Le routage IP

Next hop(e)

QUIZ : Rappel IP

Adresse réseau et broadcast de 172.67.146.155/16 ?

=> Réseau : 172.67.0.0/16
Broadcast : 172.67.255.255

Particularité de l'adresse 10.13.246.42 ?

=> Adresse IPv4 pour réseau privé (RFC 1918) (routable)

Particularité de l’adresse fe80::5b5e:35fa:8c55:ba68 ?

=> Adresse IPv6 unicast lien local (Link-Local)
Communication sur le lan uniquement (non-routable)

Particularité de l’adresse fd21:515e:8f6::1 ?

=> Adresse IPv6 unicast locales uniques (Unique Local)
Pour réseau privé (RFC 4193) (routable)

Sommaire

De quoi s'agit-il ?

Le routage

Internet Protocol

Rappels

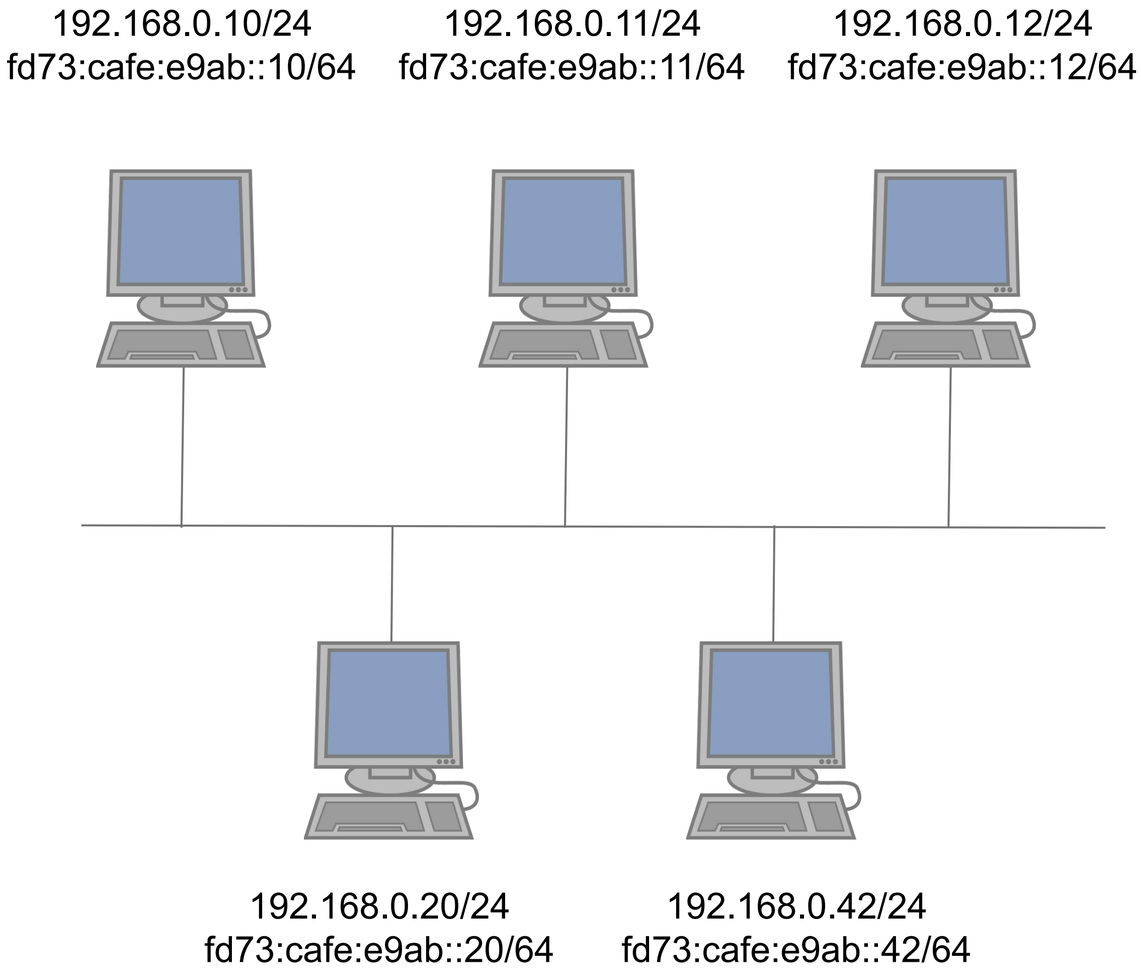
- Protocole pour faire de l'interconnexion de réseaux (de l'internet)
- Les noeuds d'un même réseau IP (logique) doivent être sur le même lien (réseau physique)
- Comment permettre la communication entre nœuds de réseaux différents ?
- à l'aide :
  - d'intermédiaires, des passerelles appelées routeurs
  - d'une technique : le routage

Schéma exemple 1

Un exemple illustré

Un réseau physique avec :

- 1 réseau IPv4 : 192.168.0.0/24
- 1 réseau IPv6 : fd73:cafe:e9ab::/64
- Configuration des noeuds homogène (même configuration réseau)



Exemple d'une communication

Un exemple illustré

Envoi de noeud 11 -> noeud 42

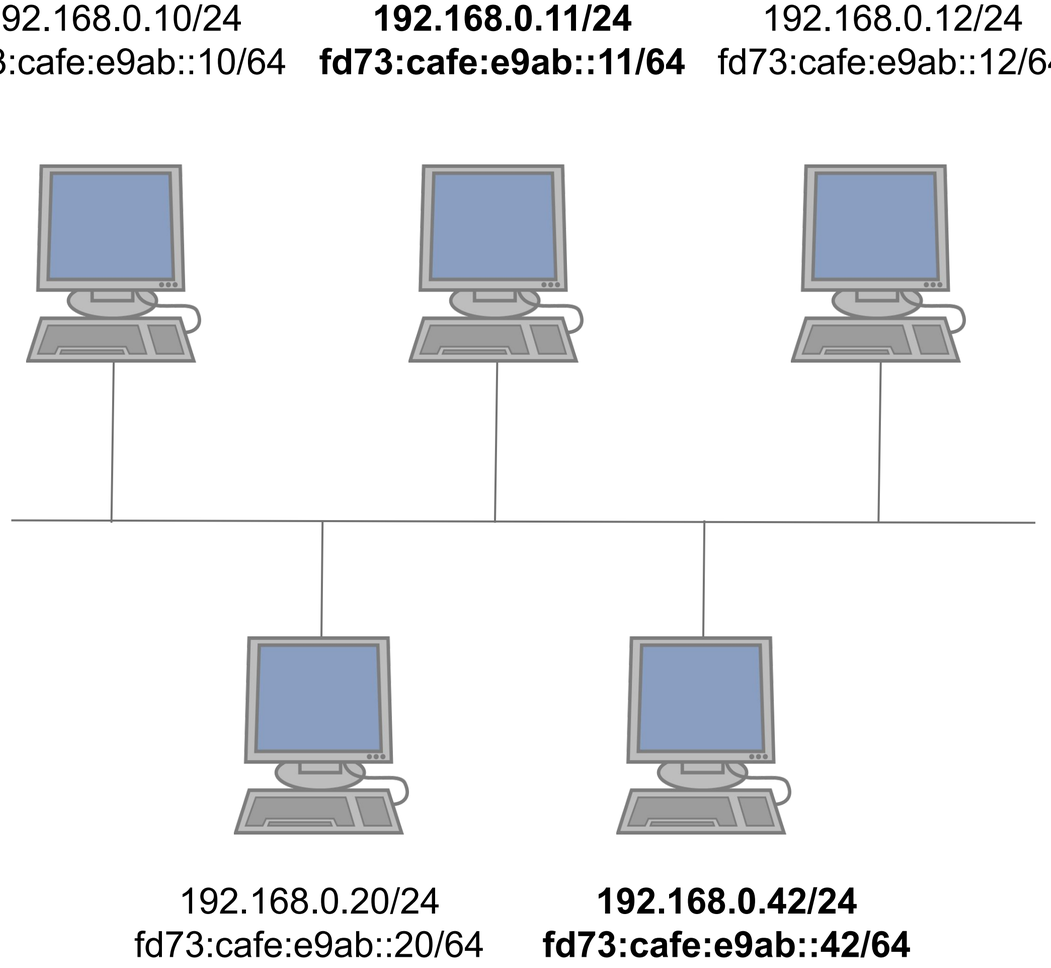
Par exemple :

