INFO-F303 - Réseaux, information et communications

1. Examen

Les réponses proviennent (ou par l'intermédiaire de résumé) de Denis Steckelmacher, Pierre Dagnely, Christophe Dumeunter, Julien Vanbergen, David Fishel et Rodrigue Van Brande. Merci à eux.

1.1. Théorie

1.1.1. Expliquez la différence entre une paire de cuivre torsadée de catégorie 3 et une paire de catégorie 5. Laquelle permet un débit plus élevé et pourquoi ?

On utilise des paires de cuivre torsadées car deux fils de cuivres parallèles créent chacun un champ magnétique proportionnelle à leur distance, ce qui induit un grand courant et donc beaucoup de bruit. Or lorsqu'ils sont torsadés, les torsades créent chacune leur champs magnétique ainsi qu'un courant induit. Ces flux induits s'inversent dans chaque boucle adjacente. Les effets sont donc atténués. Plus les torsades sont petites, plus l'effet est efficace. Les fils de catégorie 5 sont plus torsadés que ceux de catégorie 3 et donc plus efface (100 Mbps contre 10 Mbps).

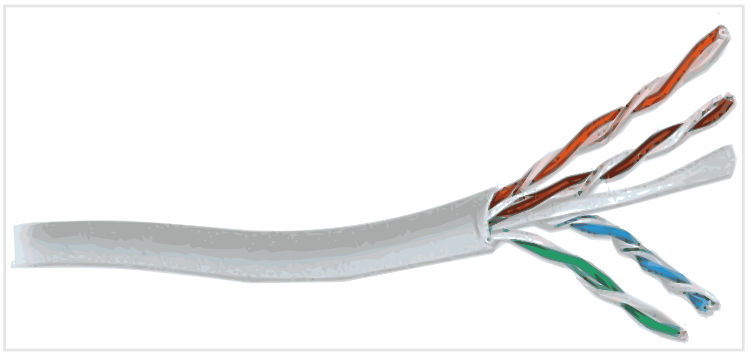


Fig. 1 - Cable torsadé

1.1.2.

(a) Expliquez la différence entre une fibre optique monomode et une fibre multimode.
(b) Laquelle permet un débit plus élevé et pourquoi ?

(a) La fibre optique est une émission d'un nuage de photon dans un câble en verre. Composé d'un câble en fibre optique inclus dans un autre, les deux ayant des indices de réfractions différents. Ceci permettant de piéger les signaux par réfraction. les photons ne peuvent plus quitter le câble et se retrouvent bloqués dans le câble du centre. On utilise deux types de fibre: le monomode et multimode.

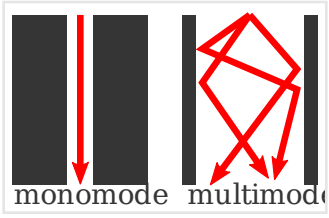


Fig. 2 - Deux modes de fibres optique

- **Multimode**: Le problème de la fibre optique est que les photons émis dans un même nuage prennent des trajectoires différentes (en fonctions des angles de réflexions au sein du câble), les photons du signal n'arrivent donc pas en même temps au bout du câble (ni dans la même partie). Or plus la distance est longue plus cet effet grandit et plus l'écart entre les photons de tête et de queue s'agrandit, on a donc un étalement puis un chevauchement des flux. Pour éviter les chevauchement on est donc obligé d'espacer les émissions en fonction des distances et capacités de réceptions.
- **Monomode**: Ici on utilise un verre plus étroit (2.4 µ) avec un seul mode de propagation, les photons vont donc en ligne droite, sans étalement. C'est efficace pour les grandes distances, mais plus chère.

(b) La fible monomode possède donc un débit bien plus élevé car il n'y a pas de dispersion de délai ; on ne doit pas attendre entre les symboles. Elle est cependant bien plus chère. La fibre multimode est moins optimale à cause des chevauchement possible et n'est du coup utilisé que pour de courte distances.

1.1.3.

(a) Expliquez le principe du multiplexage en longueur d'onde (WDM). Quel est son intérêt ?
(b) Comparez WDM aux techniques classiques de multiplexage TDM et FDM.

(a) La technique **WDM** est utilisée pour la fibre optique. On peut remarquer que la longueur d'onde étant l'inverse de la fréquence, cette technique utilise donc le même principe que la **FDM**. En fait, on envoie des photons avec des longueurs d'onde différentes selon l'utilisateur dans la fibre. Ces photons passent dans un prisme pour combiner toutes les lumières à l'entrée de la fibre puis sont à nouveau séparés à la sortie de la fibre à l'aide d'un autre prisme.

Le protocole gère 3 classes de trafic:

- Un trafic à débit **constant** en mode connecté, tel celui d'une *vidéo compressée*.
- Un trafic à débit **variable** en mode connecté, tel celui d'un *transfert de fichier*.
- Un trafic **constitué de datagrammes** en mode non connecté, tels *des paquets **UDP***.

(b) Pour les deux protocoles orientés connexion, l'idée de base est qu'une station A souhaitant communiquer avec une station B doit au préalable insérer une trame de demande de connexion dans un slot libre sur le canal de signalisation de B. Si B accepte, la communication peut avoir lieu par l'intermédiaire du canal de données de A.

- **FDM** (**Frequency Division Multiplexing**):
  Utilisé en radio et en télévision. On attribue un sous-canal de fréquences à chaque personne qui veut recevoir des informations. Par exemple, le premier usager à la bande de fréquence de 0 à 1KHz, le 2 à la bande de 1 à 2KHz,... Tout le monde peut donc communiquer de manière continue à l'aide de son sous-canal.
- **TDM** (**Time Division Multiplexing**):
  Tout le monde peut utiliser le même canal de communication mais seulement pour un temps donné. Par exemple, toutes les secondes, on change d'utilisateur et chacun à son tour peut utiliser le canal comme il le veut. On doit donc savoir qui peut utiliser le canal quand et il faut avoir une bonne synchronisation.
- **WDM** (**Wavelength Division Multiplexing**):
  Principe similaire à **FDM**, chaque utilisateur possède sa propre longueur d'ondes.

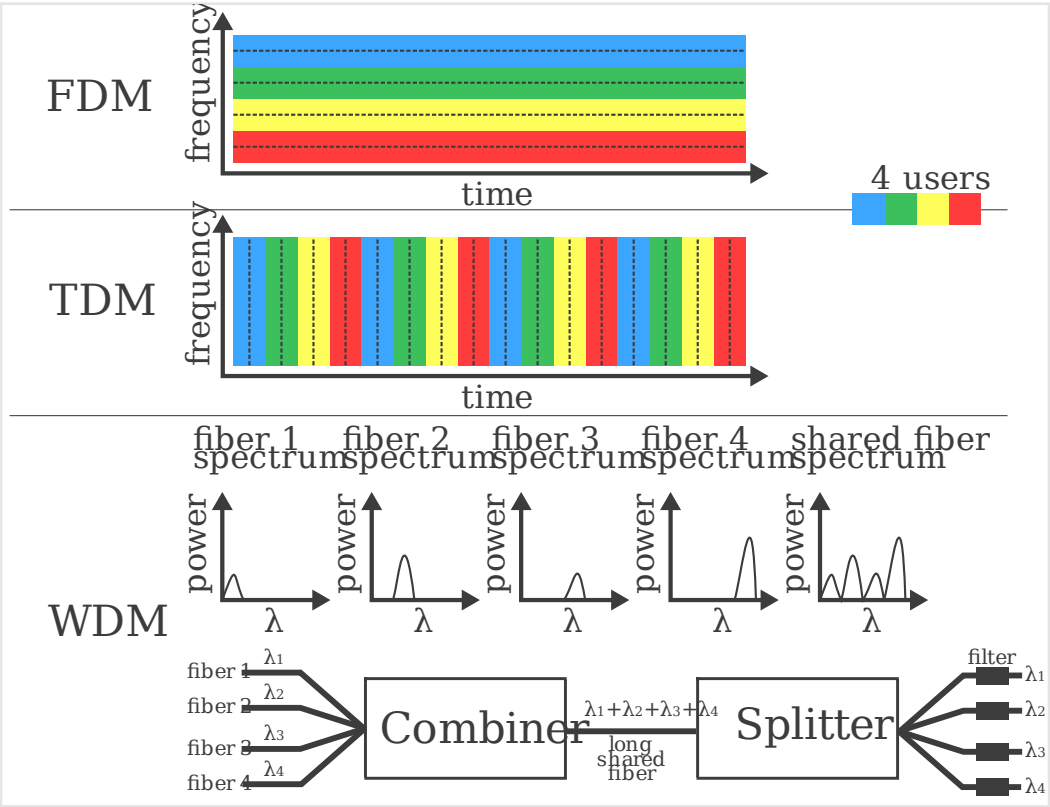


Fig. 3 - FDM TDM WDM

1.1.4.

(a) Expliquez le principe des codes polynomiaux (CRC) pour la détection d'erreurs.
(b) Pour un polynôme donné, citez un exemple d'erreur que le code CRC associé ne pourra pas détecter.
(c) Quelle est l'importance du degré du polynôme utilisé ? Pourquoi ?

(a) Un **contrôle de redondance cyclique** ou **CRC** (Cyclic Redundancy Check) est un outil logiciel permettant de détecter les erreurs de transmission ou de transfert par **ajout, combinaison et comparaison** de **données redondantes**, obtenues grâce à une procédure de hachage. Les CRC sont évalués (échantillonnés) avant et après la transmission ou le transfert, puis comparés pour s'assurer que les données sont strictement identiques.
Le codage binaire est très pratique pour une utilisation dans des appareils électroniques tels qu'un ordinateur, dans lesquels l'information peut être codée grâce à la présence ou non d'un signal électrique. Cependant le signal électrique peut subir des perturbations (distortion, présence de bruit), notamment lors du transport des données sur un long trajets. Ainsi, le contrôle de la validité des données est nécessaire pour certaines applications (professionelles, bancaires, industrielles, confidentielles, relatives à la sécurité, ...).

C'est pourquoi il existe des mécanismes permettant de garantir un certain niveau d'intégrité des données, c'est-à-dire de fournir au destinataire une assurance que les données reçues sont bien similaires aux données émises.
La protection contre les erreurs peut se faire en mettant en place des mécanismes logiques de détection et de correction des erreurs. La plupart des systèmes de contrôle d'erreur au niveau logique sont basés sur un ajout d'information (on parle de «redondance») permettant de vérifier la validité des données.

Le contrôle de redondance cyclique (noté CRC, ou en anglais Cyclic Redundancy Check) est un moyen de contrôle d'intégrité des données **puissant et facile** à mettre en oeuvre. Il représente la **principale méthode** de détection d'erreurs utilisée dans les télécommunications. Le contrôle de redondance cyclique consiste à protéger des blocs de données, appelés trames ("frames" en anglais). A chaque trame est associé un bloc de données, appelé code de contrôle. Le code CRC contient des éléments redondants vis-à-vis de la trame, permettant de détecter les erreurs, mais aussi de les réparer.

Le principe du CRC consiste à traiter les séquences binaires comme des **polynômes binaires**, c'est-à-dire des polynômes dont les coefficients correspondent à la séquence binaire. Ainsi la séquence binaire `0110101001` peut être représentée sous la forme polynomiale suivante:

$$0 * X^9 + 1 * X^8 + 1 * X^7 + 0 * X^6 + 1 * X^5 + 0 * X^4 + 1 * X^3 + 0 * X^2 + 0 * X^1 + 1 * X^0$$

De cette façon, le bit de poids faible de la séquence (le bit le plus à droite) représente le degré 0 du polynôme (`X^0 = 1`), le 4ème bit en partant de la droite représente le degré 3 du polynôme (`X^3`)... Une séquence de n bits constitue donc un polynôme de degré maximal `n-1` . Toutes les expressions polynomiales sont manipulées par la suite avec une arithmétique modulo 2. Dans ce mécanisme de détection d'erreur, un polynôme prédéfini (appelé polynôme générateur et noté `G(X)`) est connu de l'émetteur et du récepteur. La détection d'erreur consiste pour l'émetteur à effectuer un algorithme sur les bits de la trame afin de générer un CRC, et de transmettre ces deux éléments au récepteur. Il suffit alors au récepteur d'effectuer le même calcul afin de vérifier que le CRC est valide.

(b) Le cas où CRC valide un faux positif est extrèmement faible et pourrait survenir uniquement dans le cas d'une double erreur. À la fois dans le message et à la fois dans la valeur du CRC.

(c) Au plus le degré u polynome est grand, au moins d'erreur il y a. Mais au plus long est le calcul et il y a aussi plus d'information à transmettre. Il est donc très important de choisir un bon degré, ni trop grand ni trop petit.

1.1.5.

(a) Expliquez la différence entre un baud et un bps (bit par seconde).
(b) Qu'est-ce qui limite le nombre de bauds sur un canal de communication ?
(c) Qu'est-ce qui limite le nombre de bps sur un canal de communication ?
(d) Comment module-t-on le signal dans les modems « dial-up » les plus courants ? Expliquez sommairement.

(a)
- Un **bps**: bit per seconde
- Un **Baud**: est un symbole ("n'importe quoi") par seconde, en gros le baud peut s'apliquer à autre chose que des bits.

(b) Nyquist impose une borne maximale au débit de symboles. Cette fréquence maximale est de `2H` , avec `H` la bande de

fréquence disponible en Hertz du canal de transmission. Quand on transmet une émission radio, on a un canal, et on doit tenir dedans. H vient du cable et 2 vient du bruit. Sur un réseau de téléphone, la bande passante est d'environ 4000 Hz la fréquence maximale de la voix humaine.

(c) Le bruit, selon la loi de **Shannon** qui dit que le débit en bits est égal à `H*log_2(1+S/N)` avec `S/N` étant le rapport du signal sur le bruit. S'il n'y avait pas de bruit, on aurait eu un log de l'infini. L'important est bien le rapport du signal par le bruit: si on a beaucoup de bruit mais qu'on sait augmenter le signal, c'est bon. Généralement le rapport signal/bruit sur un réseau téléphonique est de 1000.

(d) C'est une forme d'accès à internet qui utilise le réseau téléphonique pour établir une connexion à un ISP. La vitesse maximale est de 56 Kbps et on ne peut pas téléphoner en même temps. Le modem contient un modulateur/Démodulateur. Le but d'un modem est de transformer des informations binaires (numériques) en un signal analogique (et vice-versa). Le plus souvent on modifie à la fois l'amplitude du signal et sa phase.
On ne passe donc pas 1 bit à la fois, mais au moins 2 (1 grâce à l'amplitude et 1 à la phase).

1.1.6. Pourquoi utilise-t-on un modem pour transmettre de l'information numérique sur une ligne téléphonique ? Comment module-t-on le signal dans les modems « dial-up » les plus courants ?

Voir ci-dessus
C'est une forme d'accès à internet qui utilise le réseau téléphonique pour établir une connexion à un ISP. La vitesse maximale est de 56 Kbps et on ne peut pas téléphoner en même temps. Modulateur/DéModulateur. Le But d'un modem est de transformer des informations binaires (numériques) en un signal analogique (et vice-versa). Le plus souvent on modifie à la fois l'amplitude du signal et sa phase. On ne passe donc pas 1 bit à la fois, mais au moins 2 (1 grâce à l'amplitude et 1 à la phase).

1.1.7. Expliquez l'utilité du **ACK** et **NAK**.

Le service de transfert fiable des données doit détecter les erreurs via un checksum et utiliser des accusés de réceptions: le récepteur doit dire explicitement à l'envoyeur s'il a reçu avec ou sans erreur le paquet. Un accusé de réception peut être un **ACK** (packet reçu) ou un **NAK** (packet reçu mais corrompu). Les accusés de réception pouvant aussi être corrompu, on leur ajoute aussi un checksum. On indique dans le **ACK** ou le **NAK** de quel paquet il est l'accusé de réception sinon on peut déstabiliser le système.

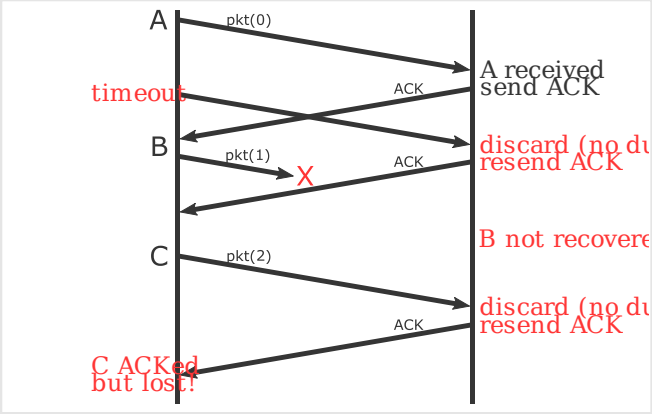


Fig. 4 - Erreur possible lors d'un **ACK** non nominatif

1.1.8.

(a) Citez et expliquez sommairement les 4 mécanismes de base permettant d'assurer la fiabilité d'un transfert d'information au travers un réseau non fiable. Justifiez leur nécessité.
(b) Décrivez dans les grandes lignes un protocole élémentaire qui les met tous en oeuvre.

(a)
- ACK / NACK
- Checksum (Error detection)
- Sequence number
- Timeout

(b)
- **Stop & wait**: utilise le ACK/NACK, le timer
- **Go-Back-N**: utilise le timer, le ACK, le sequence number

- **Selective Repeat**: utilise le timer, le ACK/NACK, le sequence number

1.1.9.

(a) Expliquez le principe du protocole **stop & wait**.
(b) Est-il toujours performant ?

(a) On utilise une boucle simple:

> send → wait positive response → send → ...

Le **NAK** n'est pas utilisé car recevoir un packet corrompu est similaire à ne rien recevoir; on ne renvoie donc rien et on attend le renvoie automatique à la fin du timer. Le problème est de choisir un bon timer car un timer trop petit peut déclencher la retransmission d'un message alors que le **ACK** est en chemin et peut donc surcharger le système. À l'inverse un timer trop grand et le système réagit trop lentement.

(b) Ce protocole a été mis au point dans les années 70 pour des réseaux peu étendu et peu performant, aujourd'hui il n'est plus performant car les **RTT** (temps de propagation) sont devenus important, et avec ce protocole ils laissent beaucoup de temps mort.

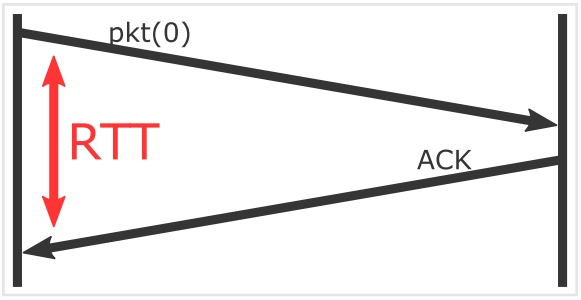


Fig. 5 - Le protocole **stop & wait** induit un temps mort

1.1.10. Expliquez le principe du parallélisme (pipelining).

Au lieu d'envoyer les packets un à un comme le protocole **stop & wait**, on envoie une rafale de packet avec les différents accusés de réceptions séparés. Ici on reste donc actif pendant une partie du **RTT**.
Il existe deux stratégies possible:

- le protocole à fenêtre glissante **GBN** (**Go-Back N**)
- protocole à fenêtre glissante **SR** (**Selective Repeat**)

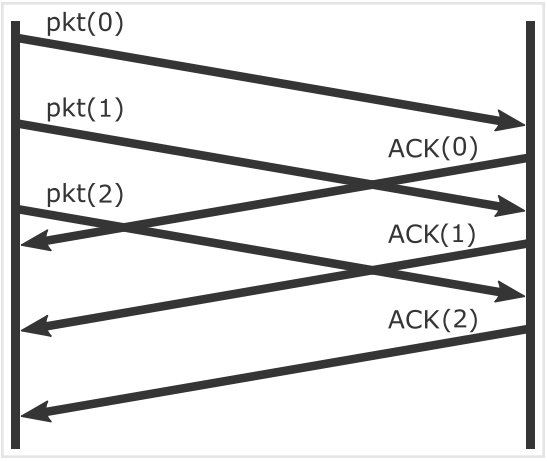


Fig. 6 - Le principe du parallélisme

1.1.11. Expliquez le principe d'un protocole à fenêtre glissante **GBN** (**Go-Back N**).

Le sender est autorisé à transmettre de multiples packets (si disponible) sans attendre d'**ACK**s, mais est contraint de ne pas avoir plus de N packets sans **ACK** dans le pipeline.
En résumé l'émetteur envoie N paquets d'un coup et le récepteur envoie N accusés de reception. De plus un **ACK** correspond aussi à la réception de tous les paquets précédents. Si on reçoit **ACK**(6), c'est que les 5 autres paquets ont été reçu.
Le système est donc plus robuste car si **ACK**(5) se perd, on ne renverra quand même pas ce paquet. Mais en contre-partie on n'accepte les paquets que dans l'ordre; si on recoit 6 avant 5, on détruit 6 (car à l'époque la mémoire était chère) et il faudra le renvoyer. Ce qui est problématique car le réseau est utilisé inutilement.

1.1.12. Quelle est la taille maximale de la fenêtre glissante **GBN** (**Go-Back N**), si les trames sont numérotées modulo k ? Pourquoi ?

