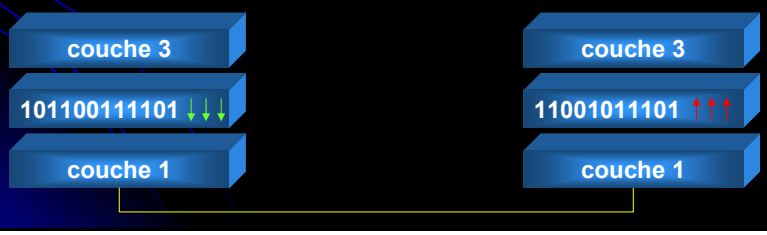
3. La couche Liaison de données

- Calquée sur le couche 2 du modèle OSI, cette couche à pour objectifs principaux de :
  - fiabiliser la couche 1
  - d'assurer le transfert, point à point, d'un train de bits : la ***trame***
  - d'assurer l'accès au support
  - d'assurer un certain nombre de validation :
    - flux
    - erreur
  - de "délivrer" la trame au bon destinataire

Protocoles de la couche 2

- Sachant que :
  - la couche 1 transmet des bits et pas des paquets
  - la couche 1 peut subir des erreurs de transmission :
    - erreurs de transmission
    - perte de bits
    - création de bits



Protocoles de la couche 2

- Avant d'exposer le cas spécique d'Ethernet, quelle est la problématique de la couche 2 :
  - pourquoi Ethernet :
    - 95 % des installations
  - protocoles successifs :
    - réponses de plus en plus complètes aux différents situations complexes pouvant se présenter et posant des difficultés

Les types de service

- Rappel :
  - nous en avons déjà parlé lors de la description du modèle OSI
- La couche 2 peut, selon la *QoS* qui lui est demandée, accomplir un certain nombre de services :
  - dans le but d'assurer une Qualité de Service
- Il faut définir le compromis :
  - fiabilité ↔ augmentation de la charge

Les types de service

- ***Unacknowledged connectionless service*** :
  - détection des trames
  - détection des erreurs de transmission
  - → niveau de service minimal
- ***Acknowledged connectionless service*** :
  - *détection des trames*
  - *détection des erreurs de transmission*
  - *gestion des acquits*
  - *retransmissions éventuelles*

Les types de service

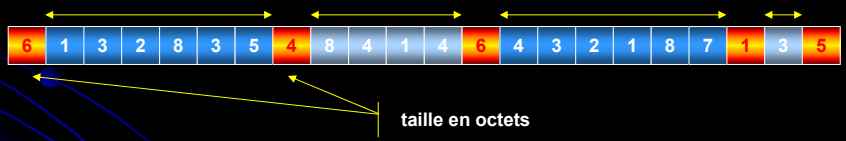
- ***Acknowledged connection-oriented service*** :
  - *établissement et fermeture de la connexion*
  - *détection des trames*
  - *détection des erreurs de transmission*
  - *gestion des acquits*
  - *retransmissions éventuelles*

Détection des trames

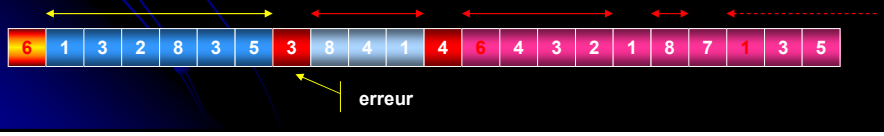
- Qu'est-ce qu'une ***trame*** ?
  - unité d'information transférée entre deux entités de la couche liaison de données
  - mais aussi : groupe logique (pour ce niveau) de ***N bits***
- En fait :
  - la taille est variable, comprise entre deux bornes

Détection des trames

- Quelle est la nature de ces bornes ?
  - en général, la taille d'une trame est de K octets
  - on pourrait ajouter, ***au début*** *de la trame*, sa longueur en octet



  - quid si une erreur se glisse dans la longueur ?

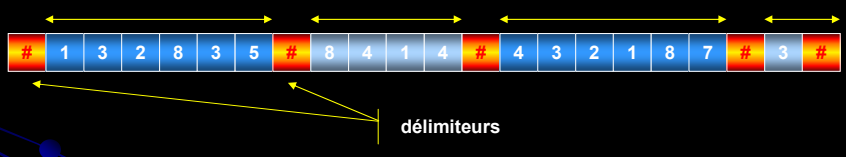


Détection des trames

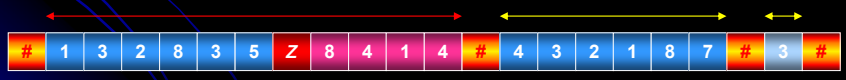
- Conséquences :
  - très dangereux car la moindre erreur de transmission sur la longueur génère des erreurs :
    - irrécupérables
    - difficilement détectables
  - mode byte et non bit :
    - les longueurs de trames sont donc forcément des multiples de … 8

Détection des trames

- on pourrait utiliser un caractère spécial pour marquer les bornes :
  - exemple : le caractère #



- quid en cas d'erreurs sur les délimiteurs ?



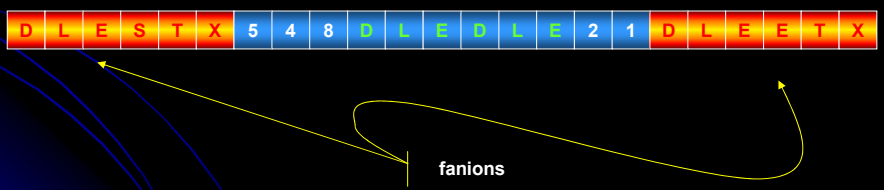
Détection des trames

- conséquences :
  - meilleure protection aux erreurs. Si un délimiteur est corrompu, seules deux trames sont erronées
  - quid si le caractére de délimitation fait partie des infos de la trame ?
    - on le remplace par un substitut : !#
    - soit la trame ci-dessous à transmettre :

    - la trame effectivement transmise sera :

Détection des trames

- en pratique, utilisation de '***fanions***' dans la technique du ***character stuffing :***
  - 'DLE' 'STX' : *DeLimitEr Start of TeXT* : au début
  - 'DLE' 'ETX' : *DeLimitEr End of TeXT* : à la fin
  - doublement en cas de présence dans la partie donnée



Détection des trames

- alternative : le ***bit stuffing***
  - 01111110 est ajouté en début et en fin de trame
  - si 5 [1] consécutifs apparaissent dans la trame, on insère automatiquement un [0]. Dans ce cas, au traitement par la couche 2 réceptrice, dès qu'elle lit 5 [1] consécutifs, elle supprime d'office le bit suivant
  - exemple :
    - paquet *reçu* de la couche réseau :
      01101111111111111111110010
    - trame *envoyée* par la couche liaison de données
      01111110011011111101111101111101100100111110
    - paquet *délivré* à la réseau :
      01101111111111111111110010

Détection des trames

- conséquences :
  - par rapport au character stuffing, les trames ne sont plus obligatoirement des multiples de ***8 : N * 8 bits***

Détection des trames

- une quatrième solution implique une collaboration avec la couche physique :
  - utilisation de codes spéciaux de la couche physique, en fait des ***codes invalides***, pour coder les débuts et fins de trames
  - exemple en Manchester :
    - [invalide 1] : haut suivi de haut
    - [invalide 2] : bas suivi de bas
  - il suffit alors d'utiliser N fois [***invalide 1***] pour marquer le ***début*** d'une trame et N fois [***invalide 2***] pour en marquer la ***fin***
- Cette solution est retenue dans certains LAN

Détection des trames

- En pratique :
  - utilisation du bit stuffing
  - et indication de la longueur
  - et code de contrôle
- Une trame reçue est considérée ***valide*** ⇔ :
  - le bon délimiteur est au début
  - la longueur est correcte
  - le bon délimiteur est à la fin
  - le code de contrôle est correct

Détection et correction d'erreurs

- Nous avons déjà évoqué ces concepts au niveau de la couche 1 :
  - détection : parité, CRC
  - correction : hamming, ...
- La couche 2 fait appel à ces mêmes concepts. Nous ne les revoyons donc pas

Détection et correction d'erreurs

- Toutefois, à titre de rappel, dans un LAN actuel, c'est la technique de la détection qui est privilégiée car les erreurs sont relativement rares pour les raisons suivantes :
  - supports fiables (du moins filaire) :
    - distances courtes
    - peu de perturbations
    - matériaux de qualité
  - applications « tolérantes » aux erreurs :
    - multimédia, ...

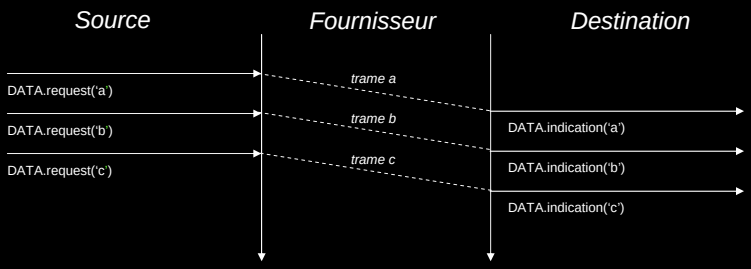
Détection et correction d'erreurs

- Pour la majorité des applications, une correction d'erreurs, supposerait :
  - un accroissement du volume à transmettre
  - et donc un gaspillage de bande passante
  - des délais allongés
  - ... *pour rien*

Protocoles de la couche 2

- Comment transmettre des trames de A vers B ?
  - hypothèses :
    - l'utilisateur de la couche liaison de données a , en permanence, des paquets à transmettre
    - la couche physique est parfaite
    - transmission de A vers B uniquement

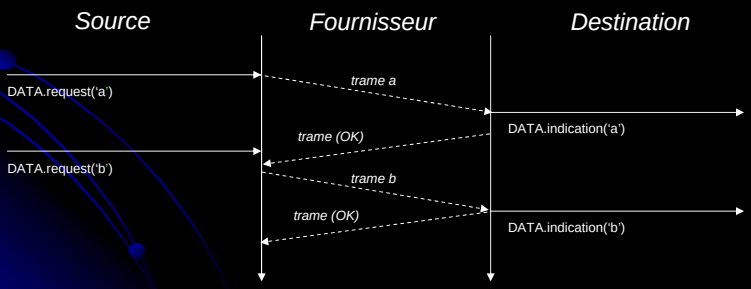
Protocole – étape 1



- la couche 2 peut se borner à ***détecter les limites*** des trames :
- contenu d'une trame : [<FLAG><Paquet><FLAG>]
- ***problème*** :
  - que faire si B est plus lent que A ?

Protocole – étape 2

- Solution :
  - asservir l'émetteur au récepteur. Le récepteur doit envoyer une trame spéciale autorisant l'émetteur à envoyer de nouvelles données

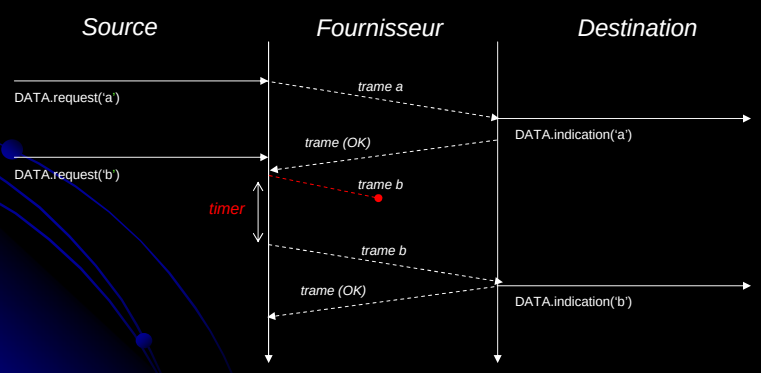


Protocole – étape 3

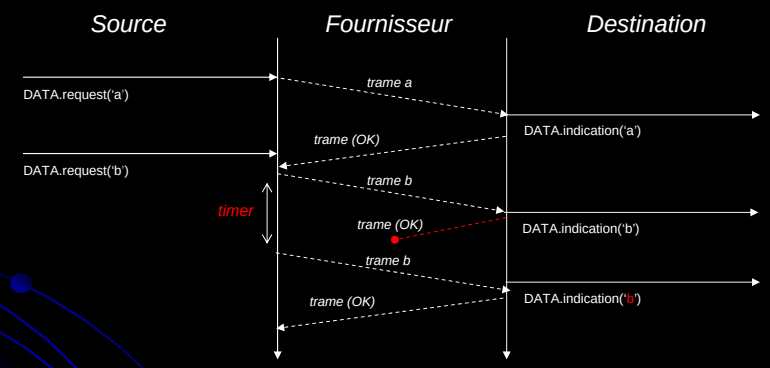
- La solution précédente est parfaite, ***mais*** :
  - la couche physique n'est pas parfaite
  - la transmission doit être full duplex
  - …
- Pour palier l'imperfection de la couche physique, nous pouvons ajouter un CRC :
  - contenu d'une trame : [*<FLAG><Paquet><CRC><FLAG>*]
- Quid des erreurs de transmission ?