ping 192.168.0.42 depuis 192.168.0.11

ou

ping fd73:cafe:e9ab::42 pour v6

Que se passe-t-il sur la machine 11 ?



Principe général

Un exemple illustré

IP sait que :

- S’il veut envoyer un paquet à une interface sur le même réseau
  => il envoi directement via le lien (la couche du dessous, ex : Ethernet)

Pour savoir si le destinataire est sur le même réseau :

- La machine prends toutes les configuration IP de toutes ses interfaces et calcule les réseaux correspondants

- Si la destination fait partie du réseau d'une de ces adresses => envoi sur l'interface physique correspondante !

Sur hôte 11

Un exemple illustré

Pour IPv4, la machine 11 dispose (au moins) des adresses :

- 192.168.0.11/24 sur l'interface (par exemple) enp0s3
- Toutes les adresses de 127.0.0.0/8 sur l'interface lo

Calculons les réseaux correspondants :

- 192.168.0.11/24 => Réseau 192.168.0.0/24

En IPv6, même chose :

- fd73:cafe:e9ab::42/64 => Réseau fd73:cafe:e9ab::/64 sur **enp0s3**
- ::1/128 => Une seule adresse

Sur hôte 11

Un exemple illustré

Dans les 2 cas, l'adresse de destination est sur le réseau de l'interface enp0s3
- On encapsule donc le paquet IP dans une trame ethernet

Pour construire la trame ethernet, il nous faut :
- adresse MAC de destination
- adresse MAC source

Pour la **source**, ce sera celle de l'interface choisie (enp0s3 dans l'exemple).
Pour la **destination**, on fait appel à **ARP** (en v4) ou **NDP** (en v6) pour récupérer l'adresse MAC correspondant à l'adresse IP de destination.

Le routeur

Sortir du lien

Et si le destinataire n'est sur aucun des réseaux de l'émetteur ?

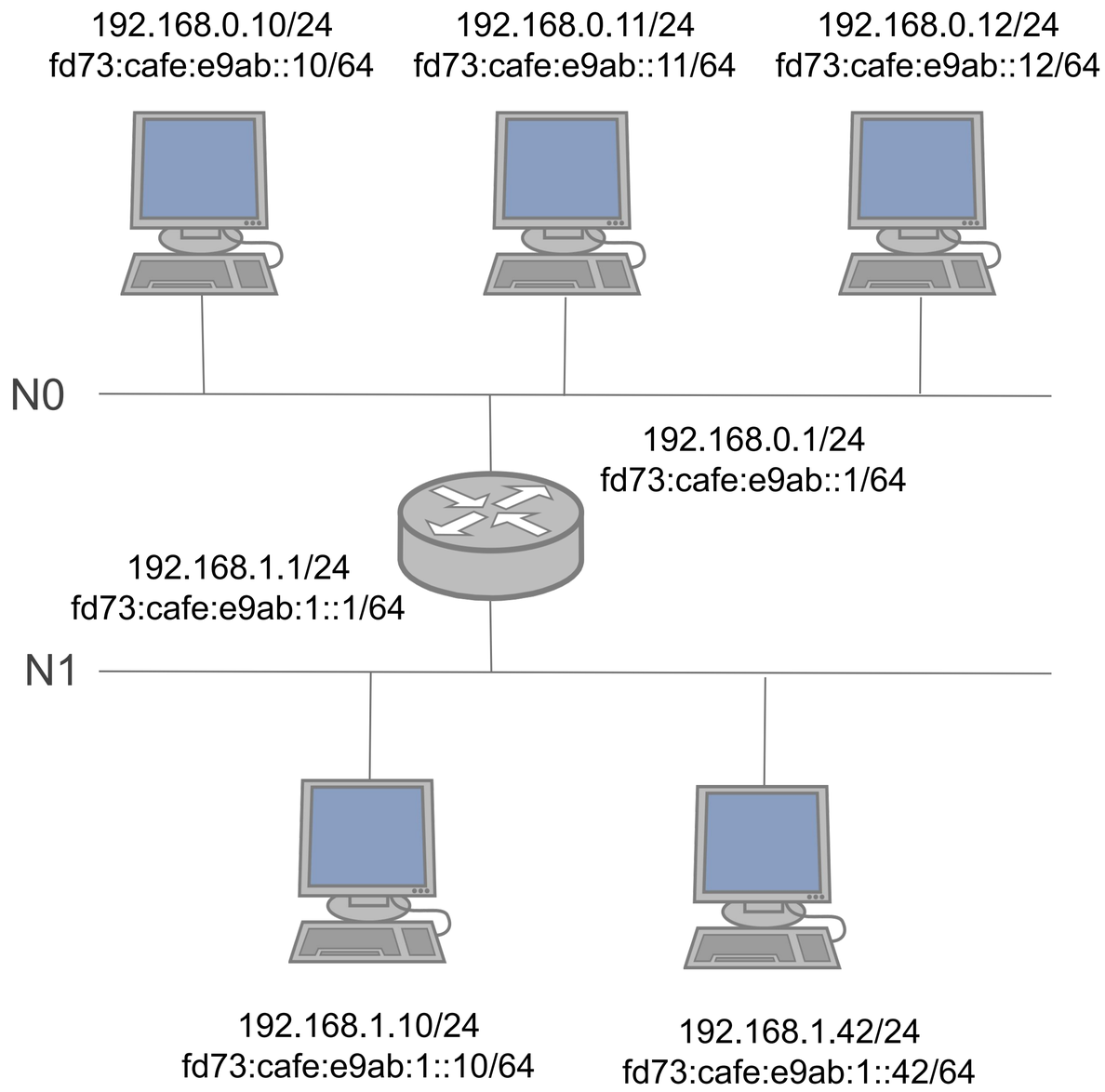
- Il lui faut un intermédiaire qui soit sur le même réseau que l'expéditeur donc joignable par le mécanisme précédent
- et que cet intermédiaire permette, directement, ou indirectement d'aller jusqu'au destinataire

Comment connaître ces intermédiaires potentiels (routeurs) ?

