MS CyMar

UI1.5 : Rappels réseaux et informatique

Cybersécurité des systèmes maritimes et portuaires

Recueils de Cours-TP

Christophe LOHR - Isabel AMIGO

2021

Sommaire

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Cybersécurité des systèmes maritimes et portuaires

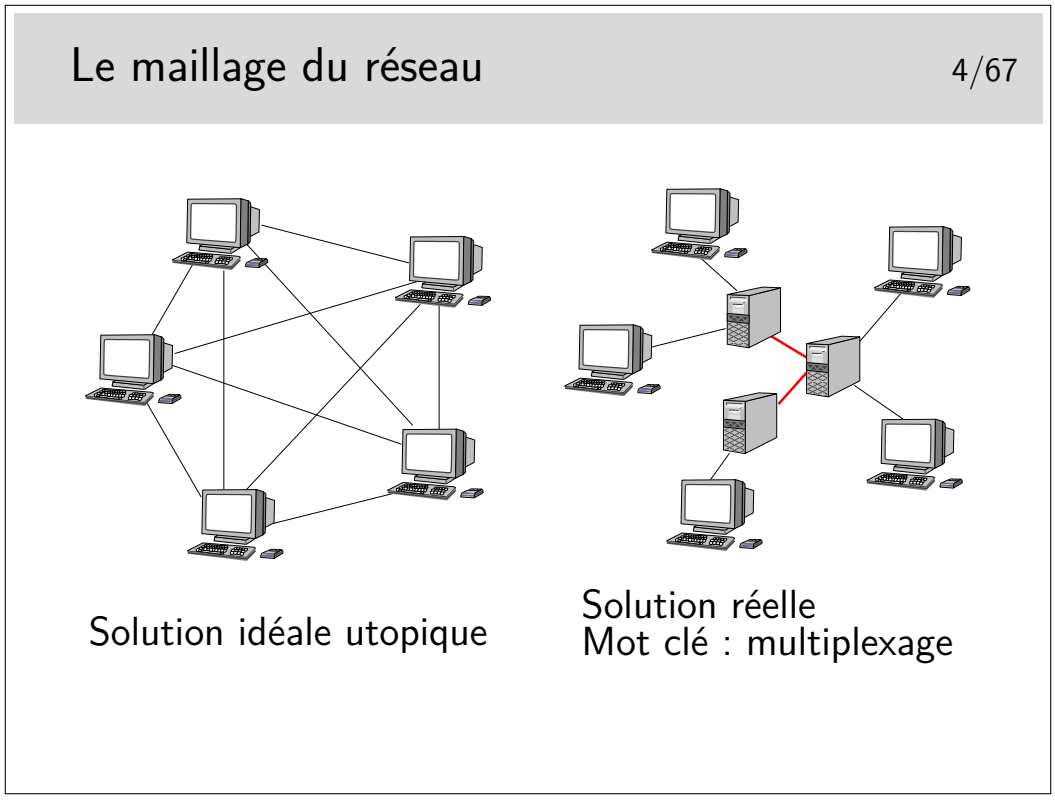
Le transport de l'information

Christophe Lohr

2021

1 Les concepts fondamentaux

1.1 Multiplexage



Les terminaux ne peuvent pas être tous interconnectés. Cette solution serait trop coûteuse en nombre de liaisons et ces dernières seraient la plupart du temps sous utilisées.

Les terminaux sont reliés à des machines intermédiaires, des relais, qui concentrent le trafic et acheminent les divers flux d'information sur des supports qui les relient.

Les divers flux de trafic sont acheminés sur les liens inter-noeuds de manière simultanée ou quasi simultanée. On dit qu'ils sont *multiplexés* et les liens entre les noeuds sont appelés de *multiplex*. Une des fonction des noeuds est d'assurer le *multiplexage* des informations sur les liens.

Un multiplex est donc une voie de communication sur laquelle on véhicule plusieurs «communications» à la fois. (Notez les guillemets, il reste à définir ce qu'est une «communication», ce n'est pas si simple)

Multiplexage et multiplexes 5/67

Différents types de multiplexes

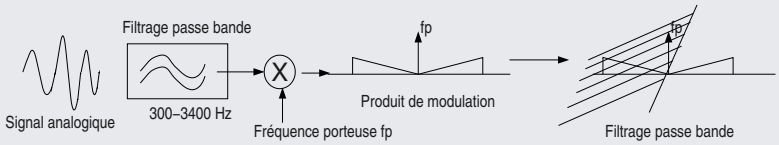
- Le multiplexage en fréquence
- Le multiplexage temporel analogique
  - Par échantillonnage du signal origine
- Le multiplexage temporel numérique
  - Par échantillonnage et numérisation
- Le multiplexage statistique
  - Par acheminement sur canal commun de segments d'informations appartenant à diverses communications

de la téléphonie... aux réseaux

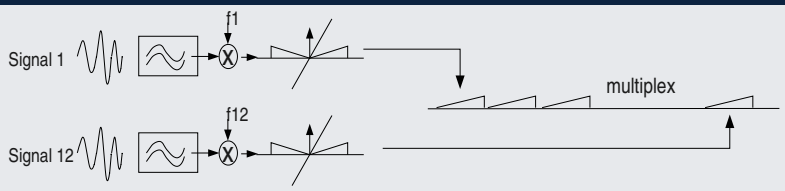
Multiplexage en fréquence 6/67

Exemple du multiplexage de canaux téléphoniques : Modulation en amplitude

Modulation d'un canal



Multiplexage



Cas du mutilpexage de canaux téléphoniques :

— Chaque canal est d'abord filtré dans la bande standard 300-3400 Hz. Le signal filtré vient moduler une porteuse en amplitude. Le résultat de cette modulation produit deux bandes de fréquences contenant l'information utile. Ces deux bandes sont centrées autour de la valeur de la porteuse : de fp-3400 à fp-300 puis de fp+300 à fp+3400.

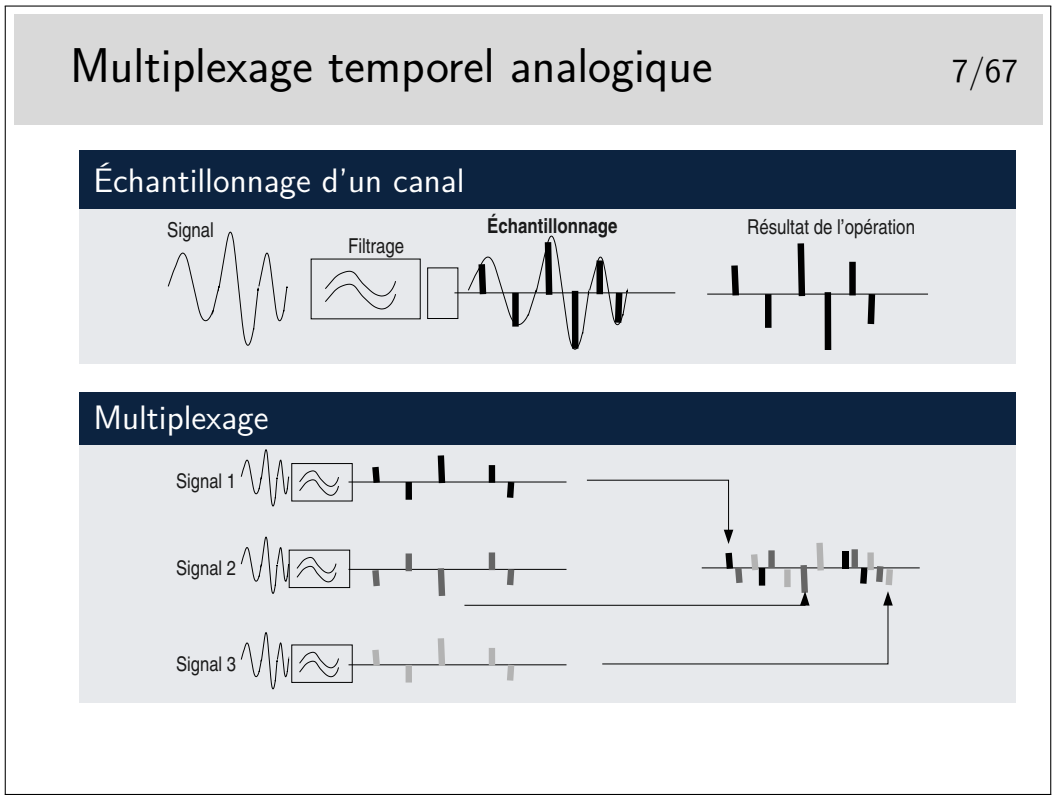
— Ce résultat de l'opération de modulation est filtré pour ne garder qu'une bande inférieure ou supérieure (ici supérieure) de largeur 4000 Hz (de fp à fp+4000).

— Le multiplexage consiste à placer sur le même support le résultat de la modulation et

filtrage de 12 canaux avec 12 fréquences porteuses différentes espacées chacune de 4000 Hz.

— On constitue ainsi ce qu'on appelle un groupe primaire qui à son tour peut être considéré comme un signal analogique et qui peut venir moduler une fréquence porteuse supérieure. Plusieurs groupes primaires peuvent ainsi être multiplexés pour constituer un groupe secondaire et ainsi de suite jusqu'à quatre niveaux.

Le multiplexage sur fibre optique est réalisé de manière similaire. Des communications différentes peuvent être transportées par une même liaison optique, mais sur des longueurs d'onde (couleurs) différentes : c'est le multiplexage en longueur d'onde ou WDM (*Wavelength-Division Multiplexing*).

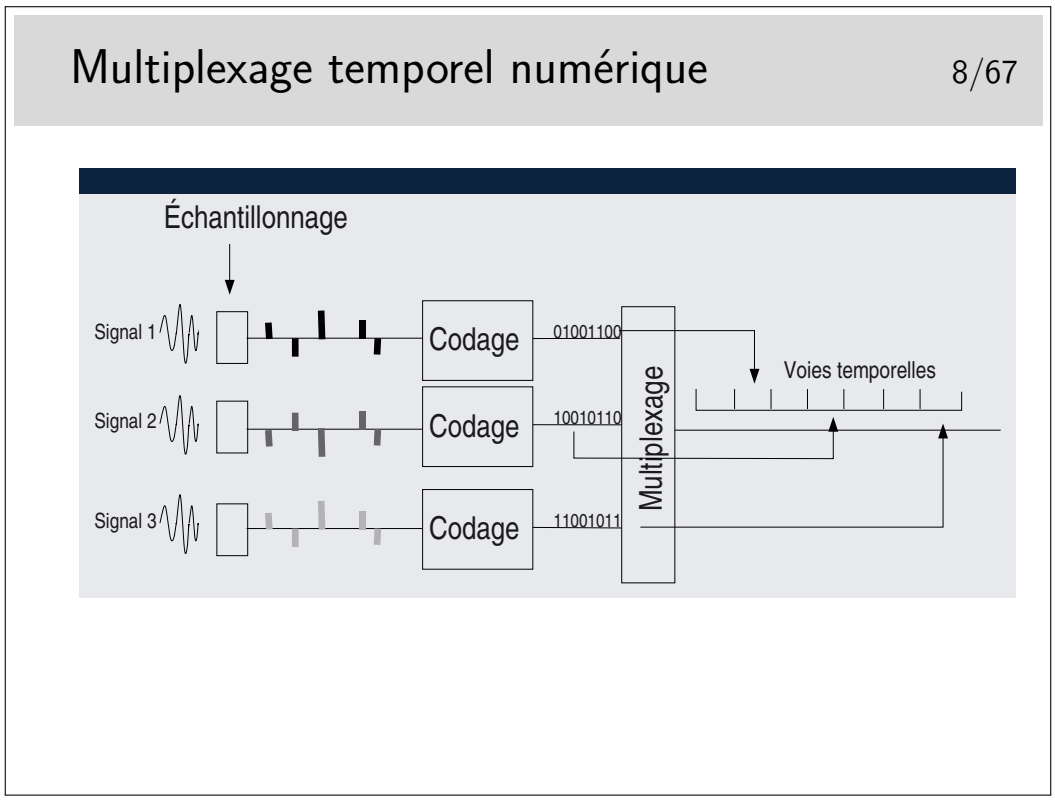


Chaque signal est échantillonné un certain nombre de fois par secondes, chacun avec un décalage dans le temps par rapport aux autres de telle manière qu'ils peuvent être véhiculés sur le même support sans se mélanger après multipexage.

La fréquence d'échantillonnage est telle que le signal doit pouvoir etre reconstitué sans déformations majeures. Shannon a montré que la fréquence d'échantillonnage devait être être le double de la fréquence maximum du signal à échantillonner. Ainsi, en téléphonie, la bande de fréquence est de 300-3400 Hz, on prend 0-4000 Hz par excès, donc la fréquence d'échantillonnage doit être de 8000 Hz.

Le multiplexage par échantillonnage analogique seul ne permet pas de bonnes performances. Le mutiplex ne doit pas être très long, quelques dizaines de centimètres par exemple dans un tout petit autocommutateur téléphonique. Si le support est plus long les échantillons se détériorent à cause des caractéristiques capacitives et selfiques du support, ils sont bruités et déformés et risquent de se mélanger.

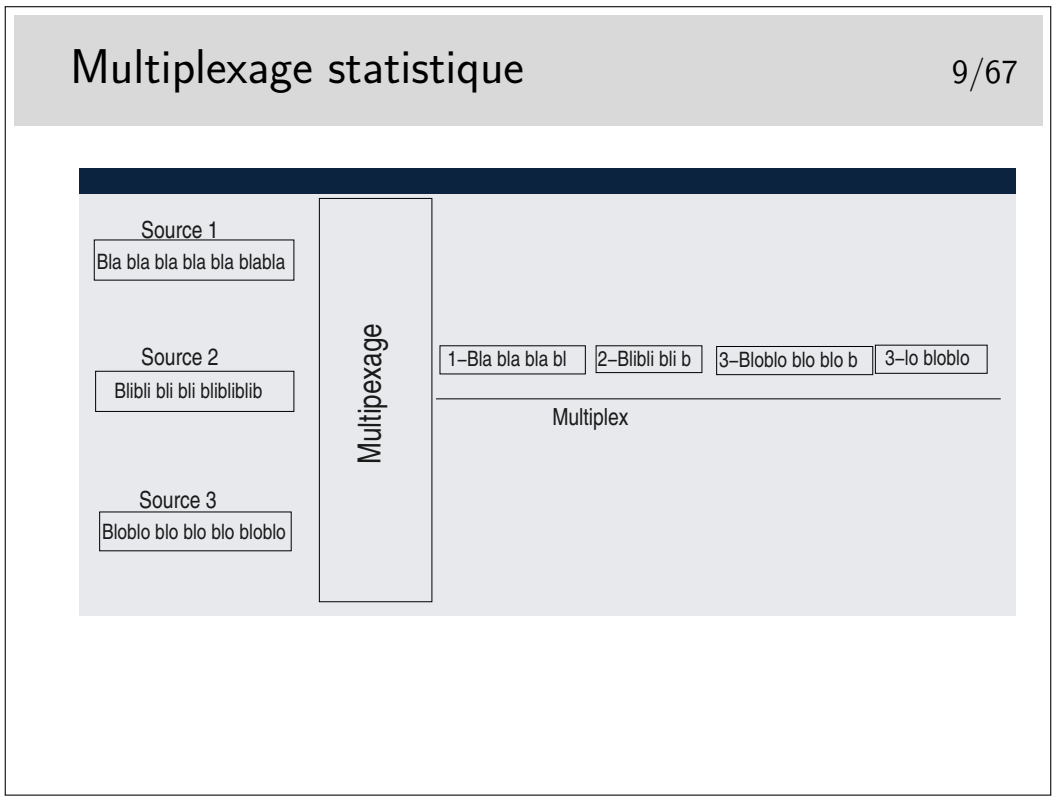
La solution réside dans la numérisation.



Chaque échantillon est transformé en un nombre binaire pouvant être de 12 bits par exemple, ramenés par des techniques de compression à 8 bits.

En téléphonie par il existe deux techniques principales de codage standardisées par le document ITU-T G711, toutes deux respectent la loi de Shannon édictant qu'il faut échantillonner à 8KHz, donc un échantillon toutes les 125μs. Elles divergent sur les points suivants :

- Loi européene dite loi A (Alaw) :
  - méthode de codage spécifique
  - 3,9μs par voie, (8 bits, 488ns par bit)
  - chaque canal a un débit de 64 Kb/s (calculez...)
  - 32 voies par multiplex à 2,048 Mb/s
  - une voie pour la synchronisation du multiplex
  - une voie pour la signalisation des voies téléphoniques
  - 30 voies utiles (ou 31 si la voie de signalisation n'est pas utilisée)
  - multiplex appelé E1 pour le téléphone
- Loi nord américaine et japonaise dite loi μ (μLaw)
  - méthode de codage spécifique
  - 24 voies par mutiplex
  - 24 voies utiles, certains bits pris sur les voies téléphoniques servent à la synchronisation et à la signalisation $\rightarrow$ 56 Kb/s par voie
  - multiplex appelé T1



Les données sont découpées en séquences de *longueur variable* qui sont ensuite injectées sur le même support, les unes à la suite des autres. Il n'y a pas d'ordre à priori. Si une seule source émet elle a toute la bande passante du support à sa disposition. Si n sources émettent simultanément elles disposent chacune de 1/n de la bande passante (si leurs données sont de taille égales).

Les séquences ne doivent pas être trop longues car alors le support est dédié trop longtemps à une seule source.

Les séquences doivent pouvoir être identifiées pour pouvoir les reconnaître à la réception et les remettre à leur bon destinataire. Elles sont munies d'une «étiquette» d'identification qui peut être un numéro de canal ou une adresse de réception.

Le canal est virtuel, il n'existe que si des données sont présentes.

Ce mode de multiplexage est utilisé pour le transfert de données. Il est universellement répandu : ethernet, IP (donc Internet), etc...

Les séquences de données sont appelées «paquets».

Parfois les paquets de données sont de taille constante, on les appelle alors des cellules (réseau de type ATM par exemple).

Avantages et inconvénients des différents types de multiplexage I

- Multiplexage temporel
  Multiplexage en fréquence
  Multiplexage en longueur d'onde
  - ressource (la voie temporelle) réservée pour la durée de la communication, même si celle-ci est silencieuse : mauvaise utilisation de la ressource
  - bande passante et délais de transfert garantis

Avantages et inconvénients des différents types de multiplexage II

- Multiplexage statistique
  - utilisation optimale du canal, si des sources sont silencieuses, le canal peut être utilisé par d'autres
  - bande passante globale partagée entre toutes les sources, pas de garantie de réservation (sauf Frame Relay et ATM au prix de complexité supplémentaire pour la réservation et le contrôle de l'utilisation)
  - pas de délai de transfert garanti
  - gigue (variation des délais) pouvant être importante
  - C'est actuellement le moyen le plus utilisé pour les données
  - Les applications temps réel sont mal adaptées à cette technique car les délais et la bande passante ne sont pas garantis (sauf surdimensionnement)

La paquetisation dans les réseaux de données 12/67

- Multiplexage statistique
  - Pendant qu'une source émet sur un multiplex, les autres sources doivent être silencieuses
  - Pour rendre équitable l'utilisation du multiplex, une source ne peut pas le monopoliser trop longtemps, il faut limiter sa durée d'émission
  - Les données sont segmentées en unités appelées «paquets»
  - Les paquets ont une taille maximale et parfois une taille minimale
  - Les paquets doivent être munis d'une entête, sorte d'étiquette, qui permet de les reconnaître et ainsi de savoir en réception vers quel destinataire acheminer le paquet

1.2 Notion de connexion

Communications avec ou sans connexions 14/67

- Avec connexion, comme le téléphone ?
  - Il faut chercher un chemin dans le réseau entre la source et la destination puis le réserver et l'établir
  - Lorsque la communication est terminée il faut libérer le chemin
  - Ces opération nécessitent des échanges d'informations spécifiques que l'on appelle la signalisation
  - Le chemin est appelé circuit
- Sans connexion, comme à la poste ?
  - On munit les données «d'enveloppes» contenant l'adresse de la destination et le réseau achemine ces données en les routant dans chaque nœud en fonction de cette adresse
  - Les unités de données véhiculées sont appelées des datagrammes

Avantages et inconvénients des modes avec et sans connexion I

- Mode orienté connexion
  - Avantages
    - Le chemin est toujours le même pour la durée de la connexion, les données sont reçues dans l'ordre ou elles ont été émises
    - La signalisation nécessaire à l'établissement de la connexion peut permettre de véhiculer des informations de demande de qualité de service
    - Les délais de traitement dans les nœuds sont généralement courts
  - Inconvénients
    - Il faut un certain délai d'établissement et de rupture de la connexion
    - Le chemin est préétabli et si une maille du réseau devient inutilisable (panne d'un nœud, rupture de la maille) la communication est rompue

Avantages et inconvénients des modes avec et sans connexion II

- Mode sans connexion
  - Avantages
    - Pas de nécessité de signalisation pour établir des chemins
    - Reroutage facilité des données en cas de rupture d'un lien dans le réseau
  - Inconvénients
    - Il n'y a pas de chemin préétabli, deux unités de données successives peuvent arriver dans le désordre si le chemin de la seconde a été plus court que le chemin de la première en cas de modification de parcours entre les deux unités
    - On peut envoyer des données vers des destinations inexistantes

Réseaux avec et sans connexion 17/67

- Orientés connexion
  - X25 : le réseau de données des années 80 (né vers 1976)
  - Frame Relay, très utilisé aujourd'hui pour interconnecter des unités dispersées de mêmes entreprises
  - ATM : dans le cœur de réseau des opérateurs (mais aussi dans votre modem ADSL...)
  - MPLS (MultiProtocol Label Switching) : dans le cœur des réseaux d'opérateurs (au service des entreprises)
- Orientés sans connexion
  - Les réseaux locaux d'entreprises : Ethernet, Token-Ring
  - IP : Internet
  - Réseau Cyclades (l'ancètre de IP, abandonné en 1978 par choix politique... [1])

1. https://www.franceculture.fr/emissions/superfail/pourquoi-la-france-a-t-elle-invente-le-minitel-plutot-quinternet

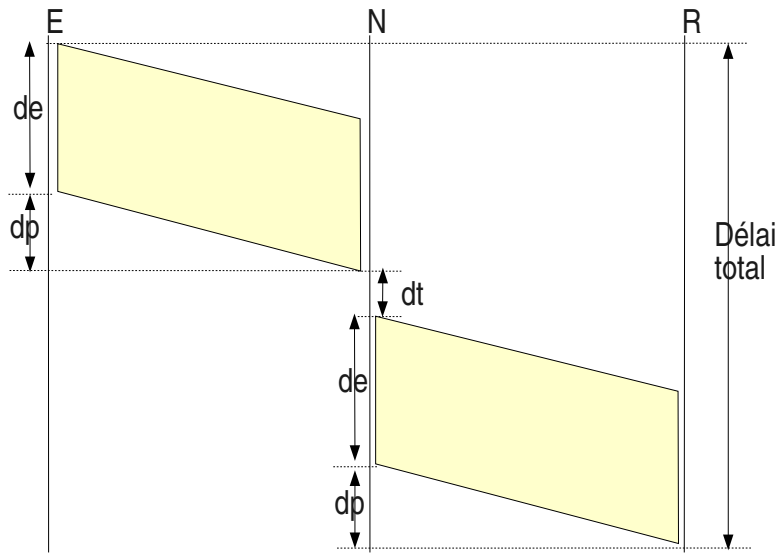
1.3 Délais et QoS

Les délais 19/67

- Entre un émetteur et un récepteur, les données vont subir différents délais
  - Durée d'émission : inversement proportionnelle au débit, proportionnelle à la longueur des segments de donnée
  - Le délai de propagation : au minimum le délai de la lumière dans le vide ($c \approx 300000 Km/s$), dépend par ailleurs du support ($2/3c$ dans le verre et le cuivre).
    Exemple : 250ms pour une liaison par satellite géostationnaire (terre - terre)
  - Le délai de traitement dans les nœuds : délai de traitement réel plus temps passé dans les files d'attente en attente de traitement et en attente de ré-émission

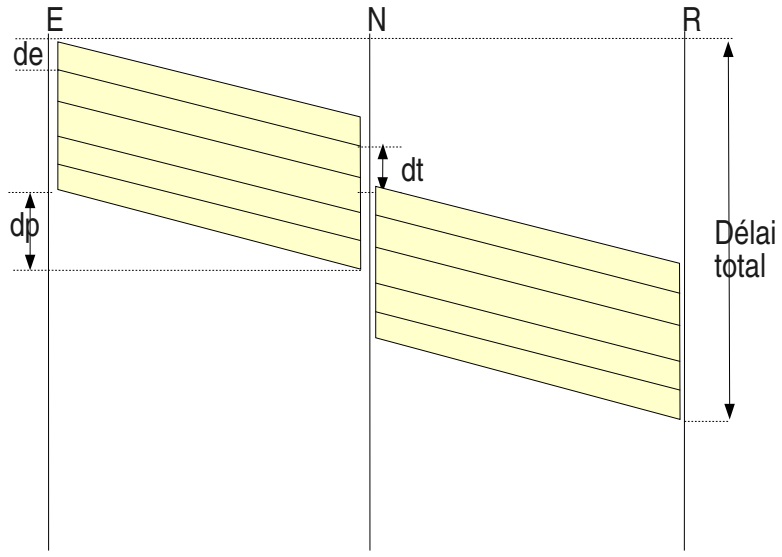
Relation célérité de la lumière dans le vide (c), fréquence (f) et longueur d'onde (λ) : $\lambda f = c$

Les délais 20/67



Délai total = $2de + 2dp + dt$ si un seul nœud intermédiaire et si les débits sont les mêmes sur les tous les liens.

Réduisons les délais avec de petits paquets 21/67



Les interfaces des nœuds fonctionnent en parallèle. Une interface peut recevoir pendant qu'une autre émet. Il reste cependant le temps de traitement, c'est à dire le temps mis par l'unité centrale du nœud pour déterminer vers quelle interface de sortie il faudra acheminer le paquet, plus le temps passé dans les mémoires files d'attente.

Notion de qualité de service liée aux délais 22/67

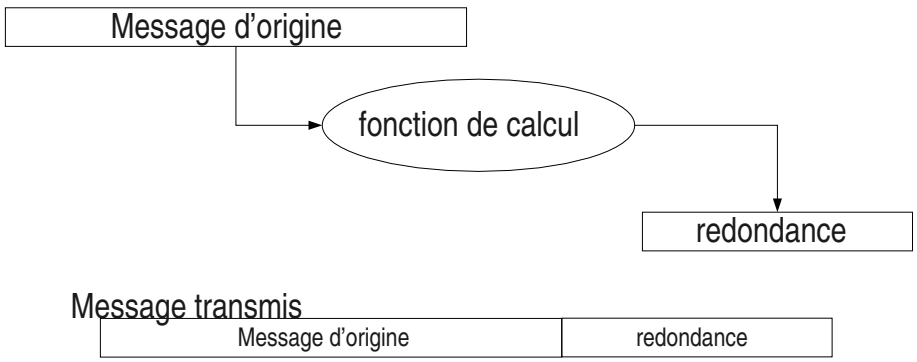
- Pour le téléphone
  - Délai de 50ms : bon confort de communication
  - Au delà de 200ms (communications par satellites geostationnaires) : qualité très moyene, nécessité d'annulation d'écho
  - 400ms : toute dernière extrémité à ne pas dépasser
- Pour les données
  - Dépend fortement du type d'application
- Variation des délais
  - Inhérente aux réseaux de transmission de données
  - Inadapté aux services de type téléphone ou vidéo (mais on tente quand même avec succès, p.ex. : VoIP)

1.4 Détection et correction d'erreur

Détection et correction d'erreur 24/67

- Une transmission binaire ne se fait pas sans erreur : des «1» peuvent devenir des «0» et réciproquement
  - Comment détecter des erreurs ?
  - Comment les corriger ?
- Détection (principe général)
  - transmet l'information par bloc plus une séquence de redondance
  - la séquence de redondance est calculée par une fonction spécifique
  - le même calcul est fait à l'arrivée et le résultat est comparé
- Correction
  - directe si la redondance est suffisante et possède des propriétés de correction
  - par retransmission

Détection d'erreur : par séquence de redondance 25/67



Message transmis

À la réception l'opération est reconduite, le résultat est comparé à ce qui est reçu et le message est accepté ou non.

Exemple : bit de parité, CRC (*Cyclical Redundancy Code*), somme de contrôle (*Checksum*), ...

Dans les cas très répandus où les opérations de détection d'erreur sont simples et où les algorithmes ne permettent pas la correction immédiate, on ne peut pas dire si les erreurs de transmission portent sur le message lui même ou la redondance, ou les deux. On jette tout simplement le message (on l'ignore).

On ne se pose pas, à ce stade, le problème de la correction. Ce n'est pas l'affaire de l'algorithme, on s'en remet pour cela à des mécanismes situés dans des couches protocolaires supérieures (si on en a besoin).

Détection et correction d'erreur 26/67

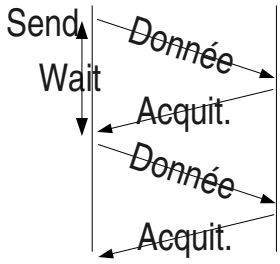
- Si la redondance est suffisante et l'algorithme sufisamment puissant il est possible de détecter les erreurs et de les corriger.
- Exemple : codes de Reed-Solomon, Turbo-codes, ...

— Reed-Solomon : `http://en.wikipedia.org/wiki/Reed%E2%80%93Solomon_error_correction`

— Turbo-codes : `http://en.wikipedia.org/wiki/Turbo_codes`

Correction d'erreur par retransmission I 27/67

- Un récepteur détecte une erreur, il jette le segment de données erroné
- Comment l'émetteur peut-il détecter qu'il y a eu une erreur ?
  - Impossible sans l'utilisation d'un mécanisme d'acquittement

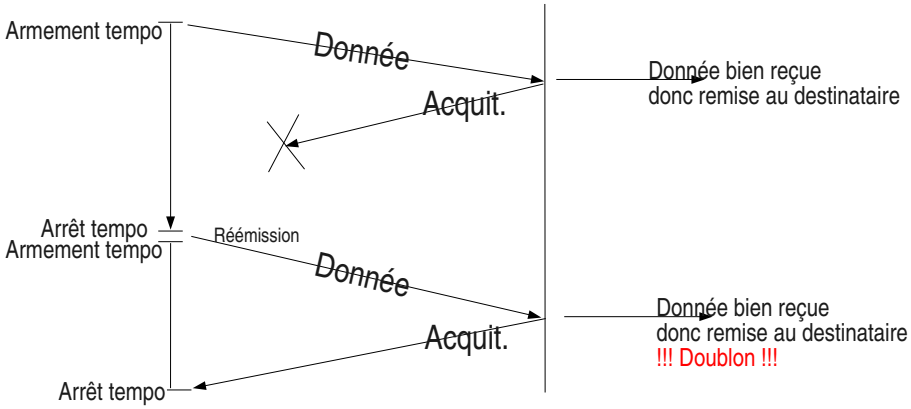


Ce mécanisme est appelé «Send & wait»

Que se passe-t'il si la donnée se perd ?
Si l'acquittement se perd ?

Correction d'erreur par retransmission II 28/67

- Utilisation de temporisateur (timer)

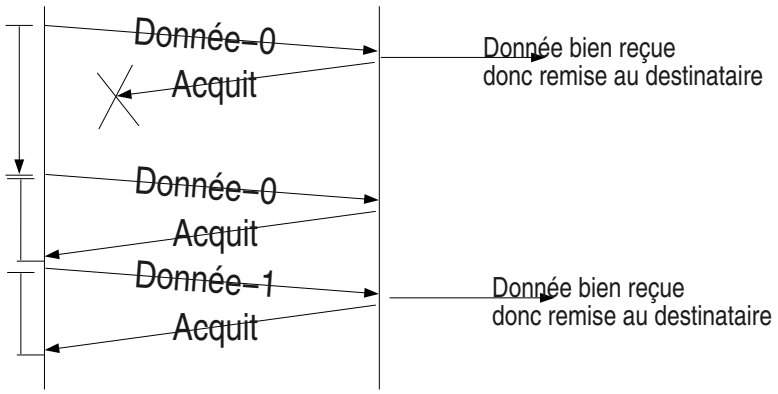


Où comment se créer d'autres problèmes en voulant en régler un... Et maintenant comment on fait pour éviter les doublons ?

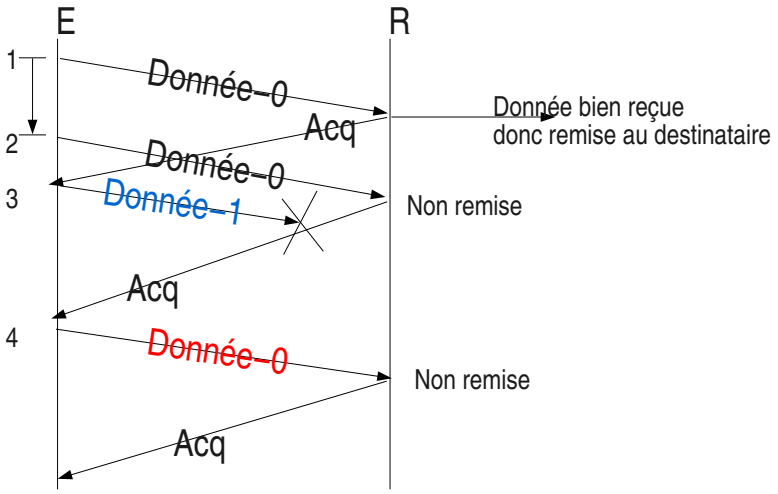
C'est une règle, hélas, très générale, en Réseaux, que de se créer de nouveaux problèmes en apportant des solutions à d'autres...

Retransmission : le problème des doublons 29/67

- On peut éviter les doublons en numérotant les données
  - Exemple : les données sont munies d'un numéro 0 et 1 alternativement



Timer trop court et malchance... 30/67

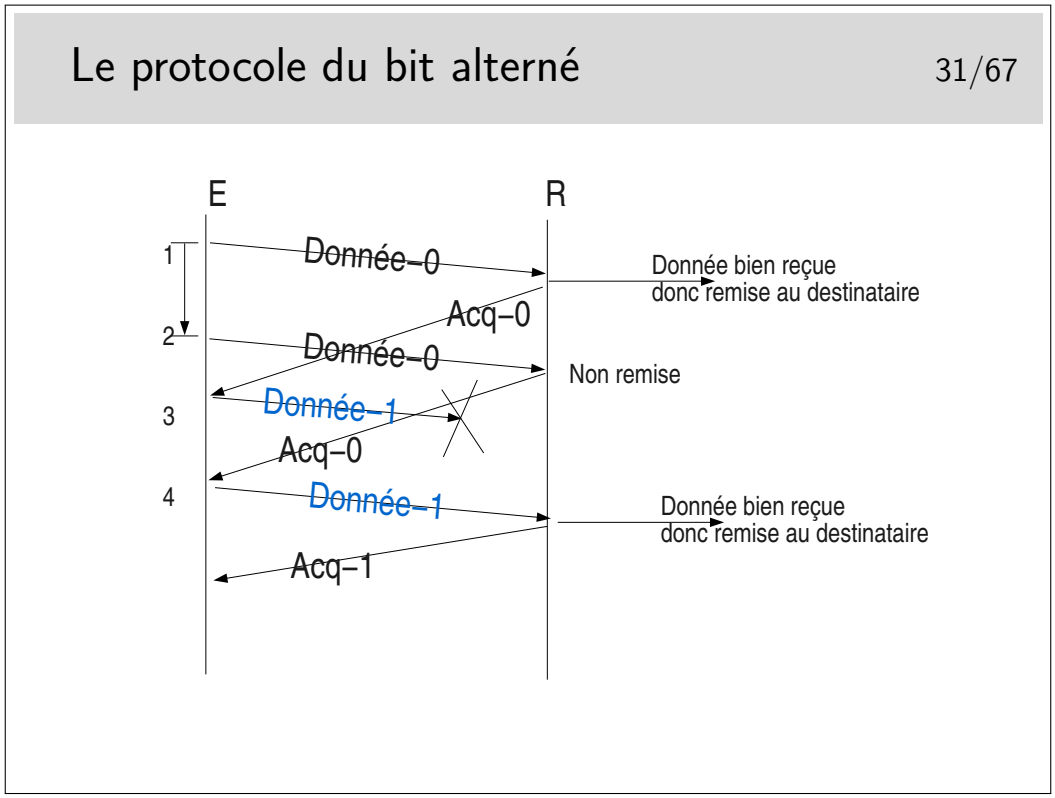


1. émission de `Donnée-0`, bien reçue, elle est remise à l'entité destinatrice
2. le timer expire trop tôt, `Donnée-0` est réémise et est bien reçue, elle n'est pas remise car elle porte le même numéro que la donnée précédente. Cependant, le récepteur renvoie un acquittement car il pense que s'il a reçu à nouveau `Donnée-0` c'est que son acquittement précédent s'est perdu..
3. un acquittement arrive, l'émetteur peut alors émettre une nouvelle donnée : `Donnée-1`
4. un acquittement arrive, l'émetteur pense qu'il s'agit de l'acquittement de `Donnée-1`.

À chaque fois qu'il reçoit un acquittement il pense que celui-ci concerne les données précédemment émises, il vide les tampons mémoires qui les contenaient.

Ainsi, en 4, s'il y a une nouvelle donnée à envoyer ce sera `Donnée-0` (pas le même contenu qu'au début du scénario), or le récepteur voyant arriver à nouveau un numéro 0 le rejettera. Le numéro est censé représenter les données, on ne compare pas celle qu'on reçoit avec les précédentes. D'ailleurs celles-ci ont été livrée à l'entité de destination, elles ne sont pas gardées en mémoire.

Dans ce scénario le segment `Donnée-1` n'est pas reçu mais il est considéré par l'émetteur comme bien reçu. Le segment `Donnée-0` suivant est bien émis et bien reçu mais il n'est pas remis à son destinataire final... Tout va mal dans ce scénario !



Les acquittements portent les numéro des données qu'ils acquittent. Ainsi, en 4, on reçoit à nouveau `Acq-0`, alors qu'on attendait `Acq-1`, il y a un problème, le contenu de `Donnée-1` précédent est toujours en mémoire (on ne le vidra que sur réception de `Acq-1`), on peut donc réémettre `Donnée-1` (le même message que précédemment).

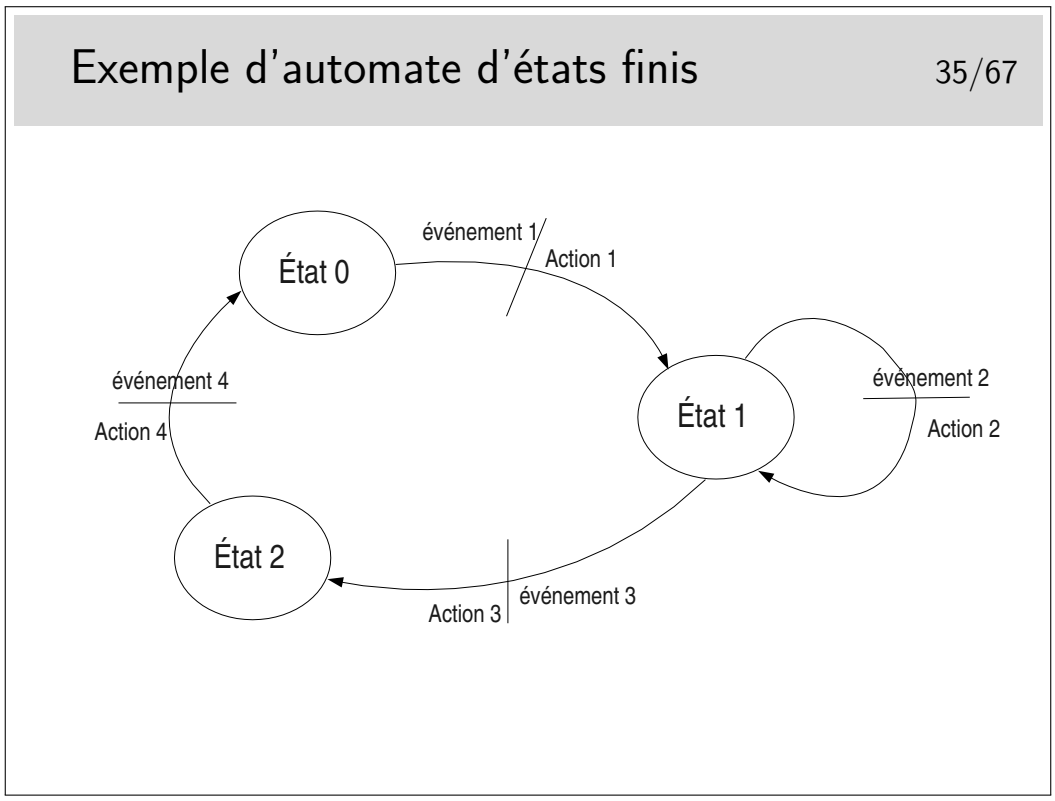
1.5 Des protocoles

Notion de protocole I 33/67

- Dans ce qui précède nous avons développé des moyens de transmettre des données sans erreurs en rajoutant de l'information et des messages spécifiques
  - À chaque ajout de mécanisme on corrige un problème, on en rajout un autre
  - Il faut que la série correction/nouveau problème converge

Notion de protocole II 34/67

- En rajoutant un protocole au dessus d'une voie de communication on modifie les propriétés de cette voie de telle manière que l'on obtient une nouvelle voie de comunication
- Pour mettre en œuvre et gérer les mécanimes il faut construire un automate logiciel comportant des états et des transitions d'états sur événements



Au départ l'automate est dans un certain état, disons l'état 0 par exemple. Seul l'arrivée de l'événement 1 peut faire que l'automate passe dans l'état 2. Au passage dans ce nouvel état l'action 1 sera effectuée. Un événement ne fait pas toujours changer d'état, on l'illustre ici par l'état 1, dans lequel on reste après que soit survenu l'événement 2 et que l'action 2 ait été effectuée.

Cette représentation est proche de la formalisation mathématique des automates par les «réseaux de Petri».

La représentation schématique est intéressante car elle est souvent plus facile à comprendre (lorsqu'il n'y a pas trop d'états ni d'événements). Cependant elle ne permet pas de s'assurer que tous les cas possibles de relation événement/transition soient envisagés. Il faut alors recourir à la représentation de l'automate sous forme de tableau.

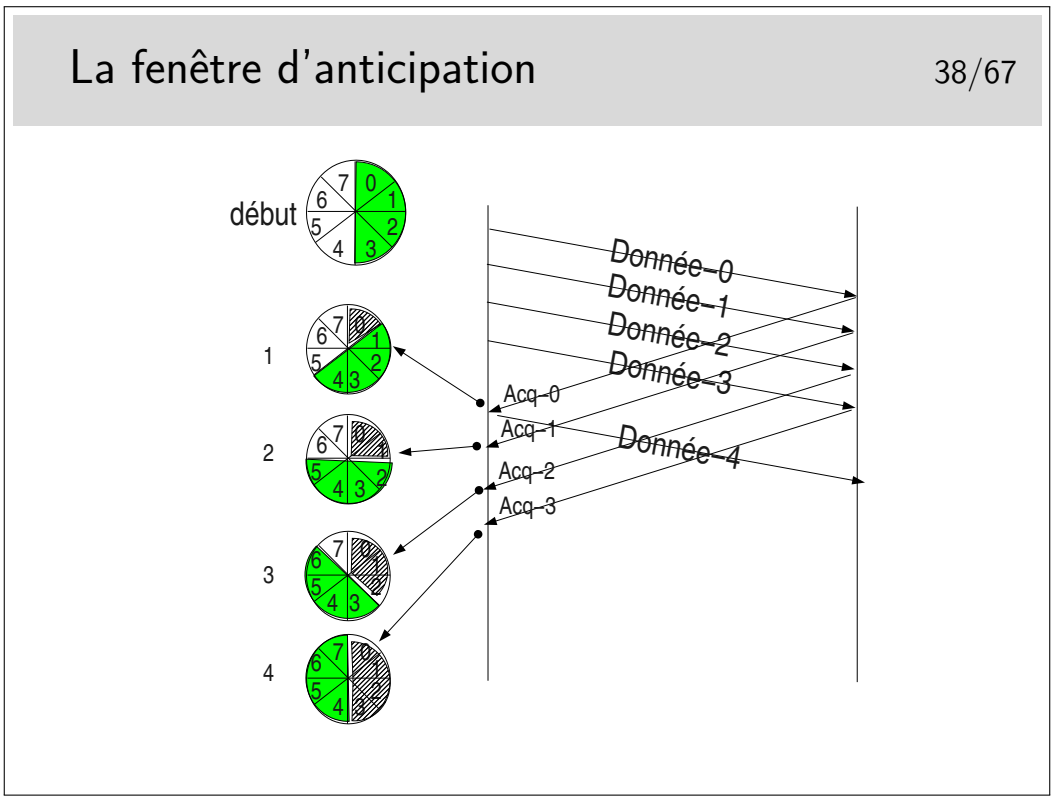
L'automate du protocole du bit alterné 36/67

Événement État	Demande d'envoi de donnée	Arrivée Ack 0	Arrivée Ack 1	Expiration Timer
État 0 nouvel état	Envoyer Data-0 Armer Timer Attente Ack 0			
Attente Ack 0 nouvel état		Arrêt timer État 1		Envoyer Data-0 Armer Timer Même état
État 1 nouvel état	Envoyer Data-1 Armer Timer Attente Ack 1			
Attente Ack 1 nouvel état			Arrêt timer Etat 0	Envoyer Data-1 Armer Timer Même état

Les protocoles à anticipation 37/67

- le mécanisme du *Send & Wait* conduit à une mauvaise utilisation de la bande passante, quand on attend on n'utilise pas la bande passante alors disponible
- Il est judicieux d'anticiper l'émission sans attendre les acquittements
  - Dans une certaine mesure...
  - Une limite, appelée fenêtre d'anticipation, permet de ne pas trop anticiper et bloque l'émission si les acquittements n'arrivent pas
  - Les acquittements existent toujours
  - La fenêtre «tourne» ou «glisse» lorsqu'un acquittement arrive

On dit que la fenêtre est «tournante» ou «glissante» selon la représentation qu'on en fait. Voir le transparent suivant.



Paramètres : les segments de données sont numérotés modulo 8. La fenêtre d'anticipation est de 4.

On ne peut pas émettre plus de 4 segments à la suite si on ne reçoit pas pendant ce temps un acquittement. Lorsqu'un acquittement arrive, la fenêtre «tourne» d'un pas, le tampon mémoire contenant la donnée correspondant à l'acquittement est libéré, la donnée est considérée comme bien envoyée, on l'efface. En pratique on ne l'efface pas, on considère son emplacement mémoire comme libre. Sur le schéma ci-dessus, on indique cette «libération mémoire» par une zone grisée dans la fenêtre.

Départ : la fenêtre englobe les numéros 0 à 4. On commence l'envoi. L'acquittement pour la donnée 0 arrive en 1. La fenêtre tourne et englobe 1 à 4. Le segment 4 devient éligible à l'émission. Il est émis si une donnée est à émettre. Le tampon mémoire contenant Donnée-0 est effacé. En 2 l'acquittement pour Donnée-1 arrive, la fenêtre tourne et englobe maintenant 2 à 5. Etc.