Protocole – étape 3a

- Pour couvrir ces dernières, on peut associer un timer :



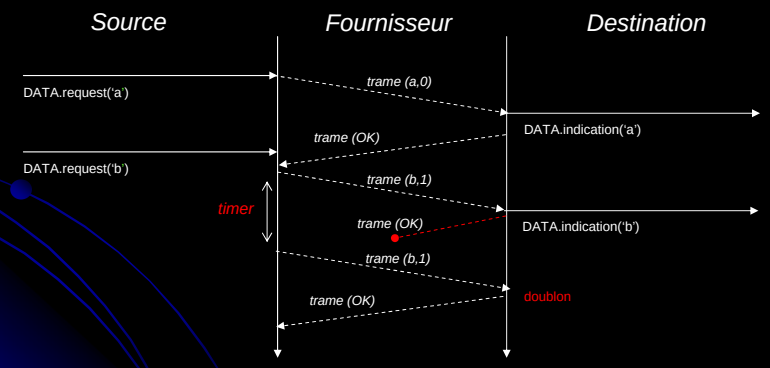
Protocole - étape 3a … oui mais



- Que s'est-il passé ?
  - l'acquit s'est perdu … génération d'un doublon

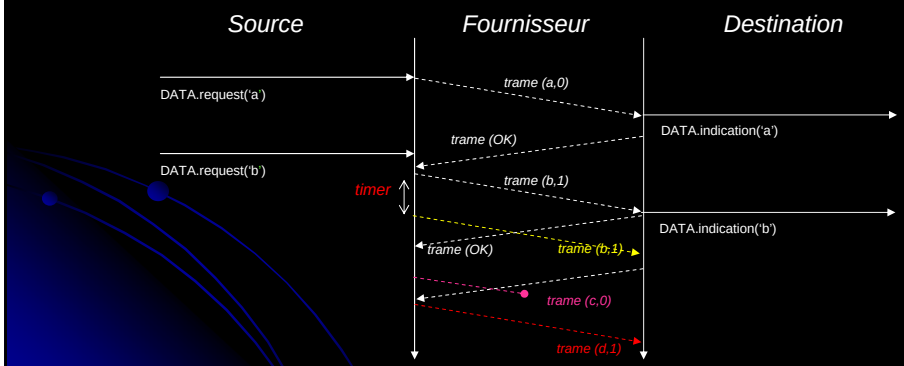
Protocole – étape 3b

- Il suffit de numéroter les trames :



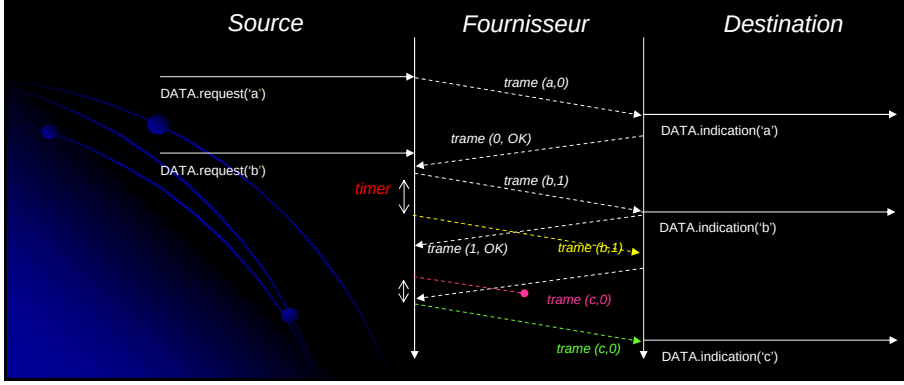
Protocole – étape 3b … oui mais

- Cela n'est pas encore suffisant. Pourquoi ?



Protocole étape – 3c

- Solution : le top …
  - numéroter les acquits. Donc, nouvelle modification de la trame



Communication bidirectionnelle

- Jusqu'à présent nous avons remédier aux problèmes de transmission. Comment assurer la ***bidirectionnalité*** ?
  - ouvrir deux connexions physique :
    - gaspillage et peu économe
  - utiliser des trames de données comme porteuses :
    - placer le champs ***ACK*** dans des trames normales :
      - → *Piggyback*
    - [*<FLAG><Type><ACK><Num><Paquet><CRC><FLAG>*]
    - [*<FLAG><Type><ACK><CRC><FLAG>*]

Communication bidirectionnelle

- Le ***piggyback*** est parfait si la communication est bidirectionnelle en permanence. Sinon problème :
  - comment envoyer l'acquit à l'émetteur si le destinataire n'a rien à dire ?
- Solution : ***retarder*** les acquits
  - si on peux acquitter immédiatement (une trame de données est prête), alors c'est parfait
  - sinon : quid ?

Communication bidirectionnelle

- un *temporisateur* est armé :
  - si une trame de données se présente, on l'utilise
  - sinon, on crée une trame spéciale rien que pour acquitter.
- si on agit pas de la sorte, le protocole est bloquant

Notion de performance

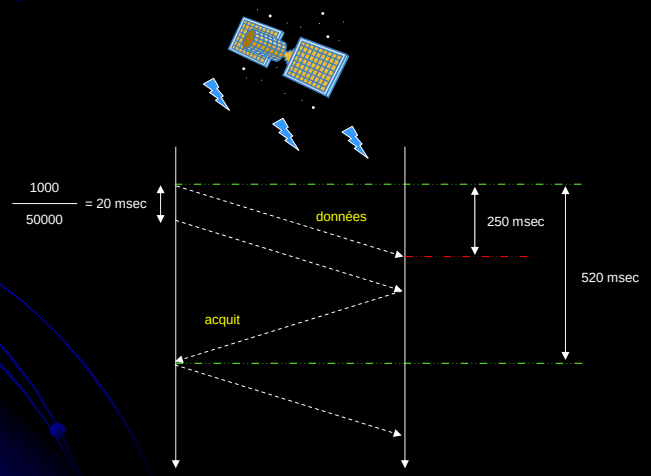
- Soit un protocole 1 bit = aller-retour :
  - il attend l'acquittement avant l'envoi de la trame suivante
  - soit un délai de 250 msec pour l'aller A $\rightarrow$ B (cas d'un satellite)
  - soit une taille max. de 1.000 bits par trame
  - soit enfin un débit 50 kbps

Notion de performance

- Les performances dépendent directement du produit : ***débit * délai aller-retour :***
  - dans cet exemple, l'émetteur est bloqué 500/520, soit 96% de son temps
  - la bande passante effectivement utilisée n'est que de 4% de son potentiel
  - la situation s'empire si :
    - trame courte
    - BP large
    - temps de transit important

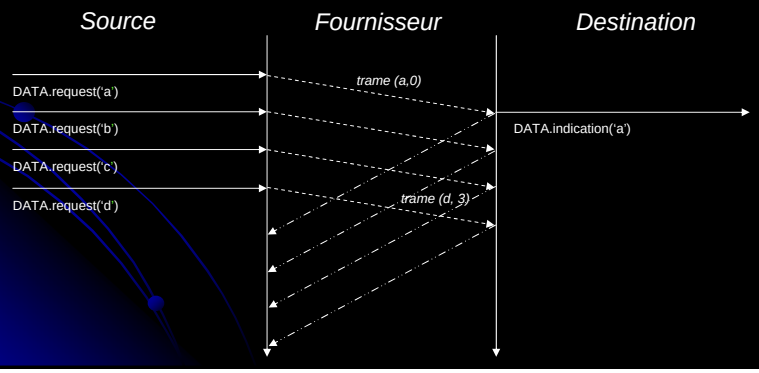
Notion de performance

- Graphiquement :



Amélioration des performances

- L'aller-retour est très pénalisant. Une amélioration est possible avec la technique du ***pipeline*** :



Amélioration des performances

- cette technique autorise l'émetteur à envoyer plusieurs trames successivement sans attendre d'acquits

- questions :
  - combien de temps attendre
  - comment gérer les trames en attente :
    - ***numérotation*** des séquences et des acquits
    - technique des ***fenêtres*** d'émission et de réception

Amélioration des performances

- Numérotation :
  - rappel : avec n bits nous pouvons représenter 2^n nombres
  - quid si on envoi $2^N + 1$ paquets :
    - utilisation du ***modulo*** :
    - exemple : si n = 2

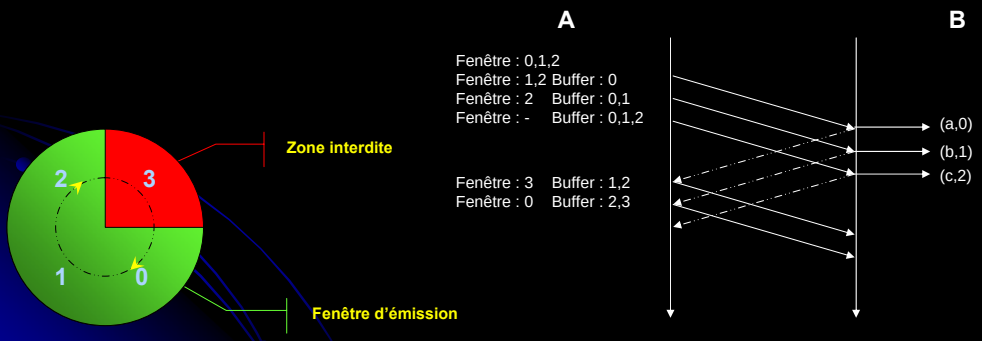
      0, 1, 2, 3, 0, 1, 2, 3, …

Amélioration des performances

- Comment éviter de saturer les mémoires du récepteur :
  - technique des ***sliding windows*** :
    - fenêtre glissante
    - à tout moment, l'émetteur tient la liste des numéros de séquence qu'il peut envoyer : ***sending window***
    - à tout moment, le récepteur tient la liste des numéros de séquence qu'il accepte de recevoir : ***receiving window***
    - les trames sont numérotées sur ***n*** bits
    - fenêtre <= 2^n trames

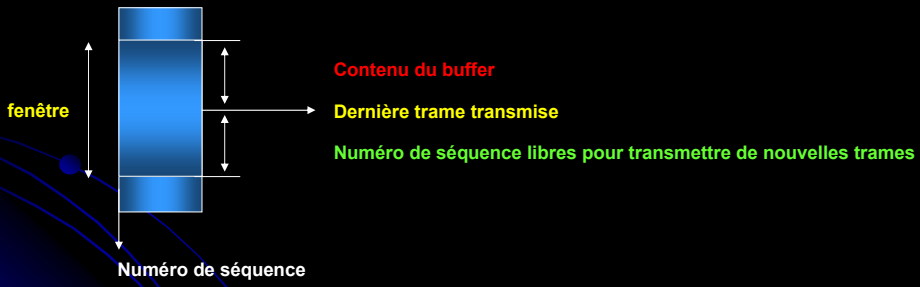
Fenêtre glissante : exemple

- Exemple d'une fenêtre sur 2 bits
  - 3 trames consécutives
  - rôle des buffers



Fenêtre glissante : fonctionnement

- Fenêtre d'émission :
  - nombre maximum de trames en attente d'un acquit



  - La réception d'un acquit décale la fenêtre vers les numéros de séquence croissants (modulo fenêtre)

Récupération des erreurs

- Comment détecter une ***erreur*** (ou une ***perte***) :
  - cas du protocole 1 bit (simple) :
    - → expiration d'un timer à l'émetteur
      - [détection à l'émetteur]
  - autre protocole (fenêtres glissantes) :
    - → expiration d'un timer à l'émetteur
      - un timer par trame !!!
      - [détection à l'émetteur]

Récupération des erreurs

- réception d'une trame avec CRC erroné :
  - [détection au récepteur]
  - transmission éventuelle d'un ***acquit négatif***
- réception de la trame N+1 ***avant*** la trame N-1 :
  - [détection au récepteur]
  - transmission éventuelle d'un acquit négatif

Acquits et fenêtres

- Quel type d'acquits transmettre :
  - en général, on place dans le champ ***ACK*** des trames transmises le numéro de séquence de la « dernière » trame transmise à l'utilisateur
  - dans certains cas, on peut aussi prévoir une trame de contrôle spécifique (*NAK*) qui indique à l'émetteur qu'une trame n'a pas été reçue afin d'accélérer la retransmission de cette trame

Mécanisme de retransmission

- Comment retransmettre une trame perdue :
  - règle :
    - l'émetteur *doit* garder en mémoire les trames tant qu'elles n'ont pas été acquittées
  - deux méthodes principales :
    - ***GO-BACK-END*** :
      - simple à implémenter et efficace si … peu d'erreurs
    - ***Selective repeat*** :
      - plus complexe à implémenter mais plus efficace si le taux d'erreurs est élevé

