Université de Batna 2

TCP/IP : Introduction

Cours Réseaux et applications -2020/2021

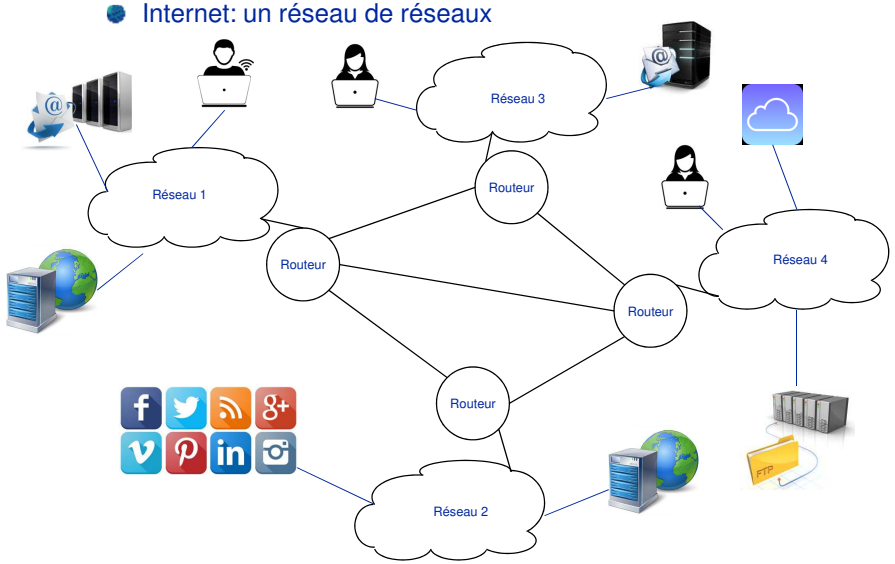
Khaled Hamouid
Département d'informatique
Université de Batna 2
Email : k.hamouid@univ-batna2.dz
Web : http://staff.univ-batna2.dz/hamouid_khaled/

Plan du cours

- Introduction
- Les protocoles de la couche transport : TCP, UDP
- Les protocoles de la couche réseau : IP, ARP/RARP, ICMP
- Adressage IP
- Routage IP

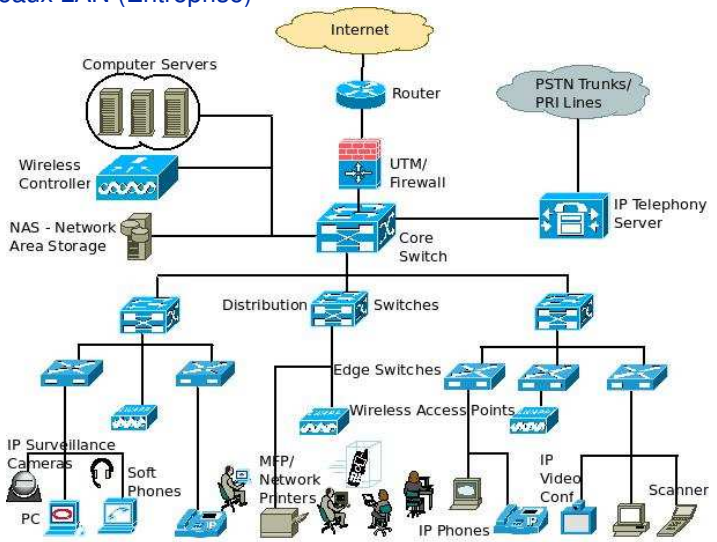
Introduction

- Internet: un réseau de réseaux

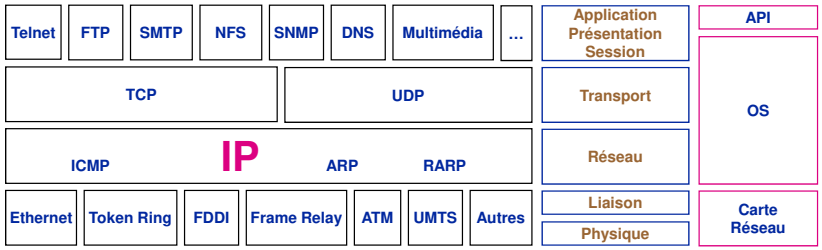


Introduction

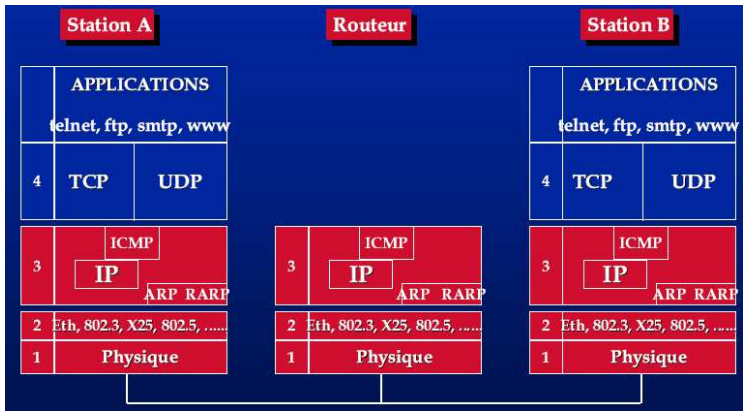
- Réseaux LAN (Entreprise)



Modèle en couches



Modèle en couches



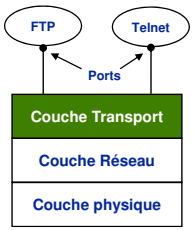
La couche transport

Rôle:

- Transfert de messages de bout en bout (remise d'application à application)
  - Via des ports
- Pallier les imperfections de la couche réseau
  - Perte, erreurs, paquets en désordre
- Adapter les vitesses d'envoi et de réception
  - Contrôle de flux
- Traiter les congestions dans le réseau
  - Contrôle de congestion
- Service
  - Fiable : orienté connexion (TCP)
  - Non fiable: sans connexion (UDP)

La couche transport : Notion de port

- **Comment identifier l'application à laquelle est adressé un datagramme ?**
  - **Les ports**
- **Port**
  - **Point d 'accès au service (application)**
  - **Définit par un numéro entier positif, permettant d'identifier l'application au niveau de la couche transport**
  - **La couche transport reconnaît l'application cible par son numéro de port**
  - **Liste des ports connus RFC 1700 (les numéros de 0 à 1023 sont réservés)**
  - Les numéros supérieurs à 1024 sont permis…



Application	Port	Application	Port
ECHO	7	Telnet	23
FTP	20	SMTP	25
Finger	79	login	513
SNMP	161		