On voit que le récepteur considère que le deuxième paquet qu'il reçoit la deuxième fois est le même que le premier qu'il a déjà reçu et non le troisième packet. Ainsi il le supprime et envoie un **ACK** pensant que le premier **ACK** s'est perdu, on perd donc le paquet et perturbe toute la chaîne.

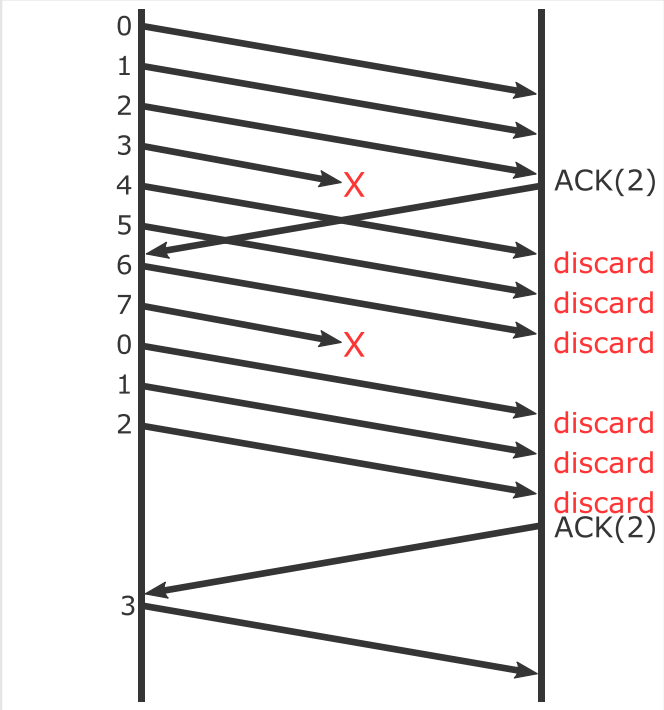


Fig. 7 - Taille maximale de la fenêtre glissante **GBN** (**Go-Back N**)

La taille maximale de la fenêtre est donc le nombre de numéro de `séquence-1` .

1.1.13. Expliquez le principe d'un protocole à fenêtre glissante **SR** (**Selective Repeat**).

GBN (**Go-Back N**) souffre, malgré ses améliorations, encore de quelques problèmes de performances. En effet, la moindre erreur dans un paquet peut entraîner la retransmission superflue d'**une grande quantité de paquets** pourtant **déjà arrivés** intègres.
Le protocole à **fenêtre glissante** *SR* (**Selective Repeat**) évite les retransmissions inutiles et demandent à l'expéditeur de ne retransmettre **que** les paquets susceptibles d'avoir été perdus ou corrompus lors de la transmission. Cela implique que le destinataire envoie un **ACK** pour **chaque paquet** correctement reçu.

Comme dans **GBN** (**Go-Back N**), une fenêtre permet de limiter le nombre de paquets en attente de confirmation. En revanche certains paquets de cette fenêtre auront **déjà** fait l'objet d'un accusé de réception. Le destinataire renvoie un accusé pour chaque paquet valide qu'il reçoit, qu'il soit dans l'ordre ou non, et stocke dans un tampon ceux qui sont encore temporairement hors séquence (notons que ce tampon ne dépassera jamais la taille de la fenêtre d'envoi de l'expéditeur).
Afin d'éviter les problèmes de confusions avec les numéros de séquence, la quantité de numéros disponible doit être **au moins deux fois plus grande** que la taille de la fenêtre (`K/2 = taille maximale`).

1.1.14. Quelle est la taille maximale de la fenêtre glissante **SR** (**Selective Repeat**), si les trames sont numérotées modulo k ? Pourquoi ?

On ne peut pas utiliser une fenêtre aussi grande que pour le **GBN** (**Go-Back N**).

$$Taille_{max} = \frac{\text{Nbr de numéro de packets}}{2} = \frac{K}{2}$$

Car il ne faut pas qu'un nombre de la fenêtre de l'émetteur corresponde à un nombre de la fenêtre du récepteur, mais d'un autre rang.
Exemple (fenêtre de taille 3 avec une numérotation de 4 chiffres):

> Èmetteur |0 1 2| 3 0 1 2 → récepteur 0 1 2 |3 0 1| 2

Le récepteur a reçu les paquets **(0, 1, 2)** et attend donc les paquets **(3, 0, 1)**, mais l'émetteur n'a pas reçu les **ACK**, il considère donc que les paquets **(0, 1, 2)** n'ont pas été reçu, et lors du time-out, il les renverra.
Mais le récepteur prendra le paquet 0 qu'il va recevoir pour celui qu'il attend, et non pour celui qu'il a déjà reçu.

Remarque: le débit étant lié à la taille de la fenêtre, il faut alors choisir un grand nombre de numéro de paquets.

1.1.15. Dans les protocoles à fenêtre glissante de type « **Selective Repeat** », quelles sont les relations qui sont satisfaites à tout instant entre les quatre valeurs suivantes: les bords inférieurs et supérieurs des fenêtres de l'émetteur et du récepteur ? Justifiez.

Quelque soit le protocole:

> Taille maximale de la fenêtre de l'envoyeur(Ns) + Taille maximale de la fenêtre du receveur(Nr) ≤ K

Pour le selective repeat, `Nr = Ns ≤ K/2` .

On peut donc conclure que le bord inférieurs du récepteur aura au maximum une valeur en plus que le bord suppérieur de la fenêtre émetrice. On peut également conclure que le bord inférieur du récepteur ne pourra jamais être avant le bord inférieur de l'envoyeur.
Cas limite avec une fenêtre de taille 3:

```
                                Envoyeur                        Récepteur
Initial:                  |0  1  2| 3  4  5  6     →      |0  1  2| 3  4  5  6

Récepteur le plus avancé:  |0  1  2| 3  4  5  6     →       0  1  2 |3  4  5| 6

Cas impossible:             0  1  2 |3  4  5| 6     →       0  1 |2  3  4| 5  6
```

1.1.16.

(a) Donnez 4 éléments majeurs des protocoles "**Go-Back-N**" et "**Selective Repeat**" qui permettent de les différencier.
(b) Pour chacun de ces éléments pris indépendamment, indiquez si **TCP** s'apparente d'avantage à l'un d'eux. Expliquez.
(c) Quelle optimisation supplémentaire, liée au contrôle d'erreur, **TCP** y apporte-t-il ?

(a)

GBN (Go-Back N)	**SR (Selective Repeat)**
Un seul **ACK**	**ACK** individuels pour chaque packet
Détruit les packet "en avance" et demande une reception "dans l'ordre"	Il existe un buffer du côté récepteur, on ne détruit donc pas les paquets arrivés "en avance"
Un seul timer correspondant au dernier packet	Timer individuels pour chaque packet
On renvoie tout les paquets à partir de celui sans **ACK**	On renvoie que les packets sans **ACK**

(b) La technique de fiabilité de **TCP** est un mixte entre la technique **GBN** (**Go-Back N**) et la technique **SR** (**Selective Repeat**).
Dans la première technique, on reprends la méthode d'un seul timer (associé au dernier paquet) ainsi que les **ACK** cumulatif (ACK(6) signifie qu'on ACK aussi 5 et les autres précédents).
Cependant dans le **GBN** (**Go-Back N**) on renvoyait toute la fenêtre après le **ACK** de retour, ici on reprend, dans la deuxièmème technique, le revoie seulement du dernier segment manquant et on verra avec les **ACK** futurs si il en manquait d'autres.

(c) L'optimisation "Fast retransmit" peut être mis en place, elle permet à **TCP** de détecter avant le timer qu'un segment est perdu. En effet, si on envoie les paquets par rafale, et qu'on recoit en retour plusieurs fois un même **ACK**, cela veut dire que plusieurs segments postérieurs sont arrivé avant le segment lié à cet **ACK**. Ce segment est donc soit perdu, corrompu ou en retard.
Par conventation, on considère que le segment est perdu après qu'on ai reçu trois **ACK** identiques.

1.1.17.

(a) En première approximation, quels sont les 3 paramètres qui influencent le débit d'une connexion **TCP** ? Expliquez.
(b) **TCP** garantit-il un partage équitable des ressources du réseau par les différentes connexions ? Pourquoi ?

(a) Il y a 3 paramètres qui peuvent entraîner des pertes et ainsi une baisse momentané du débit :
- **p**: *Taux d'erreur*;
- **W**: *Fenêtre de congestion* (mesuré en **MSS** (*Max Segment Size*));
- **RTT**: *Round-Trip Time*.

La formule de l'efficacité est $\sqrt{\frac{3}{2}}\frac{MSS}{RTT}$ et donc est meilleure avec des MSS plus gros et des petits RTT. Des packets plus gros dont "l'aller-retour" est rapide.

(b) Quand **TCP** est en concurrence avec **UDP**, **TCP** va ralentir mais **UDP** ne va pas réduire son trafic. **TCP** va donc être désavantagé par rapport à **UDP**.
On remarque également avec **TCP** qu'aux goulets d'étranglement, si on a 2 connections **TCP** concurrentes avec les même **RTT** et qu'une a commencée avant l'autre, elles vont se partager la bande passante de manière de plus en plus équitable au fur et à mesure que le temps avance. Si par contre elles n'ont pas le même **RTT**, celle qui aura le plus petit **RTT** sera avantagée. C'est à la fois une bonne chose et une mauvaise chose. En effet, si on regarde au niveau locale, les ressources ne sont pas réparties équitablement (mauvaise chose) mais au niveau globale, la connexion avec un **RTT** plus grand sera passée par plus de noeuds puisqu'elle vient de plus loin et donc aura consommé plus de ressources sur d'autres noeuds que la connexion avec un petit **RTT**.

1.1.18. 10 processus clients communiquent simultanément avec un processus serveur attaché au **port** 8000.

(a) Combien de sockets vont être ouverts par le serveur si les processus communiquent par **UDP** ? Pourquoi ?
(b) Même question s'ils communiquent par **TCP**.

(a) **UDP** ne crée pas de connexion persistante. L'emetteur ne fait que créer un packet en y attachant l'**IP** ciblé, le **port** ciblé ainsi que ses propres coordonnées puis l'envoie. Il ne se soucie pas si il atteint le recepteur ou non.
Ainsi le processus serveur attaché au **port** 8000 recevra (si il n'y a aucune perte) les 10 messages envoyés par les processus clients. Du coup 1 seul socket sera ouvert.

(b) **TCP** crée une connexion persistante, du coup il faut ouvrir et réserver un socket pour chaque processus client.
Du coup **10 sockets** seront ouverts. Mais il ne faut pas oublier le socket qui sert à écouter et à accepter les nouvelles connexions. Donc c'est un total de **11 sockets** qui seront ouverts.

1.1.19.

(a) Comment l'émetteur **TCP** détecte-t-il une congestion ?
(b) Décrivez le mécanisme de contrôle de congestion de **TCP**.
(c) Quelle distinction **TCP** fait-il entre congestion légère et congestion sévère ? Comment réagit-il dans chaque cas ?
(d) Si on néglige les effets du contrôle de flux, ce contrôle de congestion détermine largement le débit moyen d'une connexion **TCP**. Quand plusieurs connexions **TCP** sont en compétition, se partagent-elles la bande passante disponible de façon équitable ? Expliquez.

(a) **TCP** peut distinguer 2 types de **congestions**:
- Soit il reçoit 3 **ACK**s consécutifs pour le même numéro de séquence (donc un des paquets intermédiaire a été perdus, mais les suivants sont passés: **faible congestion**)
- Soit un **ACK** n'arrive pas dans le temps imparti (timeout, beaucoup de paquets perdus: **congestion sévère**).

(b) Au démarage de la transmission, **TCP** envoie les données avec une fenêtre de taille 1 **MSS** (**Maximum Segment Size**), correspondant au nombre de paquets qui peuvent être en parcourt simultanément. La taille de la fenêtre est doublée à chaque itération (en incrémentant la taille à chaque **ACK** reçu), de sorte qu'elle a une **croissance exponentielle**.
- S'il détecte une **faible congestion**, il divise la taille de la fenêtre par deux et change de mode pour incrémenter la taille de la fenêtre à chaque itération (+1 pour chaque fenêtre totalement envoyée) pour adopter une **croissance linéaire**. Il approche ainsi **dichotomiquement** la taille moyenne de fenêtre optimale (c'est à dire le nombre de paquets en parcourt, et donc la vitesse d'envoi).
- S'il détecte une **congestion sévère**, il réduit la taille de fenêtre à 1 et recommence en mode de **croissance exponentiel**. Il peut éventuellement repasser en mode de **croissance linéaire** lorsqu'il a atteint la moitié de la taille de fenêtre qui a provoqué un timeout (puisque doubler sa taille provoquera probablement de nouveau un timeout).

(c) Voir (a) et (b).

(d) Sachant que le timeout est calculé à partir du **RTT**, deux sessions **TCP** en compétition pour la même connexion approcheront toujours la vitesse optimale pour leurs **RTT**.
TCP répartira la connexion de façon équitable en ce sens que chaque session utilisera une bande passante inversément proportionelle à son **RTT**, évitant la retransmission en bloc de paquets inutiles sur des connexions trop lentes.

1.1.20.

(a) Expliquez la différence entre un simple ordinateur et un routeur
(b) Expliquez le principe d'une table de routage

(a) Un **routeur** est un matériel de la **couche 3** qui relie plusieurs réseaux. Sachant qu'une carte réseau ne peut être relié qu'à un seul réseau; il doit donc avoir une interface dans chacun des réseaux auquel il est connecté. C'est donc simplement une machine avec plusieurs interfaces (plusieurs cartes réseau), chacune reliée à un réseau. Son rôle va être d'**aiguiller les paquets reçus** entre les diférents réseaux.
Il y a enfaite peu de chose qui différencie un simple ordinateur d'un routeur. La principale différence est qu'un routeur **accepte de relayer** des paquets qui ne lui sont **pas destiné** alors qu'une simple machine les jettera à la poubelle.

(b) Imaginons que nous sommes un routeur ayant comme adresse MAC l'adresse `00:11:22:33:44:55` et comme adresse IP `192.168.0.1/24`. Nous recevons la trame suivante (dans laquelle nous indiquons aussi l'en-tête de couche 3) sur une de nos interfaces:

MAC Next	**MAC Prev**			**IP SRC**	**IP DST**		
00:11:22:33:44:55	01:2B:45:56:78:ED	IP	???	10.0.0.1	136.42.0.28	Données à envoyer	CRC

Une adresse MAC est propre à un réseau local (règle couche 2), on ne connait donc pas l'adresse mac source (MAC_{SRC}) ni l'adresse mac de destination (MAC_{DEST}). Ici, la trame arrive sur l'interface de notre machine ayant pour adresse IP `192.168.0.1/24`. Son réseau ne contient pas d'adresse UP `10.0.0.1` = la machine `10.0.0.1` ne fait pas partie du réseau. L'adresse MAC_{SRC} que nous voyons ici est celle du dernier routeur qui nous a envoyé la trame.

Que se passe-t-il quand notre machine reçoit cette trame ?
La carte réseau recoit la trame et lit l'adresse MAC `00:11:22:33:44:55`. C'est la sienne. Il lit la suite pour voir qui l'envoie et à quel protocole de couche 3 que la couche 2 doit l'envoyer. Il est inscrit IP, donc il envoie la trame en enlevant l'en-tête Ethernet, ce qui donne le datagramme IP, à la couche 3 et plus précisément au protocole IP.
La couche 3, donc le protocole IP, lit l'ensemble des informations de l'en-tête IP, puisqu'il sait maintenant que ce

datagramme lui est destiné. Et là, l'adresse IP de destination n'est pas la sienne. Ce qui est souvent normal pour un routeur de recevoir un message qui ne lui est pas destiné.
Son role va maintenant être d'aiguiller le datagramme vers sa destination. Il possède une table de routage dans laquelle est indiqué le prochain routeur auquel il doit envoyer le datagramme pour que celui-ci arrive à sa destination. La table de routage va donc lister les routeurs auxquels il peut envoyer son datagramme pour joindre une destination donnée. La destination donnée ne va pas être une machine, mais un réseau.

Le principe est d'avoir d'un côté la liste des réseaux que l'on veut joindre, et de l'autre la liste des routeurs à qui nous devons envoyer le datagramme pour joindre les réseaux. On appelle aussi ce routeur une passerelle.

Destination	**Gateway**	**Masque**	**Flags**	**Iface**
localhost	*	255.255.255.255	UH	lo0
124.178.240.0	*	255.255.255.0	U	eth1
124.178.48.0	*	255.255.240.0	U	eth0
124.178.64.0	*	255.255.224.0	U	eth2
124.178.96.0	124.178.64.2	255.255.224.0	UG	eth2
default	124.178.240.3	0.0.0.0	UG	eth1

Il est important de bien comprendre et retenir ce qui précède, car le routage est la base du fonctionnement d'Internet. Attention, toute machine connectée à un réseau possède une table de routage, même une imprimante, un téléphone, ou un PC,...

1.1.21.

(a) Expliquez l'établissement de connexion « 3-way handshake » utilisé dans le protocole de transport **TCP**, en indiquant les paramètres importants présents dans les échanges et leurs rôles.
(b) Expliquez avec l'aide d'un exemple pourquoi un « 2-way handshake » ne serait pas suffisant.

(a) Comme son nom l'indique, le **3-way handshake** se déroule en trois étapes:

1. **SYN** (*synchronized*):
   Le client qui désire établir une connexion avec un serveur va envoyer un premier paquet **SYN** au serveur. Le numéro de séquence de ce paquet est un nombre aléatoire A.

2. **SYN-ACK** (*synchronize-acknowledge*):
   Le serveur va répondre au client à l'aide d'un paquet **SYN-ACK**. Le numéro du **ACK** est égal au numéro de séquence du paquet précédent (**SYN**) incrémenté de un (A + 1) tandis que le numéro de séquence du paquet **SYN-ACK** est un nombre aléatoire B.

3. **ACK** (*acknowledge*):
   Pour terminer, le client va envoyer un paquet **ACK** au serveur qui va servir d'accusé de réception. Le numéro de séquence de ce paquet est défini selon la valeur de l'acquittement reçu précédemment par exemple A + 1 et le numéro du **ACK** est égal au numéro de séquence du paquet précédent (**SYN-ACK**) incrémenté de un (B + 1).

Une fois le **3-way handshake** effectué, le client et le serveur ont reçu un acquittement de la connexion. Les étapes 1 et 2 définissent le numéro de séquence pour la communication du client au serveur et les étapes 2 et 3 définissent le numéro de séquence pour la communication dans l'autre sens.
Une communication full-duplex est maintenant établie entre le client et le serveur.

(b) Le **2-way handshake** n'est pas suffisant car on saute l'etape 2 du **3-way handshake**. Si par exemple, un Client A veut parler avec un serveur B, il faut que B sache que A peut entendre ce qu'il dit. Car dans le **2-way handshake**, A envoi à B et B répond à A. Mais B ne sait pas si son message est reçu par A.
Il n'y a donc également aucune confirmation que le client A ai bien pris le bon numéro de packet pour le serveur B.

1.1.22.

(a) Décrivez l'architecture générique d'un routeur et le rôle de chaque composant.
(b) Comment peut-on perdre des paquets dans les **ports** d'entrée ?
(c) Comment peut-on perdre des paquets dans les **ports** de sortie ?
(d) Qu'est-ce que le blocage HOL ?

(a) Quatre éléments principaux peuvent être identifiés dans l'architecture d'un routeur :
- **Buffer d'entrée**:
  Prend en charge les fonctionnalités de la couche physique et de la couche liaison. Il assure une fonction de consultation et d'acheminement des paquets entrant dans la matrice de commutation vers le **port** de sortie approprié.
- **Buffer de sortie**:
  Emmagasine les paquets qu'il reçoit de la part de la matrice de commutation. Assure les fonctionnalités inverses de la couche liaison et de la couche physique.
- **Matrice de commutation**:
  Relie les **ports** d'entrée et les **ports** de sortie.
- **Processeur de routage**:
  Chargé de l'exécution des protocoles de routage, de la mise à jour des informations et des tables.

(b) Le point le plus problématique est le fait que les routeurs doivent opérer à des vitesses élevées, impliquant des millions de consultations par seconde; on souhaite en effet que l'opération de consultation soit plus rapide que le temps qu'il faut au routeur pour recevoir un nouveau paquet. Dès que le **port** de sortie a été identifié, le paquet peut entrer dans la matrice de commutation. Cependant un paquet peut se voir refuser temporairement l'accès si elle est occupée à traiter des paquets qui ont été reçus sur d'autres liaisons d'entrée. Il est alors placé en file d'attente sur son **port**. Si la matrice de commutation n'est pas assez rapide, et du coup à mesure de l'augmentation de ces files, le risque de perte de paquets augmente.

(c) Si aucun ou peu de phénomène de mise en attente ne se produit à l'entrée, mais que tous les paquets venaient à être dirigés vers le même **port** de sortie, il serait très rapidement saturé. Un gestionnaire de paquets doit déterminer quel paquet de la file doit être transmis. Cette règle est généralement la règle **FIFO** (**First In First Out**).

(d) **HOL** (**Head Of the Line blocking**) signifie que le routeur n'a pas de vue globale des buffer input, il ne prend donc en compte que le premier paquet du buffer. Si deux paquets doivent aller au même **port** de sortie, un des paquets se retrouve en attente avec tous ceux qui le suivent, même si ceux-ci doivent aller à un **port** actuellement libre.