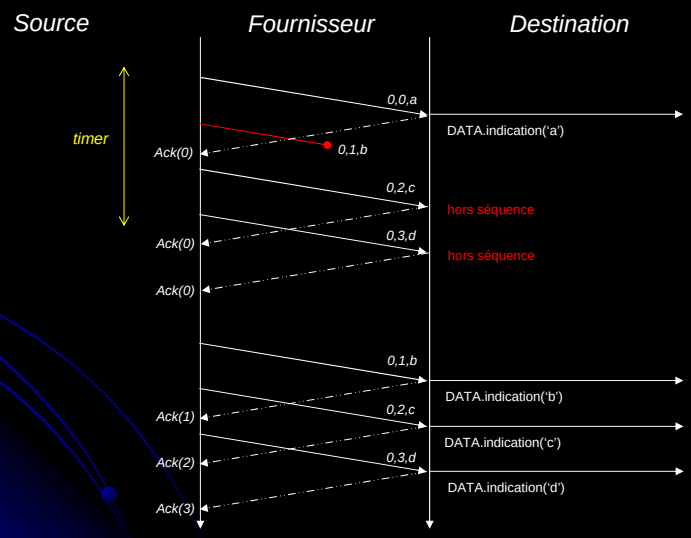
GO-BACK-END

- Principe :
  - en cas d'erreur, on recommence la transmission ***à partir de la trame perdue***
  - on retransmet donc la trame perdue ***et*** toutes celles transmises après !!! (dernier acquit)
  - avantages :
    - simple pour le receveur :
      - il ne doit pas mémoriser les trames reçues après une trame erronée
    - détection des erreurs avec un simple timer
    - facile à implémenter
    - peu gourmant en ressource pour les machines

GO-BACK-END

- désavantages :
  - les performances chutent rapidement avec l'augmentation du taux d'erreurs sur la ligne physique
- la nature du réseau et ses caractéristiques physiques sont donc très importantes dans cette technique

GO-BACK-END : exemple



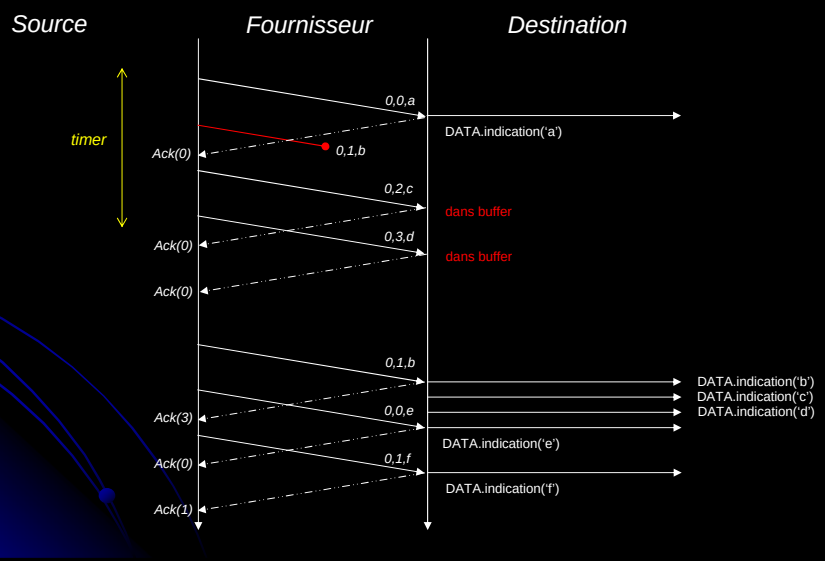
Selective repeat

- Principe :
  - lorsqu'une trame est perdue, on ne retransmet ***que cette dernière***
  - Avantages :
    - évite de retransmettre inutilement des trames déjà transmises (correctement)
    - performances meilleures que GO-BACK-END lorsque la couche physique est mauvaise

Selective repeat

- Inconvénients :
  - implémentation plus complexe :
    - le receveur doit stocker les trames reçues « hors-séquence » dans un buffer
    - le recevoir doit aussi réordonner les trames reçues avant de les délivrer à son utilisateur (sa couche 3)

Selective repeat : exemple



Taille de la fenêtre

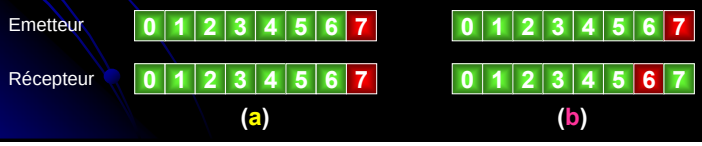
- Combien de ***trames non acquittées*** peux-t-on gérer ?
  - dans un protocole à fenêtre de largeur N, comme go-back-end, on peut avoir N trame :
    - ainsi, si les trames sont numérotées sur 3 bits, nous pouvons avoir 7 et non 8 trames non acquittées
    - exemple :
      - A envoi les trames de 0 → 7
      - A reçoit acquit superposé (cumulatif) pour la trame 7
      - A envoi une nouvelle série de huit trames : 0 → 7
      - de nouveau, A reçoit un acquit cumulatif pour la trame 7

Taille de la fenêtre

- La ***question*** est alors la suivante :
  - toutes les trames appartenant au second envoi sont-elles :
    - arrivées dans l'ordre
    - ou bien ont-elles été toutes perdues (les trames suivant une trame erronée sont perdues) ?
  - ***dans les deux cas***, B enverrait la trame 7 comme acquittement !!!
  - le fait de limiter à N le nombre d'acquit permet d'induire un ***glissement*** dans la numérotation des acquits

Selective repeat & taille fenêtre

- La possibilité d'envoyer des trames de manière non séquentielle induit de nouvelles contraintes sur la technique des fenêtres glissantes :
  - Exemple :
    - soit une fenêtre utilisant 3 bits
    - au départ, l'émetteur peut envoyer 7 trames et le récepteur peut en recevoir 7 : état (a)



(a) (b)

Selective repeat & taille fenêtre

- l'émetteur envoi donc ces 7 trames (0 → 6) :
  - la fenêtre du récepteur autorise ce dernier à accepter toutes les trames numérotées 0 → 6
  - si tout c'est bien passé, le récepteur les acquitte et ajuste sa fenêtre : état (b), il accepte 7 et 0 → 5 et vide ses buffers
  - malheureusement, des problèmes de transmission engendrent la perte de tous les acquits :
    - le timer de l'émetteur expire et il renvoi la trame 0
    - elle parvient au récepteur qui l'accepte puisque son numéro appartient à sa fenêtre !!!
    - le récepteur acquitte alors la trame 6 puisque les trames 0 → 6 ont été correctement reçues

Selective repeat & taille fenêtre

- l'émetteur, apprenant que toutes les trames précédemment transmises ont été bien reçues, met sa fenêtre à jour et envoi les trames 7 et 0 → 5
- la trame 7 est acceptée par le récepteur et envoyée directement à la couche 3
- immédiatement après, la couche 2 du destinataire vérifie si elle a déjà reçu une trame correcte numérotée 0. C'est le cas, elle transmet le paquet contenu dans la trame à la couche réseau et c'est la catastrophe puisque la couche 3 reçoit un mauvais paquet

- le problème vient du fait que la nouvelle fenêtre de l'émetteur se superpose à l'ancienne. La série suivante de trames peut contenir des trames dupliquées ou de nouvelles trames

Selective repeat & taille fenêtre

- dans ce cas, le récepteur n'a aucun moyen de faire la différence

- Solution :
  - il faut s'assurer qu'il n'y aura jamais superposition de l'ancienne fenêtre à la nouvelle.
  - il suffit de choisir une taille de fenêtre au plus égale à la moitié du numéro de séquence le plus élevé, soit ***(N + 1) / 2***

HDLC

- Famille de protocoles :
  - *SDLC* : Synchronous Data Link Control (IBM)
  - *ADCCP* : Advanced Data Communication Control Procedure (AINSI)
  - *HDLC* : High-level Data Link Control (ISO)
  - *LAP* : Link Access Procedure (CCITT)
  - Tous ces protocoles sont basés sur le même moule et les mêmes idées de base. Ils sont toutefois incompatibles à cause de différences mineures

HDLC

- Fourniture d'un service :
  - fiable
  - orienté connexion
- Format des trames :

- Trois types de trames :
  - trame de données
  - trame « superviseur »
  - trame non numérotée

HDLC – trame de données

- Champ de contrôle : [0,Seq,Poll/Final,Next]
  - contient notamment :
    - ***seq*** = numéro de séquence de la trame
    - ***next*** = numéro de séquence du paquet suivant que l'émetteur de cette trame s'attend à recevoir
  - les seq et les next sont sur 3 bits
  - il existe une version étendue travaillant sur 7 bits au lieu de 3

HDLC – trame superviseur

- Champ de contrôle : [1,0,Type,Poll/Final,Next]
  - idem que la trame de données
- Cette trame ne véhicule pas de données
- Par contre :
  - ***Receive ready*** *(RR)* :
    - trame d'acquit qui indique que le receveur est toujours prêt à recevoir des données
  - ***Reject*** :
    - indication qu'une erreur à été détectée. → 'Next' indique le numéro de la première trame erronée

HDLC – trame superviseur

- ***Receive not ready*** *(RNR)* :
  - trame d'acquit qui indique que le receveur est débordé (contrôle de flux). L'émetteur doit s'arrêter jusqu'à ce que le receveur envoie une autre trame de contrôle
- ***Selective reject*** :
  - demande de retransmission

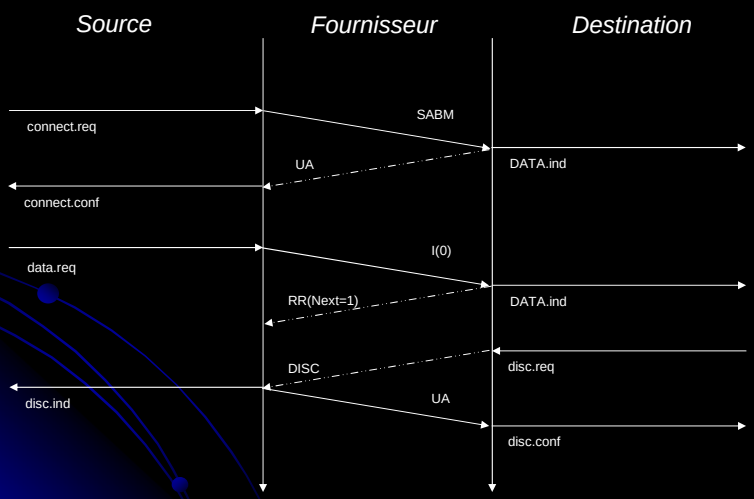
HDLC – trame non numérotée

- Champ de contrôle : [11,Type,Poll/Final,Modifier]
  - 32 types différents
- ***DISC*** :
  - indication de fin de connexion
- ***SABM*** : Set Asynchronous Balanced Mode
  - utilisé pour établir une connexion
- ***SABME*** :
  - idem, mais avec indication de la prise en charge du protocole étendu de 3 à 7 bits

HDLC – trame non numérotée

- ***FRMR*** : (Frame reject) :
  - indique qu'une trame sémantiquement incorrecte est arrivée
- ***UA*** : (Unnumbered Acknowledgment) :
  - sert à l'acquittement des trames de contrôle

HDLC – exemple



La couche Liaison de données

- Pour remplir ces tâches, la couche liaison de données se subdivise en 2 sous-couches :
  - ***MAC*** : Medium Access Control
    - prise du support afin de pouvoir en disposer, pour émettre, de manière exclusive : pourquoi ?
    - politesse
  - ***LLC*** : Logical Link Control
    - on s'assure que ce que l'on envoie est correctement reçu (ordre, erreur, flux, …)

Réseau Locaux - Ethernet

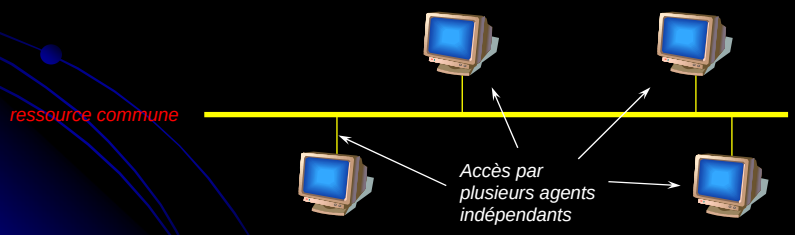
- Après avoir exposé la problématique de la couche 2, abordons en une implémentation concrète en étudiant ***Ethernet***.
- Enjeu :
  - pour connecter 2 ordinateurs, il suffit de les relier par un fil (croisé)
  - comment en connecter N sachant que chacun d'entre-eux doit pouvoir discuter avec n'importe lequel des autres ?