La couche transport : Notion de connexion

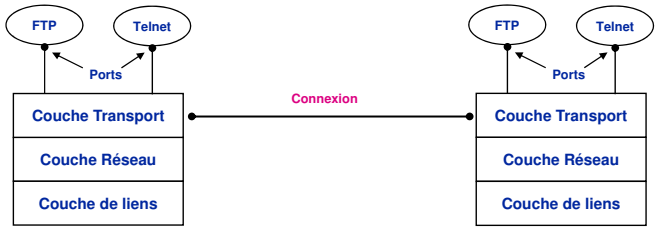
- **Connexion de niveau transport**
  - **Le couple (@ IP, # port) définit une extrémité d'une connexion**
  - **Une connexion est définie par ses deux extrémités**

    (@ IP source, # port source) (@ IP destination, # port destination)

- **Exemple**

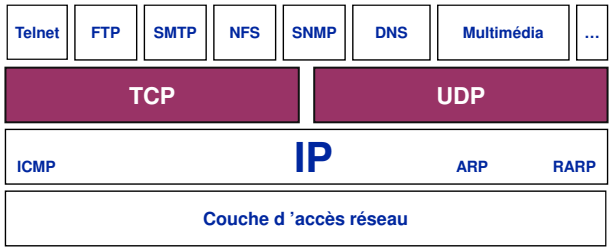
  (18.26.0.36, 1069) et (128.10.2.3, 25)

  (128.9.0.32, 1184) et (128.10.2.3, 53)
  (128.2.24.9, 1184) et (128.10.2.3, 53)



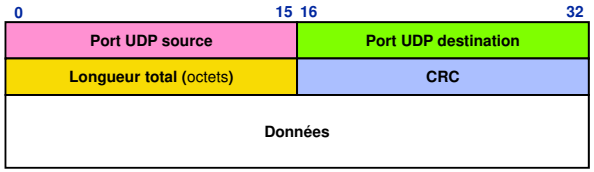
Les protocoles de la couche transport

- UDP: User Datagram Protocol
  - Protocole de transport non fiable
  - Fonctionne en mode non connecté
- TCP: Transmission Control Protocol
  - Protocole de transport fiable
  - Fonctionne en mode connecté



Le protocole UDP

- **UDP (User Datagram Protocol) (RFC 768)**
  - **Fournit le mécanisme de base de transport d'informations entre les applications**
  - **Offre un service de transport non fiable (TFTP, SNMP, DNS, …)**
  - **Fonctionne en mode non connecté**
  - **Utilise le protocole IP pour acheminer ses paquets entre les machines**
  - **Champ protocole dans l'entête IP: 17**



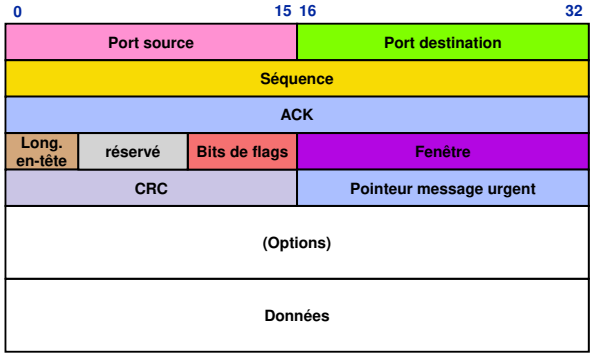
Le protocole TCP

- **TCP (Transmission Control Protocol) (RFC 793)**
  - **Offre un service de transport fiable (FTP, SMTP, …)**
  - **Fonctionne en mode connecté**
  - **Utilise le protocole IP pour acheminer ses paquets entre les machines**
  - **Champ protocole dans l'entête IP: 6**
- **Service redu par TCP:**

1. **Fiabilité**
   - **TCP permet d'effectuer un contrôle sur les données transférées (données endommagées, perdues, dupliquées) et un reséquencement si la couche IP ne les délivre pas dans l'ordre.**
2. **Contrôle de flux**
   - **TCP permet d'effectuer un contrôle de flux (notion de fenêtre), pour mieux exploiter le réseau et pour éviter d'engorger le récepteur (dont l'espace mémoire est faible)**
3. **Contrôle de congestion**
   - **Des algorithmes pour calculer la taille de la fenêtre d'émission adéquate afin d'éviter de congestionner le réseau**
   - **Réduire le débit d'émission si le réseau est encombré**

Les champs d'un segment TCP

- **Séquence**: donne la position du premier octet du segment dans le flux de l'émetteur
- **ACK:** contient le numéro de l'octet attendu par l 'émetteur du message (acquitte les précédents octets reçus correctement)
- **Long. En-tête. (4 bits):** indique la taille de l'entête en mots de 32 bits
- **Fenêtre:** nombre d'octets que le récepteur du message peut accepter (contrôle de flux)
- **CRC:** checksum sur le segment en entier (contrôle d'erreur)

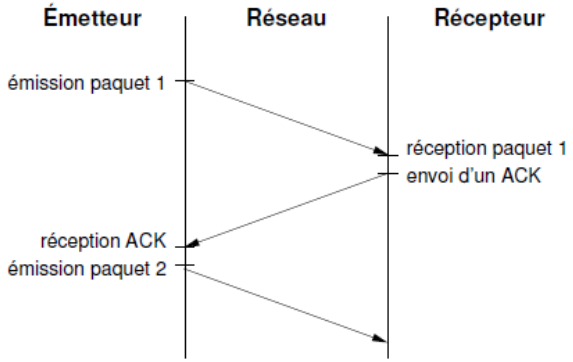


Le protocole TCP : Fiabilité

- Utilisation du mécanisme d'accusé de réception qui implique les éléments suivants
  - Des accusés de réception (ACK)
  - Un temporisateur (en cas de perte ou autres)
  - Des numéro de paquets (numéro de séquence)

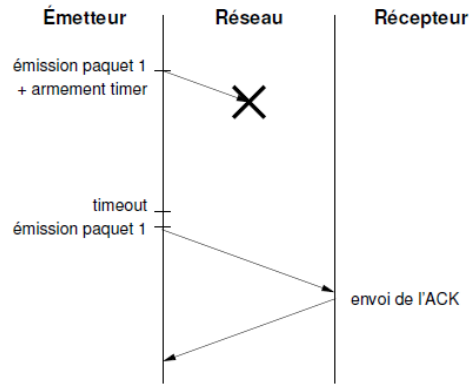
Le protocole TCP : Fiabilité

- Accusé de réception (ACK) : 1ère méthode
  - Principe: l'émetteur attend l'ACK pour chaque paquet envoyé
  - Inconvénient: si paquet ou ACK perdu alors blocage



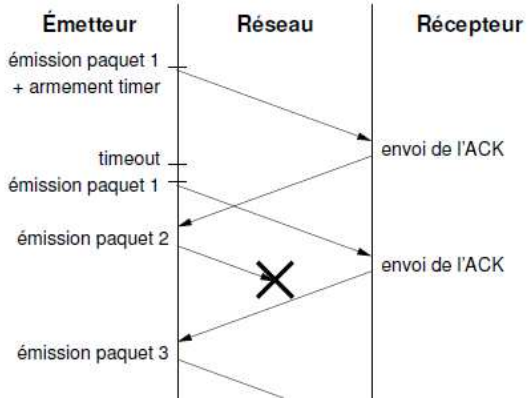
Le protocole TCP : Fiabilité

- Accusé de réception (ACK): 2ème méthode
  - Principe : l'émetteur arme un temporisateur (*timer*) pour chaque paquet envoyé
    - Si *timeout* alors réémission
    - Si ACK reçu alors désactiver *timer*



