

# SEMAINE 3

## APPROCHES DES NOTIONS DE RESEAUX

\*\*\*


*\_ Olivier PAUL \_*

*Maître de conférences à Télécom SudParis*

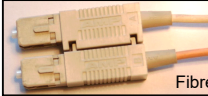
\*\*\*


### Leçon 1 : Le transport de l'information sur le support physique

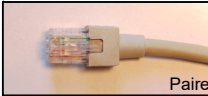
**Avec quels supports physiques ?**

**■ Support hertzien**  
  
Point d'accès sans fil

**■ Supports tangibles**

  
Fibre optique


  
Câble coaxial

  
Paire torsadée

209/09/2014

Institut Mines-Télécom


Modèle de présentation Institut Mines-Télécom



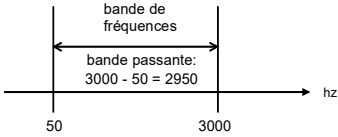
Il existe une grande variété de supports physiques. Tous ne sont pas liés à un élément tangible comme le support hertzien qui est utilisé dans les réseaux sans fils ou satellitaires. Parmi les autres supports, on trouve les supports optiques comme la fibre ainsi que les supports métalliques tels que le câble coaxial ou la paire torsadée. Ces supports sont terminés par des connecteurs permettant de coupler le support physique à un système communicant.

## Quelle capacité de transmission ?

■ **L'information**



■ **La bande passante**



■ **Le débit**


$$D = B \log_2(P)$$

3

08/09/2014

Institut Mines-Télécom

Modèle de présentation Institut Mines-Télécom

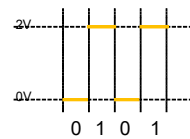


Une définition populaire de l'information est « quelque chose de nouveau, de différent ». Afin de transporter une information entre un émetteur et un récepteur sur un support physique, il faut donc faire varier quelque chose : un signal circulant entre les deux. On peut comprendre intuitivement que la fréquence avec laquelle on va pouvoir faire varier ce signal a une influence sur la capacité de transmission entre deux systèmes. Cette capacité de transmission se mesure généralement en terme de bande passante mesurée en hertz. Celle-ci mesure l'intervalle de fréquence dans lequel l'émetteur peut émettre un signal qui sera compréhensible par le récepteur (par exemple, dans le cadre de communications entre deux personnes, la bande passante est de l'ordre de 3000 Hz ; la voix humaine pouvant être émise dans la bande de fréquence environ comprise entre 50 Hz et 3000 Hz). Cette capacité dépend du support physique lui-même, de sa nature, de sa longueur, ainsi que des capacités de l'émetteur et du récepteur. Il a été démontré que la bande passante, et le débit en bits/s pouvant être atteint par des communications pour un type de codage donné, et dans un environnement donné, sont proportionnels. On trouvera fréquemment ces deux quantités utilisées l'une pour l'autre.

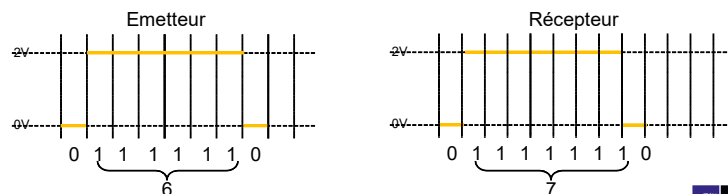
Comparons par exemple deux supports physiques dans un même environnement. L'un, A, pour lequel la bande de fréquence utilisable est comprise entre 300 et 3300 Hz. L'autre, B, pour lequel la bande de fréquence est comprise entre 25700 et 27000 Hz. Lequel des deux aura la capacité de transmission la plus importante ? Ce sera avec A, puisque pour A, la bande passante sera de  $3300 - 300 = 3000$  Hz alors que pour B, elle sera de  $27000 - 25700 = 1300$  Hz.