Réseau Locaux - Ethernet

- Tout d'abord, la méthode utilisée pour les interconnecter physiquement déterminera la ***topologie*** du réseau : (cfr. chapitre précédant).
  - bus
  - étoile, ...
- Ethernet est basé sur un principe de ***diffusion***, les topologies adéquates sont donc restreintes :
  - bus
  - anneau

Réseau Locaux - Ethernet

- Pour Ethernet, il en découle :
  - plusieurs machines (logiquement indépendantes)
  - une ressource commune : le médium de transport



Accès au support

- Dans le cas d'une ressource partagée, garantir l'accès au support est capital.
- Cet accès doit, entre autre, être :
  - ***possible*** : pas d'attente infinie
  - ***équitable*** : chacun doit avoir la même chance d'y accéder
  - ***exclusif*** : celui qui à l'accès au moment t est le seul à pouvoir parler

Allocation statique

- N ordinateurs cherchent à accéder à un réseau de capacité W bits/sec. Solution :
  - ***réserver*** une capacité de W/N bits/s pour chaque machine
  - ***AMRF*** : Accès Multiple par Répartition de Fréquences
    - basé sur le multiplexage en fréquences
    - division de la bande passante totale en autant de canaux que d'utilisateurs
    - le débit dépend donc fortement du nombre d'utilisateurs
    - pas adapté au LAN
- utiliser ***TDM*** pour réguler la transmission :
  - émission lors du time slot

Allocation statique

- conséquences :
  - chaque machine reçoit W/N du débit total
  - peu satisfaisant :
    - car dans un réseau d'ordinateurs :
      - le trafic est souvent en rafales et peu de machines émettent en même temps
      - cela génère une piètre utilisation du canal et des délais de transmissions plus importants

Méthode d'accès

- Hypothèses de base :
  - *N ordinateurs* veulent accéder à la même ressource
  - il n'y a qu'un seul canal disponible
  - *collision* :
    - si deux (ou plus) machines transmettent en même temps, une collision se produit
  - *transmission* des trames. Soit :
    - n'importe quand
    - à des moments prédéfinis
  - *écoute* du canal :
    - possible
    - impossible

Méthode d'accès

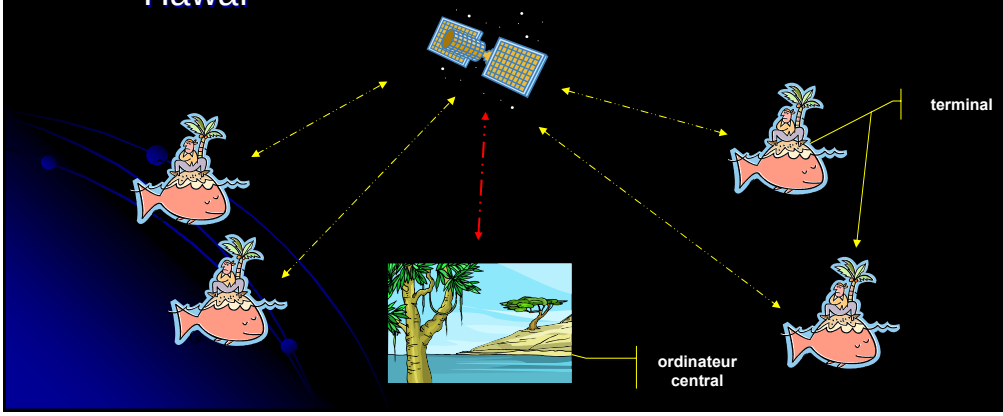
- Comment réguler l'accès au canal unique ?
- Il existe deux catégories de solutions :
  - *solutions statistiques ou optimistes* :
    - on laisse les machines transmettrent plus ou moins à leur guise en « espérant » qu'une seule émettra à la fois
    - utilisation d'un algorithme distribué pour résoudre le problème de la collision

Méthode d'accès

- *solutions déterministes ou pessimistes* :
  - on veut éviter à tout prix que deux ordinateurs ne transmettent en même temps
  - un seul ordinateur peut transmettre à la fois
  - algorithme distribué pour accorder les autorisations de transmettre

ALOHA

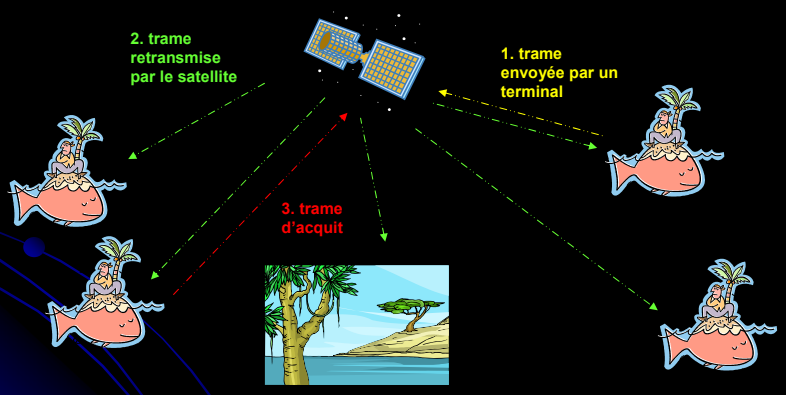
- Problème à résoudre :
  - accès à des terminaux distants sur les îles de Hawaï



ALOHA

- *Solution* :
  - un satellite couvre l'archipel
  - une antenne est implantée sur chaque île
  - pour transmettre, on envoi au satellite, qui retransmet vers toutes les îles
- *Mais* :
  - l'accès est non règlementé : une station émet dès qu'elle a des données à transmettre
  - il y a un nombre élevé de collisions
  - le délai d'attente est aléatoire avant une retransmission
  - l'efficacité est faible surtout en cas de charge du réseau

ALOHA

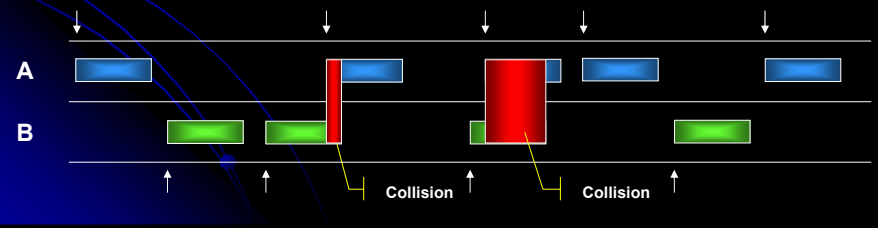


ALOHA

- Transmission des données :
  - un terminal peut vérifier que sa donnée a bien été transmise en *écoutant* le canal satellite
  - la réception correcte d'une trame est confirmée via une *trame ACK* spéciale :
    - durée de temps réservée pour la trame d'ack afin d'éviter les collisions

ALOHA

- Comment réguler l'accès au satellite ?
  - si un seul terminal transmet, il entendra sa propre transmission via le satellite → OK
  - si deux (ou plus) terminaux transmettent simultanément, leur signal combiné sera indécodable et ils devront retransmettre → KO



ALOHA

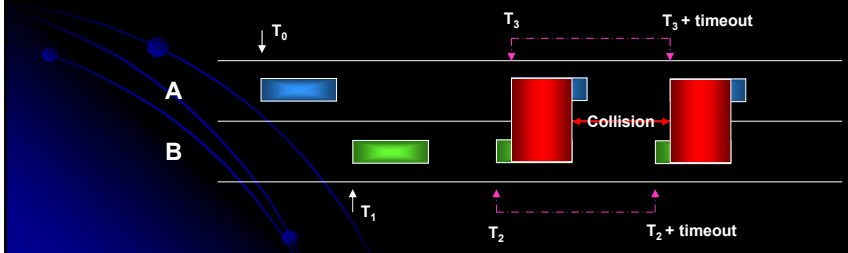
- Algorithme d'accès au canal
  - première solution

```
N=1;
while ( N<= max) do
send frame;
wait for ack on return channel or
timeout:
if ack on return channel
exit while;
else
/* timeout */
/* retransmission is needed */
N=N+1;
end do
/* too many attempts */
```

  - interprétation ...

ALOHA

- Inconvénients du système présenté :
  - deux stations en collision risquent d'y rester indéfiniment et le système est bloquant
  - pourquoi ?



ALOHA

- Comment éviter cette synchronisation gênante entre machines ?
  - nouvel algorithme d'accès au canal

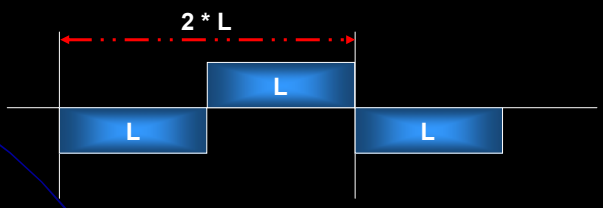
```
N=1;
while ( N<= max) do
send frame;
wait for ack on return channel or timeout:
if ack on return channel
exit while;
else
/* timeout */
/* retransmission is needed */
wait for random time;
N=N+1;
end do
/* too many attempts */
```

ALOHA

- Performance d'ALOHA :
  - en pratique, utilisation d'un peu moins de ***20%*** du canal
  - fonctionne ***correctement à faible charge***
  - mais, à ***forte charge***, le système devient ***instable*** :
    - les trames qui ont subi une collision sont retransmises et ces retransmissions peuvent, à leur tour, subir une collision
    - ce nombre va en augmentant

ALOHA

- Pour qu'une transmission soit réussie :
  - il faut qu'aucune trame ne soit transmise par une autre station avant et pendant la trame courante

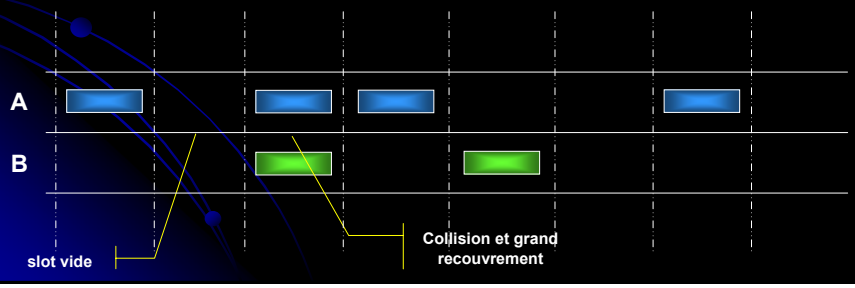


Slotted ALOHA

- Comment améliorer ALOHA, c'est-à-dire, comment améliorer la gestion d'accès au support ?
  - idée :
    - diviser le temps en slot et ne permettre à une station de transmettre qu'au début de chaque slot :
      - nécessite un signal d'horloge externe pour indiquer le début des slots (cfr. chapitre 2)
    - → réduction des risques de collision grâce à la disparition des trames partiellement en collision

Slotted ALOHA

- En slotted ALOHA, un slot est :
  - ***vide*** (aucune trame transmise)
  - ***occupé*** (une seule trame transmise)
  - ***en collision*** (plusieurs trames transmises, mais recouvrement quasi complet)



Slotted ALOHA

- Performances de slotted ALOHA :
  - sous les mêmes conditions que ALOHA :
    - l'utilisation du canal passe à ***37%*** :
      - 37% de slots vides
      - 37% de slots occupés
      - 26% de collisions
  - le gain principal réside dans le temps épargné sur les trames en recouvrement partiel
  - de plus, ce système tient mieux la charge que ALOHA. Toutefois, il est encore loin de la perfection

Slotted ALOHA

- L'algorithme d'accès au support pourrait être :

```
N=1;
while ( N<= max) do
wait until start of slot;
send frame;
wait for data on return channel or timeout:
if data on return channel
exit while;
else
/* timeout */
/* retransmission is needed */
wait for random number of slots;
N=N+1;
end do
/* too many attempts */
```

CSMA – Carrier Sense Multiple Access

- Comment améliorer slotted ALOHA ?
- Idée :
  - pour améliorer l'utilisation du canal, on pourrait apprendre la ***politesse*** aux ordinateurs …
  - ***politesse*** :
    - écouter le canal avant de commencer à transmettre
    - ne pas transmettre si quelqu'un transmet déjà
    - → on ne fait donc plus « ce que l'on veut »

CSMA

- Attention :
  - un tel système n'est utilisable que si ***tous*** les ordinateurs peuvent ***écouter les transmissions*** de tout le monde
  - il est donc possible sur un réseau local
  - mais n'est, en général, pas possible sur des réseaux radio ou satellite

CSMA

- Dans le cas de CSMA, l'algorithme de transmission pourrait être :

```
N=1;
while ( N<= max) do
wait until channel becomes free;
send frame immediately;
wait for ack or timeout:
if ack received
exit while;
else
/* timeout */
/* retransmission is needed */
N=N+1;
end do
/* too many attempts */
```

CSMA – non persistant

- Pour améliorer encore CSMA, 3 variantes ont été imaginées :
  - en plus de la politesse, nous pourrions leur ***apprendre la patience*** :
    - en effet, transmettre une trame dès la fin de la trame précédente est un comportement assez agressif
    - si le canal est libre, transmettre mais sinon, *attendre un temps aléatoire* avant de le réécouter
    - un certain nombre de phénomènes de contention sont ainsi évités

CSMA – non persistant

- L'algorithme ressemble à ceci :

```
N=1;
while ( N<= max) do
   listen channel;
   if channel is empty
      send frame;
      wait for ack or timeout
      if ack received
         exit while;
      else /* retransmission is needed */
         N=N+1:
   else
      wait for random time;
end do
```

CSMA – p-persistent

- Cette variante est à mi-chemin entre CSMA et CSMA non persistent :
  - le délai d'attente devient une probabilité d'émettre

```
N=1;
while ( N<= max) do
   listen channel;
   if channel is empty
      with probability p
         send frame;
         wait for ack or timeout
         if ack received
            exit while;
         else /* retransmission needed */
            N=N+1;
   else
      wait for random time;
end do
```

Améliorations à CSMA

- Avec CSMA dans un réseau en bus, lorsqu'une collision se produit, celle-ci affecte la trame dans son intégralité :
  - dès qu'une trame est en ***collision***, elle est inutilisable, son information est *altérée*
  - pourquoi ne pas ***arrêter la transmission*** ? Cela nous ferait gagner un débit utile