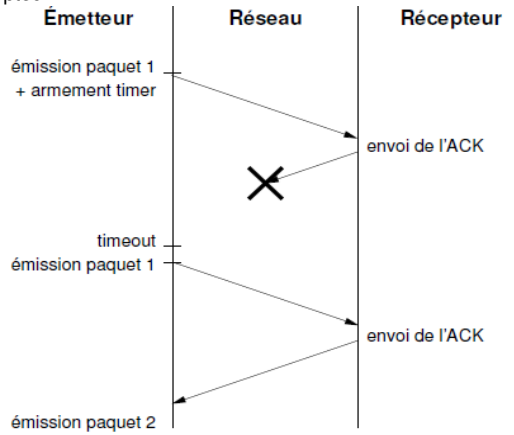
Le protocole TCP : Fiabilité

- Accusé de réception (ACK): 2ème méthode
  - Inconvénient 1 : temporisateur trop court
    - Le même paquet sera reçu et accepté plusieurs fois
    - L'ACK d'un paquet peut être pris pour celui d'un autre paquet



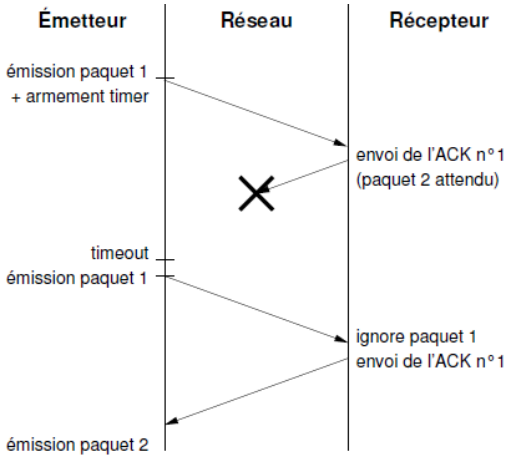
Le protocole TCP : Fiabilité

- Accusé de réception (ACK): 2ème méthode
  - Inconvénient 2 : ACK perdu
    - Le paquet dont l'ACK est perdu sera accepté puiseurs fois par le récepteur



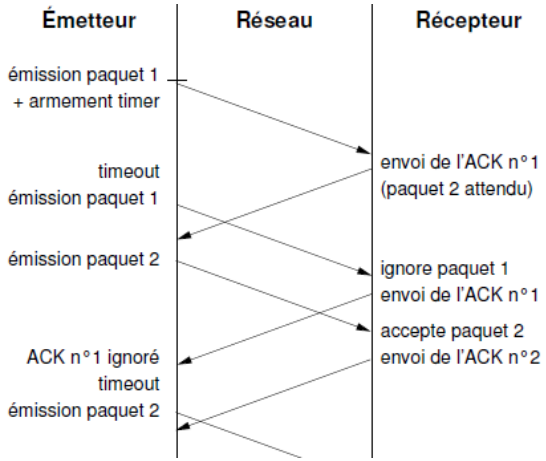
Le protocole TCP : Fiabilité

- Accusé de réception (ACK)
  - Solution ACK perdu : Numéro de séquence



Le protocole TCP : Fiabilité

- Accusé de réception (ACK)
  - Solution temporisateur trop court : Numéro de séquence



Le protocole TCP : Fiabilité

- **Un échange fiable TCP est assuré par la notion de session TCP (le mode connecté)**
  - **Préserver l'ordre des messages**
  - **Détecter / signaler les erreurs et les pertes**
  - **Retransmission des paquets perdus**
- **Phases d'une session TCP:**

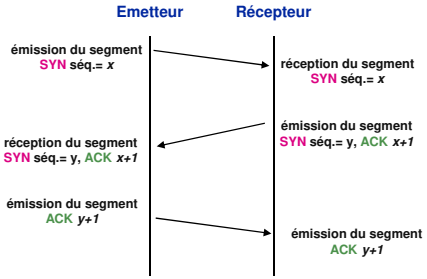
1. ***Initialisation (ouverture) d'une connexion TCP***
   - **Échanges de messages de synchronisation pour se mettre d'accord sur les numéros de séquences**
2. ***Transfert de données***
   - **Transmission des messages marqués par des numéros de séquences**
   - **Acquittement des octets reçus**
   - **Retransmission des segments perdus**
3. ***Fermeture de connexion***
   - **Fermer la connexion après avoir envoyé et reçu tous les segments du flux de données**

Le protocole TCP : Fiabilité

- Les modules TCP de l'émetteur et récepteur communiquent via des segments
- Deux types de segments TCP
  - *Segment de données:* Ce sont les segments qui portent les données envoyées par l'application source
  - *Segment de connexion TCP*
    - Ne porte pas de données
    - Porte des indicateurs de connexion (flags)
    - But: gérer la connexion (initialisation, acquittement, fermeture, resynchronisation, etc.)

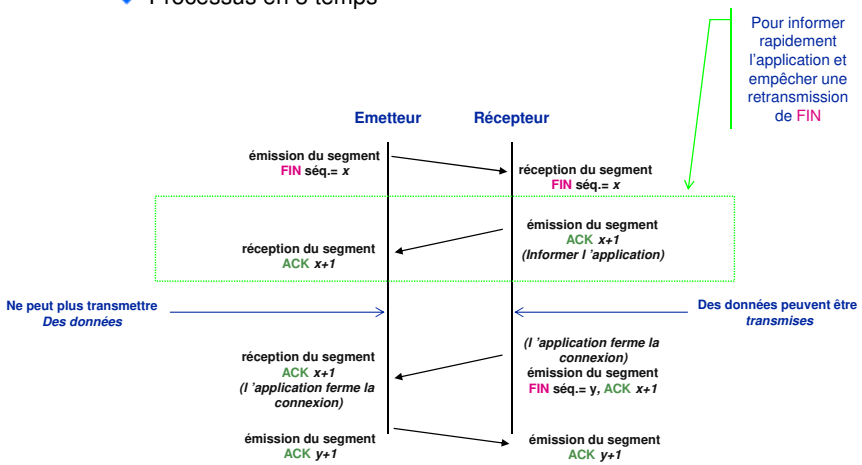
Le protocole TCP : Fiabilité

- **Ouverture d'une connexion TCP**
  - **Processus en 3 temps**
  - **But: Permet également de mettre d'accord l'émetteur et le récepteur sur les numéros de séquences initiales**