- La liste des routeurs directement accessible d'une machine ainsi que les adresses qu'ils peuvent permettre de joindre fait partie de la configuration d'un noeud IP et s'appelle: **la table de routage**

Schéma exemple 2

Un exemple illustré (suite)



Un réseau physique en plus avec :

- 1 réseau IPv4 : 192.168.1.0/24
- 1 réseau IPv6 : fd73:cafe:e9ab:1::/64

Un routeur présent sur les 2 réseaux :

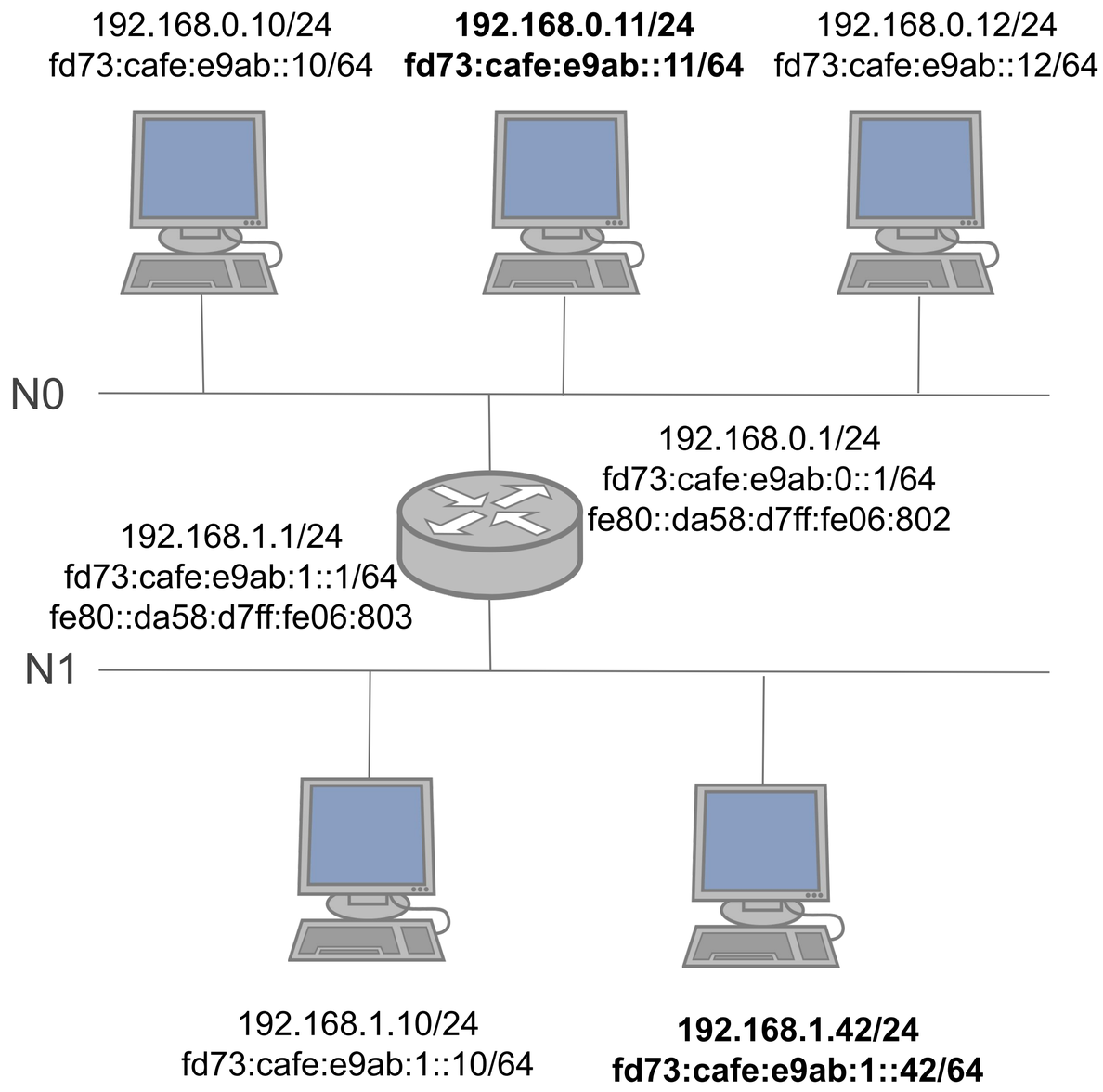
- 192.168.0.1 et fd73:cafe:e9ab::1
- 192.168.1.1 et fd73:cafe:e9ab:1::1

Cas 2

Un exemple illustré (suite)

Envoi de noeud 11 (sur N0)
-> noeud 42 (sur N1).
Par exemple :
ping 192.168.1.42 depuis 192.168.0.11
ou ping fd73:cafe:e9ab:1::42 pour v6

Que se passe-t-il sur la machine 11 ?



Sur hôte 11

Un exemple illustré

Hôte 11 calcule les réseaux auxquels il est directement connecté
=> le destinataire n'en fait pas partie

Hôte 11 consulte sa **table de routage**

- Pour que la communication soit possible (dans ce cas précis) elle doit comporter une ligne disant par exemple :
  - Pour aller à 192.168.1.0/24 (réseau destination), passe par 192.168.0.1 (interface du routeur sur N0)
  - Pour aller à fd73:cafe:e9ab:1::/64 (réseau destination), passe par fe80::da58:d7ff:fe06:802 (interface du routeur sur N0)

Sur hôte 11 (suite)

Un exemple illustré

Ou, comme l'hôte 11 n'a accès qu'à un seul routeur, une indication plus générale :

- Quand tu ne sais pas, envoi à 192.168.0.1/fe80::da58:d7ff:fe06:802
  => Passerelle par défaut (default gateway)

Sur hôte 11

Un exemple illustré

Hôte 11 doit donc envoyer :

- Son paquet IP pour 192.168.1.42 / fd73:cafe:e9ab:1::42
- Dans une trame à destination de l'adresse MAC correspondant au routeur
    - 192.168.0.1 en IPv4
    - fd73:cafe:e9ab::1 et fe80::da58:d7ff:fe06:802 en IPv6

Donc, le paquet **IP** conserve l'**adresse destination finale**
Mais la requête **ARP/NDP** est faite sur l'**adresse du routeur** et la trame est envoyée à l'**adresse MAC du routeur**

Sur le routeur

Un exemple illustré

Le routeur reçoit une trame qui lui est adressé
- i.e. l'adresse MAC destination est la sienne

L'ethertype présent dans la trame indique qu'elle contient IP
=> Donc il analyse le paquet IP contenu dans la trame

L'adresse IP de destination n'est pas la sienne
- Si on était dans le cas d'un hôte classique du réseau => on jette le paquet

Mais ce noeud est un routeur, donc il doit transmettre le paquet
=> Il procède à la même mécanique d'envoi de paquet IP, mais pour ce paquet dont il n'est pas l'émetteur

Les tables de routage

C'est quoi ?

Chaque noeud d'un réseau IP (routeur ou pas) dispose d'une table de routage.
Chaque entrée de cette table est composée (au moins) :

- Une destination (adresse de réseau et masque)
- Saut suivant (next hop) : l'adresse de la passerelle pour y aller

Note : en IPv6, on utilise les adresses lien local des routeurs/passerelles
Eventuellement on peut y trouver aussi :

- L'interface associée : celle par laquelle ce noeud peut joindre la passerelle
- Une métrique : une mesure de la qualité de cette route (moins = mieux)
- parfois d'autres choses (comme le protocole utilisé)

Un moment de réflexion

Admettons que les destinations dans la table de routage soit des adresses d'interface...

Combien y aurait-il d'entrées dans une table de routage, au max ?