Améliorations à CSMA

- Comment ***détecter les collisions*** ?
  - si une station écoute pendant qu'elle émet :
    - en **l'absence de collision**, elle entendra le signal qu'elle a émit à l'identique
    - en **cas de collision**, le signal qui lui reviendra sera perturbé
  - ***CSMA/CD*** :
    - Carrier Sense Multiple Access with Collision Detection
    - pas besoin de ACK puisque la station écoute le signal

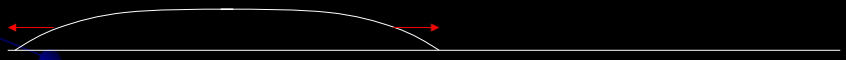
CSMA/CD

- L'algorithme pourrait ressembler à ceci :

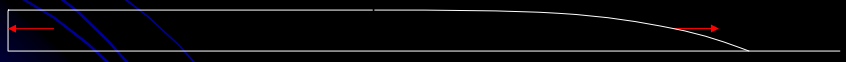
```
N=1;
while ( N<= max) do
   wait until channel becomes free;
   send frame and listen;
   wait until (end of frame) or (collision)
   if collision detected
      stop transmitting;
      /* after a special jam signal */
   else
      /* no collision detected */
      wait for interframe delay;
      exit while;
   N=N+1;
end do
/* too many attempts */
```

CSMA/CD - exemple

1. Début de l'envoi de la trame



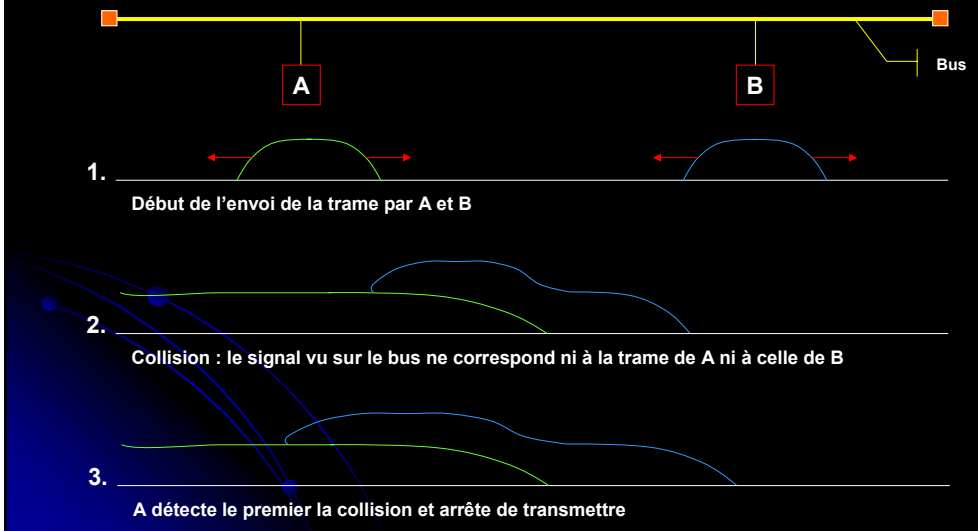
2. La trame se propage dans les 2 sens [vitesse : environ 5 µsec/km]



3. A gauche, la trame atteint le bout du bus et à droite elle parvient à la destination B

4. La trame quitte progressivement le bus

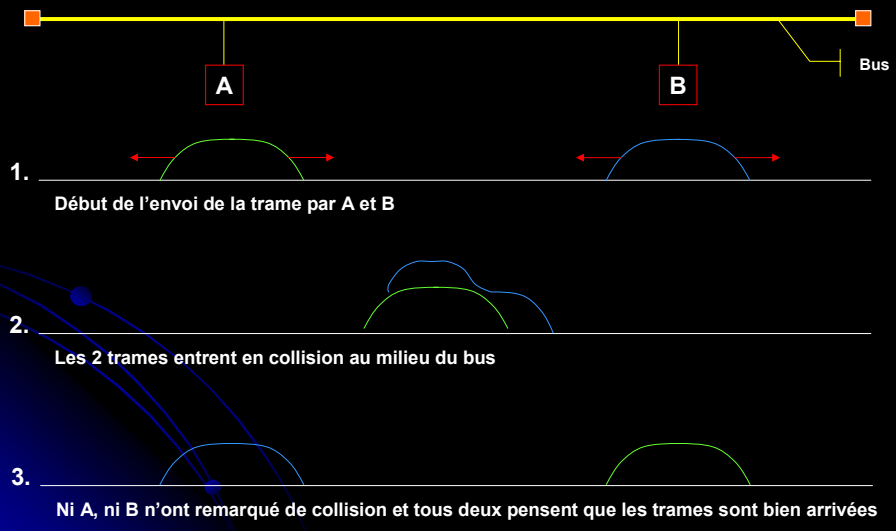
CSMA/CD - collisions



CSMA/CD - collisions

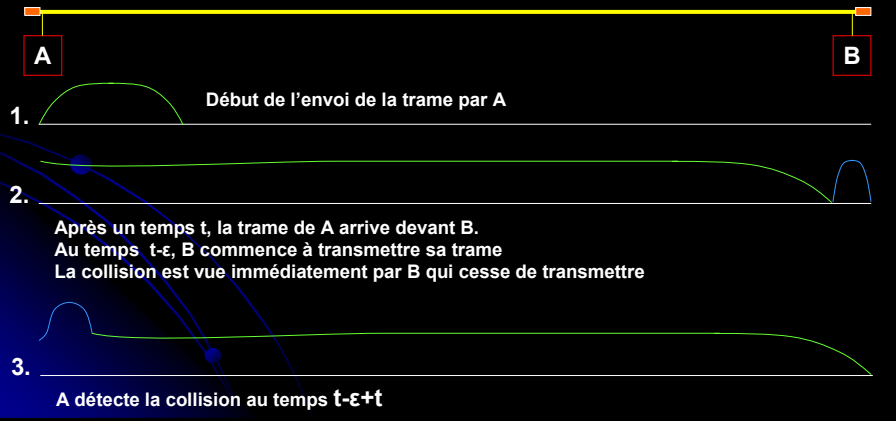
- Avantages de la détection de collisions :
  - ***améliore le débit*** utile puisqu'on arrête de transmettre des trames corrompues
  - une station peut se rendre compte que la trame qu'elle envoi n'arrive pas à destination :
    - elle peut donc faire de la ***retransmission automatique*** et donc cela réduit le besoin de mécanismes de retransmission complexes
  - Question :
    - une station peut-elle détecter toutes les collisions affectant ses trames ?

CSMA/CD - collisions



CSMA/CD - collisions

- Comment être certain de détecter les collisions ?
  - cas le plus défavorable : extrémités du bus



CSMA/CD - collisions

- Condition pour qu'une station puisse détecter toute collision qui affecte ses trames :
  - la trame doit être transmise pendant un ***temps*** au moins ***aussi long*** que le délai ***aller-retour*** (2 * t) sur le bus :
    - il en découle que si le débit du bus et le délai aller-retour maximum sont fixés alors la trame doit avoir une taille minimale
  - amélioration :
    - pour s'assurer que les collisions sont toujours détectées par l'émetteur, une station qui détecte une collision émet un signal de brouillage : ***jamming***

Exponential backoff

- CSMA/CD : ***que faire*** en cas de ***collision*** ?
  - si les deux stations responsables de la collision retransmettent immédiatement, on a une nouvelle collision
- Solution :
  - attendre un ***temps aléatoire*** après la collision :
    - aléatoire :
      - après une collision, le temps est découpé en slots (= paquet de taille minimale) :
        - après la première collision, attendre aléatoirement 0 ou 1 slot avant de retransmettre
        - après la deuxième collision, attendre aléatoirement 0, 1, 2 ou3 slots avant une réémission
        - après la $i^{ème}$ itération : délai aléatoire de 0 … 2^i-1

CSMA/CD with exponential backoff

- L'algorithme devient alors :

```
N=1;
while ( N<= max) do
   wait until channel becomes free;
   send frame and listen;
   wait until (end of frame) or (collision)
   if collision detected
      stop transmitting;
      /* after a special jam signal */
      k = min (10, N);
      r = random(0, 2k – 1) * slotTime;
      wait for r time slots;
   else
      /* no collision detected */
      wait for interframe delay;
      exit while;
   N=N+1;
end do
/* too many attempts */
```

Ethernet 802.3

- Initialement proposé par ***Digital, Intel et Xerox*** puis standardisé par IEEE et ISO
- Utilise ***CSMA/CD avec exponential backoff***
- Caractéristiques :
  - ***débits*** : 10, 100 Mb/s, 1 Gb/s (10 Gb/s proche)
  - ***câblage*** :
    - initialement coaxial → 10Base2
    - paire torsadée → 10BaseT ou 100BaseTX, …
  - ***délai aller-retour maximum*** du bus : 51,2 µsec :
    - imposé par la norme Ethernet 802.3

Ethernet 802.3

- De ce délai aller-retour imposé par la norme et selon le débit de notre LAN, nous pouvons en ***déterminer*** :
  - la longueur du bus
  - la longueur maximal des trames
- ***Hypothèses*** :
  - débit : 10 Mb/s
  - support : filaire :
    - vitesse propagation électricité vaut donc 270.000 km/s
  - vitesse = distance / temps

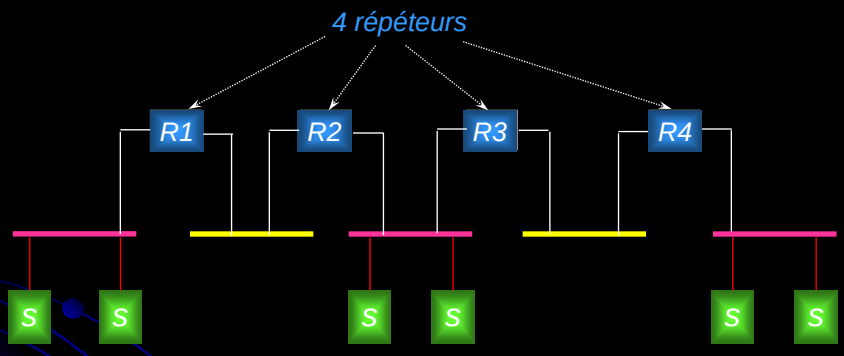
Longueur des trames

- Taille minimale d'une trame:
  - $d = t * v \rightarrow d = 51{,}2 * 10^{-6} * 10 * 10^{6}$
    $= 512$ bits = ***64 octets***
  - grandeur d'une trame : minimum 64 octets
- Taille maximale d'une trame :
  - cette longueur a été normalisée. La normalisation l'impose à 1.518 bits

Longueur du bus

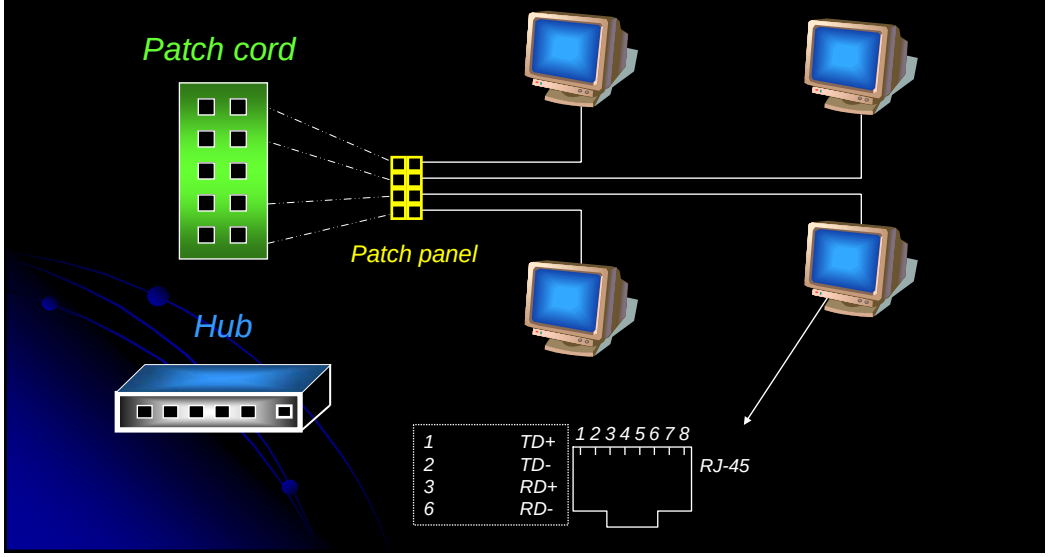
- La longueur maximale d'un LAN Ethernet est également directement fonction de son débit et de la taille minimale des trames :
  - avec un aller retour en 51,2 µsec
  - si le débit est de 10 Mb/s, alors la longueur vaut :
    - $10 * 10^6$ m/s * 51,2 * 10^{-6} s = 512 mètres = ***500 mètres*** par brin
  - il est possible de combiner plusieurs brins selon le principe suivant :

