Chapitre 3

Vue d'ensemble sur la couche liaison

La couche liaison de données a pour but de permettre à deux machines adjacentes dans un réseau de communiquer de façon fiable et efficace. Par **machines adjacentes**, il faut entendre que les deux machines sont physiquement connectées par un canal de transmission (câble coaxial, ligne téléphonique...) qui délivre les bits dans l'ordre dans lequel ils ont été émis.

Dans le cas d'un **canal de communication parfait**, la couche liaison de données n'a rien à faire : la machine A se contente d'émettre ses données sur le support physique, et la machine B de les collecter. Malheureusement, un canal de communication présente en général deux défauts :

- il est **bruité**, ce qui a pour conséquence des erreurs de transmission ;
- les deux machines n'ont pas le même débit, ce qui a pour conséquence qu'il faut réguler les flux de données échangées pour éviter que les destinataires lents ne soient submergés par des expéditeurs rapides, tout en essayant d'être le plus rapide possible ; on parle de **régulation** (ou **contrôle**) **des flux**.

3.1 Détermination des trames

3.1.1 Paquet, trame et somme de contrôle

Pour être en mesure de fournir un service à la couche réseau, la couche liaison des données doit utiliser le service de la couche physique. Celle-ci assure le transport de flux de bits sur le support et leur remise à la station de destination. Ces flux de bits peuvent comporter des erreurs : le nombre de bits reçus peut être inférieur, égal ou supérieur au nombre de bits émis ; les bits peuvent de plus avoir changé de valeur.

Pour résoudre les problèmes liés aux erreurs de transmission, la couche liaison des données n'accepte d'émettre qu'un certain nombre d'octets à la fois, appelé **paquet** (*packet* en anglais). La taille (fixe ou plus usuellement comprise entre un minimum et un maximum) d'un paquet dépend du protocole de liaison utilisé.

De plus chaque paquet est encapsulé dans une **trame** (*frame* en anglais) : chaque trame comprend un **en-tête** de trame (*header* en anglais), un champ de données pour héberger le paquet et un **en-queue** de trame (*trailer* en anglais), comme le montre la figure 3.1. L'en-tête

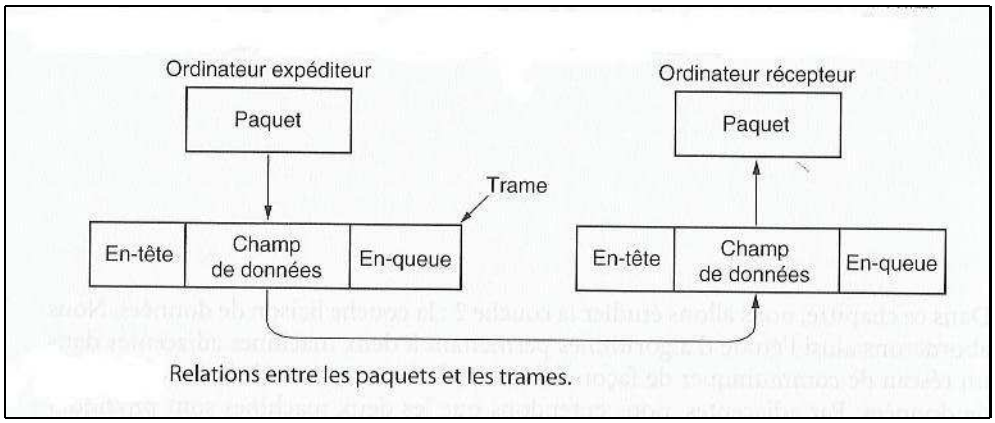

FIG. 3.1 – *Structure d'une trame*

et l'en-queue comportent des données de contrôle.

La donnée de contrôle la plus utilisée est la **somme de contrôle** (*checksum* en anglais) : la machine émettrice calcule un entier à partir des données suivant un algorithme bien précis qu'elle place parmi les donnée de contrôle ; lorsque la trame arrive à destination, la machine réceptrice calcule un entier suivant le même algorithme à partir des données reçues ; si le résultat obtenu est différent de celui calculé par la machine émettrice tel que reçu par la machine réceptrice, la couche liaison de données sait qu'une erreur de transmission s'est produite et peut prendre des mesures : ne pas tenir compte de la trame reçue et envoyer à l'émetteur un rapport d'erreur, par exemple. La façon de calculer la somme de contrôle dépend du protocole. Si le résultat est le même que celui calculé par la machine émettrice, on ne peut rien dire mais on considère que la trame reçue est identique à celle envoyée.

