Université Internationale de Casablanca

LAUREATE INTERNATIONAL UNIVERSITIES

Protocoles réseaux et interconnexion

Dr. Mohammed BOUTABIA

Université Internationale de Casablanca

77

Couche réseau: le Protocole IP

78

Commutation de circuit et commutation de paquet

- Commutation de circuit: le chemin est établi avant l'échange de données (circuit électrique fermé entre le transmetteur et le récepteur)
- Exemple: RTC
- Commutation de paquet: le chemin n'est pas établi au préalable chaque paquet est acheminé de manière individuel
- Exemple: le réseau IP

79

Propriété de connexion

- Un protocole est orienté connexion si durant la communication, une connexion de bout en bout est établi au préalable et maintenu jusqu'à terminaison de la communication
- Un protocole sans connexion (connectionless) si aucune connexion n'est établit entre les parties communicante avant et duranr l'envoie des données

80

Les standards internet

- Développés par l'IETF : Internet Engineering Task Force
- Documents = RFC : Request For Comments
  - Documents techniques et détaillés définissant les protocoles tels que HTTP, TCP, IP…
  - + de 2000 RFCs
- Un document passe par plusieurs étapes avant d'être un RFC: draft=> experimental=>RFC

81

Propriétés du protocole IP

- IP est un protocole sans connexion
- Les paquets sont routés indépendamment
- IP est indépendant du réseau physique (permet d'interconnecter différents technologies)
- Ne fournit pas de fiabilité
- Pas de contrôle de flux
- Pas de correction d'erreur
- Les paquets IP peuvent arriver en désordre, dupliqués ou même perdues

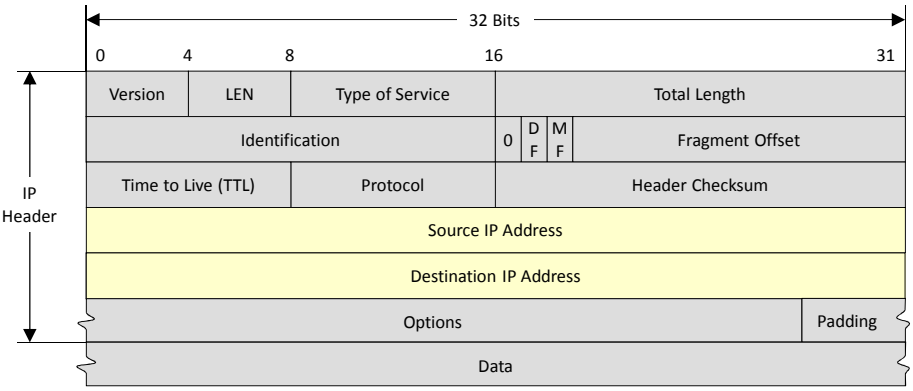
82

Routage

- La fonction principale de la couche IP est d'acheminer ou **router** un paquet d'une station à une autre
- Le routage se base sur l'dresse IP (adresse logique) qui figure dans le paquet IP
- Un **Routeur** est l'équipement réseau qui permet de router les paquet d'un réseau à un autre
- L'adresse destination est l'élément clé pour effectuer une décision de routage

83

Adresse source et destination



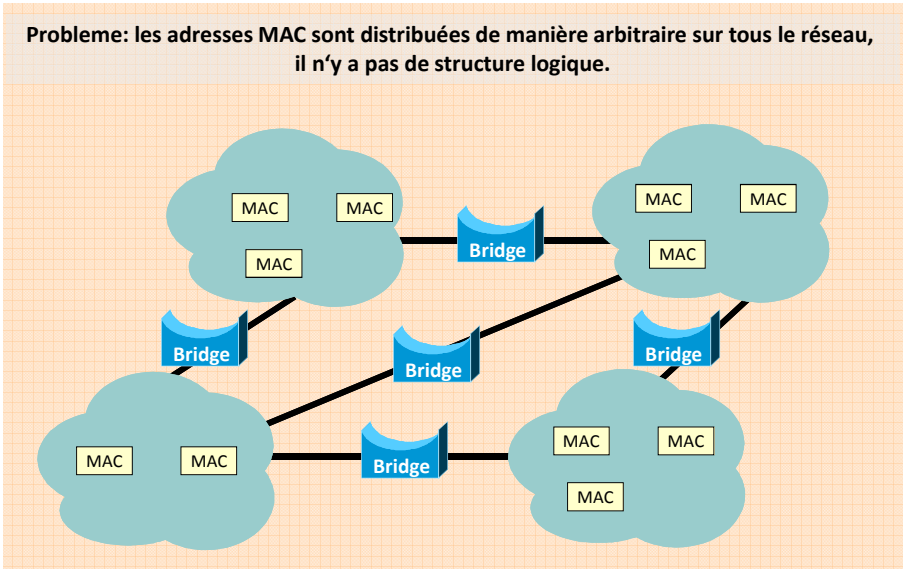
84

Adressage

- L'adresse physique (MAC) peut identifier une station de manière unique
- Les équipements réseaux sont éparpillés dans tout le monde sans ordre (pas d'organisation)
- Le nombre énorme d'équipements => une base colossal de stations
- Délivrer des paquets sur la base d'une adresse physique n'est pas pratique

85

Probleme: les adresses MAC sont distribuées de manière arbitraire sur tous le réseau, il n'y a pas de structure logique.

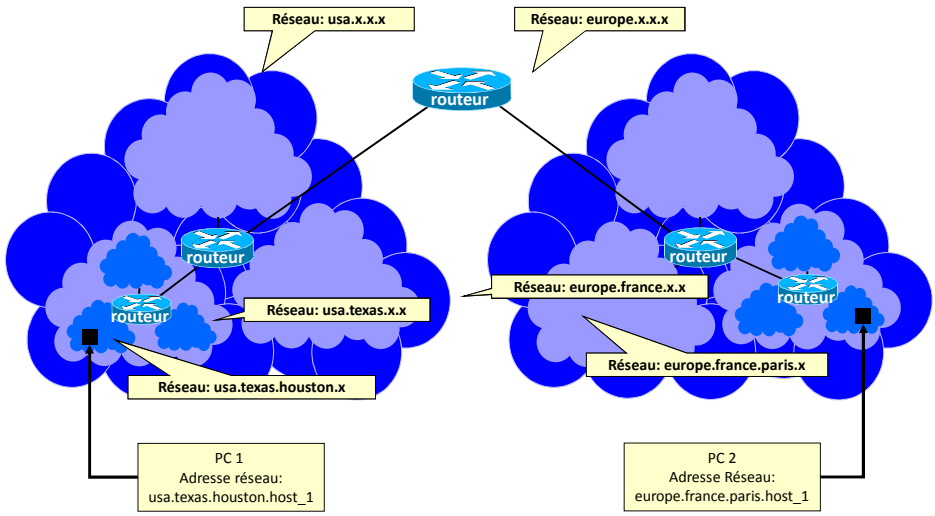


86

Adressage logique

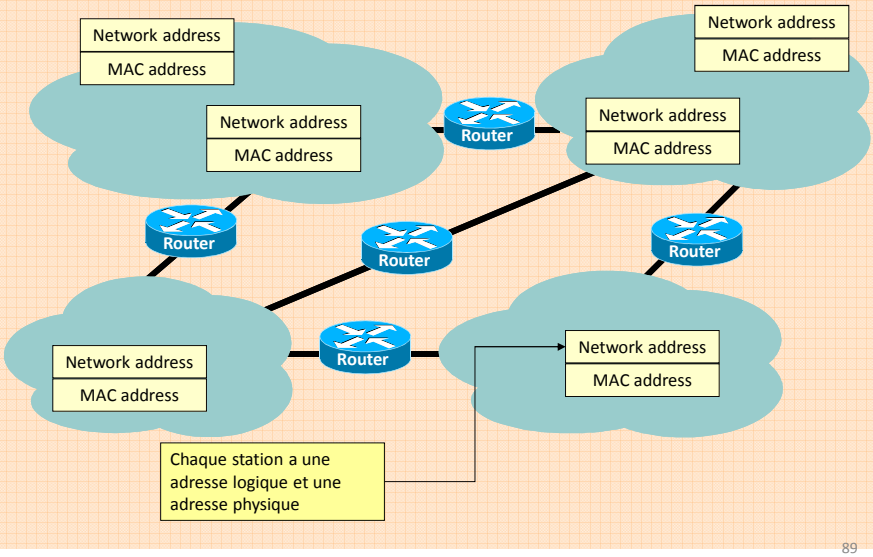
- L'adressage avec adresse logique suit une structure hiérarchique => analogie avec l'adressage postale
- L'adresse logique est allouée pendant la configuration du réseau
- La transmission physique du paquet se base seulement sur l'adresse physique
- Seulement au niveau du routeur que l'adresse de niveau 3 est utilisée

87



88

Solution: Adresses réseau. Elles ont une structure logique qui peut être utilisé dans la livraison des paquet même dans des grands réseaux