## Codage numérique

### Variation de l'intensité/la tension



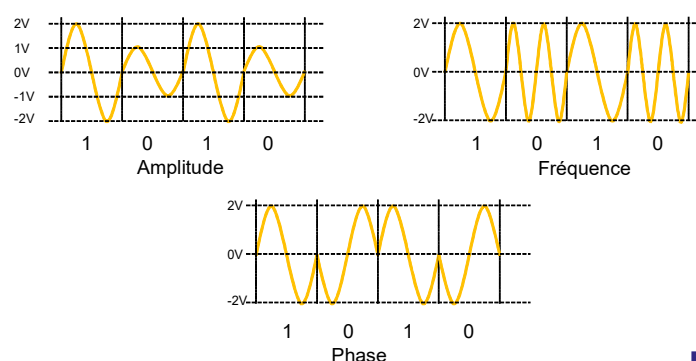
### Problèmes de synchronisation



Une caractéristique simple du signal, que l'on peut penser faire varier, est son intensité ou sa tension. Dans le cas le plus simple, on peut faire varier celle-ci entre deux états haut et bas ; l'un pour la valeur binaire '0', l'autre pour la valeur '1'. Imaginons cependant maintenant que l'émetteur veuille émettre une suite de '1'. Si l'horloge de l'émetteur n'est pas synchronisée avec celle du récepteur et possède une fréquence plus rapide, comme dans notre exemple, ce dernier pourra recevoir un nombre de '1' différent du nombre émis. Ici, les 6 '1' transmis sont compris comme 7 '1' par le récepteur. Cette synchronisation est difficile à obtenir en pratique.

## Codage par modulation

### Caractéristiques modulables

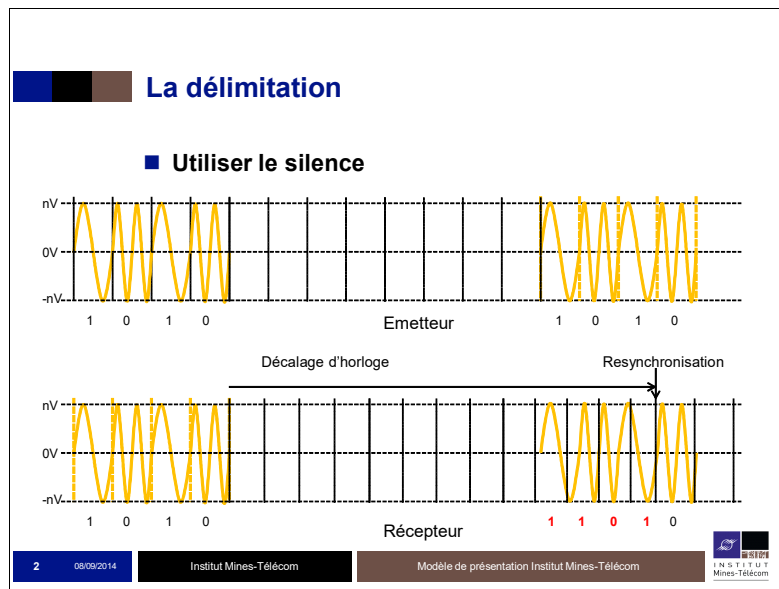


On est alors amené à moduler le signal ; c'est-à-dire à coder l'information binaire en un signal de forme sinusoïdale, périodique, en faisant varier ses caractéristiques. Avec un signal de ce type, on peut également jouer sur l'amplitude pour coder les '0' et les '1' en utilisant une amplitude forte pour la valeur '1', et une amplitude faible pour la valeur '0'. On peut varier la fréquence de l'alternation ; c'est-à-dire le nombre de variations du signal par

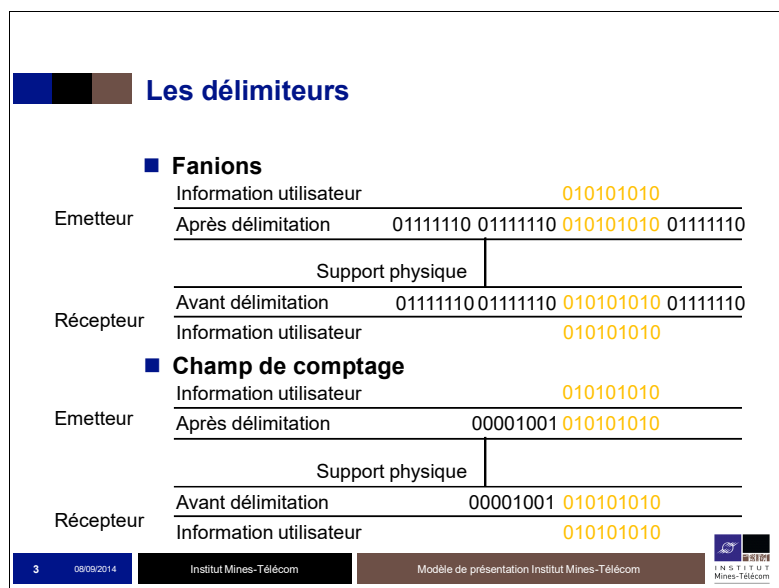
période de mesure, en codant par exemple un '1' par une alternance lente, et un '0' par une alternance plus rapide. Enfin, on peut également varier la phase du signal ; c'est-à-dire créer ou pas un décalage du signal en début de période de mesure afin de coder un '1' ou un '0'. Il est également possible de combiner plusieurs méthodes afin de pouvoir transmettre plusieurs bits par période de mesure.

