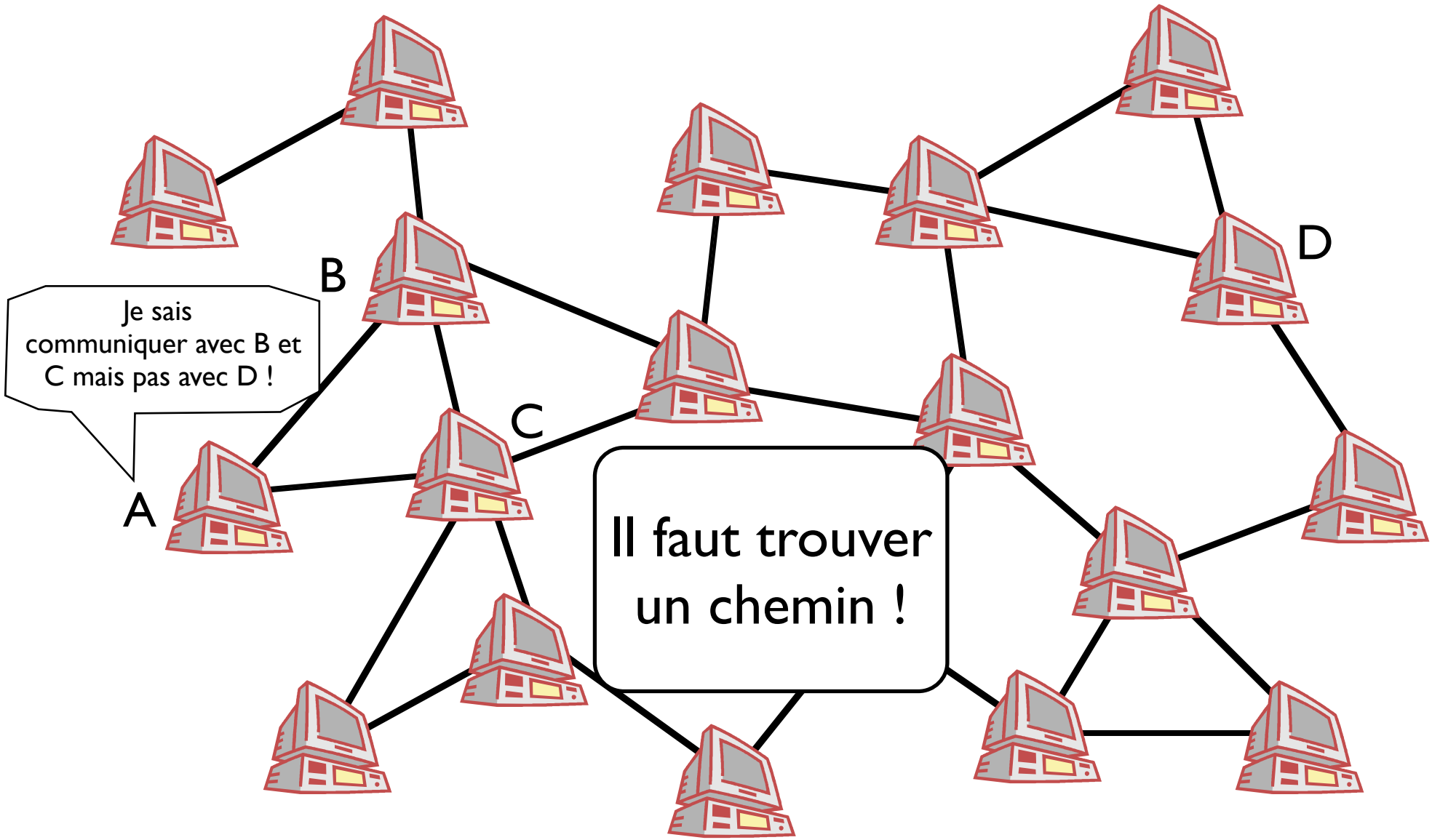


1 câble

Pourquoi router ? (2)