En IPv4 : 2^{32}

En IPv6 : potentiellement 2^{128}

On aurait donc à la fois un problème de stockage de la table et un problème de temps nécessaire à router un paquet IP

Encore une histoire de réseau

1 réseau = un ensemble de machine

Une adresse de réseau au sens IP caractérise un ensemble d'interfaces situées au même endroit dans le réseau global

En utilisant des adresses de réseau comme destination dans la table de routage => on en raccourcis drastiquement la taille !

Plus les réseaux sont grands plus la table de routage est petite

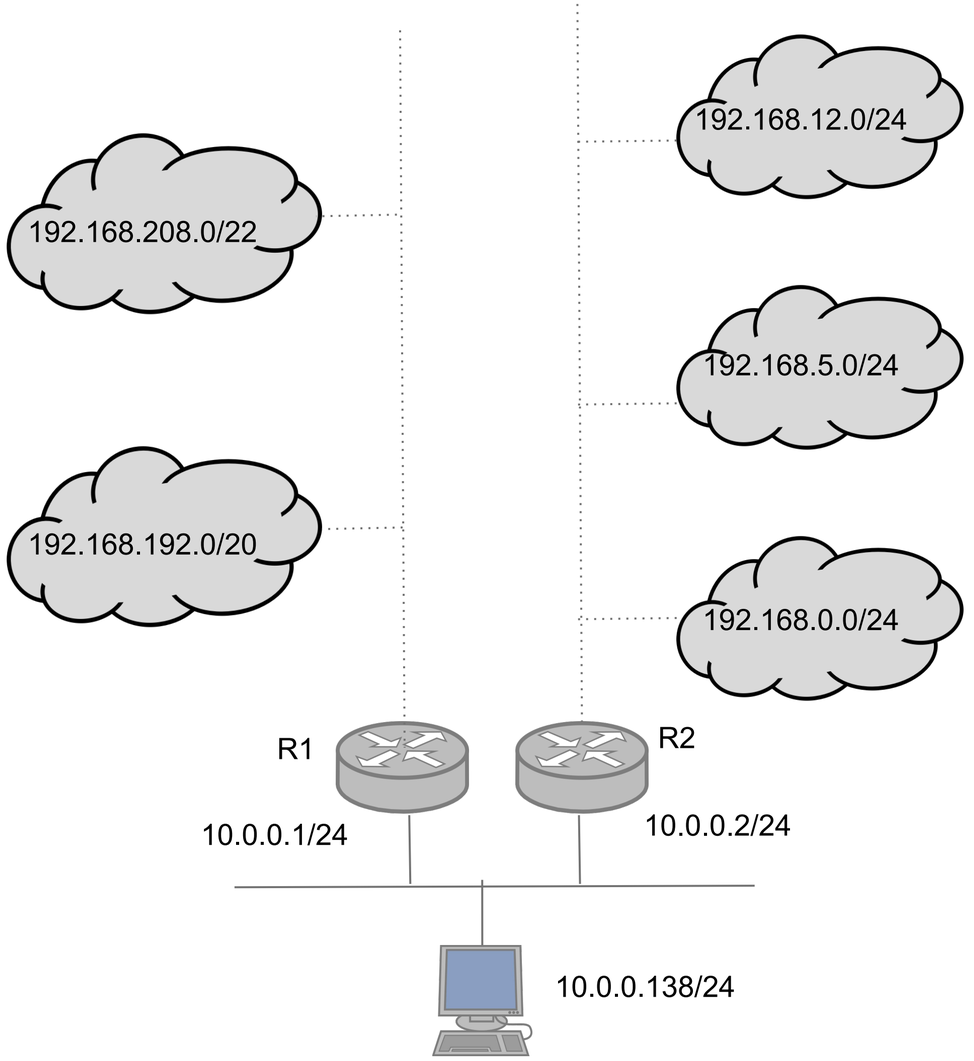
On utilise donc des sur-réseaux comme préfixe de routage en agglomérant, quand c'est possible, plusieurs adresses de réseau en une seule.

Construire une table

Un exemple illustré de table de routage

Supposons le cas suivant :
Un noeud : 10.0.0.138 est sur le même réseau que 2 routeurs R1 : 10.0.0.1 et R2 : 10.0.0.2
R1 permet d'accéder indirectement aux réseaux :
- 192.168.208.0/22
- 192.168.192.0/20

R2 permet d'accéder indirectement aux réseaux :
- 192.168.0.0/24
- 192.168.5.0/24
- 192.168.12.0/24



Construire une table

Un exemple illustré de table de routage

Dans ce cas :

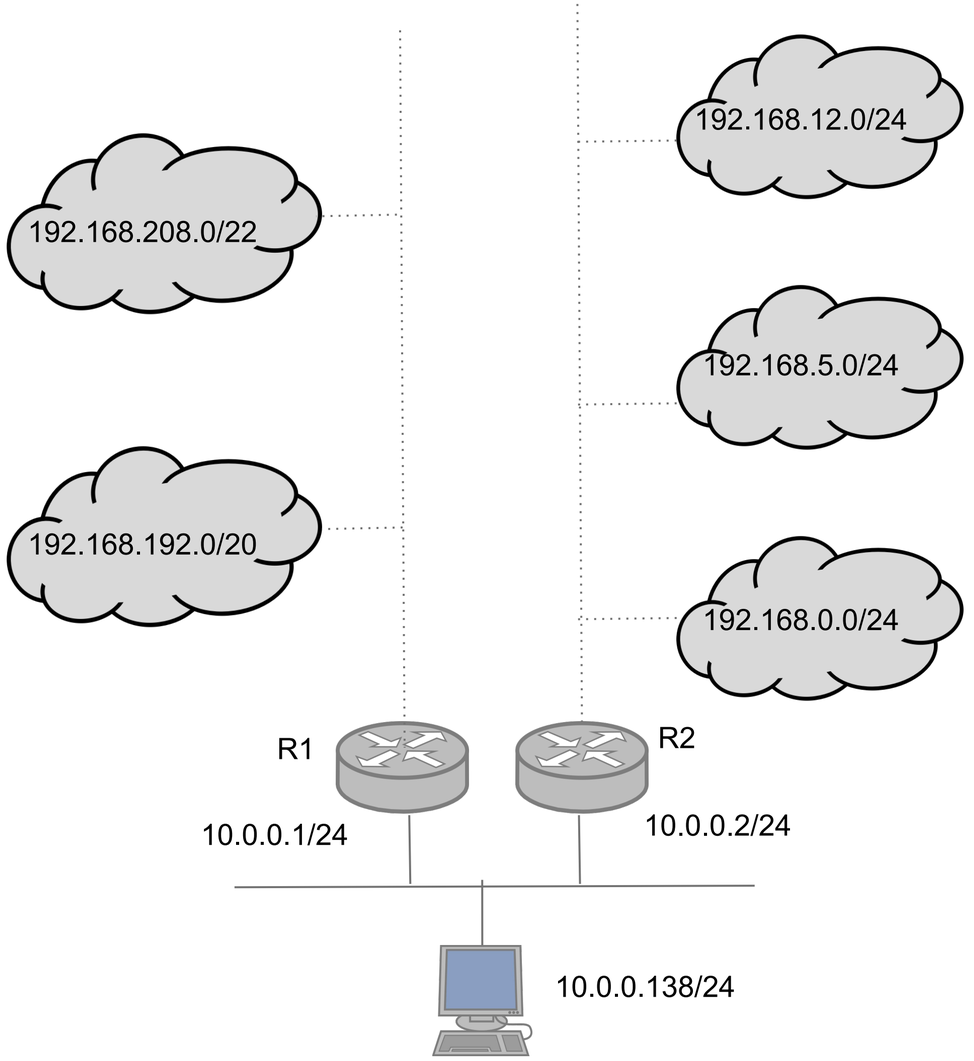
- On peut créer 2 préfixes de routages
- Chaque préfixe englobant tous les réseaux accessible via la même passerelle