3.1.2 Découpage en trames

Le découpage en trames des flux de bits est plus difficile qu'il n'y paraît à première vue. Plusieurs méthodes ont été envisagées.

3.1.2.1 Insertion de silences

La première idée à laquelle on peut penser est d'insérer des silences pour délimiter les trames, exactement comme on place des espaces entre les mots. Malheureusement la couche physique garantit rarement les délais : il est donc possible que des silences disparaissent ou soient insérés dans les trames durant les transmissions.

3.1.2.2 Décomptage des octets

La seconde idée consiste à utiliser un champ dans l'en-tête de la trame qui indique le nombre d'octets que celle-ci contient. Lorsque la couche liaison de données de la station réceptrice lit ce champ, elle en déduit le nombre d'octets de la trame et donc la délimitation de celle-ci. Cette technique est illustrée à la figure 3.2 ([TAN–81], p. 204), qui montre quatre trames contenant

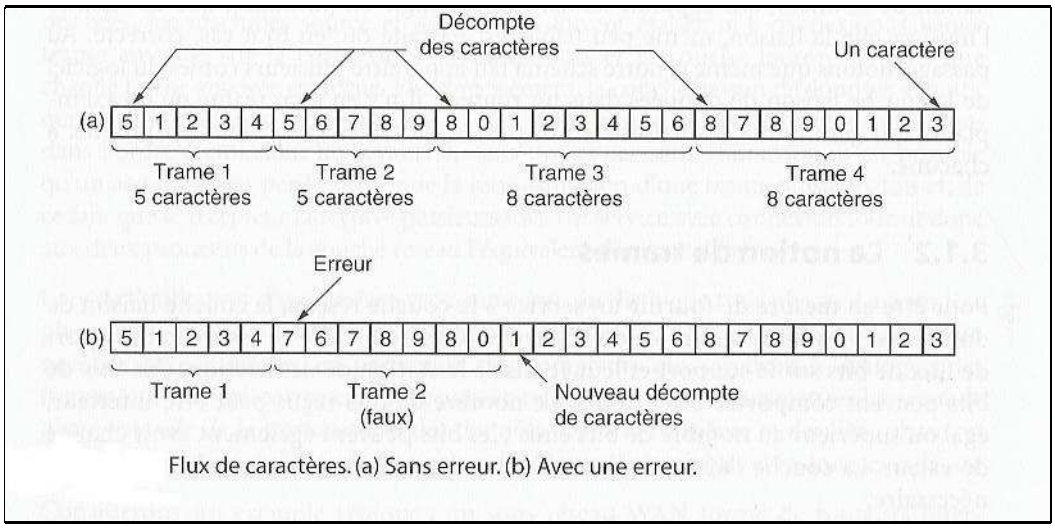

FIG. 3.2 – *Décomptage des octets*

respectivement 5, 5, 8 et 8 octets.

Cette méthode pose un problème lorsque l'octet indiquant le nombre d'octets de la trame est affecté par la transmission. Par exemple, si l'octet "nombre d'octets" de la seconde trame est changé de 5 en 7, le récepteur perd la synchronisation et ne peut plus trouver le début de la trame suivante. Grâce à la somme de contrôle, le récepteur se rendra certainement compte que cette trame est mauvaise, mais il ne sait pas où commence la suivante. S'il décide de demander à l'émetteur de retransmettre les données, celui-ci ne connaît pas le nombre d'octets à laisser passer pour trouver le début de la retransmission. Pour cette raison, cette méthode est rarement utilisée.

3.1.2.3 Fanions de signalisation avec caractères d'échappement

La troisième idée résout le problème de la **resynchronisation** après une erreur de transmission en délimitant chaque trame par des octets spéciaux de début et de fin. Par le passé, les protocoles employaient souvent des octets différents de début et de fin. À présent, ils utilisent le même octet, appelé **fanion de signalisation** (*flag* en anglais). Si le récepteur perd la synchronisation, il lui suffit de rechercher le fanion de signalisation pour trouver la fin de la trame (ou d'une trame suivante). Deux fanions consécutifs désignent la fin d'une trame et le début de la suivante.