Règles des répéteurs 5-4-3-2-1

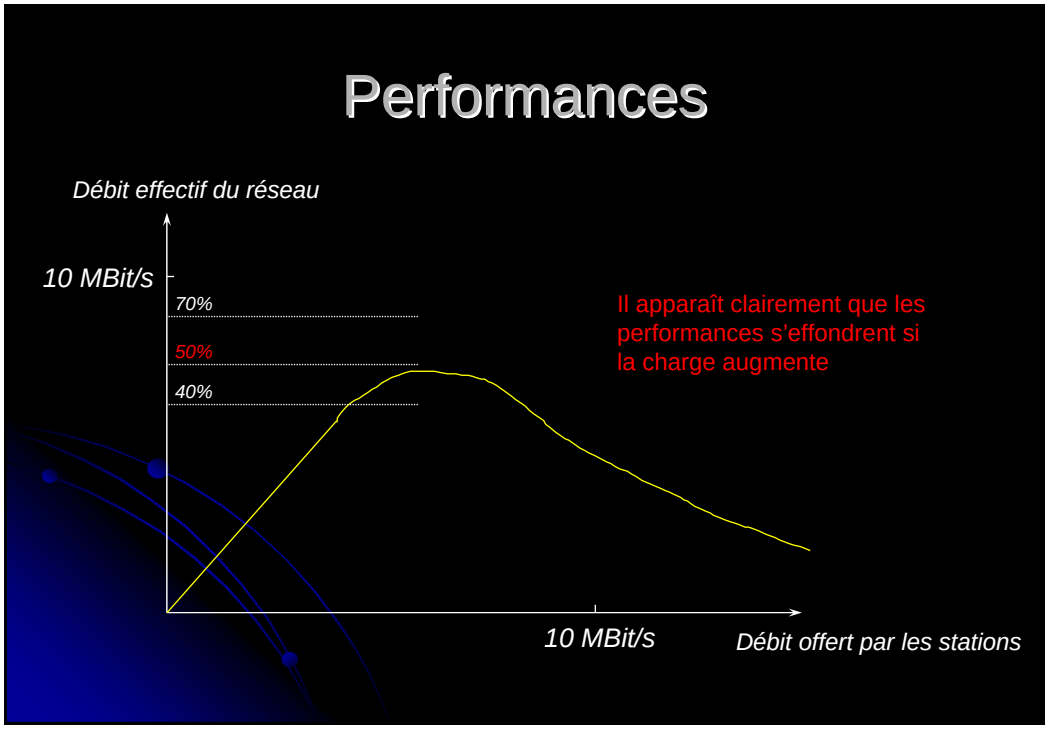


- 5 segments
- 4 répéteurs
- 3 segments avec stations
- 2 segments de liens
- 1 domaine de collision

Câblage courant (10BaseT)



Performances



Ethernet 802.3

- La topologie typique d'Ethernet :
  - est donc un réseau en bus sur câble coaxial
  - est donc un réseau (forme d'étoile) en micro-bus sur paires torsadées

Ethernet 802.3

- *Problèmes* restant à résoudre :
  - comment transmettre une trame de A vers B ?
    - comment B détecte qu'une trame lui est destinée ?
  - Comment transmettre une trame à toutes les stations connectées au réseau ?
  - Comment transmettre une trame à un groupe de stations connectées au réseau ?
- Cette problématique est résolue par la technique de ***l'adressage***

Adresses Ethernet

- Sur un bus, toutes les stations voient ***toutes les trames***
- Comment une station peut-elle savoir qu'une trame lui est ***destinée*** ?
  - chaque trame contient :
    - l'adresse du destinataire
    - l'adresse de l'émetteur

Adresses Ethernet

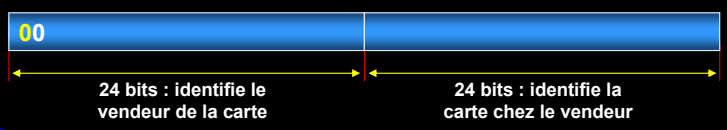
- Comment choisir l'adresse d'une station ?
  - par ***configuration manuelle*** :
    - adresses locales
  - par ***construction*** :
    - adresse encodée en ROM par le fabriquant de la carte
    - chaque constructeur possède un pool d'adresse MAC qu'il affecte à ses cartes
    - méthode la plus fréquente

Adresse Ethernet

- Chaque carte Ethernet contient une adresse unique :
  - cela permet de garantir que sur n'importe quel réseau, deux stations n'auront pas la même adresse

Adresses Ethernet

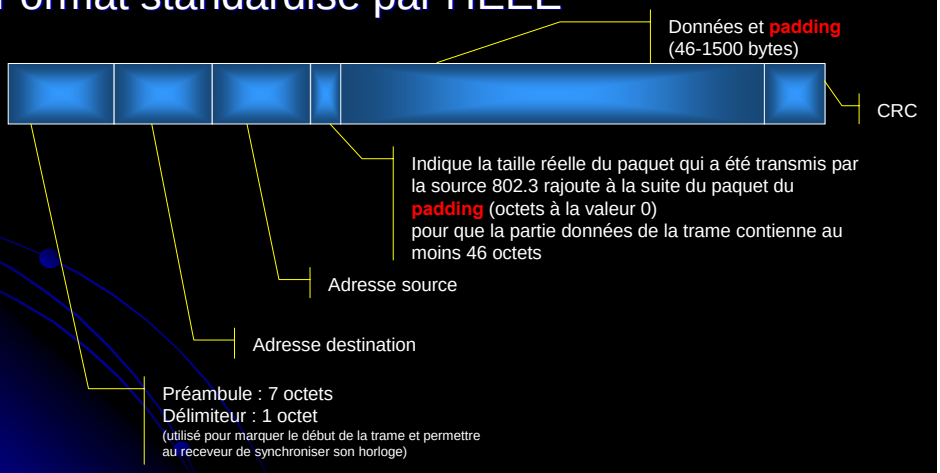
- Format des adresses (MAC) :
  - encodées sur 48 bits :
    - adresse source :



    - adresse destination :
      - si bit de poids fort = 0 → adresse d'un *station*
      - si bit de poids fort = 1 → adresse *multicast*
      - adresse destination = 111111..111 → *broadcast*

Format de la trame Ethernet 802.3

- Format standardisé par l'IEEE



Format de la trame Ethernet 802.3

- Il faut noter qu'en 1997, une modification à 802.3 a été introduite
- Dans cette nouvelle version de la norme, la sémantique du champ Longueur a été modifiée :
  - ce champ peut maintenant indiquer soit la longueur de la trame (valeurs inférieures à 1518) ou une indication de type comme dans Ethernet DIX (trame originelle)
  - cette double sémantique est possible car le champ type prend toujours des valeurs supérieures à 1518

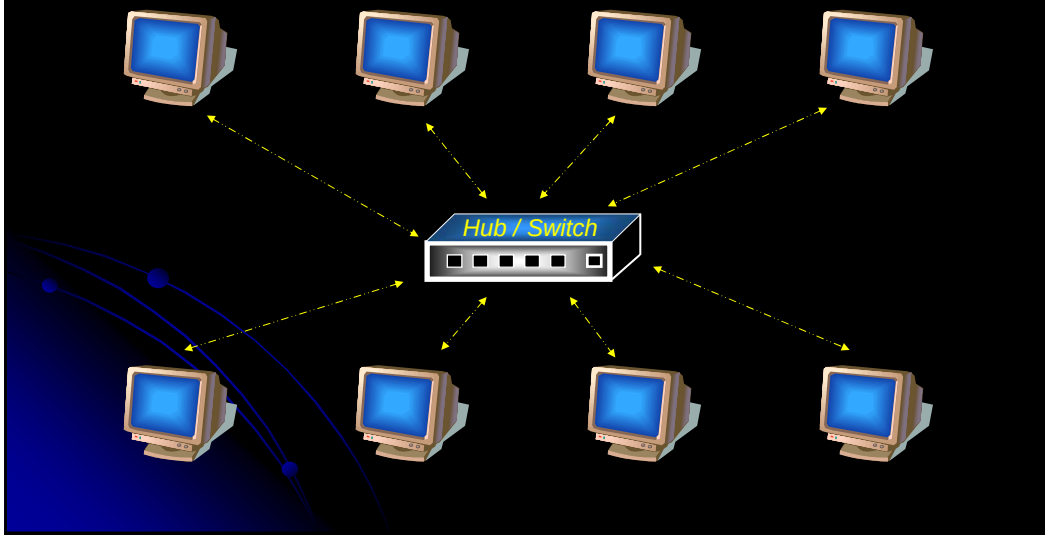
Service Ethernet

- Le service fourni par un réseau local Ethernet est un service :
  - sans connexion, sans acquit
  - *unacknowledged connectionless*
- Même si le service Ethernet est en théorie non fiable, un bon Ethernet doit :
  - délivrer les trames à destination avec une très haute probabilité
  - ne pas modifier l'ordre des trames transmises

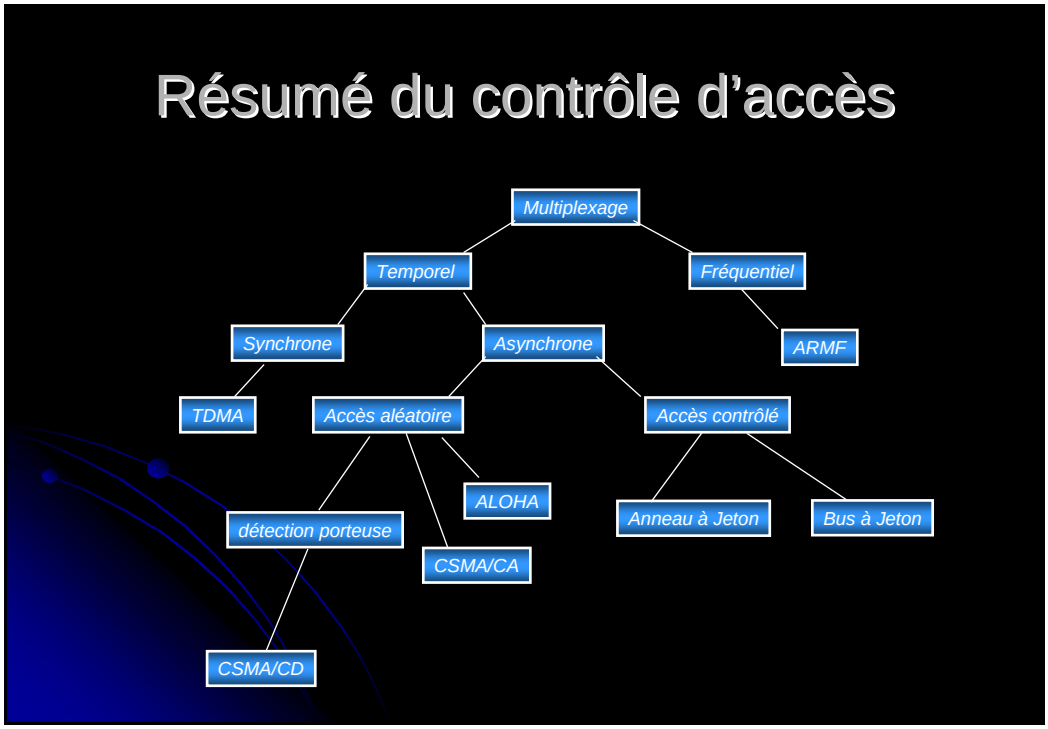
Ethernet aujourd'hui

- Le câblage est essentiellement :
  - ***10/100Base-T*** : paire torsadée (UTP-3 ou UTP-5) sur une distance de 100 mètres
  - ***10/100Base-F*** : fibre optique sur une distance de 2.000 mètres
- Chaque ordinateur est ***connecté à un Hub/Switch*** qui fait office de bus concentré en un point

Ethernet aujourd'hui



Résumé du contrôle d'accès



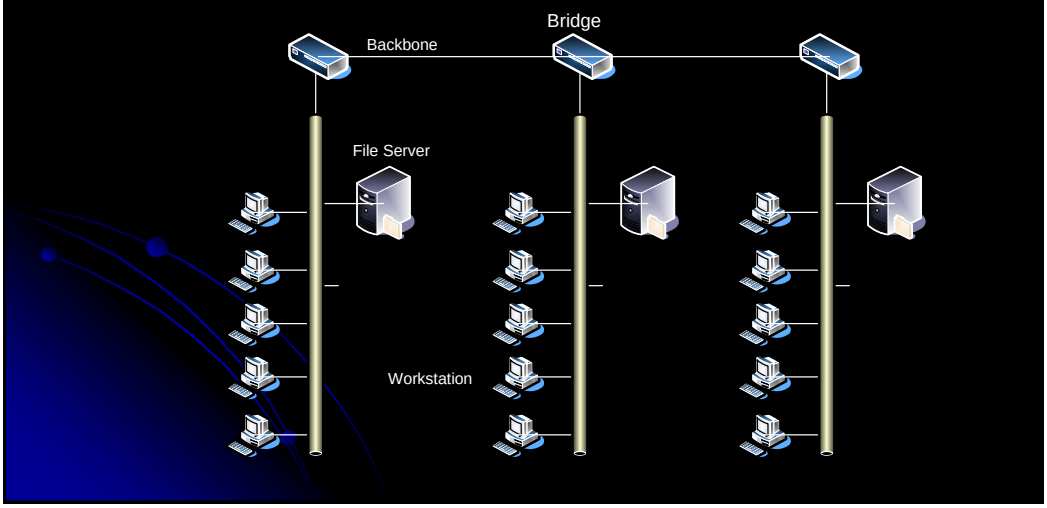
Pont transparent

- Examen des adresses de cette couche pour transmission
- Fonctionne pour tous protocoles supérieurs car il ne regarde pas les données (IPv4, IPv6, AppleTalk, …)
- Fonctionnement en mode transparent