1.1.23.

(a) Décrivez le protocole **CSMA**.
(b) Pourquoi et comment a-t-il été amélioré ?
(c) Citez les paramètres qui caractérisent un réseau **CSMA**. Quelle relation entre ces paramètres faut-il viser pour que le réseau **CSMA** ait des performances acceptables ? Expliquez.

(a) La technologie **CSMA** (**Carrier Sense Multiple Access**) est un ensemble de protocoles qui permet à plusieurs machines d'utiliser un même média de communication. Celle-ci vérifie que le support est disponible avant de commencer l'envoi d'une trame.
Dans la version simple de **CSMA**, une machine ne peut pas transmettre si elle détecte de l'activité sur le média, elle attend la fin de la transmission. Cependant, en raison du temps de propagation, surtout sur des longues distances, deux machines pourraient considérer le bus comme libre et commencer à écrire en même temps, pour se retrouver en collision quelques instants après. Il est à noter qu'il est pas possible d'éliminer les collisions sur un bus mais il est possible de développper une méthode pour les limiter et réussir à partager le bus entre plusieurs machines.

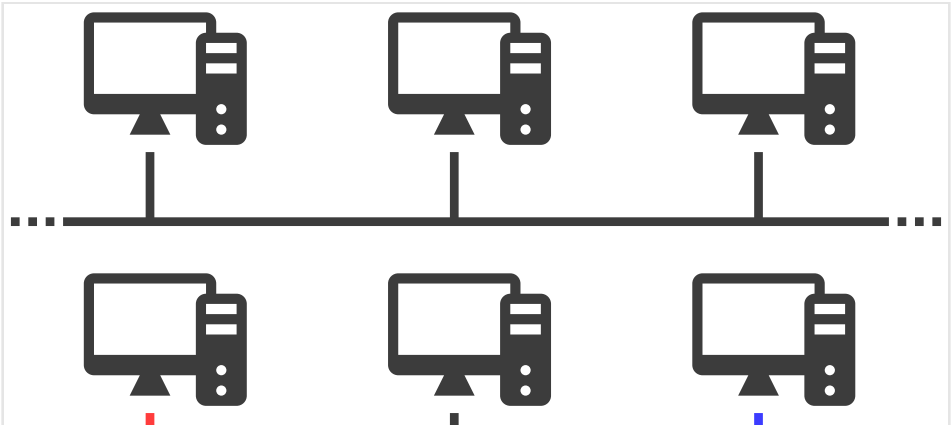


Fig. 8 - Illustation d'une collision

(b) La technologie **CSMA** a été amélioré par différents modes d'accès:

- **CSMA persistant**:
  Consiste à toujours écouter le canal, et on émet dedans dès qu'il est libre.

- **CSMA non-persistant**:
  Dans ce mode chaque station vérifie régulièrement que le média soit disponible. Si ce n'est pas le cas, elle attend un temps **aléatoire** pour revérifier si le média est enfin libre. Enfin, lorsque le média n'est pas occupé, la station transmet ses informations immédiatement. Cette approche réduit les collisions mais le temps d'attente initial peut être très long.

- **CSMA p-persistant**:
  Si une station souhaite émettre et que le canal est libre, elle émet avec une probabilité `p` . Sinon, elle attend un intervalle de temps avant de retenter avec la même probabilité `p` .
  En cas de collision, la station attend un temps aléatoire avant de recommencer la procédure. Cela permet d'éviter que deux stations se jettent sur le canal en même temps.
  La difficulté est d'avoir un bon `p` : trop bas et le délai augmente, trop haut et on a plein de collisions.

(c) Pour que **CSMA** ait des performances acceptables, il faut prendre en compte:

- B (la capacité en bits/s du canal)
- F (la taille des trames)
- L (la taille du canal)
- c (la vitesse de propagation)
- $tau = \frac{L}{c}$ (délai de propagation)
- $T = \frac{F}{B}$ (délai de transmission)

`tau` est équivalent à la durée pendant laquelle une collision peut survenir. Après un temps `tau` , le canal est réservé implicitement pendant `T-tau` secondes, car s'il n'y a pas eu de collision après un temps `tau` , tout le monde est au courant que le canal est pris et il n'y a plus de collision possible. Les performances du réseau sont donc meilleures quand

$$a = \frac{tau}{T} = \frac{BL}{cF}$$

est très petit, donc quand la période dangereuse pendant lesquelles des collisions peuvent survenir est très petite par rapport au temps total d'une trame. Il faut donc travailler avec de grandes trames, réduire la taille du réseau ou réduire le débit.

1.1.24.

(a) Qu'est-ce que le **CSMA/CD** ? En quoi améliore-t-il le **CSMA** ?
(b) Quelle contrainte le **CSMA/CD** introduit-il par rapport au **CSMA** ? Pourquoi ?
(c) IEEE 802.3 (plus communément appelé **Ethernet**) est un protocole de type **CSMA/CD** dont la méthode d'accès a été améliorée. Quelle est cette amélioration ?
(d) Expliquez pourquoi, si l'on veut garder le même format de trame, la méthode **CSMA/CD** exige de raccourcir le réseau pour atteindre des débits plus élevés. Il est toutefois possible de ne pas respecter cette longueur maximale du réseau, qui devient très contraignante à haut débit. Dans quelles conditions ?

(a) La technologie **CSMA** (**Carrier Sense Multiple Access**) est un ensemble de protocoles qui permet à plusieurs machines d'utiliser un même média de communication. Celle-ci vérifie que le support est disponible avant de commencer l'envoi d'une trame.
Dans la version simple de **CSMA**, une machine ne peut pas transmettre si elle détecte de l'activité sur le média, elle attend la fin de la transmission. Cependant, en raison du temps de propagation, surtout sur des longues distances, deux machines pourraient considérer le bus comme libre et commencer à écrire en même temps, pour se retrouver en collision quelques instants après. Il est à noter qu'il est pas possible d'éliminer les collisions sur un bus mais il est possible de développper une méthode pour les limiter et réussir à partager le bus entre plusieurs machines.

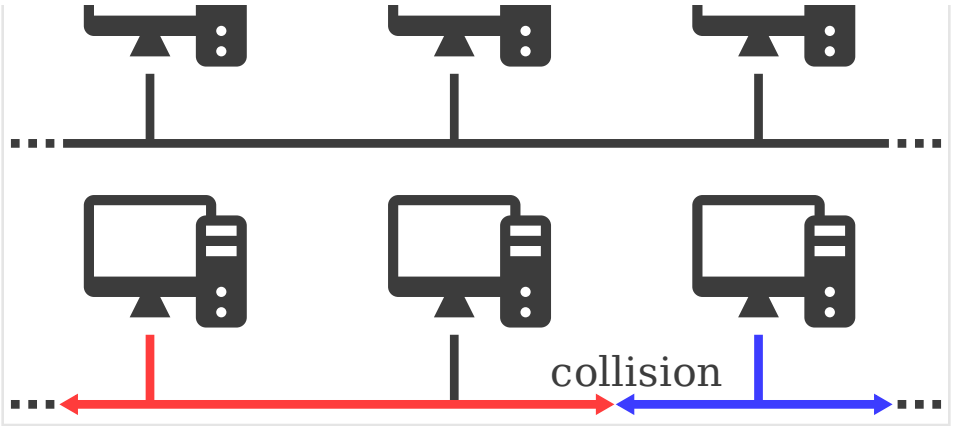


Fig. 9 - Illustation d'une collision

Le **CSMA/CD** rajoute la contrainte **Collision Detection**: si, lorsqu'un bit a été écrit, l'état mesuré est différent, la machine considère qu'il y a collision et arrête immédiatement d'écrire sur le bus. Elle attend ensuite pour un temps déterminé aléatoirement afin que deux machines en collision ne recommencent pas à émettre en même temps.
Son objectif est de limiter le nombre de collisions en organisant le droit à la parole. L'idée est de mettre en place une règle qui permettrait de n'avoir presque plus de collisions.

(b) Le **CSMA/CD** rajoute la contrainte **Collision Detection** car il est possible qu'en raison du temps de propagation, surtout sur des longues distances, deux machines pourraient considérer le bus comme libre et commencer à écrire en même temps.

(c) **Ethernet** repose sur un algorithme d'accès multiple **CSMA/CD** dont la méthode d'accès a été améliorée. Pour que l'émetteur détecte une collision, il doit être en train d'émettre. Il faut donc que la trame soit suffisament longue pour que le délai soit suffisament long et que l'emetteur entend la collision. Sinon l'émetteur transmet tout puis n'entend pas qu'il a eu de collision. Sur un réseau de 10 Mbps et 2.5 km, il faut qu'une trame fasse au moins 250 bits. Ethernet a choisi 512 bits minimum. Pour encore améliorer les choses, l'émetteur attend de manière exponentielle (entre `0` et `2collisions-1` temps de transmission de 512 bits) quand il détecte une collision. Ainsi, si les collisions sont rares, on n'attend pas trop. S'il y en a plein, on attend plus longtemps pour augmenter l'efficacité. Quand une collision survient, l'émeteur envoie un **jam**: un signal fort histoire que tout le monde sur la ligne soit au courant de la collision.

(d) F : Frame Size
B : Bandwidth
L : Ingueur du canal
F_{min} : $\frac{2BL}{C} = \pm BL * 10^{-8}$ bits

Si `F` est plus petit, le délai de transmission est plus court et donc on peut envoyer plus de trames et éviter des collisions.

Pour éviter cela, efficacité du réseau = $\frac{1}{1 + 2BL + \frac{e}{CF}}$

On doit donc augmenter la bande passante ou la distance.

1.1.25. On ne peut pas dire que les commutateurs **Ethernet** exécutent un protocole de routage (au sens de la couche 3), mais ils construisent toutefois des tables d'acheminement comme si un protocole de routage était à l'oeuvre. Expliquez comment ces tables sont construites, y compris quand plusieurs commutateurs sont inter-connectés.

Le commutateur **construit donc dynamiquement** une table qui associe des adresses **MAC** avec les ports correspondant. Lorsqu'il reçoit une trame destinée à une adresse dans cette table, le commutateur renvoie la trame sur le port correspondant. Si port destination = port émetteur, la trame n'est pas transmise. Si l'**adresse du destination est inconnue** dans la table, alors la trame est traitée comme un **broadcast**, c'est-à-dire qu'elle est transmise à tous les ports du commutateur sauf celui de réception.

1.1.26.

(a) Décrivez le protocole ARP.
(b) Expliquez pourquoi la table ARP est necessaire.
(c) Décrivez le déroulement de A à Z d'une requête ARP.
(d) A quel couche appartient ce protocole ?

(a) La machine A veut communiquer avec la machine B. S'il regarde sa table de routage, il sait qu'il doit passer son message par la passerelle par défault pour joindre B. Donc pour lui envoyer la trame, il doit connaître son adresse MAC. Or, il ne la connait pas.
Il faudrait pouvoir lui demander son adresse MAC mais pour lui demander il faudrait connaître son adresse MAC. C'est pour cela que le protocole ARP a été mis en place. Il peut envoyer un message à l'adresse de broadcast en demandant "est-ce que `IP` peut m'envoyer son adresse MAC ?" Grâce à l'adresse de broadcast ce message sera envoyé à tout le monde, et donc la cible le recevra et pourra lui renvoyer son adresse MAC.
ARP est donc un protocole qui permet d'associer une adresse MAC de couche 2 à une adresse IP de couche 3.

(b) Les broadcasts risquent de saturer le réseau à chaque fois que l'on veut envoyer une information. C'est pour cela qu'on a mis aussi en place la table ARP. Pour éviter d'avoir à renvoyer en permanence des broadcasts ARP à chaque fois que l'on veut envoyer une information à une machine, nous allons utiliser une table qui va garder les associations adresses IP <-> Adresses MAC pendant un court moment.
Ainsi, si j'envoie un paquet à ma passerelle, je noterai son adresse MAC dans ma table ARP et la prochaine fois que je voudrai lui parler, je n'aurai plus à envoyer de broadcast sur le réseau.

La table ARP va donc associer adresse **IP et adresse MAC** correspondante. Les informations contenues dans la table ARP ont une durée de vie limitée. En gros, une valeur va rester environ deux minutes dans la table avant d'être effacée s'il n'y a pas eu de dialogue avec cette adresse entre-temps. C'est pour cela que l'on dit que la table ARP est **dynamique**. Elle évolue au cours du temps en fonction des machines avec lesquelles il dialogue.

(c) Exemple: nous sommes la machine `192.168.0.1` et voulons envoyer un message à la machine `192.168.1.2` .
Nous savons que nous voulons joindre d'abord le routeur `192.168.0.254` , mais ne connaissons pas son adresse MAC.

C'est là que le protocole ARP entre en jeu:
- On regarde d'abord dans la table ARP locale si on possède l'association entre l'adresse IP `192.168.0.254` et son adresse MAC;
- Si on la possède, on envoie l'information et c'est terminé;
- Sinon, on envoie un broadcast ARP sur le réseau;
- La machine `192.168.0.254` va nous répondre avec son adresse MAC;
- Nous allons noter cette adresse MAC dans notre table ARP;
- Nous allons enfin pouvoir envoyer notre information.

Voilà comment font les machines pour passer d'une adresse IP à joindre à l'adresse MAC correspondante: grâce au protocole ARP !

(d) Le protocole ARP est un protocole de couche 2 **et** 3 ! Il manipule des informations de couche 2: les adresses MAC, et des informations de couche 3: les adresses IP.
Ainsi, on dit que ce protocole est "à cheval" entre ces deux couches.

1.1.27. Un chercheur connecte son ordinateur portable à un commutateur **Ethernet** de son département. Il démarre son browser pour afficher la page web de `www.google.com` .

(a) Identifiez les protocoles mis en oeuvre, et dans l'ordre chronologique, entre le moment où l'ordinateur se connecte et le moment où la page d'accueil de Google s'affiche.
(b) Précisez au passage le rôle de chaque protocole et décrivez-les succinctement.

(a) Une topologie d'exemple compreant des switches et des routeurs est utilisée. On suppose que le routeur dispose d'un serveur **DHCP**
1. On connecte le laptop, avec un navigateur web ouvert;
2. Le laptop cherche un serveur **DHCP** sur le réseau et envoie une trame `DHCPDISCOVER` comme un broadcast à l'adresse MAC `ff:ff:ff:ff:ff:ff` ;
3. Le switch recoit la trame et la relaye en la broadcast partout;
4. La trame arrive au routeur, qui reconnait la demande **DHCPDISCOVER** et répond avec un **DHCPOFFER**. Il va alors proposer une adresse **IP**, un masque, une passerelle par défaut ainsi qu'un serveur **DNS**;
5. Le laptop apprend son adresse **IP**, l'**IP** du routeur, son serveur **DNS**, etc;
6. Le laptop répond par un **DHCPREQUEST**. Celui-ci est aussi envoyé en broadcast et sert à prévenir quelle offre est accepté.
7. Le serveur DHCP dont l'offre a été acceptée valide la demande et envoie un `DHCPACK` qui valide l'allocation du

bail (on parle en effet de "bail", car cette attribution d'adresse IP a une durée limitée. Une fois expiré, il faut redemander une adresse IP);
8. Le laptop forge une requête **DNS**;
9. Le laptop envoie un **WHO IS ARP** pour trouver l'adresse **MAC** du routeur;
10. Le laptop recoit cette adresse **MAC** et le switch retient quelle est l'adresse **MAC** du laptop;
11. La requête **DNS** est envoyée sur le réseau et arrive au routeur;
12. Le routeur décapsule la trame, reconnait le paquet **IP** qu'elle contient, et voit que l'**IP** de destination est hors du subnet. Il envoie donc ce paquet sur Internet;
13. Le serveur **DNS** de Google finit par recevoir le paquet, et répond;
14. Le laptop obtient l'**IP** du serveur de Google;
15. Un message **TCP** SYN est envoyé à cette **IP**;
16. Le serveur ouvre un socket et renvoie un ACK;
17. Le laptop envoie alors sa requête HTTP;
18. Le serveur renvoie les réponses;
19. La page web apparait dans le navigateur web.

DHCP (aves tous les protocoles des couches inférieurs) pour récupérer une adresse **IP** pour se connecter. Ensuite une requète **DNS** pour savoir quelle est l'adresse **IP** de Google. Mais on a besoins d'une requète **ARP** pour connaitre la bonne adresse **MAC** du serveur **DNS**. Des protocoles de routages (**RIP**,**OSPF**,..). On établi une connexion **TCP** avec le serveur de Google. On peut finalement faire notre requête **HTTP**. On se rend compte qu'au démarrage, tout un tas de protocoles sont utilisés. Les paquets sont emboités les uns dans les autres, il y a plein de couches et plein d'acteurs qui intéragissent. On a également vu qu'un lien est compliqué et qu'il cache plein de choses. Un lien est en fait un assemblage d'élément qui se présente à **IP** comme si c'était un lien physique, ne touchant pas à la couche 3 du réseau.

(b)
- **DHCP** (*Dynamic Host Configuration Protocol*):
  Utilisé pour l'attribution dynamique d'adresse **IP**.
- **DNS** (*Domain Name System*):
  Faire la correspondance entre les noms de domaines et les adresses **IP**.
- **ARP** (*Adress Resolution Protocol*):
  Pour connaitre l'adresse **MAC**.
- **Routage**
  Permet de trouver le chemin adéquat à partir de notre machine vers celle avec laquelle on veut communiquer.
- **TCP** (*Transmission Control Protocol*):
  Protocole de transfert des données fiables orienté connexion.
- **HTTP** (*Hypertext Transfer Protocol*):
  Pour récupérer des pages web, protocole de la couche d'application.

1.1.28.

(a) Décrivez le protocole de vecteur de distance (DV).
(b) Décrivez le protocole d'états de liens (LS).
(c) Expliquez les différences fondamentales entre ces deux protocoles.

(a) Le protocole de ce type ont un fonctionnement assez simple :
- Tous les routeurs du réseau transmettent à interval régulier des mises à jours sur tous leurs ports. Ces updates sont des paquets IP mit en broadcast (`ff:ff:ff:ff:ff:ff`) et contenant la quasi totalité de leur table de routage.
- Tous les routeurs recoivent ces updates en comparant le contenu avec les entrées de leur table de routage.
- Chaque routeur a donc l'impression d'être le centre du réseau, il ne voit qu'un chemin à empreinter vers un routeur pour atteindre un autre routeur.

Les vecteurs de distance présentent au moins deux inconveinients majeurs:
- La charge réseau induite par les updates est conséquent, chaque routeur met régulièrement sur le réseau l'ensemble de sa table de routage. Ce trafic est non négligeable surtout dans le cas de grands réseaux.
- Le protocole n'est aussi pas très réactif avec le cas de **RIP**.

Les protocoles à vecteur de distances s'appuient sur l'algorithme de Ford-Bellman.

(b) Suite aux inconvéinients produits par le protocole de vecteur de distance (DV), le protocole d'**état de lien** (LS) a été inventé. À l'inverse de ces protocoles de vecteurs, les protocoles dits à états de lien comme **OSPF** s'appuient sur l'algorithme de Dijkstra et chaque routeur connait l'**entièreté de la topologie du réseau**:
- Chaque routeur transmet, comme précédemment, des updates à ses voisins mais uniquement lors d'un changemant d'état (coûts, indisponibilité, etc, ...)
- Avec un algorithme implémenté, le routeur construit le réseau et le maintient à jour à chaque update.

Malheureusement, il y a au moins quelques inconvénients:
- Les routeurs nécéssitent des performances CPU et des capacités mémoires supérieures que pour les vecteurs de distance (plus cher!)
- Ce type de protocole très réactif nécessite des réglages précis des timers de transmission car sa réactivité peut justement nuire à la stabilité du routage (dans le cas de micro-coupure de liens, on risque de changer toutes les tables et ca peut nuire au réseau).

(c)

Vecteur de distance (DV)	**État de lien (LS)**
Tous les routeurs du réseau transmettent à interval régulier des updates sur tous leurs ports.	Les routeurs n'émettent des updates que lorsque des liens changent d'état (coûts, indisponibilité, etc, ...)
les updates émis contiennent la quasi totalité de leur table de routage	Les updates émis ne contiennent que des descriptions de liens ayant changé d'état, ils sont donc moins volumineux.
Les routeurs doivent attendre l'interval d'update pour recevoir l'information.	Les routeurs retransmettent immédiatement les mises à jour à leurs voisins, ils compilent leur table de routage après, le protocole est donc beaucoup plus réactif !

1.1.29. Décrivez le protocole RIP.

Routing Information Protocol (RIP, protocole d'information de routage) est un protocole de **routage IP** de type Vector Distance (à vecteur de distances) s'appuyant sur l'algorithme de détermination des routes décentralisé Bellman-Ford. Il permet à chaque routeur de communiquer aux routeurs voisins la métrique, c'est-à-dire la distance qui les sépare d'un réseau IP déterminé quant au nombre de sauts ou « hops » en anglais.
Pour chaque réseau IP connu, chaque routeur conserve l'adresse du routeur voisin dont la métrique est la plus petite. Ces meilleures routes sont diffusées toutes les 30 secondes.