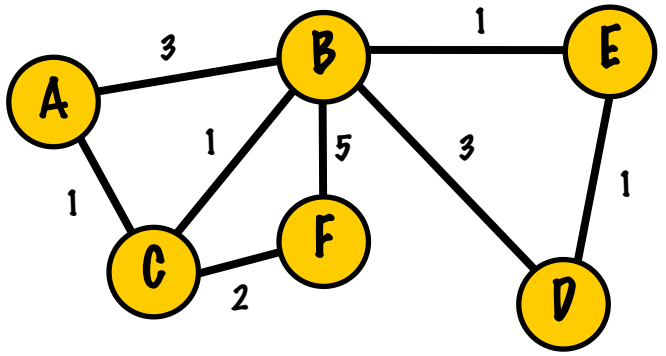
Pourquoi router ? (3)



Problème de plus court chemin (1)

- Théorie des graphes
- Un graphe comprend
  - des noeuds
  - des liens
- Coûts des liens
  - identiques
    - tous égaux à 1
  - différents et constants
    - par ex. inversement proportionnels à leur capacité d'émission (exemple Cisco)
  - différents et dynamiques
    - liés à une mesure de congestion

- Coût d'un chemin
  - Composition des coûts des liens qui le composent
  - Par ex.
    - Coût (A→C→B→D→E) = c(A,C) + c(C,B) + c(B,D) + c(D,E) = 6
- Comment trouver le chemin d'un noeud à un autre de coût minimal ?



Problème de plus court chemin (2)

- À la main
  - Uniquement pour de petits graphes
  - Sinon, grâce à un algorithme (automatisable)
- Deux grandes approches pour le routage dynamique
  - Algorithmes par états de lien
    - Chaque noeud a une connaissance complète du réseau
  - Algorithmes à vecteur de distances
    - Chaque noeud connait uniquement ses voisins et le coût de ses liens

Routage par états de lien

- Algorithme de Dijkstra (1959)
- Hypothèse
  - Chaque noeud connaît entièrement la topologie du réseau
- Algorithme
  - exécuté pour chaque noeud du réseau
  - retourne un arbre des plus courts chemins
  - et donc, sa table d'acheminement
- Exercice en TD

Routage par vecteur de distances (1)

- Notations
  - $d_X(Y)$: coût du plus court chemin de X vers Y
  - $\mathbf{V_x} = \{V_1, V_2, \ldots, V_M\}$: l'ensemble des noeuds voisins de X
- Equation de Bellman-Ford
  - $d_X(Y) = \min_{\mathbf{V_x}} (c(X,V_i) + d_{V_i}(Y))$
  - coût exact du plus court chemin de X vers Y
- On va manipuler $D_X(Y)$ = estimation courante de $d_X(Y)$
- Recherche d'un point fixe par itération

Routage par vecteur de distances (2)

- Au niveau de chaque routeur
  - Table d'acheminement
    - | Destination | Coût | Saut suivant |
  - Vecteur de distances
    - | Destination | Coût |
    - envoyé à ses voisins et reçu de ses voisins
- Exercice en TD