Le contrôle de flux 39/67

- Mécanisme permettant à un récepteur d'asservir la capacité à émettre de son correspondant en fonction de ses capacités de traitement
  - Permet d'informer l'émetteur qu'il doit réduire son débit
  - Permet de ne pas inonder les tampons mémoire du récepteur

Un émetteur et un récepteurs ne fonctionnent pas obligatoirement à la même vitesse (au même débit). Les tampons mémoire de réception se vident lorsque les applications destinatrices viennent y puiser les données reçues. Si la machine de réception est lente, si l'application réceptrice prend trop de temps à traiter les données et ne vient pas les retirer suffisamment rapidement des tampons de réception, ceux-ci se remplissent dangereusement. Lorsque les tampons sont pleins les données à recevoir seront perdues. Le contrôle de flux permet d'éviter ces pertes en évitant que les données ne soient envoyées.

Le contrôle de flux pourra être couplé à des mécanismes d'acquittement, ce sont les trames RR et RNR du protocole HDLC-LAPB que nous verrons plus loin : RR pour *Receiver Ready* ou encore «tout va bien, envoyez !», RNR pour *Receiver Not Ready* ou encore «OK, j'ai bien reçu vos données mais arrêtez vous quelque temps».

2 Modélisation et standardisation

2.1 Standardisation

Les organismes de standardisation 42/67

- Les organismes officiels nationaux et internationaux
  - OSI : Organisation de Standardisation Internationale (ISO en anglais), et ses branches nationales : AFNOR en France, DIN en Allemagne, ANSI aux USA
  - UIT-T : Union Internationale des Télécommunications, secteur des Télécommunications (il y a aussi le secteur Radio)
  - ETSI : European Telecommunications Standards Institute
- Les organismes de l'industrie et de la recherche
  - IEEE : Institut of Electrical and Electronics Engineers
- L'Internet
  - IETF : Internet Engineering Task Force, étudie et développe les protocoles et services au dessus du protocole IP

2.2 Principe de la modélisation

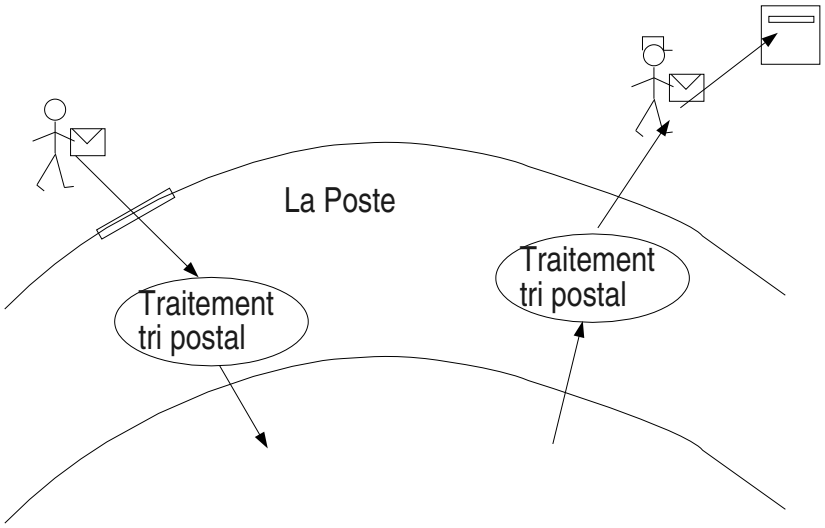
La modélisation de la problématique réseau 44/67

- Comment «voir» le réseau ?
  - Comme un utilisateur : «*Je me connecte, je ne sais pas comment ça marche, je ne veux pas savoir comment ça marche, mais ça marche ! Et j'utilise.*»
  - Comme un développeur d'application communicantes : «*Par quel moyen programmatique mes applications peuvent-t'elles communiquer ? Dois-je savoir comment fonctionne le réseau ? Tout le réseau ? Une partie du réseau ?*»
  - Comme le gestionnaire du réseau : «*Dois envisager le réseau dans son ensemble, du câble aux applications ?*»

Exemple d'un réseau existant : La poste 45/67

- En tant qu'utilisateur :
  - *Je dois savoir où est situé le bureau de poste, l'adresse de mon correspondant et avoir une boite aux lettres. Je dois pouvoir interagir avec le service offert par La Poste. La poste doit me fournir un service et des moyens d'accès à ce service.*
  - *Je n'ai pas à savoir comment fonctionne le service de La Poste de l'intérieur*
- En tant que postier :
  - *Je dois savoir traiter les lettres, les orienter vers les bons sacs postaux, placer les sacs postaux dans les bons camions, voitures trains, avions...*
  - *Je n'ai pas à savoir comment fonctionne le service de transport qui achemine physiquement les sacs postaux.*

Services et interfaces d'utilisation 46/67



La Poste offre un *SERVICE*

On accède à ce service par des moyens appelés, en terme «réseaux», des *PRIMITIVES DE SERVICE*.

On interagit avec le service via les primitives de service dont le paramètre principal est une sorte d'adresse : l'adresse où est situé le bâtiment de La Poste pour aller «poster» sa lettre, l'adresse du destinataire de la lettre pour que le postier sache dans quelle boite aux lettres déposer celle-ci. Cette sorte d'adresse est appelée en termes réseaux le *POINT D'ACCÈS AU SERVICE* (le Service Access Point ou SAP).

Notion de couche, d'interface entre couches et d'indépendance entre couche.

Question : le travail d'acheminement de la lettre est-il terminé lorsque le postier distributeur (le facteur) a déposé la lettre dans la boite aux lettres du destinataire ?

Réponse : oui et non !!!

— Oui pour le service postal.

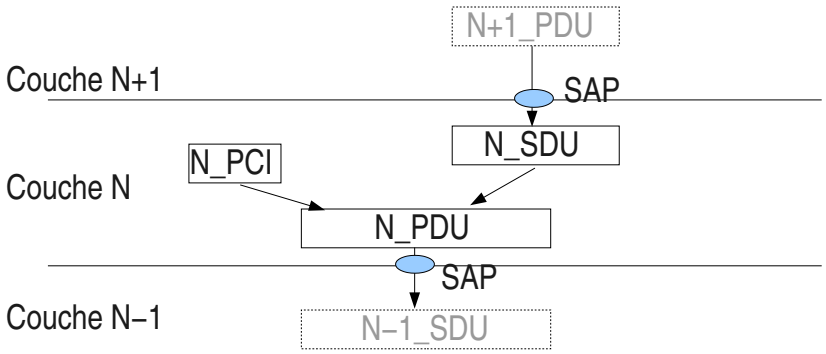
— Non pour le destinataire final. Il peut s'agir de la boite aux lettres de la famille Dupont. La lettre peut être pour Philippe Dupont et non pour Jacques Dupont. Le SAP «adresse des Dupont» n'est pas le seul SAP à considérer. Le SAP «Jacques» et le SAP «Phillipe» sont aussi à prendre en compte, mais pas au niveau du service postal (on dira plus tard : «pas dans la même couche»).

2.3 Le modèle ISO (OSI)

Le modèle ISO (OSI) 48/67

- Interconnexion de Services Ouverts (OSI en anglais)
- Définit la notion de service et de couche de service ainsi que les relations entre les entités distantes d'une même couche (les protocoles de communication)
- Définit aussi les relations entre couches (les primitives de service et les SAP)
- Définit les différentes couches, leur rôle ainsi que leurs protocoles

Le modèle OSI : Couches et unités de données 49/67



L'unité de donnée fournie par l'utilisateur est une unité de donnée à SERVIR. C'est une UNITE de DONNEE de SERVICE : une *SERVICE DATA UNIT* (SDU) en anglais.

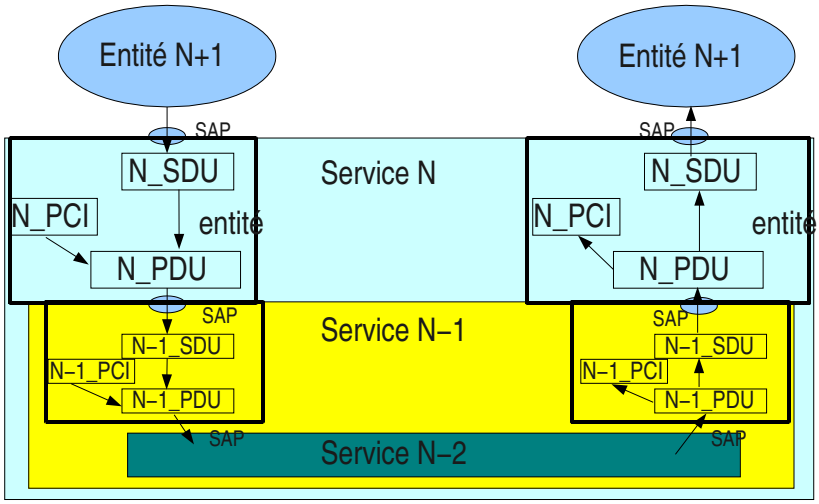
La couche assurant le service utilise un certain mécanisme qui lui est propre, un *protocole* particulier, nécessitant un échange de données spécifiques. Ces données protocolaires n'ont rien à voir avec les données utiles, elles servent à la gestion du transfert de celles-ci.

Les données protocolaires (Protocol Control Information) sont ajoutées au segment de données utiles (les données de service, la SDU) pour former une nouvelle unité : l'unité de données de protocole ou *PROTOCOL DATA UNIT* (PDU).

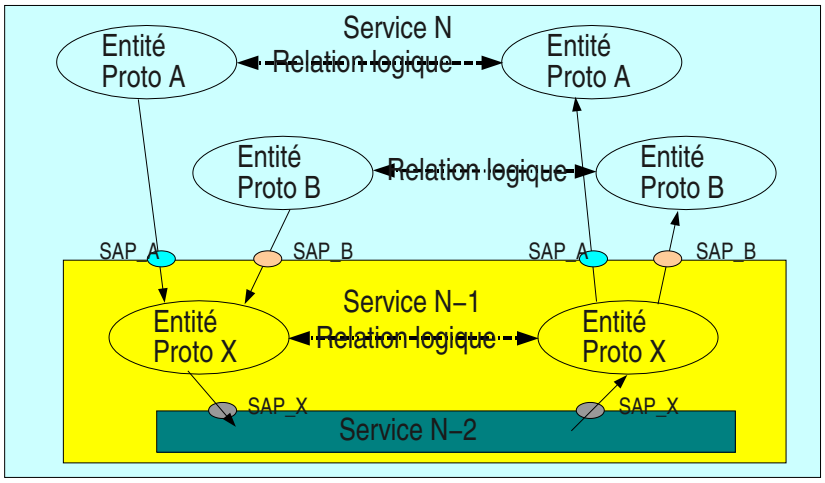
Notion d'entité protocolaire 50/67

- Une entité protocolaire est un «programme informatique», une application ou un module opérationnel du système d'exploitation mettant en œuvre un protocole
- Une même couche peut être composée de plusieurs entités mettant en œuvre des protocoles différents pour assurer un même service
  - Exemples :
    - Le téléchargement de fichier peut être réalisé via le protocole ftp ou le protocole du web http
    - Des machines peuvent partager des fichiers via les protocoles NFS (monde Unix), SMB (monde Windows) ou AppleTalk (monde MacIntosh)

Les services et les couches 51/67



Relations entre entités de même niveau et niveaux différents 52/67



La figure ci-dessus est un exemple de ce qui est possible. Un service N met en œuvre deux types de protocoles différents pour assurer un service. Les entités protocolaires A peuvent communiquer entre elles. De même pour les entités protocolaires B. Les entités A ne peuvent communiquer avec les entités B car elles ne «parlent pas» le même langage (le même protocole).

Il est possible que les entités A et les entités B utilisent le même protocole X sous jacent (de couche $N-1$). Lorsque l'entité A de gauche émettra un PDU vers l'entité A de droite, il faudra que ce PDU soit muni de l'identité du SAP entre l'entité A (niveau N) et l'entité X de niveau $N-1$ (donc SAP A). Sinon l'entité X de droite ne saurait pas vers quelle entité réceptrice A ou B envoyer le PDU. De même pour les entités B.

Les 7 couches du modèle OSI
La couche 1 (*Physical Layer*)

1 - La couche physique

- Définit les moyens pour transformer les bits constituant les données en information analogique transportable sur les liens physiques entre la machine et son nœud de raccordement au réseau
- Définit les caractéristiques matérielles et électriques (ou optiques) des supports physique

Les 7 couches du modèle OSI
La couche 2 (*Data Link layer*)

2 - La couche liaison

- Définit les moyens d'acheminer des données structurées au dessus du niveau physique, entre la machine et son nœud de raccordement. La structure de base est la trame
- Définit les moyens de contrôler la fiabilité de la trame en réception
- Peut définir des mécanismes de contrôle de flux et de récupération d'erreurs

Niveau physique et niveau liaison : on ne traverse pas le réseau.

Les 7 couches du modèle OSI
La couche 3 (*Network Layer*)

55/67

3 - La couche Réseau

- Définit les moyens pour acheminer l'information entre machines d'extrémités en traversant les nœuds du réseau
- Implique une nécessité d'identification des machines terminales : un adressage
- Implique de mettre en œuvre des mécanismes de routage dans les nœuds
- Les unités de données de ce niveau sont appelées des paquets

Rappel de l'épisode précédent (niveau physique et niveau liaison) : on ne traverse pas le réseau !

Épisode présent (niveau 3) : on traverse le Réseau !

Les 7 couches du modèle OSI
La couche 4 (*Transport Layer*)

56/67

4 - La couche transport

- Dernière des couches basses
- Interface entre les applications et la couche réseau
- Fournit une abstraction du réseau
  - Corrige les imperfections de la couche réseau
    Dernière chance pour que les applications soient assurées du bon transfert de leurs données
  - Le modèle OSI fournit 5 classes de fonctionnalités différentes pour corriger les imperfections du réseau. De la classe 0 pour le très bon réseau à la classe 4 pour le mauvais réseau.

Tout ceci est très théorique. En pratique, il faut considérer ce niveau transport par type d'architecture : IP, Novell IPX, AppleTalk, IBM SNA, Decnet de Digital (racheté par Compaq, racheté par HP).

Les 7 couches du modèle OSI
Les couche 5 (*Session Layer*)

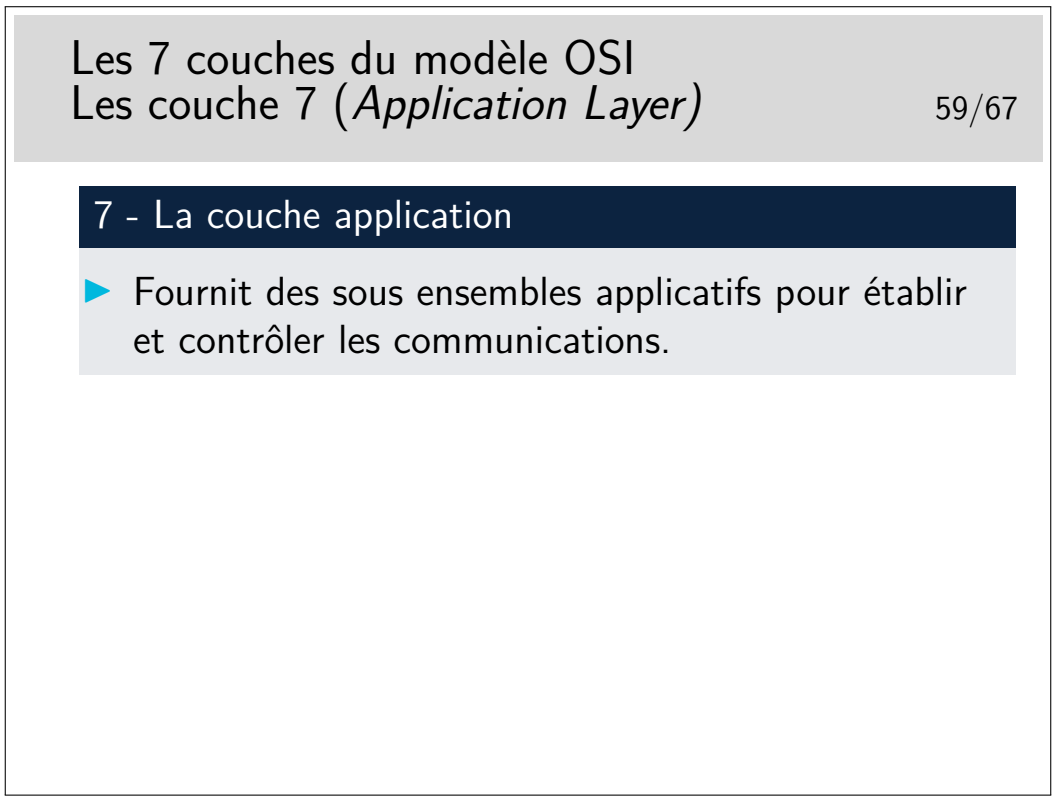
5 - La couche session

- Une communication entre deux applications est considérée comme une session
- La session est organisée, contrôlée
  - Il est prévu des points de synchronisation du dialogue

Les 7 couches du modèle OSI
Les couche 6 (*Presentation Layer)*

6 - La couche Présentation

- Fournit les moyens de décrire les types d'information échangés entre les application : langage de description de types : ASN.1
- Fournit les moyens de coder ces types de données dans un format indépendant de celui des machines (problème de la représentation *big* ou *little endian*)



Tout ceci est à nouveau très théorique. Il faut le replacer dans le contexte de chaque architecture (IP, IPX, etc.)

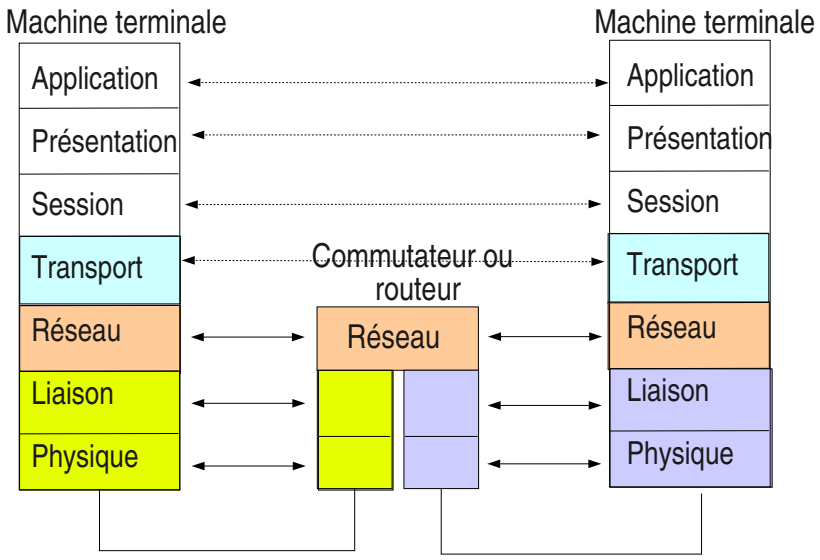
La couche application, qu'on «résume», ici, d'une seule phrase est la plus complexe de toutes dans la réalité des recommandations ISO et ITU-T. Même si ses mérites sont grands, sa complexité et le peu d'outils de développements (API) ont fait qu'il n'existe pas beaucoup de réalisations pratiques (courrier électronique X400, annuaire X500 et quelques autres). Ces applications ont vu le jour vers la fin des années 80, à une époque où l'Internet se répandait déjà beaucoup dans les réseaux du monde de l'enseignement et de la recherche. Le début des années 90 a été décisif, après ne courte période d'incertitude, l'ouverture des standards autour de IP, face à au monde OSI plus fermé à fait pencher la balance en faveur de IP.

Aujourd'hui, il reste cependant le modèle, incontournable, tout au moins pour les couches bases. Il nous aide à placer les concepts et les fonctionnalités des différents réseaux qui existent.

Les 7 couches du modèle OSI : pour résumer... 60/67

- Couches physique (1) et liaison (2) : de la machine au réseau
- Couche réseau (3) : de la machine à la machine en traversant le réseau
- Couche transport (4) : une abstraction du réseau (couches basses) pour l'applicatif
- Couches hautes (5 6 7) : des services, géré au niveau applicatif ou middleware

Où sont implémentées les couches ? 61/67



Toutes les couches sont implémentées dans les machines terminales.

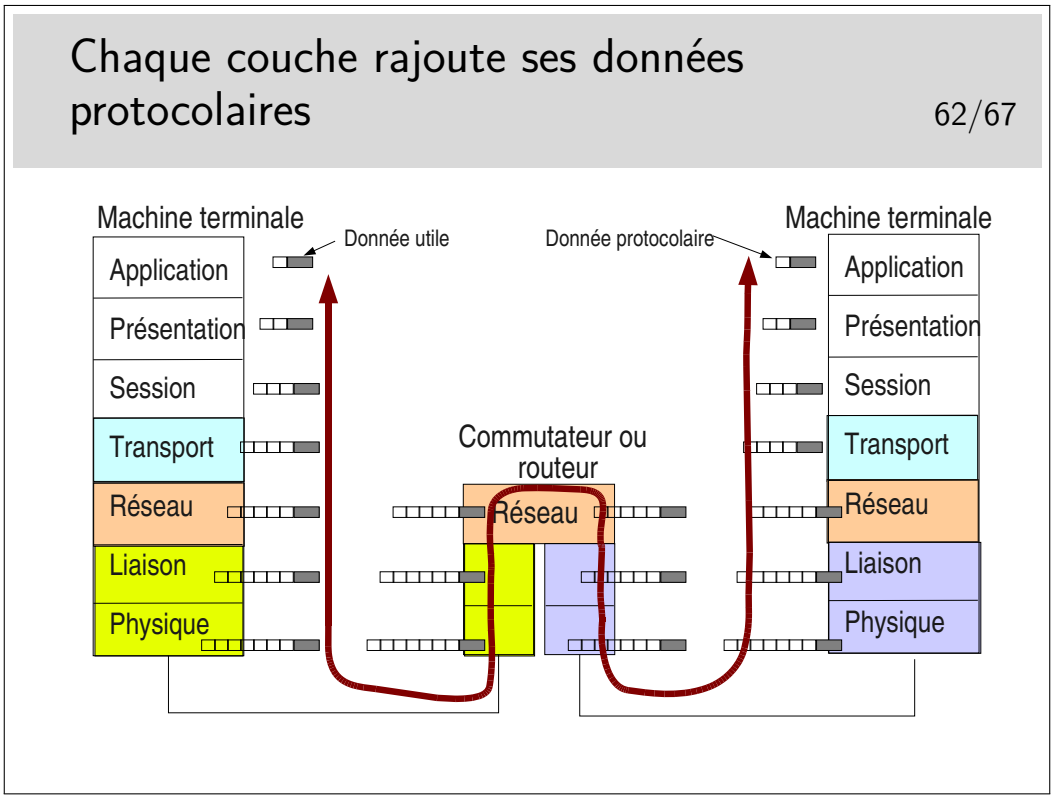
Seules les couches 1 à 3 sont nécessaires dans les machines du cœur de réseau.

Il est très important de noter l'indépendance entre les couches. Voyez ici les couches physique et liaison concernant le lien coté gauche et le lien coté droit. Ces niveaux peuvent être totalement différents du point de vue physique mais aussi du point de vue couche liaison. On pourrait avoir, coté gauche un lien Ethernet sur paire torsadée et de l'autre un lien sur fibre optique.

Les débits peuvent aussi être différents. On pourrait avoir 1GB/s coté gauche et 155Mb/s coté droit.

Les couches Physique et liaison sont en général implémentées sur un même support matériel,

une carte d'interface du type de celle que vous pouvez avoir sur votre ordinateur personnel



Notion de rendement : rapport entre le volume des données utiles et le volume total de données véhiculées sur un niveau.

Par exemple, pour une application cliente telnet, sur TCP IP sur Ethernet (nous verrons ces protocoles plus loin dans le cours) il est courant d'envoyer les données octet par octet dans le sens client vers serveur (au rythme où sont frappées les touches du clavier). Pour chaque octet envoyé, telnet ne rajoute rien mais TCP (niveau 4) rajoute au moins 20 octets (plus 12 avec les options des implémentations TCP d'aujourd'hui), IP (niveau 3) rajoute 20 octets. On a donc au moins $1 + 20 + 20 = 41$ octets et au plus $1 + 20 + 12 + 20 = 53$ octets. Dans le cas 41 octets, Ethernet rajoute 5 octets de bourrage pour arriver à la taille minimale requise, donc 46 octets. Par ailleurs Ethernet rajoute 18 octets d'informations protocolaires donc $46 + 18 = 64$ octets dans un cas et $53 + 18 = 71$ octets dans l'autre.

Le rendement est de 1/64 dans un cas et 1/71 dans l'autre. Ce n'est pas très efficace mais c'est ainsi !

3 Modélisation et standardisation de IP

L'organisation de l'Internet 64/67

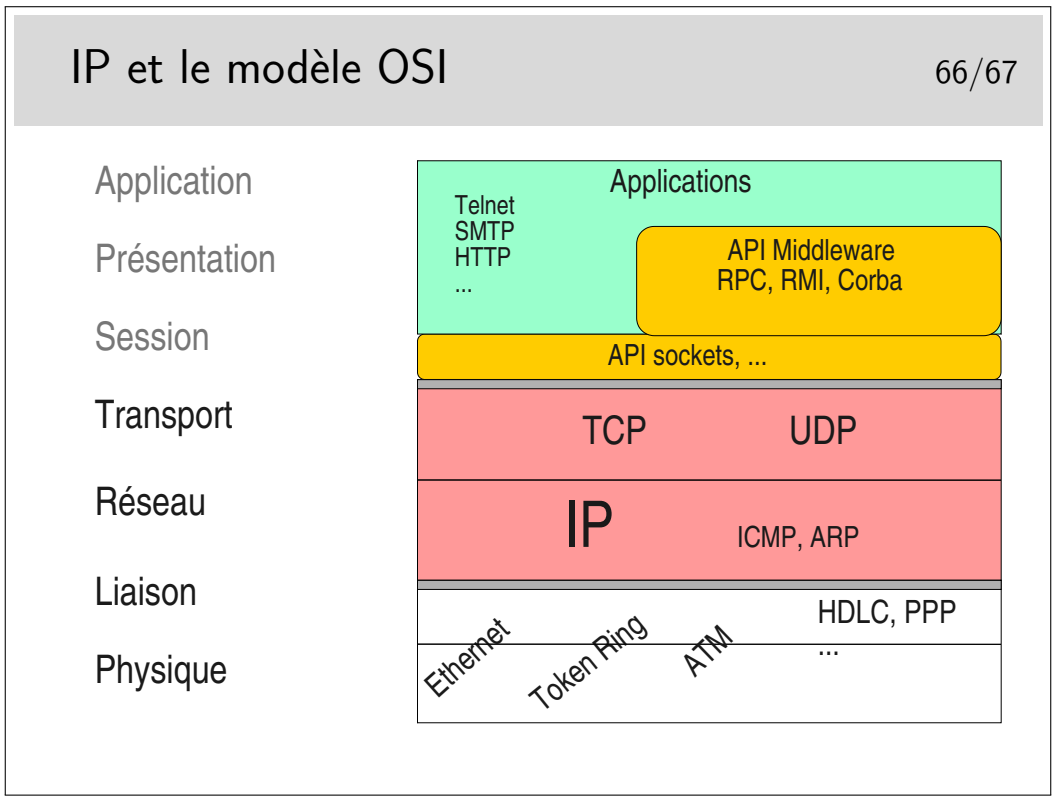
- Organisme de tutelle : l'ISOC (Internet SOCiety), qui regroupe
  - Internet Architecture Board (IAB)
    - Responsable final des travaux de l'IETF
  - Internet Engineering Steering Group (IESG)
    - Contrôle les travaux de l'IETF
  - Internet Engineering Task Force (IETF)
    - L'organisme de standardisation
  - Internet Research Task Force (IRTF)
    - Responsable de la recherche à long terme
  - Internet Assigned Numbers Authority (IANA)
    - La gestion des adresses, des numéros de protocoles, du DNS

Il existe aussi l'ICANN (The Internet Corporation for Assigned Names and Numbers) société de droit privé destinée à remplacer IANA. L'ICANN a été fondée en 1998 et on a parfois du mal à faire la distinction entre ses responsabilité et celles de IANA qui continue d'exister.

Voir un article de synthèse sur ce lien : `http://www.renater.fr/Projets/ICANN/`

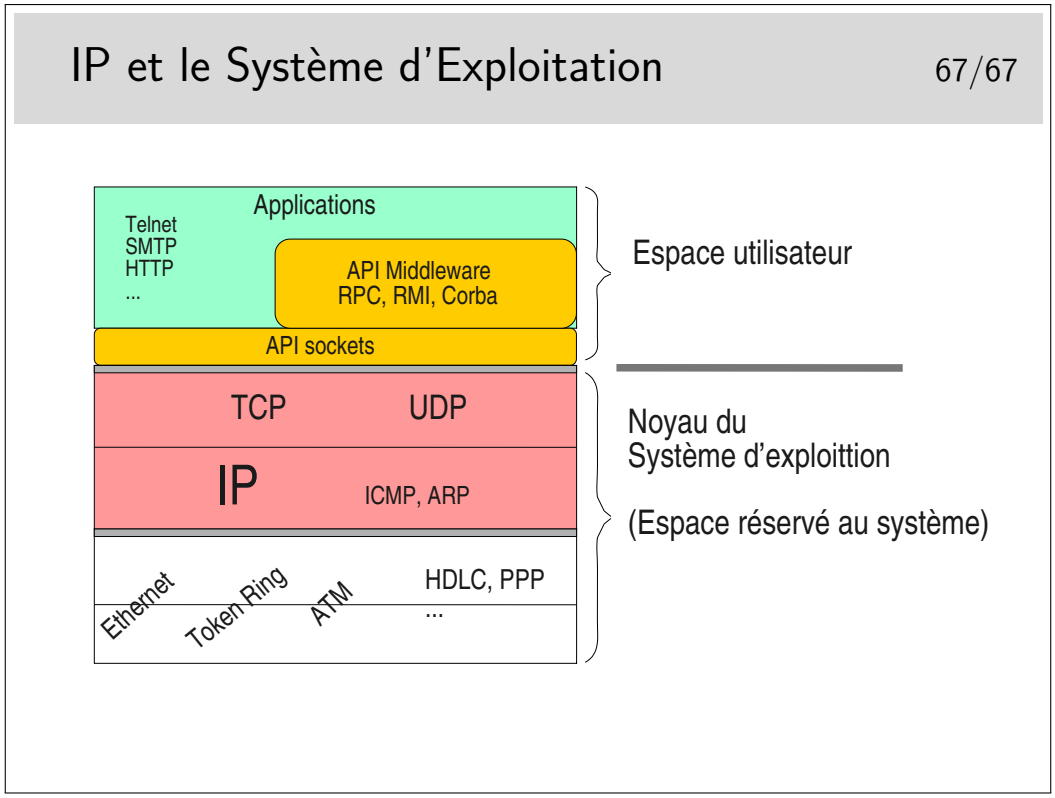
La standardisation Internet 65/67

- Les membres des groupes de l'IETF travaillent par échange de courriers électroniques, sur des documents appelés *drafts*, dont la validité est de 6 mois
- Périodiquement ils se rencontrent dans des «meetings» pour valider des choix techniques
- Un *draft* peut évoluer vers une nouvelle version, valable 6 mois de plus
- Lorsque que le groupe de travail arrive à un consensus, le *draft* est promu au rang de RFC (Request For Comment)
  - Les RFCs sont les «normes» Internet, l'équivalent des recommandations ISO et ITU-T
  - Mais pas seulement...
- RFC et drafts sont en publications gratuite sur différents sites ftp et web (`www.ietf.org`)



Clairement, IP est au niveau 3 et TCP/UDP au niveau 4.

Quand bien même ces protocoles ne sont pas conformes à ceux qui ont été spécifiés pour le modèle OSI, on peut leur trouver ces places dans le modèle.



Les couches protocolaires jusqu'au niveau 4 sont inclues dans le noyau. Les applications (les programmes des utilisateurs) y accèdent via des interfaces qui les masquent.

Ces interfaces sont constituées par des bibliothèques de fonctions formant des APIs utilisables par les programmeurs d'applications. La plus répandue est la bibliothèque socket tout à fait bien adaptée aux protocoles Internet (TCP, UDP, IP). Les couches protocolaires sont vues comme des fichiers qu'on écrit ou qu'on lit. Le transfert de données est aisé puisqu'il se fait

comme avec des fichiers. Le portage est aisé vers le monde Windows qui implémente aussi cette interface programmatique issue des Unix de Berkeley (les sockets BSD).

Un autre type d'API est lui aussi très répandu puisqu'il est utilisé pour NFS et NIS. Au départ spécifié par SUN dans les années 80 il est depuis universel sous Unix. Il s'agit des Remote Procedure Calls ou RPC. La couche RPC offre une abstraction du réseau et permet d'appeler des fonctions en local alors qu'elles s'exécutent à distance.

Le concept a été poussé encore plus loin et adapté aux langages orientés objet comme C++ avec CORBA (Common Object Broker Architecture) qui permet d'appeler des méthodes sur des objets situés sur des machines distantes. Cette interface n'est cependant pas standard (du point de vue API) et plusieurs implémentations existent, la plupart commerciales et d'autre libres comme ORBit qui est utilisé par GNOME sous Linux.

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Cybersécurité des systèmes maritimes et portuaires

La couche transport

Christophe Lohr

2021

1 Introduction

Les protocoles de transport au dessus de IP 3/44

- TCP : Transmission Control Protocol rfc793
  - Transport en mode «connecté»
  - Contrôle de flux et détection d'erreur avec retransmission
- UDP : User Datagram Protocol rfc763
  - Mode datagramme
  - Pas de contrôles, pas d'assurance de délivrance
- Nouveaux
  - DCCP Datagram Congestion Control Protocol rfc4340 2006
  - SCTP Stream Control Transmission Protocol rfc4960 2007
  - MPTCP Multipath TCP rfc6826 2013
  - QUIC A UDP-Based Multiplexed and Secure Transport draft

TCP et UDP, deux besoins extrèmes, mais relativement faciles à utiliser. De nombreux protocoles "universitaires" proposent des compromis, des services intermédiaires (p-ex. le protocole POC : Partial Order Connection), mais certains commencent à émerger sérieusement comme DCCP et SCTP.

DCCP : un genre d'UDP mais en mode connecté et avec contrôle de congestion. Pas de préservation de l'ordre des message ni de garantie de transmission des donnée, mais garantie sur les acquittement.

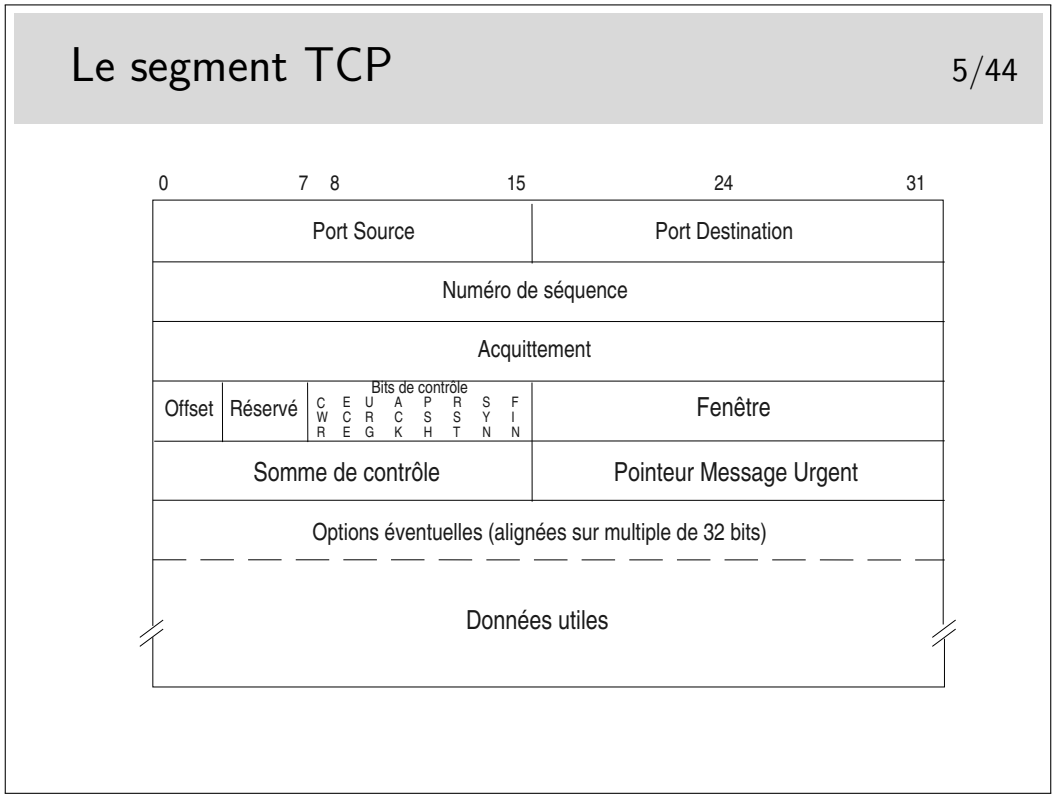
SCTP : un genre de TCP mais orienté *flux de messages* et non pas flux d'octets (flux : préserve l'ordre et guarantie la transmission) ; gère le multi-flux au sein d'une même connection, et le *multihoming* (annoncer que l'on va changer d'adresse IP pour la suite de la session).

MPTCP : compatible avec le TCP classique (programme avec des sockets TCP) ; définit des nouvelles *options* pour annoncer des sous-flux, en parallèle ou en secours, rejoindre une autre IP sans clore la session (*multihoming*), etc.

QUIC : connexion "comme TCP", mais multi-flux (plusieurs conversations dans une même connexion), chiffré de bout en bout façon TLS, et encapsulé dans de l'UDP pour pouvoir passer

les équipements legacy intermédiaires. À l'origine développé par Google vers 2012, toujours en discussion à l'IETF.

2 TCP



Ports source et destination : identifient les applications en relation, généralement l'une d'elle est serveur, son port correspond alors au numéro du protocole (exemple : 80 pour les serveurs Web)

Numéro de séquence : numéro du premier octet des données. C'est le rang du premier octet véhiculé par ce segment en comptant depuis le début de l'échange. Le numéro de début est tiré aléatoirement entre 0 et 232-1.

Acquittement : numéro du prochain octet attendu

Offset : indique la longueur de l'entête en mots de 32 bits (si égal à 5 alors pas d'option)

Bits de contrôle (CWR, ECE, Urg, Ack, Psh, Rst, Syn, Fin)

Fenêtre : permet le contrôle de flux, le récepteur indique avec ce champ combien d'octets il est prêt à recevoir (une valeur de 0 indique que ses tampons mémoire sont pleins)

Somme de contrôle : permet de savoir si le segment a été altéré ou non pendant sa transmission

Pointeur de données urgentes : indique l'emplacement de ces données dans le segment (si le bit Urg est positionné, sinon ce champ est ignoré, bien qu'il existe toujours)

Les bits de contrôle I 6/44

CWR	ECE	URG	ACK	PSH	RST	SYN	FIN

- ECE Explicit Congestion Notification-Echo (rfc 3168), l'entité TCP recevant un segment contenant ce bit à 1 doit réduire son débit
- CWR Congestion Window Reduced (rfc 3168) réponse à la réception du bit ECE pour indiquer que la requête a bien été prise en compte
- URG indique que le champ *Urgent Pointer* est significatif.
- ACK indique que le champ *Acknowledgement Number* est significatif.
- PSH fonction *PUSH*. Les données doivent être immédiatement remises à la couche supérieure.
- RST *reset*, la connexion est rompue.
- SYN synchronisation des indicateurs numéro de séquence. Segments d'initialisation de connexion.
- FIN terminaison de connexion.

Les bits de contrôle II 7/44

Exemple : segment IP et TCP dont le bit ACK est à 1

```
 0: 0800 2074 ef05 0800 0914 18e7 0800 4500
16: 0028 8954 0000 4006 db07 c02c 4b13 c02c
32: 4b08 1770 fdbe 0162 6e86 8e21 a873 5010 /* 0001 0000 */
48: 2210 bba3 0000 023a b3a1 1829
```

Les données «urgentes» 8/44

- Données appelées parfois "hors bande" ou *Out of Band* (OOB)
- Données à traiter en priorité par la couche réceptrice
- Elles sont véhiculées dans le flux normal en suivant le chemin normal. IP n'est pas sensible à ces données, leur caractère "urgent" est significatif seulement aux extrémités
- L'arrivée de ces données a un caractère aléatoire pour les applications destinatrices
- Les applications ne lisent pas ces données dans le flux normal
- Une application devant pouvoir accepter de telles données doit avertir le système pour que celui-ci lui envoie une interruption (un signal logiciel) afin qu'elle puisse traiter en priorité la donnée. L'application doit prévoir une routine spéciale de traitement pour la lecture de ces données
- Sémantique mal définie : le RFC6093 recommande aux nouvelles application ne ne plus l'utiliser...

Le bit de contrôle RST 9/44

- Utilisé par une entité TCP connectée avec une autre entité TCP distante pour avertir d'un problème
- Une application se terminant normalement fait une fermeture sur le port TCP utilisé (souvent une *socket*), ceci se concrétise par un échange de segments avec le bit FIN positionné et la connexion est rompue
- Si l'application se termine brutalement sans fermer la connexion, l'entité TCP associé envoie vers l'autre extrémité un segment avec le bit RST positionné. L'autre extrémité est ainsi prévenue de la terminaison de la connexion

TCP est orienté connexion 10/44

- La connexion TCP :
  - Relation établie point à point entre les deux extrémités
  - Normalement transparente aux routeurs (sauf si ceux-ci mettent en œuvre du filtrage où des mécanismes de QoS)
  - Caractérisée par un contexte dans les machines d'extrémité (numéros IP local et distant, ports local et distant)

La «connexion» TCP I 11/44

- La connexion est établie par un échange de paquets initiaux : le *three way handshake*
  - Ce n'est pas une connexion au sens strict du terme, il n'y a pas de chemin virtuel établi dans le réseau, les segments TCP sont portés par IP, protocole orienté datagramme
  - La connexion TCP se traduit par un contexte mémorisé dans les machines d'extrémité (le client et le serveur), ce contexte est le quintuplet :

```
[protocole, port local, port distant, @IP locale, @IP distante]
```

La «connexion» TCP II 12/44

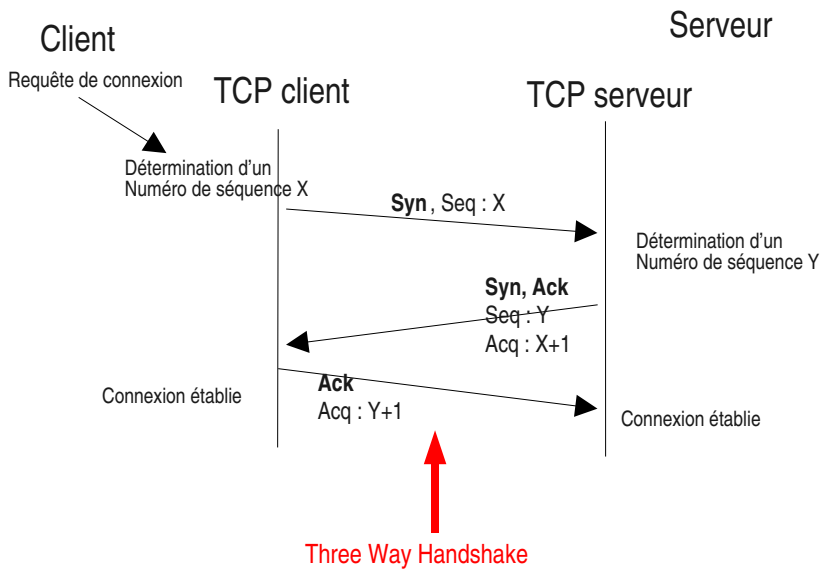
- Si les applications connectées n'ont rien à se dire, les entités TCP correspondantes ne s'envoient aucun segment, il n'y a plus alors de trafic
- Il est possible de faire en sorte que les entités s'échangent des segments d'alerte (sans données utiles puisqu'il n'y en a pas à envoyer) toutes les deux heures en positionnant une option dans TCP
- Si l'entité TCP distante ne répond pas au segment d'alerte ceci signifie que l'application associée s'est terminée sans prévenir ou que la machine s'est arrêtée. L'application locale sera prévenue lors de la prochaine lecture du port TCP

Cette option n'est pas une option au sens dans lequel ce mot sera vu plus loin. Il s'agit simplement d'un paramètre interne à l'entité TCP, positionné par l'application à l'aide d'une fonction spécifique (`setsockopt(SO_KEEPALIVE)`).

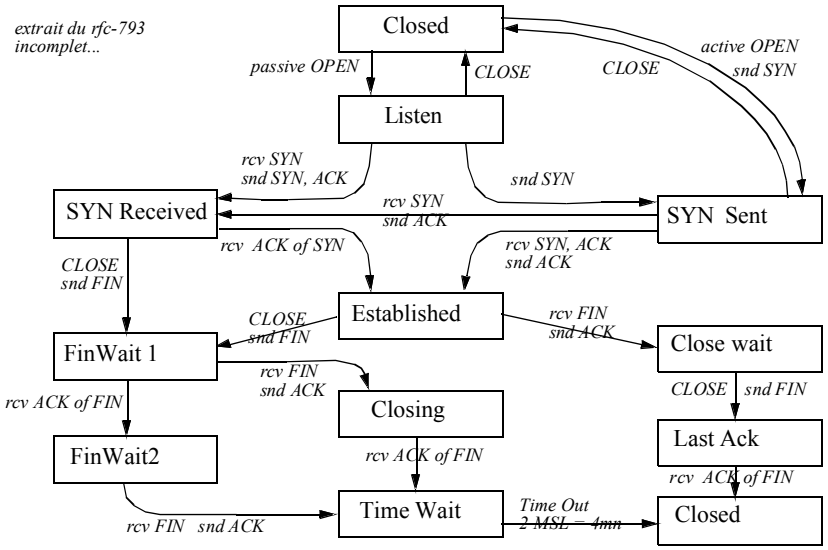
Connexions actives et passives 13/44

- Connexion passive
  - Ce n'est pas une connexion mais un point d'accès TCP ouvert par une application en fonction de serveur
  - Le contexte TCP créé attend une requête de connexion émanant d'un client
- Connexion active
  - La connexion réelle, initialisée par une application en mode client

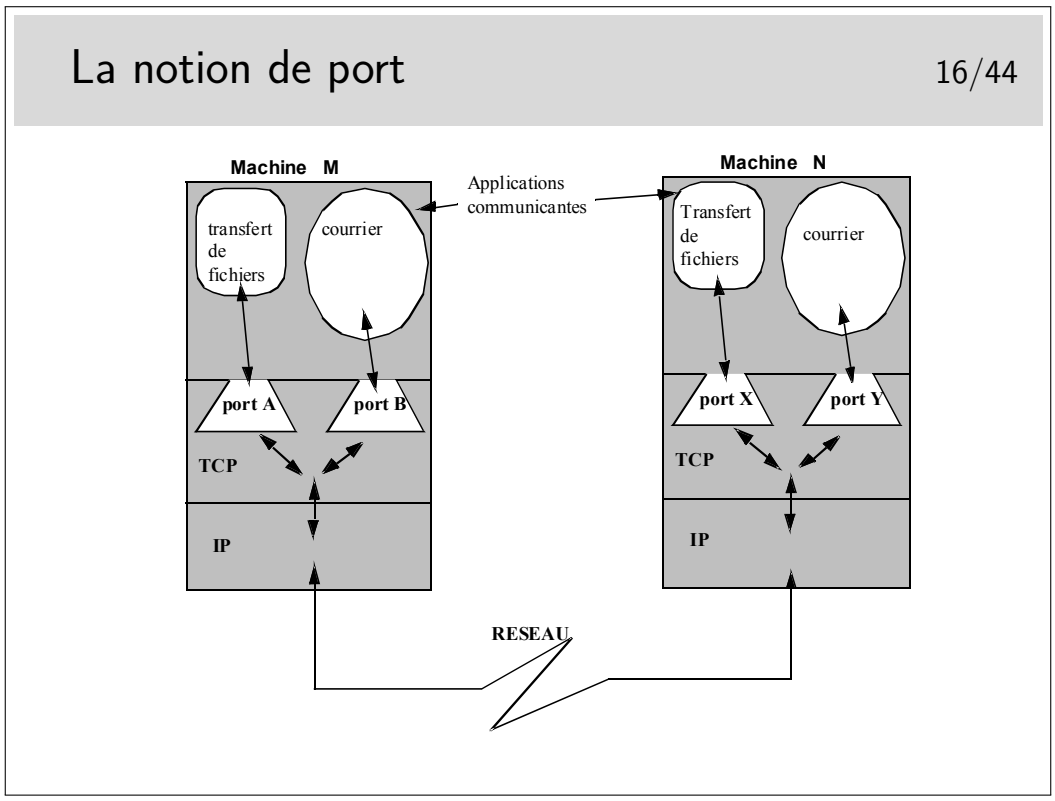
Établissement de la connexion 14/44



La machine d'états TCP 15/44



Ces états sont visualisables sous Windows dans une fenêtre de commande avec la commande `netstat -p tcp`. Sous Linux on fera `netstat -at`.