Reste un problème de mise en œuvre. N'oublions pas que les données à transmettre sont binaires. Il se peut donc que le profil binaire du fanion de signalisation se présente dans les données, interférant avec l'encadrement. Pour résoudre ce problème, la couche liaison des données de l'émetteur insère un octet (caractère) spécial d'échappement (ESC) avant chaque fanion "accidentel". La couche liaison de données du récepteur élimine le caractère d'échappement avant de transmettre les données à la couche réseau. On peut ainsi distinguer l'encadrement du fanion de celui des données par l'absence ou la présence du caractère d'échappement qui le précède.

Ceci ne résout pas tout le problème de la mise en œuvre. La question suivante est : que se passe-t-il si un caractère d'échappement se présente dans les données binaires ? La mise en œuvre consiste à le faire précéder également d'un caractère d'échappement. Ainsi un caractère d'échappement seul désigne-t-il une séquence d'échappement, alors qu'un double caractère d'échappement indique qu'un échappement apparaît dans les données. La figure 3.3 montre quelques

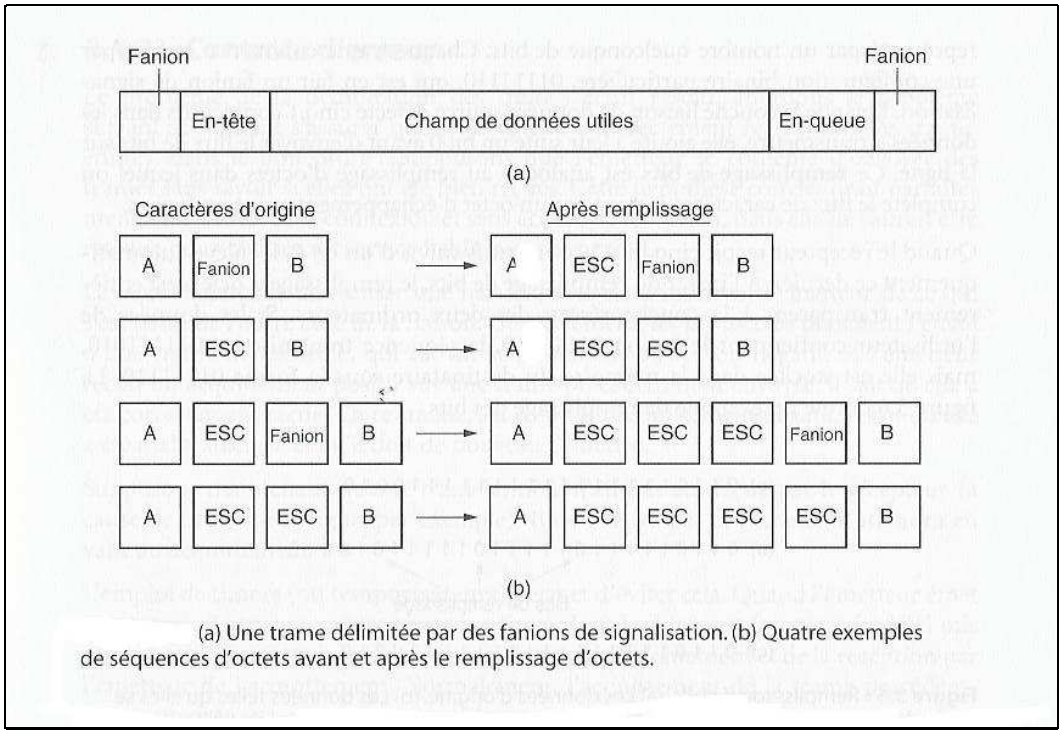

FIG. 3.3 – *Fanion de signalisation*

exemples de mise en place de cette **technique de remplissage**. Dans tous les cas, la séquence d'octets transmis après le **vidage** des caractères d'échappement est exactement identique à la

séquence d'octets d'origine.

3.2 Garantie de transmission correcte

Le problème de la délimitation des trames étant résolu, abordons le problème suivant : comment s'assurer que le récepteur a correctement reçu toutes les trames émises, dans le bon ordre?

La tâche essentielle est de garantir autant que faire se peut la communication en répétant la transmission de paquets de données erronés ou perdus. Cela ne peut pas se faire sans le concours du destinataire, car lui seul sait quels paquets sont déjà arrivés. Comment mettre ceci en place?