Plan

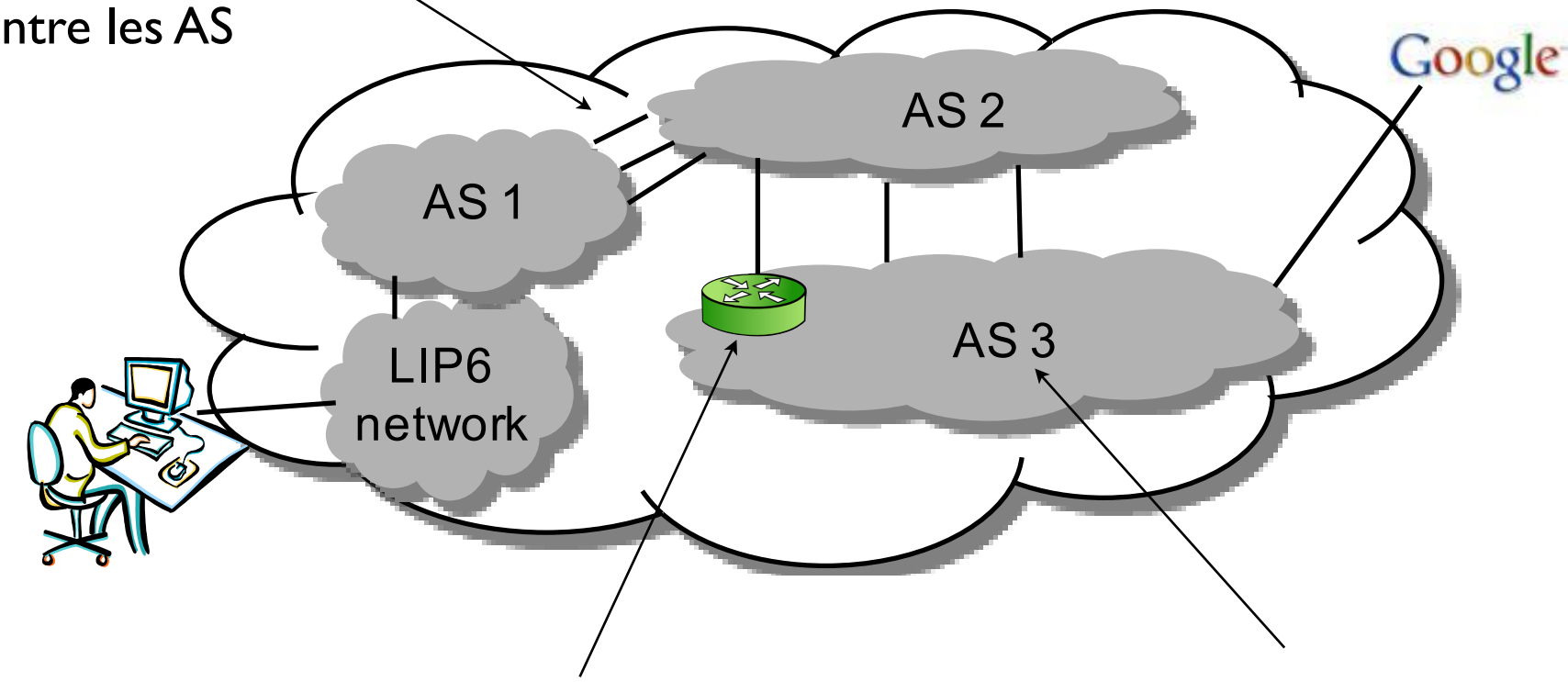
1. Introduction
2. IP : Internet Protocol
3. Tables d'acheminement
4. Algorithmes de routage
5. Le routage dans Internet

Routage hiérarchique (1)

- Internet est une fédération de réseaux autonomes interconnectés
  - "Autonomous Systems" - AS
  - chaque administrateur est maître de son réseau et de son routage
- Échelle gigantesque
  - + 800 000 préfixes et +100 000 AS en 2021
  - taille ingérable des tables de routage (même en agrégeant les @)
- Confidentialité
  - topologie interne doit rester secrète
  - ainsi que les accords négociés avec les autres AS
- Politiques et intérêts divergents
  - pas de métriques standardisées pour le coût d'un lien
  - contrôler d'où vient et où part le trafic
  - décharger son réseau au détriment des autres

Routage hiérarchique (2)