Les ports sont, en quelque sorte, les adresses des applications à l'intérieur des machines. Selon la terminologie OSI, un port est un SAP.

Quelques ports bien connus (voir aussi le fichier `/etc/services` sous Unix/Linux, ou `C:\WINNT\system32\drivers\etc\services` sous Windows)

21 : ftp (le port du canal de commande de ftp)

20 : ftp-data (le port pour la phase effective du transfert de fichiers par ftp)

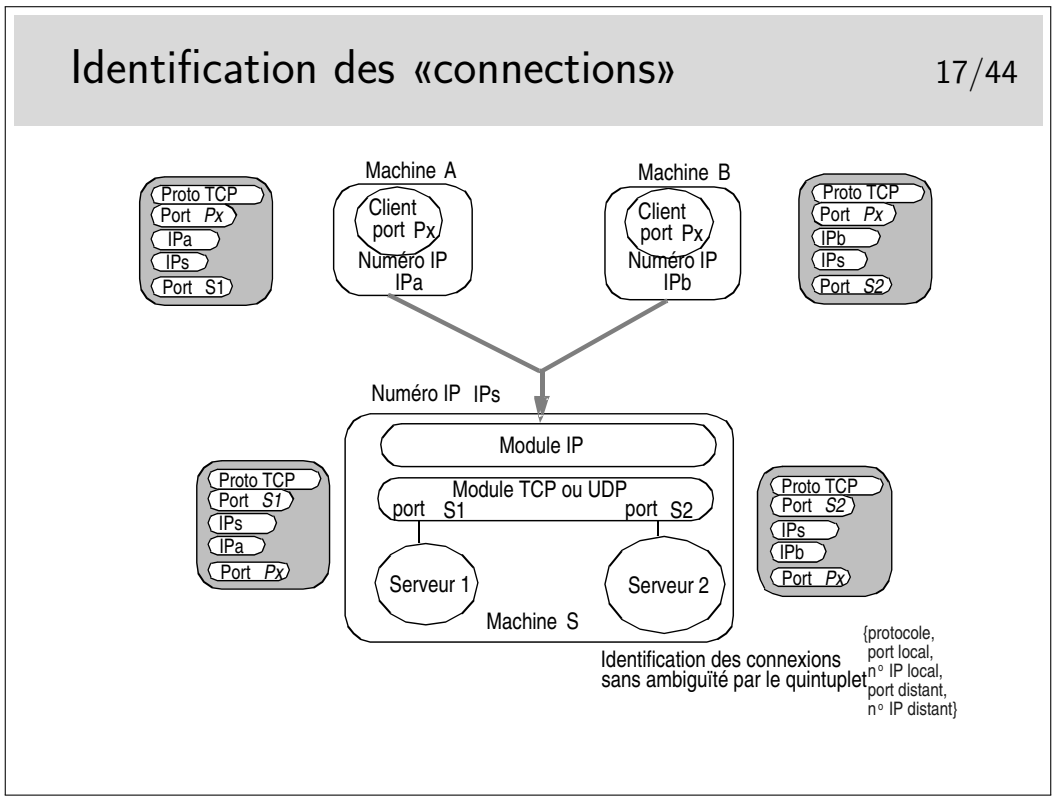
22 : ssh (les connexion à distance sécurisée par chiffrement)

23 : telnet : la même chose mais sans chiffrement

25 : le protocole d'envoie du courrier électronique Internet, SMTP

80 : le protocole http, donc le WEB

etc...



Pour résumer ce qu'est une «connexion TCP» :

— c'est une relation entre deux entités TCP résidant sur des machines différentes (ou entre deux contextes de la même couche TCP d'une machine où s'exécutent les applications en communication). En terminologie OSI, dans la couche Application on utilise le mot «association» plutôt que «connexion». Il serait plus approprié ici aussi.

— il n'y a pas de connexion associée dans le réseau

— dans chaque entité TCP en relation un contexte est créé et est associé à l'application ne cause de son coté

— ce contexte comprend :

  — une machine d'état finis gérant les états de la connexion

  — un quintuplet d'identification de connexion

  — des tampons mémoire d'émission et de réception

  — diverses variables complémentaires comme, entre-autre, le compteur associé à la gestion des congestions

— C'est l'application qui demande la création de ce contexte, à l'aide de fonctions d'une API spécifique telle que les Sockets (API issue du monde Unix, BSD au départ et ensuite reprise par tous les Unix et portée sous Windows)

Les options TCP I

- Caractéristiques
  - Le champ offset de l'entête étant codé sur 4 bits (il indique le nombre de mots de 32 bits de l'entête), la longueur max de l'entête est de 15 mots de 32 bits (15dec = 1111bin). L'entête minimum étant de 5 mots de 32 bits (20 octets), la longueur du champ option est de 10 mots de 32 bits max.

Les options TCP II

- Options standards
  - mss : *maximum segment size*, permet de négocier la taille maximale des segments TCP afin d'éviter des segmentations coûteuses. cette option est utilisées au moment de la connexion. Longueur 4 octets.
  - pas d'opération (NOP) : option nulle, permet d'aligner la prochaine option sur un début de mot de 32 bits. Plusieurs NOP peuvent se suivre si besoin. (lg : 1 octet)
  - fin de liste : permet d'aligner la fin des options sur un mot de 32 bits (lg 1 octet)

Les options TCP III 20/44

- Multiplicateur du champ fenêtre (window scale)
  - valeur permettant de multiplier la fenêtre par un multiple d'une puissance de 2 (jusqu'à 2^{14}) (rfc 1323). Ceci permet d'utiliser plus efficacement la bande passante pour des sources à haut débit et/ou des communications à grande distance en permettant un flux le plus continu possible
- Horodatage des données (rfc 1323)
  - L'option contient deux valeurs : un indicateur d'heure d'émission et un acquittement. Une source peut ainsi calculer le temps d'aller et retour sur un chemin en plaçant sa valeur d'horloge dans le premier champ. Lorsque l'acquittement du segment correspondant arrivera, cette valeur sera contenue dans le second champ de cette option. Il sera alors facile de retrancher cette valeur de la valeur courante de l'horloge.

Les options TCP IV 21/44

- Négociation de l'acquittement sélectif (rfc 2018)
- Acquittement sélectif : la première option permet d'indiquer qu'une source TCP est capable d'utiliser cette fonctionnalité, la seconde met en œuvre ce type d'acquittement permettant de demander uniquement la retransmission de données perdues en évitant de retransmettre des données suivantes bien reçues

Capture d'une séquence TCP 22/44

rfc-1323 et rfc 2018 (relevé avec tcpdump lors d'un début de requête web)

```
linux1.1082 > 206.132.41.203.www: S 2047350264:2047350264(0) win 16060 <mss
1460,sackOK,timestamp 43244276>
206.132.41.203.www > linux1.1082: S 2319802134:2319802134(0) ack 2047350265 win
32120 <mss 1460, sackOK, timestamp 190973015>
linux1.1082 > 206.132.41.203.www: . ack 2319802135 win 16060 <nop,nop,timestamp
43244311 190973015>
linux1.1082 > 206.132.41.203.www: P 2047350265:2047350896(631) ack 2319802135 win
16060 <nop, nop, timestamp 43244311 190973015>
206.132.41.203.www > linux1.1082: . ack 2047350896 win 31856 <nop, nop, timestamp
190973051 43244311>)
206.132.41.203.www > linux1.1082: P 2319802135:2319803583(1448) ack 2047350896 win
31856 <nop, nop, timestamp 190973063 43244311>
linux1.1082 > 206.132.41.203.www: . ack 2319803583 win 14612 <nop,nop,timestamp
43244367 190973063>
206.132.41.203.www > linux1.1082: P 2319803583:2319805031(1448) ack 2047350896 win
31856 <nop, nop, timestamp 190973063 43244311>
```

S : bit Syn

ack : bit ack

P : bit Push

. : pas de flag (autre que ack)

sackOK : acquittements sélectifs en fonction

mss : maximum segment size

en italique : complément d'information fourni par tcpdump mais ne figurant pas réellement dans le paquet

numéro de séquence du dernier octet, correspondant à l'acquittement alors attendu. Entre parenthèses : nombre d'octets du paquet

Les trois premières trames correspondent à une ouverture de connexion (Three Way Handshake)

Le contrôle des connexions : la commande `netstat` 23/44

- La commande `netstat` avec l'option `-a` (sous Unix/linux, ou `-p tcp` sous Windows)

```
[linux30]$ netstat -atn
Connexions Internet actives (serveurs et établies)
 Proto   Recv-Q   Send-Q   Adresse locale      Adresse distante         Etat
 tcp          0        0   0.0.0.0:32768       0.0.0.0:*               LISTEN
 tcp          0        0   127.0.0.1:32769     0.0.0.0:*               LISTEN
 tcp          0        0   0.0.0.0:111         0.0.0.0:*               LISTEN
 tcp          0        0   0.0.0.0:6000        0.0.0.0:*               LISTEN
 tcp          0        0   0.0.0.0:22          0.0.0.0:*               LISTEN
 tcp          0        0   0.0.0.0:631         0.0.0.0:*               LISTEN
 tcp          0      272   192.168.100.30:22   192.168.100.18:1994   ESTABLISHED
```

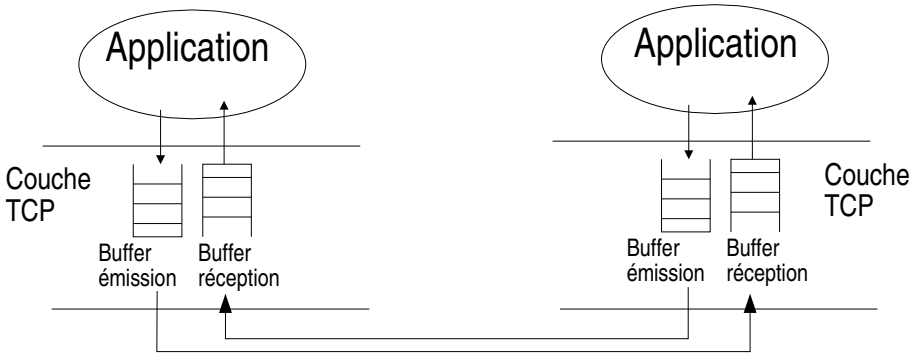
L'option `-n` de netstat permet de ne pas traduire les adresses et numéros de ports (les services). L'affichage est direct et on ne perd pas de temps en requêtes DNS pour afficher les noms des machines.

L'option `-t` restreint l'affichage au seul protocole TCP. Avec l'option `-u` on affiche les services UDP en cours (les applications serveur UDP). Il n'y a pas de connexion en UDP.

Avec la seule option `-a` on affiche toutes les connexions (ou serveurs lancées pour UDP), y compris les services n'utilisant que la communication Unix (par exemple le serveur d'affichage X-Window local et ses applications clientes).

Voir `-p tcp` ou `-p udp` sous Windows

TCP et les applications 24/44



Les données sont bufferisées et TCP les émet selon ses propres règles de contrôle de flux. Les applications n'ont pas le contrôle.

Selon la bibliothèque de développement (sockets sous Unix/Linux ou Windows par exemple) il peut être possible de dimensionner les tampons mémoire d'émission et de réception.

Le contrôle de flux TCP I 25/44

- Contrôle de flux d'extrémité à extrémité : le champ window
  - Les routeurs ne mémorisent pas "la connexion" TCP, ils ne peuvent donc pas participer au contrôle de flux. Celui-ci ne peut être réalisé que par les extrémités via le champ fenêtre.
  - Les entités TCP d'extrémités peuvent enregistrer les données reçues dans un tampon mémoire de taille limitée (8, 16, 32 Ko, ...)
  - Si l'application réceptrice ne lit pas suffisamment vite les octets s'accumulent dans le tampon mémoire de réception.

../..

Le contrôle de flux TCP II 26/44

../..
- L'entité TCP réceptrice envoie des acquittements avec une valeur de fenêtre qui diminue pour venir à 0 s'il le faut.
- 5s après avoir reçu un tel acquittement (`win=0`), l'entité émettrice teste le récepteur en lui envoyant un octet et continue ainsi en doublant l'intervalle (jusqu'à une borne de 1 min). Ce moyen permet de forcer le récepteur à renvoyer un acquittement et donner ainsi sa fenêtre. La valeur de 5s est en réalité variable selon les implémentations.
- Dès que les octets reçus par l'entité TCP réceptrice seront consommés par l'application réceptrice, la fenêtre pourra revenir à une valeur différente de 0.

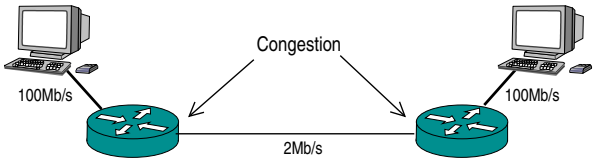
Sur Linux (noyau 2.6.xx), les paramètres liées à la gestion de la taille de la fenêtre de réception dans la pile TCP sont accessibles par la commande :

```
$ cat /proc/sys/net/ipv4/tcp_rmem
4096 87380 2076672
```

Les trois nombres donnent respectivement la valeur minimale (4096 octets), par défaut (87380 octets) et maximale (2076672 octets) de la fenêtre d'une session TCP.

Le problème de la congestion 27/44

- Il n'y a pas que les entités d'extrémités en jeu, il y a aussi le réseau. Le mécanisme de fenêtre ne permet pas d'adapter le contrôle de flux aux congestions du réseau.
- S'il y a congestion dans le réseau, des paquets peuvent être perdus, les retransmissions qui en découleront participeront au renforcement de la surcharge totale.
- Le congestion peut être due à des liens saturés mais aussi à des différences de débits entre segments de réseau.



Comment adapter TCP ?

Adaptation de TCP pour le contrôle de congestion I 28/44

- Détermination du RTT (*Round Trip Time)*
  - Avec les options spécifiques
  - Permet de «régler» un temporisateur de retransmission

Adaptation de TCP pour le contrôle de congestion II

29/44

- Le mécanisme du «*slow start*»
  - Une fenêtre de congestion est définie : `cwnd` (*congestion window*). On ne peut pas émettre plus que `cwnd` octets.
  - Juste après la connexion `cwnd` vaut 1mss (*max segment size*).
  - Un RTT plus tard, ce nombre double, et double à chaque RTT jusqu'à un seuil fixé à l'origine à 65535 (on ne peut cependant émettre que `min(cwnd, w)`, `w` étant la valeur de la fenêtre de l'entête TCP).
  - Lorsqu'une congestion a lieu, une perte se produit, le temporisateur expire et provoque une retransmission, le seuil est divisé par 2 et `cwnd` est remis à 1mss.
  - Si le seuil est dépassé, la progression de `cwnd` devient linéaire.

Les valeurs de RTT sont typiquement de :

10 ms entre sites intra Renater,

35 ms entre des sites intra Europe,

90 ms entre des sites situés en France et aux USA

210 ms entre des sites situés en Europe et en Asie

Adaptation de TCP pour le contrôle de congestion III

30/44

- Le mécanisme du «*fast recovery*» (entre autres) vient compléter le low start
  - Si on reçoit un même acquittement plusieurs fois, cela signifie deux choses : d'une part c'est qu'on a bien envoyé des segments et que ceux-ci ont été bien reçus (sinon on n'aurait pas reçu d'acquittement du tout) et d'autre part il doit y avoir un «trou» dans la séquence de segments transmis. Il suffit alors de ne retransmettre que le segment qui semble perdu.
  - Lorsque que plusieurs pertes successives ont lieu, ce mécanisme ne suffit plus car il ne permet que de récupérer le premier segment perdu. Il faut alors utiliser le mécanisme des acquittements sélectifs.

Comment éviter la segmentation avec TCP 31/44

- Le mécanisme du *Path MTU Discovery (rfc 1191)*
- le problème
  - la couche TCP émission connaît le MTU de l'interface IP de sortie (1500 par défaut sur Ethernet). Sur le chemin vers le récepteur un lien peut avoir un MTU inférieur (700 par exemple), le routeur amont sur ce lien devra segmenter.
- la solution
  - les routeurs sont interdits de segmentation TCP (bit D à 1). Si un paquet se présente de taille trop grande il est rejeté et le routeur qui le rejette émet un paquet ICMP vers l'émetteur. Ce paquet contient une information significative. Voir exemple :

Exemple de PATH MTU Discovery 32/44

Exemple : extrait d'un relevé avec tcpdump

- émission du premier paquet (seq = 1, taille 1448=MTU(1500)-TCPhead(20)-TCPoptions(12)-IPhead(20))

```
Em > Rec P 1:1449(1448) ack 1 win 16060 <nop,nop,timestamp 830140
827243>
```

- réception du mesage ICMP émis par un routeur intermédaire (celui qui devrait segmenter)

```
192.168.200.17 > Em: icmp: Rec unreachable - need to frag (mtu 700)
```

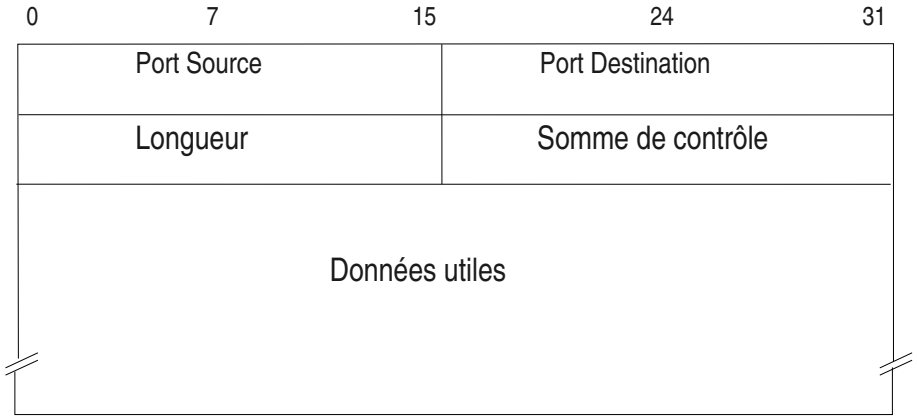
- On a bien compris... On émet des paquets de 648 octets (700-12-20-20=648)

```
Em > Rec . 1:649(648) ack 1 win 16060 <nop,nop,timestamp 830140
827243>
```

En italique : ce que tcpdump nous indique concernant la longueur du SDU véhiculé.

3 UDP

Le datagramme UDP 34/44



UDP 35/44

- Pas de connexion
- Pas de contrôle de flux
- Pas d'assurance de la remise
- Mode message : alignement des données reçues sur les données émises

UDP n'apporte que peu de choses par rapport à IP. Il garde de TCP la notion de port qui permet d'identifier les applications.

(A noter le champ longueur qui n'existe pas dans TCP)

Il n'y a pas de contrôle de flux, pas d'assurance de remise.

Il n'y a pas de «connexion».

On a cependant l'assurance que, si un datagramme arrive, il correspond très exactement à ce qui a été envoyé en ce qui concerne l'alignement des données. En d'autre termes, on peut dire

que UDP est orienté «message». Si un message est reçu, le début correspond au début envoyé, la fin à la fin...

Ce n'est pas vrai en TCP. Lorsqu'une application reçoit un buffer de données via TCP elle ne peut être certaine que cette réception correspond bien à l'émission. Certes les données sont les mêmes, elles ne sont pas alignées. TCP est orienté «flot d'octets», il assure que les octets sont bien transmis.

TCP, UDP et l'alignement des données 36/44

- TCP est orienté «flot d'octets», il ne permet pas d'assurer le «cadrage» des données reçues sur celui des données émises

  Exemple de possibilité :

- UDP est orienté «Message»

TCP garantit que les octets sont bien transmis, il ne permet pas l'alignement de la réception sur l'émission. Autrement dit, un premier «message» peut voir son début partir dans un segment TCP et arriver dans un autre. Il faut pouvoir rassembler les octets appartenant à un message donné à l'arrivée. Pour cela on peut utiliser la technique TPKT recommandée dans le RFC 1006 qui consiste à commencer les messages par un octet de version (3), un octet inutilisé (sans doute pour faire joli, ou plus sûrement pour que l'ensemble soit aligné sur 4 octets), et deux octets de longueur de message. En réception il suffit de se synchroniser sur le premier message...

UDP ne garantit pas que les messages arrivent mais s'ils sont reçus, alors, on trouve tout d'un coup (à condition toutefois que la lecture prévoit de de lire un nombre suffisant d'octets).

4 Protocoles applicatifs

Les protocoles applicatifs I 38/44

- Mis en œuvre par les applications
- Des API existent
  - Sockets (BSD) : le réseau est vu comme un fichier (on écrit et on lit le réseau comme un fichier), le protocole de communication est à implémenter «à la main»
  - Transport Services API : une API générique, draft IETF
  - RPC : l'API masque les aspects réseau, on appelle des procédure distantes de la même manière que des procédures locales
  - CORBA : *Common Request Broker Architecture*, extension aux applications réparties du concept «programmation objet»
  - ... multiples *middlewares* spécifiques ...

Les protocoles applicatifs II 39/44

- Codage des données
  - Protocoles «texte» : smtp, http (1.x), sip, sdp
  - Protocoles codés :
    - ASN.1/BER/PER : H323, snmp
    - XDR : NFS, NIS
- Notions de «session»
  - p.ex. protocole sip, sdp, beep, et même les *cookies* sur le web, etc.

smtp : simple mail transfert protocol (rfc-821)

http : hyper text transfert protocol (le protocole du Web, rfc-1945)

sip : session initiation protocol (un des protocoles pour la téléphonie sur IP, rfc-2543)

sdp : session description protocol (descripteur de flux multimedia, inspirateur de sip : rfc-2327)

beep : blocks extensible exchange protocol, un framework pour créer des protocoles applicatifs, incluant des mécanismes d'authentification et de gestion de session, rfc 3080 3081 3117

ASN.1 : Abstract Notation, version 1 (on espère qu'il n'y aura pas de version 2, ou alors plus simple...) : langage de spécification de types de données. Comme un langage informatique ou on ne définirait que des types et des variables de ces types.

On lui associe une syntaxe de transfert (en clair : des mécanismes de codage en ligne) tels que le BER (Basic Encoding Rules) ou le PER (Packet encoding Rules).

ASN.1/BER : utilisé pour snmp (simple network management protocol, rfc 1155 à 1158, 1212, 1213, etc)

ASN.1/PER est utilisé pour définir les données de signalisation pour la téléphonie sur IP dans l'architecture H323 (définition des PDUs H225 et H245, plusieurs centaines de pages, avec rien que du ASN.1 goûteux et savoureux).

On retrouve aussi ASN.1 dans les spécifications de la structure des certificats de sécurité X509.

JSON : JavaScript Object Notation, un format de données textuelles structurées façon Javascript (Objets). Très utilisé dans les applications Web (échanges de données entre le navigateur de l'utilisateur et le serveur, mis en forme dans une page html)

CBOR : Concise Binary Object Representation, similaire à JSON, mais avec une représentation binaire (donc plus compact).

Exemple d'échange HTTP : la requête 40/44

```
GET /index.fr.php HTTP/1.1
Host: www.enst-bretagne.fr
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:1.
Gecko/20030624 Netscape/7.1 (ax)
Accept: text/xml,application/xml,.....
Accept-Language: fr,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://www.enst-bretagne.fr/
```

Exemple d'échange HTTP : la réponse 41/44

```
HTTP/1.1 200 OK
Date: Fri, 09 Jul 2004 09:20:06 GMT
Server: Apache/1.3.22 (Unix) PHP/4.0.4pl1 mod_fastcgi/2.2.10 PHP/3.0.....
X-Powered-By: PHP/4.0.4pl1
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html
Content-Language: fr

<HTML>
<HEAD>
<TITLE>[ENST Bretagne] école formation ingénieur</TITLE>
<!-L'école nationale supérieure des télécommunications de Bretagne offre
un large choix de formation: ingénieur, mastères, télécom, thèse, DEA...->
<meta name="robots" content="noindex,nofollow">
<meta name="description" content="L'école nationale supérieure des télécommunications de Bretagne offre un
large choix de formation: ingénieur, mastères, télécom, thèse, DEA...">
<meta name="keywords" content="école ingénieur, formation ingénieur, école nationale supérieure
télécommunications, Bretagne, mastères, télécom, thèse, DEA, ENST, GET, étude télécommunication,
enseignement supérieur, technologie information, communication, TIC, concours, mines-ponts, alternance,
projets européens, recherche, laboratoire, grandes écoles, formation continue, diplôme, Brest, Rennes,
ECOLE INGENIEUR, FORMATION INGENIEUR">
<LINK rel="stylesheet" href="css/style.css">
...
```

Coté client, c'est au programmeur de l'application de préparer «à la main» les chaînes de caractères constituant les messages de requêtes du protocole. Coté serveur il faut savoir interpréter ces messages (le mot «savoir» étant traduisible en «implémentation de code»), il faut savoir aussi constituer les messages de réponse. Ces messages sont donc des suites de caractères ascii, placés dans des tampons mémoire qu'il suffit ensuite d'envoyer sur des sockets, coté émission. Coté réception il suffit de lire les sockets (comme on lirait des fichiers) et d'analyser en,suite ce qui est mémorisé dans les tampons de lecture.

Le langage est simple, il obéit à une grammaire (au sens informatique du terme) parfaitement spécifiée. Il est possible, à partir des spécifications de générer facilement des fonctions de traitement des messages. Il existe des outils adaptés aux traitements lexicaux, voire sémantiques, de telles grammaires : lex et yacc sous unix, flex et bison en logiciels libres.

Exemple d'échange smtp 42/44

(Exemple extrait du rfc-821)

```
S: MAIL FROM:<Smith@Alpha.ARPA>
R: 250 OK

S: RCPT TO:<Jones@Beta.ARPA>
R: 250 OK

S: RCPT TO:<Green@Beta.ARPA>
R: 550 No such user here

S: RCPT TO:<Brown@Beta.ARPA>
R: 250 OK

S: DATA
R: 354 Start mail input; end with <CRLF>.<CRLF>
S: Blah blah blah...
S: ...etc. etc. etc.
S: <CRLF>.<CRLF>
R: 250 OK
```

`S:` indique l'émetteur, `R:` indique le récepteur. (`S:` et `R:` ne font pas partie du protocole.)

`MAIL FROM:`, `RCPT TO:`, `DATA` sont des entêtes du protocole, de même que les séquences `<CRLF>` qui signifient simplement *retour à la ligne* pour `CR` (Carriage Return) et *saute de ligne* pour `LF` (Line Feed).

On peut noter que le cacatère `.` (point) est aussi partie intégrante du protocole.

Voici donc le protocole de base du courrier Internet si utilisé aujourd'hui. Ses spécification datent d'août 1982 !

Que pensez-vous des aspects sécurité d'un tel protocole ?

Fonctionnement du courrier électronique I 43/44

- Les utilisateurs ont une adresse de format standard
  - Par ex. :
    - `nom_utilisateur@nom-du-serveur.domaine`
  - ou plus concis :
    - `nom_utilisateur@domaine`
- Les outils mis en œuvre sont
  - Le client (MUA : *Mail User Agent* en langage OSI) : netscape, outlook, lotusNotes, etc...
    - Parle SMTP lors de l'envoi
    - Parle POP ou IMAP (sécurisé ou non) pour recevoir
  - Les serveurs (MTA : *Message Transfert Agent*) : ISS, sendmail, Postfix, etc...
    - Intermédiaires et final
  - Le DNS : requêtes MX (*Mail Exchanger*)

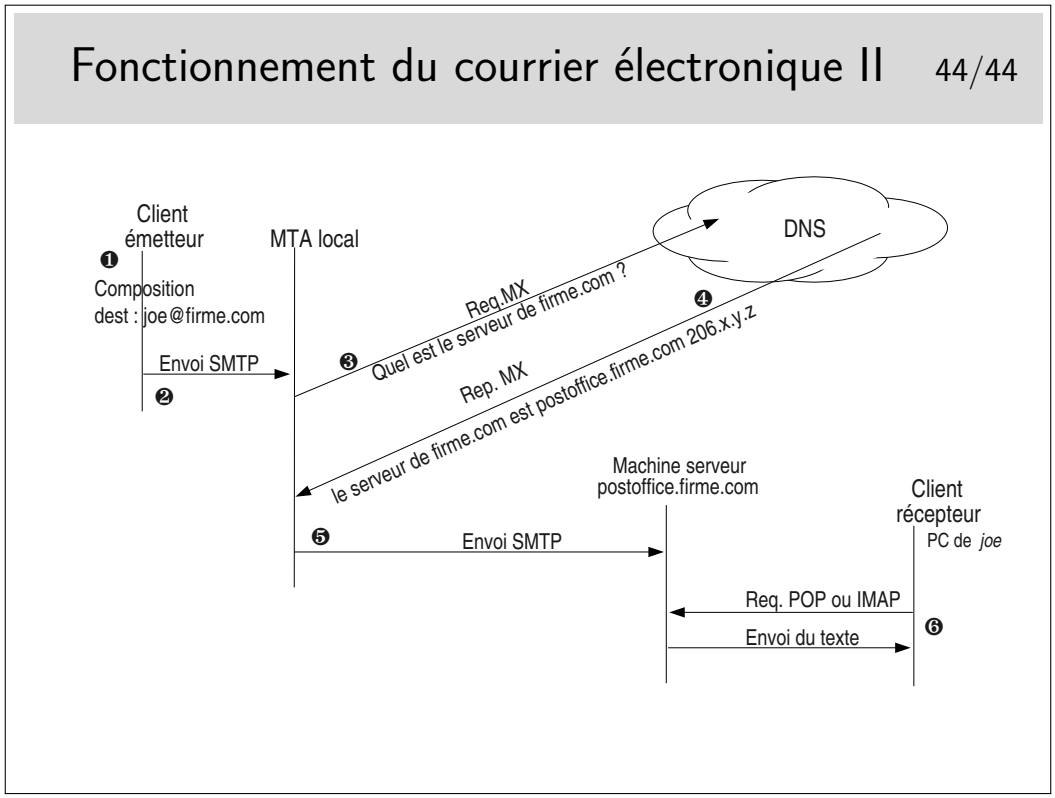
POP : Post Office Protocol

IMAP : Internet Message Access Protocol

Les protocoles POP et IMAP sont par défaut sans chiffrement et nécessitent l'échange de mots de passe utilisateur. Ces échanges se font par défaut en clair pour POP. IMAP supporte différents mécanismes d'authentification, notemment un système de challange dans lequel le mot de passe est chiffré (pour autant, les messages ne le sont pas...) Les outils clients ainsi que les serveurs peuvent être configurés pour fonctionner avec chiffrement (POPS ou IMAPS : POP/IMAP + SSL).

POP permet de télécharger sur son poste client les courriers reçus sur le serveur final. Les messages sont totalement transférés. Ils peuvent cependant rester sur le serveur.

IMAP permet de ne charger que les entêtes, il permet aussi de rechercher les messages par mot clé, les messages peuvent rester stockés sur le serveur, et de gérer des sous-répertoires (*folders*) sur le serveur.



Il peut y avoir plus de machines intermédiaires que ne le montre la figure ci-dessus. En particulier, coté réception, la machine serveur (appelée ici `postoffice.firme.com`) peut renvoyer vers un serveur interne déservant le département de l'utilisateur joe, ce denier paramètre alors son client pour qu'il se connecte par POP ou IMAP sur ce serveur particulier.

Il peut y avoir plusieurs serveurs d'arrivée, classés selon des priorités dans le DNS.

Il peut être intéressant de regarder l'entête complète des messages que l'on reçoit pour avoir la liste des MTA traversées par ces messages. Cette curiosité étant encore plus intéressante en cas de problèmes d'acheminement.

Les MTAs peuvent filtrer les messages, au départ comme à l'arrivée. Ils peuvent être munis de détecteurs de SPAMs et de virus.

Vous pouvez tester le DNS avec des requêtes MX de la manière suivante (sous Windows) :

```
c:\nslookup
> set type=MX
> microsoft.com
```

Ou sous Unix : `host -t MX microsoft.com` (la commande `nslookup`, bien qu'encore présente, est maintenant considérée comme rendue obsolète par la commande `host`).

Quizz
Networking Introduction 2020

September 29, 2020

Overview - Open qustions

1. **What's the internet?**

   Your understanding of the Internet
   a) what's an application? where do applications run?
   b) describe the internet with your own words

2. **Network Edge**

   Network edge
   a) What is an access network?
   b) Mention some access technologies you know
   c) Which equipement can we typically find at the edge of the network?

3. **Network Edge 2**

   When a resident uses a DSL service to connect to the Internet, what is the role of the telephone company?

4. **Network Core**

   Consider a packet-switched network
   a) What equipements can we typically find at the core network?
   d) What are the main functions of routers?
   e) What is a packet header?

5. **Protocols**

   a) What is a protocol?
   b) Why are protocols important?
   c) Name some computers protocols you know

6. **Layers**

   a) Why the internet has adopted a layered model?
   b) What does the term encapsulation refer to?

7. **Efficiency**

   Assume a system uses five protocol layers. If the application program creates a message of 100 bytes and each layer (including the fifth and the first) adds a header of 10 bytes to the data unit, what is the efficiency (the ratio of application layer bytes to the number of bytes transmitted) of the system?

8. **Packetizing**

   Consider a packet-switched network like the Internet. Using the TCP/IP protocol suite, we need to transfer a huge file. Mention an advantage and a disadvantage of sending large packets.

The TCP/IP Model - Multiple Choice Questions

1. **Vocabulary**

   The transport-layer packet in the TCP/IP protocol suite is called :

   (a) a message
   (b) a datagram
   (c) a segment or a user datagram
   (d) a frame

2. **Layers 1**

   In the TCP/IP protocol suite, the layer responsible for moving frames from one hop (node) to the next is:

   (a) physical
   (b) data-link
   (c) transport
   (d) network

3. **Layers 2**

   In the TCP/IP protocol suite, the physical layer is concerned with the movement of _____over the physical medium.

   (a) programs

(b) dialogs
(c) protocols
(d) bits

4. **Layers 3**

   The _____layer is responsible for the delivery of a message from one process to another.

   (a) physical
   (b) transport
   (c) network
   (d) application

5. **Ports**

   In the TCP/IP protocol suite, a port number is the identifier at the _____.

   (a) application layer
   (b) transport layer
   (c) network layer
   (d) physical layer

6. **IP reliability**

   The Internet Protocol (IP) is _____protocol.

   (a) a reliable
   (b) a connection-oriented
   (c) a reliable and connection-oriented
   (d) an unreliable

7. **Encapsulation 1**

   In TCP/IP, a message at the application layer is encapsulated in a packet at the _____layer.

   (a) network
   (b) transport
   (c) data-link
   (d) physical

8. **Encapsulation 2**

   In TCP/IP, a message at the transport layer is encapsulated in a packet at the _____layer.

   (a) network
   (b) transport
   (c) data-link
   (d) physical

9. **Encapsulation 3**

   In TCP/IP, a message belonging to the network layer is decapsulated from a packet at the _____layer.

   (a) network
   (b) transport
   (c) data-link
   (d) physical

10. **Encapsulation 4**

    In TCP/IP, a message belonging to the transport layer is decapsulated from a packet at the _____layer.

    (a) network
    (b) transport
    (c) data-link
    (d) physical

IMT Atlantique
Département Informatique
Technopôle de Brest-Iroise - CS 83818
29238 Brest Cedex 3
URL: **www.imt-atlantique.fr**

Networking Introduction
Fall 2020

Lab DNS: Domain Name System

Edited: September 3, 2021
Version: 1.3-2020-09

Report filled-in by:

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

1. Objectives

This lab aims at helping you to:

- Understand how the domain names system of the Internet works
- Get familiarised with the packet analyser software Wireshark initial objective

2. Pre-LAB

- Study the linux commands `ifconfig, ip address, ip route, dig, host, ping`. You can, for instance, read the man page of these commands by typing on a UNIX terminal `man <command>` (For the `ip` tools commands, you must replace the space with a dash, e.g. `man ip-address`).
- Investigate the different types of resource records existing for the Internet class in the DNS, and their purpose. In particular, complete Table 1.
- Investigate what are iterative DNS queries and recursive DNS queries.
- Readings:
  - The introduction of this lab
  - Serveur DNS faisant autorité : définition, by Stéphane Bortzmeyer: https://www.bortzmeyer.org/serveur-dns-faisant-autorite.html
  - Résolveur DNS : définition https://www.bortzmeyer.org/resolveur-dns.html
  - On the usage of DNS by CDNs: https://labs.ripe.net/Members/emileaben/how-ripe-atlas-help
  - On censorship applied through DNS queries: https://labs.ripe.net/Members/stephane_bortzmeyer/dns-censorship-dns-lies-seen-by-atlas-probes (optional)
  - Nouvelles attaques facilitant l'empoisonnement DNS: https://www.bortzmeyer.org/dns-attaques-shulman.html

3. Introduction

3.1. The Domain Names System

We, as humans, are used to refer to Web pages, mailboxes, and other network resources by using a readable, easy to remember name, like for instance www.imt-atlantique.fr. However, network equipments understand and use numerical addresses (e.g. IP addresses). Having only IP address to refer to network resources would mean, for instance, accessing the Web page hosted at 2001:660:7302:2::21 (or 192.108.117.237), which is not only hard to remember, but supposes that if the Web page changes its location, then we, users, should be aware of this change. As a consequence, a high-level readable names system is used in the Internet in order to allow to decouple machine *names* from machine *addresses*.

How to achieve this decoupling and the mapping between names and addresses at the Internet scale? How to have unique names all across the Internet? How to avoid having a central entity managing the mapping? How to accomplish a system at the Internet-scale avoiding huge files/data bases? The way the Internet community has solved this is called Domain Name System (DNS), and has become a key component of the Internet.

The essence of DNS is a hierarchical, domain-based naming scheme and a distributed database system for implementing this naming scheme. It is primarily used for mapping host names (often referred to as Fully Qualified Domain Names (FQDN)) to IP addresses but can also be used for other purposes [8]. DNS

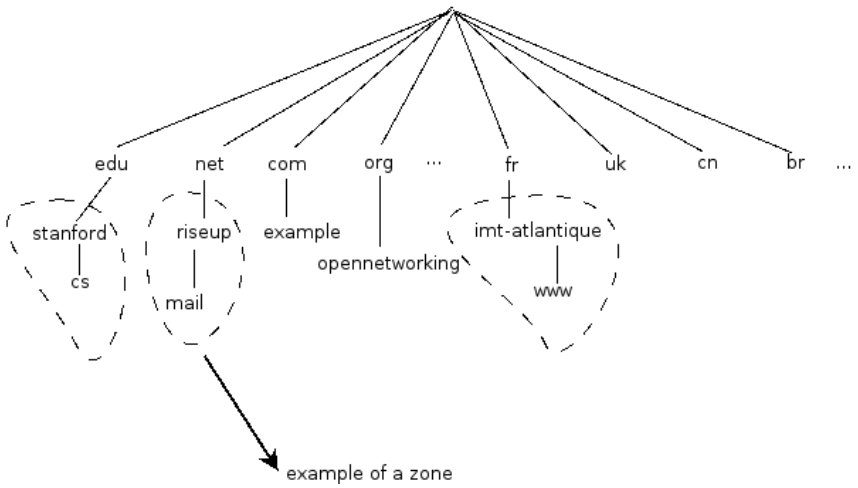


Figure 1: Example of the DNS Name space

is defined in multiple RFCs [1], [2], [3] and more than 20 others. RFCs (Request for Comments) are documents published by the Internet Engineering Task Force [7], Internet community defining Internet standards.

3.1.1. The DNS Name Space

In order to avoid name collisions, the DNS name space is organized in a hierarchical way. The top of the naming hierarchy for the Internet is managed by the ICANN (Internet Corporation for Assigned Names and Numbers), an organization created in 1988 for this purpose. Figure 1 shows some of the so-called top-level domains and several sub-domains, in a tree representation. The leaves of the tree represent domains that have no sub-domains (but can of course contain one or thousands of *records*). Each domain is named by the path upward from it to the (unnamed) root. The components (*labels*) are separated by a dot.

Each domain, sub-domain or *zone* of the tree can be managed by a different authority. In this way, the database containing the name system information is hierarchically divided into non-overlapping zones. Divisions are made by *delegation* of a zone by the managing authority of the immediately upwards zone. The data base corresponding to a zone is managed by the zone's *authoritative name servers*.

The data base of the DNS is formed by the so called *resource records*. These records can be of different types, according to the information they contain. For instance, a record of class internet and type AAAA maps a name into an IPv6 address. In general, a resource record contains 5 fields, namely *owner (domain name), Time to live, Class, Type, rdata (value).*

Question 3.1.

Within the Internet class, there are several types of records. Investigate the purpose of each type of record and complete Table 1. Write down below the name of further records, and their purpose, if you know more of them.

We have provided here a very brief description of the DNS. Students should refer to other sources such as [8, 9] to further understand this key component of the Internet.

Type	Purpose	Example domain name	Example value
A	Maps a domain name to an IPv4	www.imt-atlantique.fr	192.108.117.237
AAAA			
NS			
MX			
CNAME			
PTR			
SOA			
RRSIG			

Table 1: Common DNS resource records in the Internet class.

3.1.2. The DNS Main Actors

In order to better understand the functioning of the DNS is important to understand -and distinguish- the following terms:

- **Authoritative name server** a server managing a zone, containing the records corresponding to that specific zone.
- **Resolver** server that is queried by a client (for instance your host) and which performs the DNS queries in order to resolve a mapping. DNS queries are addressed to authoritative name servers. Resolvers usually store records into their cache.
- **Resource Record** constitute the DNS database, see description in the previous subsection.
- **Root servers** authoritative name servers maintaining records of the top-level domains.

3.1.3. DNS Security Extensions (DNSSEC)

DNSSEC is a set of extensions that makes it possible for resolvers to validate the **authenticity** and **integrity** of DNS data. The original design of DNS did not consider security aspects, and DNSSEC has been an important effort aiming to include data authentication and integrity. This set of extensions do not provide confidentiality neither availability, though. DNSSEC prevents users to get fake or manipulated data, that could be created by attacks against DNS resolvers such as cache poisoning (See readings above). DNSSEC is mainly defined by three RFCs: [4], [5], and [6].

3.2. The Packet Analyser Wireshark

Wireshark [10] is an opensource network packet analyser. It allows you to capture network packets and to display the packets data in a detailed user friendly way. “You could think of a network packet analyzer as a

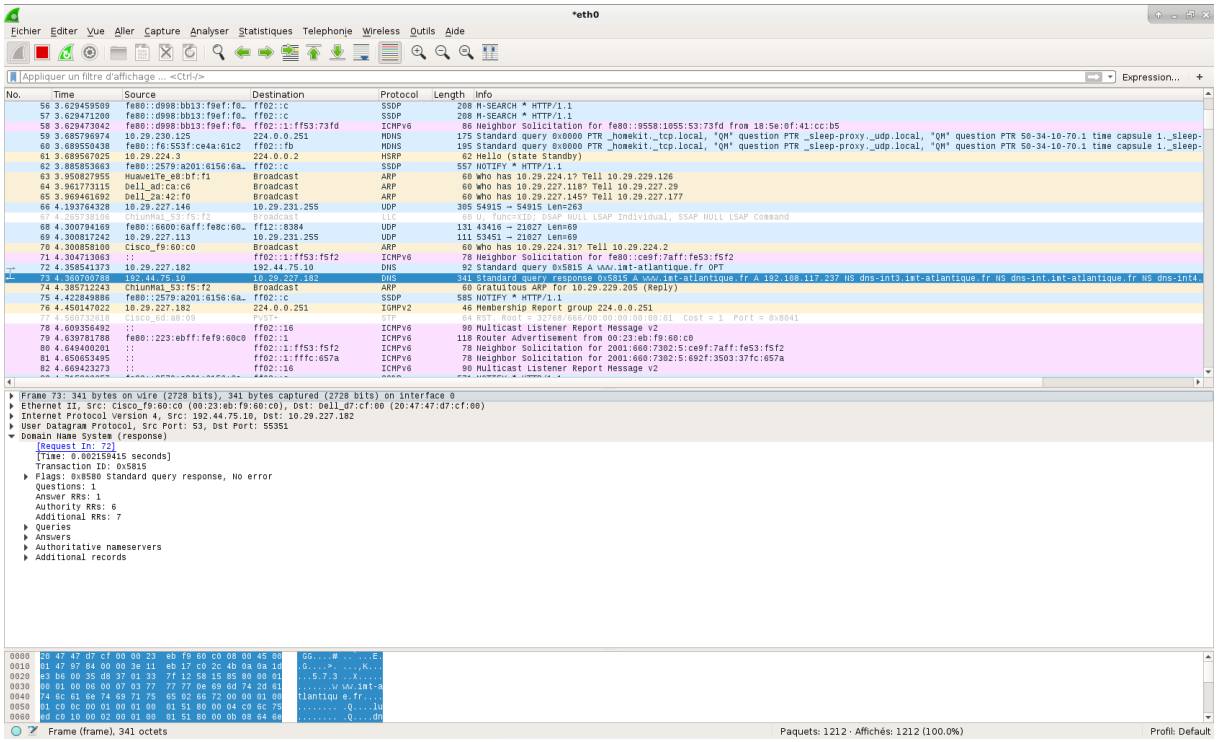


Figure 2: A wireshark packet capture example source [10]

measuring device used to examine what's going on inside a network cable, just like a voltmeter is used by an electrician to examine what's going on inside an electric cable (but at a higher level, of course)." [10]

Figure 2 shows an example wireshark capture. We can distinguish the sequence of captured packets, followed by the detailed, human-readable content of the selected packet, finally followed by the packet data in hexadecimal.

3.3. Capturing traffic

In order to capture traffic in a wired interface in a unix system using wireshark make sure to have the adequate privileges (e.g. on Debian and based distributions, the user should be part of the *wireguard* group).

```
wireshark &
```

Once in wireshark, before starting a capture, you must select the interfaces where you want to capture traffic. To do so, click `Capture` → `Options` → select interface `start`.

At any moment you can save one capture to analyse it later on. You can also change the visualisation options, to get more or less columns, activate/desactivates colors, etc.