89

Composition d'une adresse IP

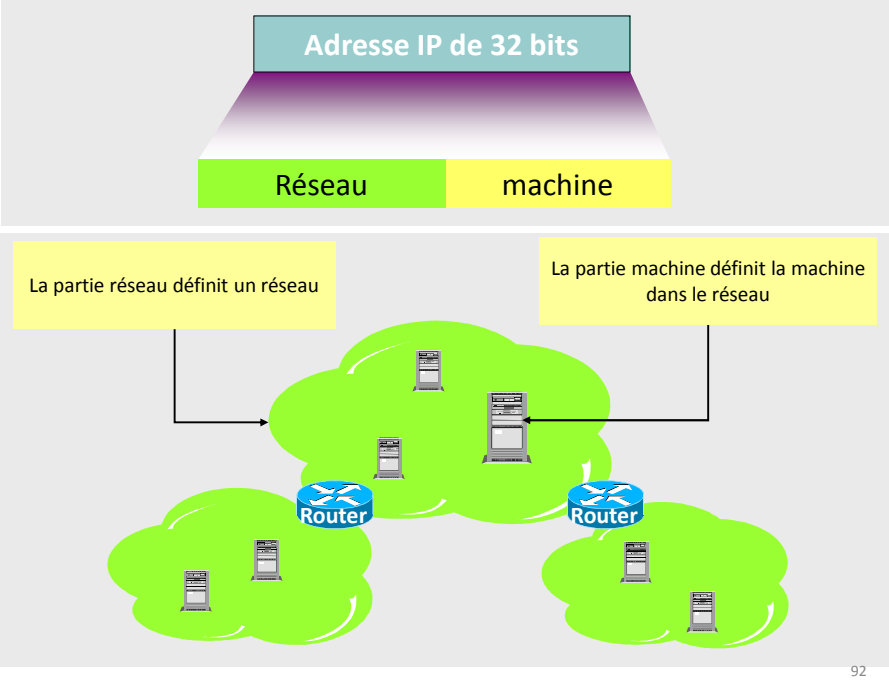
- Partie réseau : appelé aussi préfixe => sur la base du préfixe que le routeur prend une décision pour router le paquet
- Partie host : numéro du poste => utiliser pour identifier le poste dans un réseau
- Toutes les stations d'un même réseau doivent avoir le même préfix et des numéro de poste différents => Une adresse IP doit être unique dans un réseau

90

Structure d'une adresse IP

- 32 bit => 4 Octet (IPv4)
- Possibilité d'avoir (2^{32}) machines (4.294.967.296) => pas toutes utilisées
- IPv4 est en pénurie d'adresse => dernier préfixe /8 attribuée en chine
- Passage à l'IPv6 obligatoire => 6,65 10^{23} adresse IP par m2

91



92

Structure de l ’adresse IP

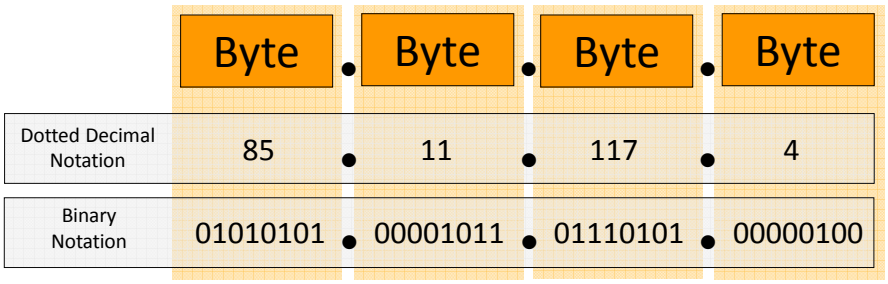
- La structure hiérarchique de l’adressage IP rend la tache facile aux routeurs
- Ils regardent seulement le préfixe réseau => tache très difficile si le routage se faisait sur la base de toute l’adresse IP (2^{32})

93

Notation binaire et décimal

- Une adresse IP est composé de 32 bit => 10110101.11111011. 10110101.11111110
- Difficile à retenir
- Notation décimal => conversion binaire décimal 81.200.1.1
- Exercice: convertir l’adresse binaire en décimal et l’inverse.

94



95

Classes d'adresses

- Les adresse sont classées dans différentes classes selon le nombre de machines dans le réseau
- Les adresses réseaux ont été classées en classes: A,B,C,D et E
- Inconvénient de la classification: perte d'adresses
- Un réseau avec seulement 40 PC doit avoir une classe C de 254 adresses => 214 adresses non utilisées

96

Classe A

- Une adresse de classe A a un préfixe réseaux de 8 bit
- 24 bits consacrés à l'identification de la machine => (2^{24} -2) adresse
- La première adresse et la dernière ne sont jamais octroyée à une machine
- On réfère à une adresse de la classe A avec /8 => 8 étant la longueur du préfix réseau
- Une adresse de classe A commence par 0 en notation binaire
- En notation décimal la classe A occupe l'intervalle : 1.0.0.0 jusqu'à 126.255.255.255

Classe B

- Une adresse de classe B a un préfixe réseaux de 16 bit
- 16 bits consacrés à l'identification de la machine => (2^{16} -2) adresse
- La première adresse et la dernière ne sont jamais octroyée à une machine
- On réfère à une adresse de la classe B avec /16 => 16 étant la longueur du préfix réseau
- Une adresse de classe B commence par 10 en notation binaire
- De 128 à 191

Classe C

- Une adresse de classe C a un préfixe réseaux de 24 bit
- 8 bits consacrés à l'identification de la machine=>(2^8 -2) adresse
- La première adresse et la dernière ne sont jamais octroyée à une machine
- On réfère à une adresse de la classe C avec /24 => 24 étant la longueur du préfix réseau
- Une adresse de classe C commence par 110 en notation binaire
- De 192.0.0.0 à 223.255.255.255

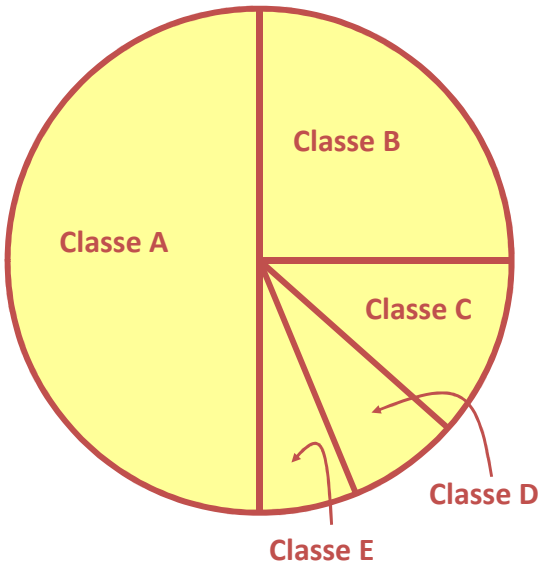
Classe D

- Les adresses de la classe D sont consacrées aux groupes multicast
- Elles commencent avec 1110
- De 224.0.0.0 à 239.255.255.255

Classe E

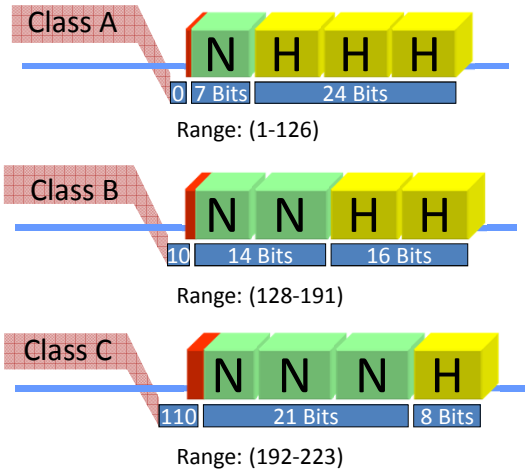
- Les adresses de classe E sont utilisées pour des fins expérimentales et ne sont pas disponibles pour une utilisation normale
- Elle commencent par 1111
- De 240.0.0.0 à 255.255.255.255
- Les adresses des classes D et E ne sont jamais utilisées pour adresser les machines d'un réseau