routage inter-AS
détermine les chemins
entre les AS

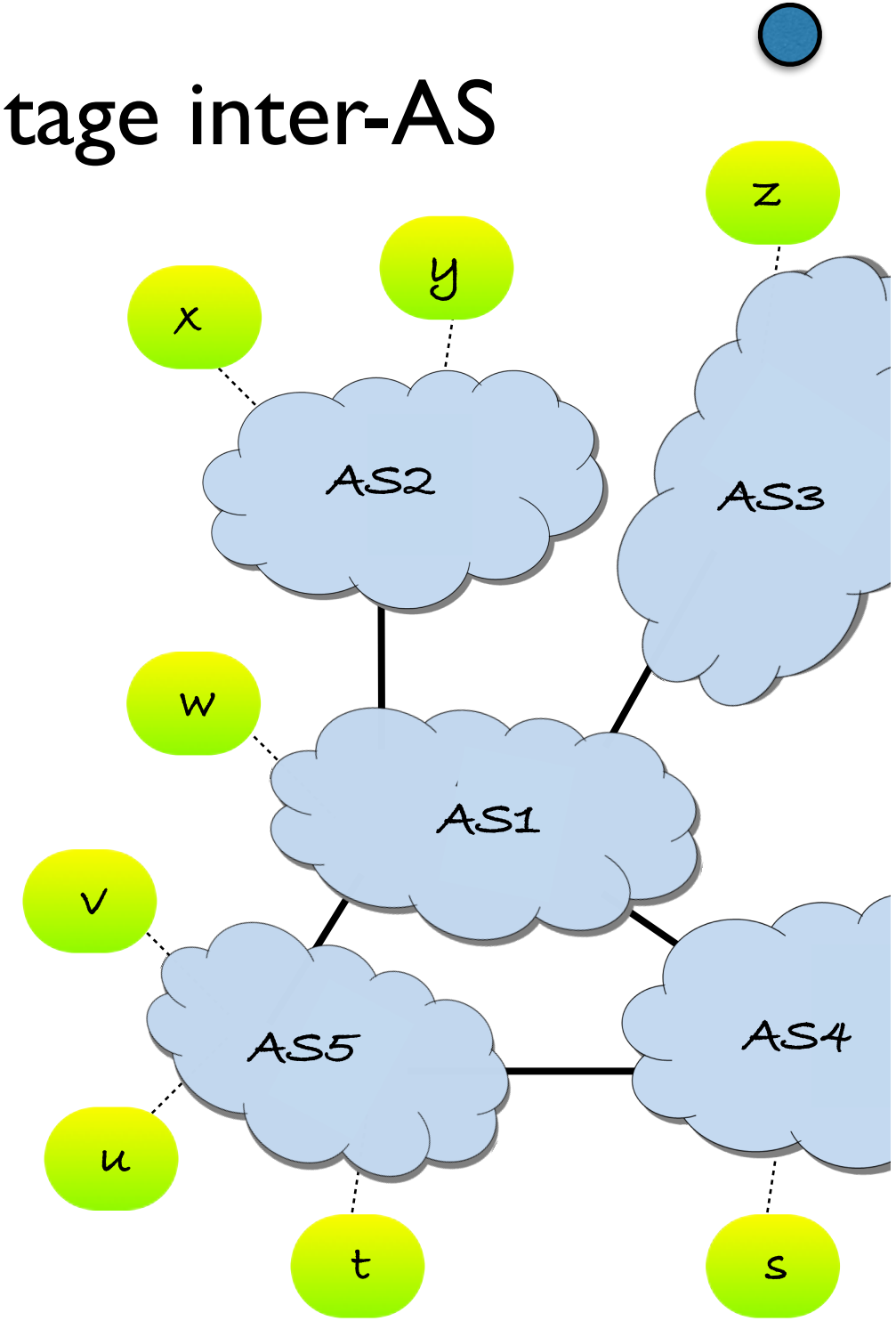


Routeur passerelle :"Gateway router"
interface avec les routeurs d'autres AS

routage intra-AS
détermine les routes vers
les routeurs passerelles