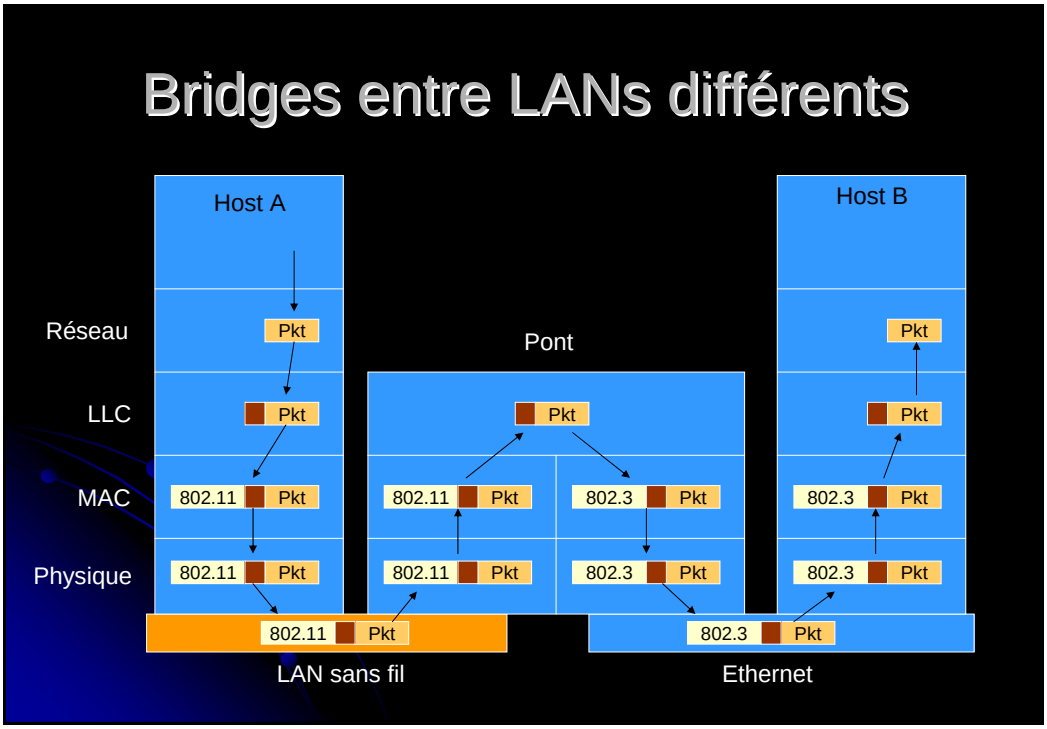
Interconnexion de LAN

- Interconnexion de LAN par des ponts (*bridges*)
- 6 Motivations :
  - autonomie départementale et choix de LAN
  - économie d'interconnexion de LAN distants
  - division logique de LAN en sous parties interconnectées par un backbone
  - distance physique entre les machines les plus éloignées trop importante
  - éviter des problèmes si un nœud génère des erreurs
  - éviter le sniffing du réseau global

Interconnexion de LAN

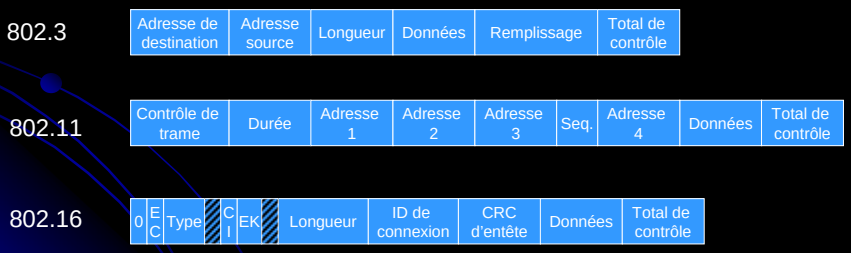


Bridges entre LANs différents



Bridges entre LANs différents

- L'interconnexion de LAN différents est assez complexe car :
  - chaque LAN utilise un ***format de trame différent***
    D'où obligation de reformater les trames



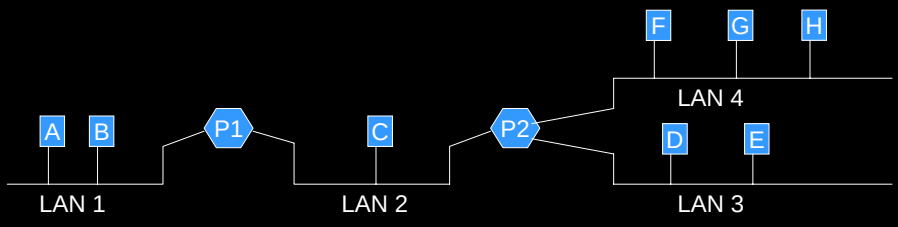
Bridges entre LANs différents

- L'interconnexion de LAN différents est assez complexe car :
  - le débit binaire n'est pas le même entre les LANs différents (1GB/s → wireless 11MB/s). D'où obligation de gérer une mémoire tampon au niveau du bridge.
  - les longueurs de trames ne sont pas les mêmes
    Que faire d'une trame trop grande ? ***Solution*** : on ne la fait pas passer. Impossible de refragmenter car pas de réassemblage prévu
  - certains protocoles (802.11, 802.16) supportent le chiffrement des trames. Quid des autres ?
  - la qualité de service est assurée dans certains protocoles (802.11, 802.16), mais pas dans tous ***HDLC ↔ Ethernet***

Interconnexion locale

- On doit pouvoir interconnecter des LANs différents sans aucune modification logicielle ou matérielle de l'existant
- Fonctionnement du pont en mode transparent

Interconnexion locale



Lorsqu'un pont reçoit une trame, il peut :
- la *supprimer*
- la *transmettre*
  - choix du LAN de destination dans une table d'adresses
  - pas de notion de routage, seul le LAN directement connecté est concerné

Interconnexion locale

- ***Table des adresses*** MAC dans chaque Bridge (pont)
- Algorithme d'apprentissage à postériori (backward learning) par ***innondation***
  - chaque trame reçue dont la destination est inconnue est envoyée sur tous les LANs sauf celui d'où elle vient
  - petit à petit, apprentissage de toutes les destinations en examinant l'adresse source des trames qui passent
  - une trame dont la destination est connue ne sera transmise que sur le LAN approprié

Interconnexion locale

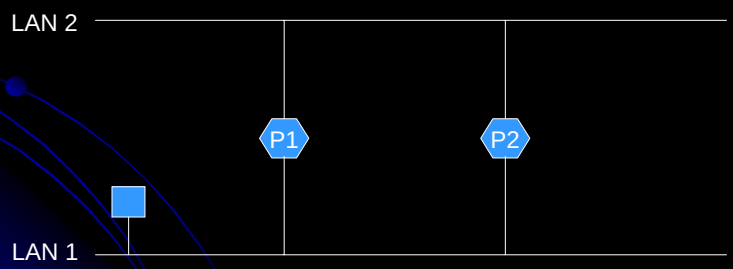
- La topologie peut évoluer :
  - changement de place d'une station
  - pont allumé ou éteint
  - …
- A chaque entrée dans la table d'adresses, on associe l'heure d'arrivée de la dernière trame en provenance de la station correspondante
- Un processus dans le bridge efface les entrées de la table dont le timestamp dépasse quelques minutes. Il y a donc une ***purge régulière***.

Interconnexion locale

- Ce qu'un pont ***doit faire d'une trame*** entrante :
  - si les LAN source et destination sont les mêmes, il rejette la trame
  - si les LAN source et destination sont différents, il transmet la trame à ce seul LAN
  - si le LAN de destination est inconnu, il diffuse la trame par inondation

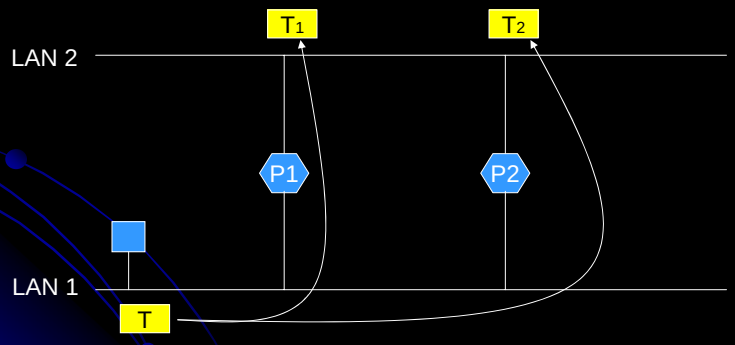
Spanning Tree

- Pour augmenter la fiabilité, on peut utiliser deux ponts entre deux LANs différents (redondance)



Spanning Tree

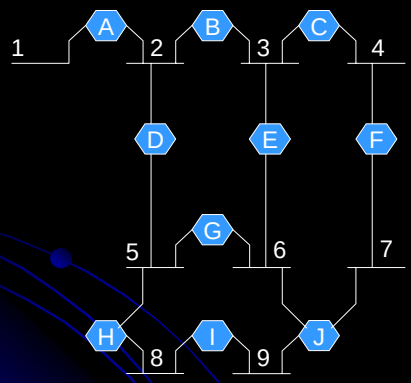
- Toutefois, ceci peut poser des problèmes…



Spanning Tree

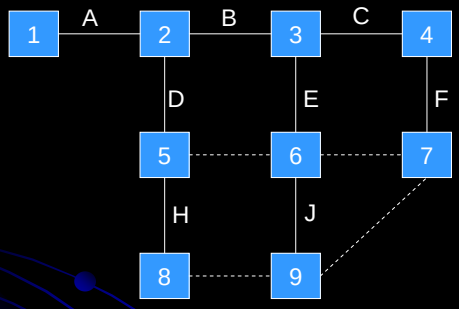
- La solution :
  - tous les bridges communiquent entre eux :
    - au moyen de trames spécifiques : ***messages 802.1***
  - on crée un ***arbre de recouvrement*** sur tous les ponts
  - certains ***liens*** entre ponts sont ***ignorés*** afin d'établir une topologie exempte de boucle

Spanning Tree



- Illustration d'un réseau de 9 LANs interconnectés par 8 ponts
- On peut représenter un LAN par un nœud et un pont par un lien (voir schéma suivant)

Spanning Tree



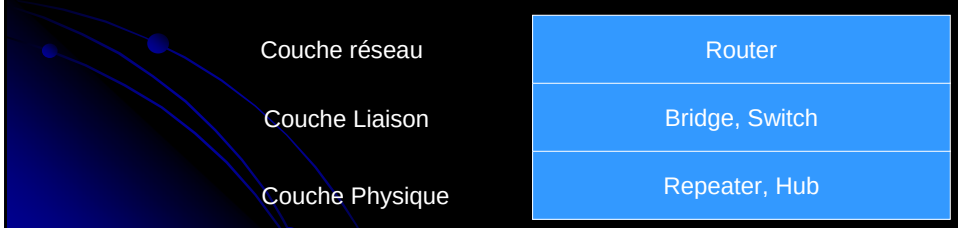
- Les liens en continu représentent les ponts faisant partie de l'arbre
- Les liens en pointillé représentent les ponts ne faisant pas partie de l'arbre
- Il existe un seul chemin possible entre deux LAN sans aucun risque de boucle

Spanning Tree

- Création du spanning tree en établissant une communication entre les ponts
- ***Première chose*** : déterminer le pont qui va jouer le rôle de ***racine***. C'est le pont qui a le plus petit numéro de série (unique et donné par le constructeur)
- Pour chaque pont, on détermine le ***chemin le plus court*** entre le pont et la racine : l'arbre recouvrant :
  - au moyen de trames spéciales 802.1 : ***racine,émetteur,coût***
- Le résultat de cet algorithme est l'établissement d'un chemin unique vers la racine et donc vers tous les LANs
- L'arbre englobe tous les LANs, mais pas nécessairement tous les ponts (évite les circuits).
- Une fois l'arbre établi, l'algorithme demeure ***actif*** pour détecter les changements de topologie

Hub, Repeater, Bridge, Switch

- Ces équipements n'agissent pas tous au même niveau :



Hub, Repeater, Bridge, Switch

- Au ***niveau physique*** :
  - Les ***repeaters*** sont des équipements analogiques qui amplifient le signal d'un segment à un autre
  - Les repeaters ne comprennent que les volts passant sur le câble
  - Les ***hubs*** n'amplifient (habituellement) pas le signal
  - Les hubs disposent de nombreuses lignes en entrée, jointes électriquement. Les trames qui arrivent sur une ligne sont envoyées sur toutes les autres lignes.
  - Le hub représente un domaine de collision :
    - micro-bus
  - Le hub n'examine pas la trame 802

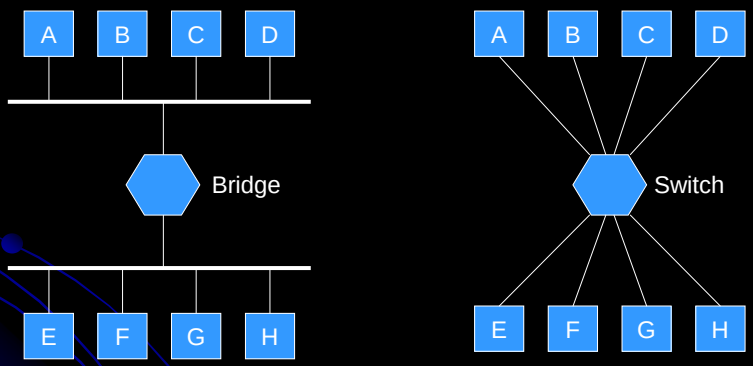
Hub, Repeater, Bridge, Switch

- Au ***niveau liaison*** : le bridge
  - Le ***bridge*** opère au niveau de la couche liaison de données
  - Le bridge sert à interconnecter 2 ou plusieurs LANs.
  - Sur le bridge, chaque ligne (port) possède les caractéristiques du LAN concerné (type, vitesse, etc.)
  - Une ligne est un domaine de collision

Hub, Repeater, Bridge, Switch

- Au niveau liaison : le switch (commutateur)
  - Le fonctionnement est aussi basé sur la couche liaison de données
  - Le ***switch*** sert à connecter des équipement terminaux (ordinateurs, imprimantes, etc.). Il lui faut donc beaucoup de ports
  - Il existe une mémoire tampon sur chaque port pour supporter les différences de vitesse de transmission des différentes lignes (conversion de trames possible)
  - Le domaine de collision est individuel à chaque ligne
  - Risque de saturation des tampons si les trames arrivent trop rapidement.