## Leçon 2 : La délimitation et la transparence

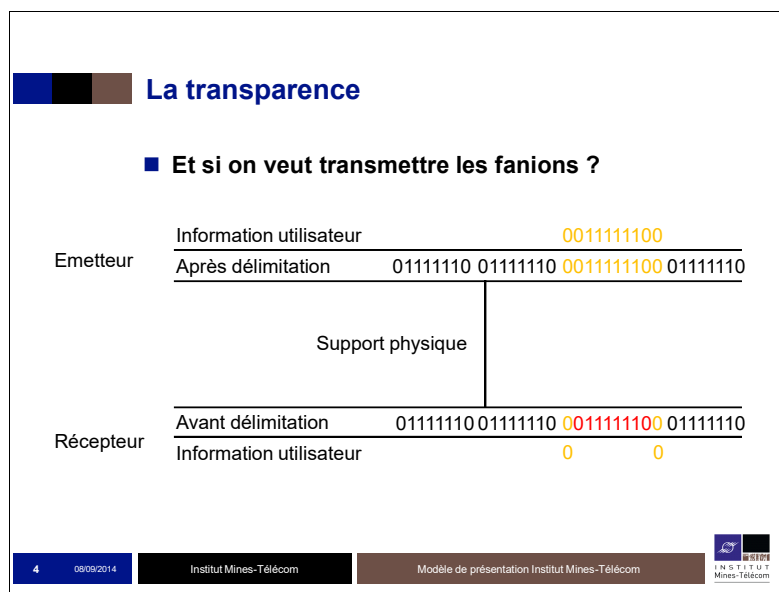
Maintenant que nous savons transférer l'information numérique sous la forme d'un signal entre deux équipements, il faut nous intéresser au cas où ils n'auraient pas d'information à transmettre. Comment peuvent-ils alors se mettre d'accord pour définir le début et la fin des unités de données ? c'est-à-dire pour réaliser ce que l'on appelle leur délimitation.



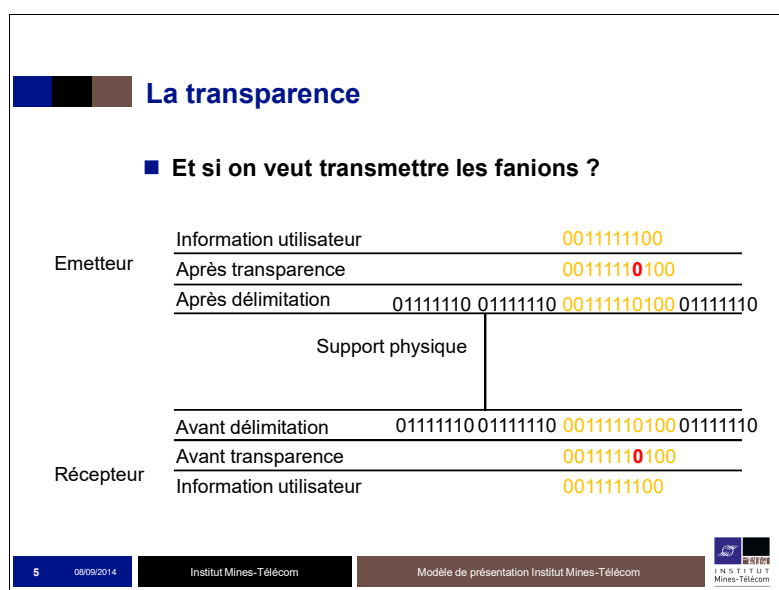
Le mécanisme le plus simple semblerait tout naturellement de ne rien émettre, comme dans une conversation humaine. Cependant, comme le signal émis par la couche physique peut servir à synchroniser l'horloge du récepteur, comme dans notre exemple, il est possible que l'absence de signal provoque une désynchronisation de celle-ci qui aboutisse à un décodage incorrect. Il faut alors prévoir une phase de resynchronisation en envoyant un signal prédéfini au récepteur avant de transmettre les données utiles. Cette solution, appelée préambule, est adoptée dans certaines versions du protocole Ethernet.