2^{32} (4,294,967,296) IP Addresses

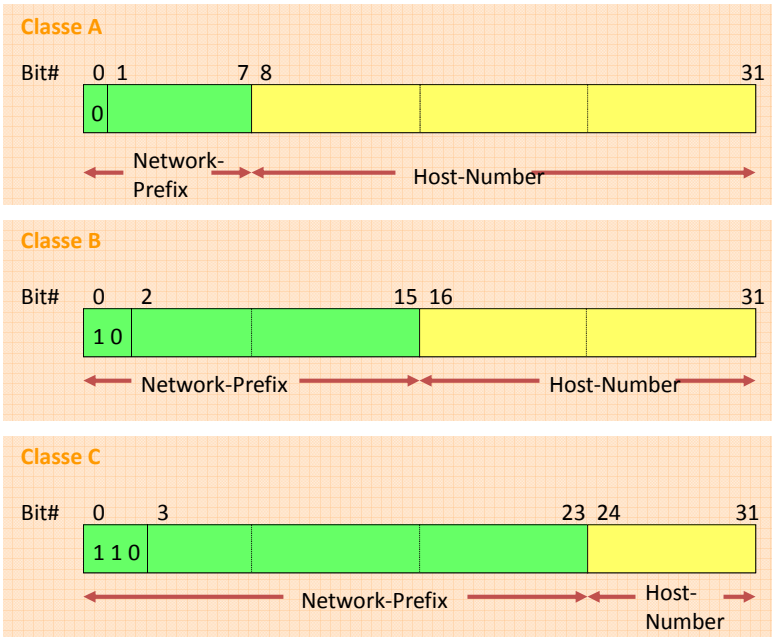


102

Structuring of the 4 Address Octets



103

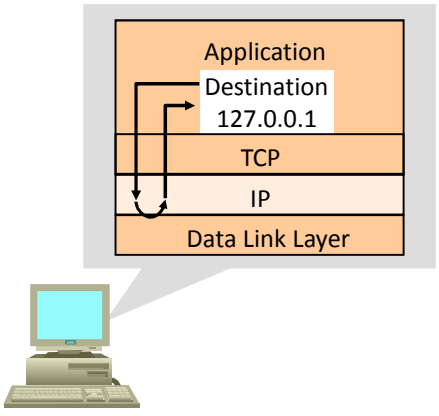


104

Adresses IP réservées

- Tous les numéro machine à 0 => ID du réseau
- Tous les numéro machine à 1 => Broadcast dirigé
- 127.x.x.x => Adresse de loopback
- 0.0.0.0 => Adresse source avant l'attribution de l'adresse ou route par default
- 255.255.255.255 => Adresse de broadcast local

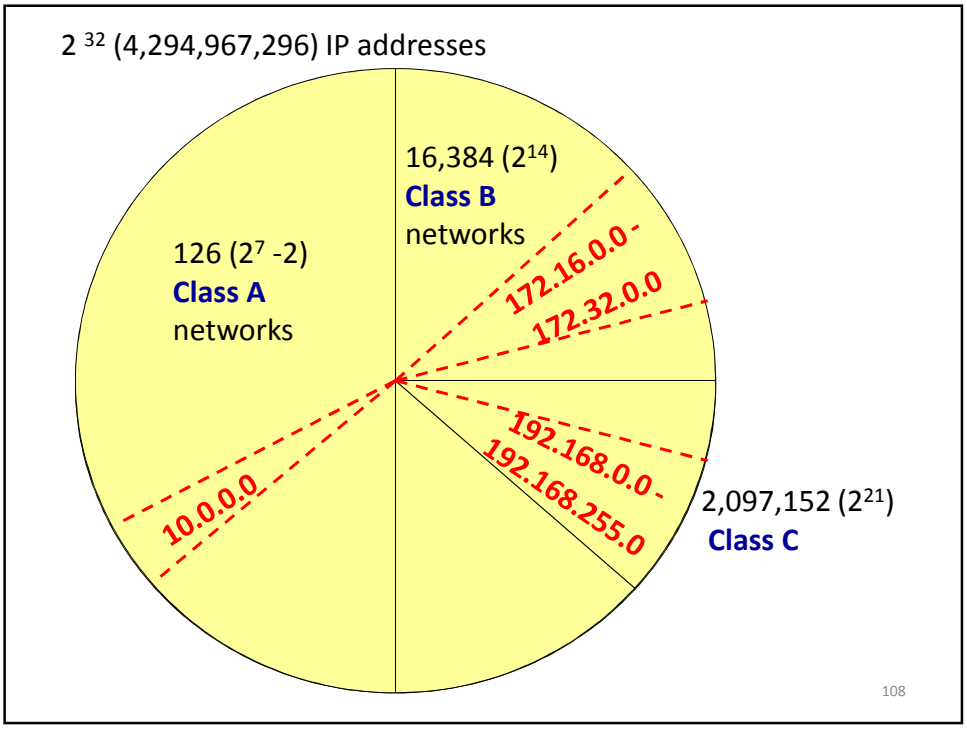
Adresse de loopback



106

Adresse publique et adresse privée

- Une adresse privée ne peut être routée sur internet
- Usage privé => réseau local
- Une adresse publique c'est une adresse affectée par un organisme internationale
- L'adresse publique est routée sur internet



Les adresse privées

Intervalle d'adresse	nombre de réseaux	la classe
10.0.0.0 - 10.255.255.255	1	A
172.16.0.0 - 172.31.255.255	16	B
192.168.0.0 - 192.168.255.255	256	C

Les types d'adresse d'un réseau IP

- Adresse réseau => tous les numéros de machine à 0 (exemple: 10.0.0.0)
- Adresse de diffusion => tous les numéros de machine à 1 (exemple: 10.255.255.255)
- Adresse de machine => préfixe du réseau plus le numéro de la machine (exemple: 10.0.0.1)

Allocation des adresses

- Allocation statique: une adresse IP est configuré manuellement par un administrateur de réseau => utilisé pour les serveurs et les équipements réseau
- Allocation dynamique: en utilisant un serveur DHCP (dynamic host configuration protocol) qui distribue les adresses dynamiquement durant une durée déterminée

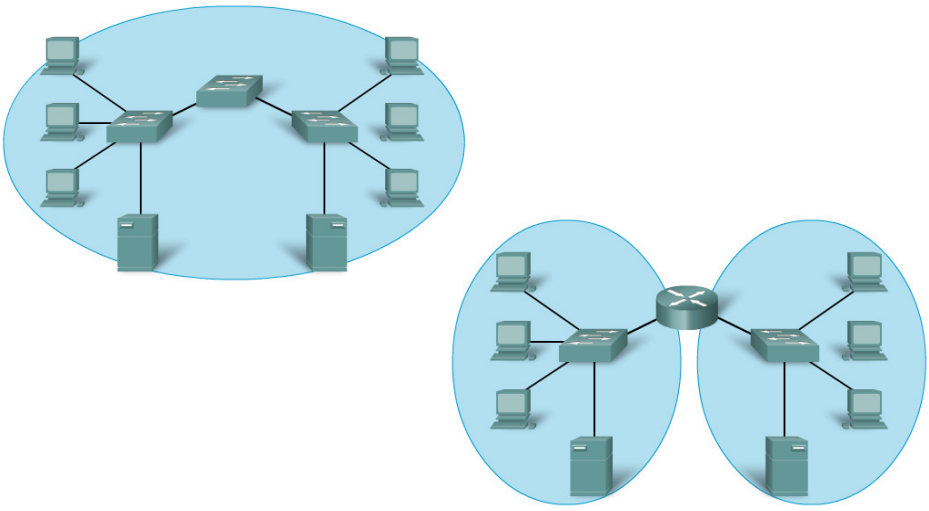
Sous réseaux

- Les sous réseaux permettent aux administrateurs réseau de diviser un grand réseau en plusieurs petits sous réseaux
- La gestion d'un réseau classe A est très compliquée => division en plusieurs sous réseaux (ex: sous réseaux par département)
- Les sous réseaux facilite le contrôle et l'administration des réseaux
- La division en sous réseaux est une affaire interne à l'organisme

Routeurs et sous réseaux

- Les sous réseaux sont connectés via des routeurs
- Le routeur est connecté par une interface à chaque sous réseau qui il interconnecte
- L'interface sur un sous réseaux fait parti de ce sous réseaux (adresse de l'intervalle du sous réseau)
- L'utilisation des routeurs limitent les domaines de diffusions => améliore les performances

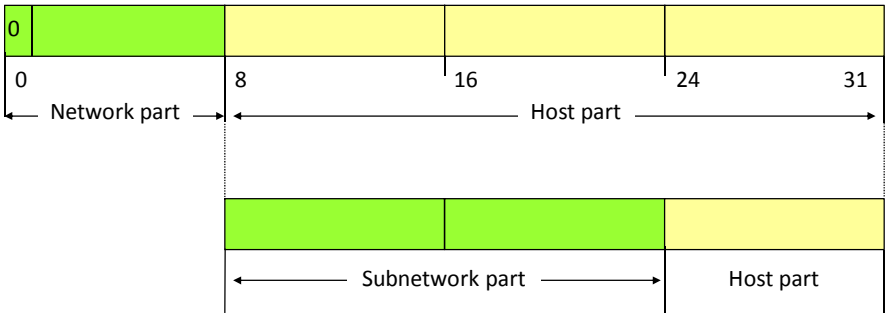
Routeurs et sous réseaux



Masque de sous réseau

- Les classes de réseau ne peuvent plus être utilisé pour distinguer entre l'adresse réseau et l'adresse machine
- Les routeurs ont besoin de connaitre l'adresse réseau pour pouvoir router le paquet
- Le masque de sous réseau permet de déterminer l'adresse réseau
- Les masques permettent de rendre quelques bits de la partie machine de l'adresse en partie réseau

Example: Class A Address



Masque de sous réseau

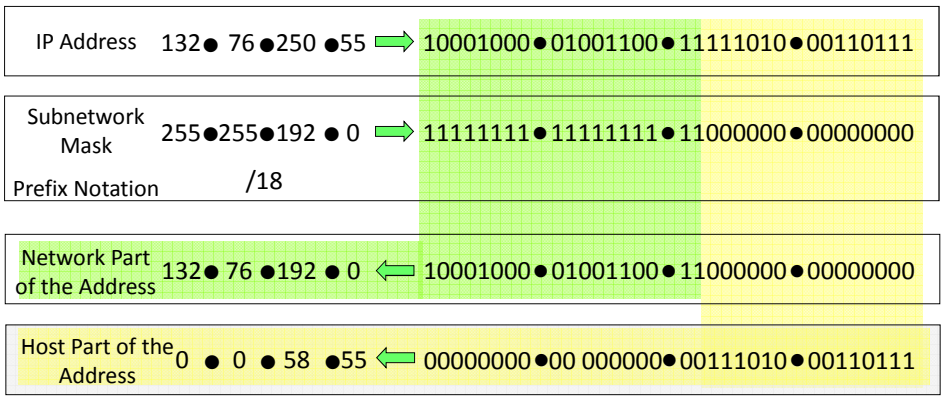
Il est utilisé:

- pour distinguer le numéro de la machine à l'adresse réseau.
- pour indiquer le nombre de bits du préfix.