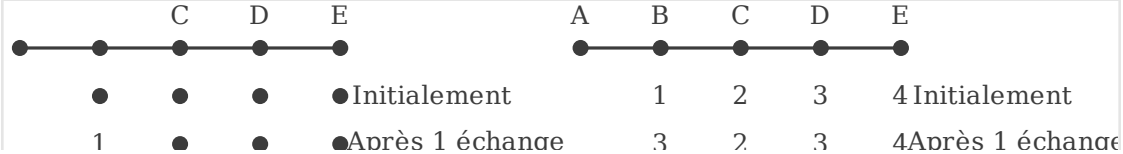
Les limitation de RIP:
- Pour éviter les boucles de routage, le nombre de sauts est limité à 15. Au-delà, les paquets sont supprimés.
- RIP ne prend en compte que la distance entre deux machines en ce qui concerne le saut, mais il ne considère pas l'état de la liaison afin de choisir la meilleure bande passante possible. Si l'on considère un réseau composé de trois routeurs A, B et C, reliés en triangle, RIP préférera passer par la liaison directe A-B même si la bande passante n'est que de 56 kbit/s alors qu'elle est de 20 Mbit/s entre A et C et C et B.

Ces limitations sont corrigées dans le protocole OSPF.

1.1.30.
(a) Dans quelle(s) situation(s) le protocole de routage à vecteur de distances (DV) risque-t-il de ne pas converger ?
(b) Décrivez un comportement pathologique possible à l'aide d'un exemple simple.
(c) Comment peut-on atténuer ce phénomène ? Expliquez.

(a)
- Des boucles de routage.
- Le trafic est acheminé de façon incohérente.
- Des entrées de table de routage incohérentes.

(b) Pour constater la rapidité avec laquelle les bonnes nouvelles se propagent, considérez le sous-réseau illustré à la figure et sur lequel la métrique utilisée est le nombre de sauts. Supposons que A soit inactif au départ et que tous les autres routeurs le sachent.
En d'autres termes, ils ont tous enregistré un délai infini vers A.



1	2	3	Après 2 échanges	3	4	3	Après 2 échange
1	2	3	Après 3 échanges	5	4	5	Après 3 échange
1	2	3	Après 4 échanges	5	6	5	Après 4 échange
				7	6	7	Après 5 échange
				7	8	7	Après 6 échange
				⋮			
				●	●	●	●
		(a)			(b)		

Fig. 10 - Comportement pathologique

(c) Solution: définir un nombre maximum de sauts, comme via le **Route poisoning**: lorsqu'une route vers un réseau tombe, le réseau est immédiatement averti d'une métrique de distance infinie (le maximum de sauts +1), plus aucune incrémentation n'est possible.

Si `Z` passe par `Y` pour aller à `X`, `Z` peut faire croire à `Y` qu'il se trouve à une distance infinie de `X`. Y étant persuadé que Z ne peut atteindre X, il opte pour une autre route.
Cette méthode n'est bonne qu'avec des triplets, elle ne marche pas lorsqu'il y a des boucles de plus de trois noeuds.

Le poison reverse (Route poisoning) est une méthode pour prévenir qu'une route...

1.1.31.

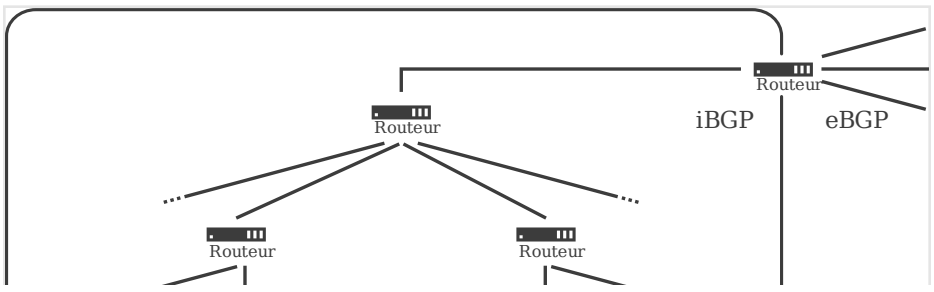
(a) Considérez un protocole de routage à états de liens (link state). Décrivez le contenu des paquets de routage, expliquez le rôle de chaque champ, et décrivez la méthode de diffusion des paquets.
(b) En quelques mots, en quoi est-ce fondamentalement différent des protocoles à vecteur de distances ?

(a)
1. Le nom de la source.
2. Un numéro de séquence: il permet de savoir à quel paquet le routeur en est (pour ne pas mettre à jour avec un vieux paquet).
3. Un âge
4. Le nom de ses voisins et les coûts associés (qui ne sont pas obligatoirement identiques pour les 2 sens d'une arête.
5. Des **ACK** flags et des Send flags pour savoir. Le send flag permet de savoir qui a déjà reçu le paquet et à qui le paquet n'a pas encore été envoyé, tandis que l'**ACK** flag sert juste à savoir si on a renvoyé l'**ACK** de ce paquet à un certain nœud.

Chaque noeud reçoit les mêmes infos. Tous les noeuds envoient toutes les infos à tous ses voisins qui n'ont pas encore reçu l'info. Ensuite ils doivent chacun effectuer un Dijkstra pour chaque destination. Pour les protocoles à vecteur de distances, on utilise la programmation dynamique. Chaque noeud envoi périodiquement son vecteur de distance à ses voisins de manière asynchrone. Quand un noeud reçoit un vecteur de distance, il met à jour le sien et si un chemin change il avertit ses autres voisins. Les algorithmes à vecteur de distance sont moins robustes mais beaucoup plus rapides.

(b) Contrairement au DV, la distance vers une destination n'est pas calculée au fur et à mesure en sommant le prochain coût estimé par le prochain saut et le coût pour atteindre ce prochain routeur. On calcule la distance directement via un arbre (spanning tree).

1.1.32. Expliquez se qu'est un **Autonomous System** (**AS**)



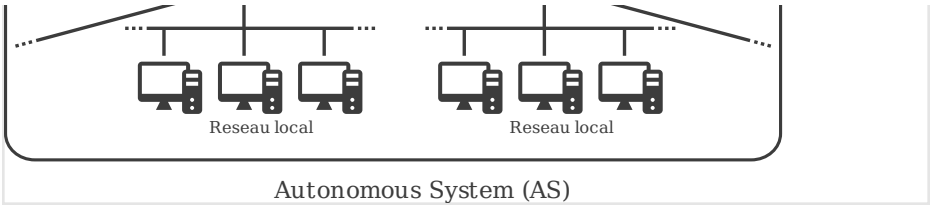


Fig. 11 - **Autonomous System** (**AS**)

Un **AS** est un ensemble de réseaux informatiques IP intégrés à Internet et dont la politique de routage interne est cohérente. Un AS est généralement sous le contrôle d'une entité ou organisation unique, typiquement un fournisseur d'accès à Internet. Au sein d'un AS, le protocole de routage est qualifié d'« interne » (par exemple, **Open shortest path first**, abrégé en **OSPF**). Entre deux systèmes autonomes, le routage est «externe» (par exemple Border Gateway Protocol, abrégé en **BGP**).

- Au sein de chaque AS on implémente des protocoles de routage qui permettent aux routeurs internes à l'AS et aux routeurs de bordure de l'AS de construire leurs tables de routage. Ces tables, bien sûr, ne connaissent que les réseaux IP internes à l'AS. Ces protocoles sont appelés des IGP: Interior Gateway Protocol.
- les routeurs de bordures des différentes AS sont interconnectés entre eux et échangent des informations sur le contenu des AS grâce à un protocole de routage. Ce protocole permet de contrôler parfaitement les informations transmises. Un AS n'aura donc pas forcément connaissance de tous les réseaux existants dans un AS voisin. Ces protocoles de routage sont appelés des EGP: Exterior Gateway Protocol.

1.1.33. Le protocole de routage inter-domaine **BGP** est plus apparenté à la famille des protocoles de routage intra-domaine à vecteur de distances (DV) qu'à celle des protocoles à état de lien (LS).

(a) Expliquez deux ressemblances importantes entre **BGP** et un protocole DV.
(b) Expliquez deux différences importantes entre **BGP** et un protocole DV, et leur raison d'être.

(a)
1. Obtention de la connaissance des voisins
2. Propage les informations dans tout le réseau.

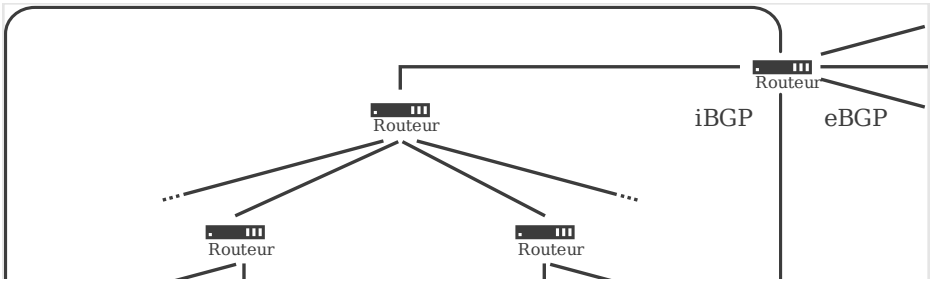
(b)
1. **BGP** mémorise toutes les routes vers toutes les destination: récupération rapide lorsqu'une destination devient inaccessible via la route initialement choisie.
2. **BGP** construit des routes sans boucles: le chemin suivi est décrit explicitement à l'aide des **Autonomous System** (**AS**) traversés.
3. Détection facile des boucles

1.1.34.

(a) Nommez et expliquez succinctement les 2 grandes familles de protocoles de routage intra-domaine (IGP) en insistant sur leurs différences.
(b) Expliquez en quoi et pourquoi le protocole de routage inter-domaine de l'Internet (**BGP**) est différent des protocoles de routage intra-domaine (IGP) déployés dans les divers systèmes autonomes (**AS**) qui composent l'Internet.

(a)
- **RIP** (**Routing Information Protocol**) : utilise un algorithme de vecteurs de distance qui se limite à 25 noeuds par paquet. Au-delà, il faudra plusieurs paquets. 0 représente l'infini. Les vecteurs de distance sont envoyés toutes les 30 secondes à ses voisins. **RIP** possède un mécanisme pour détecter les pannes, si un router n'a pas reçu de message d'un voisin depuis 180 secondes, il le déclare mort. **RIP** est un protocole de la couche d'application (PAS de la couche réseau), les messages sont envoyés dans des paquets **UDP**.
- **OSPF** (**Open Shortest Path First**) : utilise un algorithme d'état des lien. Chaque message est envoyé seulement à ses voisins qui le renvoient à leurs voisins,.. jusqu'à ce que l'entièreté du système autonome ai reçu le message. Les messages sont directement envoyés depuis la couche de transport (PAS partie de la couche d'application).

(b)



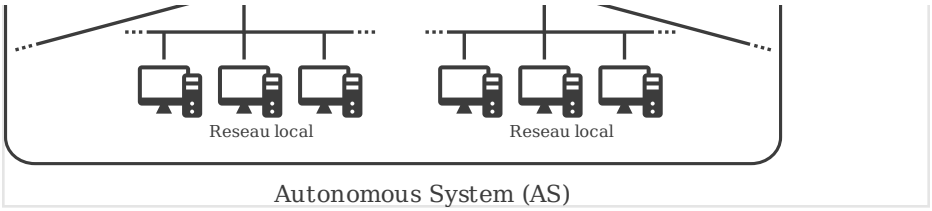


Fig. 12 - **Autonomous System** (**AS**)

Les connexions entre deux voisins **BGP** (neighbors ou peers) sont configurées explicitement entre deux routeurs. Ils communiquent alors entre eux via une session TCP sur le port 179 initiée par l'un des deux routeurs. **BGP** est le seul protocole de routage à utiliser TCP comme protocole de transport.
Il existe deux versions de **BGP** : Interior **BGP** (**iBGP**) et Exterior **BGP** (**eBGP**). **iBGP** est utilisé à l'intérieur d'un Autonomous System alors que **eBGP** est utilisé entre deux AS.
En général, les connexions **eBGP** sont établies sur des connexions point-à-point ou sur des réseaux locaux (un Internet Exchange Point par exemple), le **TTL** des paquets de la session **BGP** est alors fixé à 1. Si la liaison physique est rompue, la session **eBGP** l'est également, et tous les préfixes appris par celle-ci sont annoncés comme supprimés et retirés de la table de routage.
À l'inverse, les connexions **iBGP** sont généralement établies entre des adresses logiques, non associées à une interface physique particulière. Ceci permet, en cas de rupture d'un lien physique, de conserver la session **iBGP** active si un lien alternatif existe et si un protocole de routage interne dynamique (**IGP**) est employé (par exemple **OSPF**).
En conclusion, **IGP** sert au routage interne alors que **BGP** sert au routage externe. De plus, **BGP** ne communique qu'avec leurs voisins directs avec un **TTL** de 1.

1.1.35.

(a) Décrivez les principes du protocole de routage inter-domaine **BGP**.
(b) Expliquez comment **BGP** permet à un réseau périphérique (« stub ») multi-connecté (« multihomed ») de ne pas accepter du trafic de transit.

(a)

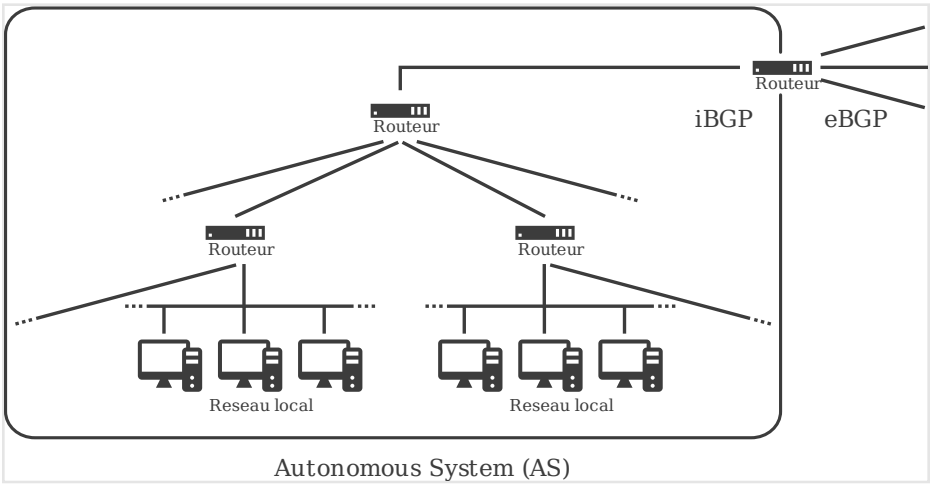


Fig. 13 - **Autonomous System** (**AS**)

Le protocole inter-domaine **BGP** est la version Exterior **BGP** (**eBGP**) qui est utilisé est l'extérieur et entre deux **Autonomous System** (**AS**) alors que Interior **BGP** (**iBGP**) est utilisé à l'intérieur d'un **Autonomous System** (**AS**). **eBGP** crée un connexion **TCP** permanente entre les autres **Autonomous System** (**AS**) qu'il peut atteindre, afin de se mettre à jour et d'envoyer des "promesses" (dire qu'il peut atteindre). Quand un routeur de la frontière utilisant **eBGP** reçoit ces mises à jour, il utilise **iBGP** pour communiquer ce qu'il a reçu aux routeur internes.

Le protocole **BGP** permet donc à chaque **Autonomous System** (**AS**) d'obtenir:

- Des informations sur la manière d'attendre un autre **Autonomous System** (**AS**) via **eBGP**.
- De propager ces informations aux routeurs internes via **iBGP**.

(b) ???? De trouver des bons chemins basés sur non-seulement les informations obtenues dans les points précédents mais en plus sur des polices. ?? ???? Les noeuds peuvent retenir de l'information, tel qu'un chemin qu'il connait qui ne lui rapporte rien. ????

1.1.36.

(a) Décrivez la règle de routage « Hot potato » et sa raison d'être
(b) Par quel protocole de routage est-elle utilisée ? Le décrire en quelques mots.
(c) Quelles autres règles de routage ce protocole utilise-t-il ?

(a) Les grands opérateurs suivent souvent la politique dite de la patate chaude (hot potato) qui consiste à router un paquet le plus rapidement possible sans la garder (d'où le terme). Le but étant de transmettre l'information le plus rapidement possible sans forcément chercher le chemin le plus court ou le plus optimisé. La raison d'être est d'éviter le congestion entre deux AS ; les packets n'empreintent pas tous le même chemin.
(b) Elle est utilisée par le protocole de routage BGP.
(c) Heu.. Mettez tous se que vous savez sur BGP..

1.1.37.

(a) Déterminez analytiquement l'expression de l'efficacité du protocole ALOHA discrétisé (slotted ALOHA) en fonction de la charge du réseau pour un grand nombre de stations actives. On supposera que chaque station émet dans un slot avec une probabilité p.
(b) Représentez l'efficacité graphiquement (avec définition des axes), et expliquez la forme de la courbe.
(c) La suppression des slots (Cf. ALOHA pur) améliore-t-elle les performances ? Pourquoi ?

(a) Si on suppose qu'on a N nœuds qui ont beaucoup de trames à envoyer. Chacun transmet sur un slot avec une certaine probabilité p. Un nœud à $p(1-p)^{N-1}$ chances d'envoyer un paquet parce qu'il faut qu'il envoie un paquet (p) *et* qu'aucun des $N-1$ autres nœuds n'envoient un paquet ($1-p$) en même temps. La probabilité que n'importe quel nœud envoient un paquet avec succès est de

$$Np(1-p)^{N-1}$$

Pour une efficacité maximale, il faut donc trouver un p tel que $Np(1-p)^{N-1}$ soit maximale. On trouve (en dérivant) que L'efficacité est maximale quand $p = \frac{1}{N}$. Si on imagine que N tend vers l'infini, on sait que \lim\limits_{n-\infty} \left(1+\left(\dfrac{G}{n}\right)\right)^{n} = e^{-G} Et donc pour $G = 1$ e^{-1} = \dfrac{1}{e} = 0.37 = 37\% \text{ d'efficacité}
(b) (Slide 5-28) ?
(c) Non, la probabilité de collision augmente. On obtient une efficacité de 18%.

1.1.38.

(a) Enoncez les différents types de matrice de commutation (« switch fabric ») rencontrées dans les routeurs, ainsi que leurs avantages/inconvénients respectifs.
(b) Expliquez la raison d'être et l'inconvénient potentiel d'une bufferisation au niveau des **ports** d'entrées.
(c) Expliquez la raison d'être d'une bufferisation au niveau des **ports** de sortie.

(a)

- **Commutation par action sur la mémoire** : Les plus simples, les premiers routeurs étaient de simples ordinateurs. Le switch entre les **ports** d'entrée et sortie étaient fait via le CPU. Lorsqu'un paquet arrive au **port** d'entrée, le processus de routing l'identifiera via une interruption.Il copiera ensuite les paquets arrivant du buffer d'entrée sur le processeur mémoire. Le processeur extrait ensuite l'adresse de destination, recherche la table appropriée et copie le paquet sur le buffer du **port** de sortie. Dans les routeurs modernes, la recherche de l'adresse de destination et le stockage du paquet dans la mémoire appropriée est exécuté par les cartes de ligne d'entrée des processeurs.
  Avantages : L'architecture logicielle est la manière la plus simple d'orienter les paquets ; on prend un paquet, on lit son port de sortie, et on l'écrit dessus.
  Inconvénients : Traite un seul paquet à la fois, lent. Les architectures matérielles sont plus efficaces, comme un bus ou un crossbar.
- **Commutation par bus** : Le **port** d'entrée transfert les paquets directement sur le **port** de sortie via un bus partagé sans l'intervention d'un processus de routage. Comme le bus est partagé, seul un paquet est transféré à la fois via le bus. Si le bus est occupé, le paquet arrivant doit attendre dans une file. La bande passante du routeur est limitée par le bus comme chaque paquet doit traverser le bus seul. Exemple : Bus switching CISCO-1900, 3-COM's care builder5.
  Avantages : C'est le délit du bus qui détermine la vitesse de commutation du routeur.
  Inconvénients : C'est qu'un seul paquet est transféré à la fois, du coup il y a un risque d'attente.
- **Commutation par réseau d'interconnexions** : Pour surmonter le problème de la bande passante d'un bus partagé, les commutateurs réseaux en croix sont utilisés. Les **port** entrées et sorties sont connectés par des bus

horizontaux et verticaux. Si nous avons `N` **ports** d'entrés et `N` **ports** de sorties, on a besoin de `2N` bus pour les connecter. Pour transférer un paquet du **port** d'entrée au **port** de sortie correspondant, le paquet traverse le bus horizontal jusqu'à une intersection avec un bus vertical qui le conduit à son **port** de destination. Si le vertical est libre, le paquet est transféré. Mais si le bus vertical est occupé à cause d'une autre entrée, la ligne doit transférer des paquets au même **port** de destination. Les paquets sont bloqués et font la file sur le même **port** d'entrée.
Avantages : Ce système peut s'implémenter en hardware, et c'est super rapide (60 Gbps). Améliore la limite de débit associé à un bus commun.
Inconvénients : Le seul problème est que les paquets ont des tailles différentes. Si les paquets avaient tous la même durée, tout aurait été synchrone, on change la matrice de routage à chaque paquet, toutes les connexions en même temps. Quand les paquets ont des tailles variables, le chef d'orchestre doit être plus compliqué. On peut également découper les paquets en petits blocs de taille fixe (complétés par des zéros par exemple). Comme ça, on retrouve le comportement synchrone. Vu de l'extérieur, cette fragmentation est invisible.