Le protocole TCP : Fiabilité

- **Libération d'une connexion TCP**
  - Processus en 3 temps



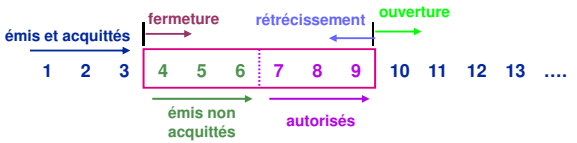
Le protocole TCP : Fiabilité

Transfert de données TCP: Principe

- L'émetteur envoie les données dans des segments de taille maximale MSS (préalablement définie)
- Le numéro de séquence du segment indiqué concerne le 1er octet des données qu'il porte
- Le récepteur envoi un acquittement pour chaque segment de données reçu
- Le numéro d'acquittement indique le prochain octet de données à recevoir
- Les acquittements peuvent être envoyés dans des segments de données ou dans des segments dédiés (de connexion)

Le protocole TCP : Contrôle de flux

- **Les machines et équipements du réseau n'ont pas les même ressources et capacités**
  - **Capacité d'émission > Capacité de réception ⇨ congestion**
  - **Capacité d'émission < Capacité de réception ⇨ faible débit**
- **TCP utilise le mécanisme de la *fenêtre glissante* pour assurer le contrôle de flux**
- **But: meilleure utilisation des ressources réseaux (on attend pas les ACK un par un avant d'envoyer le paquet suivant) ⇨ augmentation des débits**



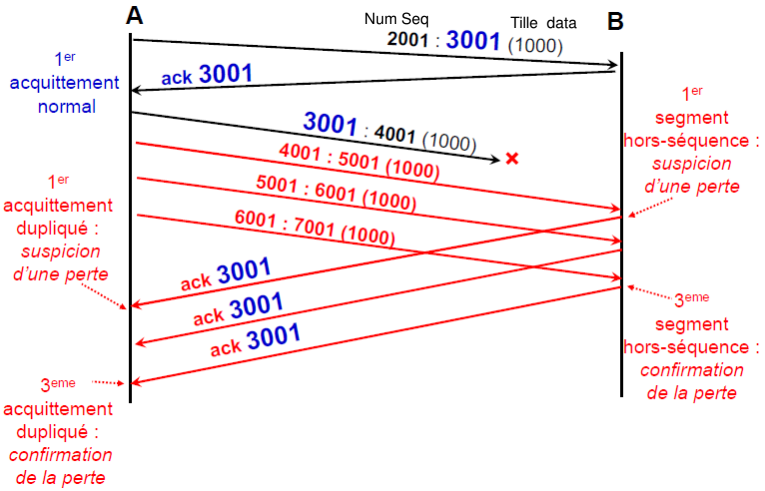
- **La fenêtre se referme à chaque envoi**
- **Elle avance (glisse) à mesure que les données sont acquittées**
- **elle rétrécit lorsque le récepteur est congestionné (réduction de la taille de la fenêtre)**
- **elle s'ouvre lorsque le récepteur augmente sa capacité de réception (demande de recevoir plus de données)**

Le protocole TCP : Contrôle de flux

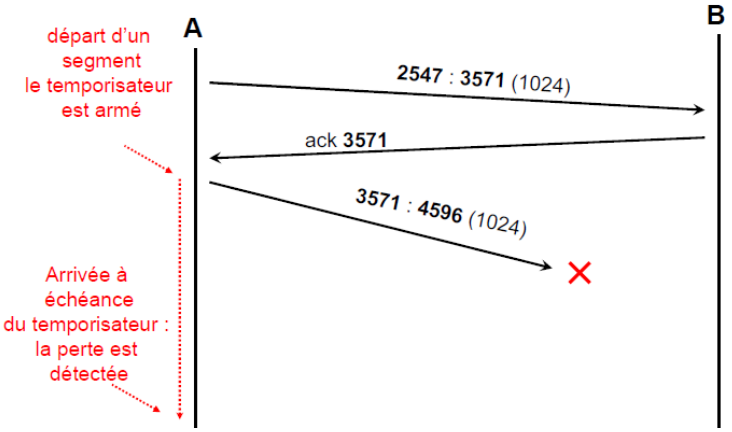
- Paquet perdu ⇒ Retransmission
- Politique de retransmission
  - Quand ? (détection de perte)
    - *Acquittement dupliqué:* réception d'un acquittement dupliqué (retransmission rapide)
    - *Temporisation :* ACK non reçu après un timeout (RTO: Round trip timeout)
  - Paquets retransmis ?
    - *Rejet sélectif:* seulement paquet perdu (non-acquitté)
    - *Rejet total:* tous les paquets ultérieurs à celui perdu

Le protocole TCP : Contrôle de flux

- Politique de retransmission : détection de perte
  - *Acquittement dupliqué*



- Politique de retransmission : détection de perte
  - *Temporisation*



Temporisation (délais d'attente) :

- Deux paramètres :
  - Délais Aller-Retour (RTT: Round Trip Time)
    - Utilisé pour estimer le RTO
    - RTT mis à jour dynamiquement (en fonction de l'état du réseau)

$$RTT = a\,RTT + (1-a)M$$

    - Généralement a=0.9
    - M= temps mis pour retour de l'ACK d'un segment
  - Délais d'attente (RTO: Round Trip Timeout)

$$RTO = \beta . RTT \text{ (Recommandé } \beta=2)$$

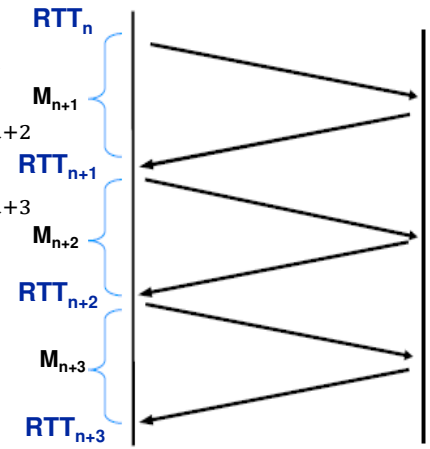
Temporisation (délais d'attente) :

- Délais Aller-Retour (RTT: Round Trip Time)