4. Hands On

4.1. Some useful unix commands.

1) Start the course's VM and open a new terminal in the VM

Question 4.1.

Use the command `ip a` (shortcut for `ip address`) to obtain the information of the network interfaces available in your working VM. Are they IP addresses v4 or v6? Write down the obtained address(es).

Question 4.2.
Which DNS resolver is your working VM using? For obtaining that information show the content of file /etc/resolv.conf. You can for example execute `cat /etc/resolv.conf`

Note that the information from the two previous questions will be useful when you will inspect network packets in Section 3.2.

4.2. DNS basic functioning and commands

Question 4.3.
Use the command `dig -x` to query the domain name associated to the 2620:0:2d0:200::7 address, and write it down. What kind of query (lookup) is it? (Clue: look at `man dig`).

Question 4.4.
Indicate the IPv4 address associated to www.brest.fr. You can again use command `dig` again.

Question 4.5.
Which type of DNS record allows you to obtain such information?

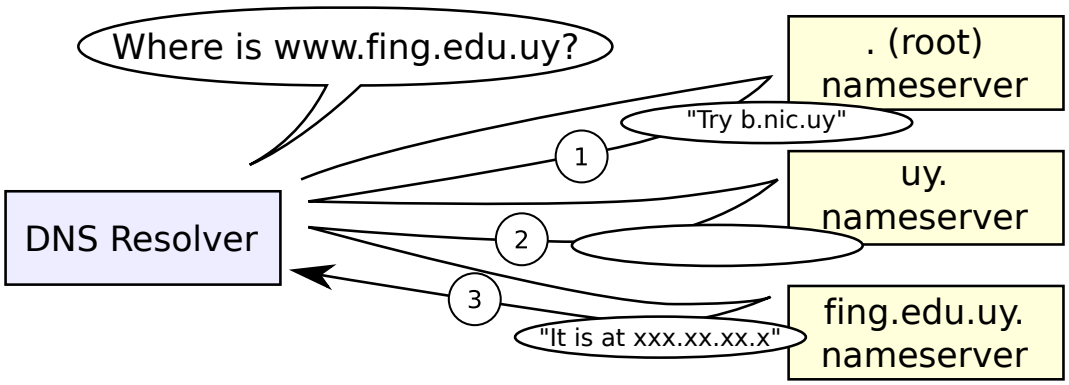


Figure 3: DNS resolution of www.fing.edu.uy. Image based on https://en.wikipedia.org/wiki/File:Example_of_an_iterative_DNS_resolver.svg

Question 4.6.

Use `dig` to obtain the IPv6 address associated to www.nic.cl. Write down the address and the full command you need to use.

Question 4.7.

Query now the A record of www.fing.edu.uy showing all the hierarchy. You can use `dig` with the `+trace` option (along with `+multiline` for a more human-readable output). Who send the messages numbered *1*, *2* and *3* from Figure 3 (Look at the "Received . . . from" lines in the output)? Complete the answers from the `uy.` and `fing.edu.uy` nameservers. What can you conclude about the functioning of the domain service? Explain.

4.3. Analysing DNS traces

We are now going to use Wireshark to see what is going on down the wire.

1) Open a terminal on your virtual machine and type the command `wireshark` to open wireshark. In wireshark, start a new capture in the `eth0` interface.

2) In another terminal, issue a one-time ping (`ping -c 1`) to www.imt-atlantique.fr

3) In Wireshark examine the captured packets.

Question 4.8.
What protocols do you see? Why?

4) Use a visualisation filter to see only the DNS messages.

5) Locate the DNS query and response sent and received due to your previous command.

Question 4.9.
Are them sent over UDP or TCP? What is the destination port for the DNS query message? What is the source port of DNS response message?

Question 4.10.
To what IP address is the DNS query message sent? Compare this to the resolver used by your working VM (see your answers to part 4.1). Are these two IP addresses the same?

6) Now, we will find out the serves managing e-mail for imt-atlantique.fr (you can refer to Table 1). In a terminal use the `dig MX` program to find out the servers exchanging mails for imt-atlantique.fr

Question 4.11.
What information did you get?

7) Observe the captured DNS traces

Question 4.12.
Examine the DNS query message. What "Type" of DNS records does the query ask for? Does the query message contain any "answers"?

Question 4.13.
Examine the DNS response message. What name servers does the response message provide? Does this response message also provide the IP addresses of the name servers?

8) Find out the authoritative name server of the tools.ietf.org domain.

9) Start a new capture, if you have stopped the previous one

10) In a terminal, run the command `dig tools.ietf.org @ip_authoritative server` where `ip_authoritative server` is one of the IPs found out in step **8**)

11) Repeat the command several times.

12) Now, type command `dig tools.ietf.org @80.67.188.188` Repeat the action several times.

13) You can now stop the capture

Question 4.14.
Observe the Wireshark capture. Compare the responses obtained from the authoritative and the recursive servers. In what do they differ? Look at the TTLs. Explain.

4.4. Validating DNS query answers with DNSSEC

For taking advantage of the security provided by DNSSEC, two conditions have to be fulfilled: (1) the domain zone that you want to query about has to be signed, and (2) your DNS resolver must be able to verify the record signatures.

Before answering the questions in this section, check first whether your DNS resolver validates or not the DNS answers. For that you can use two tools:

- The dnssec-failed.org domain, whose **signature is invalid**. Run `dig dnssec-failed.org`, and you should get a **SERVFAIL** message if the resolver validates the answer, or a **NOERROR** message with the full answer if it does not.
- The http://dnssec.vs.uni-due.de/ website: provides an easy to use "DNSSEC resolver test". Simply click on the `Start test` button.

If you find out the resolver is not DNSSEC-able, compare the query for the *dnssec-failed.org* domain against a validating resolver:

14. Query a validating server, such as `ns0.ldn-fai.net` or `1.1.1.1`, about the domain dnssec-failed.org:
    `dig dnssec-failed.org @1.1.1.1`.

Question 4.15.
What do you get? Explain.

15. Query now a non-signed domain and a signed domain:
    `dig +dnssec hola.com @1.1.1.1`
    `dig +dnssec ripe.net @1.1.1.1`

Question 4.16.
The difference is not very visible for the end user, but compare the **flags** in the **HEADERs** of both answers. What is (or should be) the difference between them (how the answer tells you the answer has **Authenticated Data (AD)** or not)?

16. Query again `ripe.net` using the `dig` command, including the +dnssec and +multiline options.

Question 4.17.
Explain what is the new record and its contents that you can read in the answer.

5. Conclusion

Question 5.1.
What is the purpose of the DNS protocol? Give an answer as complete as possible (which can be brief at the same time).

Question 5.2.
Why does the DNS protocol uses UDP as a transport protocol? What do you think are the advantages and disadvantages for this?

Question 5.3.
What are advantages and disadvantages of a distributed data base, as the one managing domain names?

Question 5.4.
Would you say that DNS is a secure protocol or not? Be as precise as you can in answer. What is DNSSEC for and what does it provide? Does DNSSEC provides confidentiality in the DNS messages exchange?

References

[1] RFC 1034 - DOMAIN NAMES - CONCEPTS AND FACILITIES, The Internet Engineering Task Force, [online] https://tools.ietf.org/html/rfc1034.

[2] RFC 1035 - DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION, The Internet Engineering Task Force, [online] https://tools.ietf.org/html/rfc1035.

[3] RFC 2181 - Clarifications to the DNS Specification, The Internet Engineering Task Force, [online] https://tools.ietf.org/html/rfc2181.

[4] RFC 4033 - DNS Security Introduction and Requirements, The Internet Engineering Task Force, [online] https://tools.ietf.org/html/rfc4033.

[5] RFC 4034 - Resource Records for the DNS Security Extensions, The Internet Engineering Task Force, [online] https://tools.ietf.org/html/rfc4033.

[6] RFC 4035 - Protocol Modifications for the DNS Security Extensions, The Internet Engineering Task Force, [online] https://tools.ietf.org/html/rfc4033.

[7] The Internet Engineering Task Force [online] https://ietf.org/about/.

[8] Computer Networks, 5th edition, Andrew S. Tanenbaum, David J? Wetherall, PRENTICE HALL

[9] Les réseaux, 4th edition, Guy Pujolle, EYROLLES

[10] Wireshark network packet analyser [online] https://www.wireshark.org/

Cybersécurité des systèmes maritimes et portuaires
The network layer principles

Isabel Amigo

2021

Notes:

Today's (and tomorrow) objectives

First pass

- understand the main objectives and challenges of network layer
- understand and distinguish key network functions: routing and forwarding
- get the feel of routing algorithms
- get the main characteristics of datagrams and virtual circuits

Second pass

- understand addressing in the Internet: IPv4 and IPv6
- in depth look to routing and forwarding:
  - in-class example
  - lab interdomain routing: OSPF

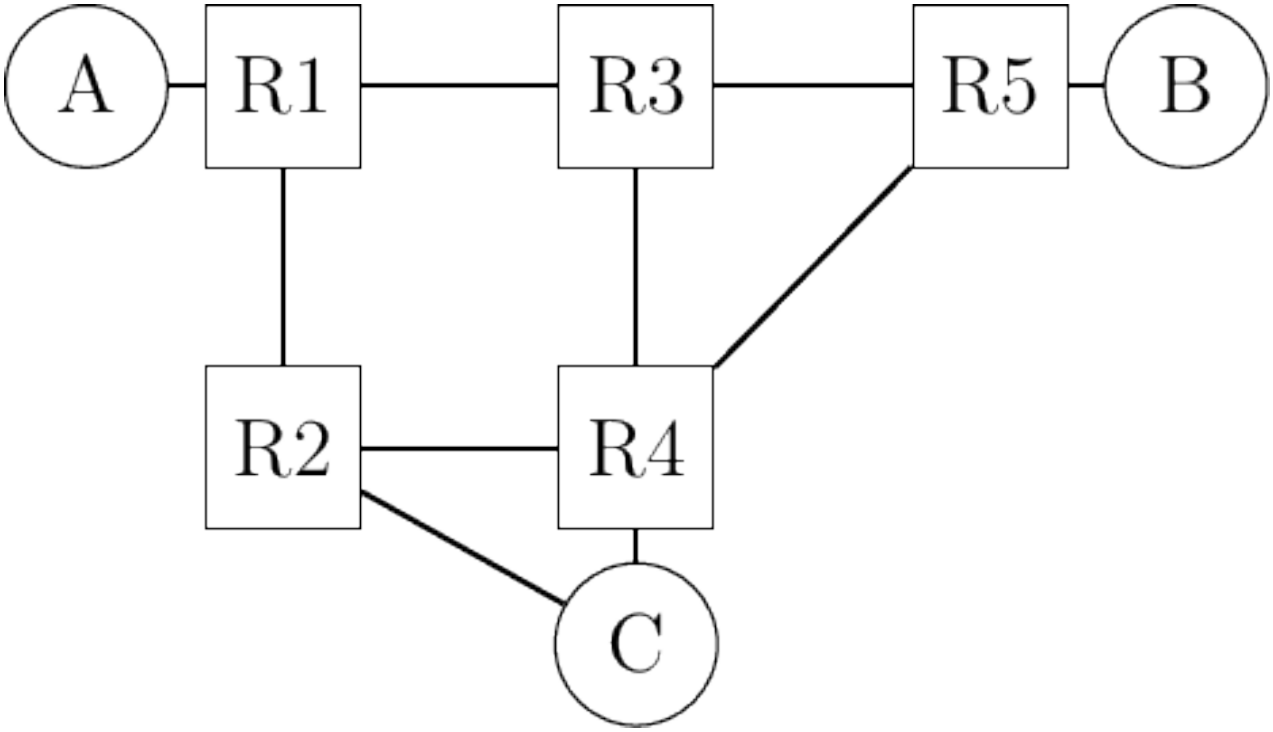
Notes:

1. Main objectives of Network layer

Notes:

Network layer enables transmission of information between hosts not directly connected

Example



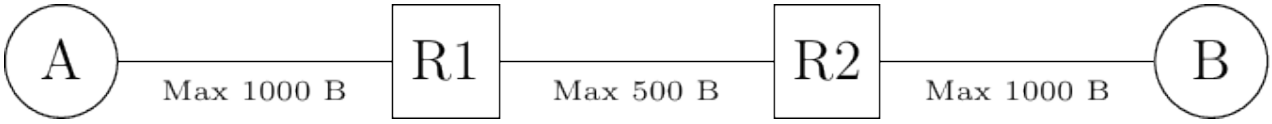
Notes: Main objective of the NL: allow hosts, connected to different networks, to exchange information through intermediate systems (routers).
Packet: unit of information in the network layer
Example:

- We consider a simple network, and imagine we want to transfer a byte from A to B (so we do not have MTU problems)
- We also consider single-homed hosts in order to simplify
- A sends the packet to R1, but R1 needs to know that the packet is not for it but for B!
- Each interface has to be uniquely identified $\Rightarrow$ address usually fixed-length addresse represented by a sequence of bits
- To send one byte of information to host B, host A needs to place this information inside a packet.
- Packet has data + control information: either a) the addresses of the source and the destination nodes or b) information of path

Network layer is also responsible for dealing with heterogeneous datalink layers

Example

A wants to send a 900 bytes packet (870 bytes of payload and 30 bytes of header) to host B via router R1. Host A encapsulates this packet inside a single frame. The frame is received by router R1 which extracts the packet. What happens next?



Notes: Hosts connected to different DL exchange packets via routers that are using other types of DL. Thanks to the NL, this exchange of packets is possible provided that each packet can be placed inside a DL frame. This is simple if all DLs support the same frame size. But more complex if this is not the case, since the packet can not be encapsulated in a frame. Consider the example. R1 has three possible options to process this packet.

1. rejects the packet and sends a control packet back to the source (host A) to indicate that it cannot forward packets longer than 500 bytes (minus the packet header) (plus action possible by source)

- fragment the packet into two parts. possibilities with different pros and cons.

2. R1 fragments the packet before transmitting them to R2. R2 reassembles the two packet fragments in a larger packet before transmitting them on the link towards B.
3. Each of the packet fragments is a valid packet that contains a header with the source (host A) and destination (host B) addresses. When R2 receives a packet fragment, it treats this

Two Possible organizations of the Network Layer

- Datagrams (packet-switched networks)
  - No 'call' set-up, each packet is independently forwarded
  - No resource reservation
- Virtual circuits
  - 'Call' set-up, a circuit is established before data transfer
  - Typically allows resource reservation

! Concern in previous slide (fragmentation) typically in datagram mode. Why?

Notes: Datagrams, inspired by the organization of the postal service. Each host is identified by a network layer address. To send information to a remote host, a host creates a packet that contains:

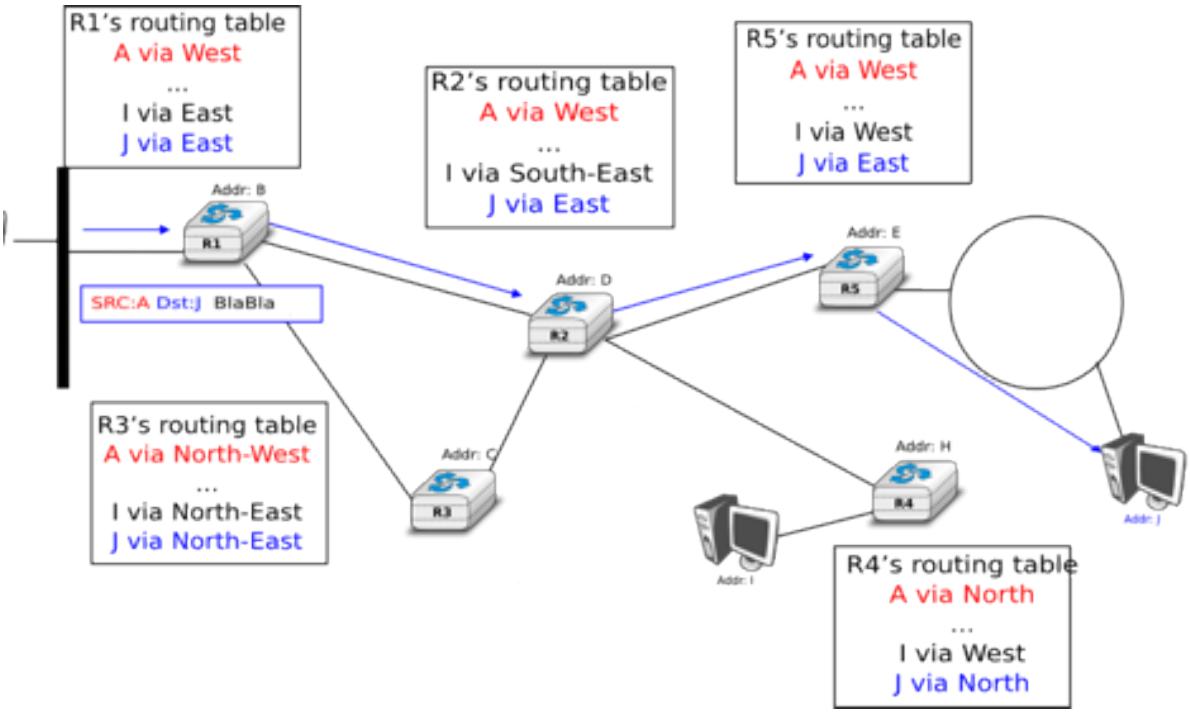
- the network layer address of the destination host
- its own network layer address
- the information to be sent

A datagram is the unit of information of packet-switched networks. Term defined in RFC1594 as"“A self-contained, independent entity of data carrying sufficient information to be routed from the source to the destination computer without reliance on earlier exchanges between this source and destination computer and the transporting network.” ”

2. Datagrams

Notes:

Hop-by-hop Forwarding: *using* forwarding tables to send packets



Notes:

- Routers use hop-by-hop forwarding.

  When a router receives a packet that is not destined to itself, it looks up the destination address of the packet in its forwarding table.

- Forwarding table: data structure that maps each destination address (or set of destination addresses) to the outgoing interface over which a packet destined to this address must be forwarded to reach its final destination.

- Router consults its forwarding table to forward each packet that it handles.

- Another similar concept is the *routing table*, some protocols like OSPF make a difference on routing table (all routes known and related information like metrics) and forwarding table (the actual table being used for forwarding). We'll see this with routing protocols.

Routing: *computing* forwarding tables

Different possible techniques, we shall focus on

- Manually
- Topology information exchange + algorithm

Computing correctly the forwarding tables is key aspect.

Q What could happen if forwarding tables accross routers are not consistent?

Notes:

- Key element for network operation: computation of forwarding tables
- Its done using either distributed or centralized algorithms
- Each algo. provides different performance, may lead to different types of paths
- In a network that has valid forwarding tables, all the paths between all source/destination pairs contain a finite number of intermediate routers.
- If forwarding tables have not been correctly computed, two types of invalid paths can occur :
  - black hole : a router that receives packets for at least one given source/destination pair but does not have an entry inside its forwarding table for this destination. Packet is discarded.
  - loops : as before the destination is not reachable from all

Data plane and control plane

Network functions are typically separated in what we call the *control plane* and the *data plane*.

Control plane

- e.g. all the protocols and algorithms that compute the forwarding tables that run on routers
- simplest control plane for a network: to manually compute the forwarding tables

Data plane

- e.g. forwarding tables and the precise format of the packets that are exchanged

---

Notes:

- The forwarding tables and the precise format of the packets that are exchanged inside the network are part of the data plane of the network. This data plane contains all the protocols and algorithms that are used by hosts and routers to create and process the packets that contain user data.
- The control plane includes all the protocols and algorithms (often distributed) that compute the forwarding tables
- While there is only one possible data plane for a given networking technology, different networks using the same technology may use different control planes.

Flat vs hierarchical addresses

Flat addressing

- ☺ approach: unique address pre-configured in network interface card
- ☺ easy lookup operation in the forwarding table (exact match)
- ☹ forwarding tables grow linearly with the number of hosts and nodes

Hierarchical addressing (analogy mail system)

- ☺ allows to significantly reduce the size of the forwarding tables
- ☹ lookup in the forwarding table is more complex
- ☹ not possible to use a permanent, pre-configured address
  Q how to obtain self address when node comes up?
- 😐 the allocation of the addresses must follow the network topology ⇒ blocks

Q Which scheme do you think is used in the Internet?

Notes: Further comments:

- Flat addressing: upon packet arrival, the node only needs to check whether this address is included in the forwarding table or not (computationally easy)
- Hierarchical addressing scheme : drawback when a host connects for the first time to a network, it must contact one network node to determine its own address ⇒ packet exchanges between the host and some network nodes. Furthermore, if a host moves and is attached to another network node, its network address will change. This can be an issue with some mobile hosts.

Routing algorithms allow to compute forwarding tables

Different flavors exist:

Distance vector algorithms: relay on protocols to exchange information for running a distributed algorithm

Link state algorithms: relay on protocols to learn network topology.

Notes:

Distance Vector routing algorithms

- use a distributed algorithm to discover shortest routes towards all destinations
- main idea: regularly each router sends *routing table to their neighbours* (distance towards each known destination)
- some extra rules to reduce problems such as 'count to infinity'
- upon convergence each router has a routing table containing for each destination the next hop and a cost
- example in the Internet: BGP

Notes:

- simple distributed routing protocol. allows routers to automatically discover reachable destinations and shortest path to them
- The shortest path is computed based on metrics or costs that are associated to each link.
- Each router maintains a routing table. The routing table R can be modeled as a data structure that stores, for each known destination address d, the following attributes : the outgoing link that the router uses to forward packets towards destination d, the sum of the metrics of the links that compose the shortest path to reach destination d, a timestamp
- router regularly sends its distance vector over all its interfaces (summary of the router's routing table that indicates the distance towards each known destination.)
- Split-horizon: only contains the routes that have not been learned via this neighbor.

Link State routing algorithms

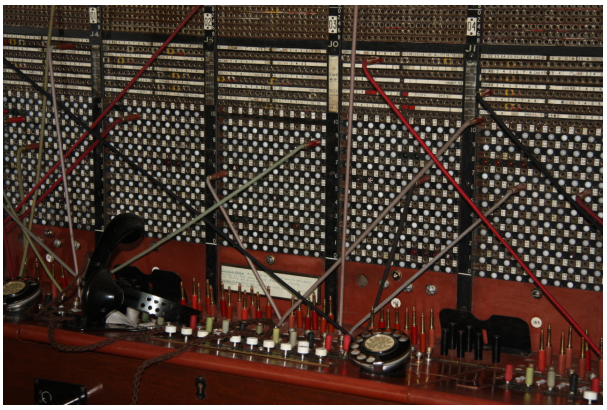
- routers exchange topology information
- regularly each router sends *the information of directly connected networks to everyone (flooding)*
- upon convergence, each router has a representation of the whole topology, and runs a shortest path algorithm (e.g. Dijkstra) to compute the routing table
- example in the Internet: OSPF
- more on Lab OSPF!

Notes:

- link-state routers exchange messages to allow each router to learn the entire network topology.
- Based on this learned topology, each router is able to compute its routing table by using a shortest path computation such as Dijkstra's
- network is modeled as a directed weighted graph. Each router is a node, and the links between routers are the edges in the graph. A positive weight is associated to each directed edge and routers use the shortest path to reach each destination.

3. Virtual Circuits

Notes:



Notes:
Telephonists (from Milton Keynes Telephone Museum) and switchboard from the manual telephone exchange at Enfield (north London). Taken out of service and replaced by automatic equipment in 1960. Now on display in the Science Museum in London.

Virtual circuits

- call setup, teardown for each call before data can flow
- each packet carries VC identifier (not destination host address)
- every router on source-dest path maintains "state" for each passing connection
- link, router resources (bandwidth, buffers) may be allocated to VC (dedicated resources = predictable service)

Notes:

What about forwarding and routing?

- hosts identified with an address
- packet forwarding based on a label on packet's header (and not on global address) and on label switching tables present at each intermediate node
- need of a signaling protocol to set-up path

Notes:

- In a network using virtual circuits, all hosts are also identified with a network layer address. However, packet forwarding is not performed by looking at the destination address of each packet.
- Each data packet contains one label: an integer which is part of the packet header.
- Network nodes implement label switching to forward labelled data packet.
- Upon reception of a packet, a network nodes consults its label forwarding table to find the outgoing interface for this packet. In contrast with the datagram mode, this lookup is very simple

Is packet-switching *better* than circuit-switching?

☺Packet-switching is great for bursty data

- resource sharing
- simpler, no call setup

☹No performance guarantees, excessive congestion possible: packet delay and loss

- protocols needed for reliable data transfer and congestion control

Q Examples of applications generating bursty and non-bursty data?

Q How to provide bandwidth guarantees, needed for some applications, on packet-switching networks?

- several "patches" exist, but till a research problem!

Notes:

Summary

Principal objectives of the network layer

- Transfer information between distant hosts, connected through routers
- Deal with heterogeneous datalink layers

For achieving such purposes:

- Addresses
- Packets

Types of network layers

- Datagrams
- Virtual circuits

Principal network functions

- Forwarding
- Routing

Notes:

Announcements

Prepare your lab!

- Introduction to Mininet $\Rightarrow$ to do at home before coming to lab
- Lab OSPF $\Rightarrow$ to prepare at home before coming to lab

---

Notes:

Acknowledgements

The contents of these slides is mostly based on the e-book Computer networking: principles and protocols
`http://beta.computer-networking.info/syllabus/default/principles/network.html`

Notes:

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

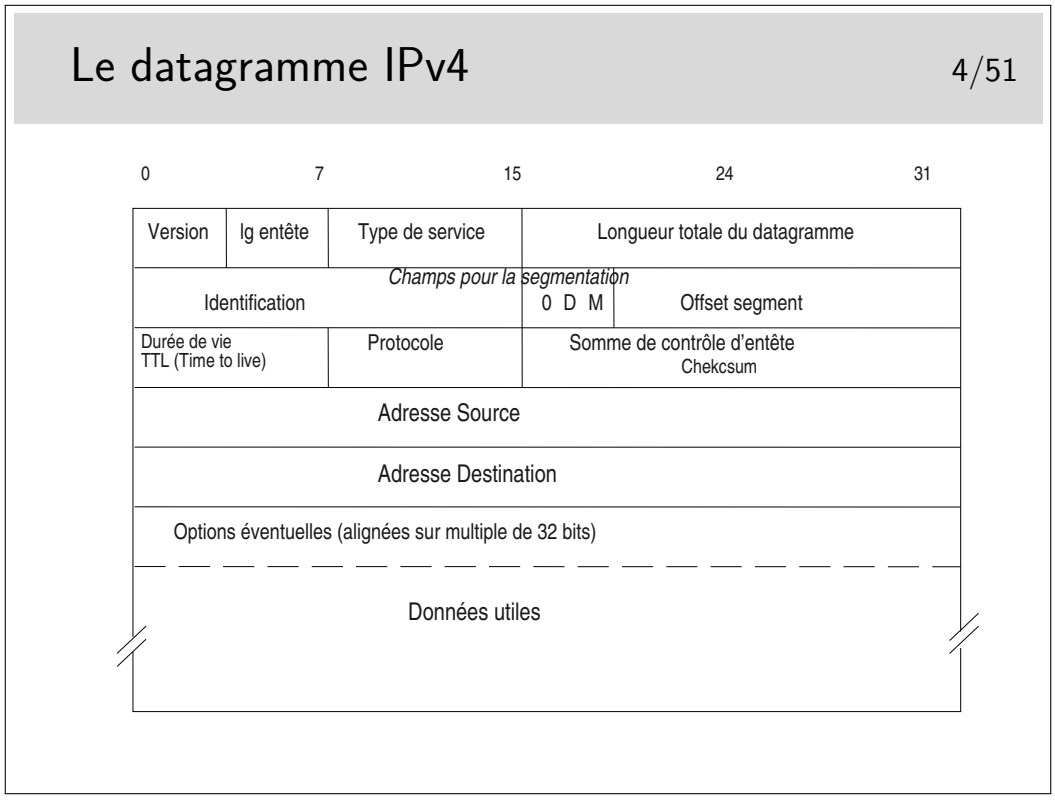
Cybersécurité des systèmes maritimes et portuaires

La couche réseau

2021

1 Le datagramme IP

1.1 Le datagramme IPv4



Longueur d'entête : en nombre de mots de 32 bits ; si supérieur à 5, indique la présence d'options (40 octets max d'options : 10 mots max de 4 octets)

Type de service : peu utilisé par le passé, permet aujourd'hui, entre autres, de coder le DSCP (DiffServ Code Point). Voir pages suivantes.

Champs pour la segmentation : on recommande d'éviter la segmentation avec TCP. Elle n'existe plus que pour UDP si les unités de données fournies ont une taille supérieure au MTU de l'interface

bit D : *Don't fragment* : interdit la fragmentation du segment s'il est à 1

bit M : *More* : indique la présence de fragments complémentaires, si le segment d'origine a été fragmenté.

Si le champ Offset est à 0 ainsi que le bit M, le segment est celui d'origine

TTL : durée de vie ; décrémenté de 1 par chaque routeur. Si le résulat passe à 0, le datagramme est jeté et un message ICMP est envoyé à la machine émettrice

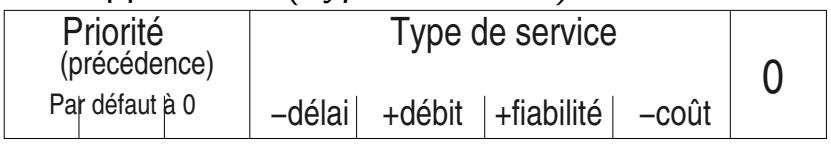
Protocole : indique le protocole supérieur véhiculé (TCP, UDP, ICMP voire même IP en cas d'encapsulation IP dans IP piour tunnelling)

Somme de contrôle d'entête : contient la somme des mots de 16 bits constituant l'entête. Permet à l'arrivée de vérifier l'intégrité de celle-ci. Si une erreur est détectée le datagramme est jeté sans autre forme d'action. On ne peut même pas envoyer un message d'erreur à l'émetteur car l'adresse source qui indique ce dernier peut être corrompue. Comme l'indique le nom de ce champ le contrôle d'intégrité n'est fait que sur l'entête.

Adresses : source et destination, sur 32 bits ; information fondamentale, au moins en ce qui concerne la destination car elle sert au routage.

Le champ Type de Service 5/51

- Encore appelé ToS (*Type of Service) rfc1349*



- Utilisation non généralisée
  - Précédence utilisable pour marquer des flux dans les nœuds du réseau (routeurs)
  - Les bits Type de Service peuvent être positionnés par les applications terminales (API socket sous Windows et linux)
  - Sous Linux ils peuvent être positionnés via la commande iptables selon divers critères

111 - Network Control (protocole de type HELLO)

110 - Internetwork Control (protocoles de routage)

101 - CRITIC/ECP

100 - Flash Override

011 - Flash

010 - Immediate

001 - Priority

000 – Routine (priorité la plus faible)

RFC 791 et 795

Le champ Type de Service en version DiffServ 6/51

- Le champ DSCP sert à coder le *PHB (Per Hop Behavior), paramètre fondamental de DiffServ (rfc 2474)*
- Exemples de PHBs :
  - Expedited forwarding (pertes faibles, latence faible, gigue faible, bande passante assurée)
  - Assured forwarding (groupe de PHBs)
  - Best effort
  - Network control

DiffServ signifie Différentiation de Service (en anglais aussi). C'est un mécanisme relativement récent permettant de marquer les paquets afin qu'ils soient traités de manière différentiée dans les routeurs. C'est un mécanisme de qualité de service.

Les options IPv4 7/51

- Format :

- Les options :
  - LSR : *Loose Source Route* : permet d'indiquer la route
  - SSR : *Strict Source Route*, comme précédemment en plus rigoureux
  - RR : *Record Route* : les routeurs traversés rajoutent leur adresse
  - RTALT : *Router Alert*, permet de passer le paquets aux couches hautes des routeurs traversés
  - etc.

L'option LSR permet d'indiquer la route à suivre mais si un routeur intermédiaire ne sait pas comment utiliser une des indications contenue dans l'option, il peut alors utiliser sa table de routage normale.

L'option SSR est plus stricte car le paquet est rejeté si un routeur ne sait pas l'orienter vers une direction indiquée.

Il y a encore l'option Traceroute, obsolète car dangereuse, EEOL et NOP pour indiquer la

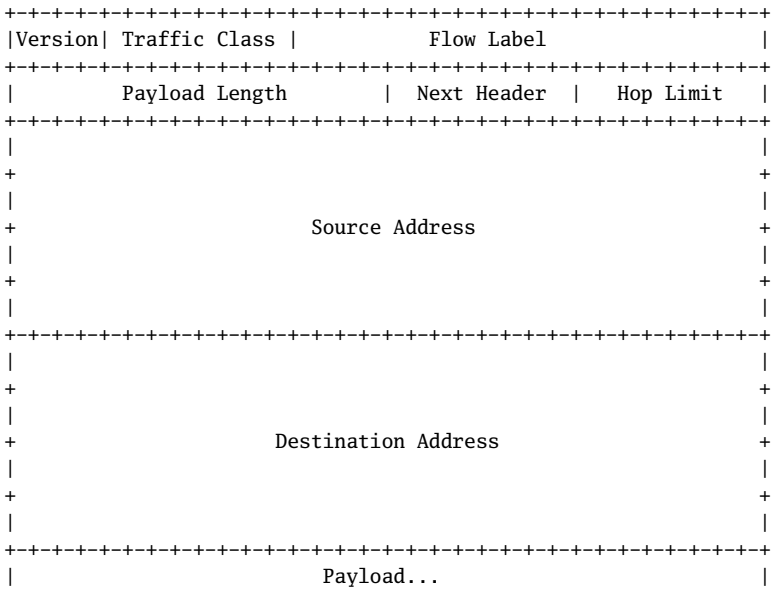
fin de la liste et du bourrage d'alignement sur 32 bits.

L'option RTALT est utilisée par IGMP et par RSVP, ces protocoles nécessitent des traitements particuliers dans les routeurs. Ces traitements sont effectués par des entités applicatives des routeurs. Ces derniers ne doivent donc pas effectuer le routage rapide des paquets munis de cette options mais ils doivent les diriger vers les entités applicatives.

`http://www.iana.org/assignments/ip-parameters`

1.2 Le datagramme IPv6

Le datagramme IPv6 – RFC 8200 I 9/51

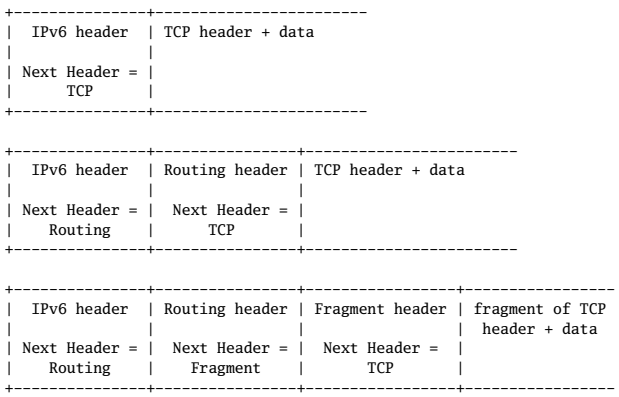


Le datagramme IPv6 II 10/51

- Version : 6 (!)
- Traffic Class : typiquement Differentiated Services et Explicit Congestion Notification, comme le TOS IPv4
- Flow Label : pour tagger les paquets d'un même flux ; visiblement peu utilisé...
- Payload Length : comme son nom l'indique... (Note : dans IPv4 on a la taille *totale* du datagramme)
- Next Header : remplace le champ Protocole IPv4 et ajoute un mécanisme un peu novateur pour gérer les options (cf. slides suivants)
- Hop Limit : l'équivalent du champ TTL IPv4 (Note : contrairement à IPv4 qui compte en secondes, ici c'est bien en nombre de sauts)
- Source & Destination Addresses : sur 128 bits (Ouf !)

Next Header 11/51

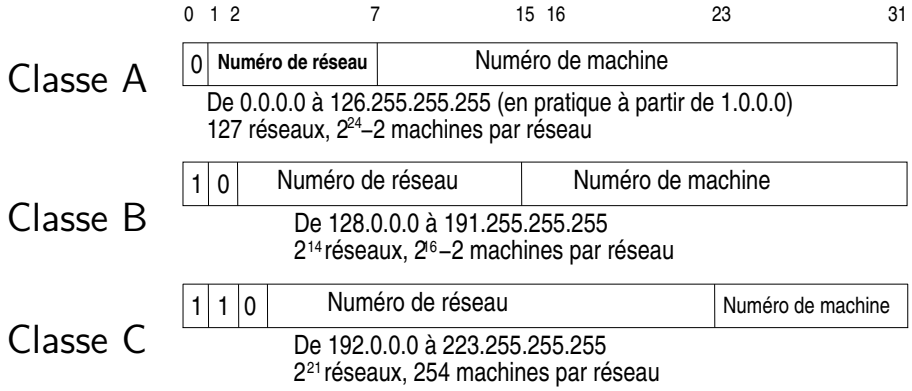
- Les headers s'enchaînent dans le payload
- ...les options IPv6 diverses, puis le protocole transporté effectivement (TCP, UDP, ...)



2 Adressage

2.1 Les adresses IPv4

Les classes d'adresses IPv4 I 14/51

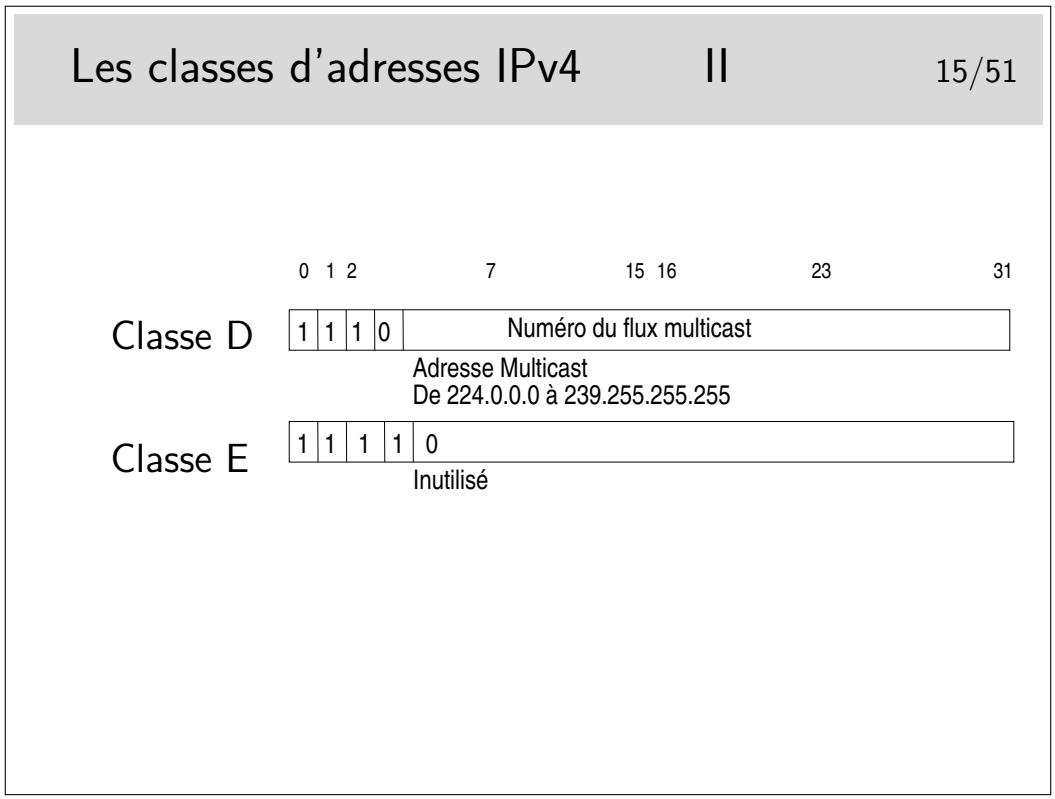


Une interface IP doit être munie d'une adresse pour pouvoir fonctionner. En général, pour des machines de bureau standard en réseau local, l'interface IP correspond à une carte réseau de type Ethernet.

L'interface IP peut aussi être correspondre à une couche complètement logicielle, c'est le cas des interfaces de niveau 2 réalisées avec le protocole PPP qui s'interpose entre IP et une interface physique comme un port série ou un port USB relié à un modem ADSL (l'empilement des couches protocolaires est alors très complexe).

Une interface matérielle peut être munie de plusieurs adresses IP, sous Unix et Linux en particulier. On a alors, par exemple, les interfaces eth0 :0, eth0 :1, eth0 :2, etc.

A noter que les adresses de classe A de la gamme 0.0.0.0 à 0.255.255.255 ne peuvent pas être utilisées, bien qu'elles ne correspondent pas à une fonction particulière.



Le multicast...

Permet d'envoyer des paquets IP à un certain nombre de machines mais pas à toutes. Des machines sur le réseau peuvent être sources de trafic multicast, elles peuvent émette par exemple à l'adresse 224.5.6.7. Il suffit que d'autres machines soient «au courant» de ce fait pour se mettre en écoute en lançant une application spécifique. Cette application demande explicitement à écouter le flux en envoyant un message du protocole IGMP (RFC-3376 pour la version 3) vers l'Internet. Ce message est relayé par les routeurs vers les sources, chaque routeur rencontré s'insère alors dans un arbre de diffusion multicast (préalablement configuré). Les paquets du flux applicatif peuvent arriver ainsi aux récepteurs ayant fait la requête.

Il faut que les routeurs soient configurés, ce n'est pas automatique.

Il existe un réseau mondial multicast, appelé le MBONE. Les sources potentielles diffusent des annonces de session à l'adresse SAP.MCAST.NET (224.2.127.254) (en UDP, port 9875).

L'outil sdr (pour unix/linux ou windows) permet d'écouter ce flux et afficher les informations de sessions.

(mot clé MBONE ; un lien : `http://www-itg.lbl.gov/mbone/`)

Les adresses particulières 16/51

- L'adresse de boucle locale : 127.0.0.1
  - Interface lo sous Linux
- Les adresses privées : rfc-1918
  - 10.0.0.0 à 10.255.255.255
  - 172.16.0.0 à 172.16.255.255
  - 192.168.0.0. à 192.168.255.255
  - Non routables dans l'Internet
    - Les machines munies de ces adresses peuvent cependant accéder l'Internet via des passerelles réalisant une fonction de translation d'adresse appelée **NAT** pour ***Network Address Translation***
  - Routables dans les réseaux privés

La fonction de NAT est mise en œuvre dans un routeur muni d'une interface configurée avec une adresse officielle (pouvant donc être routée dans l'Internet). Les paquets sortant sont modifiés, le champ «adresse source» est remplacé par l'adresse officielle de l'interface de sortie du routeur. Ce dernier mémorise l'opération pour effectuer la modification inverse pour les paquets en retour.

Des extensions du concept peuvent affecter aussi les ports TCP ou UDP.

Le mécanisme pose un problème pour les protocoles tels que ftp. En effet, ftp demande la création d'une connexion «entrante» pour réaliser les transferts de fichiers, les paquets de demande de connexion doivent être corrélés avec la connexion sortante existante. Les traitements sont alors plus complexes que pour les connexions simples comme celle pour le Web par exemple). Ces concepts demandent de comprendre ce qu'est une connexion au niveau supérieur (il n'y a pas de connexion IP). Ce point sera abordé plus loin dans le chapitre sur TCP.

Sous linux, les mécanismes de translation d'adresse sont directement intégrés au noyau. Une commande permet d'en paramétrer les caractéristiques, il s'agit de iptables qui peut bien plus encore en ce qui concerne les opérations de filtrage pour la sécurité.

Les adresses particulières pour la diffusion 17/51

Différentes possibilités :

- *Broadcast* sur le réseau local : 255.255.255.255
  - Peu utilisée
- Partie réseau normale, partie machine à 255
  - Généralement l'adresse de broadcast mise en œuvre
    - Exemples :
    - 192.168.100.255
    - 172.16.255.255
  - Partie réseau normale, partie machine à 0
  - Ancienne adresse de broadcast pouvant encore être utilisées sur des machines SUN dont l'OS est SUNOS-4

Les adresses de réseaux 18/51

- Les réseaux avec un netmask standard
  - On indique les 4 octets (entre 0 et 255), comme pour une adresse normale, les derniers octets étant à 0 (le dernier pour une classe C, les deux derniers pour ne classe B, les trois derniers pour ne classe A)
    - Exemple : 192.168.100.0
- Les réseaux «subnettés»
  - On fait figurer tous les octets (entre 0 et 255), y compris les bits de l'extension
    - Exemple : 192.168.100.32 (netmask 255.255.255.224 par exemple)
    - Voir page suivante

Le *subnetting* 19/51

- Extension de la partie «Réseau» de l'adresse en empruntant quelques bits de poids forts de la partie machine
- Par exemple 255.255.255.224 pour une classe C
  - Tout à 1 sauf la partie machine (224d = 1110 0000b)
- Le masque indique quels bits
- Permet de créer des «sous» réseaux
  - Les sous réseaux sont raccordés entre eux via des routeurs, comme des réseaux «normaux»
- Il y a toujours un netmask
  - Il est standard s'il ne comporte pas d'extension de bits par rapport à la classe d'adresses : 255.255.255.0 pour une lasse C par exemple

Pardon pour l'Académie, mais «subnetting» est plus joli que «sous-réseautage»

Avec le masque 255.255.255.224 on prend 3 bits de la partie machine. On pourra créer 8 sous réseaux à partir du numéro 192.168.100.0 :

— `192.168.100.0`
— `192.168.100.32`
— `192.168.100.64`
— `192.168.100.96`
— `192.168.100.128`
— `192.168.100.160`
— `192.168.100.192`
— `192.168.100.224`

Question : à quel sous réseau appartient la machine de numéro `192.168.100.72` ?

Le netmask général et ses notations 20/51

- Les adresses sans classe
  - Concept CIDR (RFC-1519) : *Classless InterDomain Routing*
  - La frontière de l'adresse réseau n'est plus figée selon la loi des classes
  - Utile pour agréger des routes dans les tables de routage des routeurs
  - Utile pour les fournisseurs de service pour affecter un sous ensemble d'adresses à un client...
  - Le netmask doit être précisé avec les adresses
- Notation
  - Classique : 255.255.255.128 (25 bits de masque)
  - Notation CIDR : /25 : exemple 192.168.100.128/25

Dans le dernier exemple, l'adresse donnée est une adresse de réseau et non une adresse de machine. Avec une telle adresse (et son masque) on pourra adresser deux fois 128-2 machines : de 192.168.100.1 à 192.168.100.126 (broadcast 192.168.100.127) et de 192.168.100.129 à 192.168.100.254 (broadcast 192.168.100.255).

La notation classique (255.255...) reste très répandue. La notation CIDR peut se rencontrer sous quelques Unix (BSD, Linux, ? ? ?), elles est plus commode d'utilisation.

Affectation d'adresse à une interface IP 21/51

- «À la main»
  - Selon les outils offerts par le système d'exploitation
    - À l'aide d'interfaces graphiques d'administration
    - Via des commandes spécifiques
- Dynamiquement
  - Via le protocole DHCP (*Dynamic Host Control Protocol*)
    - Un serveur est configuré pour donner l'information
  - Sur connexion via liaison point à point et le protocole PPP (Point to Point Protocol). La machine à configurer contacte un serveur situé à l'autre extrémité de la liaison. Le serveur peut lui fournir son adresse

Le protocole PPP est utilisé par exemple dans les connexions aux fournisseurs de service IP via des modems et le réseaux téléphonique général.

Dans les connexions via ADSL, c'est aussi PPP qui est utilisé pour affecter l'adresse à la

machine qui se connecte mais le mécanisme est beaucoup plus complexe.