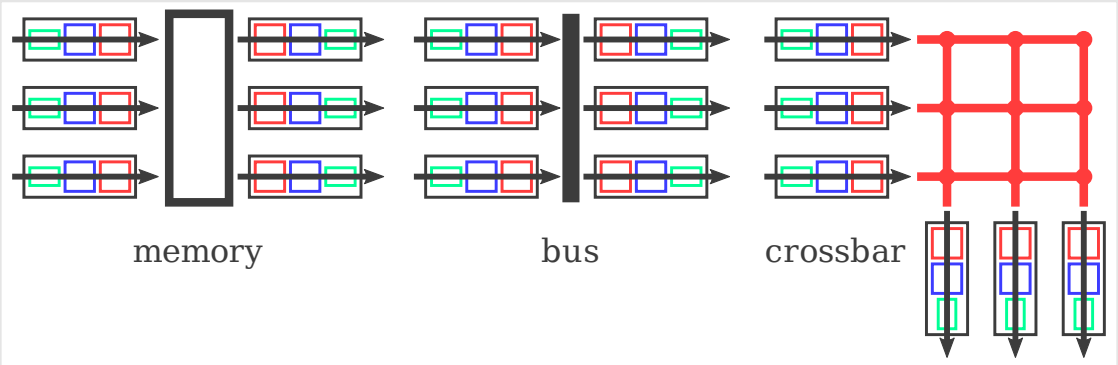


Fig. 14 - Matrice de commutation

(b) Le point le plus problématique est le fait que les routeurs doivent opérer à des vitesses élevées, impliquant des millions de consultations par seconde; on souhaite en effet que l'opération de consultation soit plus rapide que le temps qu'il faut au routeur pour recevoir un nouveau paquet. Dès que le **port** de sortie a été identifié, le paquet peut entrer dans la matrice de commutation. Cependant un paquet peut se voir refuser temporairement l'accès si elle est occupée à traiter des paquets qui ont été reçus sur d'autres liaisons d'entrée. Il est alors placé en file d'attente sur son **port**. Si la matrice de commutation n'est pas assez rapide, et du coup à mesure de l'augmentation de ces files, le risque de perte de paquets augmente.
(c) Si aucun ou peu de phénomène de mise en attente ne se produit à l'entrée, mais que tous les paquets venaient à être dirigés vers le même **port** de sortie, il serait très rapidement saturé. Un gestionnaire de paquets doit déterminer quel paquet de la file doit être transmis. Cette règle est généralement la règle **FIFO** (**First In First Out**).

1.1.39.

(a) Expliquez le principe du « Longest Prefix Match » lors de l'acheminement de paquets **IP**.
(b) Quel est son intérêt ?

(a) Dans une table d'acheminement d'un routeur, lorsqu'il y a plusieurs matchs, le routeur utilise la règle du Longest Prefix Match, il trouve le longest matching entry dans la table et envoie le paquet à l'interface lien associé.
Exemple d'IPv4 firwadubg tabke :
192.168.20.16/28
192.168.0.0/16
L'adresse 192.168.20.19 est donné, seulement ces 2 entrées sont matchés. Le routeur prendra le Longest Prefix Match, c-a-d 192.168.20.16/28 car le masque 28 (255.255.255.240) est plus grand que le masque 16 (255.255.0.0)
(b) Cela rends la route du packet beaucoup plus spécifique. Ca sert aussi détecter le cas où une table donne un sous-réseau. Car par exemple, la majorité des routeur ont souvent une entrée default, celui-ci est le seul avec un masque 0 (0.0.0.0) donc la dernière possibilité dans le Longest Prefix Match.

1.1.40.

(a) Expliquez le rôle et le principe général des codes détecteurs d'erreur.
(b) Pourquoi ne peuvent-ils être efficaces à 100% ?
(c) Donnez un exemple de code détecteur d'erreur plus élaboré que le bit de parité, et expliquez son principe.

(a) Il s'agit de détecter si il y a une erreur dans le paquet transmis. Pour cela, on introduit un ou plusieurs bits de contrôle

dans le paquet afin de savoir si le code a été changé ou pas.

(b) Car si il y a un seul bit de parité, on ne peut détecter que des erreurs impaires. Il faut mettre plusieurs bits de parités mais cela limite quand même toutefois le nombre d'erreurs détectables. De plus, rien ne dit qu'une erreur ne peut pas se produire sur un de ces bits.

(c)
- **Bit de parité** : Soit D des informations à transmettre de d bits. L'expéditeur ajoute aux données un unique bit tel que la somme des d + 1 bits soit paire (ou impaire selon ce qui est convenu par le modèle). Cette méthode est un peu faible, car si un nombre pair d'erreurs s'est glissé dans les données, elles ne seront pas détectées ; elle ne peut donc être appliquée que sur des systèmes pour lesquels les erreurs n'arrivent pas en rafale et dont la probabilité est extrêmement faible.
- **Checksum** : les codes **CRC** (**Code de Redondance cyclique**) sont une technique de détection d'erreur très fréquente et opère de la manière suivante : Lors de l'envoi, l'émetteur ajoute un nombre sur 16 bits à la fin des données (D), de manière à ce que ces données D et ce nombre CRC forment un nombre exactement divisible par G (modulo 2). En pratique l'émetteur divise D par G et met dans CRC le reste de cette division. Le destinataire, connaissant G, n'a qu'a diviser le paquet par G, s'il n'obtient pas zéro c'est qu'il y a eu des erreurs. Le point cruciale est le choix de G(x), puisqu'il ne faut pas que les données entachées d'erreur soient divisible par lui. Plus le polynôme est grand, plus on détece d'erreurs, il faut donc choisir entre la taille et la performance.

1.1.41.

(a) Définissez les différents types de « Resource Records (RR)» utilisés par le protocole **DNS** et expliquez leur rôle.

(b) Donnez le scénario d'échange de messages **DNS**, par la méthode itérative, permettant à un client de trouver l'adresse **IP** d'un serveur web dont l'URL est **www.company.com**, *en indiquant les RR présents dans ces messages*. On supposera que les caches **DNS** sont vides.

(a) Les enregistrements sur **DNS** sont sous le format : (*name*, *value*, *type*, *ttl*). On a plusieurs types possibles, entre autre :
- **A** : *name* est hostname et *value* est l'**IP** de l'hostname. **A** fournit le mapping standard **hostname-to-IP** (**A IPv4**, **AAAA IPv6**) ;
- **NS** : *name* est un domaine name is domain (exemple: google.be) et *value* est le hostname d'un serveur **DNS** autoritaire qui connait l'adresse **IP** du domaine. Il permet à un client de connaître le serveur à contacter pour ce domaine ;
- **CNAME** : *name* est un alias pour le hostname présent dans *value* ;
- **MX** : nom du serveur mail associé à *name*.

(b) Le scénario de la méthode itérative est :
- Le client émet une requête **www.company.com in A** à son serveur **DNS** local.
- Le serveur local, n'ayant pas l'adresse requise en cache, contacte un des root servers (défini dans sa configuration), avec la même requête.
- Le serveur root contacté lui renvoie alors le nom et l'adresse authoritaire (champs **NS** et **A** du serveur principal pour le **TLD .com**)
- Le serveur local réémet à nouveau la même requête vers le serveur du **TLD .com**. Ce serveur **TLD** renvoie lui aussi les champs **NS** et **A** pour le serveur faisant authorité sur le domaine **www.company.com** (par exemple *ns110.ovh.net*).
- Le serveur local contacte alors le serveur authoritaire, qui lui renvoie la zone pour le domaine **company.com**, qui contient un champ **CNAME** pour **www.company.com**. Si ce champ pointe vers le domaine contenu dans l'enregistrement **A** de la zone, la recherche s'arrête, le client a obtenu l'ip désirée.
- Sinon, tant qu'un enregistrement **A** n'a pas été trouvé, le serveur recommence les mêmes étapes à partir du domaine obtenu dans le champ **CNAME**.

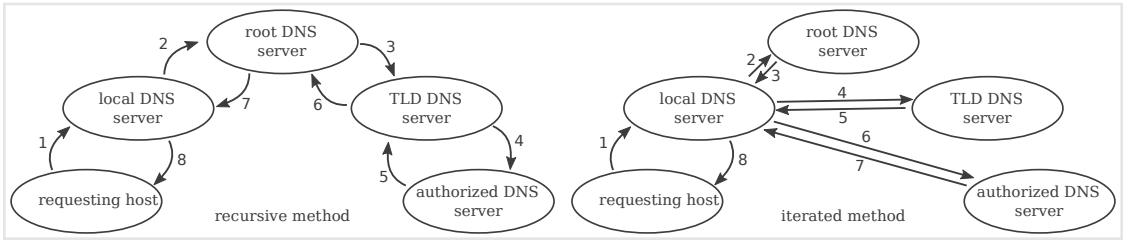


Fig. 15 - DNS name resolution (recursive and iterated method)

1.1.42. Vous créez votre entreprise « MeMyself&I » et vous obtenez le nom de domaine « memyselfandi.com ». Vous souhaitez déployer votre propre serveur **DNS** pour ce domaine (dns.memyselfandi.com, 111.111.111.111), ainsi qu'un serveur Web www.memyselfandi.com, 111.111.111.112).

(a) Quelles informations doivent être ajoutées dans la hiérarchie **DNS** et à quel niveau ? Soyez précis.

(b) Donnez un scénario typique d'échange de messages **DNS** permettant à un client de trouver l'adresse **IP** de votre serveur web, en précisant bien les éléments importants des messages **DNS**. On supposera que les caches **DNS** sont vides.

(a) Pour rendre le serveur web www.memyselfandi.com accessible, on indique d'abord 2 informations dans le serveur DNS TLD (Top-Level Domain) (.com dans ce cas) :
- Le nom du serveur DNS privé qui fait autorité pour notre serveur Web, ce qui sera décrit par le RR suivant : (memyselfandi.com, dns.memyselfandi.com, NS)
- L'adresse de ce même serveur autoritaire : (dns.memyselfandi.com, 111.111.111.111, A)

Ensuite il faudra rajouter les informations appropriées dans le serveur DNS privé:
- L'adresse du serveur web : (www.memyselfandi.com, 111.111.111.112, A)
- Un RR de type MX pour nom de domaine memyselfandi.com
- (Si www.memyselfandi.com est un alias, un RR de type CNAME pour connaitre son vrai hostname)

(b)

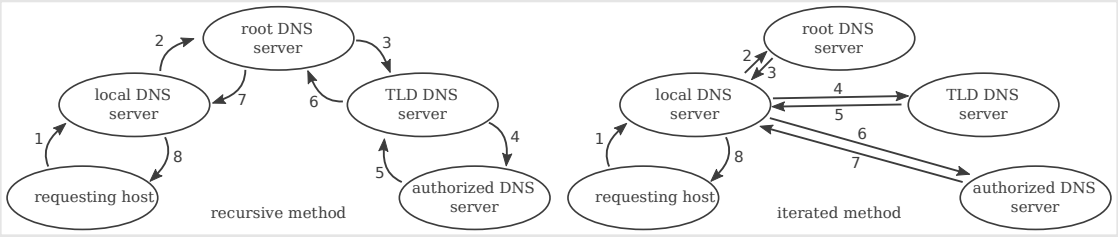


Fig. 16 - DNS name resolution (recursive and iterated method)

1.1.43.

(a) Pourquoi la couche de transport (**UDP** et **TCP**) comporte-t-elle une fonction de **démultiplexage** ?
(b) Décrivez les techniques de **démultiplexage** effectuées par **UDP** et **TCP** en mettant bien en évidence leurs différences ?

(a) Toutes les couches de transport effectuent un rôle de **démultiplexage**. C'est la couche de transport qui oriente les données entrantes vers les bons sockets. En effet, les couches de réseau et en-dessous ne gèrent que des paquets, qui contiennent des données et qui passent d'une machine à l'autre. C'est la couche de transport, sur une machine, qui reçoit les paquets et qui les redirige vers le bon processus. De même, quand des processus écrivent sur le réseau, c'est la couche de transport qui les multiplexe et les écrit tous sur le même réseau.
(b) Un **UDP** et **TCP** commence dans les deux cas par un numéro de port source (2 octets) puis un port de destination (2 octets aussi). C'est cela qui permet d'effectuer le **démultiplexage**. Les adresses **IP** sont inutiles car la couche de réseau s'est déjà occupée d'acheminer le paquet vers la bonne machine, on est donc déjà sur la bonne machine. De plus, les adresses **IP** source et destination sont contenues dans le paquet (couche réseau) et ne doivent pas être répétés dans le segment **TCP** ou **UDP**.
- Pour **UDP**, un socket cible est choisi en regardant l'**IP** et le port de destination d'un segment. La source n'est pas regardée. Plusieurs segments de plusieurs sources sont donc envoyés sur le même socket de destination.
- **TCP** est un peu plus compliqué qu'**UDP** pour le **démultiplexage**. En effet, avec **UDP**, l'application fournit les adresses **IP** et numéros de port quand nécessaire. De plus, avec **UDP**, les paquets de toutes les origines sont envoyés sur le même socket de destination. Avec **TCP**, il y a une connexion par origine. Il faut donc regarder les **IP** et ports source et destination de chaque segment pour trouver le socket qui doit le recevoir. Ce **démultiplexage** par la source est ce qui permet à **TCP** d'autoriser plusieurs sockets par port. Si 3 clients sont connectés à un serveur web, les 3 clients vont envoyer leurs données à l'adresse **IP** du serveur, port 80. C'est donc la source qui permet de différencier les connexions utilisées.

1.1.44. Expliquez le principe de NAT et la structure d'une table NAT.

La NAT vient résoudre deux problème problème majeurs :
- Tout d'abord la pénurie d'adresse IP sur Internet, résolu par la mise en place de plages d'adresses IP pour une utilisation privée
- Et du coup le second qui est de pouvoir accéder à Internet en utilisant ces adresses IP privées.

En fait, si on reprends le principe de la couche 3, il y a bien un facteur limitant au nombre de machines possibles sur Internet, c'est le nombre d'adresses IP disponibles. Rappel: Une adresse IP est codée sur 4 octets, soit 32 bits. Elle peut donc prendre 2^32 valeurs, soit environ 4 294 967 296. Ca peut paraitre beaucoup mais avec le nombre d'appareil connectés ne cesse d'augmenter et on approche très rapidement au quota. La transition de IPv4 à IPv6 est en place mais il faut une solution temporaire pour ne pas priver les machines actuels du réseau du jour au lendemain.

Vu que certains réseaux étaient privés et que les machines sur ces réseaux n'avaient pas besoin d'être jointes depuis Internet (elles étaient de simples clients, mais pas des serveurs), il n'était pas nécessaire de leur fournir une adresse IP publique à chacune d'entre elles. Ainsi, on s'est dit qu'on pourrait donner des adresses IP privées à ces machines. Cette plage d'adresses privée n'est donc pas utilisée sur Internet, elle est réservée pour tous les réseaux du monde entier qui n'ont pas besoin d'être joints depuis Internet (par exemple 192.168.x.x ou 10.x.x.x ou 172.16.x.x).

Donc, la NAT associe n adresses privées à une seule adresse publique. Ainsi, on peut connecter n machines en n'utilisant qu'une seule adresse publique. On économise donc des adresses. Lorsqu'un paquet est envoyé vers l'extérieur, il passe par le dispositif **NAT** qui converti l'adresse **IP** interne en adresse **IP** officielle du routeur. Le dispositif **NAT** et un pare-feu sont souvent combinés dans le même équipement, offrant ainsi une certaine sécurité en contrôlant précisément ce qui entre sur le réseau et en sort.

Structure d'une table NAT							
réseau privé				**box**			
IP SRC	**IP DST**	**PORT SRC**	**PORT DST**	**IP SRC**	**IP DST**	**PORT SRC**	**PORT DST**
192.168.0.1	217.70.184.38	10277	80	82.238.22.47	217.70.184.38	2356	80
192.168.0.2	217.70.184.38	10277	80	82.238.22.47	217.70.184.38	2357	80

On voit maintenant que lorsqu'un paquet reviendra avec comme port destination 2356, la box saura qu'il s'agit d'un paquet à destination de 192.168.0.1 et que, lorsqu'il reviendra avec comme port destination 2357, ce sera pour la machine 192.168.0.2. Vu que c'est la box elle-même qui choisit le port source, on est sûrs qu'on n'aura jamais deux fois le même port.

1.1.45. Quand des flux **TCP** et **UDP** partagent un même lien congestionné, comment réagissent ces deux types de flux et quelles en sont les conséquences ?

- **TCP** peut distinguer 2 types de congestions:
  - soit il reçoit 3 **ACK**s consécutifs pour le même numéro de séquence (donc un des paquets intermédiaire a été perdus, mais les suivants sont passés: **faible congestion**)
  - soit un **ACK** n'arrive pas dans le temps imparti (timeout, beaucoup de paquets perdus: **congestion sévère**).

  Au démarage de la transmission, **TCP** envoie les données avec une fenêtre de taille 1 **MSS** (**Maximum Segment Size**), correspondant au nombre de paquets qui peuvent être en parcourt simultanément. La taille de la fenêtre est doublée à chaque itération (en incrémentant la taille à chaque **ACK** reçu), de sorte qu'elle a une **croissance exponentielle**. S'il détecte une **faible congestion**, il divise la taille de la fenêtre par deux et change de mode pour incrémenter la taille de la fenêtre à chaque itération (+1 pour chaque fenêtre totalement envoyée) pour adopter une **croissance linéaire**. Il approche ainsi **dichotomiquement** la taille moyenne de fenêtre optimale (càd le nombre de paquets en parcourt, et donc la vitesse d'envoi). S'il détecte une **congestion sévère**, il réduit la taille de fenêtre à 1 et recommence en mode de **croissance exponentiel**. Il peut éventuellement repasser en mode de **croissance linéaire** lorsqu'il a atteint la moitié de la taille de fenêtre qui a provoqué un timeout (puisque doubler sa taille provoquera probablement de nouveau un timeout).
- **UDP** quant à lui est un processus de transport extrêmement simple et léger, doté d'un modèle de service minimum. Il ne nécessite aucun échange préalable entre deux processus, de plus les échanges de données sont non-fiables, ce qui signifie que lorsqu'un processus expédie un message, il ne se soucie pas de son arrivée.

1.1.46.

(a) Expliquez la technique de fragmentation de paquets IP en justifiant le rôle des différents champs pertinents de l'en-tête des paquets.
(b) Quelle méthode est utilisée pour éviter la fragmentation des paquets IP ?

(a) ?
(b) ?

1.1.47. Expliquez comment un routeur construit les entrées de sa table d'acheminement pour les préfixes **IP** extérieurs à son domaine.

Il utilise des masques, Interface et passerelle afin de diriger correctement les paquets. Les tables contiennent les adresses de destination, le manque, les adresses des passerelles (routeurs intermédiaires) permettant d'atteindre la destination, l'adresse de la carte réseau (interface) par laquelle le paquet doit sortir du routeur.

1.1.48. Considérez 3 réseaux **Ethernet** (`N_1` , `N_2` et `N_3`), un commutateur **Ethernet** (`C`) et un routeur (`R`) interconnectés selon une topologie en ligne `N_1` - `C` - `N_2` - `R` - `N_3` . Une station `H_A` (d'adresse `IP_A`) est attachée au réseau `N_1` (par l'adresse `MAC_A`) et une station `H_B` (d'adresse `IP_B`) est attachée au réseau `N_3` (par l'adresse `MAC_B`). `C` a deux adresses `MAC` : MAC_{11} sur `N_1` et MAC_{12} sur `N_2` . `R` a deux adresses `MAC` et deux adresses `IP` : MAC_{22} et `IP_2` sur `N_2` et MAC_{23} et `IP_3` sur `N_3` .

(a) Dessinez la configuration. `H_A` envoie un paquet `IP` à `H_B` . Si l'on suppose que les correspondances entre adresses `IP` et `MAC` sont connues de tous, décrivez les trois trames qui circulent respectivement sur les réseaux `N_1` , `N_2` et `N_3` en vous limitant aux champs d'adresses des trames et aux champs d'adresses et de `TTL` (`T` ime `T` o `L` ive) du paquet `IP` contenu dans la trame. Justifiez.

(b) Par quel protocole les correspondances entre adresses `IP` et `MAC` ont-elles été découvertes ? Décrivez les échanges de ce protocole qui réalisent les mises en correspondance nécessaires lorsque `H_A` envoie son paquet `IP` à `H_B` . Mentionnez toutes les adresses présentes dans les messages échangés.

(a) ?
(b) ?

1.1.49. Citez une fonction majeure de chacune des 5 couches de la pile de protocoles Internet.

Couche	Description	Noeud	Protocoles
Application	supporte les applications	Processus	HTTP, FTP
Transport	le transfert des données entre processus	Processus Socket	TCP, UDP
Réseau	routage des données d'une source à une destination	Hôtes	IP, protocoles de routages
Lien	transfert des données entre 2 noeuds voisins du réseau	Noeuds	Ethernet, WiFi
Physique	transfert des données via un cable	?	?

1.1.50.

(a) Pourquoi est-il plus difficile de fixer la durée du timer de retransmission de **TCP** que celle du timer de retransmission d'un protocole de liaison de donnée ?

(b) Comment fixe-t-on la durée du timer de retransmission de **TCP** ?