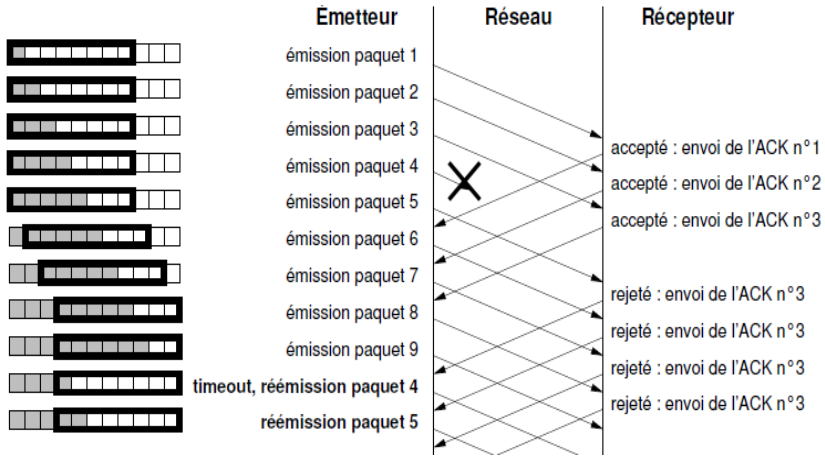
$$RTT_{n+1} = 0{,}9 * RTT_n + 0{,}1 * M_{n+1}$$

$$RTT_{n+2} = 0{,}9 * RTT_{n+1} + 0{,}1 * M_{n+2}$$

$$RTT_{n+3} = 0{,}9 * RTT_{n+2} + 0{,}1 * M_{n+3}$$

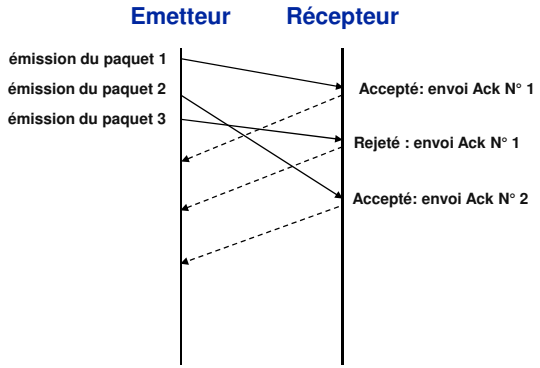


- Paquet perdu ou déséquencement (coté récepteur)
  - **Rejet total :** paquets suivants rejetés



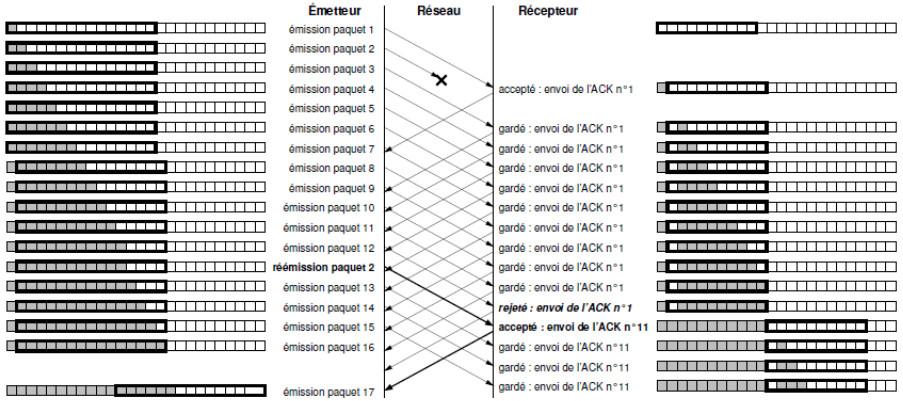
Le protocole TCP : Contrôle de flux

- Paquet perdu ou déséquencement (coté récepteur)
  - **Rejet total :** paquets suivants rejetés



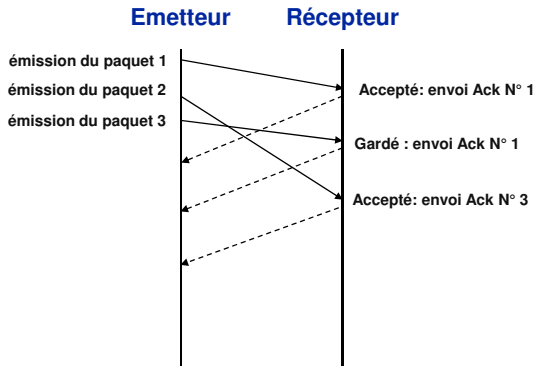
Le protocole TCP : Contrôle de flux

- Paquet perdu ou déséquencement (coté récepteur)
  - **Rejet selectif :** paquets suivants gardés selon la capacité du récepteur



Le protocole TCP : Contrôle de flux

- Paquet perdu ou déséquencement (coté récepteur)
  - **Rejet selectif :** paquets suivants gardés selon la capacité du récepteur



Le protocole TCP : Contrôle de congestion

- **Congestion**
  - Paquets inutiles
  - Transmission à débit dépassant la capacité du réseau
- **Détection de congestion**
  - Pertes de paquets
  - Délais excessifs
- **Contrôle de congestion**
  - Adapter le débit d'émission en fonction de la perception de l'état du réseau
  - Des algorithmes pour calculer la fenêtre d'émission adéquate afin d'éviter les congestions

Contrôle de congestion : Algorithme "Slow Start" et diminution multiplicative

- **rwnd : Reception Window; cwnd : Congestion Window; F: Transmission Window**
- **ssthresh : Seuil du slow start (redémarrage lent)**

- **Algorithme : Démarrage lent (Slow start) "*Prudent*"**

{ **Etablissement de la connexion**
Seuil = rwnd

} **cwnd=1** **F=min (rwnd, cwnd)**

Après réception d'ACK des segments ⇨ cwnd= cwnd* 2

En cas d'erreur, ssthresh = cwnd/2, GOTO 2 (Slow Start)

Si cwnd ≥ ssthresh ⇨ entrer dans le mode évitement de congestion (congestion avoidance)

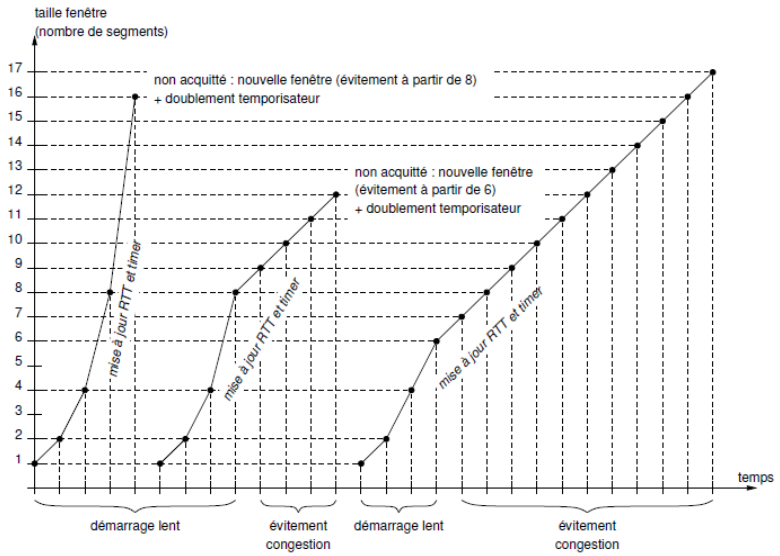
Algorithme "Congestion Avoidance"

- **Algorithme: « *évitement de congestion* »**

{ **Ajouter seulement 1 à cwnd pour chaque réception d'ACK**
cwnd= cwnd + 1

} **En cas d'erreur,** ssthresh = cwnd/2, **entrer dans le mode Slow Start**

Algorithme "Congestion Avoidance"