Hub, Repeater, Bridge, Switch



Hub, Repeater, Bridge, Switch

- Plusieurs transmissions possibles port à port en parallèle (bande passante totale > hub)
- Mécanismes de réémission
  - ***Store and Forward*** : on stocke complètement une trame avant de la réémettre
    Plus lent, mais plus sûr car on peut détecter des erreurs de transmission. Si mauvais réseau
  - ***Cut-Trought*** : la trame est réémise dès que l'adresse MAC destination est connue
    Plus rapide, mais toutes les erreurs de transmission sont réémises. Si bon réseau

VLANs

- Au départ, les LANs étaient représentés physiquement par un grand nombre de câbles
  L'appartenance d'une station à un LAN dépendait essentiellement de sa ***localisation géographique*** : tous les hosts du 3ième étage appartenaient au LAN du 3ième étage

VLAN

- Début des années 90, l'apparition du ***10baseT*** demande un ***nouveau câblage***.
  Ce dernier est réalisé en étoile (un câble TP vers chaque host). On introduit la notion de Patch Panel
- Les connections sont réalisées au moyen de Hubs placés près des Patch Panels.
- ***Possibilité de séparer*** logiquement des parties de réseaux présentes dans le ***même espace physique***
  Pour avoir k LAN, il faut k Hubs différents

VLAN

- Bien que ces différents LAN seront interconnectés, il est utile de séparer les LAN physiquement car :
  - on ***évite*** d'avoir un trop ***grand domaine de collision*** où l'on peut capturer les paquets (sniffer)
  - on ***diminue la charge globale***, et on évite qu'un grand consommateur de ressources ne perturbe un autre groupe d'utilisateurs
  - on ***limite le domaine de broadcast***. Il est possible qu'une station défaillante émette un « broadcast storm »

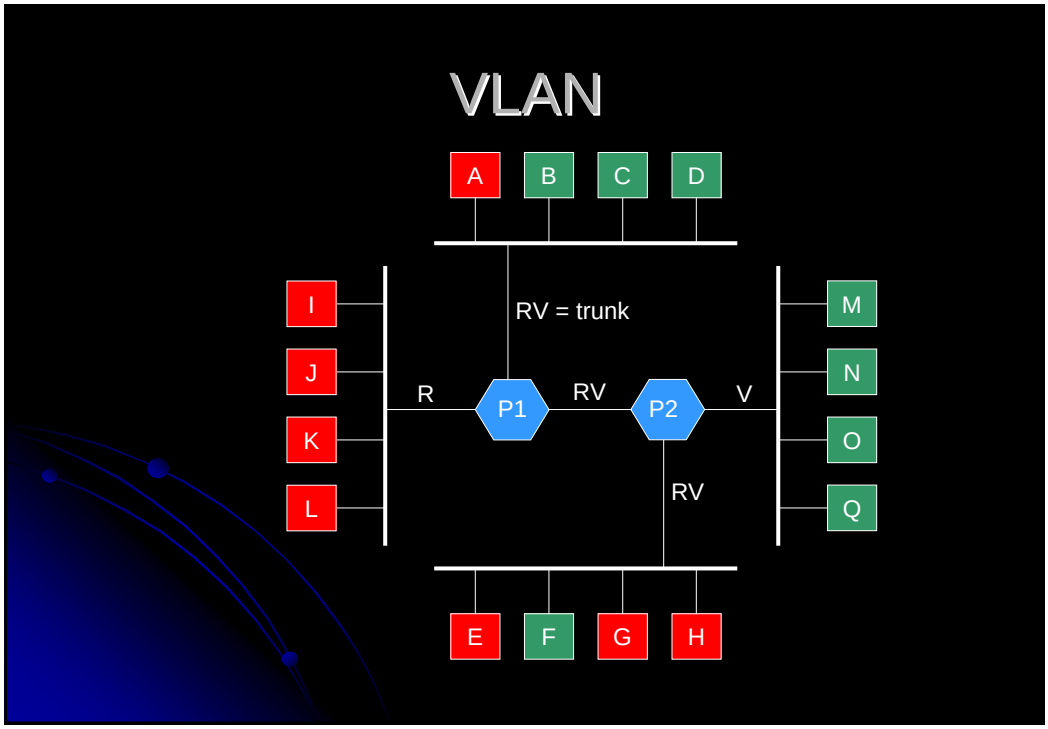
VLAN

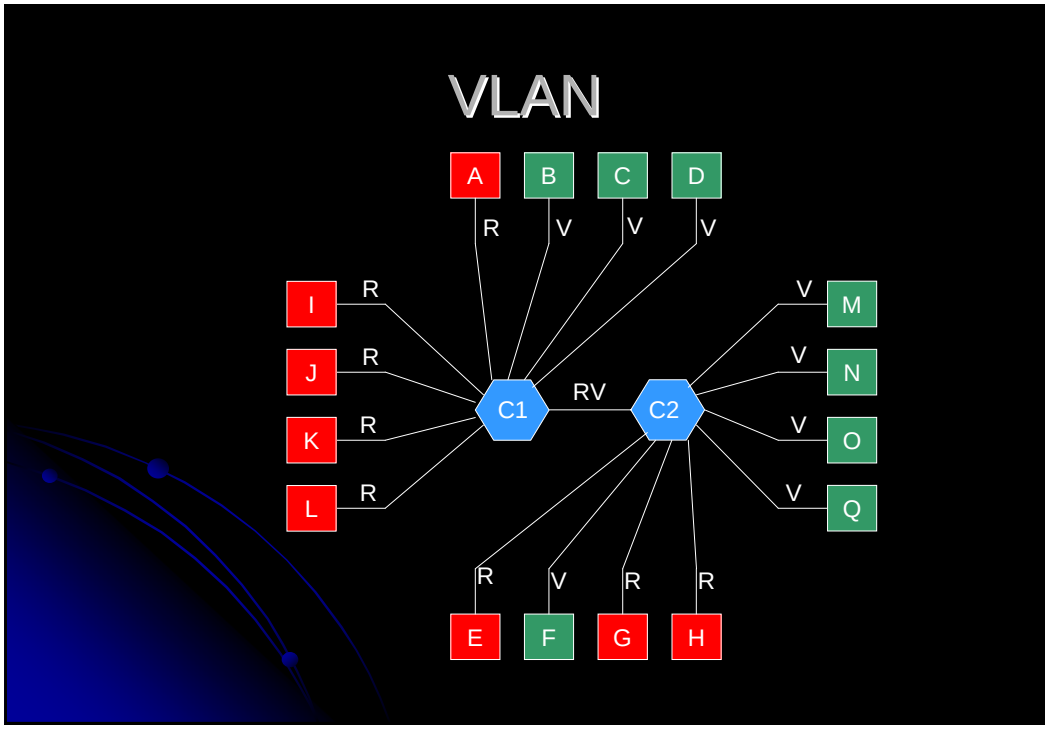
- Si un utilisateur change de bureau ou de service, l'administrateur réseau doit modifier le câblage de la connexion de cet utilisateur (Patch Panel).
  Si ces changements sont fréquents, cela peut prendre beaucoup de temps.
  De plus, il n'est pas impossible que l'utilisateur ne puisse plus être connecté au nouveau LAN (manque de prises, distance trop longue, …)

  → D'où nécessité de faire le même travail dans un environnement logiciel et non hardware : ***le VLAN***

VLAN

- VLAN signifie « Virtual LAN »
- Le VLAN est conçu pour fonctionner sur des switchs spéciaux
- Pour mettre en place un réseau basé sur les VLANs, l'administrateur doit :
  - Définir le nombre de VLAN
  - Lister les hosts appartenant à tel ou tel VLAN
  - Donner un nom à chaque VLAN (souvent une couleur)


VLAN
A
B
C
D
I
J
K
L
RV = trunk
R
P1
RV
P2
V
RV
M
N
O
Q
E
F
G
H


VLAN
A
B
C
D
R
V
V
V
I
J
K
L
R
R
R
R
C1
RV
C2
V
V
V
V
M
N
O
Q
R
V
R
R
E
F
G
H

VLAN

- Dans chaque switch, il existe une table de configuration des VLANs.
- Cette table indique quel VLAN est accessible depuis quel port
- Tout trafic émis dans un VLAN reste dans ce VLAN (trafic normal ou broadcast)
- On peut associer plusieurs VLANs sur un même port (Trunk)

VLAN

- Jusqu'ici on assumait que l'appartenance d'une trame à un VLAN était connue.
  Pour déterminer cette appartenance :
  - on peut associer un VLAN à un port
  - on peut associer un VLAN à une adresse MAC
  - on peut associer un VLAN à chaque protocole de couche 3 ou a chaque adresse IP

VLAN

Avantages et inconvénients des 3 méthodes

- Assignation d'un *VLAN par port* :
  Simple à réaliser, mais nécessite que toutes les machines accessible par ce port appartiennent au même VLAN
- Assignation d'un *VLAN par adresse MAC*
  Nécessité de gérer une liste des adresses MAC dans l'équipement (switch ou bridge)

VLAN

Avantages et inconvénients des 3 méthodes

- Assignation d'un *VLAN a un protocole* de couche 3 ou a une adresse IP :
  Cela nécessite de rompre la règle d'indépendance des couches, et donc risque de voir le système ne plus fonctionner en cas de changement majeur de la couche 3
  Exemple lors du passage de IPv4 à IPv6 (format des paquets, …)

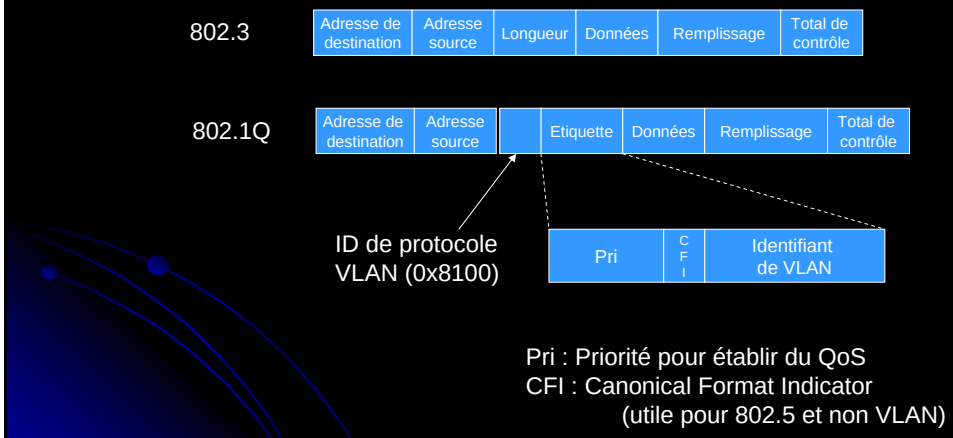
VLAN : 802.1Q

- Ce qui importe plus, c'est de connaître le VLAN de la trame, et non de la machine émettrice
- La nouvelle norme 802.1Q modifie le header de la trame Ethernet pour y ajouter un identifiant de VLAN

VLAN : 802.1Q

- Problèmes possibles :
  - Faut-il changer toutes les cartes réseaux pour supporter le 802.1Q ?
  - Si non, qui va générer les nouveaux champs ?
- Solutions ….
  - Seuls les switchs ont besoin des champs 802.1Q, donc on ne doit pas remplacer les cartes réseaux
  - C'est le premier switch qui ajoute les champs 802.1Q et le dernier sur la route les retire.

VLAN : 802.1Q



Pri : Priorité pour établir du QoS
CFI : Canonical Format Indicator
(utile pour 802.5 et non VLAN)

Matériel de niveau 2

- to do ...