(a) **TCP** n'utilise qu'un seul timer, comme le Go-Back-N. Le problème est que **TCP** doit à la fois s'adapter aux réseaux locaux (ou les timeouts peuvent être très cours) qu'à l'internet (ou le chemin utilisé par un paquet et peut-être très long et donc les délais aussi).

(b) Pour estimer le **RTT**, on va avoir un échantillon **RTT** qui va stocker le temps entre le moment ou le byte est transmis et le moment où on reçoit l'**ACK**. Comme on va avoir une grosse fluctuation des valeurs avec cette technique, il faut essayer de « lisser » ces valeurs. On va donc avoir besoin de la moyenne et la variance.

$$estimation = (1 - a) * estimationPrécédente + a * nouvelEchantillon$$

avec $a = 0.125$

$$marge\ (\text{de sureté}) = (1 - b) * margePrécédente + b * |echantillon - estimation|$$

avec $b = 0.25$

$$Timeout = estimation + 4marge$$

' 4 ' parce qu'avec cette valeur, seulement 1% des paquets étaient renvoyés trop tôt.

1.1.51. Expliquez la raison d'être des protocoles **DHCP** et NAT, et expliquez leur fonctionnement à l'aide de scénarios typiques.

De nos jours on utilise surtout des adresses **IP** dynamique. Le **DHCP** (**Dynamic Host Configuration Protocol**) s'occupe d'attribuer une adresse **IP** à un ordinateur qui se connecte. **DHCP** est un protocole de la couche application qui utilise **UDP** et **IP**.Quand un ordinateur se connecte à un subnet, il broadcast pour voir si un serveur **DHCP** se trouve dans ce subnet. Il lui répond en envoyant une adresse **IP**, en broadcast vu que le client n'a pas encore d'adresse. Le client répond pour dire que l'adresse l'intéresse, toujours en broadcast au cas où il y aurait d'autre serveur **DHCP** dans la zone, afin de les prévenir. Le **DHCP** envoie ensuite un **ACK** pour lui dire que l'adresse a bien été réservée. **NAT** (**Network Access Translation**), de nos jours

on a presque plus d'adresses **IP** disponibles. Les NATs permettent de se servir de l'adresse **IP** d'une seule machine pour envoyer des données à tout le réseau. Il bidouille avec les numéros de **port**, le routeur fait une table avec d'un côté les adresses **IP** privées et l'adresse **IP** extérieure ainsi que le **port**.

1.1.52.

(a) Expliquez comment les commutateurs **Ethernet** apprennent où se trouvent les stations et par quel type d'adresse ils les identifient.
(b) Comment les pannes de stations ou leur mobilité sont-elles prises en compte ?
(c) En quelques mots, quelle contrainte topologique doit être respectée pour que cet apprentissage fonctionne, et comment la réalise-t-on ?

(a) Le protocole Ethernet utilise les adresses MAC. Pour connaître l'adresse, le commutateur va lancer une trame sur le broadcast (ff:ff:ff:ff:ff:ff) et mettre à jour sa table ARP.
(b) Les pannes sont prises en compte à l'aide d'un TTL, quand on n'a pas de nouvelles d'une machine depuis un moment, elle sort de la table. Les données sont mises à jour avec les data les plus récentes.
(c) La topologie était historiquement en bus, mais ça pose des problèmes si le câble est abîmé à un endroit; de plus il y avait des risques de collision. Actuellement, on utilise une structure en étoile avec un switch au centre.

1.1.53. Citez et définissez les différentes sources de délai que subit un paquet dans un réseau datagramme.

$$d(noeud) = d(processing) + d(queueing) + d(transmission) + d(propagation)$$

1.1.54.

(a) Décrivez sommairement le fonctionnement du système **DNS**.
(b) Comparez les deux modes de fonctionnement du protocole (avantages et inconvénients).

(a) Le système **DNS** (**Domain Name System**) est le service d'annuaire d'Internet. Lorsqu'on recherche une adresse **URL** tel que `http://google.com` ou `http://facebook.com`, le système d'**DNS** traduit l'adresse **URL** en une adresse **IP** (entre `0.0.0.0` et `255.255.255.255` pour **IPv4**). Ce système est fréquemment utilisé par d'autres protocoles tels que **HTTP**, **SMTP** et **FTP** pour délivrer les adresses **IP** correspondant aux noms de serveurs demandés. Par exemple pour `http://ulb.be`, le système **DNS** formule une requête auprès d'un serveur de nom, à laquelle il reçoit une réponse concernant l'adresse **IP** correspondante et ensuite le navigateur établit une connexion **TCP** avec le processus serveur répondant à cette adresse.
(b)
1. Une version simplifiée du **DNS** consisterait en un serveur de noms unique contenant toutes les correspondances existantes. Ce système a l'air simple, mais impossible à mettre en oeuvre, pour causes : fragilité d'un site central unique, volume de trafic trop important, base de données centralisée trop éloignée de certains utilisateurs, problèmes de maintenance dus au volume énorme des données à stocker. **DNS** se doit donc d'être un système distribué.
2. **DNS** utilise un grand nombre de serveurs de noms, organisé de manière hiérarchique et distribué dans le monde entier. Il existe trois types de serveurs de noms: les **serveurs de noms locaux**, les **serveurs de nom racine** et les **serveurs de nom de source autorisée**.
Chaque fournisseur d'accès possède un **serveur de noms local**, vers lequel vont toutes les recherches **DNS** formulées au sein de ce réseau local. Un **serveur de noms local** est forcément proche du client. Lorsqu'un serveur local de noms n'est pas en mesure de répondre à une demande il se transforme en client **DNS** et interroge un **serveur de nom racine**, si celui-ci a une réponse il l'envoie au serveur de noms **DNS**, qui la transmet alors à l'auteur de la demande; si le **serveur de nom racine** ne peut lui non plus satisfaire la demande directement, il répond en donnant l'adresse **IP** d'un **serveur de nom de source autorisée** qui connaîtra certainement la correspondance recherchée. Tout serveur est enregistré auprès d'au moins deux **serveurs de noms de source autorisée**, en général il s'agit tout simplement du fournisseur d'accès.
Un **serveur de nom est qualifié** de source autorisée pour un serveur donné, s'il dispose en permanence d'archives **DNS** permettant d'établir la conversion pour ce serveur.

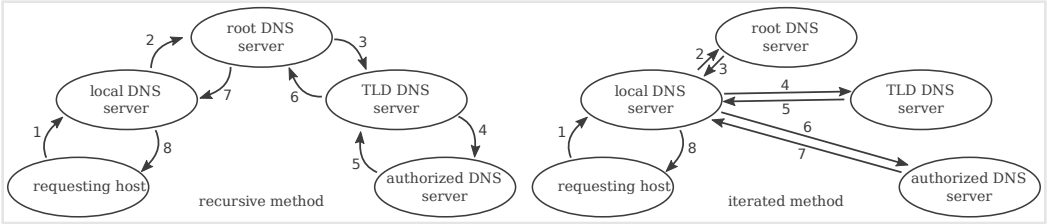


Fig. 17 - DNS name resolution (recursive and iterated method)

Toutes ces recherches que nous venons de voir étaient du type récursives, mais **DNS** autorise également des recherches itératives à n'importe quel moment du processus de recherche : si un serveur n'est pas en mesure de répondre favorablement à la demande, il renvoie directement l'adresse **IP** du prochain serveur de nom de la chaîne. En général, toutes les demandes d'une même recherche **DNS** sont récursives, mis à part celle émanant du serveur local de nom adressée au serveur racine qui est de nature itérative. Cette démarche est préférable, les serveurs racines traitant généralement de grands volumes de demandes.

1.1.55.

(a) Expliquer les principes de la programmation socket donnant accès aux services **TCP** et **UDP**.
(b) Quelles sont les différences importantes entre ces deux **API** ?
(c) Dans une entité de transport, comment les sockets **TCP** et **UDP** sont-ils identifiés ? Pourquoi ?

(a) Le socket est une prise associée à un port permettant l'accès aux couches réseaux. Cela permet d'envoyer et recevoir des données via des services comme **UDP** et **TCP**.
(b) **UDP** est un service non-connecté (non-fiable) qui envoie les données sans se socier de leur bonne réception par le destinataire. Tandis que **TCP** est un service orienté connexion qui permet de s'assurer que les messages atteignent leur destination. Ceci est fait au moyen d'une connexion établie préalablement entre le client et le serveur qui s'envoie des accusés de réception. Si un accusé de réception n'est pas reçu après l'envoi d'un message, le message est considéré comme perdu et renvoyé au destinataire.
(c) Le socket est identifié via le numérod e port sur lequel il se trouve. Il faut donc l'**IP** ainsi qu'un numéro de port pour envoyer un message. Cela permet à l'ordinateur de répartir le message qu'il reçoit au bon endroit.

1.1.56.

(a) Expliquez les circonstances dans lesquelles l'émetteur **TCP** peut recevoir trois doublons d'acquits venant du récepteur **TCP**.
(b) Décrivez deux actions importantes de l'émetteur **TCP** lorsque cela se produit et expliquez-en les raisons.

(a) Quand il reçoit 3 doublons du même **ACK** d'affilé, il va comprendre qu'il y a un segment qui n'a pas été reçu.
(b) L'émetteur ne doit donc pas attendre la fin du timeout pour renvoyer le segment.

1.1.57.

(a) Expliquez le principe général du contrôle de *flux* de **TCP**.
(b) Expliquez deux mécanismes associés ayant pour but de permettre à **TCP** de s'adapter aux spécificités des applications ou de se protéger vis-à-vis de celles-ci.

(a) Cela permet de réguler la vitesse sur le réseau afin d'éviter de surcharger le receveur. Pour se faire, l'expéditeur a une variable "receive window" pour qu'il sache combien de place disponible se trouve dans le buffer du receveur. Comme **TCP** ne peut pas surcharger les buffer : $LastByteRcvd - LastByteRead <= RcvBuffer$. Le receiver window (rwnd) se définit comme suit :

$$rwnd = RcvBuffer - (LastByteRcvd - LastByteRead)$$

(b) Nagle Algorithm : Quand les données viennent du socket un byte à la fois, on envoie le premier byte et on buffer le reste jusqu'à ce que le premier byte soit reçu. On envoie alors le reste par RTT en 1 fois. Silly Window Problem - Clark' solution : Le receveur envoi une mise à jour de la fenêtre si et seulement si le buffer est à moitié vide ou si un segment entier peut être reçu.

1.1.58. Combien d'adresses **IP** doit-on attribuer à un routeur ? Pourquoi ?

Une seule car cela représente le routeur au sein du réseau. L'adresse **IP** est unique et fait donc référence à un objet bien précis. (C'est pas plutôt une adresse par interface ?)

1.1.59. Sachant que la couche de transport est équipée de mécanismes (Cf. **TCP**) pour récupérer les erreurs de bout-en-bout, pourquoi la couche de liaison de données implémente-t-elle aussi toute une série de fonctions de ce type, comme la détection d'erreurs, voire même la retransmission de trames erronées dans certains cas.

TCP permet seulement de vérifier que le paquet soit bien reçu mais ne vérifie pas son contenu. Lors du transport, il pourrait y avoir une modification des données. La vérification du contenu est effectuée après afin de ne pas ralentir le débit.

1.1.60.

(a) Dans un réseau local composé de plusieurs segments **Ethernet** interconnectés par des commutateurs **Ethernet**, un ordinateur peut-il conserver son adresse **IP** si on le change de segment ? Pourquoi ?
(b) En est-il de même si les segments sont interconnectés par des routeurs ? Pourquoi ?
(c) Pourquoi est-il plus intéressant d'interconnecter des segments **Ethernet** par des commutateurs **Ethernet** plutôt que par des hubs ?

(a) Oui car ils sont identifiés par leurs adresses **MAC**, l'adresse **IP** n'est pas très importante.
(b) Non, car ici, le protocole **DHCP** attribuera une nouvelle adresse à l'ordinateur.
(c) Car on ne peut mettre que maximum 4 hubs interconnectés par des cables de 100m maximum alors que les commutateurs n'ont pas de limites.

1.1.61.

(a) Quel mécanisme est utilisé par un serveur Web pour conserver de l'état relatif aux usagers ? Expliquez le principe en l'illustrant sur un scénario.
(b) Expliquer le fonctionnement de HTTP avec **proxy**-cache à partir d'un scénario impliquant le client, le serveur et le **proxy**. Expliquez le gain d'efficacité lorsque l'objet est en cache.

(a) Les **cookies** sont définis par le **protocole de communication HTTP** comme étant une suite d'informations envoyée par un **serveur HTTP** à un **client HTTP**, que ce dernier renvoie à ce même **serveur HTTP** lorsqu'il lui pose la question. Le **cookie** est l'équivalent d'un petit fichier texte stocké sur le terminal de l'internaute. Existants depuis plus de 20 ans, ils permettent aux développeurs de sites internet de conserver des données utilisateur afin de faciliter leur navigation et de permettre certaines fonctionnalités. Les cookies ont toujours été plus ou moins controversés car contenant des informations personnelles résiduelles pouvant potentiellement être exploitées par des tiers. Il est envoyé en tant qu'en-tête **HTTP** par le serveur web au navigateur web qui le renvoie inchangé à chaque fois qu'il accède au serveur. Un **cookie** peut être utilisé pour une authentification, une session (maintenance d'état), et pour stocker une information spécifique sur l'utilisateur, comme les préférences d'un site ou le contenu d'un panier d'achat électronique. Par exemple de scénario, lorsqu'on se connecte sur un site, un **cookie** peut être créé pour se connecter directement sur le compte sans devoir retaper le login/password. Ou encore garder en mémoire le panier d'achat sur un site e-commerce pour un achat ultérieur. Ce n'est donc principalement qu'une question de facilité pour l'utilisateur.
(b) Quand on utilise un **proxy**, c'est avec le **proxy** qu'on établit la connexion **TCP**, pas avec le vrai serveur. Après, le **proxy** se sert de l'info dans le header de la requête http pour rediriger la requête. Le **proxy** permet aussi de retenir des infos en cache. Si le serveur est down et que l'inforomation est en cache sur le **proxy**, on peut encore obtenir l'inforomation. Les **proxy**s permettent également de réduire les délais de réponse et le trafic vers le site, car s'il a déjà l'inforomation, on ne doit pas la demander au serveur. Il permet aussi de filtrer les connexions et de répartir les charges. On gagne donc en efficacité quand l'objet est en cache.

1.1.62.

(a) Dans un protocole de transport, si l'on numérote les segments modulo 2, montrez par un contre exemple qu'il est également nécessaire de numéroter les acquits pour assurer la fiabilité du transfert.
(b) Dans quelle(s) situation(s) le protocole à bit alterné est-il quasiment aussi efficace qu'un protocole à grande fenêtre glissante ? Expliquez.

(a) Un paquet sur 2 contiendra donc la valeur 0 et l'autre, la valeur 1. Grâce à ça, si l'**ACK** est corrompu, le receveur va s'apercevoir que le nouveau paquet reçu est celui qu'il avait déjà reçu et va donc le jeter. Il va ensuite envoyer un **ACK** à nouveau pour passer au paquet suivant.
(b) Lorsqu'il y a qu'un seul paquet à envoyer. Sinon le stop & wait fait perdre beaucoup de temps et empêche la ligne d'avoir un débit correct.

1.1.63.

(a) Dans **TCP**, comment fixe-t-on les numéros des premiers segments transmis dans chaque sens d'une connexion ?
(b) Si l'on attribuait systématiquement la valeur 0 (par exemple) à ces premiers numéros, quel serait le risque et comment pourrait-on l'éviter en conservant toutefois cette numérotation ? Quel serait l'inconvénient ?

(a) **TCP** utilise le **3-way handshake** et comme son nom l'indique, il se déroule en trois étapes:

- **SYN** (**synchronized**) : Le client qui désire établir une connexion avec un serveur va envoyer un premier paquet **SYN** au serveur. Le numéro de segment de ce paquet est un nombre aléatoire A.
- **SYN-ACK** (**synchronize-acknowledge**) : Le serveur va répondre au client à l'aide d'un paquet **SYN-ACK**. Le numéro du **ACK** est égal au numéro de segment du paquet précédent (**SYN**) incrémenté de un (A + 1) tandis que le numéro de segment du paquet **SYN-ACK** est un nombre aléatoire B.
- **ACK** (**acknowledge**): Pour terminer, le client va envoyer un paquet **ACK** au serveur qui va servir d'accusé de réception. Le numéro de segment de ce paquet est défini selon la valeur de l'acquittement reçu précédemment p.e. A + 1 et le numéro du **ACK** est égal au numéro de segment du paquet précédent (**SYN-ACK**) incrémenté de un (B + 1).

Une fois le **3-way handshake** effectué, le client et le serveur ont reçu un acquittement de la connexion. Les étapes 1 et 2 définissent le numéro de segment pour la communication du client au serveur et les étapes 2 et 3 définissent le numéro de segment pour la communication dans l'autre sens. Une communication full-duplex est maintenant établie entre le client et le serveur.

(b) En effet, on repère une connexion avec les 2 adresses **IP** et les 2 numéros de **port**. Si cette connexion arrive à son terme et que juste après avoir fermé cette connexion, on rétabli une connexion avec les même **IP** et **ports** comment peut-on distinguer les 2 connexions (il pourrait y avoir un vieux paquet de la première connexion qui s'était baladé sur le réseau et qui est arrivé beaucoup plus tard)?

1.2. Pratique

Efficacité

$$\begin{aligned}\eta &= \frac{\text{débit utile}}{\text{débit brut}} \\ &= \frac{\text{temps de transmission des données}}{\text{temps total d'un échange}} \\ &= \frac{\text{nombre de bits de données transmis}}{\text{nombre total de bits transmis } + \left(V \times T_{\text{délais}}\right)}\end{aligned}$$

Temps de transmission `T_t`

$\begin{array}{rcl} T_t & = & \dfrac{\text{Nbre N de bits transmis}}{\text{débit de la source (en bps)}}\\ & = & \text{Temps entre l'envoi des bits 1 et N } \end{array}$

Temps de transmission `T_p`

$$T_p = \frac{\text{Distance (en m)}}{\text{Vitesse de propagation (en m/s)}}$$

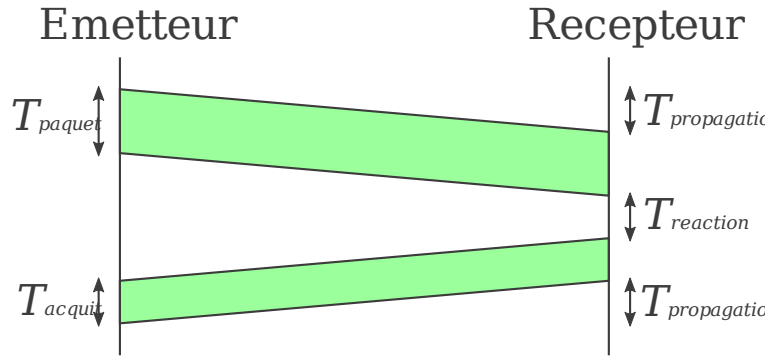
Ligne terrestre

$$\text{Vitesse de propagation} = 2 \times 10^8 \text{m/s}$$

Ligne satellite

$$\text{Vitesse de propagation} = 250 - 300\text{ms}$$

Simple envoi



Transformations

x octets	=	$8x$ bits
x Kbps	=	$x10^3$ bps
x ms	=	$x10^{-3}$ s

Débit utile

$$\text{Débit utile} = \frac{N}{T_{paquet} + T_{acquit} + 2T_p}$$

Temps paquet

$$T_{paquet} = \frac{N + octets_{en-tête}}{canal}$$

Temps acquit

$$T_{acquit} = \frac{octets_{en-tête}}{canal}$$

1.2.1. Soit A et B deux machines située sur un même réseaux de 1Mbps (10^6). A envoie des données à B (la taille maximale des segments a été négociée à 512 bytes de données), les en-têtes **TCP** et **IP** forment un total de 40 bytes. B envoie un acquit (segment **TCP** vide) à A à la réception d'un segment, après le temps de traitement de celui-ci fixé à 100ms. La station B a une fenêtre de réception de 36 MSS. Le seuil initial pour l'algorithme de slow-start est de 16 MSS. Le temps de propagation

est négligeable.

(a) Que vaut le RTT ?
(b) Combien de temps faudra-t-il à A pour arriver à une fenêtre de congestion de taille maximale ?

(a) Étape par étape :

1. Temps de A vers B : $\frac{(512+40)8}{10^6} = 0.004416\,\text{s}$
2. Temps de traitement de A : $0.1\,\text{s}$
3. Temps de B vers A : $\frac{(40)8}{10^6} = 0.00032\,\text{s}$

Donc 1 RTT $= 0.00416 + 0.1 + 0.00032 = 0.10448\text{s}$

(b)

RTT	ACK	CWND (MSS)	Envoyé (MSS)
0	0	1	1
1	1	2	2
2	2	4	4
3	4	8	8
4	8	16	16
5	16	17	17
...	...	...	...
24	35	36	36

Donc 24 RTT $= 24 * 0.10448 = 2,50752\text{s}$

1.2.2. Soit A et B deux machines située sur un même réseaux de 1Mbps (10^6). A envoie des données à B (la taille maximale des segments à été négociée à 256 bytes de données), les en-têtes **TCP** et **IP** forment un total de 40 bytes. B envoi un acquit (segment **TCP** vide) à A à la réception d'un segment, après le temps de traitement de celui-ci fixé à 20ms. La station B a une fenêtre de réception de 12 MSS. Le seuil initial pour l'algorithme de slow-start est de 8 MSS.