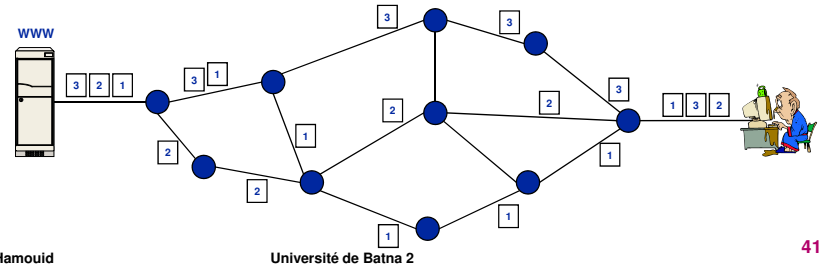


Interface de programmation TCP-IP

- **L'interface SOCKET**
  - Interface de programmation (API)
  - Frontière entre l'OS et l'espace utilisateur
  - Elle n'implique pas forcément une communication via le réseau (communication locale)
  - 5 paramètres :
    - Le type de protocole utilisé (commun pour les deux processus)
    - L'@ IP de la machine A
    - Le # port associé au processus en A
    - L'@ IP de la machine B
    - Le # port associé au processus en B
  - Serveur
  - Client

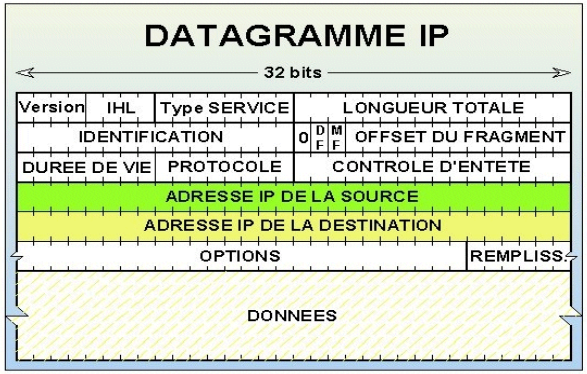
Le protocole IP (Internet Protocol)

- un service de remise de paquets non fiable (Best effort)
  - Mode non connecté
  - Best Effort (Le mieux qu'il peut)
- Mode datagramme
  - les paquets sont traités et acheminés indépendamment les uns des autres
- Remise des datagrammes à des hôtes
  - Directe : ARP (Résolution d'adresses IP/MAC)
  - Indirecte: Routage
- Adaptation aux MTU (fragmentation)
- Signalisation des erreurs via ICMP



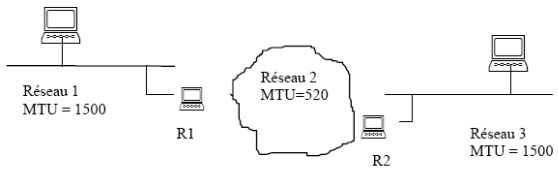
Le protocole IP (Internet Protocol)

- Entête Datagramme



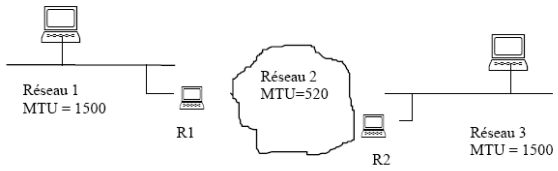
Le protocole IP (Internet Protocol)

- La fragmentation: principe
  - Chaque réseau physique admet des unités de données de tailles plus ou moins grandes MTU (Maximum Transfer Unit), Exemple : Ethernet (1500 octets), TokenRing (16 ko), FDDI (4470 octets)
  - Si la taille d'un Datagramme IP est supérieur au MTU, le routeur d'accès découpe (fragmente) le datagramme en fragments.
  - seul le destinataire a la capacité de réassembler les différents fragments.
  - la perte d'un seul fragment implique une retransmission complète du segment TCP d'origine



Le protocole IP (Internet Protocol)

- La fragmentation: champs entête utilisés
  - Identificateur (16 bits)
  - Flag (3 bits)
    - bit 0: réservé
    - bit 1 : dit bit DF (Don't Fragment)
    - bit 2 : dit bit MF (More Fragment)
  - Fragment Offset (13 bits): déplacement du fragment par rapport au datagramme original exprimé en 8 octets



La fragmentation: exemple (MTU=1500)

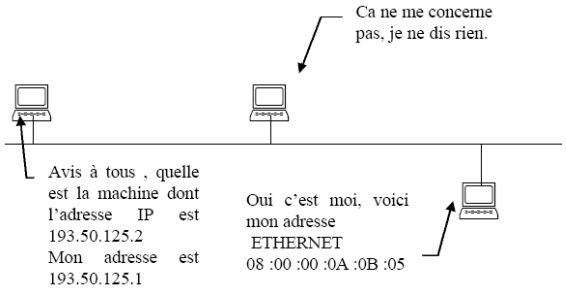
Original IP Packet

Identifier	Total Length	DF	MF
345	5140	0	0

IP Fragments

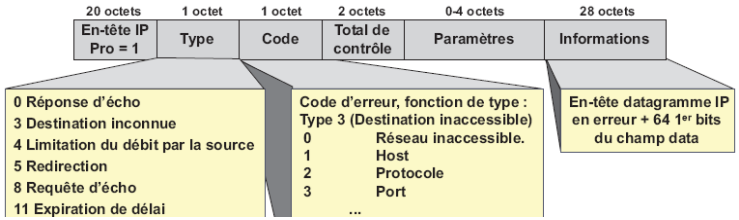
Identifier	Total Length	DF Flag	MF Flag	Offset
345	1500	0	1	0
345	1500	0	1	(0+1480)/8=185
345	1500	0	1	(1480*2)/8=370
345	700	0	0	(1480*3)/8=555

- Problème : trouver une adresse MAC à partir de l'@ IP
- Adress Resolution Protocol
  - Permet de trouver l'adresse physique d'une machine sur le même réseau en donnant uniquement son adresse IP
- L'@ IP est totalement indépendante de l'adresse physique
  - Stockage des @ physiques dans une table ARP (cache)
  - Le cache est remis à jour périodiquement
  - Sous Unix, visualisation de la table : *arp -a*