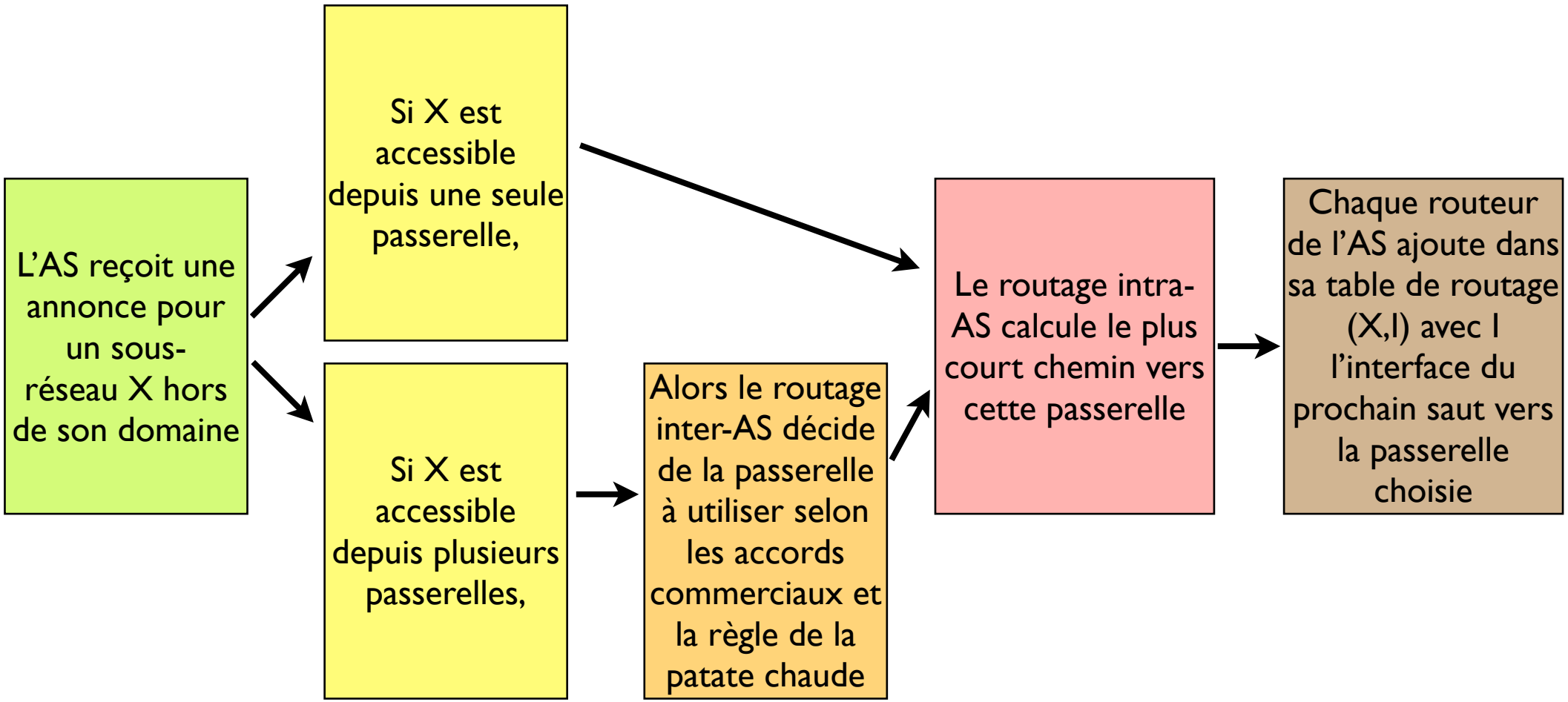
Missions du routage inter-AS

- Annoncer aux autres AS les sous-réseaux desservis par soi
- Apprendre les sous-réseaux accessibles depuis les AS voisins
  - Puis en informer ses routeurs internes
  - Et propager l'information aux autres AS