Le protocole DHCP permet de fournir une adresse IP, le netmask associé, le routeur par défaut pour l'interface en cours de configuration, le nom de domaine DNS ainsi que le ou les serveurs DNS. Tout ceci pour une période de temps configurée par le gestionnaire du serveur DNS. Les clients doivent renouveler leur demande d'adresse à la fin du temps alloué.

2.2 Les adresses IPv6

Adresses IPv6 – RFC 4291 23/51

- Adresses sur 128 bits
- Notation hexa par bloc de 16 bits
  `2001:db8:cafe:deca:0:0:0:1`
- Compression de zéros
  `2001:db8:cafe:deca::1`

Sous-réseaux 24/51

- Deux parties :
  - **préfixe** ou **identifiant de sous-réseau** (subnet ID) : partie gauche de l'adresse
  - **identifiant de machine** (host ID) : partie droite
- notation CIDR :
  - `2001:db8:cafe:deca:a9e:1ff:fe6b:25c9/64`
  - subnet correspondant:
    `2001:db8:cafe:deca::`

Exemples de préfixes 25/51

- Préfixe de documentation `2001:db8::/32`
- Préfixe link-local `fe80::/10`
- Préfixe multicast `ff02::/10`
- Exemple de préfixe «end-user»
  `2001:db8:fada:ba00:/56`

Exemples d'adresses 26/51

- Unicast
  `2001:db8:cafe:deca:a9e:1ff:fe6b:25c9/64`
- Link-local
  `fe80::a9e:1ff:fe6b:25c9/64`
- Notation IPv4
  `2001:db8::cafe:192.168.0.1` égal à
  `2001:db8::cafe:c0a8:1`
- Localhost `::1`
- Tout les bits à 0 `::`

Affectation d'adresses à une interface IP 27/51

- Il est *normal* d'avoir plusieurs adresses IPv6 à une interface (adresse de portée *lien*, de portée *globale*)
- Configuration «à la main» : plutôt rare
- Dynamiquement
  - Auto-configuration, grâce au préfixe annoncé par le router
  - Via DHCPv6

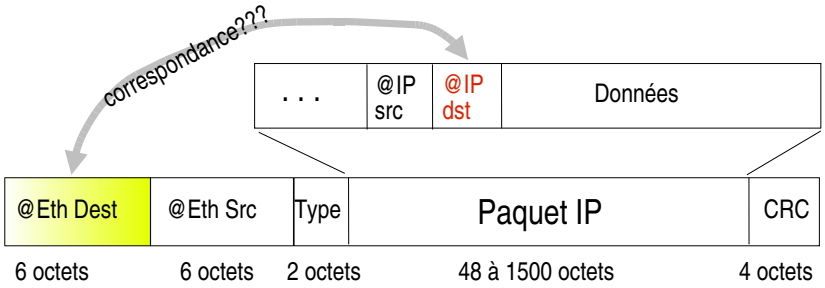
3 Protocole ARP / NDP

Relation entre adresse IP et adresse MAC dans les réseaux locaux 29/51

- Concernent les machines reliées à un réseau local de type Ethernet ou 802.11
- Les adresse IP se gèrent
  - Elles sont affectées «à la main» via des outils spécifiques du système d'exploitation des machines (interfaces graphiques ou commandes telles que ifconfig sous Unix/Linux)
  - Elles peuvent être affectées automatiquement via le protocole DHCP, mais cette possibilité est configurée elle aussi à la main
- Les adresses MAC ne se gèrent pas
  - Elles sont préaffectées par le constructeur de la carte interface que l'on achète ou qui est fournie avec la machine
  - Parfois le pilote (drivers) de la carte permet que cette adresse puisse être modifiée (via ifconfig)

...Adresses MAC et adresses IP... 30/51

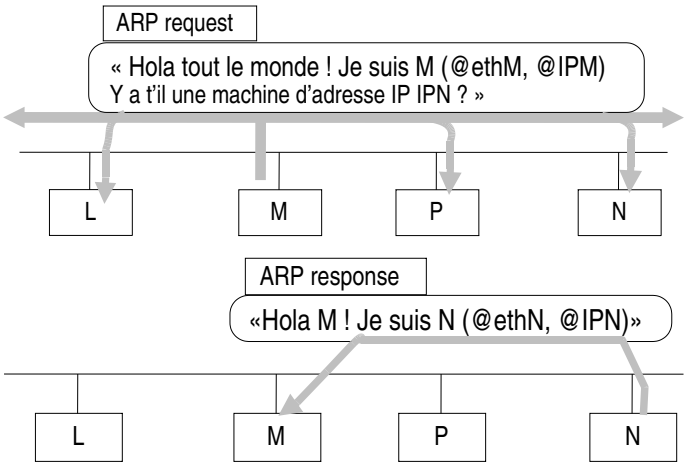
- Problème : une machine M doit émettre un paquet IP vers une machine N voisine dont on ne connaît que le numéro IP (sur le même LAN)
- Si on est sur Ethernet le paquet IP sera véhiculé par une trame telle que celle-ci



Comment la machine M peut elle retrouver l'adresse Ethernet de la machine N en ne connaissant que l'adresse IP de N ?

Adresses MAC et adresses IP : la solution ARP (IPv4) 31/51

- Adress Resolution Protocol - rfc826



Soyez curieux : la commande `arp`

Sur un réseau local, sous Windows ou Unix/Linux, ouvrez une fenêtre de commande et tapez : `arp -a` pour voir la table de résolution arp instantanée. Si vous ne voyez pas la machine de votre voisin et que vous connaissez son adresse IP (ou son nom), faites un «ping» dessus : ping 192.168.100.5 par exemple.

Si la réponse du ping est positive, refaites alors immédiatement `arp -a`, vous verrez apparaître la résolution concernant la machine voisine.

Adresses MAC et adresses IP : Neighbor Discovery Protocol (IPv6) 32/51

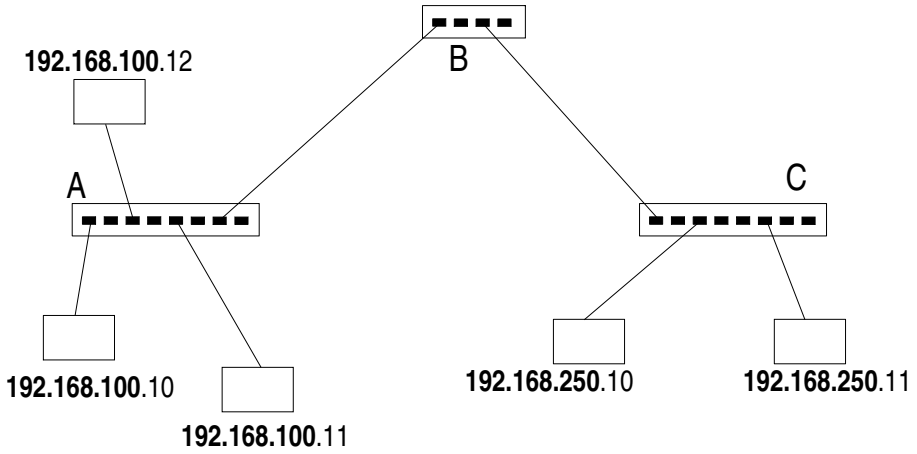
- Neighbor Discovery Protocol (NDP)
- Intégré dans Internet Control Message Protocol version 6 (ICMPv6) – rfc4443
- Fonctionne de manière similaire à ARP (IPv4)

Soyez curieux : la commande `ip neigh`

4 Principe du routage

Question... 34/51

Quelle est la nature des organes A, B et C ?



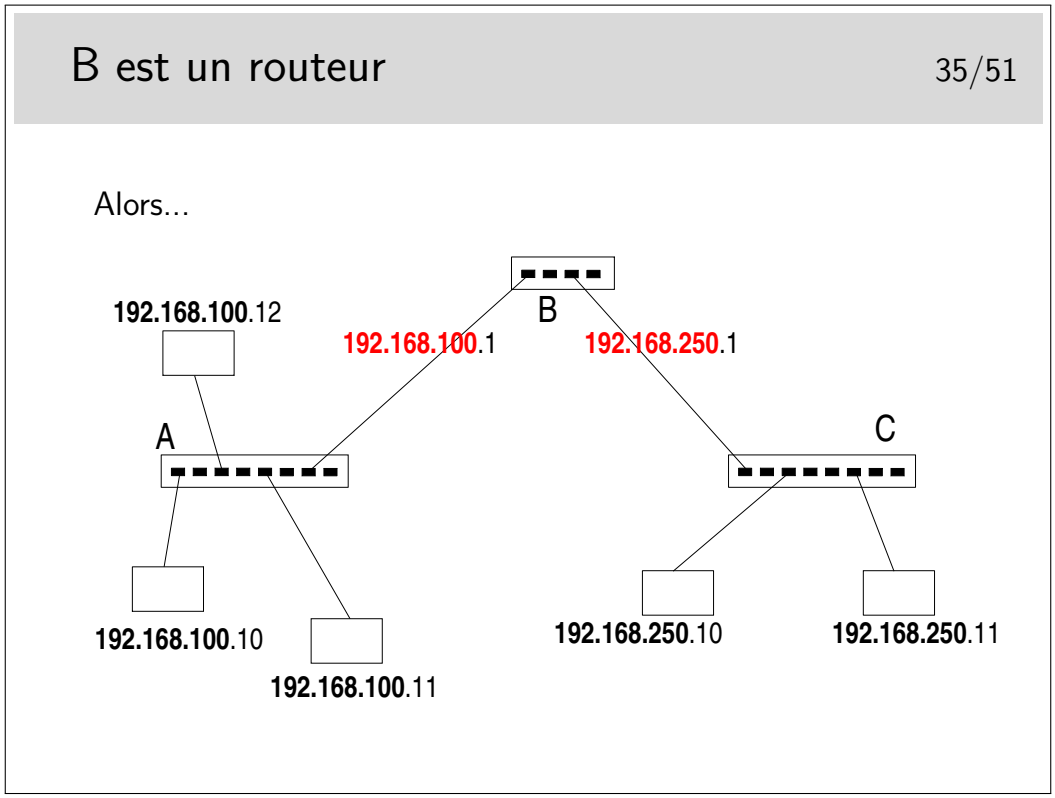
Vous avez le choix entre trois types d'organes :

— des hubs,

— des switches ou commutateurs ou pont multiports

— des routeurs

Aide technique : le netmask est standard pour la classe d'adresses IP utilisée...
Et cette classe est la classe ? ? ? ? ?....
? ? ?
? ? ?
...C !
Ce sont des adresses de classe C...
Or les parties «Réseau» ne sont pas identiques entre la partie droite du schéma et la partie gauche...
Donc... Il ne s'agit pas de mêmes «Réseaux». Donc, forcément B est un... et A et C sont des ... ou des ...



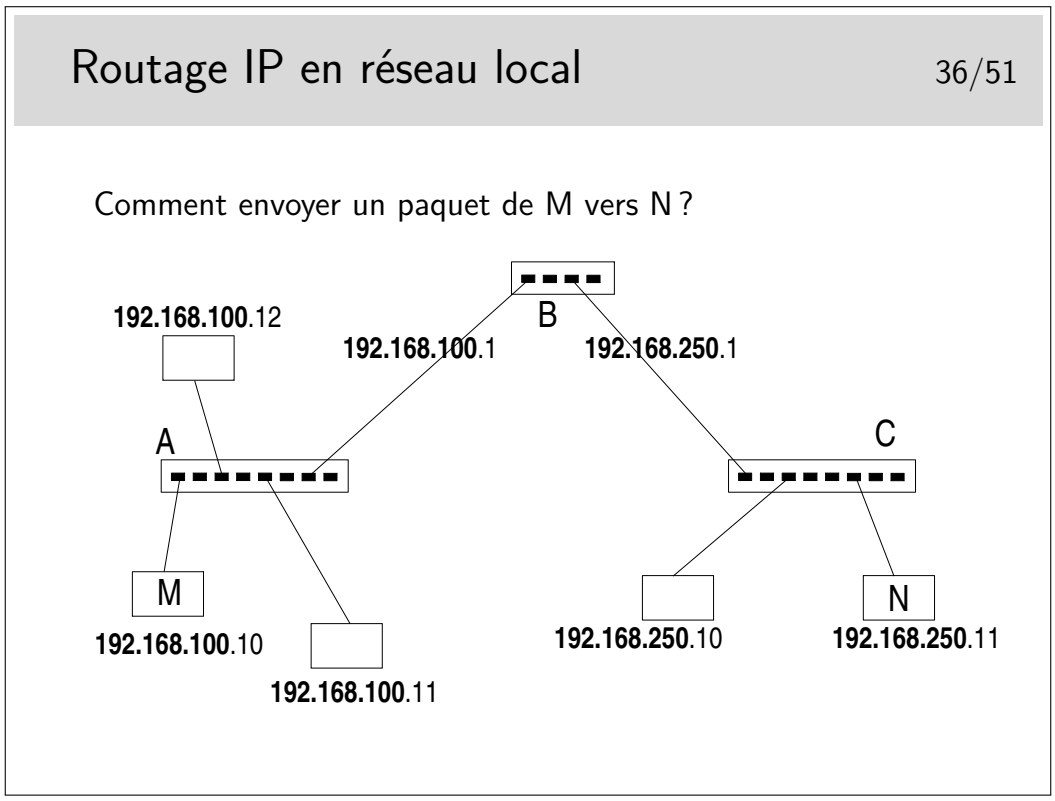
Réponse à la question... A et B sont des hubs ou des commutateurs (ou switches ou ponts multiports). Même si fondamentalement les fonctionnalités des hubs et des commutateurs sont différentes (hubs : répéteurs, organes physiques, niveau 1 ISO – switches : niveau 2 ISO) on ne peut pas faire la différence sur le schéma.

B est un routeur. Il raccorde des réseaux dont les machines sont identifiées par des numéros IP de classe C dont la partie réseau est différente. Il raccorde dont des réseaux. Il fonctionne au niveau 3 ISO.

B est un routeur, donc chacun de ses ports est identifié par une adresse IP dont la partie réseau identifie le réseau auquel ce port est raccordé.

Le premier port de B est raccordé sur le réseau 192.168.100.0, on lui donnera donc une adresse libre de ce réseau (ici .1). Un autre port est sur le réseau 192.168.250.0, on lui donnera par exemple l'adresse .1 sur ce réseau.

Ce n'est pas pour faire joli, c'est vraiment fonctionnel. Si on ne le fait pas le réseau ne peut pas fonctionner.



Rien n'est magique... L'application sur M qui désire envoyer un message à une application sur N doit savoir que N existe. Sur M on doit connaître l'adresse IP de N.

On doit aussi savoir par où on passe. En M on doit savoir que pour atteindre N il faut passer par le routeur B.

Mais B est identifié par deux adresses IP !!!

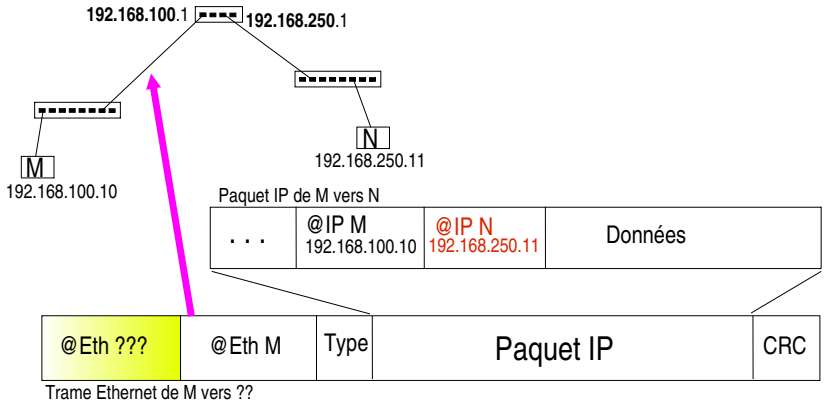
En M, nous sommes sur un réseau local, nous ne pouvons atteindre que des machines se trouvant sur le même réseau local que nous. Nous ne pouvons donc atteindre le routeur que par son port qui est situé sur le même réseau physique, celui identifié par le préfixe IP 192.168.100.

...Comment faire ?...

À suivre...

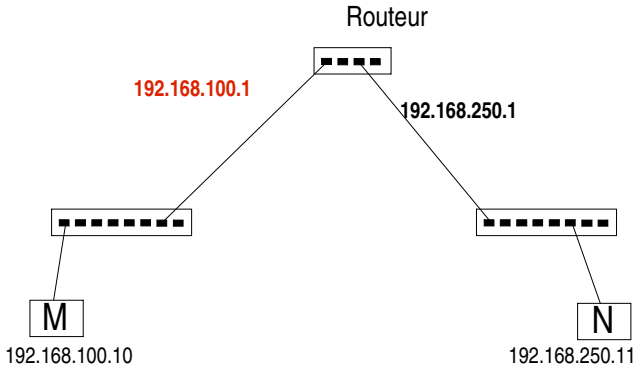
Paquet IP de M vers N, et son porteur... I 37/51

En réseau local



Comment, en M, déterminer l'adresse Mac de la trame Ethernet qui va emporter le paquet vers sa destination ?

Table de routage I 38/51

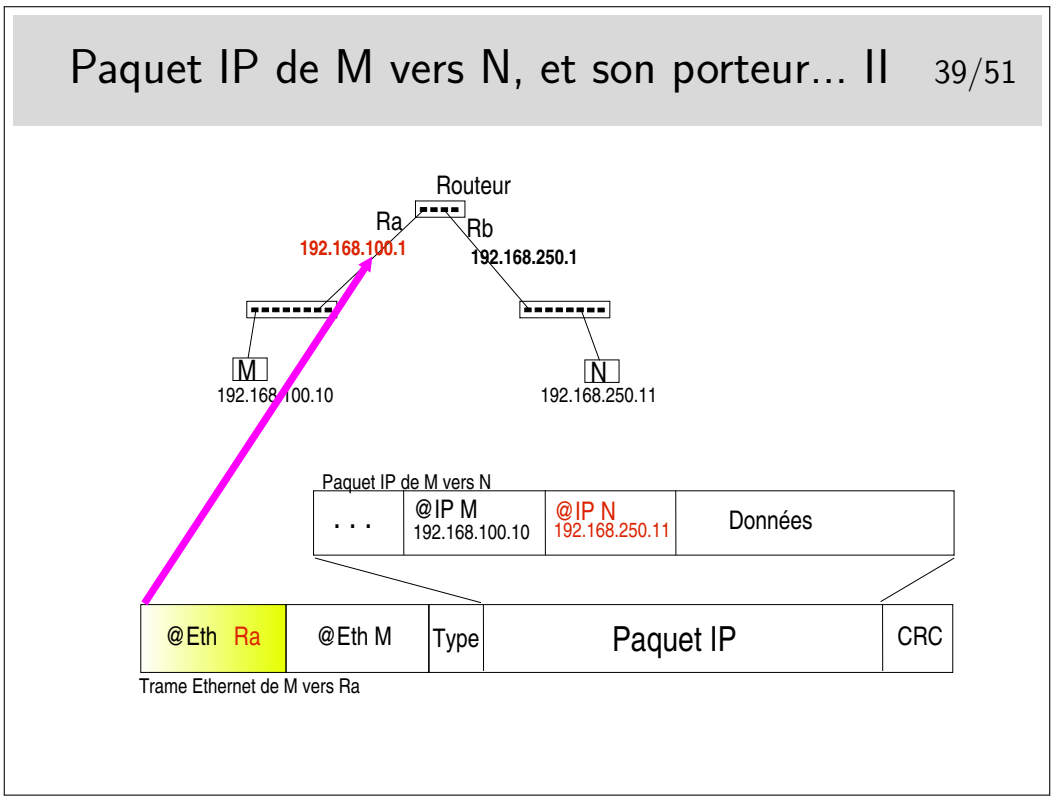


En M il faut une table de routage qui dit : « pour aller en 192.168.250.11 passer par 192.168.100.1 »
À l'aide de cette table il sera facile de faire une résolution ARP pour trouver l'adresse MAC du routeur. L'adresse destination de la trame Ethernet sera celle du routeur

Il faut une table de routage en M. Donc dans la machine terminale!!!

Chaque machine, quelle soit terminale ou intermédiaire, travaillant en IP incorpore des fonctions de routage.

Cela ne signifie pas que chaque machine terminale soit un routeur... Pour qu'une machine puisse joue le rôle de routeur il faut qu'elle possède au moins deux interfaces munies d'adresses IP (autres que 127.0.0.1, correspondant à l'interface boucle locale) et que son module IP soit autorisé à effectuer la fonction de relayage (forwarding).



Quand on disait qu'affecter une adresse à chaque interface de routeur n'était pas un effet de cosmétique...

Si en M on ne connaît pas l'adresse IP de l'interface du routeur qui est du même coté que M (sur le même réseau) alors la résolution ARP ne peut se faire.

Il ne suffit pas que l'interface du routeur et M soient «du même coté», il faut aussi qu'ils soient sur le même réseau local pour que ARP fonctionne (on rappelle que la requête ARP est transmise par broadcast Ethernet et que ce type de message ne passe pas les routeurs).

Ici la notion de réseau local est celle qui a été vue dans la partie précédente du cours, à savoir un réseau où la diffusion est possible vers toutes les machines. Ce pourrait être un VLAN car ce type de topologie définit des domaines de broadcast.

Lorsque des machines sont reliées entre-elles via des liaison point-à-point, il n'y a pas de résolution ARP.

Table de routage II 40/51

Chaque «entrée» dans la table contient au moins

- Une direction (réseau ou machine)
- Une indication de route
  - machine par laquelle les paquets doivent être acheminés
    - Cette machine doit être accessible
  - interface locale
- Un coût
  - Notion de «distance» ou de «coût»
    - Nombre de sauts nécessaires
    - Débit
    - ...

Les directions sont spécifiées par des numéros de réseaux indiqués sur 4 octets et accompagnés de leur netmask, par exemple 192.168.10.0 255.255.255.0, ou en notation CIDR 192.168.10.0/24. Attention, la notation CIDR peut ne pas être comprise par tous les routeurs. Elle l'est pour les machines Unix Linux ou BSD.

Une direction peut être une machine, dans ce cas l'adresse IP de la machine est indiqué directement et le netmask vaut 255.255.255.255.

Les indications de route peuvent être données en indiquant l'adresse IP du routeur par lequel il faut envoyer les paquets pour la direction correspondante. Plus précisément il s'agit de l'adresse IP de l'interface du routeur immédiatement accessible. C'est absolument obligatoire dans le cas ou l'interface du routeur en question est sur un réseau local classique.

Dans le cas ou l'interface du routeur est sur une liaison point-à-point, on peut ne donner que le nom de l'interface locale. Voir transparent suivant.

Le coût correspond à la notion de «plus court chemin». Moins le coût est grand meilleur semble le chemin (ce n'est pas toujours vrai, un chemin peut être plus court en vraie distance (moins de sauts par exemple) et cependant moins efficace en débit).

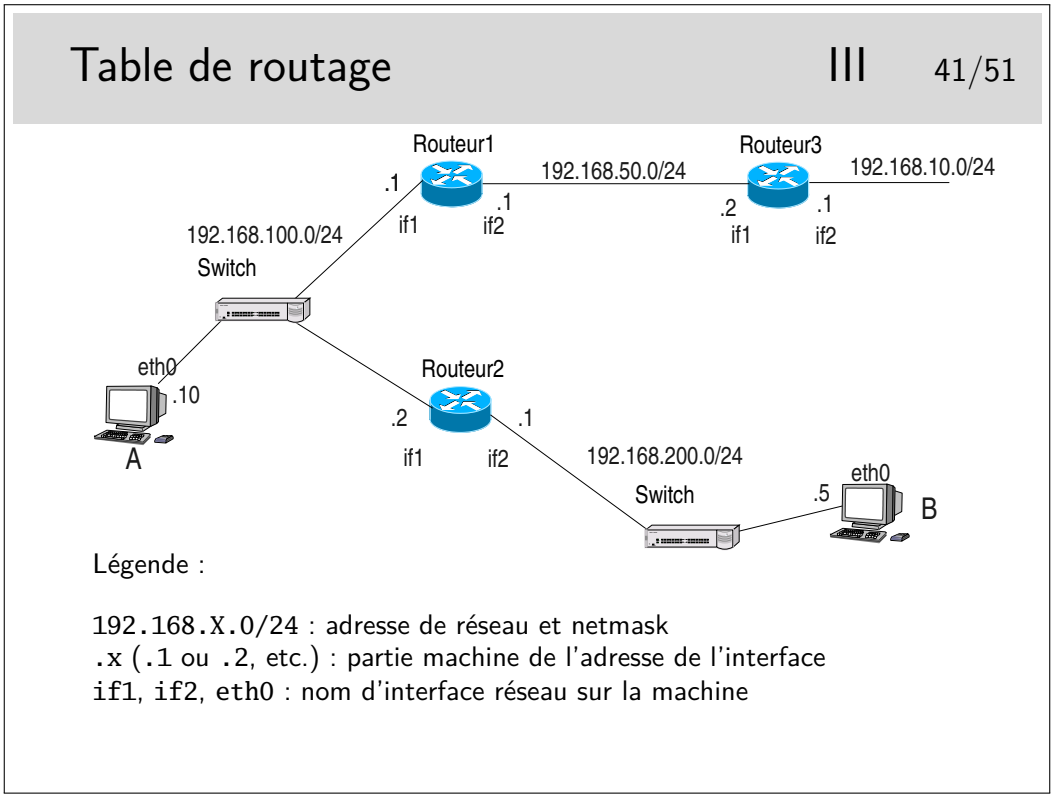


Table de routage en A

direction	netmask	gateway	interface
192.168.100.0	255.255.255.0	192.168.100.10	eth0
192.168.200.0	255.255.255.0	192.168.100.2	eth0
default	0.0.0.0	192.168.100.1	eth0

Rq : ici la mention de l'interface n'est pas très importante, elle est redondante car il n'y a qu'une seule interface physique (il y a quand même l'interface boucle locale qu'on n'a pas fait figurer dans la table de routage, mais qui exite et provoque des entrées spécifiques dans la table)

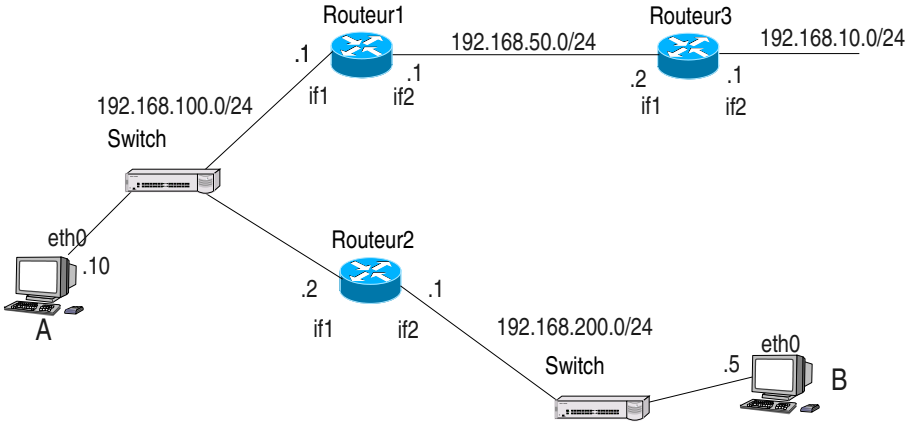
Table de routage en Routeur1

direction	netmask	gateway	interface
192.168.50.0	255.255.255.0	192.168.50.1	if2
192.168.100.0	255.255.255.0	192.168.100.1	if1
192.168.200.0	255.255.255.0	192.168.100.2	if1
default	0.0.0.0	192.168.50.2	if2

Rq : dans la ligne par défaut, la seule mention de l'interface suffirait car le lien entre les routeurs 1 et 2 est de type point-à-point.

Compléter pour le routeur 2 et la machine B...

Table de routage IV 42/51



Considérons les tables de routage correctement servies partout sauf en A
A peut atteindre if1 de Routeur3 (192.168.50.2), mais pas if2 (192.168.10.1)
Pourquoi ne peut-on pas dire en A : pour aller en 192.168.10.0/24 passer par 192.168.50.2 ? *Que faut-il dire ?*

On suppose la table de routage en A partiellement remplie. L'interface if1 du routeur2 est accessible, une commande ping, depuis A, vers l'adresse de cette interface fonctionne. Donc on peut supposer (et on a raison) que cette interface est «visible» depuis A. Cependant si on tente de créer en A une route via cette interface la commande doit échouer. Pourquoi ?

Exemple de table de routage sous Windows 43/51

```
C:\>route print
===========================================================
Liste d'Interfaces
0x1 ........................... MS TCP Loopback interface
0x1000003 ...00 01 02 6e 7c 46 ...... 3Com EtherLink PCI
===========================================================
Itinéraires actifs:
  Destination réseau     Masque réseau   Adr. passerelle   Adr. interface   Métrique
            0.0.0.0           0.0.0.0      192.44.75.1    192.44.75.184      1
          127.0.0.0         255.0.0.0        127.0.0.1        127.0.0.1      1
        192.44.75.0     255.255.255.0      192.44.75.184    192.44.75.184      1
      192.44.75.184   255.255.255.255        127.0.0.1        127.0.0.1      1
      192.44.75.255   255.255.255.255      192.44.75.184    192.44.75.184      1
          224.0.0.0         224.0.0.0      192.44.75.184    192.44.75.184      1
    255.255.255.255   255.255.255.255      192.44.75.184    192.44.75.184      1
Passerelle par défaut: 192.44.75.1
===========================================================
```

Quelle est le numéro de l'interface de boucle locale (interface interne non raccordée à un réseau physique) ?

Quelle sont les adresses de broadcast possibles ?

Par quelle interface sont acheminés les paquets de broadcast ?

Que pouvez vous déduire de tout ceci concernant l'adresse de la machine en question ?

Que se passe-t'il si une application de cette machine envoie un paquet IP vers une autre application de la même machine ? Quel chemin est emprunté par le paquet ?

Quelle adresse code la destination par défaut ?

Et votre système dans tout ça... 44/51

Quel est votre paramétrage ?

- Sous Windows (Ouvrez une fenêtre de commande et testez les commandes suivantes :)
  - `ipconfig` (`winipcfg` sous Windows 9x/ME) avec le sélecteur `/all`
  - `route print` pour afficher la table de routage
  - `nslookup` pour traduire des noms de machine en adresse IP et inversement
  - `arp -a` si vous êtes sur un LAN pour lire la table de traduction arp
  - `tracert`
- Sous Linux (Ouvrez un terminal et testez :)
  - `ifconfig` avec ou sans `-a`
  - `route` avec ou sans `-n`
  - `nslookup` ou `host`
  - `arp -a`
  - `traceroute` (avec ou sans `-n`)

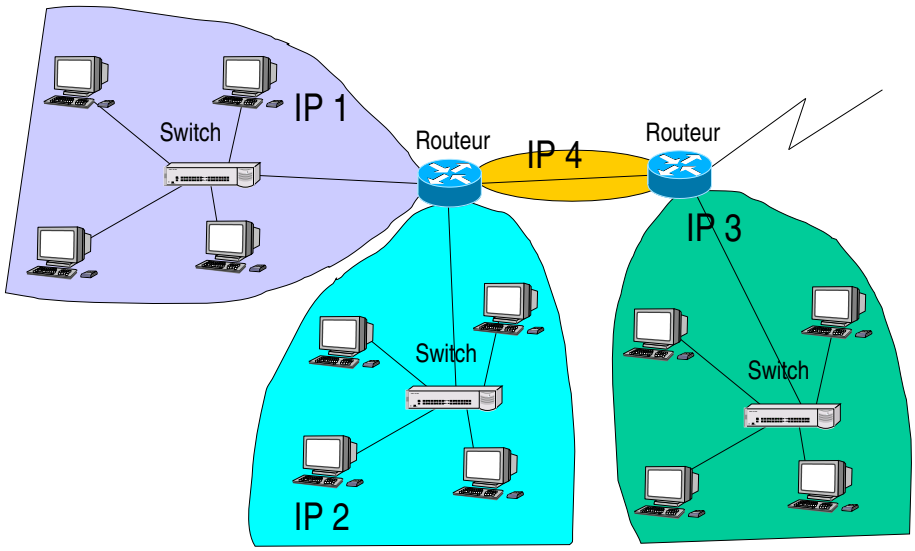
Et n'oubliez pas la commande ping... C'est la première commande à utiliser quand le réseau ne va pas très bien... «Mon voisin est-il joignable ? Alors :

`ping mon_voisin` (plutôt `adresse_IP_de_mon_voisin`)

Note : Vous remarquerez peut être que Windows affiche plus de routes que Linux. En fait Linux gère plusieurs tables de routage et a pour habitude de n'afficher que la table `main`, sauf si on lui demande gentiment. (Pour avoir la liste des tables de routage : `cat /etc/iproute2/rt_tables` ; pour afficher la table principale : `ip route show table main` ; pour afficher la table avec les règles pour le multicast et le broadcast : `ip route show table local` ; etc.)

5 Les routeurs

Le réseau IP de l'entreprise 46/51



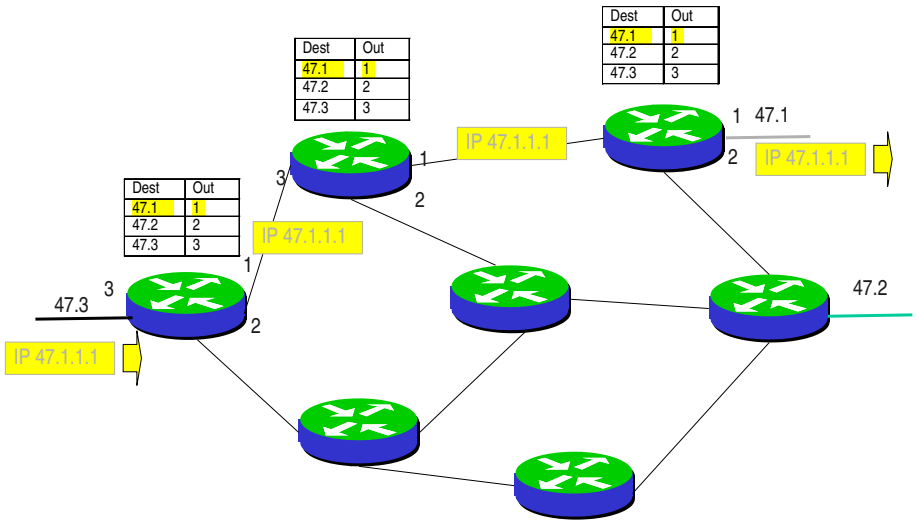
Ce que l'on va appeler le «réseau local» de l'entreprise est en réalité un réseau de niveau 3 composé de routeurs interconnectant des réseaux locaux séparés physiquement ou logiquement (VLANs).

L'adressage pourra être de type «privé», avec une seule adresse officielle en sortie et une fonction de type NAT dans le routeur de sortie.

Fonctions des routeurs 47/51

- Chaque paquet IP entrant dans le routeur voit son adresse destination examinée et comparée sur un certain nombre de bits avec le contenu d'une table (table de routage) associant des directions avec une interface de sortie
  - Cette fonction est aussi mise en œuvre dans les machines terminales
  - Le champ TTL est décrémenté par chaque routeur, le champ *cheksum* doit être recalculé à chaque fois

Acheminement bond par bond 48/51



6 Gestion des erreurs avec ICMP

Le protocole ICMP (IPv4) 50/51

Internet Control Message Protocol (rfc 792)

- Sert à véhiculer des messages d'erreur ou de demande d'information
  - Demande d'écho et réponse (commande ping)
  - Destination non accessible
    - Le réseau ne peut être atteint
    - La fragmentation est nécessaire et le bit D est à 1 (PMTU discovery)
    - etc
  - Redirection, il existe une meilleure route
  - Durée de vie dépassé
  - etc

La commande **`ping`** n'utilise que ICMP (porté par IP).

La commande **`traceroute`** (**`tracert`** sous Windows) joue sur le dépassement de la durée de vie. Un premier paquet est créé «à la main» et son champ ttl est mis à 1. Il contient un paquet UDP à destination d'un port inconnu. Le paquet est routé, il atteint le premier routeur qui décrémente alors le ttl. Le résultat valant 0 le paquet est jeté et un message ICMP est émis

vers la source du paquet jeté. Le message est véhiculé par un paquet IP comportant l'adresse du routeur, ce qu'on attend dans traceroute. On peut alors afficher l'identité de ce routeur ainsi que le temps mesuré entre l'envoi du paquet initial et le l'arrivée du message ICMP. On fait cela trois fois pour avoir une estimation du temps moyen d'aller et retour puis on recommence en mettant cette fois le ttl à 2.

Le protocole ICMPv6 (IPv6) 51/51

Internet Control Message Protocol v6 (rfc 4443)

- Mêmes fonctions que ICMP (IPv4)
  - Erreurs diverses, ping, redirections, etc.
- En plus :
  - Neighbor Discovery Protocol (équivalent du ARP pour IPv4)
  - Router Solicitation, Router Advertisement : Auto-configuration de l'adresse. Au démarrage, un host recherche s'il y a un routeur présent sur LAN ; le routeur diffuse le préfixe IPv6 à utiliser ; le host se choisit une adresse dans ce préfixe.

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

TP - Analyse des protocoles IP, TCP et UDP

Cybersécurité des systèmes maritimes et portuaires

2021

Vous trouverez en annexe une feuille qui vous permettra de noter exactement quelles commandes et quels paramètres vous utiliserez pour configurer le réseau ainsi que les réponses aux questions posées dans ce TP.

1 Mise en place du réseau et paramétrages

Le réseau de test est architecturé comme le montre la figure 1.

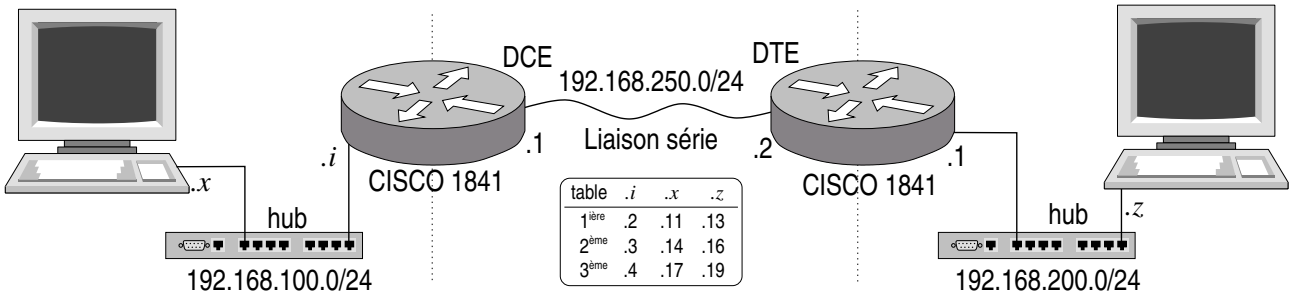


FIGURE 1 – Réseau de test

Les machines terminales sont des PCs munis du système d'exploitation Linux. Le mot de passe administrateur vous sera donné en début de séance. La configuration a déjà été réalisée.

Les routeurs CISCO sont déjà préconfigurés (nous apprendrons à les configurer dans un TP ultérieur). Entre ces routeurs a été mis en place une liaison série (encapsulation HDLC, débit 64Kb/s).

Dans cette partie du TP il vous est demandé de relever la configuration réseau de vos machines : notez leurs adresses IP (IPv4 et IPv6 <scope global>), ainsi que leur route par défaut.

Rappels sur les commandes Unix réseau :

ip address (que l'on peut raccourcir en `ip addr`) permet d'afficher ou attribuer une addresse IP à une interface internet. [1] Exemples :
`man ip` et `man ip-address` : la documentation de cette commande, à lire en premier
`ip addr` : liste toutes les interfaces disponibles
`if addr eno2 192.168.xxx.yyy/24 broadcast 192.168.xxx.255 up`

ip route permet de gérer la table de routage de la machine. [2] Exemple :
`man ip` et `man-route` : la documentation de cette commande, à lire en premier
`ip route` : afficher la table de routage
`ip route add default gw 192.168.xxx.zzz` : positionne la route par défaut, i.e. l'adresse IP du routeur pour 'sortir'
`ip route del ...` : pour détruire une route.

ping (ping-pong) permet de tester la liaison IP vers une autre machine. Concrètement des paquets ICMP ECHO font l'aller-retour vers l'addresse IP testée. On vérifie ainsi que la configuration IP (adresses, routes, etc.) fonctionne correctement. Exemple :
`ping -n 192.168.100.11` (faire Ctrl-C pour arrêter)

ethtool permet de gérer les fonctions matérielles des cartes ethernet modernes (optionnel).
Exemple : `ethtool eno2` : affiche les possibilités de la carte ethernet.

ip est une commande générique et plus moderne qui tend à remplacer `ifconfig` `route` etc.
Exemple : `ip add show`, ou `ip route show`

1. Les nostalgiques peuvent utiliser la commande `ifconfig` qui fait quasiment la même chose.
2. Les nostalgiques peuvent utiliser la commande `route` qui fait quasiment la même chose.

Pour comprendre un peu mieux comment fonctionne les protocoles IP, TCP, UDP, il faut rendre le système un peu moins intelligent... Les commandes indiquées ci-dessous ont un but purement pédagogique, et en temps normal on laisse généralement les optimisations par défaut. Notez que ces paramétrages ont déjà été fait pour préparer ce TP.

— Si l'*autotuning TCP/IP* est activé (ce qui est souvent le cas par défaut sur Linux), le système s'autorise à changer dynamiquement la taille du buffer de réception (et parfois ré-augmenter la fenêtre TCP). Nous avons donc désactivé cela sur nos PCs pour garder un comportement TCP plus basique [3] :

```
# Disable TCP autotuning
sysctl -w net.ipv4.tcp_moderate_rcvbuf=0
# Force TCP receive buffers to a static size
sysctl -w net.ipv4.tcp_rmem="87380 87380 87380"
# Flush current routing cache
sysctl -w net.ipv4.route.flush=1
```

Pour information, le paramétrage par défaut est :

```
sysctl -w net.ipv4.tcp_moderate_rcvbuf=1
sysctl -w net.ipv4.tcp_rmem="4096 87380 3416064"
```

— L'*offload* sur une carte ethernet consiste à faire réaliser par l'électronique de la carte des opérations qui relèvent plutôt des couches IP-TCP-UDP du système d'exploitation (comme le calcul du checksum des paquets IP, ou le découpage en fonction du MTU des segments TCP). Certes, si la carte sait le faire, cela augmente les performances du système, mais ce mélange des rôles est nuisible sur le plan pédagogique... En fait, l'analyseur de protocole que nous allons utiliser risque de capturer des paquets sortants à une étape dans le système d'exploitation où ils ne sont pas encore complètement bien formés. Nous avons donc demander la désactivation de tout ce qui est possible de désactiver sur notre carte Ethernet avec :

```
ethtool --offload eno2 rx off tx off sg off tso off ufo off gso off gro off lro off
rxhash off
```

On peut obtenir quelques messages d'erreur si certaines fonctions ne peuvent pas être désactivées. C'est sans grande importance. Pour afficher ce qui a pu être désactivé : `ethtool --show-offload eno2`

wireshark Par la suite, sur chacun des PC, nous allons utiliser l'outil `wireshark` (anciennement appelé `ethereal`). Cet outil place la carte ethernet en mode *promiscuous* (cf. `man ifconfig`) : dans ce mode la carte remonte à l'OS tous les paquets ethernet qu'elle voit passer sur le bus réseau ethernet. Ainsi, `wireshark` est capable de capturer tous les paquets qui transitent sur le réseau ethernet auquel est raccordée la machine, et de les afficher de manière sympathique et lisible par un humain.

Toujours dans un but pédagogique, nous avons (déjà) modifié la configuration initiale de la manière suivante :

— Dans le menu `Capture / Options`, cocher `Update list of packet in real time` et `Automatic scrolling in live capture` (on visualise mieux l'aspect dynamique des échanges de paquets), et décocher `Enable network name resolution` (les requêtes DNS ont tendance à polluer la capture).
— Dans `Edit / Preferences / Protocoles / IPv4` décochez :
  — `Reassemble fragmented IP datagrams` (Car justement on veut voir ça dans le TP d'aujourd'hui.)
  — `Validate the IP checksum if possible` (Au cas où la commande `ethtool` de la section récédante n'aurait pas pu désactiver `tx-checksumming`.)
  — `Enable IPv4 geolocaltion` (Ça, c'est franchement inutile.)
— Dans `Edit / Preferences / Protocoles / IPv6` décochez :
  — `Reassemble fragmented IPv6 datagrams`
  — `Enable IPv6 geolocaltion`
— Dans `Edit / Preferences / Protocoles / TCP` décochez :
  — `Validate the TCP checksum if possible`
  — `Relative sequence numbers` (C'est finalement intéressant de voir que ça ne commence pas réellement à 1...)
  — `Analyze TCP sequence numbers` (On va analyser ça nous même dans ce TP.)
— Dans `Edit / Preferences / Protocoles / UDP` décochez :
  — `Validate the UDP checksum if possible`

2 Étude de UDP

Allez dans le répertoire `/opt/TP/TPAnalyseProto`. Vous allez utiliser les programmes `recepteurUDP` et `emetteurUDP` dont vous trouverez les sources en annexe.

3. Cela a déjà été fait par le script `./configure`, inutile de taper ces commandes

— Le recepteur lit des données de taille maximale 100000 octets. On le lancera sur l'un des PC de la manière suivante :
`./recepteurUDP` *`NuméroPort`*
Le numéro de port doit être supérieur à 7000 pour éviter des conflits avec d'autres applications pouvant utiliser des ports inférieurs à cette valeur (par exemple 8003).
— L'émetteur ne peut émettre qu'un seul message de taille paramétrable fournie en argument au lancement. Sur un autre PC, on lancera l'émetteur de la manière suivante :
`./emetteurUDP` *`numéroIPdistant NuméroPortRécepteur nbOctetsMessage`*

Questions :

1. Analysez le code C de l'émetteur et du récepteur. Quelle primitive (fonction) permet d'indiquer l'utilisation du protocole UDP ? Quelle est la taille maximale des messages que l'émetteur est capable de construire ? (Note : lisez le `man` des fonctions et appels systèmes que vous découvrez dans le code C.)
2. Lancez l'exécution du récepteur sur une machine d'extrémité puis lancez l'émetteur sur l'autre machine. Les programmes affichent sur leur console la taille des buffers de socket alloués par l'OS pour la transmission et la réception (buffers alloués dans la couche UDP et non dans les programmes eux-mêmes). Quelle est cette taille ?
3. Quel est le volume de données maximum que peut envoyer réellement l'émetteur ? Essayez en IPv4 et en IPv6 (sur l'adresse de scope global). Comparez ce volume à la réponse à la question 1.
4. Lancez l'émetteur en lui faisant envoyer un message 1000 octets puis un autre de 2000 octets.
   Faites un chronogramme des échanges en étudiant les trames échangées. Éventuellement, vérifiez votre travail en sélectionnant le menu `Statistics / Flow Graphs`.
5. Pour le message de 2000 octets vous constatez une fragmentation. Quelle couche opère cette opération ? Quelle en est la cause ?
6. Comment s'opère cette fragmentation ?
7. Que se passe t'il si le récepteur ne lit pas les données envoyées par l'émetteur ? Si l'émetteur se trompe de numéro de port ? Si l'émetteur se trompe d'adresse IP récepteur (essayez une mauvaise adresse dans le réseau local de l'émetteur, puis une mauvaise adresse un autre réseau local derrière le routeur, et enfin une adresse privée non routable dans Internent et non routée à l'intérieur de l'école p.ex. 172.16.16.16) ?