Par exemple : *3ème nombre décimal* <128 ou >=128

La table de routage de 10.0.0.138 :

- 192.168.128.0/17 via 10.0.0.1
- 192.168.0.0/17 via 10.0.0.2

Note : principe identique en IPv6 mais avec les adresses lien local des interfaces des routeurs



La commande ip route (Unix)

Manipuler la table de routage

Quelques exemples :

ip route : affiche table de routage ipv4

ip -6 route : affiche table de routage ipv6

ip route add 192.168.128.0/17 via 10.0.0.2 : ajout d'une route

ip route add default via 10.0.0.254 : ajout d'une passerelle par défaut

Pour une configuration persistante => /etc/network/interfaces

```
wilder@host:~$ ip route
default via 10.0.0.254 dev enp0s3 proto dhcp metric 600
10.0.0.0/24 dev enp0s3 proto kernel scope link src 10.0.0.138 metric 600
192.168.128.0/17 via 10.0.0.2 dev enp0s3
192.168.0.0/17 via 10.0.0.1 dev enp0s3
```

Activer le routage

Transformer un linux en routeur

Par défaut, la plupart des linux sont configurés comme hôte de réseau

- c'est à dire : ils jettent les paquets qui ne leur sont pas destiné

Mais le noyau linux sait faire du routage

Pour activer le routage sur linux, on modifie un paramètre du système :

- **net.ipv4.ip_forward** (0 = hôte, 1 = routeur)
- **net.ipv6.conf.all.forwarding** (0 = hôte, 1 = routeur)

La commande sysctl (linux)

Paramétrage routeur d'un linux

La commande sysctl :

- **sysctl net.ipv4.ip_forward** : consultation de l'état du routage IPv4
- **sysctl -w net.ipv4.ip_forward=1** : activer routage v4 (temporaire)
- **sysctl net.ipv6.conf.all.forwarding** : consultation de l'état IPv6
- **sysctl -w net.ipv6.conf.all.forwarding=1** : activer routage IPv6 (temporaire)

Pour une configuration persistante => **/etc/sysctl.conf** (fichier de configuration)

- **sysctl -p /etc/sysctl.conf** : recharger la configuration depuis le fichier

La commande route (Windows)

Manipuler la table de routage

Quelques exemples :

- **route print** : affiche des tables de routage ipv4 et ipv6
- **route add 192.168.128.0/17 10.0.0.2** : ajout d'une route
- **route delete 192.168.128.0/17** : retrait d'une route

Ou via CmdLet PowerShell :

- **Get-NetRoute** : affiche la table de routage
- **New-NetRoute** : ajout d'une route
- **Remove-NetRoute** : retrait d'une route

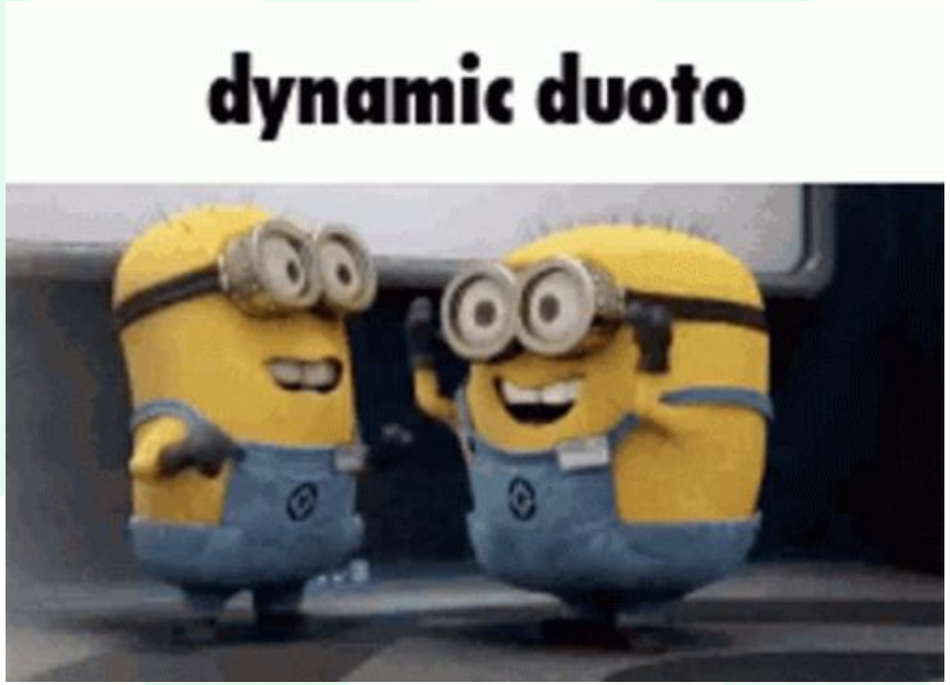
En conclusion

Vers le routage avancé

La construction de tables de routage statique (définie manuellement) par les administrateurs d'un réseau est envisageable pour des petits réseaux subissant rarement des changements.

Pour des réseaux moins stable et de plus grande envergure, il est nécessaire de passer à du routage dynamique et de déployer des protocoles dédiées comme OSPF et BGP.

Routage dynamique



Une définition

Vers le routage avancé

Le routage dynamique est essentiel, surtout dans de grandes architectures réseau comportant plus de 5 routeurs.

Dynamique = commutation automatiquement vers des liaisons de secours (pannes) et MAJ automatiques des tables de routage entre les routeurs.

Les protocoles dynamiques

Vers le routage avancé

RIP : adapté pour les petits réseaux

EIGRP : pour les grands réseaux, uniquement sur du matériel Cisco

OSPF : également pour les grands réseaux, protocole ouvert

BGP : utilisé pour le routage sur internet

RIP (Routing Information Protocol)

Vers le routage avancé

Principalement utilisé dans les petits réseaux en raison de sa simplicité.

Port par défaut : UDP 520.

Limitations: Maximum 15 sauts, pas adapté pour les grands réseaux.

3 versions :

- RIPv1 et RIPv2 (pour IPv4)
- RIPng (pour IPv6)

RIP ne prend pas en compte l'état de la liaison (corrigé dans OSPF).

EIGRP (Enhanced Interior Gateway Routing Protocol)

Vers le routage avancé

Protocole conçu par Cisco pour ses matériels.

Utilisé dans les réseaux complexes.

Port par défaut : 88.

Avantages: Convergence rapide, prend en charge plusieurs protocoles de couche réseau

OSPF (Open Shortest Path First)

Vers le routage avancé

Principalement utilisé dans les réseaux de tailles moyenne.

Port par défaut : 89.

Avantages: Supporte les sous-réseaux de taille variable, plus rapide que RIP

BGP (Border Gateway Protocol)

Vers le routage avancé

Ce protocole est utilisé pour le routage sur internet entre les différents systèmes autonomes.