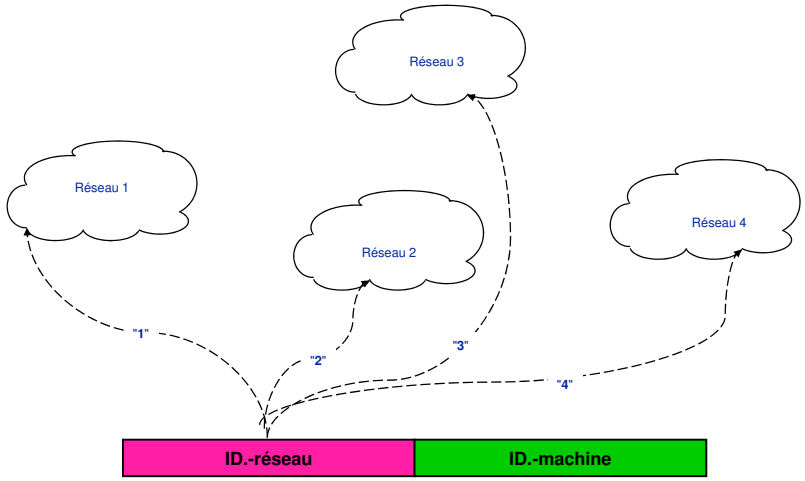


- Problème : trouver une adresse IP à partir de l'@ physique
- Reverse Adress Resolution Protocol
  - Permet de demander une adresse IP en indiquant l'adresse physique
  - Utilisé par des équipements de « boot » par certains équipements
- Utilisé par :
  - Les macintosh avec boite kinetics
  - Les stations sans disque
  - Les terminaux X

- permet d'informer d'une erreur réseau (message d'erreur) ou de formuler une demande d'état à un système (message d'information).
- Implémenté sur tous les équipements IP : stations, routeurs
- Les messages ICMP sont encapsulés dans un datagramme IP (Protocole = 1).



Adressage IP (1)



Adressage IP (2)

Notation *décimale pointée (dotted decimal notation). Ex:* 129.25.245.25
Taille : 32 bits (octet . octet . octet . octet)

Classe A: N1.H1.H2.H3
N1 = 0 à 127
Réseaux: 126
Machines: 16 777 214

Exemple: 125.14.28.32

Classe B: N1. N2.H1.H2
N1 = 128 à 191
Réseaux: 16 383
Machines: 65 534

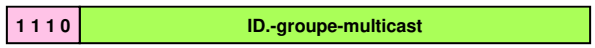
Exemple: 160. 60.18.122

Classe C: N1. N2. N3.H1
N1 = 192 à 223
Réseaux: 2 097 151
Machines: 254

Exemple: 200. 120.212.22

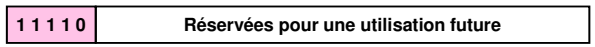
Adressage IP (3)

Classe D: X.Y.Z.T
X = 224 à 239
@s-multicast: $2^4 * 2^{24}$
(224.0.0.0 à 239.255.255.255)



Exemple: 230. 230. 230. 230

Classe E: X.Y.Z.T
X = 240 à 247



Adresses privées et publiques

- **Adresse publique**
  - Utilisée de façon unique sur Internet, et pas sur LAN
  - Routée sur internet et visible de l'extérieur
  - Exp: adresse serveur web internet, adresses allouées par FAI
- **Adresses privées**
  - Utilisée dans les réseaux locaux privés
  - Ne sont pas routables sur internet, ni visible de l'extérieur
  - Unique seulement au sein du sous réseau local
  - Les classes A, B et C comprennent chacune une plage d'adresses IP privées à l'intérieur de la plage globale.

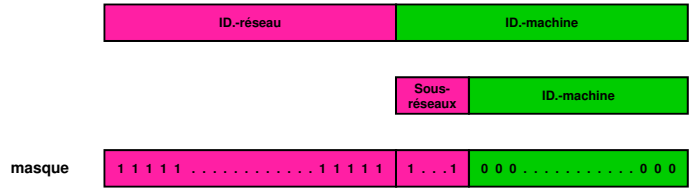
10.0.0.0 à 10.255.255.255	1 réseau de classe A
172.1.0.0 à 172.16.255.255	16 réseau de classe B
192.168.0.0 à 192.168.255.255	256 réseau de classe C

  - L'interconnexion entre les adresses publics et privées nécessite une traduction d'adresses (NAT: Network Address Translation)

Cas particuliers d'@ IP

- Le premier octet de l'adresse IP = 127
  - utilisé comme @ de loopback (rebouclage). Ex.: 127.0.0.1
- Tous les bits de l'ID.-machine sont à 0
  - adresse de réseau attribuée par interNIC . Ex.: 196.125.25.0
- Tous les bits de l'ID.-machine sont à 1
  - interprétée comme toutes les machines du réseau.
  - Ex.: 196.125.25.255 (diffusion sur le réseau entier)
- Tous les bits de l'@ IP sont à 0
  - utilisée par l'algo. de routage pour désigner la route par défaut et par BOOTP.
  - Forme: 0.0.0.0

Découpage en sous-réseaux



Avantages:

- trafic réduit dans le réseau (diffusion restreinte), donc optimisation des performances du réseau
- gestion plus facile du parc de machines en définissant des classes
- Réduire le gaspillage des adresses

Sous-réseaux

Méthode de calcule

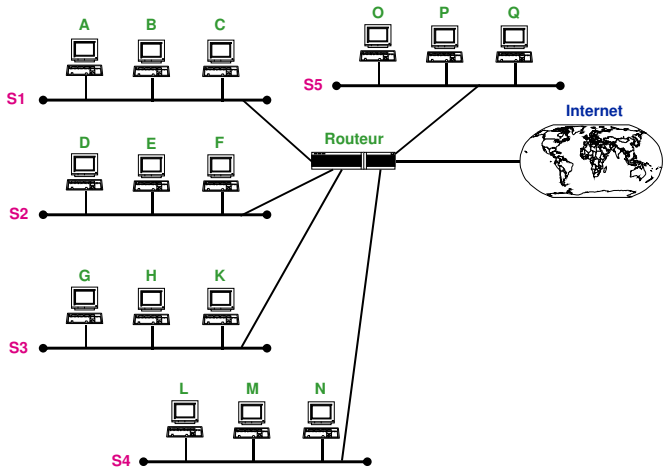
- **Empruntez le nombre de bits suffisants**
  - **Nombre de bits pour N sous réseaux : $log_2(N)$**
- **Calculez le nouveau masque de sous réseau**
  - **Bits réseau + bits sous réseaux à 1, Bits hôtes à 0**
- **Identifiez les différentes plages d'adresses IP**
  - **Identifiez les adresses de sous-réseau et de broadcast**

    Exp: @SR = 192.168.1.128/25, @broadcast=192.168.1.255

    @SR = 192.168.1.0/25, @broadcast=192.168.1.127
  - **Déterminez les plages d'adresses utilisables pour les hôtes**
    - **Exp @SR = 192.168.1.0, Plage adresses=[192.168.1.1—192.168.1.126]**