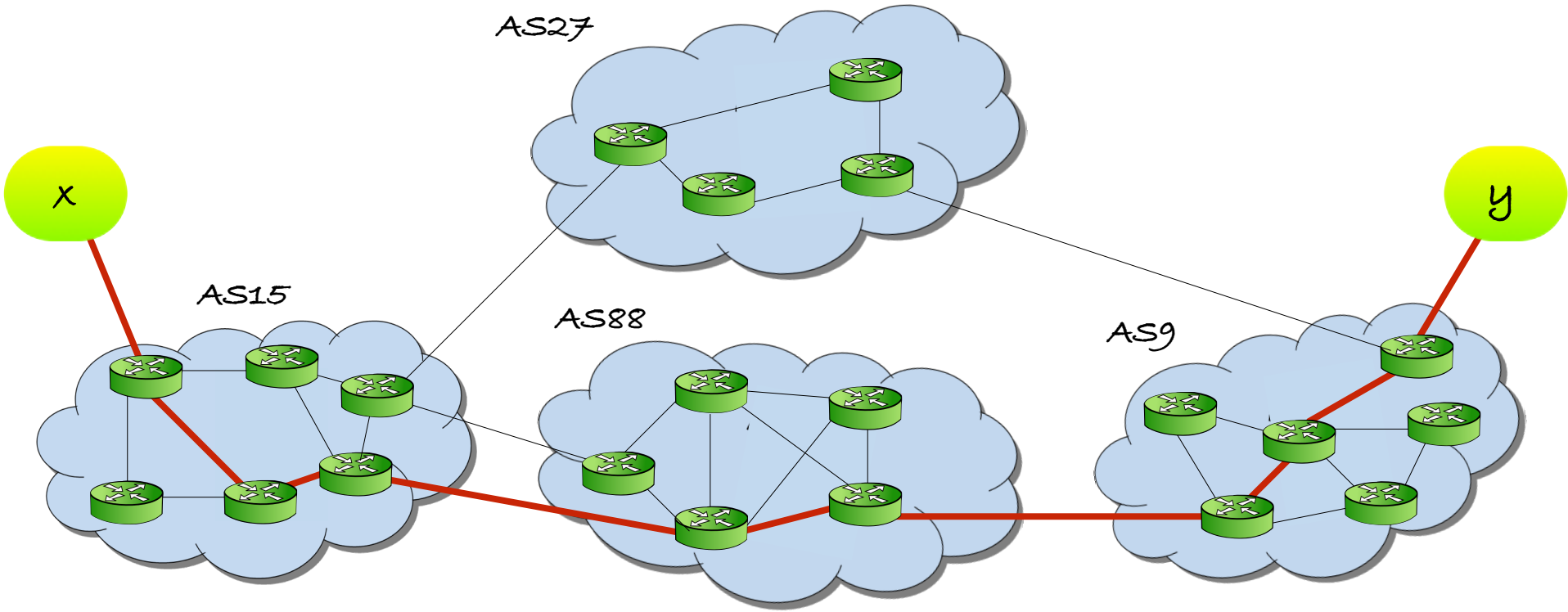
Rôle des routages inter et intra AS

- Couplage entre le routage inter-AS et intra-AS



Où fait-on du plus court chemin ?

- Qui choisit le prochain AS ? et donc la passerelle ?
  - Le routage inter-AS
  - Donc pas toujours le plus court chemin
- Qui choisit la route vers la passerelle ?
  - Le routage intra-AS
  - Donc selon le plus court chemin