3.2.1 Première idée : confirmation ou infirmation

Le destinataire peut transmettre une **confirmation** (*Acknowledge* en anglais, abrégé en `ACK`) à son correspondant pour chaque paquet parvenu à bon port et dont la somme de contrôle a permis d'établir que ce paquet est *a priori* intact. En revanche, si la vérification donne lieu à la découverte d'une erreur, il envoie une **infirmation** (*Non-Acknowledge* dans l'anglais de TCP, abrégé en `NAK`).

L'expéditeur attend de recevoir un message de confirmation ou d'infirmation en provenance de son correspondant. Après réception de la confirmation, celui-ci expédie le paquet suivant. Dans le cas d'une infirmation, l'expéditeur reprend la transmission du dernier paquet jusqu'à ce qu'il obtienne une confirmation, avant de passer au paquet suivant, comme le montre la figure 3-4 ([TJ–97], p.139).

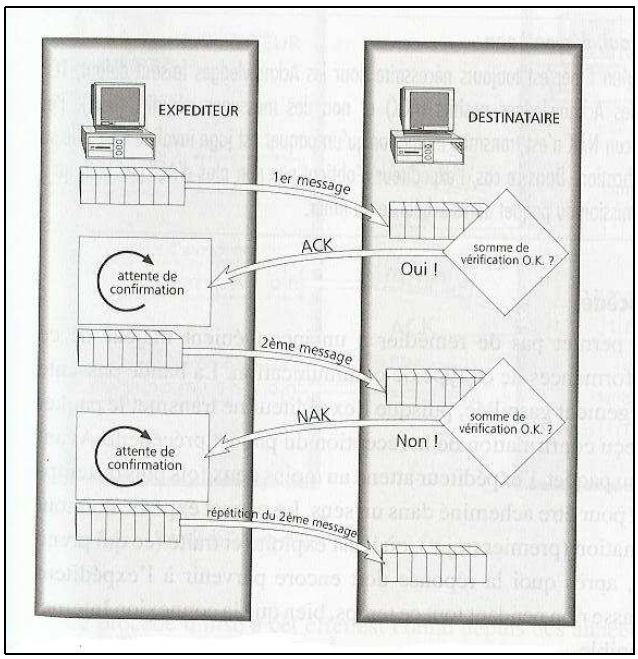

FIG. 3.4 – *Confirmation et infirmation*

Ce procédé ne peut pas être mis en place tel quel. En effet, qu'advient-il des messages non reçus, qui se sont perdus en cours de route?

3.2.2 Deuxième idée : confirmation et délai

Dans une version améliorée, l'expéditeur envoie son message et initialise un minuteur (*timer* en anglais). Il attend alors soit une confirmation, soit qu'un certain délai (*time out* en anglais) se soit écoulé. S'il reçoit la confirmation avant que le délai ne se soit écoulé, il envoie le message suivant. Sinon, une fois le délai échu, il considère qu'aucun message du destinataire ne lui parviendra et il transmet une nouvelle fois le message.

Si l'on transmet plusieurs fois le même message, il se peut que le récepteur l'accepte plus d'une fois et qu'il le délivre ainsi plusieurs fois. Pour éviter une telle situation, on numérote généralement les messages de telle sorte que le récepteur ne tienne pas compte des messages retransmis s'il n'y a pas lieu d'être. Ceci est un deuxième élément de contrôle (le premier étant la somme de contrôle).

Cependant l'utilisation d'un minuteur conduit à réduire fortement le débit. En effet la bande passante mobilisée est alors largement gaspillée puisque l'expéditeur ne transmet le paquet suivant qu'après avoir reçu confirmation de la réception du paquet précédent. Avant de transmettre un nouveau paquet, l'expéditeur attend au moins deux fois plus de temps qu'il n'en faut au paquet pour être acheminé dans un sens : le paquet expédié doit tout d'abord parvenir à destination (premier trajet), où il est exploité et traité (ce qui prend également un moment), après quoi la réponse doit encore parvenir à l'expéditeur (second trajet). Il ne se passe rien pendant tout ce temps, bien que la connexion soit intégralement disponible.