D'autres solutions existent. Certains protocoles, comme HDLC, utilisent des fanions ; c'est-à-dire une suite de bits prédéfinie. Ces fanions, d'une valeur binaire 01111110, sont envoyés en permanence par l'émetteur lorsque celui-ci n'a aucune donnée utile à envoyer. On les trouve donc placés en début et en fin des données utiles à transmettre. Lorsqu'un récepteur reconnaît un fanion, il recopie les données qui le suivent jusqu'à ce qu'un nouveau fanion soit trouvé. D'autres protocoles, tels que le protocole IP, utilisent un champ de comptage dont l'emplacement est bien connu ; par exemple en début d'unité de donnée. Celui-ci contient la taille de l'unité de données, ici indiquée en binaire. Elle permet au récepteur de savoir où l'unité de données se termine.



L'utilisation de fanions pose cependant un problème. Que va-t-il se passer si l'utilisateur veut envoyer une suite binaire analogue au fanion ? Ceci pourrait perturber le récepteur qui reconnaîtrait faussement une fin d'unité de données à l'endroit où se trouverait cette suite. Les informations utiles reconnues par le récepteur seraient alors différentes de celles envoyées.




Afin de résoudre ce problème, l'émetteur peut être amené à modifier l'information binaire à transmettre avant de la placer entre les deux fanions. Ainsi, dans le cas du protocole HDLC, lorsqu'une suite de 5 bits à 1 est repérée dans les données binaires transmises par l'utilisateur, celle-ci est complétée par un '0' par l'émetteur de manière à empêcher la formation du motif fanion. De manière opposée, le récepteur, lorsqu'il lit une suite de 5 bits à '1' suivie d'un '0', supprime celui-ci. Ce mécanisme rend un service appelé service de transparence.

Petit exemple pour terminer cette séance : supposons que l'utilisateur du système émetteur utilise les services de délimitation et de transparence. Supposons qu'il envoie la suite de bits 01111100. Quelle sera la suite de bits envoyée sur le support physique ? La fonction de transparence va provoquer l'ajout d'un bit '0' après les 5 '1'. Le motif, avant délimitation, sera donc 0 11111 000. La délimitation va ensuite provoquer l'ajout des fanions 01111110 en début et en fin d'unité de données. Le motif émis sur le support physique sera donc 0111111001111100001111110.

## Leçon 3 : La détection des erreurs

On a vu, dans notre séquence précédente, comment l'information binaire pouvait circuler entre deux systèmes au travers de la couche physique. Cependant, cette circulation n'est pas toujours parfaite. En effet, différents phénomènes peuvent altérer l'information reçue par le récepteur. Ceux-ci sont de deux types. Des phénomènes internes aux systèmes communicants, comme des erreurs logicielles ou matérielles, peuvent provoquer la modification des informations binaires stockées et manipulées dans les systèmes. Des phénomènes externes, comme des bruits électromagnétiques, l'endommagement du support physique ou des connecteurs, l'utilisation d'un support de mauvaise qualité provoquant un affaiblissement trop important, peuvent empêcher un récepteur de décoder le signal envoyé par l'émetteur (Vous avez déjà sans doute rencontré ces phénomènes en parlant à une autre personne dans un environnement bruyant ou lorsque celle-ci est trop éloignée.).

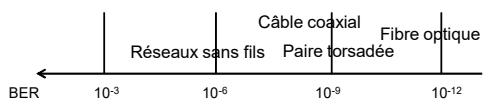


### Mesurer les erreurs

- **Le BER**
  - $e$ : nombre de bits reçus erronés par le récepteur.
  - $t$ : nombre total de bits reçus par le récepteur.

$$BER = \frac{e}{t}$$

- **Valeur typiques**




Support	BER typique
Réseaux sans fils	$10^{-3}$
Câble coaxial	$10^{-6}$
Paire torsadée	$10^{-9}$
Fibre optique	$10^{-12}$