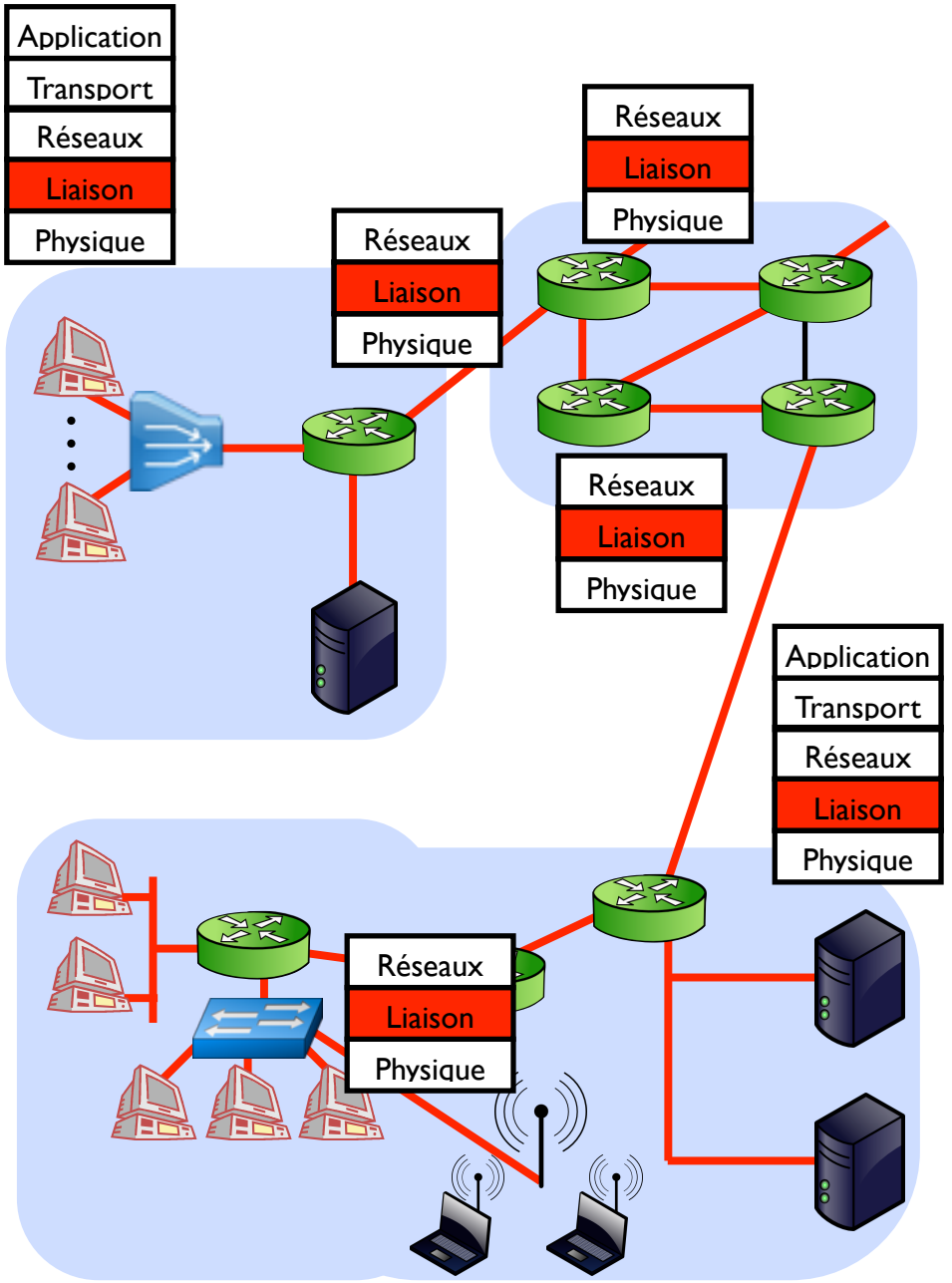
Couche
Liaison de données

Plan

1. Introduction
2. Détection et correction d'erreurs
3. Protocoles MAC
4. Adressage sur la couche Liaison & ARP
5. Ethernet
6. Commutateurs
7. VLANs
8. Technologie Wi-Fi

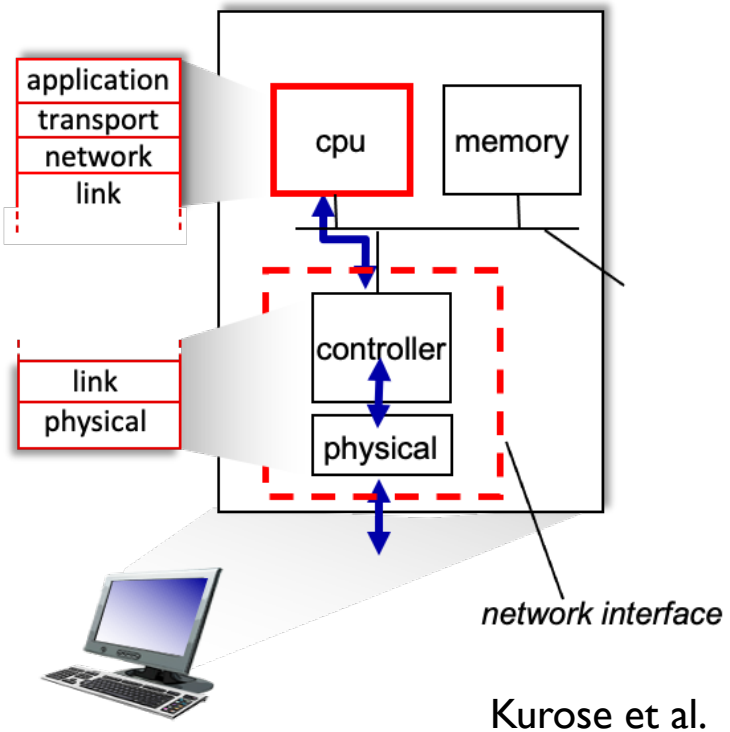
Introduction

- Couche Liaison sur tous les noeuds
  - Terminaux, routeurs et commutateurs
- Assure le transfert d'un datagramme sur un lien
  - entre deux ou plusieurs noeuds voisins
- Encapsule les datagrammes dans des trames
- Nombreux protocoles
  - Filaires, sans fil ; dédiés ou partagés
  - Les plus utilisés : Ethernet et Wi-Fi



Où est implémentée la couche Liaison de données ?

- Dans la carte réseau
  - Interface / adaptateur réseau
  - NIC : Network Interface Card
- Intégrée à la machine
  - cartes parfois extérieures
- Reliée au système par des bus
- Mini-ordinateur
  - CPU, mémoire
  - Fonctionne en parallèle de l'OS
- Hardware, firmware, software



Kurose et al.