(a) Que vaut le RTT ?
(b) Combien de temps faudra-t-il à A pour arriver à une fenêtre de congestion de taille maximale ?

(a) Étape par étape :

1. Temps de A vers B : $\frac{(256+40)8}{10^6} = 0.002368\,\text{s}$
2. Temps de traitement de A : $0.02\,\text{s}$
3. Temps de B vers A : $\frac{(40)8}{10^6} = 0.00032\,\text{s}$

Donc 1 RTT $= 0.002368 + 0.1 + 0.00032 = 0.102688\text{s}$

(b)

RTT	ACK	CWND (MSS)	Envoyé (MSS)
0	0	1	1
1	1	2	2
2	2	4	4
3	4	8	8
4	16	9	9
...	...	...	...
7	11	12	12

Donc 7 RTT $= 7 * 0.102688 = 0.718816\text{s}$

1.2.3. Un émetteur envoie des paquets à un récepteur à l'aide d'un protocole de transport *stop and wait*. Le débit brut du réseau est de 1 Mbps. Chacun de ces paquets contient 1000 bits de données utiles. L'émetteur retransmet un paquet s'il ne reçoit pas d'acquit (24 bytes) avant l'expiration de son timer, fixé à 1 s. Sachant qu'en moyenne un paquet (données ou acquit) sur 100 n'arrive pas à bon port, calculez l'efficacité moyenne de la connexion. On supposera négligeable les délais de

propagations et les overheads introduits par l'encapsulation des données.

Débit utile $= \frac{bits}{T_t + T_{ACK} + T_p} = \frac{1000}{\frac{1000}{10.10^6} + \frac{24.8}{10.10^6} + 0}$

Efficacité $= \frac{\text{débit utile}}{\text{vitesse}} = \frac{1000}{1000 + 24.8} = 0.839 = 83,9\%$

Timer?

1.2.4. Deux entités A et B ont établi une connexion **TCP** passant par deux routeurs `R` et `S`. Les liaisons $A \leftrightarrow R$, $R \leftrightarrow S$ et $S \leftrightarrow B$ ont un débit de respectivement 10 Mbps, 1 Mbps, et 1 Mbps. Chacune de ces liaisons a un temps de propagation de 10 ms. `A` souhaite envoyer des données à `B` le plus rapidement possible. La fenêtre de réception de `B` est de 18 MSS, le MSS ayant été négocié à 10 Kb, en-tête compris. Le seuil de l'algorithme de *slow-start* est initialement fixé à 12 MSS. A chaque réception d'un segment, `B` répond par un acquit de 24 octets, en-tête compris. Un timer de retransmission de 1 s est enclenché à chaque début d'envoi d'une rafale. Combien de temps faut-il à `A` pour arrive à un fenetre de congestion de taille maximale, sachant que la troisième rafale sera entièrement perdue et qu'il n'y aura pas d'autres pertes ?

$T_{MSS|10Mbps} = \frac{10000}{10.10^7} = 0.0001$ s

$T_{MSS|1Mbps} = \frac{10000}{10.10^6} = 0.001$ s

$T_{propagation} = 10$ ms$= 0.01$ s

$T_{ACK|10Mbps} = \frac{24.8}{10.10^7} = 0.00000192$s

$T_{ACK|1Mbps} = \frac{24.8}{10.10^6} = 0.0000192$s

$$\begin{aligned} RTT &= T_{MSS|10Mbps} + 2T_{MSS|1MBps} + 2T_{propagation} + T_{ACK} + 2T_{ACK|1Mbps} \\ &= 0.0001 + 2 * 0.001 + 2 * 0.01 + 0.00000192 + 2 * 0.0000192 \\ &= 0.02214032s \end{aligned}$$

$\frac{rwnd(10000)}{RTT} = \frac{12(10000)}{0.02214032} = 5419975.863 = 5419975$bps

$cwnd = \frac{1000000 * 0.02214032}{10000} = 2.214032 = 2$

RTT	ACK	CWND (MSS)	Envoyé (MSS)	Seuil (MSS)
0	-	1	1	12
1	1	2	2	12
2	2	4	4	12
3	-	1	1	6
4	1	2	2	6
5	2	4	4	6
6	4	6	6	6
7	6	7	7	6
8	7	8	8	6
...	...	...	...	6
18	17	18	18	6

Il faut donc Timeout + 18 RTT : $1 + 18 * 0.02214032 = 1.39852576s$

1.2.5. Trois stations `S_1`, `S_2` et `S_3` se partagent un segment de réseau de type 802.3 (CSMA/CD, 10Mbps). La première station désire émettre une trame de 1000 bits alors que les deux autres stations souhaitent émettre chacune deux trames de 1000 bits. La durée d'un slot de contention a été fixée à $2\tau = 2.10^{-6}$ s. Lorsque plusieurs stations veulent accéder au réseau, on supposera que la probabilité de retransmission dans un slot est constante et égale à $p = \frac{1}{2}$. Calculez :

(a) la durée moyenne d'envoi des 2 premières trames,
(b) la durée moyenne d'envoi des 5 trames.

Sachant que les stations commencent à émettre leur première trame en même temps.

(a) La probabilité qu'une des 3 stations acquiert le canal dans un slot(libre) est :

$A_3 = kp(1-p)^{k-1} = 3\frac{1}{2}(1-\frac{1}{2})^{3-1} = 0.375$

La probabilité qu'une des 2 stations acquiert le canal dans un slot(libre) est :

$A_2 = kp(1-p)^{k-1} = 2\frac{1}{2}(1-\frac{1}{2})^{2-1} = 0.5$

$C_3 = \frac{2\tau}{A_3} = \frac{2 * 10^{-6}}{0.375} = 0.00000533$

$C_2 = \frac{2\tau}{A_2} = \frac{2 * 10^{-6}}{0.5} = 0.000004$

$T = \frac{1000}{10 * 10^6} = 0.0001s$

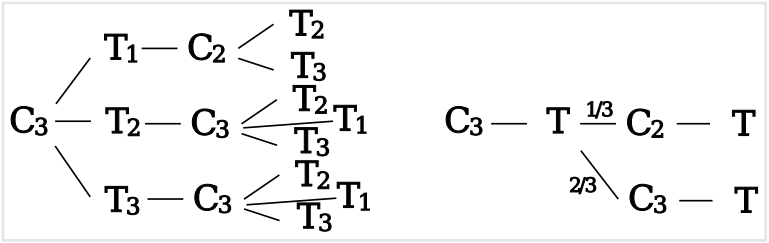


Fig. 18 - CSMA

$\frac{5}{3}C_3 + \frac{1}{3}C_2 + 2T = 0.0002102$

(b)

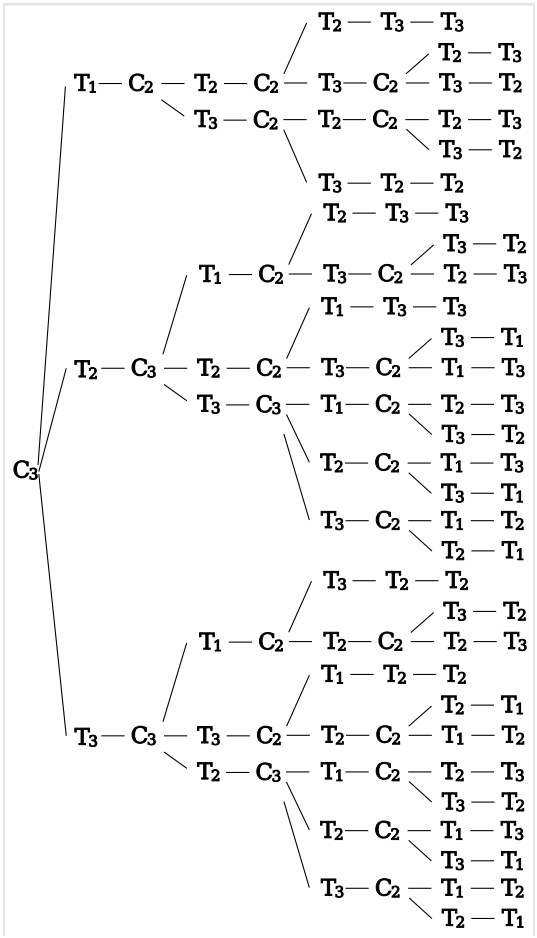


Fig. 19 - CSMA

$$arbre1 = \frac{1}{3}C_3 + \frac{5}{6}C_2 + \frac{20}{12}T$$
$$arbre2 = arbre3 = \frac{7}{9} * C_3 + \frac{4}{9}C_2 + \frac{20}{12}T$$
$$\frac{arbre1}{3} + \frac{arbre2}{3} + \frac{arbre3}{3} = \frac{17}{9}C_3 + \frac{31}{18}C_2 + 5T = 0.0005169$$

1.2.6. Trois stations S_1 , S_2 et S_3 se partagent un segment de réseau de type 802.3 (CSMA/CD, 10Mbps). La première station désire émettre une trame de 1000 bits alors que les deux autres stations souhaitent émettre chacune deux trames de 1250 bits alors que les deux autres stations souhaitent émettre chacune deux trames de 1000 bits. La station S_3 doit attendre l'envoi de la première trame de S_2 avant de commencer à émettre sur le réseau. La durée d'un slot de contention a été fixée à $2\tau = 4.10^{-6}$ s. Considérez les temps de réaction des différentes stations comme nul. Lorsque plusieurs stations veulent accéder au réseau, on supposera que la probabilité de retransmission dans un slot est constante et égale à $p = \frac{1}{4}$. Calculez la durée moyenne d'envoi des 2 premières trames. Sachant que les stations commencent à émettre leur première trame en même temps, sous les contraintes déjà évoquées précédemment.

La probabilité qu'une des 2 stations acquiert le canal dans un slot(libre) est : $A_2 = kp(1-p)^{k-1} = \frac{2}{4}\left(1 - \frac{1}{4}\right)^{2-1} = 0.375$

La probabilité qu'une des 3 stations acquiert le canal dans un slot(libre) est :

$A_3 = kp(1-p)^{k-1} = \frac{3}{4}\left(1 - \frac{1}{4}\right)^{3-1} = 0.421875$

$C_2 = \frac{2\tau}{A_2} = \frac{4 * 10^{-6}}{0.5} = 0.00001067$

$C_3 = \frac{2\tau}{A_3} = \frac{4 * 10^{-6}}{0.421875} = 0.00000948$

$T_1 = \frac{1250}{10 * 10^6} = 0.000125s$

$T_{2,3} = \frac{1000}{10 * 10^6} = 0.0001s$

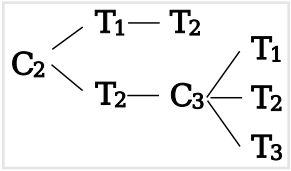


Fig. 20 - CSMA

$$C_2 + \frac{1}{2}(T_1 + T_2) + \frac{1}{2}(T_2 + C_3 + \frac{T_1}{3} + \frac{T_2}{3} + \frac{T_3}{3}) = 0.000232$$

1.2.7. Quatre stations S_1 , S_2 et S_3 et S_4 se partagent un segment de réseau de type 802.3 (CSMA/CD, 10Mbps). La première station désire émettre une trame de 1000 bits, la deuxième une trame de 1250 bits et les deux autres stations souhaitent émettre chacune deux trames de 1500 bits. La durée d'un slot de contention a été fixée à $2\tau = 2.10^{-6}$ s. Lorsque plusieurs stations veulent accéder au réseau, on supposera que la probabilité de retransmission dans un slot est constante et égale à $p = \frac{1}{3}$. Calculez la durée moyenne d'envoi des 3 premières trames. Sachant que les stations commencent à émettre leur première trame en même temps.

La probabilité qu'une des 2 stations acquiert le canal dans un slot(libre) est : $A_2 = kp(1-p)^{k-1} = \frac{2}{3}\left(1 - \frac{1}{3}\right)^{2-1} = 0.444$

La probabilité qu'une des 3 stations acquiert le canal dans un slot(libre) est : $A_3 = kp(1-p)^{k-1} = \frac{3}{3}\left(1 - \frac{2}{3}\right)^{3-1} = 0.444$

La probabilité qu'une des 4 stations acquiert le canal dans un slot(libre) est : $A_4 = kp(1-p)^{k-1} = \frac{4}{3}\left(1 - \frac{4}{3}\right)^{4-1} = 0.395$

$$C_2 = \frac{2\tau}{A_2} = \frac{2 * 10^{-6}}{0.444} = 0.0000045$$
$$C_3 = \frac{2\tau}{A_3} = \frac{2 * 10^{-6}}{0.444} = 0.0000045$$
$$C_4 = \frac{2\tau}{A_4} = \frac{2 * 10^{-6}}{0.395} = 0.0000050625$$
$$T_1 = \frac{1000}{10 * 10^6} = 0.0001s$$
$$T_2 = \frac{1250}{10 * 10^6} = 0.000125s$$
$$T_3 = T_4 = \frac{1500}{10 * 10^6} = 0.00015s$$

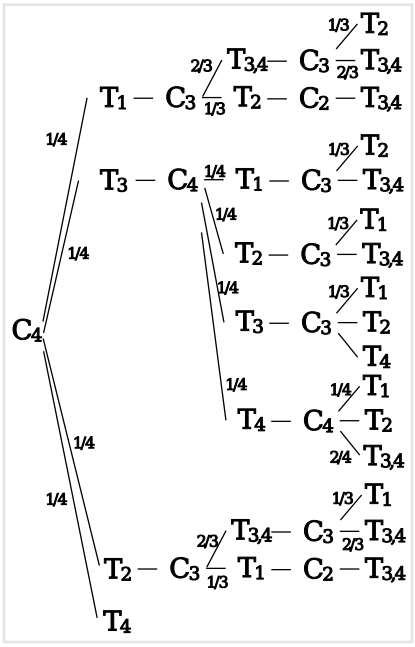
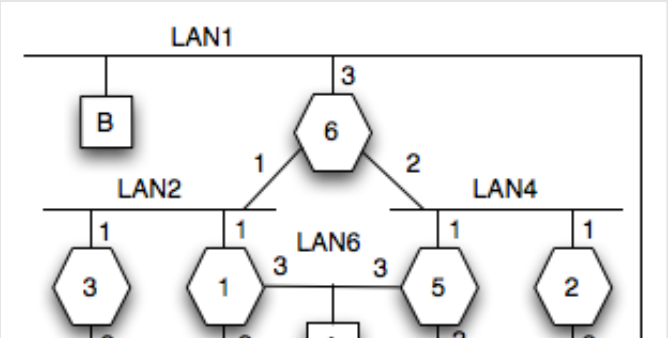


Fig. 21 - CSMA

$$arbre1 = C_4 + \frac{5}{3}C_3 + \frac{1}{3}C_2 + T_1 + \frac{5}{9}T_2 + \frac{13}{9}T_{3,4} = 0.0003952$$
$$arbre2 = C_4 + \frac{5}{3}C_3 + \frac{1}{3}C_2 + T_2 + \frac{5}{9}T_1 + \frac{13}{9}T_{3,4} = 0.0004063$$
$$arbre3 = \frac{9}{4}C_4 + \frac{3}{4}C_3 + \frac{23}{48}T_1 + \frac{23}{48}T_2 + \frac{98}{48}T_{3,4} = 0.0004288$$
$$\frac{arbre1}{4} + \frac{arbre2}{4} + \frac{2arbre3}{4} = 0.0004148$$

1.2.8. Soit le réseau représenté sur la figure. Nous supposons que l'algorithme de *spanning tree* est stabilisé et que toutes les tables d'acheminement des ponts sont initialement vides.



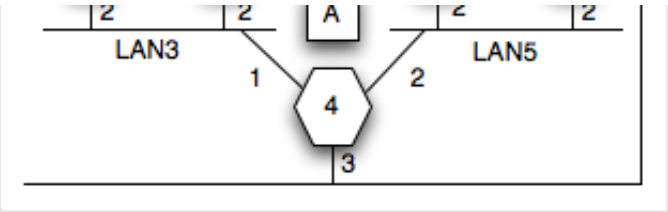


Fig. 22 - Topologie

(a) Donnez les tables relatives au *spanning tree* de chaque pont.
(b) La station B envoie une trame à la station A dont elle connaît l'adresse **MAC**, celle-ci lui répond en lui envoyant à son tour une trame. Décrivez l'évolution des tables d'acheminement des ponts ainsi que les différentes trames qui circulent sur chaque LAN.
(c) Le pont 4 tombe en panne. Décrivez un scénario possible d'échange des BPDU entre les ponts et l'évolution des tables relatives au *spanning tree* jusqu'à stabilité.

(a)

1	**port**	**root ID**	**cost**	**sender ID**	**type**
	1	1	0	1	FP
	2	1	0	1	FP
	3	1	0	1	FP

2	**port**	**root ID**	**cost**	**sender ID**	**type**
	1	1	1	5	BP
	2	1	1	4	RP

3	**port**	**root ID**	**cost**	**sender ID**	**type**
	1	1	0	1	RP
	2	1	0	1	BP

4	**port**	**root ID**	**cost**	**sender ID**	**type**
	1	1	0	1	RP
	2	1	1	4	FP
	3	1	1	4	FP

5	**port**	**root ID**	**cost**	**sender ID**	**type**
	1	1	1	5	FP
	2	1	1	4	BP
	3	1	0	1	RP

6	**port**	**root ID**	**cost**	**sender ID**	**type**
	1	1	0	1	RP
	2	1	1	5	BP
	3	1	1	4	BP

(b)
- IP Source = IP de B
- IP Destination = IP de A

Etape	**1**	**2**	**3**
Mac source	B	4(eth1)	1(eth3)
Mac Destination	4(eth3)	1(eth2)	A

(c) Le pont 4 tombe en panne, le pont 6, 5 et 2 ne recoivent plus de BPDU(1,1,4) de la part du pont 4 et l'entrée le concernant arrive à expiration ; ils vont donc commencer à émettre (BP → FP). Le pont 2 va recevoir BPDU(1,1,5) sur son port 2 et va mettre sa table à jour au profit du port 1 qui passe de (BP → RP) mais le port 2 devient BP.

1	port	root ID	cost	sender ID	type
	1	1	0	1	FP
	2	1	0	1	FP
	3	1	0	1	FP

2	port	root ID	cost	sender ID	type
	1	1	1	5	RP
	2	1	1	5	BP

3	port	root ID	cost	sender ID	type
	1	1	0	1	RP
	2	1	0	1	BP

5	port	root ID	cost	sender ID	type
	1	1	1	5	FP
	2	1	1	5	FP
	3	1	0	1	RP

6	port	root ID	cost	sender ID	type
	1	1	0	1	RP
	2	1	1	5	BP
	3	1	2	6	FP

1.2.9. Considérons le réseau de la figure. La table contient diverses informations sur les routeurs.

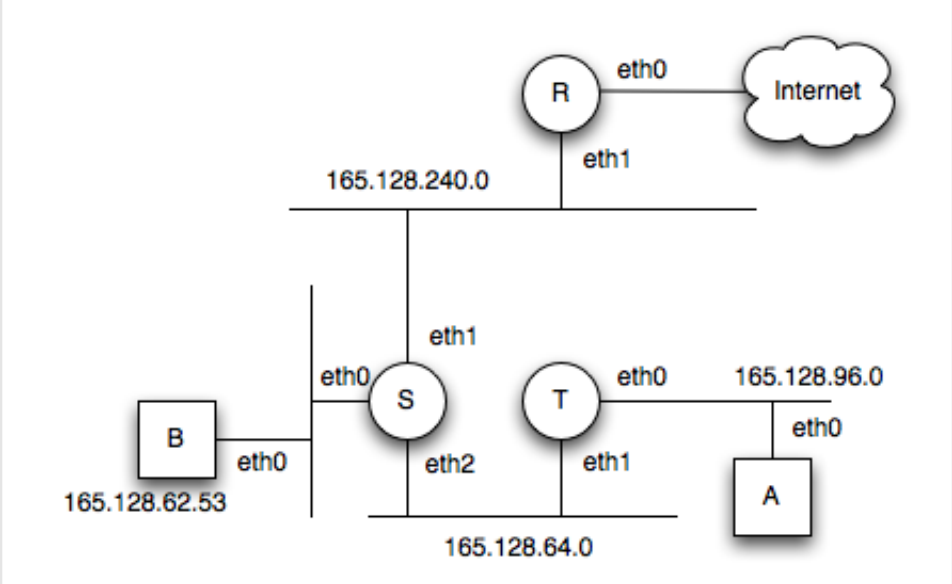


Fig. 23 - Topologie

Routeur	Interface	Netmask	Host ID
R	eth0	255.255.255.0	1
	eth1	255.255.255.0	1
S	eth0	255.255.240.0	2
	eth1	255.255.255.0	2
	eth2	255.255.224.0	2

Routeur	Interface	Netmask	Host ID
T	eth0	255.255.224.0	3
	eth1	255.255.224.0	3

Fig. 24 - Information sur les routeurs

(a) Donnez la table de routage de S.
(b) La station C dont l'adresse **IP** est 165.128.76.193 est ajoutée au réseau. Où la station C sera-t-elle connectée ? Quel sera son ID ?
(c) Donnez l'adresse **IP** de l'interface eth0 du routeur S.
(d) Combien d'adresses **IP** différentes pourraient être attribuées à A?