Il est composé de:

- une suite contigüe de 1 dans les bits signifiants de poids fort (réseau)
- une suite contigüe de 0 dans les bits signifiants de poids faible (machine)

Masque de sous réseau



exemple

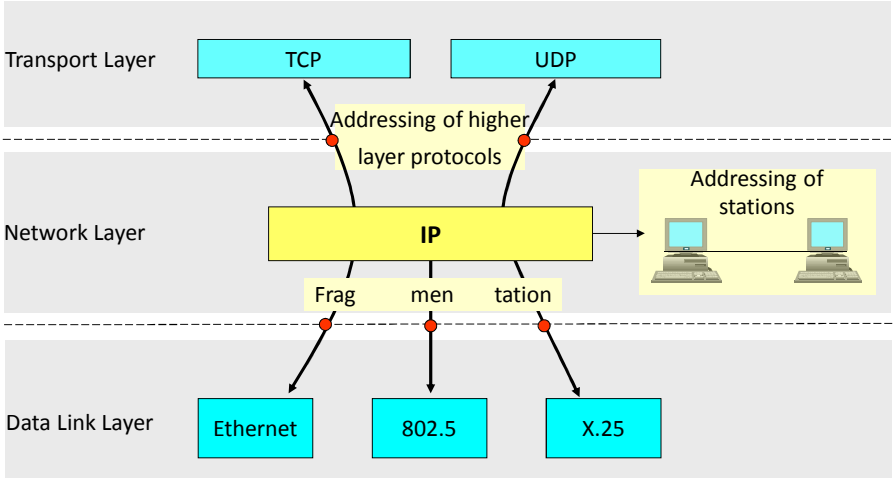
- Diviser le réseau de classe C 192.168.1.0 en 4 sous réseaux
- Donner le masque de sous réseau correspondant
- Donner les adresse des sous réseaux
- L'adresse 192.168.1.95 appartient elle à quel réseau?

Fonctions du protocole IP

- Fragmentation
- adressage des protocoles de niveau supérieur
- Adressage logique
- intégrité de la transmission
- Qualité de service

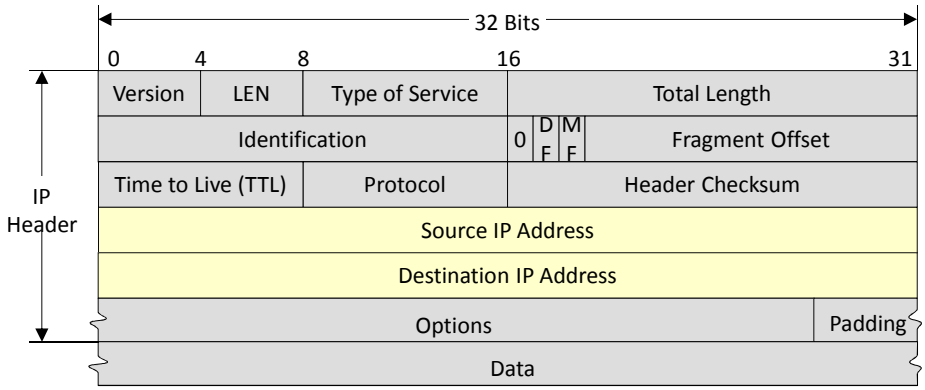
Further requirements:

- **Integrity: Delivery to the correct destination**
- **Quality of Service: Sometimes best effort is not enough**



Les champs du protocole IP

champ	description
version	La version du protocole IP (4 ou 6)
LEN	Longueur de l'entête IP
Type of service	Indication sur la qualité de service du paquet
Total length	Taille totale du paquet (entête + donnée)
identification	Identification d'un datagramme fragmenté
DF	Don't fragument
MF	More fraguments
Fragument offset	Position du fragment par rapport au paquet d'origine
Time to live (TTL)	Compteur qui se décrémente à chaque saut
Header checksum	Détection d'erreur sur l'entête
Protocol	Protocole de couche supérieur
padding	bourrage



Problème des adresses IPv4

- Le nombre des utilisateurs ne cesse d'augmenter et les réserves d'adresse sont au point de l'épuisement
- Les derniers /8 sont entrain d'être distribués
- Les registry sont très regardant pour l'attribution des adresses aux opérateurs maximum 1024 adresses pour chaque demande

Solutions possibles

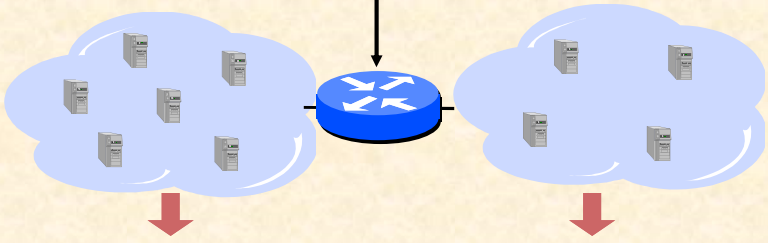
- Optimisation de l'utilisation des plages d'dresse grâce à la technique du VLSM (Variable length subnet mask)
- Passage à l'adressage IPv6 => passage obligatoire avec une phase de coexistence avec IPv4
- Solution intermédiaire: Network Address Translation

NAT

- NAT consiste à remplacer l'adresse locale (privée) par une adresse globale (publique) pour permettre l'accès à internet aux utilisateurs d'un réseau privée
- NAT est installé sur le routeur d'extrémité qui lie le réseau au monde extérieur
- Si une machine a besoin de sortir sur internet le routeur rempalce l'adresse source privée dans les paquets par une des adresses public disponible pour cette fonctionnalité

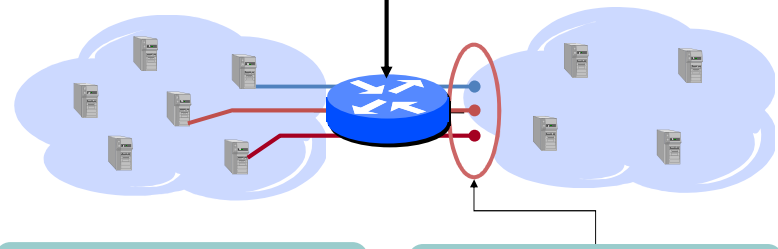
NAT: Network Address Translator

- NAT
  - **Traduit les adresses locales en adresses publiques et vis versa**
  - **Plusiuers machine privée partagent quelques adresses globales**

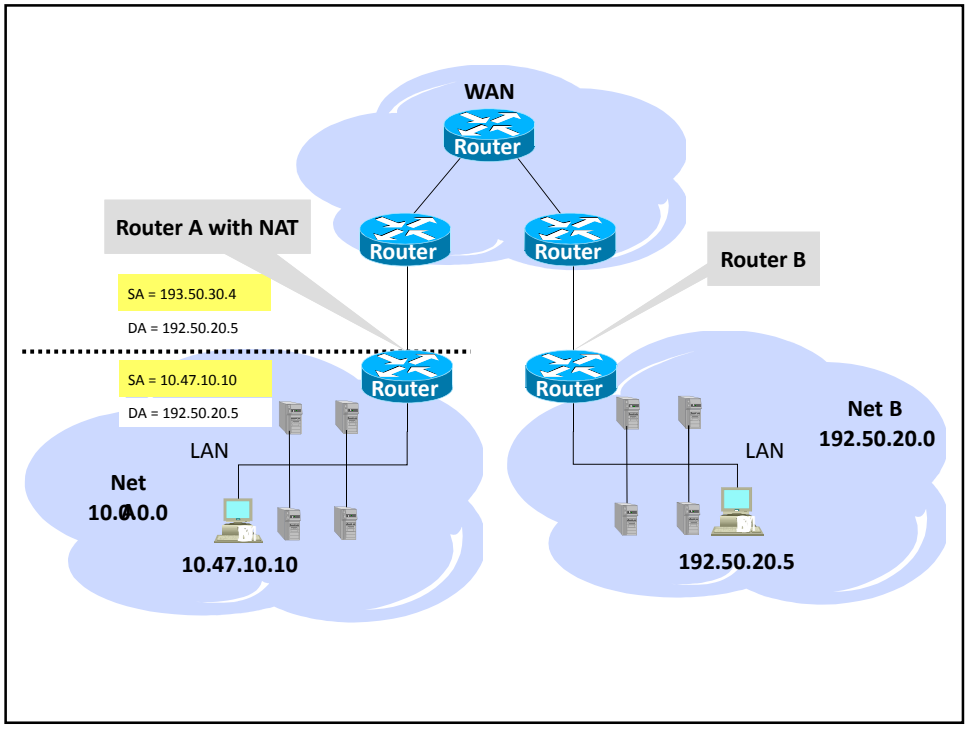


- Réseau privée
  - **Utilise des adresses locales (privées)**
  - **Les adresses locales ne peuvent pas être utilisées à l'éxterieur**
- Réseau public
  - **Utilise des adresses publiques**
  - **Les adresses publiques sont uniques globalement**

- NAT translates between local addresses and public ones
  - **NAT maps local addresses to public IP addresses**



- Réseau privé
  - **Seulement un petit nombre de Machines sont connectées au réseau public**
- Réseau public
  - **quelques adresses publiques peuvent suffir**



Quand retourner l'adresse au pool?