Plan

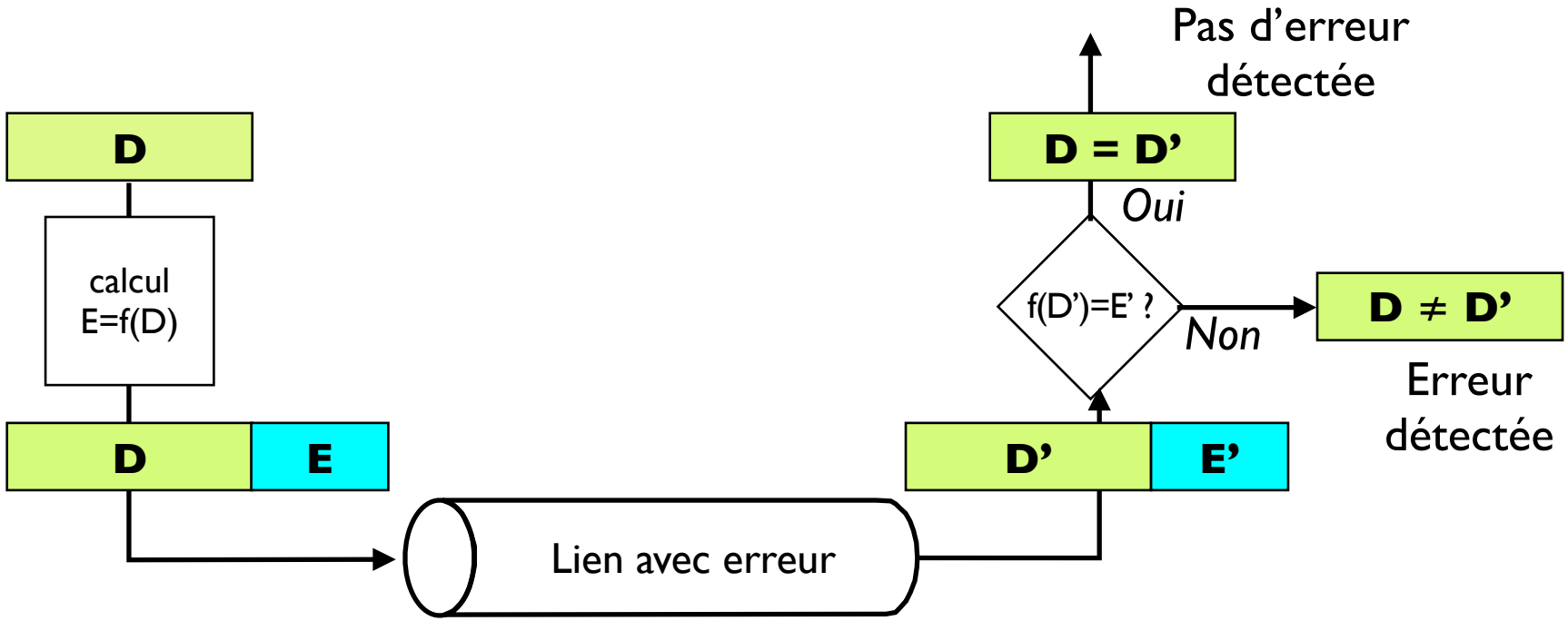
1. Introduction
2. Détection et correction d'erreurs
3. Protocoles MAC
4. Adressage sur la couche Liaison & ARP
5. Ethernet
6. Commutateurs
7. VLANs
8. Technologie Wi-Fi

Détection et correction d'erreurs

- Erreur sur les liens de transmissions
  - atténuation du signal, bruit, interférences
  - surtout sur les liens sans fil (radio)
- Détection d'erreurs détermine si la trame reçue est considérée comme en erreurs
  - Parfois, correction d'erreurs par le récepteur sans retransmission
- Service de détection et correction d'erreurs est optionnel au niveau 2,
  - mais détection d'erreur souvent implémentée dans les protocoles de niveau 2
  - réalisé au niveau hardware (efficace)
- Pourquoi faire de la détection d'erreurs à plusieurs niveaux ?

Détection et correction d'erreur

- **D** : Données à protéger
- **E** : Bits de Détection et/ou Correction d'erreur
  - construits à partir de **D** et concaténés à **D**



- Note : la détection d'erreur n'est jamais fiable à 100% !
  - les protocoles peuvent manquer des erreurs (rare)
  - si taille de E ↗, alors précision des détection et correction ↗