3 Étude de TCP

Nous allons maintenant utiliser deux programmes communiquant via TCP. Un serveur (de nom `lazyServerTCP`) et un client (`clientTCP`). Vous trouverez en annexe le code source C de ces deux programmes.

Le serveur `lazyServerTCP` accepte les connexions demandées par les clients mais il ne consomme pas les données envoyées par ceux-ci (il ne lit pas les données). On simule ainsi un serveur qui serait ralenti par le traitement au fil de l'eau des données reçues, ce ralentissement serait tel qu'il ne pourrait pas lire le flux entrant pendant un temps relativement long.

Les données reçues remplissent peu à peu le buffer socket de réception et TCP met en œuvre son contrôle de flux afin d'avertir l'émetteur de réduire son taux d'émission, voire de l'arrêter.

Le client est configuré pour émettre sans fin des messages de 100Ko.

— Sur la machine Linux de numéro IP `192.168.200.`*`x`* lancez le serveur en lui fournissant en paramètre un numéro de port supérieur à 7000 (pour éviter des conflits avec d'autres applications pouvant utiliser des ports inférieurs à cette valeur) :
`./lazyServerTCP` *`numéroPort`*
— Lancez l'analyseur `wireshark` et configurez-le comme précédemment.
— Sur la machine Linux de numéro IP `192.168.100.`*`x`* lancez l'analyseur `wireshark` dans les même conditions que sur la machine serveur.
— Sur cette même machine lancez le programme `clientTCP` de la manière suivante :
`./clientTCP` *`numéroIPserveur numéroPort`*

Questions :

1. Analysez le code du client et du serveur. Quelle primitive permet de préciser l'utilisation du protocole TCP ?
2. Faites un chronogramme des échanges des 4 premiers segments TCP en précisant en particulier les numéros de séquence et d'acquittement.

3. Quelles sont les options de TCP utilisées lors de la connexion et lors des échanges ? (l'usage de ces options n'est pas obligatoire, certaines implémentations de TCP ne les mettent pas en œuvre, tout au moins les nouvelles options). Quelle est l'utilité de ces options ?
4. Les programmes client et serveur affichent la taille de leurs buffers de socket, quelle est cette taille ?
5. À la différence d'UDP, TCP met en place un mécanisme de contrôle de flux permettant au récepteur d'asservir le débit de l'émetteur à ses capacités de traitement. Comment s'opère ce contrôle de flux TCP ?
6. Quel est le volume de données émis dans chaque message porteur de données d'application ? Comment expliquez vous la valeur trouvée ?
7. Quel est le volume de données émis par le client avant que le flux ne soit stoppé par les messages du serveur ? Comment expliquez vous cette valeur (pensez au temps de traversée) ?
8. Du côté client et du côté serveur, relevez les valeurs de `Recv-Q` et `Send-Q` à l'aide de la commande `netstat -tn` (lisez le man). La valeur du `Recv-Q` (côté serveur) doit vous faire penser à quelque chose ; de même la somme de `Recv-Q` (serveur) et `Send-Q` (client).
   Combien d'octets le client dit avoir déjà envoyé avant d'être mis en attente par l'OS sur le `write()` suivant ? [4]
9. Faites un chronogramme des messages échangés, vu du coté client puis vu du coté serveur (faites deux chronogrammes).

4 PATH MTU Discovery

Vous allez étudier maintenant comment s'opère l'adaptation de TCP à la plus petite valeur de MTU (Maximum Transmit Unit) sur le chemin entre l'émetteur et le récepteur (à noter que le client comme le serveur sont à la fois émetteur et récepteur).

Arrêtez le serveur et le client lancés pour le test précédent.

Configurez le MTU de la liaison série entre les routeurs pour qu'il soit de 700. Il faut intervenir sur le routeur qui est du côté de la machine "cliente". Sur le PC qui administre ce routeur, lancez l'émulateur de terminal `gtkterm` et tapez :

```
cisco1#show interfaces Serial 0/0/0 (repérer le MTU actuel)
cisco1#configure terminal
cisco1(config)#interface Serial 0/0/0
cisco1(config-if)#mtu 700
cisco1(config-if)#exit
cisco1(config)#exit
cisco1#show interfaces Serial 0/0/0 (repérer le nouveau MTU)
cisco1#exit
Ctrl-A Q
```

Relancez le programme `lazyServerTCP`, l'analyseur `wireshark` coté client et le programme `clientTCP` dans les mêmes conditions que précédemment.

Questions :

1. Quels sont les messages échangés en début de session et entre quels organes sont ils échangés ? (faites un chronogramme)
2. Comment s'opère l'adaptation au nouveau MTU ? Y a t'il segmentation au niveau IP ?
3. Arrêtez le client et le serveur. Stoppez la capture `wireshark` puis relancez la. Relancez le serveur puis le client. Examinez les traces des échanges, quelles sont les différences ? Quelle conclusion peut-on en tirer ?
4. Comment se passeraient des envois de même volume avec UDP ? (vérifiez au besoin avec les programmes UDP utilisés précédemment). Y a t'il mise en œuvre du *PMTU Discovery* ?
5. Retrouvez ce Path MTU dans le cache de la table de routage avec `ip route show table cache` ou `ip route get <address>`
6. Si vous avez besoin de recommancer vos essais, pensez à purger le cache de la table de routage : `ip route flush cache`.

4. Notez que la relation avec `SO_SNDBUF` et `SO_RCVBUF` est assez compliquée : Linux s'autorisant à faire de l'*autotunning*, ou prend en compte ces paramètres de manière spécifique. Voir le `man 7 tcp`, et `http://www.psc.edu/networking/projects/tcptune/#Linux`

7. Dernier point indépendant des précédents : en conservant la configuration présente et en utilisant les programmes client et serveur UDP, comparez la segmentation IP faite par Linux (voir exercice 2) et par les routeurs Cisco. Vous inhiberez le *PATH MTU Discovery* de Linux par la commande : `sysctl -w net.ipv4.ip_no_pmtu_disc=1`

   Cette commande positionne un paramètre du noyau accessible via le pseudo fichier `/proc/sys/net/ipv4/ip_no_pmtu_disc`.

   Notez qu'il y a d'autres paramètres ajustables concernant le *PATH MTU Discovery* : `/proc/sys/net/ipv4/route/min_pmtu` la taille minimum acceptée (552 par défaut), `/proc/sys/net/ipv4/route/mtu_expires` le temps pendant lequel l'information est gardée en mémoire par le noyau (600s par défaut).

5 Contrôle de congestion

Nous avons vu que dans TCP, le récepteur peut avertir l'émetteur qu'il est congestionné en réduisant la valeur de la *windows size* dans les paquet qu'il lui retourne (la taille de son buffer de réception). C'est la gestion de la congestion de bout en bout, facile.

Plus difficile, la gestion de la congestion sur le chemin consiste pour l'émetteur à estimer le nombre de paquets qu'il peut émettre sur le réseau avant de s'attendre à recevoir un acquittement. Ce nombre de paquets est la fenêtre d'anticipation, appelée encore fenêtre de congestion dans le jargon TCP. Il existe de nombreux algorithmes pour se choisir une bonne fenêtre d'anticipation. [5] Au jour d'aujourd'hui, l'algorithme par défaut sous Linux est *Cubic*.

Nous n'étudierons pas ces algorithmes complexes. Par contre, on peut regarder le résultat. Utilisez la commande `ss` (*Socket Statistics*, lisez le `man`) qui donne des informations un peu plus poussées que `netstat` :

`/sbin/ss -i -e -t -p`

Retrouvez vos sockets TCP. Suivant leur état (connecté, en cours, en attente, etc.) vous aurez diverses informations internes : les options TCP utilisées (timestamp, acquittement sélectif, window scale, etc.), le `rtt` mesuré sur la connexion (moyenne/variance), le `rto` (*Retransmission Time Out*), le `ato` (*delayed Acknowledgment Time Out*), le `ssthresh` (*Slow Start Threshold*), et le fameux `cwnd` (*Congestion Window*).

Notez que la commande `ip route show table cache` vue précédemment vous donne certaines de ces valeurs gardées en cache.

5. `http://en.wikipedia.org/wiki/TCP_congestion_avoidance_algorithm`

130

Annexe A

Code source C

Listing 1 – emetteurUDP.c

```
/* Auteur: Alain Leroy, Christophe Lohr - Telecom Bretagne */
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

#define BUFSIZE 100000

int main(int argc, char **argv) {
  int sfd, s, nb, ssz, rsz;
  struct addrinfo hints;
  struct addrinfo *result, *rp;
  struct sockaddr *sa;
  socklen_t salen, lg;
  char buf[BUFSIZE];
  ssize_t nwrite;

  if (argc != 4) {
    printf("Usage : %s nom_machine_recepteur port_recepteur nombre_octets_emettre\n", argv[0]);
    exit(EXIT_FAILURE);
  }

  nb = atoi(argv[3]);
  if (nb <= 0) {
    printf("Donnez un nombre d'octets a emettre superieur a 0\n");
    exit(EXIT_FAILURE);
  }

  /*
   * Obtention de l'adresse IP du distant, a partir de son nom par
   * consultation du fichier /etc/hosts ou de la base hosts des NIS
   * ou du DNS (Domain Name Service)
   * cf. man getaddrinfo(3)
   */
  memset(&hints, 0, sizeof(struct addrinfo));
  hints.ai_family = AF_UNSPEC;          /* IPv4 ou IPv6 */
  hints.ai_socktype = SOCK_DGRAM;       /* Datagram socket */
  hints.ai_flags = 0;
  hints.ai_protocol = 0;                /* Any protocol */

  s = getaddrinfo(argv[1], argv[2], &hints, &result);
  if (s != 0) {
    fprintf(stderr, "getaddrinfo: %s\n", gai_strerror(s));
    exit(EXIT_FAILURE);
  }

  /* getaddrinfo() retourne une liste de structures d'adresses.
     On essaie chaque adresse jusqu'a ce que socket(2) reussisse. */
  for (rp = result; rp != NULL; rp = rp->ai_next) {
    /* Ouverture de la socket */
    sfd = socket(rp->ai_family, rp->ai_socktype, rp->ai_protocol);
    if (sfd >= 0)
      break;
  }
  if (rp == NULL) {      /* Aucune adresse valide */
    fprintf(stderr, "Impossible d'ouvrir une socket vers %s\n", argv[1]);
    perror("socket");
    exit(EXIT_FAILURE);
  }

  /*
   * Construction de la structure d'adresse du distant
   */
  sa=malloc(rp->ai_addrlen);
  memcpy(sa, rp->ai_addr, rp->ai_addrlen);
  salen=rp->ai_addrlen;

  freeaddrinfo(result); /* Plus besoin */


  /* Option des sockets: taille des buffers */
  lg = sizeof(rsz);
  if ( getsockopt(sfd, SOL_SOCKET, SO_RCVBUF, &rsz, &lg) == 0 )
    printf("SO_RCVBUF par defaut: %d octets\n", rsz);
  else
    perror("getsockopt SO_RCVBUF");

```

```
    lg = sizeof(ssz);
    if ( getsockopt(sfd, SOL_SOCKET, SO_SNDBUF, &ssz, &lg) == 0 )
      printf("SO_SNDBUF par defaut: %d octets\n", ssz);
    else
      perror("getsockopt SO_SNDBUF");

    rsz = 80000;
    if ( setsockopt(sfd, SOL_SOCKET, SO_RCVBUF, &rsz, sizeof(rsz)) == 0 )
      printf("SO_RCVBUF apres forcage: %d octets\n", rsz);
    else
      perror("setsockopt SO_RCVBUF");


    /* Initialisation du message a transmettre */
    memset(buf, (unsigned char)'a', BUFSIZE);

    /* Ecriture socket */
    nwrite = sendto(sfd, buf, nb, 0, sa, salen);
    printf("Ecrits %zd octets sur la socket\n", nwrite);
    if (nwrite < 0)
      perror("Erreur ecriture");

    close(sfd);

    exit(EXIT_SUCCESS);
}
```

Listing 2 – recepteurUDP.c

```
/* Auteur : Alain Leroy, Christophe Lohr - Telecom Bretagne */
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/socket.h>
#include <netdb.h>

#define BUFSIZE 100000

int main(int argc, char **argv) {
  int sfd, s, rsz, ssz;
  struct addrinfo hints;
  struct addrinfo *result, *rp;
  socklen_t lg;
  ssize_t nread;
  char buf[BUFSIZE];
  struct sockaddr_storage peer_addr;
  socklen_t peer_addr_len;
  char host[NI_MAXHOST];


  if (argc != 2) {
    printf("Usage : %s  port_recepteur\n", argv[0]);
    exit(EXIT_FAILURE);
  }

  /* Construction de l'adresse locale (pour bind) */
  memset(&hints, 0, sizeof(struct addrinfo));
  hints.ai_family = AF_INET6;           /* Force IPv6 */
  hints.ai_socktype = SOCK_DGRAM;       /* Datagram socket */
  hints.ai_flags = AI_PASSIVE;          /* Pour l'adresse IP joker */
  hints.ai_flags |= AI_V4MAPPED|AI_ALL; /* IPv4 remappe en IPv6 */
  hints.ai_protocol = 0;                /* Any protocol */

  s = getaddrinfo(NULL, argv[1], &hints, &result);
  if (s != 0) {
    fprintf(stderr, "getaddrinfo: %s\n", gai_strerror(s));
    exit(EXIT_FAILURE);
  }

  /* getaddrinfo() retourne une liste de structures d'adresses.
     On essaie chaque adresse jusqu'a ce que bind(2) reussisse.
     Si socket(2) (ou bind(2)) echoue, on (ferme la socket et on)
     essaie l'adresse suivante. cf man getaddrinfo(3) */
  for (rp = result; rp != NULL; rp = rp->ai_next) {
    /* Creation de la socket */
    sfd = socket(rp->ai_family, rp->ai_socktype, rp->ai_protocol);
    if (sfd == -1)
      continue;
    /* Association d'un port a la socket */
    if (bind(sfd, rp->ai_addr, rp->ai_addrlen) == 0)
      break;                /* Succes */
    close(sfd);
  }
  if (rp == NULL) {      /* Aucune adresse valide */
    perror("bind");
    exit(EXIT_FAILURE);
  }
  freeaddrinfo(result); /* Plus besoin */
```

```
63
    /* Option des sockets: taille des buffers */
    lg = sizeof(rsz);
    if ( getsockopt(sfd, SOL_SOCKET, SO_RCVBUF, &rsz, &lg) == 0 )
      printf("SO_RCVBUF par defaut: %d octets\n", rsz);
68  else
      perror("getsockopt SO_RCVBUF");

    lg = sizeof(ssz);
    if ( getsockopt(sfd, SOL_SOCKET, SO_SNDBUF, &ssz, &lg) == 0 )
73    printf("SO_SNDBUF par defaut: %d octets\n", ssz);
    else
      perror("getsockopt SO_SNDBUF");

    rsz = 80000;
78  if ( setsockopt(sfd, SOL_SOCKET, SO_RCVBUF, &rsz, sizeof(rsz)) == 0 )
      printf("SO_RCVBUF apres forcage: %d octets\n", rsz);
    else
      perror("setsockopt SO_RCVBUF");

83
    /* Boucle de communication */
    for(;;) {
      peer_addr_len = sizeof(struct sockaddr_storage);
      nread = recvfrom(sfd, buf, BUFSIZE, 0,
88                     (struct sockaddr *) &peer_addr, &peer_addr_len);
      if (nread == -1) {
        perror("Erreur en lecture socket\n");
        exit(EXIT_FAILURE);
      } else {
93      s = getnameinfo((struct sockaddr *) &peer_addr,
                        peer_addr_len, host, NI_MAXHOST, NULL, 0, NI_NUMERICHOST);
        if (s == 0)
          printf("Recu %zd octets de %s\n", nread, host);
        else
98        printf("Recu %zd octets (erreur: %s)\n", nread, gai_strerror(s));
      }
    }
  }
```

Listing 3 – clientTCP.c

```
  /* Auteur : Alain Leroy, Christophe Lohr - Telecom Bretagne */
  #include <sys/types.h>
  #include <sys/socket.h>
4 #include <netdb.h>
  #include <stdio.h>
  #include <stdlib.h>
  #include <unistd.h>
  #include <string.h>
9
  #define BUFSIZE 100000

  int main(int argc, char **argv) {
    int sfd, s, ssz, rsz;
14  struct addrinfo hints;
    struct addrinfo *result, *rp;
    socklen_t lg;
    char buf[BUFSIZE];
    ssize_t nwrite;
19

    if (argc != 3) {
      printf("Usage : %s nom_machine_distante port_serveur\n", argv[0]);
      exit(EXIT_FAILURE);
24  }

    /*
     * Obtention de l'adresse IP du distant, a partir de son nom par
     * consultation du fichier /etc/hosts ou de la base hosts des NIS
29   * ou du DNS (Domain Name Service)
     * cf. man getaddrinfo(3)
     */
    memset(&hints, 0, sizeof(struct addrinfo));
    hints.ai_family = AF_UNSPEC;           /* IPv4 ou IPv6 */
34  hints.ai_socktype = SOCK_STREAM;       /* Stream socket */
    hints.ai_flags = 0;
    hints.ai_protocol = 0;                 /* Any protocol */

    s = getaddrinfo(argv[1], argv[2], &hints, &result);
39  if (s != 0) {
      fprintf(stderr, "getaddrinfo: %s\n", gai_strerror(s));
      exit(EXIT_FAILURE);
    }

44
    /* getaddrinfo() retourne une liste de structures d'adresses.
       On essaie chaque adresse jusqu'a ce que connect(2) reussisse.
       Si socket(2) (ou connect(2)) echoue, on (ferme la socket et on)
```

```
      essaie l'adresse suivante. cf man getaddrinfo(3) */
  for (rp = result; rp != NULL; rp = rp->ai_next) {
    /* Ouverture de la socket */
    sfd = socket(rp->ai_family, rp->ai_socktype, rp->ai_protocol);
    if (sfd == -1)
      continue;
    /* Connexion au distant */
    if (connect(sfd, rp->ai_addr, rp->ai_addrlen) != -1)
      break;              /* Succes */
    close(sfd);
  }
  if (rp == NULL) {      /* Aucune adresse valide */
    perror("connect");
    exit(EXIT_FAILURE);
  }
  freeaddrinfo(result); /* Plus besoin */

  /* Caracteristiques de la socket : taille des buffers */
  /* Taille du buffer de reception */
  lg = sizeof(rsz);
  if ( getsockopt(sfd, SOL_SOCKET, SO_RCVBUF, &rsz, &lg) == 0 )
    printf("SO_RCVBUF: %d octets\n", rsz);
  else
    perror("getsockopt SO_RCVBUF");

  /* Taille du buffer d'emission */
  lg = sizeof(ssz);
  if ( getsockopt(sfd, SOL_SOCKET, SO_SNDBUF, &ssz, &lg) == 0 )
    printf("SO_SNDBUF: %d octets\n", ssz);
  else
    perror("getsockopt SO_SNDBUF");


  /* Initialisation du buffer avec le caractere 'a' */
  memset(buf, (unsigned char)'a', BUFSIZE);

  /* Boucle de communication */
  for(;;) {
    /* Ecriture socket */
    nwrite = write(sfd, buf, BUFSIZE);
    printf("Ecrits %zd octets sur socket\n", nwrite);
  }
}
```

Listing 4 – lazyServerTCP.c

```
/* Auteur : Alain Leroy, Christophe Lohr - Telecom Bretagne */
/*
 * Un serveur sur TCP qui ne fait aucun traitement lors d'une
 * requete d'un client
 */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <netinet/tcp.h>
#include <arpa/inet.h>
#include <string.h>

//#define BUFSIZE 100000

int main(int argc, char **argv) {
  int sfd, ns, s, rsz, ssz;
  struct addrinfo hints;
  struct addrinfo *result, *rp;
  socklen_t lg;
  //ssize_t nread;
  //char buf[BUFSIZE];
  struct sockaddr_storage from;
  socklen_t fromlen;
  char host[NI_MAXHOST];

  /* Le serveur a besoin de connaitre
   * le numero du port sur lequel il attend les requetes du client
   */
  if (argc != 2) {
    printf("Usage : %s  port_serveur\n", argv[0]);
    exit(EXIT_FAILURE);
  }

  /* Construction de l'adresse locale (pour bind) */
  memset(&hints, 0, sizeof(struct addrinfo));
  hints.ai_family = AF_INET6;           /* Force IPv6 */
  hints.ai_socktype = SOCK_STREAM;      /* Stream socket */
  hints.ai_flags = AI_PASSIVE;          /* Adresse IP joker */
```

```
44    hints.ai_flags |= AI_V4MAPPED|AI_ALL; /* IPv4 remapped en IPv6 */
      hints.ai_protocol = 0;                /* Any protocol */

      s = getaddrinfo(NULL, argv[1], &hints, &result);
      if (s != 0) {
49      fprintf(stderr, "getaddrinfo: %s\n", gai_strerror(s));
        exit(EXIT_FAILURE);
      }

      /* getaddrinfo() retourne une liste de structures d'adresses.
54       On essaie chaque adresse jusqu'a ce que bind(2) reussisse.
         Si socket(2) (ou bind(2)) echoue, on (ferme la socket et on)
         essaie l'adresse suivante. cf man getaddrinfo(3) */
      for (rp = result; rp != NULL; rp = rp->ai_next) {
        /* Creation de la socket */
59      sfd = socket(rp->ai_family, rp->ai_socktype, rp->ai_protocol);
        if (sfd == -1)
          continue;
        /* Association d'un port a la socket */
        if (bind(sfd, rp->ai_addr, rp->ai_addrlen) == 0)
64        break;              /* Succes */
        close(sfd);
      }
      if (rp == NULL) {       /* Aucune adresse valide */
        perror("bind");
69      exit(EXIT_FAILURE);
      }
      freeaddrinfo(result); /* Plus besoin */

      /* Positionnement de la machine a etats TCP sur listen */
74    listen(sfd, 5);
      printf("Et le serveur attend ...\n");

      /* Boucle Serveur */
      for(;;) {
79      fromlen = sizeof(struct sockaddr_storage);
        ns = accept(sfd, (struct sockaddr *)&from, &fromlen);
        if (ns == -1 ) {
          perror("accept");
          exit(EXIT_FAILURE);
84      }

        /* Reconnaissance de la machine cliente */
        s = getnameinfo((struct sockaddr *)&from, fromlen,
                        host, NI_MAXHOST, NULL, 0, NI_NUMERICHOST);
89      printf("Machine appellante: %s\n", host);

        /* Caracteristiques des sockets, taille des buffers */
        lg = sizeof(rsz);
        if ( getsockopt(sfd, SOL_SOCKET, SO_RCVBUF, &rsz, &lg) == 0 )
94        printf("SO_RCVBUF: %d octets\n", rsz);
        else
          perror("getsockopt SO_RCVBUF");

        lg = sizeof(ssz);
99      if ( getsockopt(sfd, SOL_SOCKET, SO_SNDBUF, &ssz, &lg) == 0 )
          printf("SO_SNDBUF: %d octets\n", ssz);
        else
          perror("getsockopt SO_SNDBUF");

104     /* Boucle de communication */
        for(;;) {
          sleep(3600); /* Ah... dormir une heure! */
          // nread = read(ns, buf, BUFSIZE); ... et le travail en commentaire :-)
        }
109   }
    }
```

136

Annexe B

Feuille réponse - TP Analyse de protocoles

B.1 Commandes de paramétrage IP des machines terminales

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

B.2 Étude de UDP

1. .....
   .....
2. .....
   .....
3. .....
   .....
4. 
5. .....
   .....
6. .....
   .....
7. .....
   .....

B.3 Étude de TCP

1. .....
.....

2.

3. .....
.....

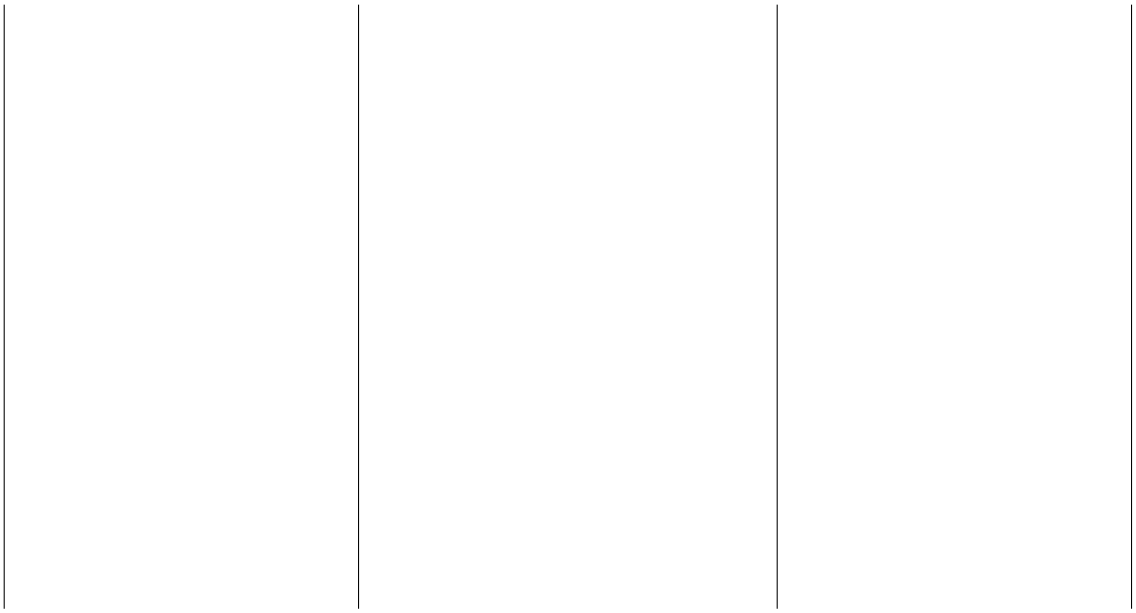
4. .....
.....

5. .....
.....

6. .....
.....

7. .....
.....

8.



B.4 PMTU Discovery

1. 

2. .....
   .....
3. .....
   .....
4. .....
   .....
5. .....
   .....

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Cybersécurité des systèmes maritimes et portuaires

La couche liaison

Christophe Lohr

2021

1 Introduction

Les réseaux locaux 3/75

- Les réseaux de l'entreprise
- Caractéristiques :
  - Topologie
    - Bus, avec ou sans fil
    - Anneau
    - Étoile
    - Bande de base ou large bande
  - Caractéristiques physiques des supports (les média)
    - Cuivre
    - Fibre optique
    - radio
  - Méthode d'accès au médium
    - Comment une station peut-elle émettre ses données sur le réseau

Les différents types de réseaux 4/75

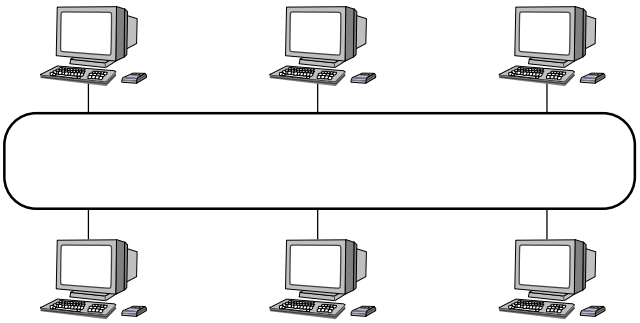
- Les réseaux locaux : LAN : Local Area Network
  - Réseaux d'entreprise
    - Courtes distances : de quelques centaines de mètres à quelques kilomètres
- Les réseaux métropolitains : MAN : Metropolitan Area Network
  - Interconnexion de réseaux locaux
  - Quelques dizaines à quelques centaines de kilomètres
- Les réseaux grande distance : WAN : Wide Area Network
  - Les réseaux nationaux et internationaux
  - Les réseaux d'opérateurs

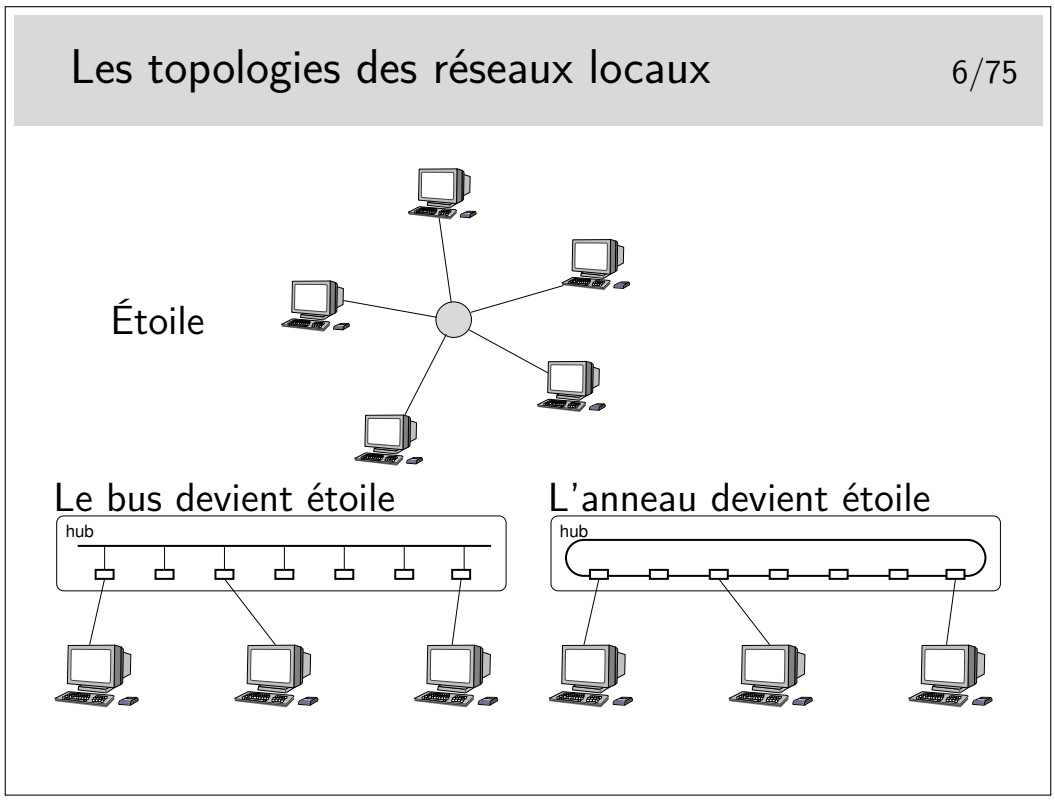
Les topologies des réseaux locaux 5/75

Bus

Remarque : un canal radio partagé peut être considéré comme un bus

Anneau





La topologie en étoile repose sur un nœud actif central qui contrôle le «droit de parole» des stations et achemine l'information. Cette structure n'a jamais vraiment été fortement déployée à grande échelle, on peut noter cependant les réseaux locaux basées sur le protocole X25, au début des années 80 et les ceux basés sur la technologie ATM dans les années 90 (mais c'était pour émuler Ethernet, c'est à dire un bus).

Les topologies à succès sont le bus (Ethernet, qui a gagné la bataille) et l'anneau (Token Ring, d'IBM, en cours d'abandon).

Au cours du temps, le réseau en bus Ethernet a, cependant, évolué vers une structure en étoile, où les nœuds (appelés «hubs») sont des boîtiers électroniques qui émulent un bus. Le «hub» fonctionne comme un bus.

La technologie Token Ring a, elle, été très rapidement implémentée dans des boîtiers électronique renfermant la structure d'anneau.

Aujourd'hui, donc, la topologie est en étoile, mais l'ensemble fonctionne comme un bus (ou encore parfois comme un anneau).

Caractéristiques des canaux 7/75

- Transmission en bande de base
  - Un seul canal physique pour toutes les stations
  - Problème d'accès concurrent
- Transmission en large bande
  - Plusieurs canaux sur le médium
  - Un canal est caractérisé par une bande de fréquence
  - Les signaux émis par les stations sont transposés dans la bande de fréquence qui est assignée aux stations
  - Un canal donné peut être vu comme un canal en bande de base (émulation d'un bus Ethernet par exemple)

Les méthodes d'accès au médium 8/75

Très liées à la topologie (topologie logique)

- Bus
  - Tout le monde partage le canal sans priorité
  - Chaque station peut, à priori, émettre lorsqu'elle le désire
    - Il en résulte des collisions
    - Il faut gérer le problème des collisions
      - Algorithme spécifique de contention (CSMA CD ou CA...)
      - Émulation d'un anneau logique
- Anneau
  - Le droit d'émettre est transmis sur l'anneau dans une trame spécifique contenant un bit spécial appelé «jeton»
  - Les trames circulent dans un sens donné

La standardisation des Réseaux Locaux 9/75

- Menée à bien par l'organisme IEEE
  - Plus particulièrement le comité 802 de l'IEEE
    - Chaque topologie et les divers protocoles et caractéristiques sont étudiés et standardisés par un sous-comité 802
    - Exemples : 802.3 pour Ethernet, 802.5 pour Token Ring
- Modèle en couches spécifique
  - Comparable au modèle OSI pour ses deux premières couches
  - Trois couches
    - La couche physique (comme pour OSI)
    - La couche «Medium Access Control» (MAC)
    - La couche «Logical Link Control» (LLC)

Quelques sous-comités IEEE-802 10/75

- 802.1 : architecture des réseaux locaux
  - Architecture générale, interconnexion (niveau 2), QoS, etc...
- 802.2 : la couche LLC
- 802.3 : Ethernet
  - 802.3u : Ethernet 100Mb/s
  - 802.3ab, z : Ethernet 1Gb/s
  - 802.3ae : Ethernet 10Gb/s
- 802.4 : le bus à jeton
- 802.5 : l'anneau à jeton (Token Ring)
- 802.11 : les réseaux sans fils (WiFi)
- 802.15 : les WPAN : Wireless Personal Area Network

Beaucoup de sous comités :

— 802.1d : techniques de pontage

— 802.1p/q : classes de service, techniques des VLANs (Virtual LAN)

— 802.3u : Ethernet

— 100Mb/s 802.3z : Ethernet 1Gb/s

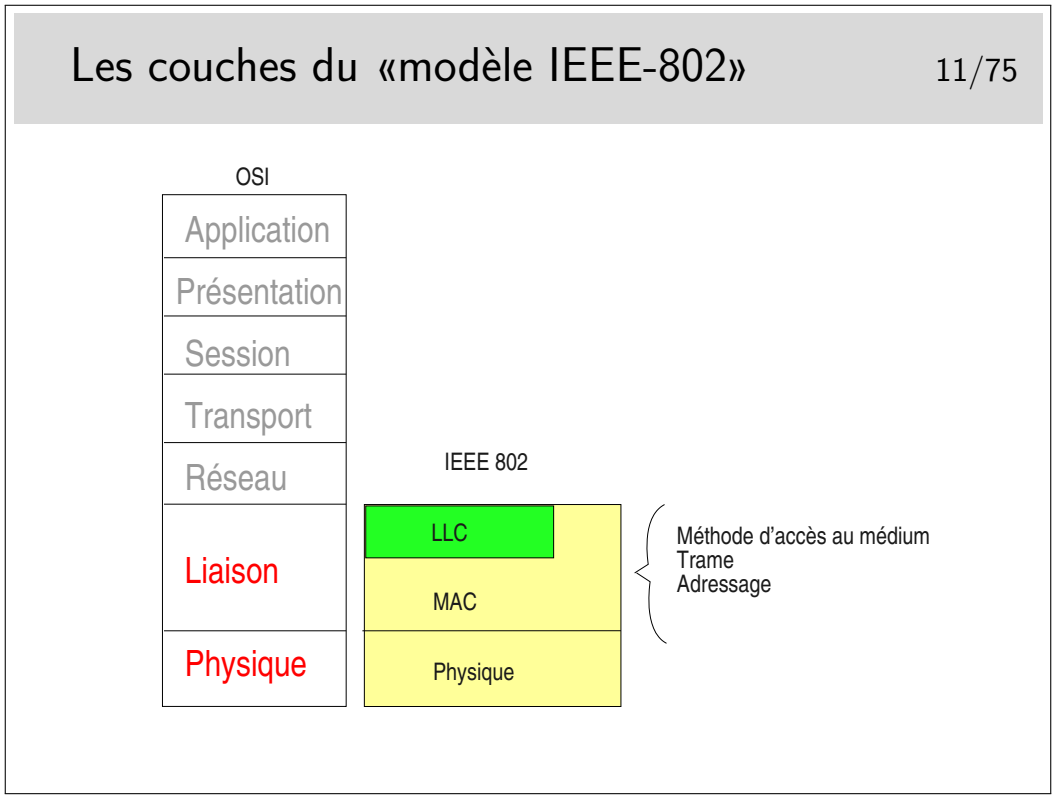
— 802.3ab : Ethernet 10Gb/s

— 802.11a : sans fils, 5Gz (projet)

— 802.11b : sans fils, 11Mb/s

— 802.11g : sans fils, 54Mb/s

Note : WiFi (Wireless Fidelity) est une dénomination commerciale pour le 802.11

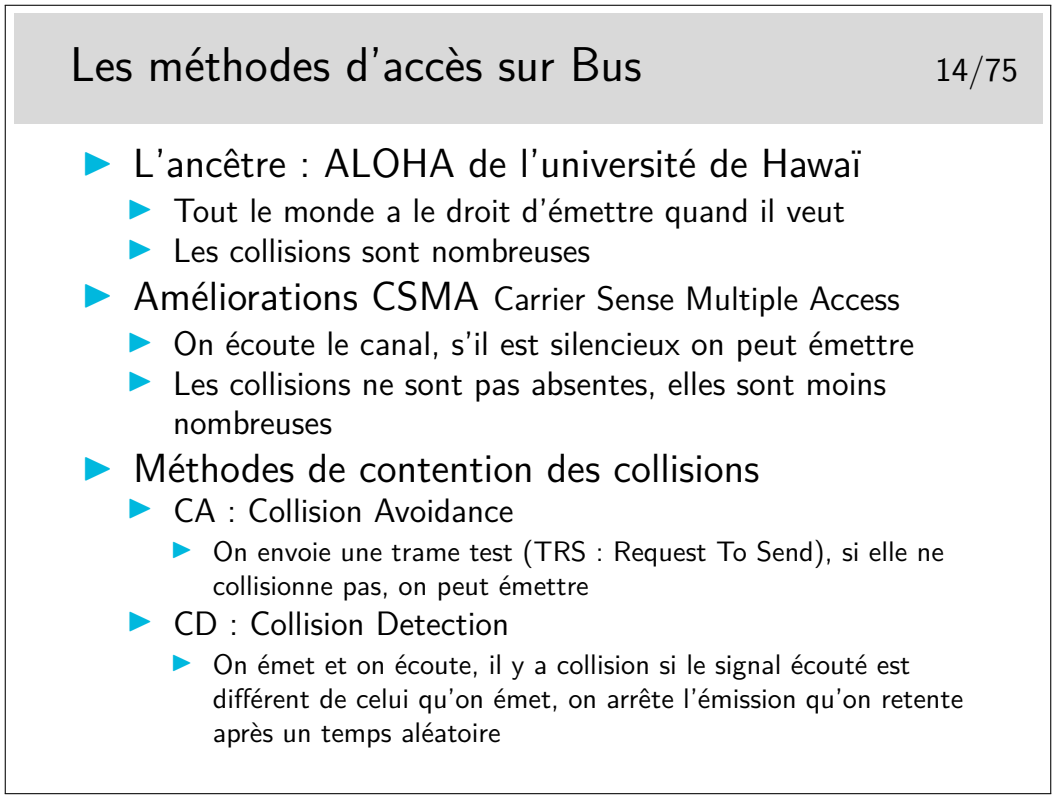


La couche LLC permet de définir différents types de liaisons (avec/sans connexion et avec/sans acquittement). Elle n'est pas toujours obligatoire, en Ethernet elle est optionnelle, certains protocoles de niveau 3 l'utilisent, d'autre pas. Par exemple IP, sur Ethernet, n'emploie pas, par défaut, la couche LLC (mais peut le faire, le choix se fait par paramétrage au niveau du système d'exploitation).

La couche MAC est centrale, elle définit l'algorithme gérant l'accès concurrent au support. Elle définit aussi une structure de trame ainsi qu'un mécanisme d'adressage (les adresses MAC, nous verrons cela plus loin).

2 La technologie Ethernet

2.1 Fondements d'Ethernet



Une autre méthode consiste à créer un anneau virtuel. Les stations s'échangent un jeton dans un ordre donné (la station A passe le jeton à la station B qui le passe à C, etc. La station qui désire émettre doit attendre de voir arriver le jeton). Ce type de réseau a été développé par des industriels des mondes de l'automobile et de l'aviation américains. Il fait partie des «réseaux locaux industriels». Il a été standardisé sous le numéro 802.4 par l'IEEE. Il ne s'est pas développé de manière importante.

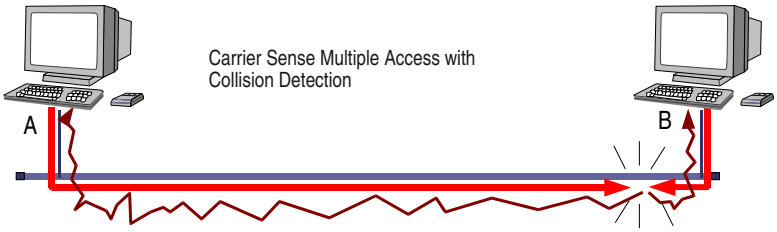
La technologie Ethernet 15/75

- La technologie impérialiste, elle a écrasé toutes les autres (ou les autres s'adaptent à elle, exemple le WiFi)
- Topologie logique : le bus
  - Aujourd'hui, topologie physique en étoile avec hubs et commutateurs (switches)
- Méthode d'accès
  - CSMA-CD : Carrier Sense Multiple Access with Collision Detection
- Origine : Intel, Xeros, Digital, première idée en 1976 (Bob Metcalfe)
- Standardisation générale : IEEE-802.3
- Des débits divers : 10, 100, 1Gb/s, voir 10Gb/s

La technologie WiFi (IEEE-802.11) n'a rien à voir, à priori, avec Ethernet, mais on commence à l'appeler couramment «l'ethernet sans fil», c'est dire... Elle s'adapte, en effet, très bien à Ethernet via des ponts simples à mettre en place.

La technologie Ethernet évolue de manière souple où chaque étape d'évolution ne remet pas en cause les versions antérieures. On peut raccorder une interface 10Mb/s sur une interface 100 ou 1000Mb/s. Les matériels d'interconnexion sont compatibles avec les trois débits.

La méthode d'accès CSMA-CD I 16/75



- La station A écoute le réseau, il n'y a pas de signal, elle peut émettre
- Le signal se propage
- La station B écoute le réseau, le signal de A ne lui est pas encore parvenu, elle décide d'émettre
- Les deux signaux vont collisionner, le signal résultant va se propager de part et d'autre et va parvenir aux deux stations qui continuent d'écouter
- Chaque station continue à émettre quelques instants pour renforcer la collision et s'arrêtent
- Chacune tire un temps aléatoire au bout duquel elles tentent une ré-émission

La méthode d'accès CSMA-CD II 17/75

- Si une station émet pendant au moins 1 RTT alors il n'y aura plus de collision non détectée
- Si une collision est détectée, chaque station en cause arrête son émission
  - Chaque station considère alors 2 intervalles de temps de valeur RTT et tire aléatoirement 1 ou 2 et ré-émet tout de suite (si 1 est tiré) ou un RTT plus tard (si 2 est tiré)
  - Si une nouvelle collision intervient, on considère alors 4 RTT, et on tire aléatoirement entre 1 et 4. On tente une émission au début de l'intervalle de temps tiré
  - En Ethernet, on peut tenter jusqu'à 16 réémissions mais on ne multiplie par 2 que jusqu'à 10 fois le nombre de RTT

Bus Ethernet : quel est le diamètre du réseau ? 18/75

Ou encore : quelle est la distance maximale entre deux stations ?

- Pour détecter une collision il faut que les deux stations soient encore en émission lorsque le signal de collision leur revient
- Cas le plus défavorable : elles sont situées aux deux extrémités du réseau et la station B émet un court instant avant que le signal de A ne lui parvienne. La collision a lieu près de B
  - Pour que A détecte la collision il lui faut attendre un temps égal à la durée de propagation de A jusqu'à B et retour
  - On appelle ce temps le Round Trip Delay, le Round Trip Time (RTT), la «tranche canal» ou encore la fenêtre de collision
  - Ce temps dépend : de la taille du segment de données (trame), de la durée d'émission de cette trame et de la distance entre les deux stations les plus éloignées

Quelle est la dimension du réseau (son diamètre ou encore la distance maximale entre deux stations) si la taille minimale des trames est de 512 bits (64 octets), le débit de 10Mb/s et la célérité du signal sur le câble de 200Km/s (c'est sous évalué...) ?

Vous devez trouver 5,12Km, ce qui est trop. Dans la réalité il faut tenir compte de l'affaiblissement en ligne qui ne permet pas de propager un signal sur une telle distance sans pertes de puissance et distorsion en ligne. Ces phénomènes obligent à placer des organes régénérateurs à des distances bien plus courtes. On appelle ces organes de répéteurs car ils «répètent» les bits, ils les régénèrent. Ce ne sont pas des amplificateurs car alors les distorsions seraient elle-mêmes

amplifiées.

Les premiers réseau Ethernet pouvaient compter des segments de 500m maximum, reliés par des répéteurs.

Les répéteurs apportent un retard, les câbles de raccordement de ces répéteurs au câble principal aussi. Le bilan des délais est tel que la taille maximale est de 2,5km pour les valeurs suivantes :

— taille minimale des trames : 512 bits

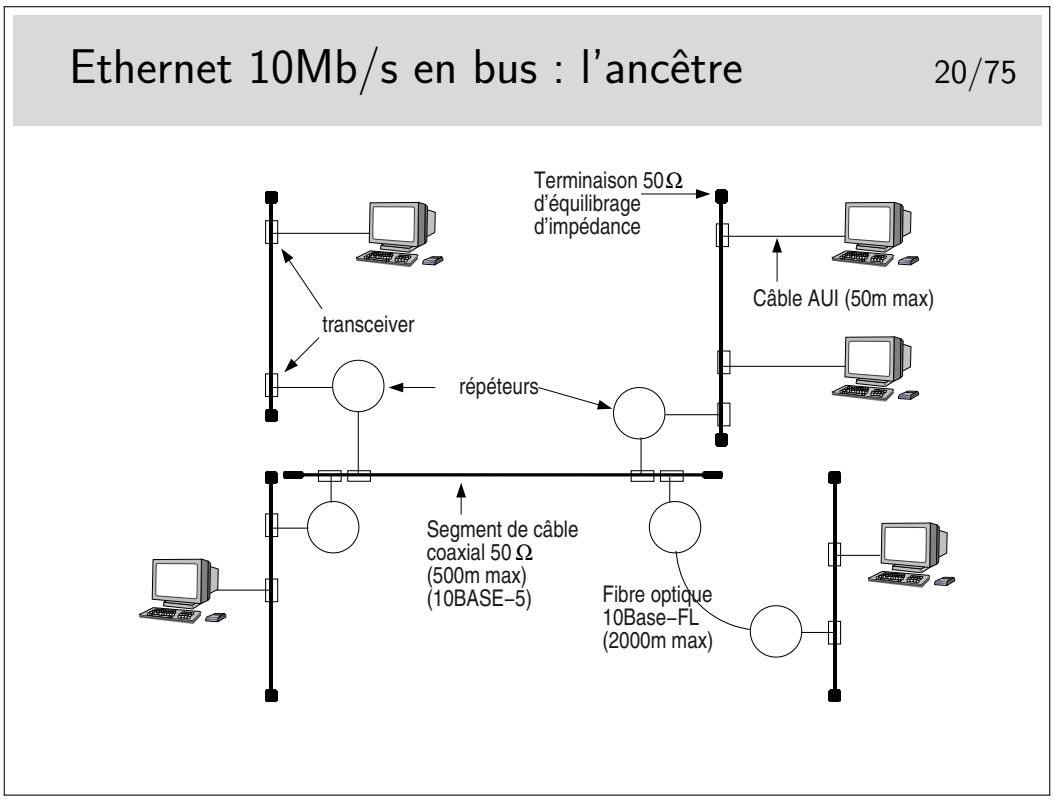
— débit d'émission : 10 Mb/s

— célérité sur les segments de câble principaux : 0,77c

— 5 segments de 500 m maximum et 4 répéteurs maximum entre deux stations.