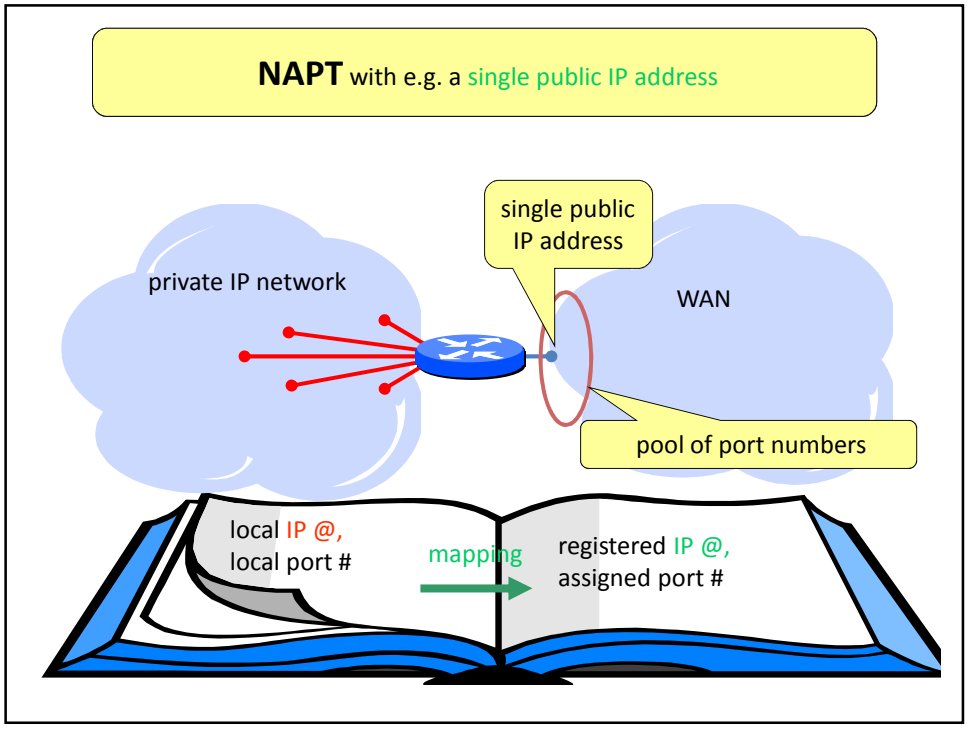
- NAT fonctionne à la demande des machines interne
- Comment il peut mettre fin à la correspondance (pas de connexion au niveau IP)
- Solution: remonter au niveau TCP pour avoir une idée sur la fin de la session ou valeur par défaut si UDP (15min)

Types de NAT

- NAT statique : la correspondance entre l'adresse privée et l'adresse public est configurée manuellement
- NAT dynamique : le choix de l'adresse publique se fait dynamiquement selon les adresses disponibles dans la pool des adresses publiques

NAPT

- L'utilisation d'une pool d'adresses publiques n'est toujours pas suffisant car le nombre de machines connectées simultanément à l'éxterieur ne doit pas dépassés le nombre des adresses dans la pool
- Solution: remplacer le couple adresse locale, numéro de port) avec le couple (adresse globale, numéro de port)
- Il est possible d'utiliser NAPT avec une seule adresse IP globale



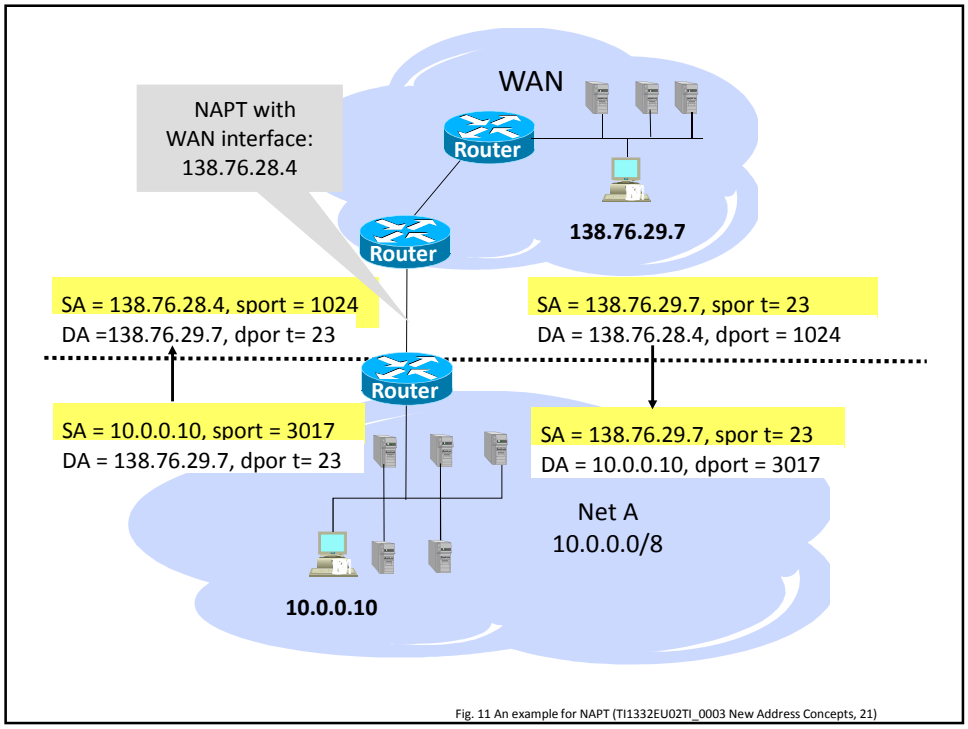


Fig. 11 An example for NAPT (TI1332EU02TI_0003 New Address Concepts, 21)

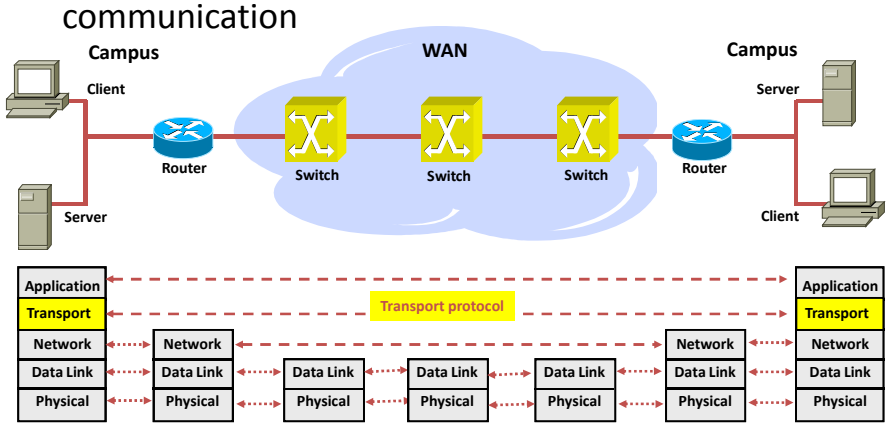
Propriétés de NAT

- Avantages
  - NAT permet de résoudre le problème de pénurie des adresses provisoirement
  - NAT est transparent pour les machines seuls les routeurs d'éxtremité qui interviennent
  - NAT permet de cacher la structure interne du réseau et rendre sa reconnaissance plus difficile
- Inconvénients
  - IP casse la connexion de bout en bout les applications qui utilise l'adresse des deux bouts doivent être ajustées aussi
  - Le routeur NAT est un point de faiblesse puisque tout le trafic doit passer par lui « single point of failure »
  - NAT affecte la performance du routeur => plus de travail pour le routeur

Protocoles de transport: TCP et UDP

Protocoles de transport

- Les protocoles de transport fournissent des services de bout en bout (End to End) les nœuds intermédiaires n'interviennent pas dans cette communication



Rôles des protocoles de transport

- Multiplexage des segments issues des applications vers la couche réseau
- Détection des erreurs au niveau bit
- Détection de perte de paquet dans les nœud intermédiaires due à une congestion
- Recouvrement: segments erronées ou perdus
- Control de flux: s'assurer que la machine distante à suffisamment d'espace mémoire pour recevoir le flux
- Contrôle de congestion: s'assurer que le réseau à la capacité pour transmettre les segments
- Les protocoles de transport ne fournissent pas tous ces services