3.2.3 Troisième idée : fenêtre glissante

Pour remédier à ce gaspillage de la bande passante, on envoie plusieurs paquets avant d'attendre confirmation : trois, cinq ou huit, par exemple. On accorde ainsi au destinataire une sorte de crédit de confiance avant d'en attendre confirmation. Dès que la confirmation du premier paquet est reçue, le paquet suivant non encore expédié est envoyé sur la ligne, comme le montre la figure 3-5 ([TJ–97], p. 141). L'opération se répète ainsi pour les confirmations suivantes. Pour chaque nouvelle confirmation concernant un paquet précédent, le paquet suivant non encore transmis est expédié.

Il s'agit du procédé dit de la **fenêtre glissante** (*sliding window* en anglais). Il doit son nom au fait que l'expéditeur place une sorte de fenêtre sur le flux d'entités devant être expédié. Au début de la transmission, ce dernier envoie en premier lieu tous les paquets inclus dans cette fenêtre (d'expédition). À chaque confirmation, la fenêtre se décale sur la droite de manière à exclure le paquet confirmé situé sur la gauche. Cependant la fenêtre n'est pas réduite. Elle se déplace simplement sur la droite du flux de paquets devant être expédiés, incluant ainsi un nouveau paquet du côté droit de la fenêtre.

Pendant la durée de la transmission, la fenêtre d'expédition défile par conséquent de gauche à droite au-dessus du flux de paquets devant être transmis, tout en divisant celui-ci en trois groupes :

- tous les paquets situés à gauche de la fenêtre ont déjà été expédiés et confirmés ; l'expéditeur n'a plus à s'en occuper ;
- tous les paquets compris dans la fenêtre ont déjà été expédiés, mais pas encore confirmés ; leur nombre dépend de la largeur de la fenêtre ;
- tous les paquets situés à droite de la fenêtre attendent d'être inclus dans la fenêtre pour être envoyés ; plus les confirmations en provenance du destinataire parviennent rapidement à l'expéditeur, plus vite ces paquets se rapprochent de la fenêtre.

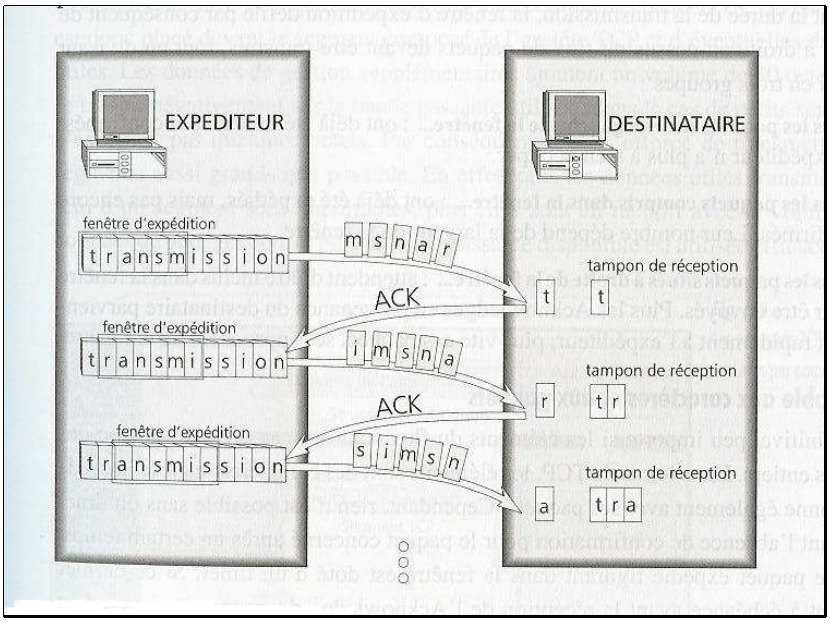

FIG. 3.5 – *Fenêtre glissante*

3.3 Contrôle de flux

Il se peut qu'un émetteur émette plus de messages que le récepteur ne peut en accepter. C'est le cas lorsque l'émetteur est sur un ordinateur rapide (ou peu chargé) et que le récepteur est sur une machine lente (ou très chargée). Dans ce cas, même si la transmission s'effectue sans erreur, le récepteur ne peut pas traiter toutes les messages émis et en perdra certains. Il faut donc mettre en œuvre un mécanisme pour éviter cette situation.