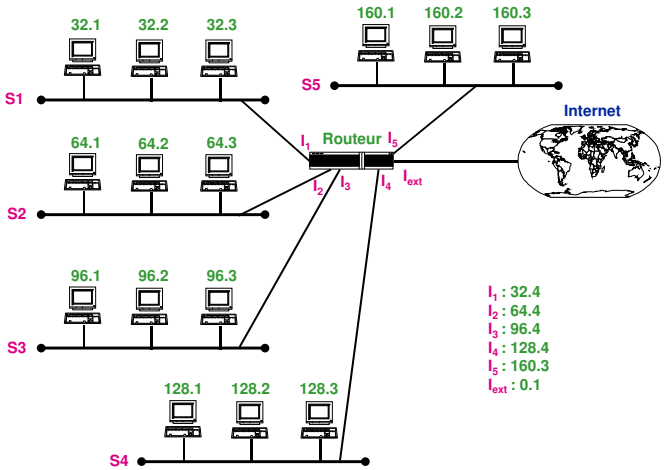
Exercice

- Organisez en 5 sous-réseaux le réseau donné ci-dessous dont l'@ IP est la suivante : 135.100



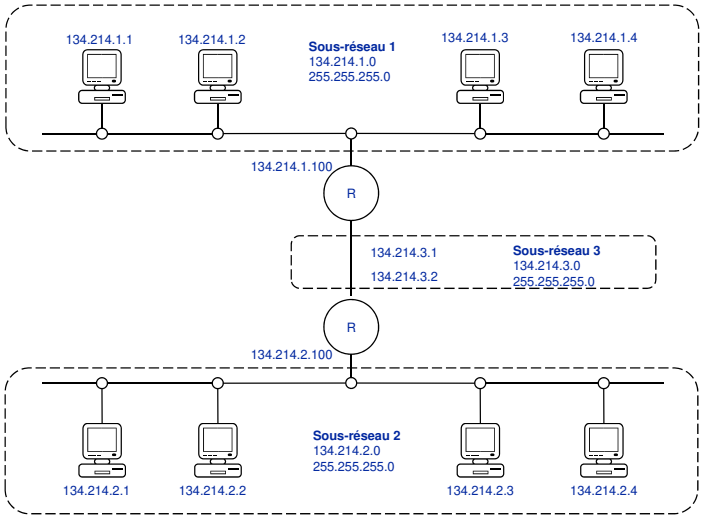
Solution

- Organisation, en 5 sous-réseaux, du réseau donné ci-dessous dont l'@ IP est la suivante : 135.100



VLSM (Masque de Sous-réseaux de Longueur Variable)

- Problème du masque de sous-réseau fixe
  - Espace d'adressage perdu avec un masque de sous-réseau unique



VLSM (Masque de Sous-réseaux de Longueur Variable)

- Problème du masque de sous-réseau fixe
  - Espace d'adressage perdu avec un masque de sous-réseau unique

Soit un réseau de classe C : 200.20.2.0 et on veut le subdiviser en 3 sous-réseaux avec 120 hôtes au maximum sur un des sous-réseaux et 60 hôtes sur chacun des deux autres.

120 + 60 + 60 = 240; Une adresse de classe C permet d'avoir 254 hôtes.

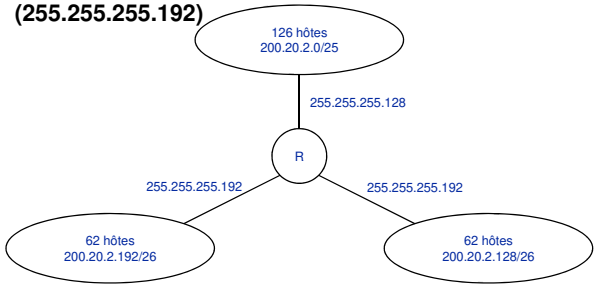
Raisonnement par hôtes:
Pour avoir 120 hôtes sur un sous-réseaux, il faudrait mettre le masque :
11111111 11111111 11111111 10000000 (2^7=128-2=126)
255. 255.255.128

Raisonnement par sous-réseaux:
Pour avoir 3 sous-réseaux, il faudrait mettre le masque :
11111111 11111111 11111111 11000000 (2^6=64-2=62)
255. 255.255.192

- **Masques de sous-réseau de longueur variable**

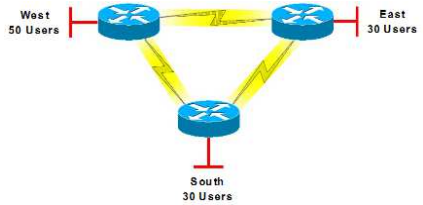
Problème du masque de sous-réseau (3)

- **Solution: Variable Length Subnet Mask (VLSM) ou «Masque de Sous-réseaux de Longueur Variable»**
- **Dans notre exemple:**
  - **On subdivise le réseau en deux sous-réseaux de 126 hôtes chacun (255.255.255.128)**
  - **Ensuite on subdivise encore un de ces deux derniers sous-réseaux en deux autres sous-réseaux de 62 hôtes chacun (255.255.255.192)**



Problème du masque de sous-réseau (4)

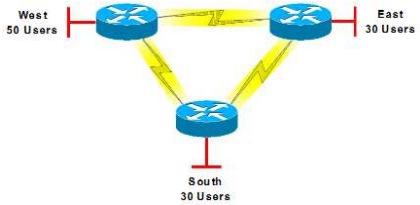
- **Exemple VLSM**
  - Une entreprise a reçu une adresse réseau de classe C : 199.1.1.0.
  - **Que peut-on faire ? Faut- il utiliser VLSM ?**



128	64	32	16	8	4	2	1
0	0	West 199.1.1.0 /26					
0	1	East 199.1.1.64 /26					
1	0	South 199.1.1.128 /26					
1	1	Lien entre West et East 199.1.1.256 /26					

Problème du masque de sous-réseau (4)

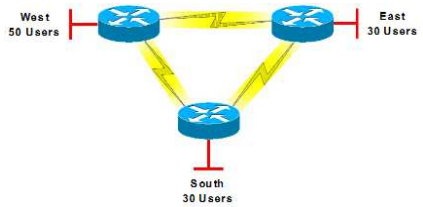
- **Exemple VLSM**
  - **Inconvénients:**
    - Espace d'adressage perdu: on peut pas prévoir une extension du réseau (ajouter d'autres sous réseaux, d'autres liaisons séries, d'autres hotes)



128	64	32	16	8	4	2	1
0	0	West 199.1.1.0 /26					
0	1	East 199.1.1.64 /26					
1	0	South 199.1.1.128 /26					
1	1	Lien entre West et East 199.1.1.256 /26					

Problème du masque de sous-réseau (4)

- **Exemple VLSM**
  - **Appliquer VLSM**



128	64	32	16	8	4	2	1
0	0	West 199.1.1.0 /26 (50 util.)					
0	1	0	East 199.1.1.64 /27 (30 util.)				
0	1	1	South 199.1.1.96 /27 (30 util.)				
1	1	0	0	0	0	West à East 199.1.1.192 /30	
1	1	0	0	0	1	West à South 199.1.1.196 /30	
1	1	0	0	1	0	East à South 199.1.1.200 /30	