Multiplexage

- Les protocoles d'application ne communiquent pas directement avec le réseau mais à travers le protocole de transport
- Chaque application communique avec le protocole de transport à travers un port => plusieurs applications peuvent résider sur la même machine en utilisant différent numéro de port
- Les paquets émanant de toutes les applications sont multiplexés par le protocole de transport sur la couche réseau

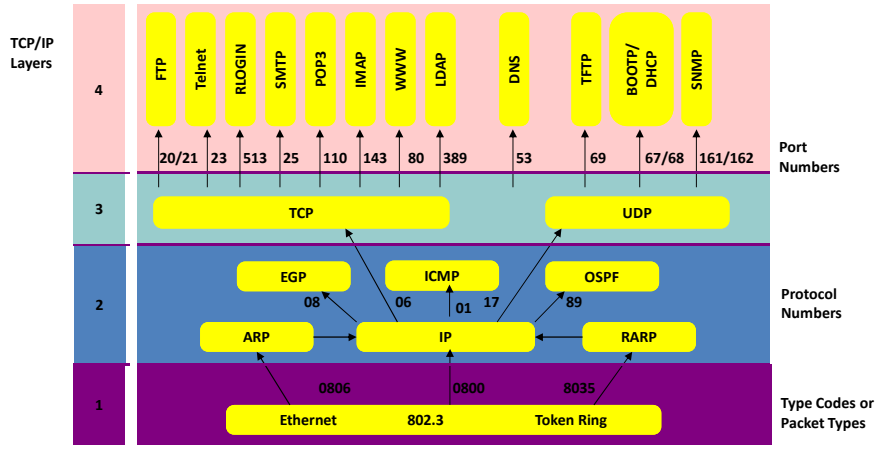
Numéro de port

- À chaque communication le processus au niveau application doit s'identifier avec un numéro de port et préciser le numéro de port de l 'application distante
- Le numéro de port est un numéro sur 16bit
- Deux types de numéro de port
  - Well known port => serveur
  - Port éphémère => client

Numéro de port

- Well known ports de 1 à 1023 qui sont utilisé par des serveurs standard comme HTTP (80) telenet (23)=> assigné par IANA (Internet Assigned number authority)
- Port éphémère de 1024 à 65535 qui sont utilisé par les clients qui n'ont pas besoin de well known port puisque ce sont eux qui initient la communication => le serveur peut répondre au port source qui se trouve dans l'entête du protocole de transport
- Le well known port est constant dans le temps alors que le port client (éphémère) est dynamique avec la condition que le triplet (protocole de transport, numéro de port, adresse IP) soit unique dans la même machine

Quelques well known ports



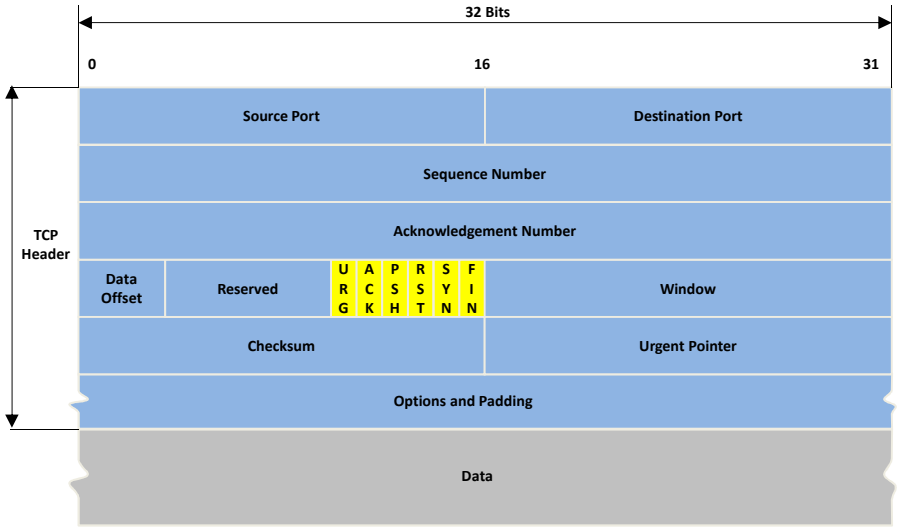
Les sockets

- Les programmeurs des applications utilisent les services réseau à travers des sockets
- Les sockets sont des API (application programming interface) qui permettent une communication entre deux processus distants
- Une communication avec socket spécifie d'une manière unique le 5 uplet suivant: (protocole de transport, adresse IP destination, port destination, adresse IP source, port source)

TCP: Transport control protocol

- TCP est un protocole de transport de bout en bout **orienté connexion et full duplex** permettant une communication inter processus distant
- TCP permet d'assurer **la fiabilité** d'envoi à l'application sur un support ou un protocole de niveau inférieur non fiable : mise en ordre, éliminer la duplication, retransmission des paquets perdus
- TCP permet de **contrôler le flux** selon la capacité du récepteur (taille du buffer de reception)
- TCP permet de **contrôler la congestion** par la régulation de la taille des données à envoyer
- TCP permet de **fragmenter et réassembler** les segments pour éviter au protocole réseau d'effectuer cette tache

Entête TCP



Les champs TCP

- Port source (16bit): identifie le port source de l'application qui envoie le segment
- Port destination (16bit): identifie le port destination de l'application distante
- Numéro de séquence (32bit): est le numéro de séquence du premier octet dans le segment sauf si SYN est validé dans ce cas le numéro de séquence est le numéro de séquence initiale
- Numéro d'acknowledgement (32bit) seulement si ACK est validé il contient le numéro de séquence du prochain segment à recevoir
- Data offset (4bit): détermine ou l'entête se termine et les données commencent c'est le nombre des mots en 32 bit contenu dans l'entête

Les champs TCP

- Reserved (6bit): réservé pour une utilisation futur doit être mis à zéro
- Flags (6bit):
  - URG: urgent pointer est signifiant (il s'a git de données urgentes)
  - ACK: le champ acknowledgment est signifiant
  - PSH: fonction push
  - RST: redémarrer la connexion
  - SYN: synchroniser le numéro de séquence
  - FIN: i l n'y a plus de données à transmettre

Les champs TCP

- Window (16 bit): taille de la fenêtre d'envoi qui indique le nombre d'octet que cette station peut recevoir (selon la taille de son buffer de réception)
- Checksum (16bit): code d'erreur qui englobe l'entête TCP et les données
- Urgent pointer (16bit): détermine la quantité de données urgent dans le segment=> le dernier octet urgent est: numéro de séquence + Urgent pointer
- Options and padding

Connexion TCP

- Avant chaque communication TCP ouvre un circuit virtuel (vu comme tel par l'application)
- Il y a deux méthodes pour l'ouverture d'un circuit virtuel:
  - OPEN passive: pour les serveurs qui attendent des requêtes des clients (port d'écoute)
  - OPEN active: pour les clients qui ouvrent la connexion par un port client à la demande de l'application

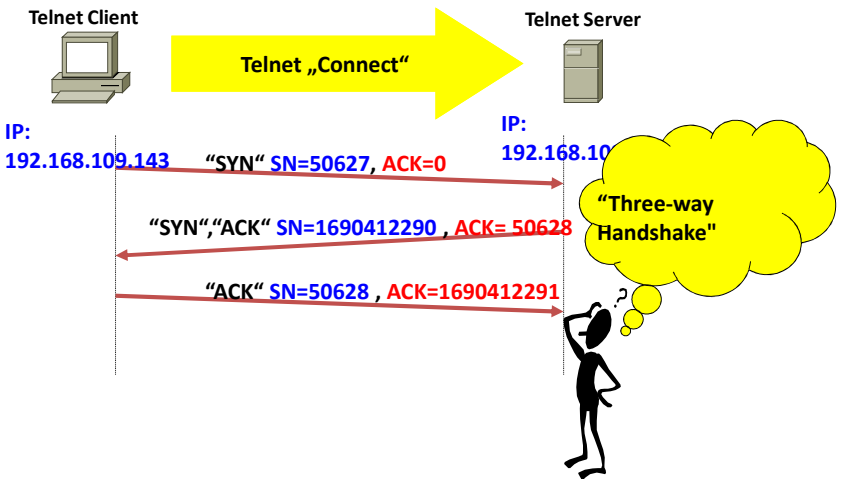
Etablissement de la connexion

- La procédure utilisée pour ouvrir un circuit virtuel s'appelle three way handshake
- Cette procédure réduit la possibilité d'une fausse connexion et permet de synchroniser les numéro de séquences
- Les segments utilisés pour l'ouverture de la connexion ne transmettent pas de données