(a)

Destination	Gateway	Masque	Flags	Iface
localhost	*	255.255.255.255	UH	lo0
165.128.48.0	*	255.255.240.0	U	eth0
165.128.240.0	*	255.255.255.0	U	eth1
165.128.64.0	*	255.255.224.0	U	eth2
165.128.96.0	165.128.64.2	255.255.224.0	UG	eth2
default	165.128.240.2	0.0.0.0	UG	eth1

(b)

x.x.76.193	x.x.0100 1100.1100 0001
x.x.240.0	x.x.1111 0000.0000 0000
x.x.255.0	x.x.1111 1111.0000 0000
x.x.224.0	x.x.1110 0000.0000 0000

Seul possible 255.0 pour le 1100 du 76. Le masque 255.0 est celui du 124.128.240.0 et son host ID sera simplement 193.
(c) 165.128.48.0
(d) $255.255.224.0 = 255.255.11100000.00000000$ donc nb de 0 : $2^{13} = 8192$

1.2.10. Considérons le réseau de la figure. La table 1 contient diverses informations sur les routeurs.

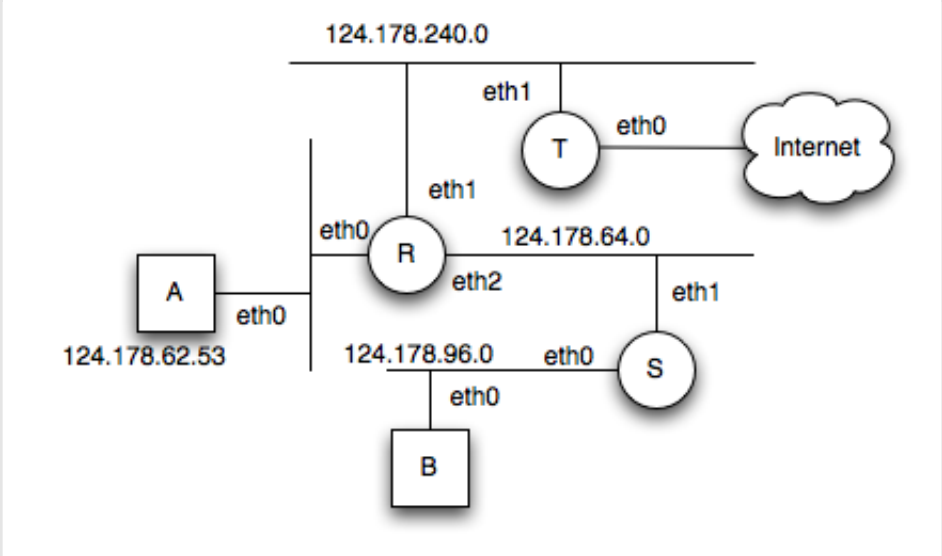


Fig. 25 - Topologie

Routeur	Interface	Netmask	Host ID
R	eth0	255.255.240.0	1
	eth1	255.255.255.0	1

Routeur	Interface	Netmask	Host ID
	eth2	255.255.224.0	1
S	eth0	255.255.224.0	2
	eth1	255.255.224.0	2
T	eth0	255.255.255.0	3
	eth1	255.255.255.0	3

Fig. 26 - Information sur les routeurs

(a) Donnez la table d'acheminement de R.
(b) Combien d'adresses **IP** reste-t-il de disponibles dans le réseau ?
(c) La station B envoie un paquet **IP** à la station A. Décrivez les trames et paquets circulant sur l'intranet, ainsi que l'évolution des différentes tables des routeurs et des stations.

(a)

Destination	Gateway	Masque	Flags	Iface
localhost	*	255.255.255.255	UH	lo0
124.178.240.0	*	255.255.255.0	U	eth1
124.178.48.0	*	255.255.240.0	U	eth0
124.178.64.0	*	255.255.224.0	U	eth2
124.178.96.0	124.178.64.2	255.255.224.0	UG	eth2
default	124.178.240.3	0.0.0.0	UG	eth1

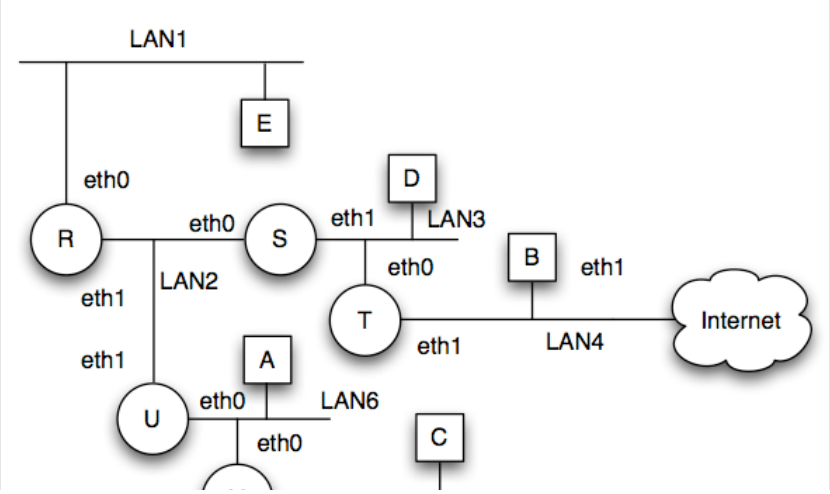
(b)

Destination	Masque	# IP libres
124.178.240.0	255.255.255.0	$2^8 - R(eth1) - T(eth1) - 2 = 252$
124.178.48.0	255.255.11110000.00000000	$2^{12} - R(eth0) - A - 2 = 4092$
124.178.64.0	255.255.11100000.00000000	$2^{13} - R(eth0) - S(eth1) - 2 = 8188$
124.178.96.0	255.255.11100000.00000000	$2^{13} - S(eth0) - B - 2 = 8188$
		$Total = 20720$

(c)
- IP Source = IP de B = 124.178.96.X
- IP Destination = IP de A = 124.178.62.53

Etape	1	2	3
Mac source	B	S(eth1)	R(eth0)
Mac Destination	S(eth0)	R(eth2)	A

1.2.11. Considérons le réseau de la figure. Les tables contient diverses informations sur les routeurs.



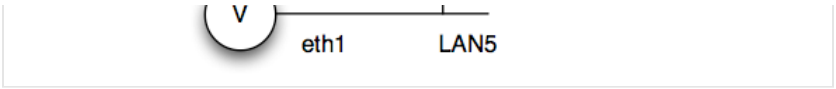


Fig. 27 - Topologie

LAN	Address
1	192.200.36.0
2	192.200.34.0
3	200.14.208.0
4	200.14.2.0
5	128.128.128.0
6	129.129.129.0

Fig. 28 - Information sur les routeurs

LAN	Address
A	13
B	125
C	234
D	50
E	79

Fig. 29 - Information sur les routeurs

Routeur	Interface	Netmask	Host ID
R	eth0	255.255.255.0	1
	eth1	255.255.255.0	1
S	eth0	255.255.255.0	2
	eth1	255.255.224.0	2
T	eth0	255.255.224.0	3
	eth1	255.255.255.0	3
U	eth0	255.255.255.0	4
	eth1	255.255.255.0	4
V	eth0	255.255.255.0	5
	eth1	255.255.255.0	5

Fig. 30 - Information sur les routeurs

(a) Donnez la table d'acheminement de R.
(b) La station A envoie un paquet IP à la station B. Décrivez les trames et paquets circulant sur l'intranet, ainsi que l'évolution des différentes tables des routeurs et des stations
(c) Oublions les informations données dans les tables et ne regardons que les LAN 5 et 6 ainsi que les routeurs R, U et V. L'adresse du LAN 6 devient : 124.178.16.0 et l'adresse du LAN 5 devient 124.178.24.0. De plus, la station A aura pour host ID 517 et la station C aura pour host ID 517. Le routeur R pourra-t-il fusionner les entrées de sa table d'acheminement sachant que A et C ont le même host ID ? Que se passera-t-il losque C recevra un paquet où le host ID est de 517 ?

(a)

Destination	Gateway	Masque	Flags	Iface
localhost	*	255.255.255.255	UH	lo0
192.200.36.0	*	255.255.255.0	U	eth0
192.200.34.0	*	255.255.255.0	U	eth1
200.14.208.0	192.200.34.2	255.255.224.0	UG	eth1
200.14.2.0	192.200.34.2	255.255.255.0	UG	eth1
128.128.128.0	192.200.34.4	255.255.255.0	UG	eth1

Destination	Gateway	Masque	Flags	Iface
129.129.129.0	192.200.34.4	255.255.255.0	UG	eth1
default	192.200.34.2	0.0.0.0	UG	eth1

Fusion de default, LAN3 et LAN4

Destination	Gateway	Masque	Flags	Iface
localhost	*	255.255.255.255	UH	lo0
192.200.36.0	*	255.255.255.0	U	eth0
192.200.34.0	*	255.255.255.0	U	eth1
128.128.128.0	192.200.34.4	255.255.255.0	UG	eth1
129.129.129.0	192.200.34.4	255.255.255.0	UG	eth1
default	192.200.34.2	0.0.0.0	UG	eth1

(b)
- IP Source = IP de A = 129.129.129.13
- IP Destination = IP de B = 200.14.2.125

Etape	1	2	3	4
Mac source	A	U(eth1)	S(eth1)	T (eth1)
Mac Destination	U(eth0)	S(eth0)	T (eth0)	B

(c) Oui, sans aucun problème car ...

1.2.12. Considérons le réseau de la figure. Les tables contient diverses informations sur les routeurs.

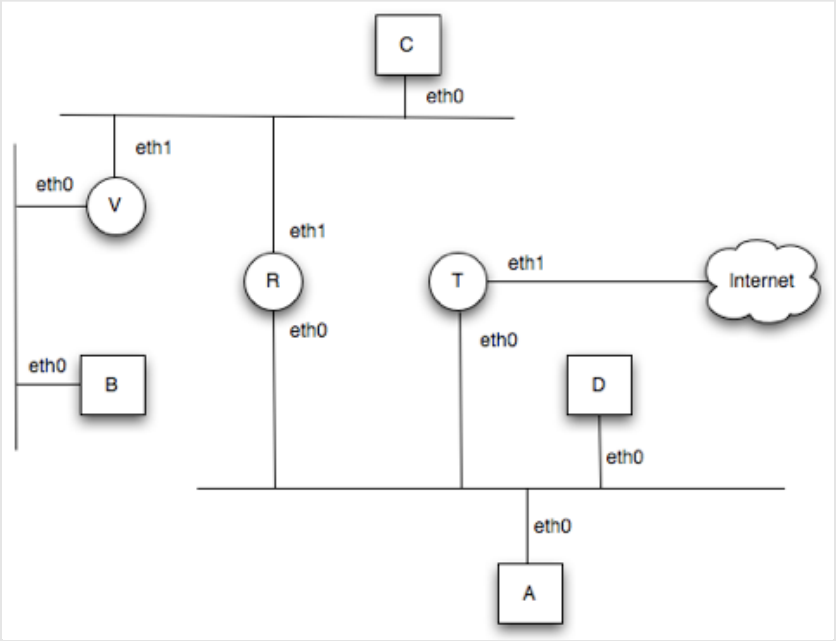


Fig. 31 - Topologie

Routeur	Interface	Host ID
R	eth0	2
	eth1	2
T	eth0	3
	eth1	3
V	eth0	4
	eth1	4

Fig. 32 - Information sur les routeurs

Station	Interface	IP	Netmask
A	eth0	140.140.140.1	0xffff0000
B	eth0	150.150.150.1	0xffffff00
C	eth0	160.160.160.1	0xffffff00
D	eth0	?	?

Fig. 33 - Information sur les routeurs

(a) Donnez l'adresse IP de D sachant que son host ID est 513.
(b) 2 Donnez les tables d'acheminement de R, V, A et B.
(c) 3 Considérons un transfert unidirectionnel de données entre A et B. Donnez les adresses IP qui sont utilisées dans les datagrammes qui circulent sur le réseau.

(a) Pour déterminer l'adresse IP de D, il nous manque son netmask et son network ID. Ces valeurs sont identiques pour toutes les adresses IP se trouvant sur le même sous-réseau que D. Nous pourrions donc utiliser $IP_{R_{eth0}}$, $IP_{T_{eth0}}$ ou `IP_A` . Nous avons :

$$\begin{aligned} IP_A &= 10001100.10001100.10001100.00000001 \\ Netmask &= 11111111.11111111.00000000.00000000 \\ IP_A\ AND\ Netmask &= 10001100.10001100.00000000.00000000 \end{aligned}$$

Car $IP_A = 140.140.140.1$ L'adresse IP de D est de la forme :

$$10001100.10001100.xxxxxxxx.xxxxxxxx$$

où la partie non définie est complétée avec le hostID. Le hostID de D est 513. L'encodage de $513 = 512 + 1 = 2^9 + 2^0$ sur 16 bits est 00000010.00000001. L'adresse du terminal D est donc :

$$\begin{aligned} IP_D &= 10001100.10001100.00000010.00000001 \\ &= 140.140.2.1 \end{aligned}$$

(b) La table d'acheminement la plus probable pour R est :

Destination	Gateway	Genmask	Flags	Iface
localhost	*	255.255.255.255	UH	lo0
140.140.0.0	*	255.255.0.0	U	eth0
150.150.150.0	160.160.160.4	255.255.255.0	UG	eth1
160.160.160.0	*	255.255.255.0	U	eth1
default	140.140.0.3	0.0.0.0	UG	eth0

Une table d'acheminement possible pour V est :

Destination	Gateway	Genmask	Flags	Iface
localhost	*	255.255.255.255	UH	lo0
150.150.150.0	*	255.255.255.0	U	eth0
160.160.160.0	*	255.255.255.0	U	eth1
140.140.0.0	160.160.160.2	255.255.0.0	UG	eth1
default	160.160.160.2	0.0.0.0	UG	eth1

Le comportement du routeur pour les entrées $140.140.0.0$ et default est le même (envoi du paquet à Reth1) et on peut regrouper ces entrées. La table d'acheminement la plus probable pour V est :

Destination	Gateway	Genmask	Flags	Iface
localhost	*	255.255.255.255	UH	lo0
150.150.150.0	*	255.255.255.0	U	eth0

Destination	Gateway	Genmask	Flags	Iface
160.160.160.0	*	255.255.255.0	U	eth1
default	160.160.160.2	0.0.0.0	UG	eth1

La table d'acheminement la plus probable pour A est :

Destination	Gateway	Genmask	Flags	Iface
localhost	*	255.255.255.255	UH	lo0
140.140.0.0	*	255.255.0.0	U	eth0
160.160.160.0	140.140.0.2	255.255.255.0	UG	eth0
150.150.150.0	140.140.0.2	255.255.255.0	UG	eth0
default	140.140.0.3	0.0.0.0	UG	eth0

Le comportement du routeur pour les entrées 160.160.160.0 et 150.150.150.0 est le même mais regrouper ces entrées est impossible. Le plus petit groupe contenant toutes les adresses de ces sous-réseaux est 128.0.0.0/2 et la table de A ne peut pas contenir l'entrée :

Destination	Gateway	Genmask	Flags	Iface
128.0.0.0	140.140.0.2	192.0.0.0	UG	eth0

Prenons, par exemple, l'adresse 128.1.1.1. Les entrées valides pour cetteadresse seraient 128.0.0.0 et default. Comme le masque de 128.0.0.0 estplus long, c'est cette entrée qui sera choisie ⇒ Erreur : cette adressedevrait être routée vers “Internet”.
Une table d'acheminement possible pour B est :

Destination	Gateway	Genmask	Flags	Iface
localhost	*	255.255.255.255	UH	lo0
150.150.150.0	*	255.255.255.0	U	eth0
160.160.160.0	150.150.150.4	255.255.255.0	UG	eth0
140.140.0.0	150.150.150.4	255.255.0.0	UG	eth0
default	150.150.150.4	0.0.0.0	UG	eth0

Le comportement du routeur pour les entrées 160.160.160.0, 140.140.0.0 et default est le même et on peut regrouper ces entrées. La table d'acheminement la plus probable pour B est :

Destination	Gateway	Genmask	Flags	Iface
localhost	*	255.255.255.255	UH	lo0
150.150.150.0	*	255.255.255.0	U	eth0
default	150.150.150.4	0.0.0.0	UG	eth0

(c) Le paquet doit transiter par R. Les adresses IP source et destination du paquet sont respectivement 140.140.140.1 et 150.150.150.1.
Important : Lors de l'envoi d'un paquet IP de A vers B l'adresse source du paquet sera `IP_A` et l'adresse de destination du paquet sera IPB depuis l'émission du paquet par A jusqu'à la réception de celui-ci par B : **on ne modifie pas les adresses source et destination d'un paquet IP en cours de route**.

1.2.13. Soit les tables suivantes :

pont	port	root ID	cost	sender ID	type
pont 1	1	1	0	1	FP
	2	1	0	1	FP
pont 2	1	1	0	1	RP
	2	1	1	2	FP
pont 3	1	1	0	1	RP
	2	1	1	2	BP

pont	port	root ID	cost	sender ID	type
pont 4	1	1	0	1	RP
	2	1	1	2	BP
	3	1	0	1	BP

Fig. 34 - tables du spanning tree

(a) Dessinez un réseau possible, respectant les tables relatives au spanning tree données à la table 1.
(b) La station A envoie une trame à la station B dont elle connaît l'adresse **MAC**, celle-ci lui répond en lui envoyant à son tour une trame. Décrivez l'évolution des tables d'acheminement des ponts, sachant que celles-ci sont initialement vides, ainsi que les différentes trames qui circulent sur chaque LAN. Pour arriver à destination, cette trame devra traverser deux ponts.
(c) Le pont 1 tombe en panne. Décrivez un scénario possible d'échange des BPDU entre les ponts et l'évolution des tables relatives au spanning tree jusqu'à stabilité.

(a)

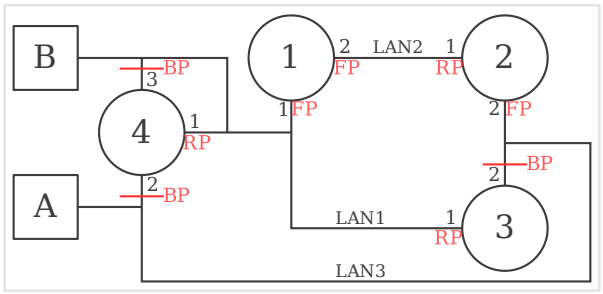


Fig. 35 - Réseau possible

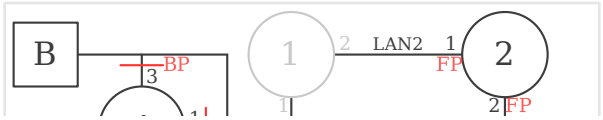
(b) Voir (a).

Etape	1	2	3	4	5	6
Mac source	A	2(eth1)	1(eth1)	B	1(eth2)	2(eth2)
Mac Destination	2(eth2)	1(eth2)	B	1(eth1)	2(eth1)	A

1. A envoie une trame $MAC_{source} = MAC_A$ et $MAC_{dest} = MAC_B$ sur `LAN3` et le pont 2 port 2 recoit la trame. Il associe aussi `MAC_A` à son port 2.
2. Le pont 2 renvoie sur le `LAN2` via le port 1. Le pont 1 recoit la trame sur le port 2, il associe `MAC_A` sur son port 2 et renvoie la trame sur le `LAN1` via son port 1.
3. Le pont 4 et le port 3 recoivent la trame, ils associent `MAC_A` sur leur port 1. B reçoit la trame.
4. B envoie une trame $MAC_{source} = MAC_B$ et $MAC_{dest} = MAC_A$ sur `LAN1` et le pont 1,3,4 recoivent la trame sur leur port 1. Ils associe aussi `MAC_B` à leur port 1.
5. Le pont 1 renvoie sur son port 2 sachant sa table et le pont 2 recoit sur son port 1.
6. Le pont 2 renvoie sur son port 2 sachant sa table et A recoit.

(c) Le pont 1 tombe en panne, il n'emet plus son BPDU et l'entrée qui lui est associé au pont 2,3 et 4 arrivent à expiration, ils commencent donc à emettre par default sur leurs ports. Le pont 3 recevant un meilleur BPDU sur son port 2 va mettre à jour sa table avec un $rootID = 2$ et mettre son port 1 à FP et son port 2 à RP. Le port 4 recoit 3 BPDU ; BPDU(2,0,2) sur le port 2 (devient RP) et BPDU(2,1,3) sur les port 1,3 (tous les deux BP car le port 2 est meilleur).

pont	port	root ID	cost	sender ID	type
pont 2	1	2	0	2	FP
	2	2	0	2	FP
pont 3	1	2	0	2	RP
	2	2	1	3	FP
pont 4	1	2	1	3	BP
	2	2	0	2	RP
	3	2	1	3	BP



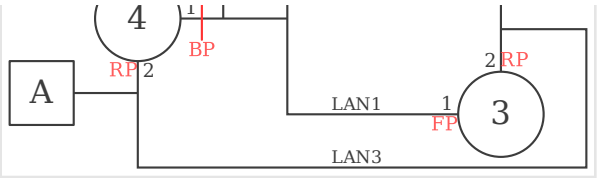


Fig. 36 - Réseau possible