Il est utilisé pour l'échange d'informations de routage entre différents ISP (Internet Service Provider)

Port par défaut : TCP 179

Protocoles de transport

OSI - TCP/IP

Rappel

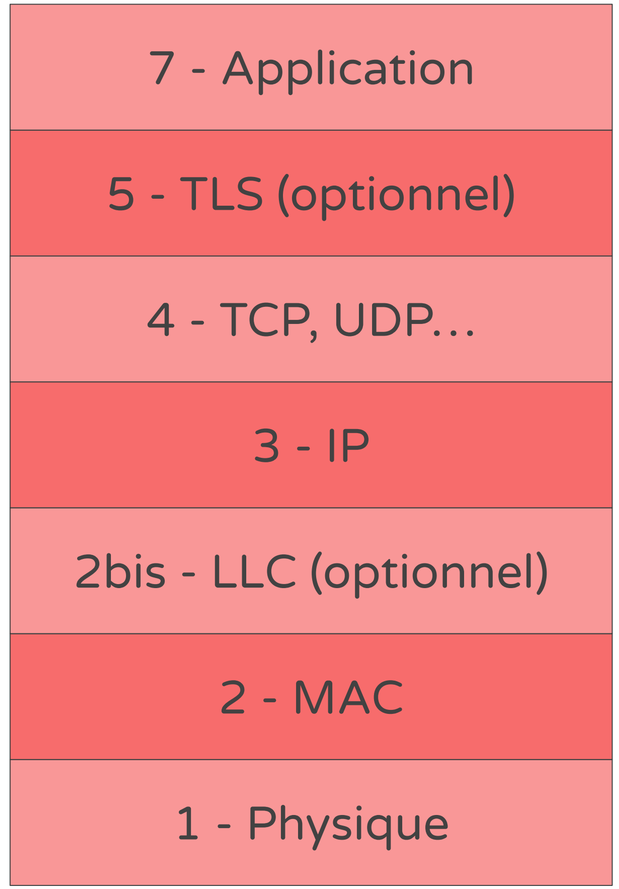
Le protocole IP

- Couche 3 (réseau) au sens OSI
- Censé véhiculer un protocole de couche 4 (transport)

Il existe plusieurs protocoles de transport

Les 2 plus courants sont :

- TCP : Transmission Control Protocol
- UDP : User Datagram Protocol



Un moment de réflexion

Supposons le cas d'un système :

Une des interfaces de ce système reçoit un paquet IP.

L'adresse de destination du paquet est bien une adresse de cette interface.

Les entêtes IP indique l'identifiant du protocole de couche 4 (TCP ou UDP).

Et ensuite ?

Ce paquet IP, à quel processus du système, à quelle couche applicative faut-il l'envoyer ?

Est-ce une requête pour un serveur ?

Est-ce une réponse à une requête précédente ?

C'est un des problèmes que résout la couche transport.

Basculer d'une communication entre interfaces en une communication entre processus (application).

UDP

Commençons par le plus simple

UDP - User Datagram Protocol

- Défini dans la RFC 768 (STD 6) en 1980 par David P. Reed
- Conçu comme un protocol minimal (TCP light)
- Permet la communication entre processus (application)
- Transporté par IP (v4 ou v6)
- Numéro de protocole : 17

La notion de port

Des identifiants de processus

UDP (et TCP) utilise la notion de port

Un port est un identifiant de processus au sein d'une interface

Une communication est donc caractérisée par :

- un couple adresse IP/port de destination
- un couple adresse IP/port source

De la même façon qu'il ne peut pas y avoir 2 adresses identiques sur un réseau, il ne peut pas y avoir 2 ports identiques sur une même adresse

Serveurs et client

Retour sur les notions de serveur et client

Un **serveur** est un processus qui attend des **requêtes** de la part de **clients**

Un **client** est un processus qui envoie une **requête** à un **serveur** (et en général attends une **réponse**).

C'est donc toujours le client qui initie une communication.

Il doit donc connaître l'adresse IP et le port du serveur.

Lorsqu'un serveur est lancé, il est lié à (au moins) un port sur une interface (adresse IP) du système sur lequel il est lancé.

Les ports

On ne fait pas n'importe quoi avec les ports

Les ports sont codés sur 16 bits (de 0 à 65535)

On distingue 3 plages :

- 0 à 1023 : Les ports systèmes (Well Known Ports) - Serveur
- 1024 à 49151 : les ports utilisateurs (Registered Ports) - Serveur
- 49152 à 65535 : les ports dynamiques (Ephemeral Ports) - Client

Les numéros de ports serveurs sont enregistrés à l'IANA pour des protocoles applicatifs habituels (voir la liste). Leur utilisation n'est pas obligatoire, mais lorsqu'un serveur tourne sur son port standard, les clients le connaissent implicitement (et donc n'ont pas besoin de le préciser). Par ex : DNS = port 53 (en TCP ou en UDP)

Ces 3 plages sont valables pour UDP et pour TCP, même si un port UDP est bien distinct d'un port TCP (et peuvent donc coexister sur la même machine)

Le datagramme UDP

Vous prendrez bien un peu d'entête ?

L'entête d'un **datagramme** UDP a 4 champs :

Le port source (16 bits)

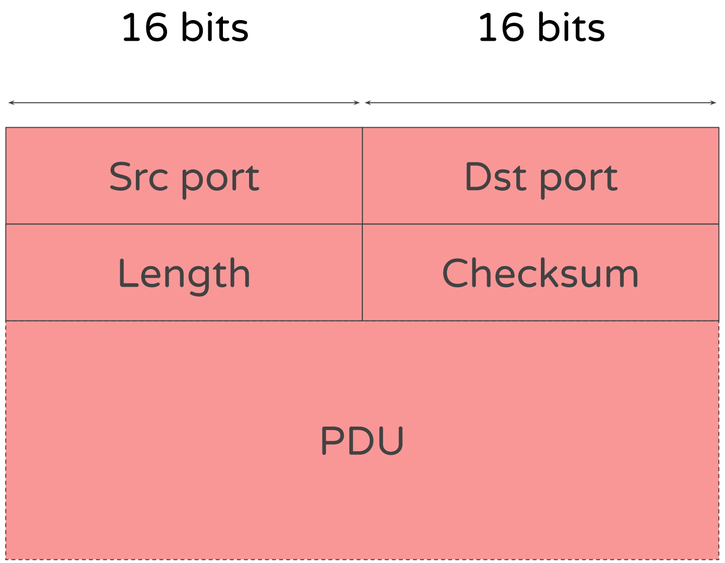
Le port destination (16 bits)

La taille du datagramme (16 bits)

Entête + PDU

La somme de contrôle (16 bits)

calculée sur entête UDP + PDU + pseudo entête IP



Le datagramme UDP

Les services rendu par UDP

En conclusion

UDP est un protocole de couche 4 particulièrement léger

Il fourni une communication entre processus (port)

Permet d'identifier certaines erreurs de transmission

- Contrôle de la taille
- Vérification de la somme de contrôle

Mais n'assure pas de fiabilité

- Si les contrôles échouent (taille invalide ou checksum invalide)

=> Datagramme jeté par le destinataire sans prévenir l'expéditeur

Adapté pour certaines applications (DNS, Diffusion d'information, Streaming...)