Etablissement de la connexion

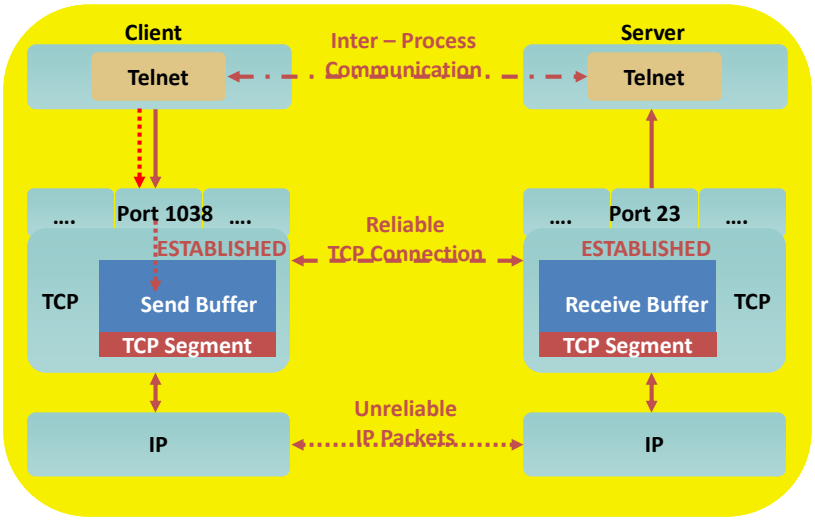
- Le client envoie un segment SYN (le flag SYN est à 1) avec le numéro de port du serveur et le numéro de séquence initial
- Le serveur répond avec son segment SYN avec le numéro de séquence initiale du serveur et acquitte (flag ACK à 1) le sgment SYN du client
- Le client acquitte le segment SYN du serveur
- Le circuit virtuel est considéré comme ouvert et les deux stations peuvent échanger des données

Etablissement de la connexion TCP



Echange de données

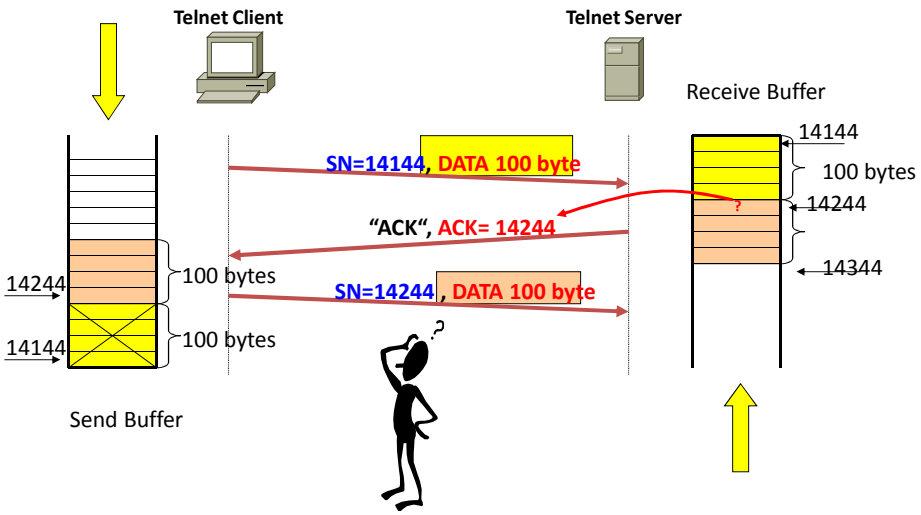
- Une fois le circuit virtuel est établit les applications commencent à échanger les données à travers les sockets
- Les données sont passé au module TCP par block (comme si l'application est entrain d'écrire sur un fichier sur le disque) est placée dans un buffer de transmission
- Périodiquement TCP envoi une portion des données vers la destination (à travers les protocoles sous jacents)
- Le module TCP de destination bufferise les données dans un buffer de réception et passé en fin de compte vers l'application



Fiabilité de transmission en TCP

- Chaque segment envoyé doit être acquitté par le retour d'un accusé de réception
- Dans un paquet Ack le flag ACK est mis à 1, et le numéro d'acknowledgment spécifie le numéro du prochain octet de données à recevoir => toutes les données jusqu'à numéro ack- 1 sont reçues avec succès
- Un acknowledgment peut transporter ou non des données dans le sens contraire (full duplex)
- Si le segment est perdu (à cause de l'infiabilité de IP) le récepteur acquitte toujours le dernier segment reçue avec succès

Exchanging Data



Fermeture de la connexion

- Une fois les application ont échangées toutes les données le circuit virtuel est fermé
- Les deux parties peuvent initier la fermeture (client ou serveur)
- Deux cas pour la fermeture de connexion TCP:
  - L'application local ou distante demande la fermeture
  - Les deux demandent la fermeture

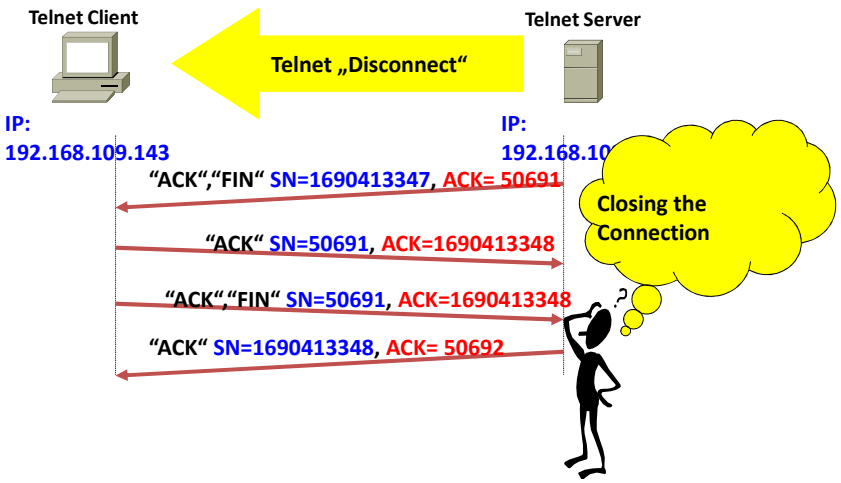
Fermeture de la connexion TCP

- L'application local demande la fermeture:
  - L'application mets un segment FIN dans la mémoire d'envoi
  - Aucune donnée n'est envoyé après l'envoie du message FIN
  - L'application peut toujours recevoir des données
  - L'application distante acquitte le segment FIN et envoie son segment FIN lorsqu'elle n'a plus de données à envoyer
  - L'application locale acquitte le segment FIN
  - La connexion est fermée

Fermeture de la connexion TCP

- Les deux machines demandent la fermeture:
  - Chaque application envoie le massage FIN
  - À la réception de ce message les deux machines attendent la réception des acknowledgment non encore reçu avant d'acquitter le segment FIN
  - les deux machines acquittent le segment FIN et ferment la connexion

Connection Termination Protocol



Travail à faire

- Analyser les échanges d'une session telnet
  - Détermination les champs TCP
  - Identifier les étapes de l'échange
  - Tracez les échanges en déterminant les numéro de séquence et numéro d'acknowledgment

Contrôle de flux: stop and wait

- Envoyer un segment de données et attendre un acknowledgment
- Si l'acknowledgment n'est pas reçu au bout d'un temps RTO (retransmission timeout) le segment est retransmis
- La fiabilité est assurée avec cet algorithme sauf que la bande passante n'est pas bien exploitée
- Le débit efficace (throughput): taille du segment/(temps de transmission + RTT)

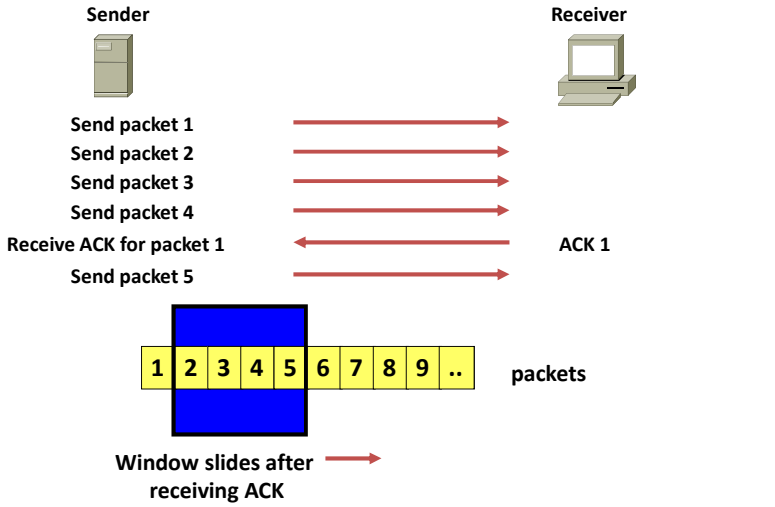
Contrôle de flux: stop and wait

- Temps de transmission= taille du segment/Débit du lien
- Le taux d'utilisation du lien est :temps de transmission/(temps de transmission + RTT) ou débit efficace/débit du lien
- Exemple: calculer le débit efficace et le taux d'utilisation d'une liaison stellite avec un délai de bout en bout de 250 ms et un débit de 56kbit/s, la taille du segment est 500octet

Contrôle de flux: fenêtre glissante

- Pour optimiser l'utilisation de la bande passante disponible une fenêtre de segment peut être envoyé selon les règles suivantes:
  - Les segments de la fenêtre sont envoyés les uns après les autres sans attendre les Ack et l'expéditeur déclenche un timeout pour chaque segment
  - Le récepteur doit acquitter chaque segment reçu en indiquant le numéro de séquence du dernier segment reçu avec succès
  - L' expéditeur glisse la fenêtre à chaque Ack reçu

The Window Principle (cont.)