Il existe deux approches pour résoudre ce problème :

- La première consiste à instaurer un **contrôle de flux avec retour d'information**, ou rétroaction (*feed-back based flow control* en anglais), pour contraindre l'émetteur à ne pas envoyer plus de messages que le récepteur ne peut en accepter. L'émetteur peut ainsi savoir, grâce à un mécanisme de rétroaction, si le récepteur est en mesure ou non de recevoir les messages émis.
- Dans la seconde méthode, le **contrôle de flux basé sur le débit** (*rate based flow control* en anglais), un mécanisme intégré au protocole limite le débit de transmission des données, sans exploiter le retour d'information.

La seconde méthode n'est jamais employée dans la couche liaison de données.

Il y a de nombreuses variantes de contrôle de flux avec rétroaction. La plupart se fondent sur le principe suivant : il est interdit à l'émetteur d'envoyer des messages s'il n'a pas reçu auparavant une permission implicite ou explicite du récepteur. Par exemple, sur une réception déjà établie, le récepteur peut envoyer le message suivant : "Tu peux m'envoyer maintenant n messages, mais après ces n émissions, suspend tes envois jusqu'à ce que je te dise de continuer".

La couche liaison s'occupe de la régulation des flux entre deux machines adjacentes alors que, bien sûr, le problème le plus important est certainement la régulation des flux entre la machine

émettrice et la machine réceptrice (qui n'ont aucune raison d'être adjacentes). Cette régulation des flux-là est l'une des fonctionnalités éventuelles de la couche transport.

3.4 Gestion des canaux

Une fois que l'on a mis en place un support physique entre deux points, reste à choisir la meilleure façon de l'utiliser. On peut décider d'y envoyer les messages les uns à la suite des autres. Mais on s'est très vite aperçu que l'on ne rentabilise pas correctement l'utilisation du support physique dans ce cas, à cause des moments où il n'y a rien à dire. On préfère envoyer plusieurs messages à la fois, sur des **canaux** différents.

Il existe deux méthodes de détermination des canaux : l'allocation statique des canaux dans lesquels il ne peut pas y avoir collision entre deux messages sur le support donné et les **systèmes à contention** dans lesquels le partage d'un canal commun entre plusieurs utilisateurs peut engendrer des situations de conflits.

3.4.1 Allocation statique des canaux

Dans le cas de l'allocation statique d'un canal, on permet à plusieurs messages de passer dans le même support physique au même instant en **multiplexant** les messages, un par canal. À l'arrivée, il faut **démultiplexer** les messages. Il existe deux techniques fondamentales de multiplexage : le multiplexage en fréquence et le multiplexage temporel.

3.4.1.1 Multiplexage en fréquence

La méthode de partage d'un canal entre plusieurs usagers choisie à l'avénement du réseau téléphonique est le **multiplexage en fréquence** ou **FDM** (pour l'anglais *Frequency Division Multiplexing*). Les messages sont transportés à travers un support physique sous la forme d'une onde. L'ensemble des fréquences possibles des ondes pour un support physique donné est un intervalle, appelé **bande passante**. Lorsqu'il y a N utilisateurs, la bande passante est divisée en N portions de même taille, et chacune est affectée exclusivement à l'un d'eux. Il ne se produit donc pas d'interférences entre les différentes communications.

Lorsque le nombre des usagers est constant et peu élevé, et que chacun d'eux est la source d'un fort trafic, FDM offre un mécanisme simple et efficace de multiplexage sur un support de transmission partagé. Toutefois, lorsque le nombre d'émetteurs est élevé et varie sans cesse, ou que le trafic est sporadique, FDM pose quelques problèmes. Pour commencer, si le spectre des fréquences disponibles est divisé en N portions et, qu'à un instant donné, moins de N utilisateurs souhaitent transmettre, une grande partie des ressources est gaspillée. Ensuite, si plus de N nœuds souhaitent communiquer, certains ne seront pas autorisés à le faire par manque de bande passante, même si parmi les émetteurs bénéficiaires d'une bande, celle-ci est peu ou pas du tout exploitée.

3.4.1.2 Multiplexage temporel

Dans le cas d'un **multiplexage temporel**, ou **TDM** pour l'anglais *Time Division Multiplexing*, on alloue à chaque utilisateur un intervalle de temps fixe qu'il voit se renouveler tous les N intervalles de temps.

Les arguments contre FDM sont également valables contre TDM. Comme aucune méthode d'allocation statique n'est satisfaisante avec un trafic sporadique, on a pensé à des méthodes dynamiques.