Ethernet : les caractéristiques générales au débit de 10Mb/s 19/75

- Taille minimale de la trame : 64 octets (512 bits)
- Taille maximale : 1518 octets
  - 1500 octets de charge utile (SDU) pour Ethernet pur
  - 1497 ou 1496 ou même 1492 en format 802.3 où la couche LLC est nécessaire
- Taille minimale : 64 octets
  - Bourrage dans le champ «données» si la longueur de celles-ci est inférieur à 46 octets
- Silence inter-trame de 9,6μs
- Tentatives de réémission en cas de collision : 16



Le médium principal est constitué par des segments de câble 50Ω de 500m maximum, reliés entre eux par des répéteurs. Il ne peut y avoir plus de 5 segments entre deux stations (4 répéteurs). Les câbles sont terminés par un «bouchon» constitué par une résistance de 50Ω permettant d'équilibrer l'impédance caractéristique du support.

Un répéteur est un organe qui «répète» sur tous les ports les bits qui arrivent sur un port. Ce n'est pas une amplification car si cela était les altérations du signal seraient amplifiées. Les bits entrant dans un port sont reconnus et régénérés sur tous les autres ports et même sur les fils émission du port sur lesquels ils arrivent.

Les stations et les répéteurs sont raccordés au câble principal via des câbles de 50m maximum. Le câble arrive sur un organe de raccordement directement fixé sur le médium appelé le «transceiver».

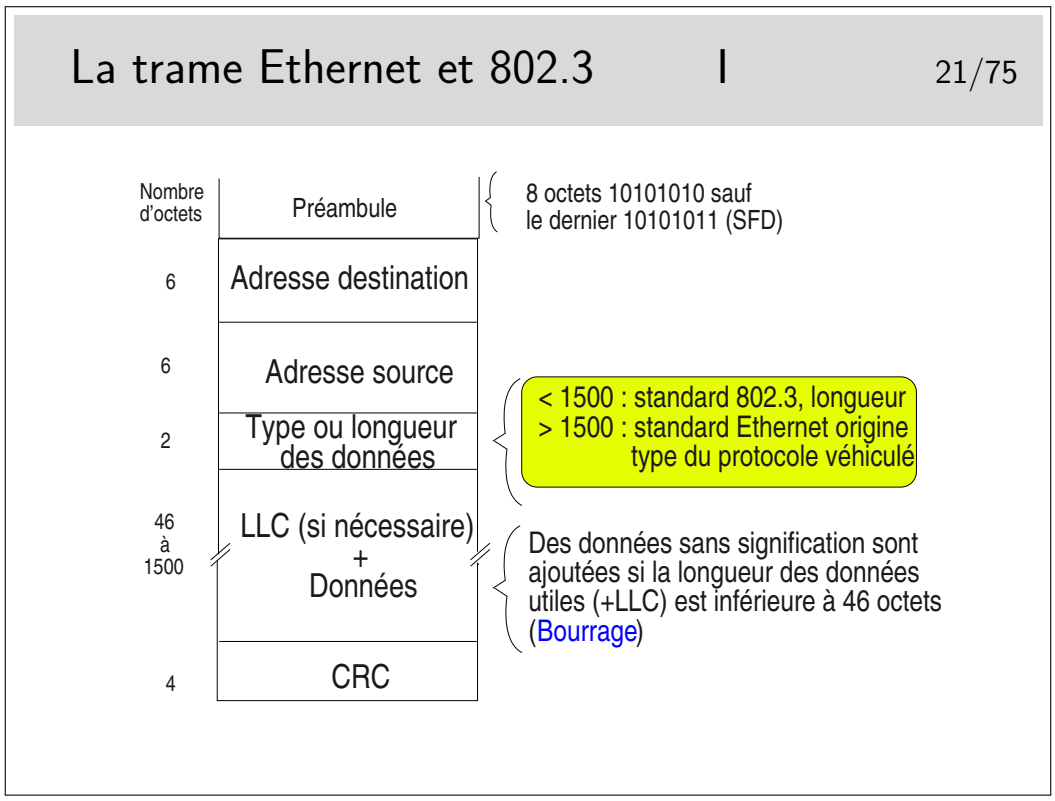
Les transceivers doivent être placés à des distances multiples de 2,5m pour des contrer le phénomène des ondes stationnaires. Un point ou un anneau de couleur noire sur le câble (qui lui est jaune) indique les emplacements possibles.

Deux segments principaux peuvent être reliés par une fibre optique de longueur 2000m maximum. Les deux répéteurs aux extrémités de la fibre ne comptent alors que pour 1.

Ce type de réseau est aujourd'hui abandonné. Les segments de câble sont remplacés par des hubs (ou des switches) et l'architecture physique est devenue ainsi une étoile tout en continuant à fonctionner avec une technique de bus. Le hub peut être comparé à ne boite dans laquelle on aurait enfermé le câble et les transceivers.

Les dénominations 10BASE-5 et 10BASE-FL sont explicitées plus loin.

Il existe une version «cheaper net» comportant des câbles coaxiaux fins (de couleur noire), avec un raccordement par prises de type BNC. Ces câbles font 185m max (10BASE-2).



SFD : Start Frame Delimitor

Les créateurs d'Ethernet (Bob Metcalfe et Intel/Xeros/Digital)) ont défini le champ «Type» pour porter l'identité du protocole véhiculé dans les données (le champ type est le SAP). Pour des soucis d'interopérabilité avec des réseaux locaux dont les trames n'ont pas ce champ type (Token Ring 802.5), le comité 802 a décidé qu'il serait obligatoire d'utiliser la couche LLC pour porter l'identité du protocole véhiculé et a transformé le rôle de ce champ en indicateur de longueur des données. Ce n'est pas une mauvaise idée en soi.

Et en pratique ?...

En pratique les deux coexistent sur les mêmes supports. Une machine peut émettre en 802.3 pour certaines applications et en Ethernet pur pour d'autres.

Par défaut IP est véhiculé en Ethernet pur (type 0800hexa), il peut être véhiculé en mode 802.3 en paramétrant l'interface.

La trame Ethernet II 22/75

- C'est un datagramme
  - Elle contient l'adresse de la station destinatrice ainsi que l'adresse de la station qui émet
  - Elle contient un CRC, on peut donc vérifier son intégrité en réception
    - Si cette vérification montre une altération, la trame est jetée
- Il n'est pas prévu à ce niveau (MAC) d'échange préalable pour établir une relation entre l'émetteur et le récepteur
  - Il n'y a pas de connexion
  - Il 'y a pas de contrôle de flux
  - Il n'y a pas de récupération sur erreur

2.2 Les adresses MAC

Les adresses Ethernet 24/75

- Définies dans la couche MAC (adresses MAC)
  - 6 octets
- 3 types d'adresses
  - Les adresses de stations, dites aussi «unicast» :
    - une adresse unique par interface matérielle (une machine peut avoir plusieurs interfaces matérielles)
    - L'adresse d'une interface est affectée par son constructeur
  - L'adresses globale, dite aussi «broadcast»
    - Permet d'envoyer une trame à toutes les stations du réseau, en une seule opération
  - Les adresses de groupes, dites aussi «multicast»
    - Permettent d'adresser un groupe de stations

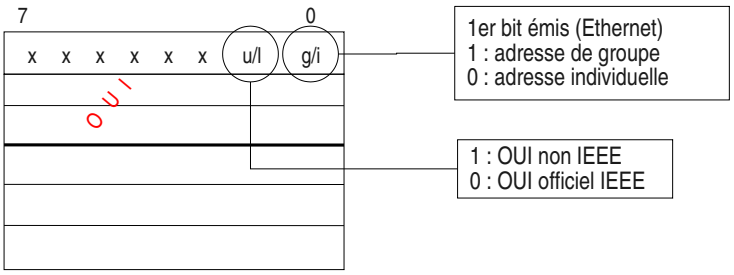
En terminologie anglo-saxonne les interfaces matérielles sont appelées NIC pour *Network Interface Card*.

Une interface voyant passer une trame de broadcast (diffusion) doit prendre en compte cette trame.

Une interface voyant passer une trame de multicast ne la prend en compte que si elle a été paramétrée pour cela.

Format des adresses Ethernet 25/75

- Format IEEE-48
  - 6 octets
    - Les 3 premiers affectés au constructeur par l'IEEE (OUI : Organisation Unit Identifier)
    - Les 3 derniers affectés par le constructeur



Les adresses de braodcast et de multicast 26/75

- Le groupe total : la diffusion ou broadcast
  - `ff:ff:ff:ff:ff:ff`: les 48 bits à 1
- Les groupes restreints : le multicast
  - Le bit de poids faible du premier octet est à 1
  - Quelques exemples :
    - `01:00:5E:xx:xx:xx`: multicast IP
    - `01:80:C2:00:00:00`: spanning tree (protocole de gestion automatique des ponts et commutateurs)
    - `09:00:4E:00:00:02`: multicast IPX

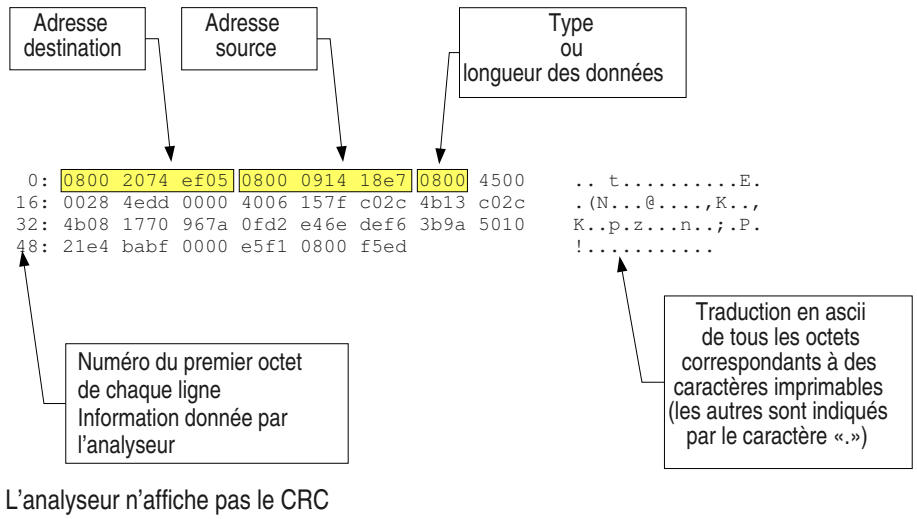
Information sur les OUI et adresses multicast :
`http://standards.ieee.org/regauth/oui/oui.txt`

Exemples d'OUI, d'adresses multicats, de types 27/75

- Quelques OUI
  - 08-00-07 Apple
  - 00-00-0C Cisco
  - 08-00-08 HP
  - 08-00-20 Sun
- Quelques adresses multicast
  - 01-00-5E-xx-xx-xx Multicast internet
  - 01-80-C2-00-00-00 spanning tree

`http://standards.ieee.org/regauth/oui/oui.txt`

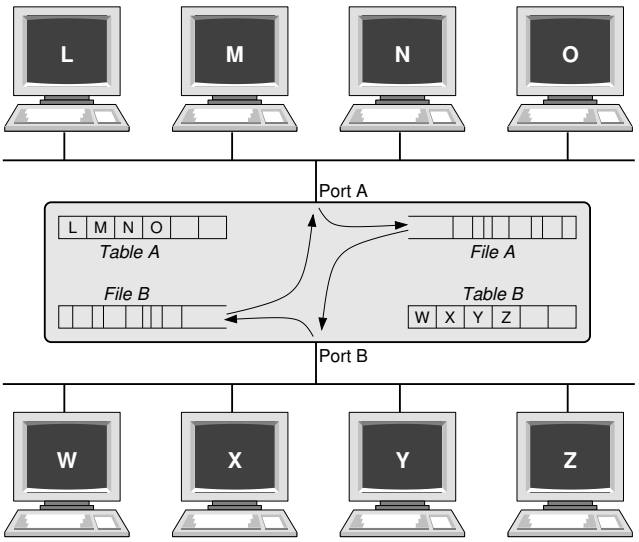
Une trame Ethernet capturée par un analyseur 28/75



L'analyseur n'affiche pas le CRC

Cette trame est au format 802.3 ou Ethernet pur ?

Interconnexion de réseaux locaux Ethernet : les ponts 29/75



Le pont fonctionne tout seul sans paramétrage préalable. Il écoute le réseau sur son port A et son port B et «voit» passer des trames. Il enregistre les adresses sources de celles-ci et détermine ainsi que L, M, N et O sont du coté A. Ces adresses sont «apprises» par le pont et enregistrées dans sa table A. De même il «apprend» que, coté B, existent les stations W, X, Y et Z. Ces dernières adresses sont stockées dans la table B.

Si une trame est émise par L à destination de O, il reconnaît que ces deux stations sont du coté A, il ne fait rien. De même pour des trames de M vers N ou X vers Z par exemple.

Par contre si une trame est émise de A à destination de Z, alors cette trame est enregistrée dans la file A puis une tentative d'émission de celle-ci sera effectuée coté B. La trame ne sera mémorisée dans la file A que si elle n'a pas collisionné. Elle pourra peut-être collisionner coté B lors de sa ré-émission de ce coté mais la collision ne sera pas détectée coté A.

Si le pont ne connaît pas la station destinatrice (elle n'a rien émis encore) il retransmet les trames vers cette destination sur le port opposé de celui sur lequel il reçoit ces trames.

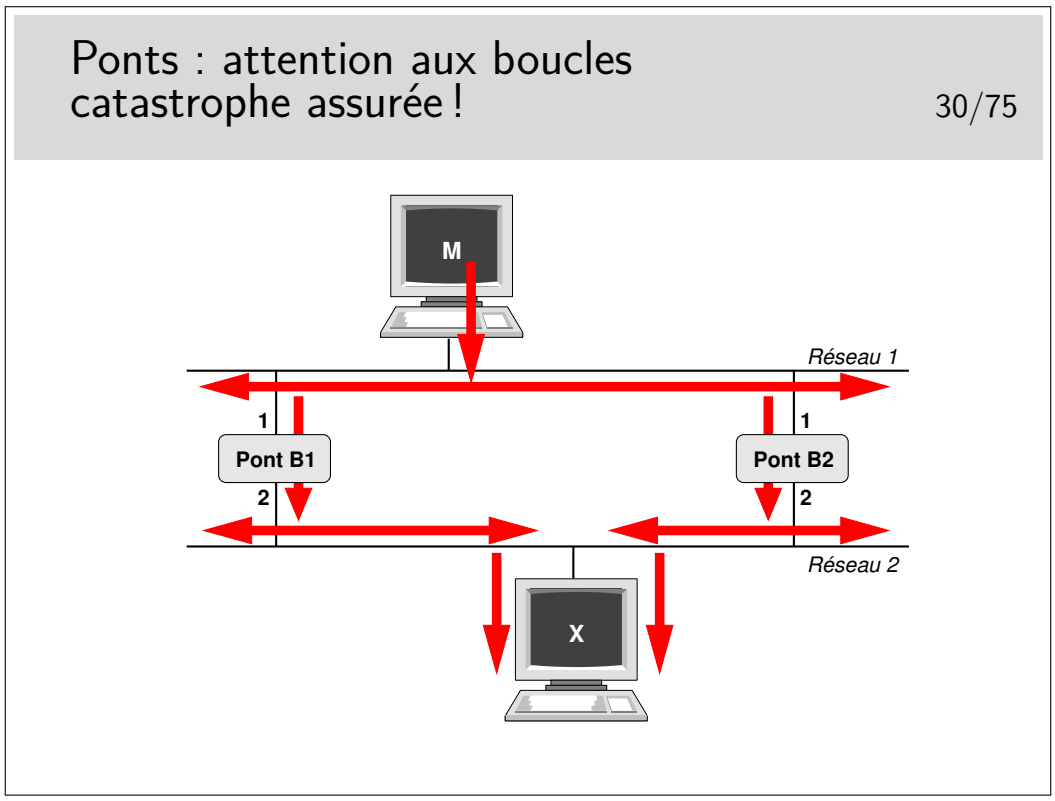
Avantages :

— le trafic est segmenté

— la connectivité totale est est maintenue

— le réseau est divisé en «domaines de collisions»

— le diamètre du réseau peut être doublé



Pour des raisons de fiabilité on peut être amené à doubler les ponts entre deux réseaux. Se pose alors le problème de boucles comme l'exprime l'exemple présenté ici.

La machine M émet une trame à destination de la machine X.

Premier point négatif : multiplication des copies de trames...

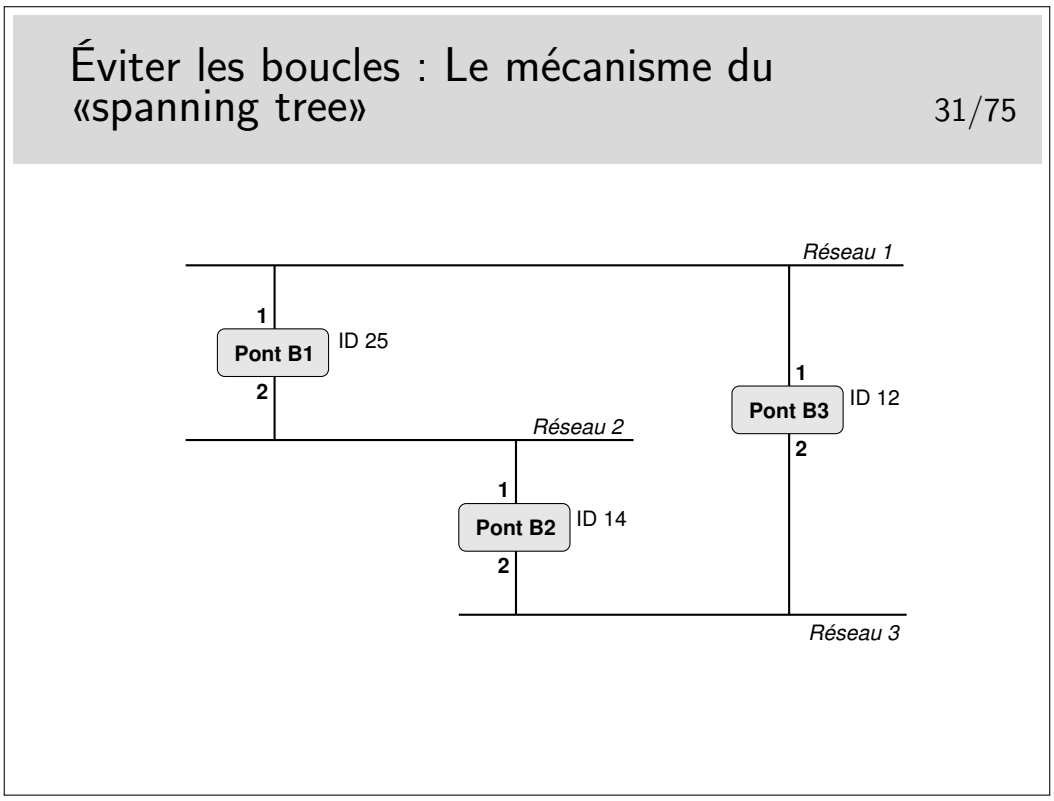
Le pont B1 enregistre la trame et la réémet sur son port 2. La trame arrive à destination. Le pont B2 fait de même. Une seconde copie arrive à destination.

Second point négatif : les trames font des boucles...

Si la station X n'a rien émis encore les ponts ne la connaissent pas. La station M émet une trame vers X. Les ponts ne connaissant pas X retransmettent cette trame comme précédemment. Il y a donc deux copies. Mais...

La copie faite par B1 arrive à destination et aussi en B2 port 2. B2 croit alors que la machine M a changé de réseau, elle est maintenant en bas. Comme il ne connaît pas X il recopie cette trame sur son port 1. Cette trame sera vue par B1 port 1 qui va la relayer vers son port 2 et va à nouveau arriver en X puis en B2 port 2 et ainsi de suite. Mais une trame tournera aussi dans le sens contraire car au début la copie faite par B2 arrive à destination.. etc... etc... Une seule trame émise suffit alors à saturer le réseau.

Pour éviter ce phénomène il faut transformer le réseau. De graphe avec des boucles il faut le transformer en arbre complet, en anglais en «spanning tree».

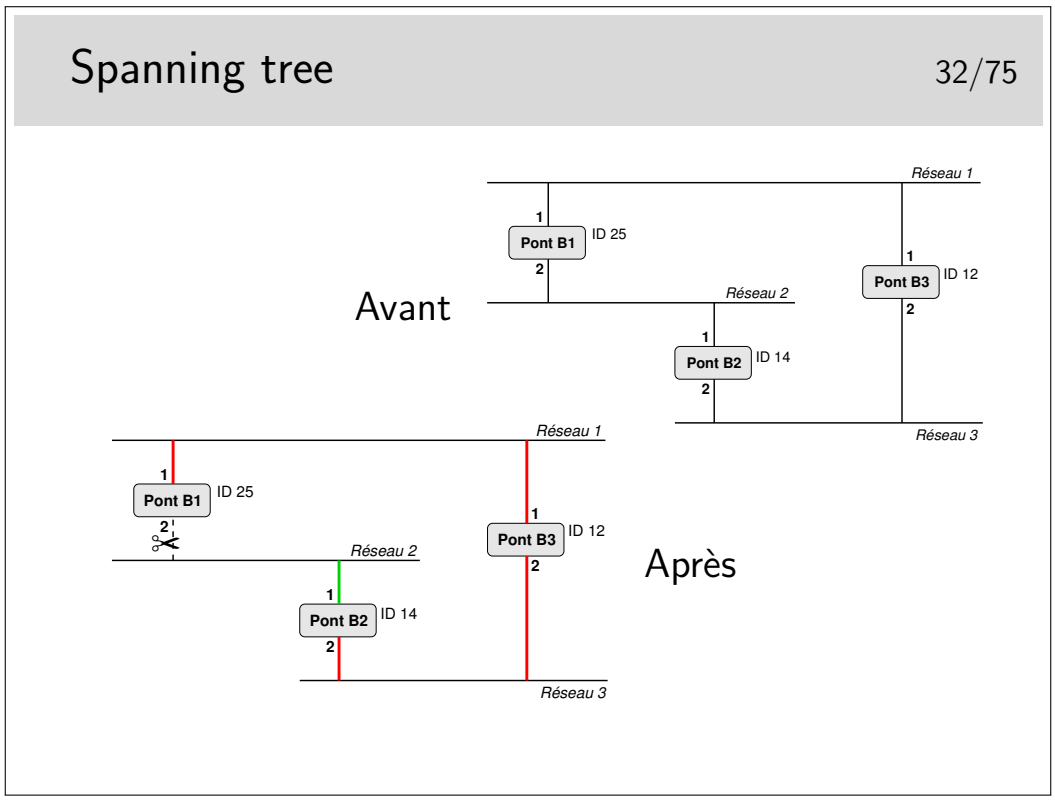


Les ponts (ou les commutateurs Ethernet) administrables peuvent mettre en œuvre cet algorithme. Sur chacun de ses ports, chaque pont annonce un identificateur qui lui est propre via une trame de type multicast (seuls les ponts sont configurés pour prendre en compte ce type de trame). Cet identificateur signifie en quelque sorte «c'est moi le pont racine». Tous les ponts commencent par annoncer qu'ils sont racine. Lorsqu'un pont reçoit un message comportant un identificateur «inférieur» au sien (on dira «meilleur»), il s'aperçoit qu'il n'est pas «racine» mais qu'un autre pont l'est plus certainement et que celui-ci est accessible via l'interface par laquelle est parvenue ce message «meilleur».

L'ensemble des ponts exécute le même algorithme, celui-ci converge finalement car un seul pont se trouve être le «meilleur», la racine (celui dont l'identificateur est le plus petit). Chaque pont sait par quelle interface atteindre le pont racine par le «plus court chemin», seule cette interface reste active pour le trafic de données, les autres interfaces permettant, elles aussi d'atteindre la racine, mais avec un chemin plus long sont inhibées pour le trafic de données. Les boucles sont ainsi évitées.

Les messages échangés s'appellent des BPDU (Bridge_PDU) et comportent les informations suivantes : `<id_pont_supposé_racine, coût_supposé, id_pont_émetteur, port_émission>`.

Les messages du protocole Spanning tree sont portés par des trames en adressage multicast. Leur adresse destination est (en standard) `01:80:c2:00:00:00`. Ce sont des trames de type 802.3. Les DSAP et SSAP de la couche LLC supérieure (voir plus loin) sont égaux à `0x42`. Les messages sont émis toutes les 2 secondes par défaut. Les identificateurs, les délais sont généralement paramétrables.



B1 émet : <25,0,25,1> sur son port 1 et <25,0,25,2> sur son port 2.

Il reçoit <12,0,12,1> par son port 1 et <14,0,14,1> par son port 2. Il constate qu'il n'est pas racine et que la racine doit être 12 (B2) accessible par son port 1. B1 considère que son port 1 est «root port».

Il émet alors : <12,1, 25,1> sur son port 1 et <12,1,25,2> sur son port 2.

B2 émet <14,0,14,1> sur son port 1 et <14,0,14,2> sur son port 2.

Il reçoit : <12,0,12,2> port 2 et <25,0,25,2> port 2 dans une première phase indiquant ainsi que le pont racine doit être accessible par son port 2 plutôt que par son port 1. B2 considère que son port 2 est «root port».

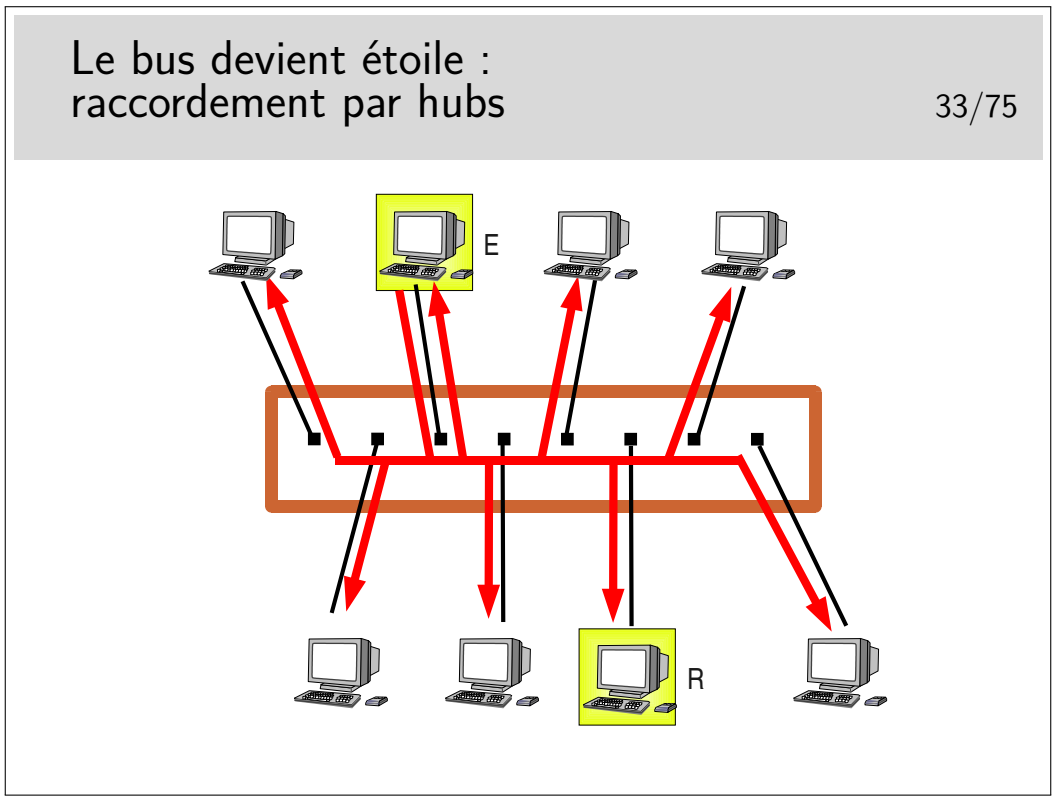
Il émet alors : <12,1,14,2> sur son port 2 et <12,1,14,1> sur son port 1.

B3 ne reçoit pas de «meilleure» configuration que celles qu'il émet. Il est donc la racine.

Comment vont alors se départager B1 (port 2) et B2 (port 1) ?

B1 reçoit port 2 : <12,1,14,1> ce qui est meilleur que ce qu'il émet sur ce même port <12,1,25,2>. Il inhibe son port 2 (pour le trafic de données, pas pour les messages de spanning tree). B2 reçoit sur son port 1 un message «moins bon» que ce qu'il émet sur ce même port, donc B2 port 1 reste actif.

Excellent tutorial animé à :
http://www.cisco.com/warp/public/473/spanning_tree1.swf



On prend le câble et les transceivers, on ramasse le tout dans un petit boîtier et le tour est joué. D'une topologie en bus nous passons à une topologie en étoile.

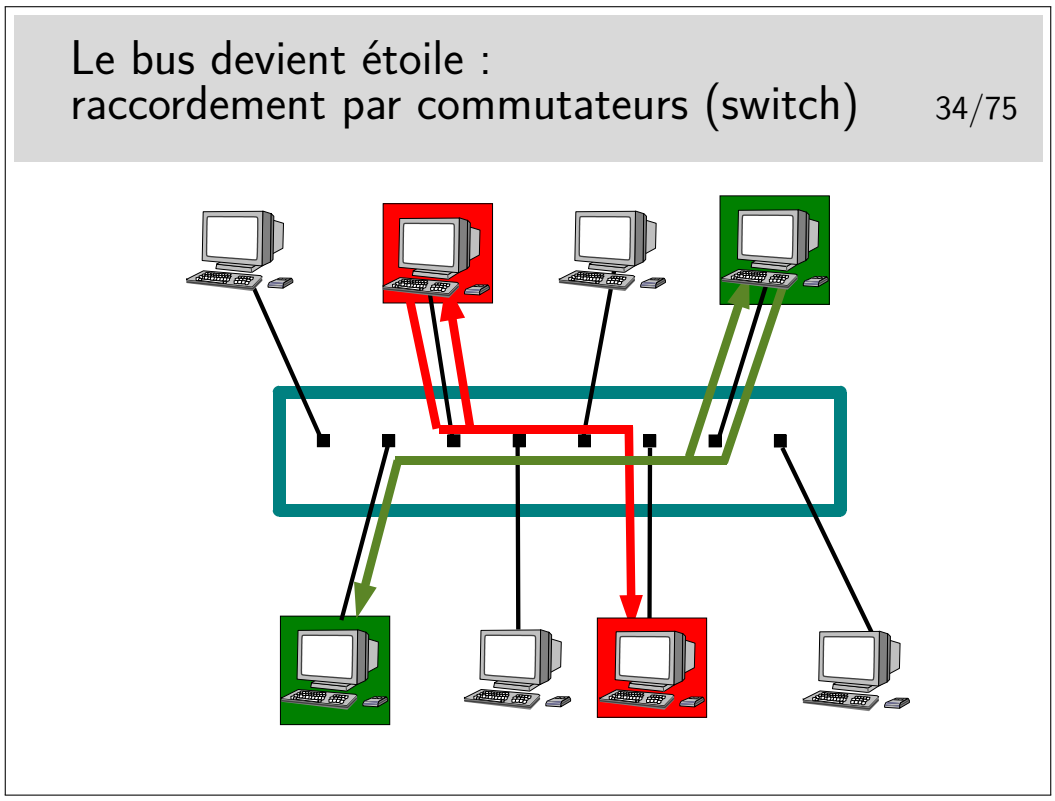
En fait ce n'est pas si simple car les raccordement changent, ce ne sont plus des transceivers mais des prises de type RJ45 et les câbles sont de type 10/100 BASE-T, à paire torsadée, de longueur max 100m.

Exemple de fonctionnement : La station E émet vers la station R. Toutes les stations reçoivent le signal, même E pour des raisons de détection de collision.

Un hub se comporte donc comme un bus.

Il est très facile d'espionner tout ce qui passe sur le hub à partir d'une machine raccordée sur un des ports.

Les hubs sont des **répéteurs**. Ce sont des organes de la **couche physique** (selon le modèle OSI), ils ne s'intéressent qu'au niveau bit.



Le commutateur ressemble à un hub mais se comporte comme un pont. Il connaît les adresses (MAC) des machines qui sont raccordées sur chacun de ses ports (raccordées directement ou via d'autres d'autres commutateurs ou hubs). C'est un pont multiport.

Il ne recopie une trame que sur le port qui mène vers la destination.

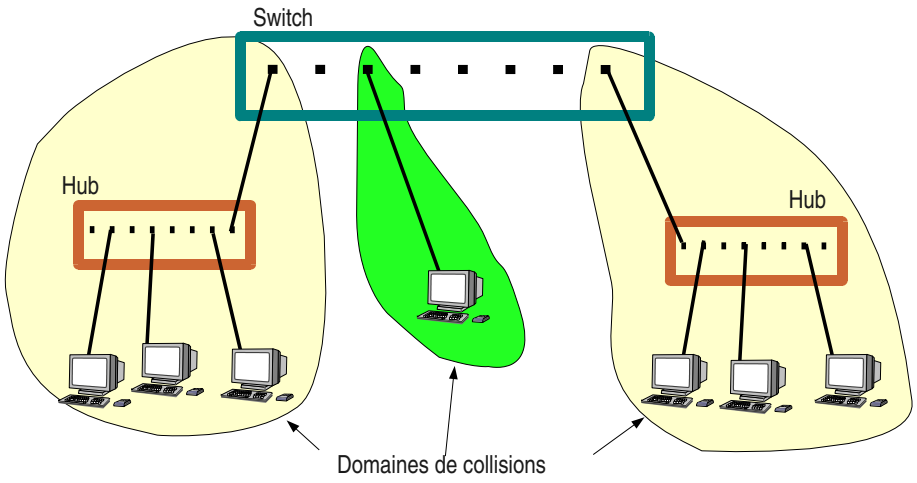
Il peut faire ce travail pour plusieurs trames simultanément à condition qu'elles n'aillent pas vers le même port de sortie, auquel cas elles sont traitées les unes après les autres. Il se comporte donc comme s'il était capable d'établir des circuits entre deux ports pour de cours instants, d'on son nom de commutateur (switch en anglais).

C'est une **commutation de niveau 2** (plus exactement de niveau MAC), à ne pas confondre avec la commutation de niveau 3 qu'on rencontre dans d'autres réseaux comme X25 ou ATM.

L'espionnage sur ce type d'appareil est plus difficile, sur un port donné on ne peut voir que les trames destinées à ce port ou les trames de *broadcast*, l'intérêt pour «l'espion» est donc très limité. Le «malfaisant» persévérant et cultivé en Réseaux peut cependant trouver des solutions...

Domaines de collisions 35/75

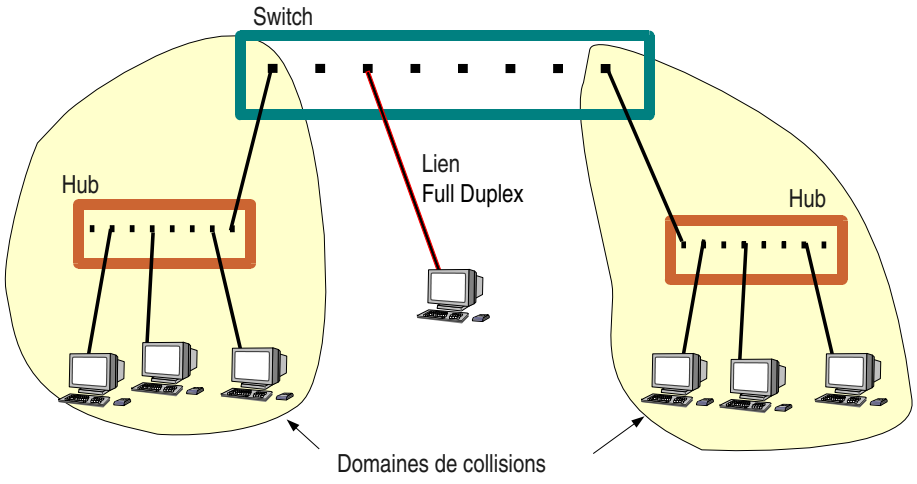
Un switch est un pont multiports



Un switch se comporte comme un pont. Sur chacun de ses ports il enregistre les adresses sources des trames qu'il voit passer. Chaque port se comporte comme un port de pont. Chaque port délimite donc un domaine de collision.

On voit sur la figure ci-dessus qu'un de ces domaines ne comporte que deux machines (un port du switch et une machine seule), Voir le transparent suivant pour une utilisation plus efficace du lien en question.

Liens full duplex 36/75



Lorsqu'une machine terminale est reliée directement à un switch le domaine de collision est restreint à la machine et au port du switch sur lequel elle est raccordée. Si on considère en plus que le lien physique de raccordement est un câble comportant un circuit différent pour chaque

sens (deux fils par pour l'émission, deux autres fils pour la réception, ou deux fibres optiques), il devient alors intéressant d'inhiber le mécanisme de détection et contention de collision pour tirer partie pleinement de la paire émission et de la paire réception présente dans le câble de raccordement en les utilisant simultanément pour véhiculer des données utiles.

On peut ainsi passer de 10/100/1000 Mb/s à l'alternat à 10/100/1000 Mb/s dans chaque sens simultanément (10 ou 100 ou 1000 MB/s selon le matériel).

Attention, en général il faut gérer le full duplex, c'est à dire qu'il faut se connecter au switch pour l'administrer (sur son port console via un PC et l'application HyperTerminal ou via telnet en IP) et configurer les ports qu'on désire voir fonctionner dans ce mode. Il faut aussi vérifier sur la machine terminale qu'elle est bien en full duplex et, éventuellement, la forcer dans ce mode. Si le full duplex n'est pas positionné des deux cotés il en résultera un fonctionnement très ralenti, une des deux extrémités détectant alors des collisions qui n'en sont pas.

2.3 Les VLANs

Le concept de LAN virtuel ou VLAN I 38/75

- Un VLAN est construit à l'aide des commutateurs dont on restreint les possibilités de commutation à des groupes de ports, la commutation peut être totale entre les membres d'un groupe mais devient impossible entre les membres de groupes différents

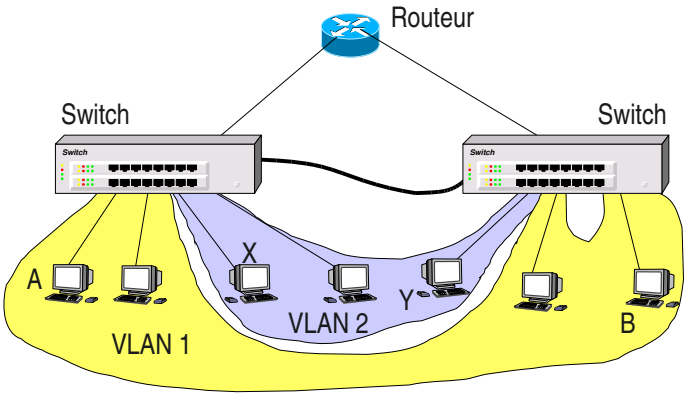
Un groupe définit un domaine de broadcast
domaine de broadcast ⇔ VLAN (plus petite commune définition)

- suivant les possibilités matérielles des switches, un VLAN peut être défini par port, par adresse MAC, par adresse IP (dans ce dernier cas, ce n'est toutefois pas assimilé à de la commutation niveau 3)

Le concept de LAN virtuel ou VLAN II 39/75

- routeur obligatoire entre VLANs, même s'il n'y a qu'un commutateur
- l'attribution d'un élément (port, adresse MAC, adresse IP) à un VLAN est réalisée par une opération de gestion
- un VLAN peut être réparti sur plusieurs commutateurs reliés entre eux
- l'interconnexion de switchs impose de marquer les trames sur les liens d'interconnexion afin de les associer à un VLAN. Exemple figure suivante : les trames échangées entre A et B ou entre X et Y doivent être différentiées sur le lien entre les deux switches. Solution : étiquetage des trames ; normes IEEE et solutions propriétaires

Les LANs virtuels : VLANs 40/75



- Sur un commutateur ou plusieurs interconnectés
  - On subdivise l'espace d'interconnexion en sous espaces étanches
  - On construit ainsi une architecture logique sur une topologie physique
  - Standardisation : IEEE-802.1p et 1q

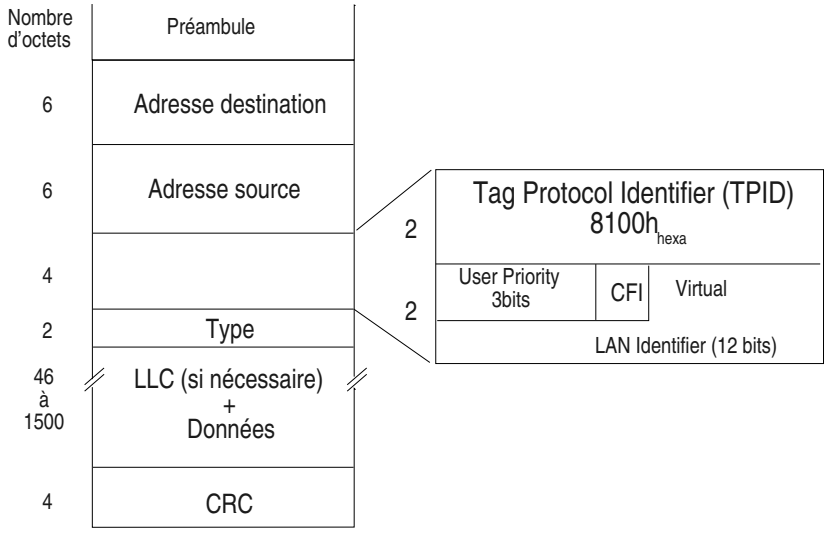
Imaginons que la machine A veuille émettre vers toutes les machines de son VLAN, elle doit émettre une trame de *broadcast* qui devra être relayée par le switch de gauche vers le switch de droite. Cependant, le switch de droite devra connaître l'identité du VLAN vers lequel diffuser la trame. Il ne faudra pas, par exemple, émettre vers la machine Y du VLAN2. Pour cela, sur le lien entre les switches, la trame devra être complémentée par un champ porteur de l' identité du VLAN.

Le transparent suivant indique comment est modifiée la trame ethernet de base.

Le lien d'interconnexion des commutateurs est parfois un «trunk». Selon les constructeurs et les types de switches, des ports spécifiques peuvent être réservés pour le «trunking».

La trame Ethernet revisitée par IEEE-802.1p/q

41/75



Le champ TPID est tout simplement une valeur de champ Type spécifique. Le champ «User Priority» apporte une possibilité de privilégier certains flux.

Le bit CFI (Canonical Format Indicator), s'il est à 1, indique que le champ information comporte des indications de routage par la source.

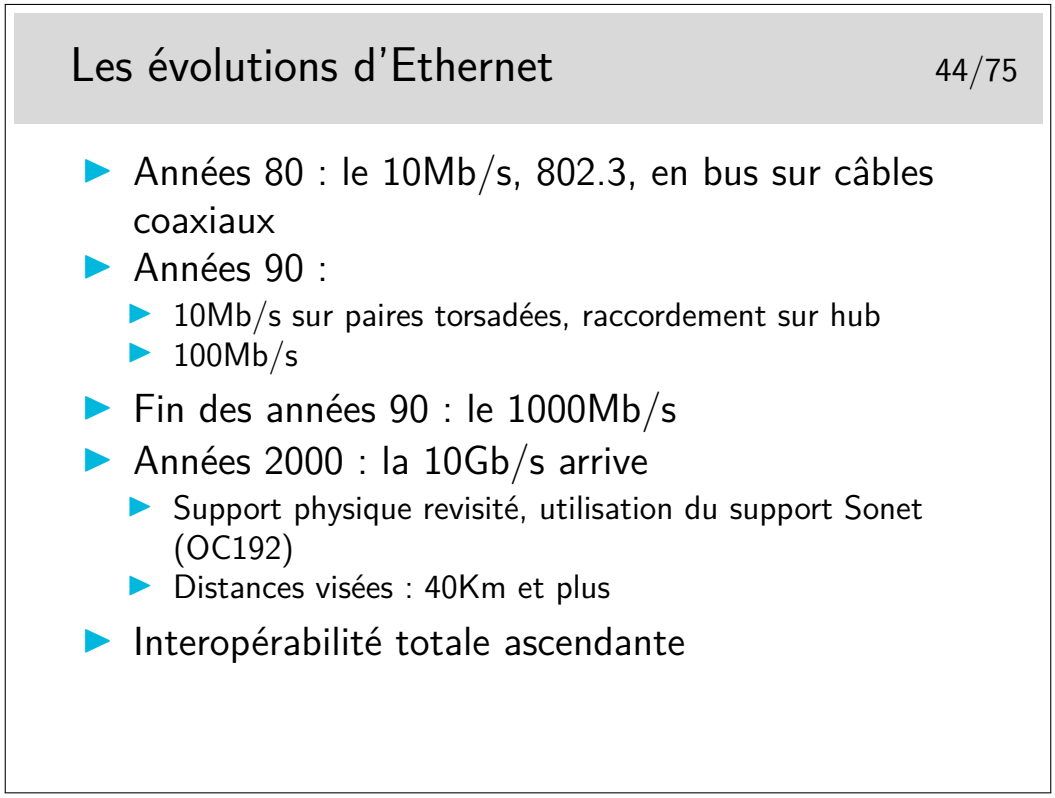
Le champ «Virtual LAN Identifier» indique à quel VLAN appartient la trame.

Compléments du standard 802.1p/q

42/75

- GARP : Generic Attribute Registration Protocol
  - mécanisme de signalisation permettant aux stations de fournir des indications (par des valeurs d'attributs) affectant les paramètres de filtrage des switches.
  - 3 attributs déjà définis :
    - groupe d'adresses MAC, facilite les mécanismes du multicast,
    - mode de filtrage des ports,
    - VLAN
- GMRP : Garp Multicast Registration Protocol
  - mécanisme permettant aux stations terminales et aux switches de s'enregistrer (et se retirer) comme participants à un groupe multicast et de diffuser cette information à l'ensemble de switches du réseau,
  - les switches utilisent cette information comme paramètre de filtrage,
  - utilise GARP comme protocole support.

2.4 Évolutions



Sonet (Synchronous Optical Network) est un type de multiplexage mis au point par la société américaine Bellcore et standardisée par l'ITU-T sous l'appellation SDH (Synchronous Digital Hierarchy).

À l'origine, Sonet/SDH est destiné au transport des voies téléphoniques numériques. Les concepts mis en œuvre permettent un multiplexage et un démultiplexage facilité par rapport aux techniques plus anciennes. C'est le support privilégié pour les réseaux de type ATM.