Contrôle de flux: fenêtre glissante

- Si un segment est perdu par ex segment 2 le récepteur va acquitter le segment 3, 4 toujours avec ack1 et la fenêtre est stoppée
- Une fois le RTO expire segment 2 est retransmis et le récepteur l'acquitte avec ack4 => la fenêtre glisse de 4 segments à la fois
- Si un ack est perdu et les autres qui suivent sont reçu l'éxpéditeur conclut que les segments sont reçu
- La taille de la fenêtre ne peut pas excéder le nombre d'octet déterminé dans le champ window

Contrôle de flux: fenêtre glissante

- Le débit efficace= w*taille du segment/(temps de transmission +RTT)
- Appliquer le contrôle de flux avec une fenêtre de 5 segments (2500octet) sur l'exemple précédent
- Le mécanisme de la fenêtre glissante permet
  - Une transmission fiable
  - Une meilleure utilisation de la bande passante
  - Un control de flux puisque le récepteur peut retarder les ack suivant l'état de son buffer

Ajustement des timeouts

- Délais variables sur Internet
  - Distance variable entre l'expéditeur et le destinataire
  - Temps de mise en attente dans les fils d'attente des routeurs intermédiaires selon l'état du réseau (charge, saturation des liens …)
- Le timeout est calculé en fonction de l'estimation du RTT (Round Trip Time)
- Moyenne pondérée des RTT mesurés permet de trouver une valeur de timeout pour le segment suivant

168

RTT et Timeout

Comment choisir la valeur du RTO?

- Doit être plus long que le RTT
  - RTT est variable
- Trop court: timeout prématuré
  - Retransmissions pas nécessaires
- trop long: réaction lente au perte => diminue les performances

Comment estimer le RTT?

- RTT est variable => utiliser une valeur lisse du RTT (tenir en considération les RTT du passé)
- SampleRTT: mesuré à partir de l'envoi du segment jusqu'à la reception des ACK
- EstimatedRTT = (1-x)*EstimatedRTT + x*SampleRTT
- Valeurs de x 0.1 , 0.2 => moins d'influence pour une nouvelle valeur du RTT

169

RTT et Timeout

Valeur du RTO

- EstimtedRTT plus une marge de sécurité
- Grande variation du EstimatedRTT => une marge plus large
- RTO = EstimatedRTT + 4*Deviation

170

Approches de controle de congestion

Deux approches pour contrôler la congestion

End-to-end congestion control:

- Pas de feedback explicite du réseau
- congestion déduite par les terminaux observée par des pertes ou délais excessifs
- approche abordé par TCP

Network-assisted congestion control:

- Les routeurs fournissent un feed back au terminaux
- Exemple: RED (random early detection) Congestion imminente -> notification à la source en perdant un paquet
- La congestion est calculée en mesurant la taille moyenne des queues dans les routeurs

171

Contrôle de congestion par TCP

- Principe:
  - Trouver le point d'équilibre:
    - Augmenter la fenêtre de contrôle de congestion
  - Détection de la congestion par l'indication de la perte d'un paquet
  - Suppression de congestion en réduisant la fenêtre
- Algorithme:
  - phase 1: slow start
    - Démarrage en douceur
  - phase 2: congestion avoidance
    - Eviter la congestion

172

Slow Start

- La taille de la fenêtre de congestion est initialisé à 1
- Croissance exponentielle de la taille de la fenêtre (x2 à chaque fois que les paquets sont transmis correctement)
  - Augmentation de 1 segment à chaque acquittement
- Arrêt après une perte de paquet ou en atteignant le seuil (SSthreshold)
- SSthreshold: définit le seuil entre la phase slow start et la phase congestion avoidance

173

Congestion avoidance

- Pour stopper l'augmentation trop rapide (le slow start est exponentiel)
- À partir du SSthreshold : augmentation de 1 segment à chaque RTT=> augmentation linéaire
- Le seuil vaut la moitié de la fenêtre lors de la dernière congestion (correspond à un RTO)

174

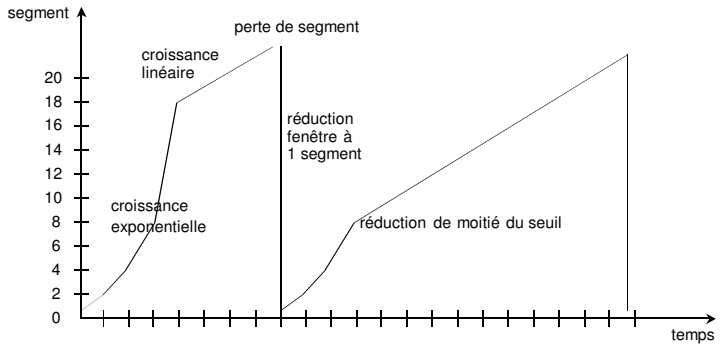
Algorithme de contrôle de Congestion

- Version tahoe:
  - Initialisation avec Cwnd=1 et ssthresh=65536
  - Phase de slow start : dupliquer le nombre de segments à chaque ack
  - TCP n'excéde pas la fenêtre de réception => envoie au maximum min(Cwnd, fenêtre de réception)
  - Si timeout (cad perte), alors cwnd=1 et ssthresh = max(cwnd/2, 2)
  - Revenir à slow start
  - Si Cwnd=ssthresh alors congestion avoidance (croissance linéaire de la taille de la fenêtre)
- La version Reno ajoute une amélioration: à la réception de 3 ack dupliqués sans atteindre le RTO diviser la fenêtre de congestion par 2 et le slow start ne redémarre pas on entre en congestion avoidance

175

Contrôle de congestion

- Evolution de cwnd



176

User Datagram Protocol: UDP

- Pas toutes les applications ont besoin de la fiabilité de transmission ou de contrôle de flux
- TCP ne peut pas transmettre les paquets de type braodcast ou multicast
- Pourquoi alors ne pas encapsuler les paquets directement sur IP?
- UDP ne fournit aucun service mais offre une interface à l'application à travers une socket UDP

Propriétés UDP

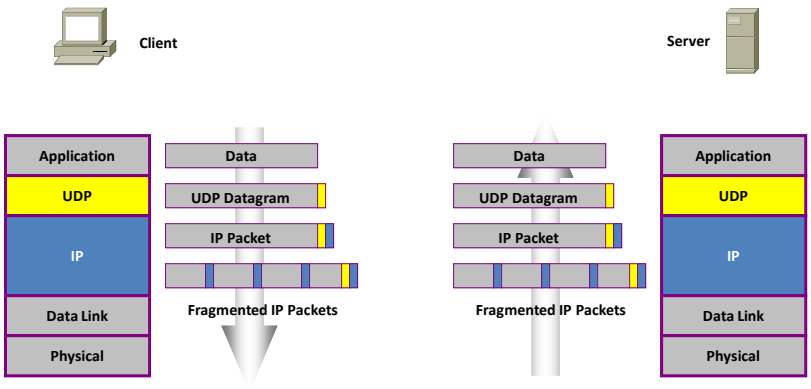
- Protocole léger: entête court
- Pas de fiabilité de transmission ou mise en ordre des paquets
- Multiplexage à travers les ports
- Sans connexion
- Support du broadcast et multicast

=> UDP hérite les caractéristiques du protocole IP

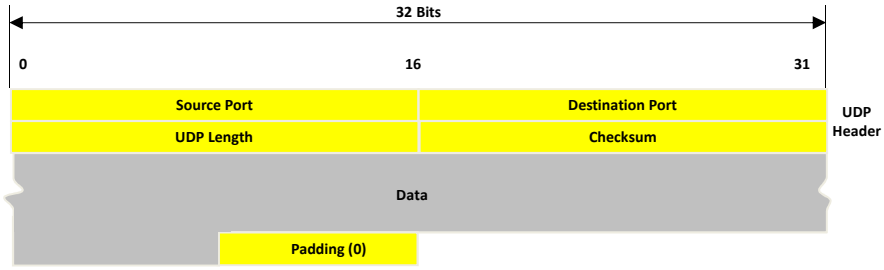
Procédure d'envoi

- Les données reçu de l'application sont rassemblées sous forme de datagramme et sont envoyés à la destination => pas de notion de flux
- Si la taille du datagramme est supérieur au MTU (Maximum Transfer Unit) le protocole IP se charge de la fragmentation du paquets
- À la réception de tous les fragments et le leur réassemblage que UDP présente le message à l'application destinataire

Datagram-centric Transport



Les champs UDP



Les champs UDP

- Port source (16bit): identifie le port source de l'application qui envoie le datagramme
- Port destination (16bit): identifie le port destination de l'application distante
- UDP length (16bit): longueur du datagramme y compris l'entête UDP
- Checksum (16bit): code d'erreur qui englobe un pseudo entête et les données => le pseudo entête permet d'inclure l'adresse IP dans le calcul du checksum

Le pseudo entête