TCP

L'autre moitié d'IP

TCP - Transmission Control Protocol

- Défini dans la RFC 793 (STD 7)
- Protocole fiable en mode connecté
- Établissement d'une connexion bi-directionnelle
- Garantie du séquencement (ordre)
- Cherche à optimiser l'utilisation du réseau
- Permet la communication entre processus (application)
- Transporté par IP (v4 ou v6)
- Numéro de protocole : 6

Une connexion fiable

Assurer la réception

Transmission de **segments** TCP : TCP découpe le PDU en **segments**

Chaque **segment** est associé à un **numéro de séquence** par la **source**

À l'**émission** d'un **segment**

=> démarrage d'un **compte à rebours** pour ce **numéro de séquence**

Envoi d'un **acquittement** par le **destinataire** à réception

À **réception** de l'**acquittement** par la source

=> **suppression** du segment et du **compte à rebours**

À la fin d'un compte à rebours

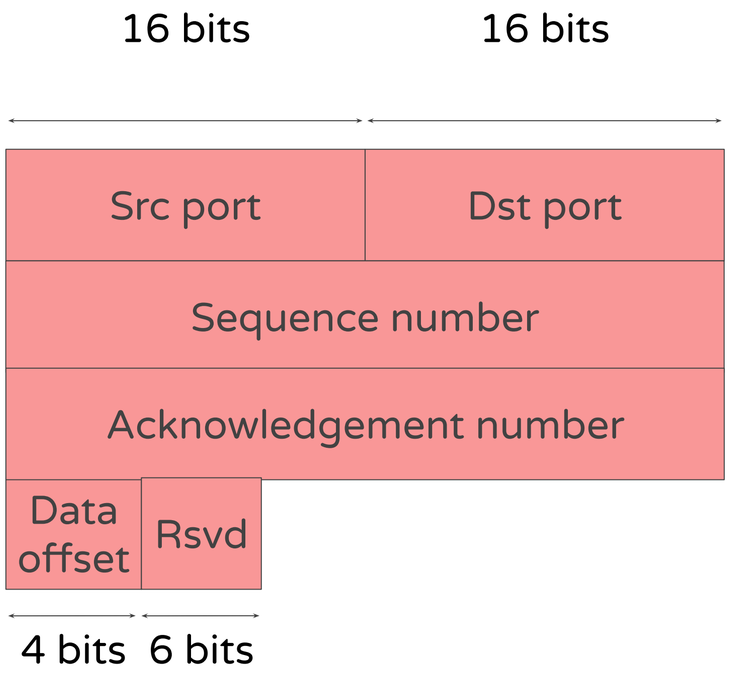
=> **ré-émission** du segment

Le segment TCP

Entête saison 2

L'entête d'un segment TCP a 15 champs (+ options)

- Le port source (16 bits)
- Le port destination (16 bits)
- Numéro de séquence (32 bits)
- Numéro d'acquittement (32 bits)
- Prochain numéro de séquence attendu
- Décalage des données (4 bits) :
- Taille de l'entête TCP en mots de 32 bits
- Réservé (6 bits) : Tous à 0



Le segment TCP

Entête saison 2

Les bits de contrôle (6 bits)

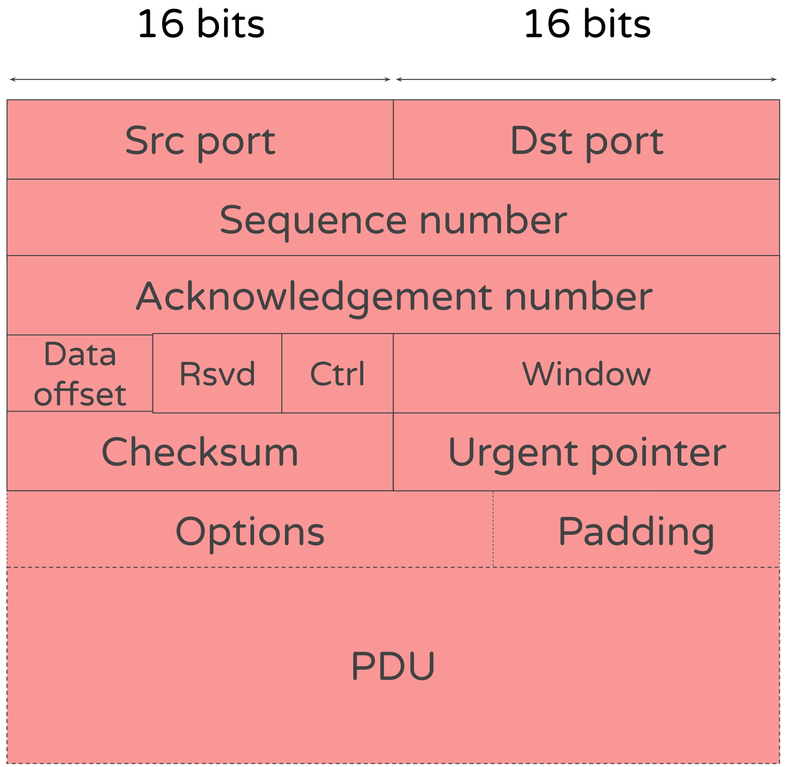
- URG : Présence d'un segment urgent
- ACK : Contient un acquittement
- PSH : Outrepasser le tampon
- RST : Reset connexion
- SYN : Synchronisation N° de séquence
- FIN : Fin de connexion (plus de données)

Taille de la fenêtre TCP (16 bits) :

- nombre d'octets attendus

Somme de contrôle (16 bits)

Pointeur données urgente (16 bits) : si URG = 1



Le segment TCP

Poignée de main TCP

The three way handshake

Lorsque qu'un serveur est en écoute

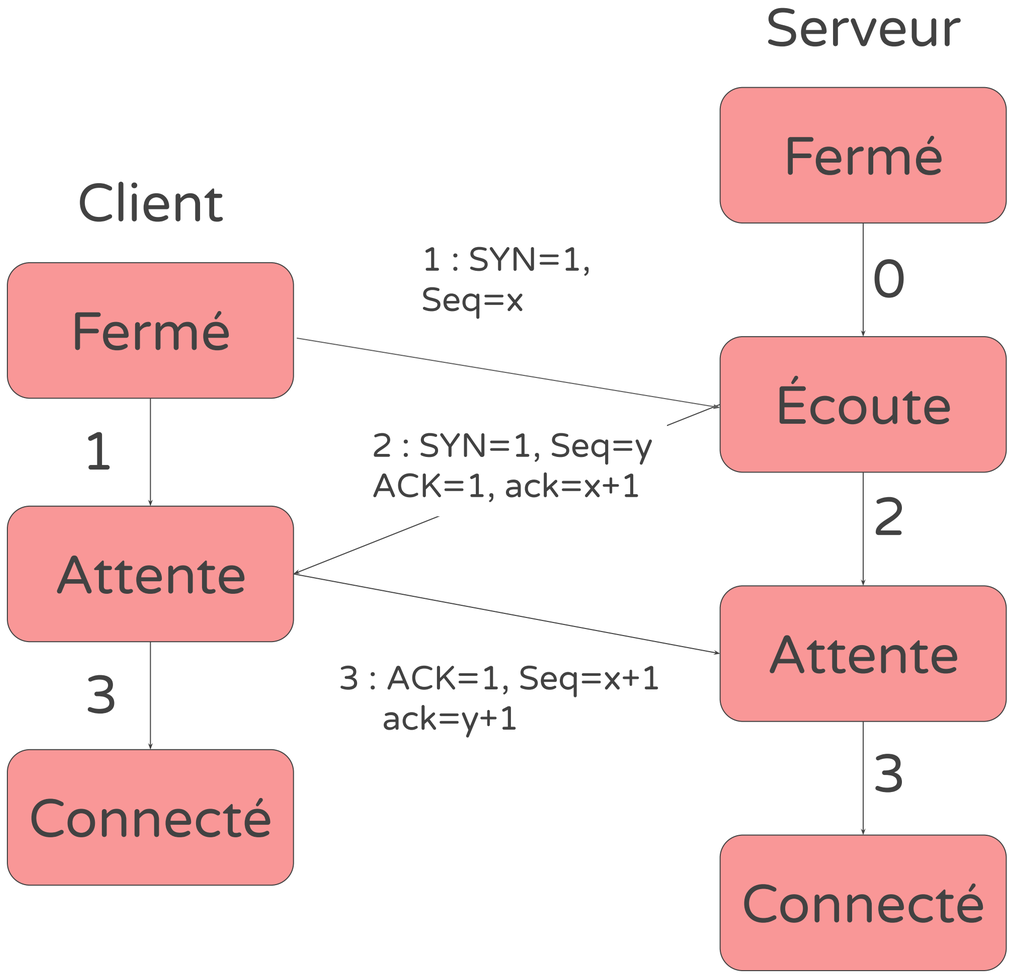
Un client peut se connecter :