Les débits courants sont de 155Mb/s (Sonet : 0C3, SDH : STM1) sur cuivre (100m) ou fibre optique et 622Mb/s. Le débit de 1,2 Gb/s est possible. Un bond technologique est réalisé pour l'Ethernet 10Gb/s.

Le câblage cuivre, les différents types de câbles 45/75

- Le standard : ANSI/TIA/EIA 568 B
- Câbles à 4 paires torsadées, connecteurs RJ45
- Catégories
  - 5 : 100MHz, 100Mb/s sur 100m
  - 5e : 100 Mhz
  - 6 : 250MHz : 1Gb/s sur 100m
  - 7 : 600MHz : 10Gb/s sur 100m (15Gb/s sur 15m)
- Blindés ou écrantés
  - UTP : Unshieldes Twisted Pair : non blindé
  - FTP : Foilded TP : écranté
  - STP : Shielded TP : blindé
  - SFTP : écranté et bindé

Un lien : `http://cablingdb.com/GlossaryPages/GlossaryC/CGlossary.asp`

2.5 Dénomination

Les dénominations des différentes versions d'Ethernet I 47/75

- Format : x BASE/BROAD y
  - x : le débit
  - BASE : bande de base, BROAD : large bande
  - y : indication sur la topologie (et/ou la longueur du câble)
- Bande de base :
  - Le 10 Mb/s
    - 10 BASE-5 : 10 Mb/s, topologie en bus constitué de segments de 500m
    - 10 BASE-2 : 10Mb/s, topologie en bus constitué de segments de 185m
    - 10 BASE-T : 10Mb/s, topologie en étoile, câbles en paires torsadées (T pour *Twisted pair*), longueur 100m

Les matériels à 10Mb/s tendent à disparaître aujourd'hui (2004). Les cartes interfaces sont toutes en 100-BASE-T, même en premier prix.

Les dénominations des différentes versions d'Ethernet II 48/75

- Le 100Mb/s
  - 100BASE-T4 : 4 paires utilisées, catégorie 3 à 5
  - 100BASE-TX : 2 paires torsadée, catégorie 5
  - 100BASE-FX : 2 fibre optique
- Le 1000Mb/s
  - 1000BASE-LX : fibre optique, grande (Long) longueur d'onde
  - 1000BASE-SX : fibre optique, courte (Short) longueur d'onde
  - 1000BASE-CX : paire torsadée, 25m max
  - 1000BASE-TX : 4 paires torsadées de catégorie 5

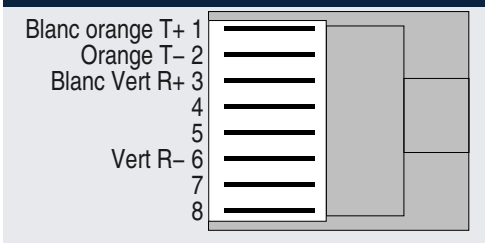
2.6 Câblage

Les câblages : les câbles catégorie 5 et leurs connecteurs 50/75

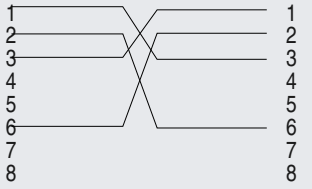
- Les câbles sont classés par catégories en fonction de leurs caractéristiques (en particulier les débits maximum
- Aujourd'hui on recommande la catégorie 5 voire 5e ou 6 pour des câblages cuivre permettant d'atteindre le 100Mb/s

Les câbles pour le 10BASE-T et 100BASE-T (EIA/TIA 568 A)

La prise RJ45



Câble croisé (interconnexion de machines hubs switches)



Un document bien illustré sur le câblage... En italien, mais Mhz et nm et Km s'écrivent comme en français... `http://www.garr.it/ws4/pdf/Montessoro.pdf`

Un câble croisé permettra de raccorder :

— deux machines directement entre-elles

— deux hubs ou deux swiches entre-eux

— un hub et un switch

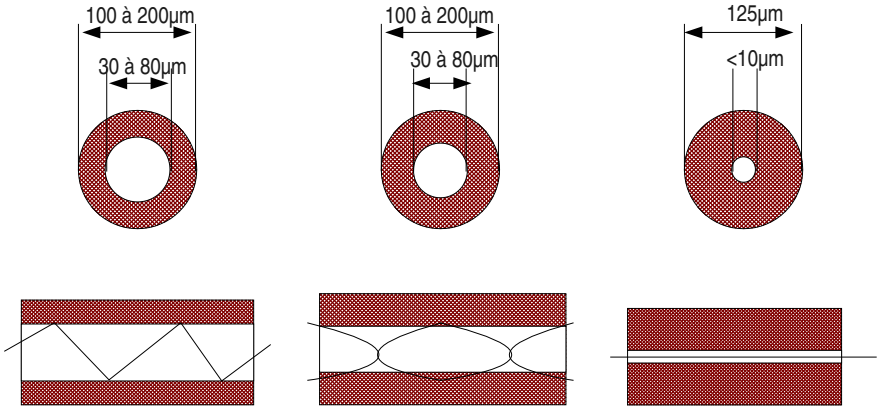
Les hubs et les switches peuvent être munis de ports directement croisés, on les repère généralement par la mention «Up Link». Un port peut aussi avoir deux prises RJ45, und roite et une croisée «up-link».

Un port peut être muni d'un petit commutateur appelé **MDI-MDIX** permettant de positionner le port en mode normal (MDI) ou en mode croisé (MDIX).

Et enfin, la fonction MDI-MDIX peut être à auto-détection : le switch détecte automatiquement s'il a à faire à un câble droit ou croisé.

Les câblages : la fibre optique 51/75

Multimode Gradient d'indice Monomode



La fibre et Ethernet gigabit 52/75

Transceiver	Fiber Diameter (microns)	Bandwidth (MHz*km)	Minimum Range (meters)
1000BASE-SX	MM 62.5 MM 62.5 MM 50 MM 50	160 200 400 500	2-220 2-275 2-500 2-550
1000BASE-LX	MM 62.5 MM 50 MM 50 SM 9	500 400 500 NA	2-550 2-550 2-550 2-5000

Transceiver	Fiber Diameter (microns)	Wavelength (nm)	Minimum Range (meters)
1000BASE-LH (Extended distance)	SM 9	1310	1 km - 49 km
1000BASE-LH (Extended distance)	SM 9	1550	50 km - 100 km

- 1000 BASE-SX : (S pour Short), $\lambda = 850nm$, codage en ligne 8B/10B
- 1000 BASE-LX : (L pour long), $1270nm < \lambda < 1355nm$, codage en ligne 8B/10B
- 1000 BASE-LH (Long Haul), fibre mono mode (SM : Single Mode), 9μ

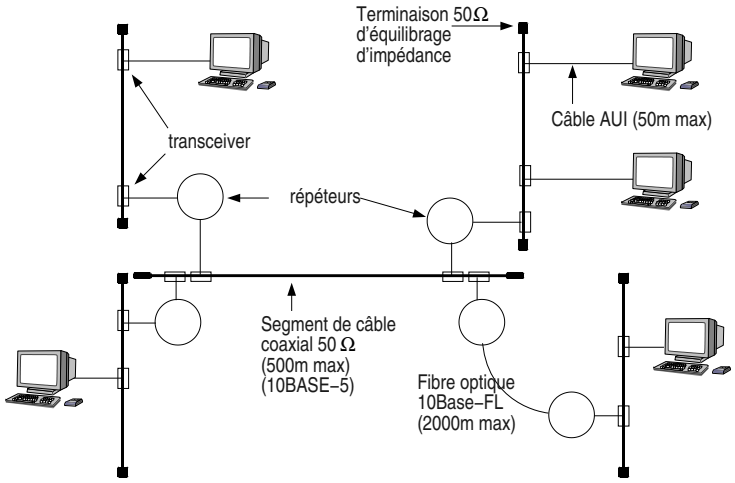
Interopérabilité des débits 53/75

- Les hubs et les switches acceptent aujourd'hui les deux débits de base 10Mb/s et 100Mb/s
- Les switches comportant des ports à 1Gb/s se généralisent
  - Un mécanisme d'autonégociation entre les extrémités dce la liaison permettent de détecter le débit maximal accepté (10 ou 100, 1000)
  - L'autonégociation permet aussi de détecter la possibilité du full-duplex et de contrôle de flux

La fonction contrôle de flux 54/75

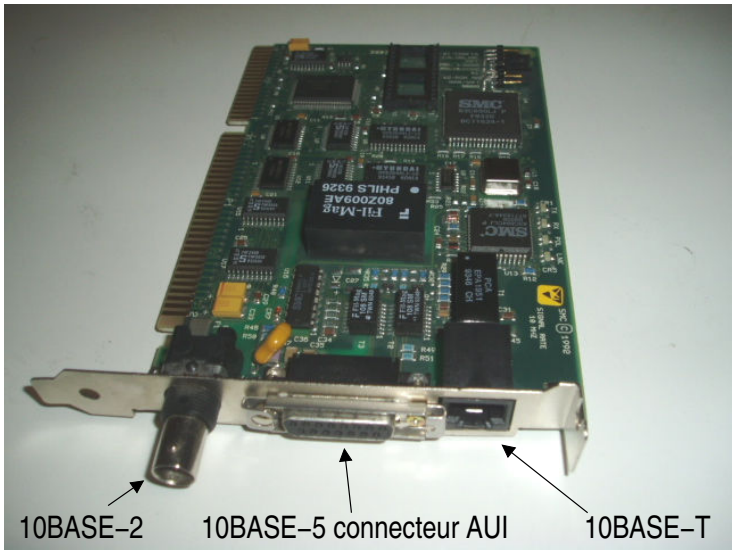
- Disponible sur certains commutateurs
- Trames «pause»
  - Adresse multicast 01-80-c2-00-00-01
  - Type 0x8808
  - Information : sur deux octets ($<$65536), contient le nombre de «pause_times» (périodes de 512 bits)
  - Indiquent la durée pendant laquelle l'extrémité ne désire pas recevoir de nouvelle trame

Transciever Ethernet 10-BASE-5 55/75



Source :
`http://tech.mattmillman.com/projects/10base5/`
`https://petri.kutvonen.net/index_files/ethernet.html`

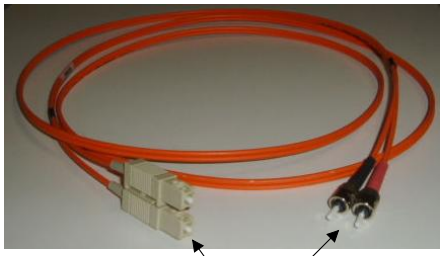
Une carte interface «multimedium» 56/75



Technologie des débuts 90.
Aujourd'hui on ne trouve plus que du 10/100/1000 BASE-T, à quelques Euros...
Et en plus il y a beaucoup moins de composants...

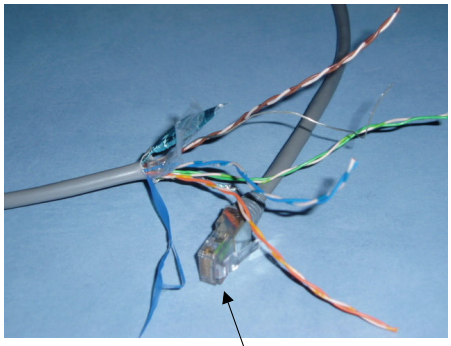
Câbles et connecteurs – Cuivre et fibre 57/75

Fibre optique



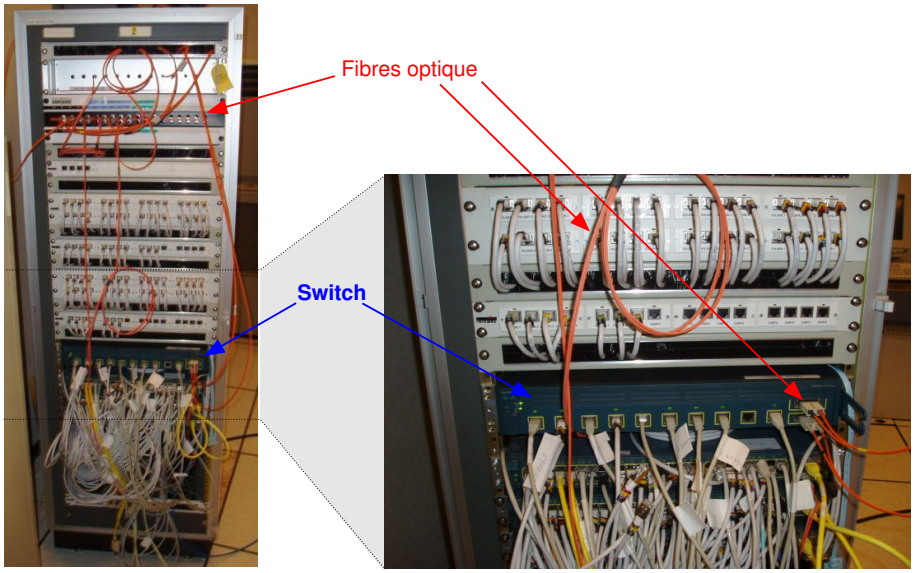
Connecteurs SC et TS

Paires torsadées



Connecteur RJ45

Une baie de brassage de câbles 58/75



L'envers du décor :

Administration des commutateurs 59/75

- Les commutateurs de bas de gamme ne s'administrent pas
- On peut se connecter sur les autres
  - Par port série dédié (et hyper-terminal sous Windows ou minicom sous linux)
  - Par telnet sur une adresse IP configurée dans l'appareil
  - Par page Web sur une adresse IP configurée dans l'appareil
  - Par SNMP
- Les actions d'administration peuvent être
  - Mettre en place ou inhiber la fonction spanning-tree et gérer cette fonction (priorités du switch, QoS, ...)
  - Mettre en place et gérer des VLANs
  - Monitorer des ports
  - Surveiller les adresses MACs enregistrées
  - Surveiller le trafic

La fonction «monitoring» consiste à rediriger le trafic normal d'un port vers un port dédié à la fonction et sur lequel on place une machine munie d'un analyseur de flux. Toutes les trames entrantes et sortantes du port surveillé sont recopiées sur le port de monitoring.

3 Les autres techniques autour des LANs

3.1 Le sans fil IEEE-802.11 (WiFi : Wireless Fidelity)

IEEE-802.11 - WiFi 62/75

- Canal radio à 2,4GHz (5GHz pour 802.11a)
  - 11b : jusqu'à 11Mb/s – 11g : jusqu'à 54 Mb/s
  - Distance relativement courte : une centaine de mètres dans de bonnes conditions de propagation, ce qui est rarement le cas en intérieur
- Méthode d'accès au médium : CSMA/CA
  - Les collisions sont évitées par un mécanisme de délai avant d'émettre
- Contrôle d'accès
  - Sécurité possible avec chiffrement : Wired Equivalent Privacy (Wep)
- Type d'application :
  - raccordement de mobiles à des réseaux de type LAN dans les entreprises et lieux publics

Un tutorial correct :
`http://www.intelligraphics.com/articles/80211.article.html`
Un lien sur la sécurité en 802.11 :
`http://www.iss.net/wireless/WLAN.FAQ.php`
Autre lien intéressant :
`http://wireless.ictp.trieste.it/school.2002/lectures/ermanno/HTML/802.11_Architecture.pdf#search='802%2011%20ssid'`

3.2 Les courants porteurs

Technique Courant Porteur 64/75

- CPL : courant porteur en ligne
  - Le réseau d'alimentation électrique est le medium
  - Interfaçage simple avec Ethernet ou USB : le câble catégorie 5/RJ45 est relié à une prise spéciale enfich une prise de courant

  - Débits annoncés : 5-20 Mb/s
    - Des produits arrivent annonçant 200Mb/s (théorique) !
  - Modulation en ligne OFDM (certains utilisent CDMA)
  - Solutions propriétaires, pas de compatibilité entre produits
    - Pourtant un standard : Home Plug
    - Des travaux à l'ETSI
  - Applications : «émulation Ethernet», raccordement de quartiers résidentiels, d'entreprises, etc...

Un lien intéressant `http://vlan.org/breve123.html`

3.3 Autres

Autres techniques Réseau 66/75

- Les PANs : Personnal Area Network
  - Très courtes distances
  - Typiquement : Bluetooth ZigBee
    - Interconnexion de petits portables (téléphones, PDAs, etc.), domotique (capteurs, actionneurs)
  - Standardisation en cours : IEEE-802.15
- Les bus spécialisés
  - USB
  - IEEE-1394 (FireWire ou iLink selon les constructeurs)
    - Raccordement d'appareils multimédia (caméra vidéo numériques, audio)
    - Canaux synchrones et asynchrones
    - 800Mb/s
    - Distance : 100m annoncée entre nœuds et hubs (4,5m en standard)
    - En standard sur les PCs aujourd'hui

Les canaux synchrones de IEEE-1394 (FireWire/iLink) sont adaptés à la transmission de flux sons et images.

Ces réseaux utilisent des protocoles spécifiques pour les couches supérieures, par exemple

AMDTP (Audio and Music Data Transmission Protocol. IEC61883-6) pour IEEE-1394. IP n'est pas implémenté en standard sur ces réseaux.

Une émulation Ethernet existe pour IEEE-1394 en utilisant les canaux asynchrone.

FireWire : IEEE-1394 chez Apple.

ilink chez Intel

3.4 La couche LLC

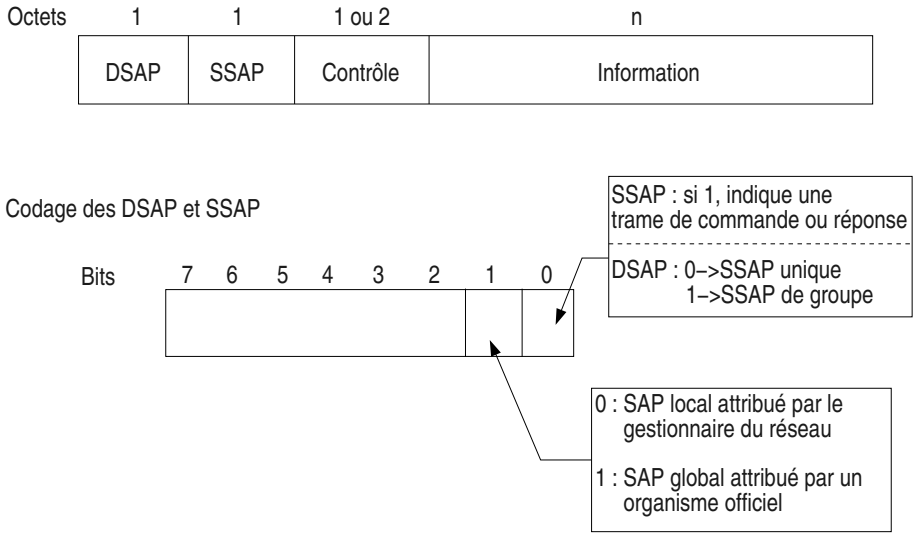
La couche LLC – IEEE-802.2 I 68/75

- Permet de combler les manques de la couche MAC par rapport au niveau 2 OSI standard (si nécessaire)
- Permet d'indiquer le SAP du protocole véhiculé dans les données utiles de la trame MAC

La couche LLC – IEEE-802.2 II 69/75

- Trois types LLC
  - LLC type 1 : mode datagramme, sert uniquement à véhiculer le SAP des données utiles
  - LLC type 2 : mode connecté, en plus de la fonctionnalité du type 1 on assure des contrôles identiques au HDLC LAPB (avec numérotation des trames modulo 128 : champ contrôle sur 2 octets), sert à véhiculer des paquets X25 par exemple
  - LLC type 3 : mode datagramme avec acquittement. Prévu pour les réseaux industriels

Format du PDU LLC 70/75



— DSAP : Destination Service Access Point

— SSAP : Source Service Access Point (i.e. *comment* interpréter le champ contrôle)

Et donc ... il reste combien de SAP possibles, donc de protocoles identifiables ?
64 ! C'est bien peu et c'est un problème...

LLC : quelques SAP 71/75

- 0x06 : IP
- 0x42 : Spanning Tree
- 0x7E : X25
- 0xE0 : Novell IPX
- 0xAA : SNAP (voir plus loin)

La notation 0x... est celle du langage C ou Java pour les nombres en hexadécimal.
Le SAP 0x7E ne vous rappelle rien concernant X25 et surtout sa couche 2 standard ?

LLC : le champ contrôle 72/75

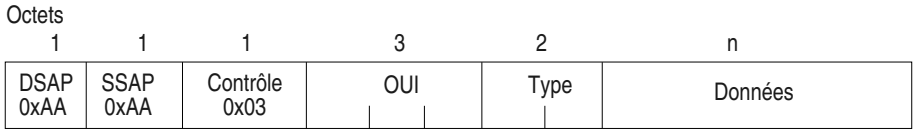
- LLC type 1
  - Le champ contrôle est codé sur un octet et vaut 0x03
- LLC type 2
  - Format identique au champ contrôle (ou commande) de HDLC LAPB
  - Trames numérotées modulo 128
    - Longueur du champ : 2 octets pour les rames I, RR, RNR et REJ
    - La trame d'établissement de la connexion s'appelle SABME (Set Asynchronous Balanced Mode *Extended*)

LLC SNAP 73/75

- SNAP : Sub Network Access Protocol
- Les champs DSAP et SSAP standards sont trop courts
  - Le champ type de la trame Ethernet pur est très bien...
  - Si on le réutilisait... En le plaçant après le champ contrôle ?
    - Oui mais sa longueur n'est que de deux octets !
    - Le tout ferait donc 5 octets (Contrôle sur 1 octet), ce n'est pas optimum pour une architecture 32 bits (l'architecture de la plupart de nos machines encore pour l'instant)
    - Et si on complétait à 8 octets en rajoutant l'OUI du concepteur du protocole ?

SNAP : même traduit un français, ça ne veut rien dire (humour (?!!!))
OUI : Organizational Unit Identifier

LLC SNAP : le format 74/75



- DSAP = SSAP = 0xAA
- Contrôle = 0x03 (trame UI : Unumbered Information)
- OUI : code attribué par l'IEEE à la société créatrice du protocole. Identique aux 3 octets de début des adresses Ethernet. Souvent à 0
- Type : identifie le protocole : identique au champ Type de la trame Ethernet

Mais non !... Ce n'est pas si compliqué !...

Et en plus tout peut être justifié... Il y a une bonne raison pour qu'il en soit ainsi. Simplement il faut chercher cette raison parfois profondément...

Exemples de trames LLC 75/75

```
 0: 0900 07ff ffff 0040 9c00 0294 0022 aaaa .......@....."..
16: 0308 0007 809b 001a 0000 0000 03e8 ffe1 ................
32: 0101 0103 e808 e103 e880 03e8 8201 9400 ................


 0: 0180 c200 0000 0000 1d07 2c63 0026 4242 ..........,c.&BB
16: 0300 0000 8000 0000 0000 0000 0000 0000 ................
32: 0000 0000 0000 00c0 0f00 0050 0000 0000 ...........P....
48: 0000 0000 ....
```

Décodons... Et répondons aux questions :

— pourquoi peut-on dire que ces trames sont de type multicast ?

— que nous enseignent les adresses destination utilisées dans ces trames ?

— pourquoi peut-on dire qu'elles sont au format 802.3 et non Ethernet pur ?

— quels sont les protocoles véhiculés ?

IMT Atlantique
Département Informatique
Technopôle de Brest-Iroise - CS 83818
29238 Brest Cedex 3
URL: **www.imt-atlantique.fr**

Cybersécurité des systèmes maritimes et portuaires
Networking Introduction
Fall 2021

Introduction to Mininet

Edited: September 3, 2021
Version: non versionné

Report filled-in by:

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

1. Objective

- Get familiar with the network emulator Mininet [1]

2. Pre-Lab

Beforme coming to the lab:

- make sure you have read the Mininet Section of this lab.
- done and understood Mininet walkthrough http://mininet.org/walkthrough/
- study the linux commands `ip` (`ip address`, `ip route`), `ifconfig`, `ping`, `route`. You can refer to their man pages by typing on a console `man <command>`.

3. Mininet

During some labs of the course we are going to work with a network emulator called Mininet [1].

Mininet provides a high-level python API that allows to easily create emulated networks (host, routers, switches, and their connections through links). It also provides some built-in topologies. During this lab we are going to emulate a given topology by executing a python script.

Underneath the high-level APIs we are going to use to create emulated networks, mininet uses Linux process-based virtualisation and network namespaces to create network-isolated nodes. Each of the nodes of the emulated network will look to us as a Linux system, where we can perform routing, switching (through virtual switches) and so on. All this will be transparent to you during the courses labs, but if you are curious, and we encourage you to be so, find out more about mininet through [1] or surfing the Internet.

Mininet is already installed in the VM of the course. In order to get a mininet console we can run the command `mn` or execute a python script that does this for us. In any case, in order to use mininet we need superuser privileges (run commands with `sudo`). From a mininet console, we can access and control any of the nodes of the emulated network.

4. Hands on

We are going to work here with the network shown in Fig. 1, used in the ARP Lab.

Question 1.

Explain with words the different components of the network in Fig 1.

4.1. Getting Familiar with Mininet

For the moment, you will get familiar on how to interact with a traditional network emulated in mininet [1]. You will work on a new approach to control a network in the SDN Lab.

1) Run the course VM

2) **Using a terminal** inside the VM, check the presence of the `net_labs` directory, if it is present, update your copy of the repository:

```
cd net_labs
git pull origin master
```

If it is not present, clone the repository and move to the `net_labs` directory

```
git clone https://redmine.telecom-bretagne.eu/git/net_labs
cd net_labs
```

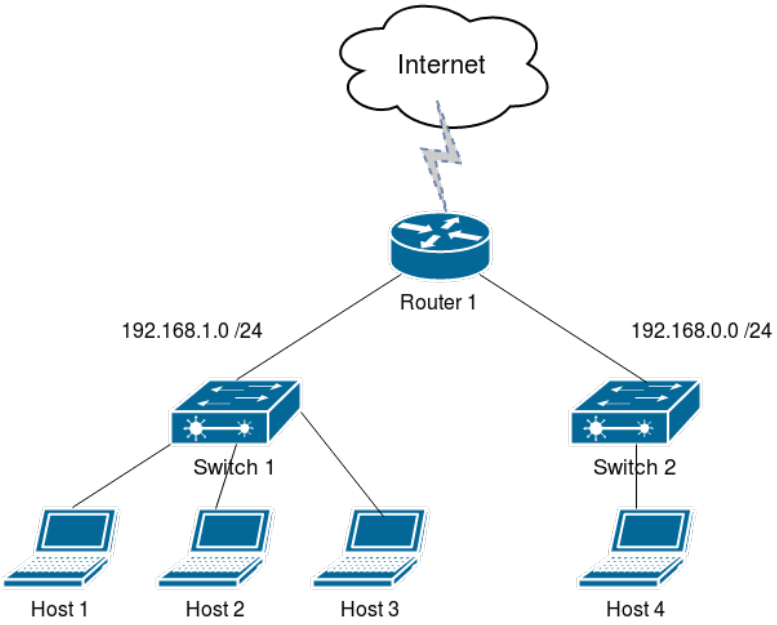


Figure 1: Topology 1

3) Go to the arp directory, for that type: `cd arp`

4) Verify the existence of the files called `topo_arp.py`, `createOVSswitchStandAlone.sh` and `killOVSswitch.sh` (you can use the command `ls`)

5) Verify that you have execution permissions for the `.sh` files, these scripts are going to be called by the python script. For that type `ls -lh` and verify the presence of an `x` at the first column of the output.

6) Execute the python script to emulate the network `sudo python topo_arp.py`. This will build up the topology and open a mininet console (`mininet>`)

7) Try the following mininet commands, and figure out what each of those commands do.

```
mininet> help
mininet> nodes
mininet> net
mininet> dump
```

Mininet also allows to run any unix command in any of the components of the network, by first indicating the name of the component as known to mininet.

8) Try the following unix commands typed from mininet console, and figure out what each of those commands do.

```
mininet> h1 ip address
mininet> h1 ip route
mininet> h2 ip a
mininet> h2 route -n
mininet> h2 ls
mininet> r1 ifconfig
```

```
mininet> h1 ping h2
mininet> h4 ping r1
```

Note that you can stop a ping with keys ctr+c.

Mininet also allows to open a terminal at any of the network equipments.

9) Try, for instance, the following

```
mininet> xterm h1
mininet> xterm r1
```

Note that this command opens a root terminal in the equipment.

10) Now, to exit from the mininet console type `exit`. You will also need to close each opened xterm.

At any moment, you can clean up the emulation and start allover again, if you are experimenting any problem with mininet, typing on a **terminal of the VM** `sudo mn -c`.

11) In order to start the next lab of the course from a clean state, run

```
sudo mn -c
```

References

[1] Mininet, An Instant Virtual Network on your Laptop (or other PC) [online] https://mininet.org

IMT Atlantique
Département Informatique
Technopôle de Brest-Iroise - CS 83818
29238 Brest Cedex 3
URL: **www.imt-atlantique.fr**

Cybersécurité des systèmes maritimes et portuaires
Networking Introduction
Fall 2020

Lab Ethernet and ARP

Edited: September 3, 2021
Version: 1.2

Report filled-in by:

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

1. Objective

- Understand the purpose and functioning of the ARP protocol
- Explore differences of single-segment and multiple-segment networks

2. Pre-Lab

Beforme coming to the lab:

- make sure you have read the introduction of this lab.
- study the linux commands `arp`, `ip-neighbour`, `ifconfig`, `ping`, `route` you can refer to their man pages by typing on a console `man <command>`

3. Introduction

3.1. The Address Resolution Protocol (ARP)

When a host A tries to send data to a host B, A's network layer builds a packet with the following IP addresses: IP source= A's IP address, IP destination = B's IP address, and with the payload. That packet is afterwards passed to A's data link layer. This layer does not understand IP addresses, but it uses an address in accordance to the technology being used. In the case of Ethernet (or IEEE 802) networks, each network interface has a 48-bit address which is usually assigned by the manufacturer in a unique basis. Ethernet interfaces communicate between them using these 48-bit addresses. It is thus necessary to have some sort of mapping between IP and Ethernet addresses, and making it possible with the lowest possible overhead and administration effort. In the internet, this is achieved thanks to the Address Resolution Protocol (ARP).

ARP is defined in RFC 826 [1], it allows to solve the mapping between network layer addresses (IP addresses in TCP/IP networks) and data link layer addresses. In broadcast networks (such like Ethernet networks) the MAC (medium access control) sublayer is in charge of managing the access to the medium (either by managing turns to stations or by managing the collisions). Usually the addresses used in broadcast mediums are called MAC layer addresses or simply MAC addresses. So the 48-bit addresses of Ethernet networks are the MAC addresses of such network, and we shall use both names interchangeably.

ARP does not relay in any administration or human intervention. When A tries to communicate with B, the protocol finds out B's MAC address having as an input B's IP address.

Please note that this protocol works for IPv4. During this lab, for teaching purposes, we are going to focus on IPv4. However, in IPv6 the same problematic must be solved as well. In the case of IPv6, is the Neighbor Discovery Protocol [2], which among other functions, performs the functions equivalent to those of ARP. The `arp` command we shall use in this lab, is only available for IPv4. For IPv4 and IPv6 you can use `ip neighbour`, which we shall as well use in this lab.

3.1.1. ARP packet format

ARP messages go directly as the payload of the MAC sublayer frame (Ethernet frame for the case of Ethernet). They contain, among other fields the origin and destination IP and MAC addresses of the frame. You can see the complete format of ARP packets in Fig. 1

3.1.2. Protocol functioning

We are going to see the ARP functioning throughout the lab. In order to prepare your lab you are encourage to read about protocol functioning, for instance, in [3, 1] or browsing the Internet.

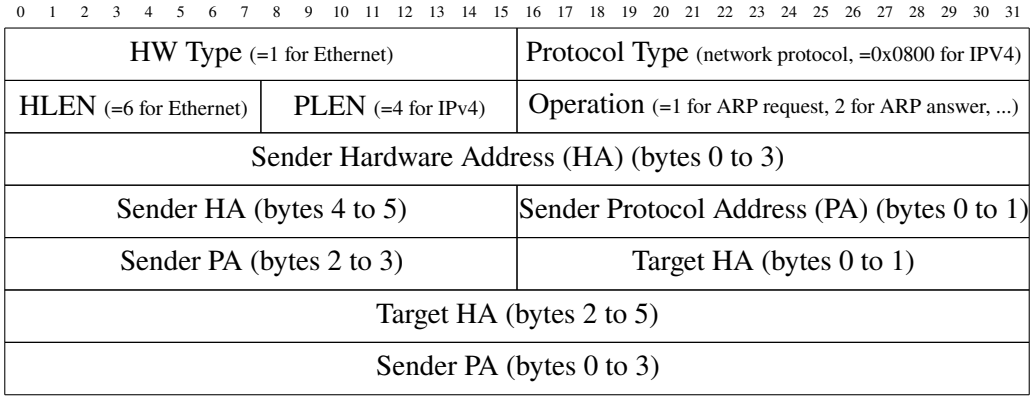


Figure 1: ARP packet format.

3.1.3. ARP cache

Machines (host, routers) stock the mapping between IP address and MAC address during a certain time, so that the protocol does not have to be executed every time a packet needs to be sent. This improves efficiency of the communication. The mapping is stored in the so-called ARP table or ARP cache. The entries in the memory are frequently erased (typically every 20 minutes) and can also be manually erased.

In Unix systems one command that allows to see the content of the ARP table is `arp -a`. See the man page for the complete documentation, you will find in particular that the command allows to delete entries of the table as well (-d <ip address>) and avoid the resolution of names associated to IP addresses (-n option).

Another available command is `ip-neighbour`, for instance to see all the content of the arp cache you can type on a terminal `ip neigh show` or to delete or entries in the cash you can type `ip neigh flush all`. You can see the complete documentation by running `man ip-neighbour`.

3.2. Recall: Network addresses and Netmasks

A network interface, to work properly, needs to get configured the address itself as well as the network mask. For instance, 192.168.122.2/24 where at the left of the / we have the address and at the right the so-called netmask or subnet mask. The netmask allows to determine which part of the address corresponds to the <prefix>, i.e. the common part of every address belonging to that same network.

Historically, there were five different classes of IPv4 addresses, A to E, as shown in Table **??**. Each of those classes implied a <prefix> of a different length. The prefix part, as said before, indicates the common part of every address belonging to the same network, while everything that comes after the prefix indicates the number of the machine in that network. Please note that according to the class of the address we can have a different number of machines in the network. Notice as well that this classful addressing scheme only allows for prefixes to be of one, two or three bytes, according to the class.

Later on, more flexibility was needed, i.e. allowing a customisable maximum number of machines at each network, or also sub-networking (talking some bits from the machine part of the address to identify a subnetwork). The so-called Classless InterDomain Routing (CIDR) was introduced. Netmasks are a key concept in CIDR. A netmask is a special binary number that along an IP address allows to easily determine the <prefix> or what is the same, all the addresses belonging to the same network, as we will see shortly.

Netmasks have the same bit-length as the associated IP address. The left-most bits of the netmask are set to one, in such a way that a bit-wise AND between the binary representation of the IP address and the netmask, results in the network address. In other words, the number of bits set to one in the netmask indicates the number of bits of the IP address that corresponds to the <prefix>. A pair IP address/netmask determines a block of IP addresses belonging to the same network. The first address of that block is

Class	Leading bits	size of <prefix> in bits	Start address	End address	Default subnet mask in dot-decimal notation	netmask in CIDR notation
Class A	0	8	0.0.0.0	127.255.255.255	255.0.0.0	/8
Class B	10	16	128.0.0.0	191.255.255.255	255.255.0.0	/16
Class C	110	24	192.0.0.0	223.255.255.255	255.255.255.0	/24
Class D (multicast)	1110	not defined	224.0.0.0	239.255.255.255	not defined	not defined
Class E (reserved)	1111	not defined	240.0.0.0	255.255.255.255	not defined	not defined

Table 1: IPv4 classes

IP address	Netmask CIDR notation	Netmask dot-decimal notation	Network address	Broadcast address
192.168.122.109	/24	255.255.255.0	192.168.122.0	192.168.122.255
192.168.122.109	/30	255.255.255.252	192.168.122.108	192.168.122.111
10.4.4.4	/16	255.255.0.0	10.4.0.0	10.4.255.255

Table 2: Examples of netmasks

reserved to designate the network, and the last one for broadcast (i.e. for all machines in the network). The remaining addresses are available for machines. Some examples are shown in Table 2.

The concept of netmask exists and is used both in IPv4 and IPv6.

4. Hands on

Throughout this lab we are going to work with the network shown in Fig. 2.

Question 4.1.

Explain with words the different components of the network in Fig 2 (imagine you are describing the network topology to someone that is not looking at the picture). Make sure you understand your own words!

4.1. The Address Resolution Protocol

4.1.1. Set-up the network topology

1) Run the course VM

2) Check the presence of the `net_labs` directory, if it is present, update your copy of the repository:

```
cd net_labs
git pull origin master
```

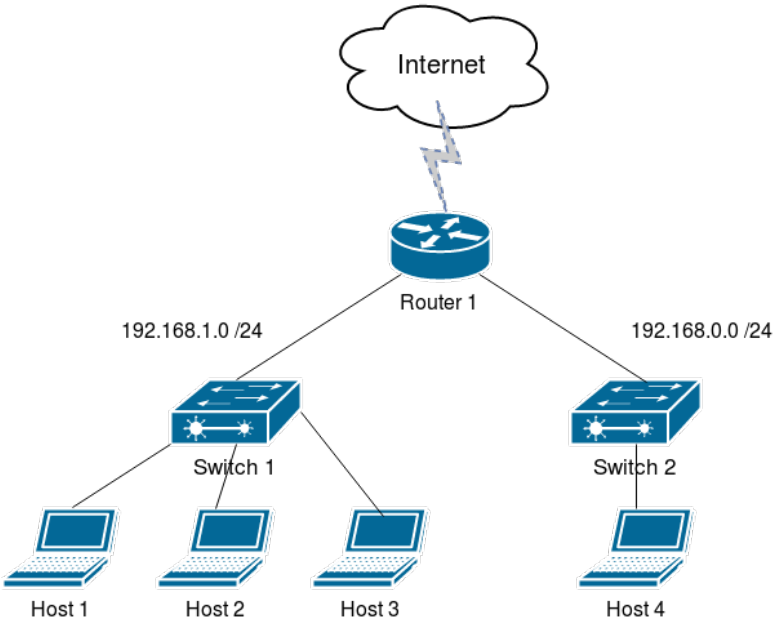


Figure 2: Topology 1

If it is not present, clone the repository and move to the `net_labs` directory:

```
git clone https://redmine.telecom-bretagne.eu/git/net_labs
cd net_labs
```

3) Go to the arp directory, for that run: `cd arp`

4) Verify the existence of the files called `topo_arp.py`, `createOVSswitchStandAlone.sh` and `killOVSswitch.sh`.

5) Verify that you have execution permissions for the `.sh` files, these scripts are going to be called by the python script. For that run `ls -lh` and verify the presence of an `x` at the first column of the output.

6) Execute the python script to emulate the network `sudo python topo_arp.py`. This will build up the topology and open a mininet console (`mininet>`)

7) Find out the MAC addresses of all the relevant interfaces. You can use any of the two options seen before in the Introduction to Mininet lab, either opening a terminal in the component either typing the command directly from mininet's console. Please note that components names are only available within mininet. Complete Table 3 with the correct information.

4.1.2. Observing the content of the ARP cache

1) In host 1's console, type the following command `arp -n` or equivalently comand `ip neigh show`

Mininet's name	Interface	IP address	MAC address
h1	h1-eth0	192.168.1.2/24	
h2	h2-eth0		
h3	h3-eth0		
h4	h4-eth0		
r1	r1-eth0		
r1	r1-eth1		

Table 3: Interface names and layer 2 and 3 addresses for the working topology.

Question 4.2.
What is the content of h1's ARP cache?

2) Perform a ping from h1 to h2, make sure to use the `-c` option to send only one packet.

3) Check again the content of h1's ARP cache with your preferred command

Question 4.3.
Do you observe any difference with respect to its previous content? Explain.

4) We will now try a ping to h4. From h1 ping h4

Question 4.4.
Do you expect any changes in the ARP cache with respect to its previous content?

5) Verify your answer to the previous question by looking at h1's ARP cache
(`arp -n` or `ip neigh show`)

Question 4.5.
Does the result match with what you expected? Explain each entry in the table.

6) Verify the content of r1's ARP cache

4.1.3. Observing ARP messages

We are now going to see the protocol in detail, understanding how the information that appears in the ARP cache is obtained. For that we are going to observe the exchanged ARP messages under different scenarios. First of all, let's start all over again from scratch, in order to make sure of the departure state. This means:

1) Put down mininet (`mininet> exit`)

2) Clean up mininet (`sudo mn -c`)

3) Run the topology again (`sudo python topo_arp.py`)

4) Start three instances of wireshark, one in h1, another one h2 and finally h3. For that:

1. From the mininet console type `hi wireshark &` for i=1 to 3.
2. Start a capture at the `hi-eth0` for each wireshark. These interfaces correspond to the interfaces connected to the switch. (see Fig. 2).

5) Perform a one-time ping from h1 to h2

6) Observe the capture at h1's eth0. Use a visualization filter to see only ARP packets.

Question 4.6.
What is the protocol stack being used?

Question 4.7.
Which ARP messages do you see? Focus on the first two messages. For each of those ARP messages specify the content of the different fields. Deduce from that the purpose of the message. You can get help from the information provided in the Introduction (see section 3.1.1). In particular, pay attention to the origin and destination. Note: you can see some unexpected behavior on the messages ordering due to how wireshark handles the capture. If ever you notice this, discuss it with the instructor.

Question 4.8.
Look at the Ethernet frames carrying each of the previous two ARP messages and indicate for each of them the value of the type field, the source, and the destination fields. You have probably noticed that some of this messages are addressed to a broadcast address while others to a unicast one. We should see more details on this on Question 20.

Question 4.9.
Sender and receiver Hardware addresses (MAC addresses) are present in the Ethernet frame as well as in the ARP message carried by the Ethernet frame. Why do you think this information is duplicated?

7) Look now at the packets captured at h2-eth0.

Question 4.10.
Which ARP messages do you see?

Question 4.11.
Explore the ICMP messages, which is the protocol stack in use?

Question 4.12.
Which is the value of the field type in the Ethernet frame carrying an ICMP message?

8) Look at the packets captured at h3-eth0

Question 4.13.
Which ARP messages do you see? Explain.

Question 4.14.
What can you conclude about the functioning of ARP according to the previous captures? Explain the functioning of the protocol.

We are now going to analyse a situation where h1 communicates with a host in another LAN, in this case, h4.

Question 4.15.
What is the content of h1's arp cache?

1) From mininet console, open a terminal in h4 (`xterm h4`) once in h4, run a wireshark instance.

2) You can close the wireshark instances opened at h2 and h3

3) Start a new capture at h1-eth0 and a new capture at h4-eth0

4) Perform now a ping from h1 to h4.

5) Observe the capture **at h1**

Question 4.16.
Do you see any ARP message concerning this ping? Which ones? Detail.

6) Observe the first ICMP message (Echo request)

Question 4.17.
Observe the IP header, which is the destination IP? Observe the Ethernet frame, which is the destination MAC address? Explain.

7) Observe the capture **at h4**

Question 4.18.
Which ARP messages do you see? Pay attention to the involved source and destination addresses.

8) Observe the first ICMP message (Echo request)

Question 4.19.
Observe the IP header, which is the source IP? Observe the Ethernet frame, which is the source MAC address? Explain.

We are now going to explore what happens when a machine updates the content of its cache.

1. Use command `ip neigh show` to check the content of h1's ARP cache.
2. Find an entry whose state is STALE. If no entry is at this state, wait some seconds, and repeat previous and this step
3. Initiate a new wireshark capture in h1's eth0, use a visualization filter for ARP messages (type arp in the Filter field)
4. Perform a one time ping from h1 to the destination you identified as in state STALE in step 2
5. Analyse your capture

Question 4.20.
Which ARP messages do you see? In particular, observe the source and destination addresses, do you see any difference with what you have observed in question 7 in this subsection? Explain.

4.2. Netmasks

The ARP section is now closed. Let's open a new section. We are now going to turn our attention to the understanding of netmasks. You will be performing some troubleshooting over two different cases of interface misconfiguration in h1.

First case study

Suppose that interface h1-eth0 is configured with the same IP address but its netmask is set to /30 (while previously it was /24). We are going to set up this scenario and explore what happens with the connectivity between the different hosts.

1) Set h1's h1-eth0 to 192.168.1.2/30. For that you will use the ifconfig command, in h1's console type: `ifconfig h1-eth0 192.168.1.2/30`.

2) Start a wireshark capture at h1-eth0, h3-eth0 and r1-eth0.

3) Run a ping from h1 to r1.

Question 4.21.
The ping should be successful. Explain why. It could be useful for building your answer to look at the content of Table 2 and complete a line with the current case study. In particular think of the number of bits available for coding your host-part of the address. A handy command line tool is `ipcalc`.

4) Run a ping from h1 to h3.

Question 4.22.
The ping should not be successful. Write down the output of the ping and explain what happened. To understand what happend, you can use another linux command useful in this case, the `route` command, which shows the routing table of the host. Use it with the `-n` option to avoid name resolution.

5) Run a ping from h3 to h1

Question 4.23.
The ping should not be successful but you should obtain an output different to the previous one. Explain. Use the wireshark captures to support your explanation.

Second case study

Now we are going to explore another case. For that you are going to set the netmask of h1-eth0 to /23

1) Configure h1-eth0 using the command `ifconfig h1-eth0 192.168.1.2/23`.

2) Complete Table 2 with the current case study. Once again you might want to use ipcalc command line tool.

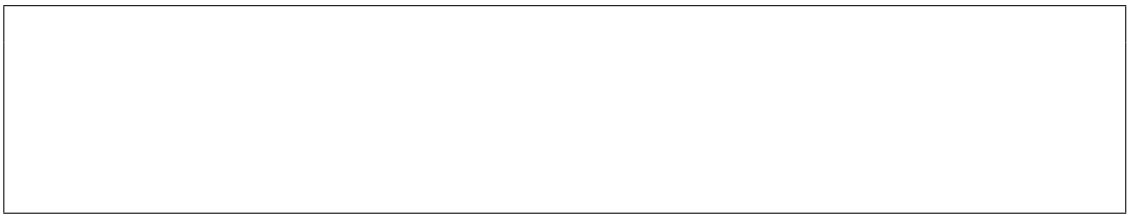
3) Run a ping with one packet option from h1 to h3

Question 4.24.
In this case the ping should be successful. Explain why.

4) Start a new wireshark capture in h1-eth0

5) Now, run a ping from h1 to h4

Question 4.25.
In this case, the ping should not be successful. Explain why. Support your answer explaining the contents of the capture.



5. Conclusion

Question 5.1.
What is the purpose of the ARP protocol?

Question 5.2.
Why do we have different addresses at layer 2 and layer 3?

Question 5.3.
Explain the purpose of addresses netmasks and how a network device uses them.

References

[1] RFC 826 - An Ethernet Address Resolution Protocol, or, Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware, IETF, [online] https://tools.ietf.org/html/rfc826.

[2] RFC 4861 - Neighbor Discovery for IP version 6 (IPv6), IETF, [online] https://tools.ietf.org/html/rfc4861.

[3] Computer Networks, Fifth Edition Andrew S. Tanenbaum, David J. Wetherall, Prentice Hall

[4] Mininet, An Instant Virtual Network on your Laptop (or other PC) [online] https://mininet.org