3.4.2 Gestion des canaux à accès multiples

Nous avons vu que les réseaux peuvent être divisés en deux catégories selon le mode de communication qu'ils utilisent : mode en point-à-point ou mode à diffusion. Dans le cas d'un réseau à diffusion, dans lequel tous les participants entrent en compétition pour utiliser un canal partagé, on doit déterminer qui, à un instant donné, est autorisé à émettre.

Les canaux de diffusion sont parfois désignés par les termes **canaux à accès multiple** ou **canaux à accès aléatoire**. Voyons les méthodes de gestion de ce type de canal.

3.4.2.1 ALOHA pur

Bien que le système ALOHA d'origine se fondait sur la diffusion radio terrestre, son principe reste applicable à n'importe quel autre système dans lequel des utilisateurs non coordonnés se disputent l'usage d'un canal unique.

L'idée de base du système ALOHA est simple : laisser les utilisateurs accéder librement au canal lorsqu'ils ont des données à transmettre. Dans ces conditions, il est clair que des **collisions** se produisent et que les trames qui en sont victimes sont détruites. Toutefois la technologie utilisée permet toujours à un émetteur de savoir si une trame a été ou non victime d'un incident. Lorsqu'une collision se produit, les participants en cause attendent un laps de temps aléatoire avant de tenter une nouvelle transmission. Si la durée d'attente doit être aléatoire, c'est pour éviter que les mêmes trames se percutent de façon répétitive, ce qui conduirait à une situation de blocage.

3.4.2.2 ALOHA discrétisé

En 1972, Larry Roberts publiait une méthode pour doubler la capacité d'un système ALOHA. Sa proposition était de diviser le temps en **slots**, ou intervalles finis, chacun d'eux correspondant à une trame. Cette approche demande que les utilisateurs s'accordent sur la délimitation des slots. Une façon de procéder à la synchronisation des stations était de disposer d'une station spéciale chargée d'émettre un bit au départ de chaque slot, telle une horloge. Chaque station doit attendre le début du prochain slot pour émettre. Le système ALOHA est ainsi transformé en système discret.

Ce système a été utilisé dans quelques sytèmes expérimentaux, puis pratiquement oublié. Lorsque l'accès à internet par le câble fut possible, ALOHA discrétisé a été sorti de son carton.

3.4.2.3 Protocoles d'accès par écoute de porteuse CSMA

Le problème avec ALOHA est qu'un système où des stations transmettent à volonté, sans égard pour l'activité des autres stations, est évidemment propice à de nombreuses collisions. Sur un réseau local, la technologie est telle que chaque participant peut détecter l'activité des autres stations et adapter son comportement en conséquence. Les protocoles qui permettent aux stations d'adapter leurs décisions à l'activité en cours sont appelés **protocoles à détection de porteuse** (CSP pour l'anglais *Carrier Sense Protocol*).

Le premier des protocoles à détection de porteuse est appelé **CSMA 1-persistant** (pour l'anglais *Carrier Sense Multiple Access*). Quand une station doit expédier des données, elle commence par écouter le support de transmission pour savoir si une autre station n'est pas déjà en train de transmettre. Si c'est le cas, elle attend que le canal se libère. Quand il est de nouveau disponible, elle transmet sa trame. Lorsqu'une collision se produit, l'émetteur d'une trame observe un temps de pause aléatoire, puis recommence la procédure depuis le début. Le nom de ce protocole vient du fait que la probabilité qu'une station souhaitant transmettre trouve le canal disponible est égale à 1.

ALOHANET ne pouvait pas fonctionner de cette manière car un terminal sur une île était dans l'impossibilité de déterminer si un terminal sur une autre île était ou non en train d'émettre.

3.4.2.4 Le protocole d'accès multiple avec détection de collision

Un autre problème survient néanmoins lorsque, après avoir attendu la fin de la transmission en cours, deux ou plusieurs ordinateurs décident d'émettre simultanément. La solution est de faire en sorte que chaque machine écoute pendant qu'elle-même transmet et alerte tous les autres émetteurs si elle détecte une interférence. Chacun attend ensuite pendant une période de temps aléatoire avant d'émettre à nouveau. Si une nouvelle collision survient, l'intervalle aléatoire est doublé (et le sera à chaque collision successive) pour donner à l'une des machines l'occasion d'émettre en premier. On parle alors de **CSMA/CD** pour *CSMA with Collision Detection*.