C->S demande de connexion

- Flag SYN à 1
- **C** choisi un numéro de séquence
- Le segment porte ce numéro

S->C accepte la connexion

- Flag SYN à 1
- **S** choisi un numéro de séquence
- ACK = 1



Une connexion bidirectionnelle

Du full duplex !

Une fois la connexion établie

Chaque côté peut envoyer des segments à l'autre :

- A condition de respecter les numéro de séquence, c'est à dire chaque **nouveau segment** doit porter le **numéro suivant** le précédent
- Les segments peuvent tous contenir aussi des acquittements (pas de segments spécifiques à l'acquittement)

Acquittement et débit

La gestion du débit

Attendre un acquittement avant d'émettre la suite nuit gravement au débit

- Débit ≃ 1 segment (Taille max négociée par TCP : MSS) / (2 * latence)

Donc TCP n'attends pas pour émettre plusieurs segments

- Mais IP ne garanti par l'ordre des paquets

La fenêtre TCP (champs **Window**) sert au destinataire à indiquer combien d'octets il est prêt à recevoir.

- Réserver de la mémoire pour stocker les segments en attente
- Remettre les segments dans l'ordre une fois l'ensemble reçu

La gestion de la taille de la fenêtre est abordée dans la RFC 7323

Le mécanisme d'acquittement

L'autre moitié d'IP

Emission d'un segment TCP :

- Numéro de séquence = Numéro précédent + 1 (sauf pour réémission)
- Si ACK = 1 (le paquet contient un acquittement)
  - Numéro d'acquittement = le dernier numéro de séquence reçu sans coupure + 1
  - X+1 indique : J'ai tout bien reçu jusqu'à X

Fin de connexion

This is the end

Quand le client ou le serveur n'a plus rien à envoyer :

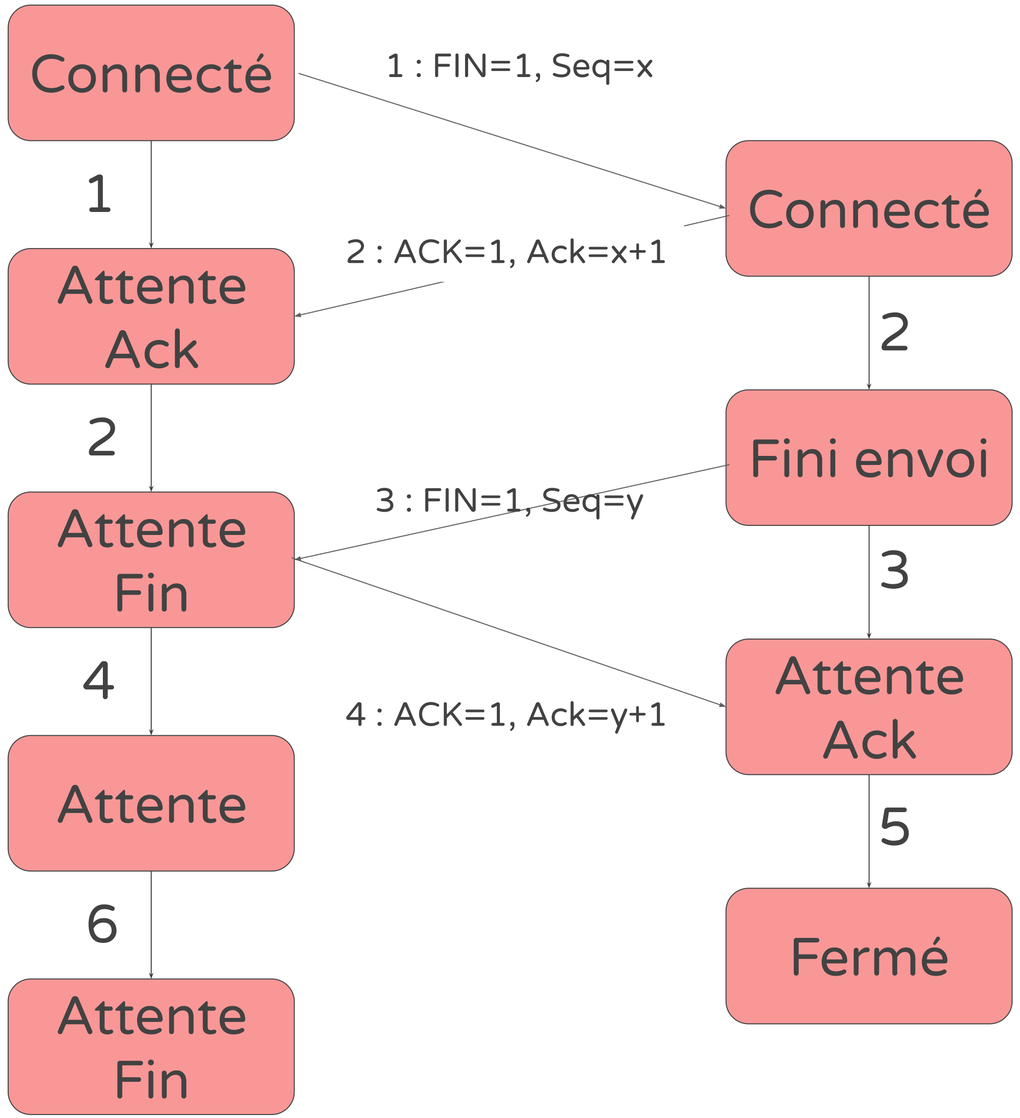
- Envoi d'un segment (N° de séquence suivant)
  - Flag FIN à 1
- Acquittement de la demande
  - Plus besoin de tampon

Quand l'autre coté à fini aussi

- Procédure identique

Temps d'attente final en cas de perte du ack

- Un nouveau FIN arrivera qui devra être acquitté



Aller plus loin avec TCP

Et plus encore !

La gestion des congestions :

- une première approche : Algorithme_TCP sur WikipediA

Les drapeaux URG et PSH :

- une courte explication (🇬🇧) : TCP Flags : PSH and URG sur PacketLife.net

Les services rendu par TCP

En conclusion

TCP est un protocole de couche 4 offrant :

- Une connexion fiable entre processus (port)
- Gestion des retransmissions
- Gestion de l'ordre des segments
- Optimisation de la bande passante et gestion de la congestion

Inconvénients : Protocole assez lourd

- Pas disponible sur les très petits ordinateurs (embarqué, IoT)
- Coûteux pour des services pas toujours nécessaires

MERCI

Des questions ?
Des remarques ?