- **BER et débit**

309/09/2014

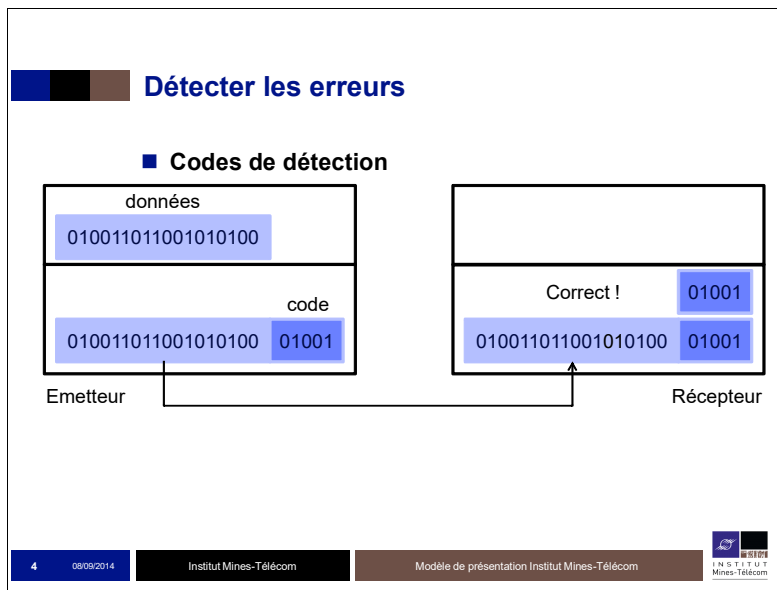
Institut Mines-Télécom

Modèle de présentation Institut Mines-Télécom



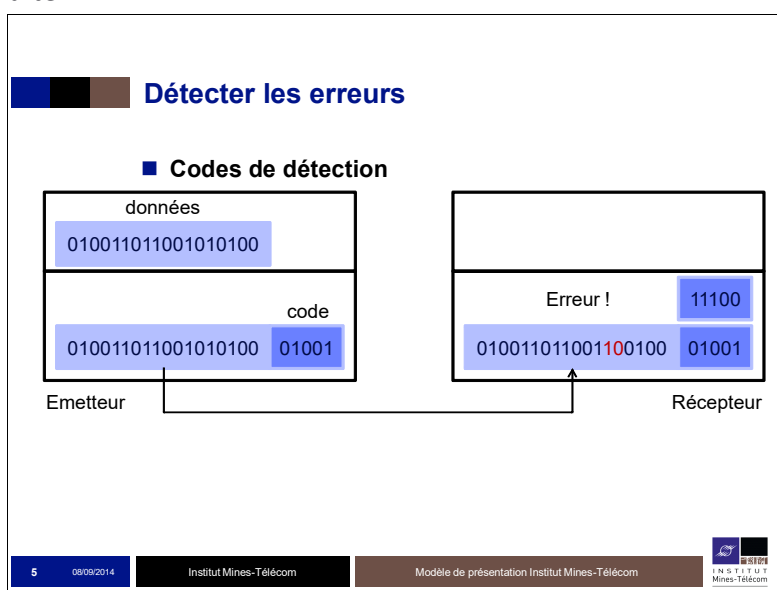
Ces erreurs sont généralement mesurées sous la forme d'un ratio entre le nombre de bits reçu erronés, et le nombre de bits transmis, ou BER : *bit error rate* en anglais. Le BER peut varier fortement d'un environnement à l'autre, et également en fonction du type de support utilisé. On mesure typiquement des BER de l'ordre de  $10^{-6}$  dans les réseaux sans fil ; c'est-à-dire un bit erroné par million de bits transmis. Dans les réseaux à supports tangibles, il est courant de mesurer des BER de l'ordre de  $10^{-9}$  pour les supports métalliques, et  $10^{-12}$  pour les supports optiques.





Même si les valeurs de BER actuelles semblent faibles, il convient, au moins pour certains usages, de détecter les erreurs afin de savoir si l'information reçue est utilisable. Ainsi, avec un support physique sans fil fournissant un BER de  $10^{-6}$  et supportant un débit de 52 Mb/s, 52 erreurs binaires sont en moyenne introduites par seconde. Si celui-ci est utilisé pour transporter des transactions bancaires, les clients de la banque préféreront sans doute que les systèmes communicants soient capables de détecter celles-ci afin que leurs comptes ne soient pas débités de montants erronés.

Afin de détecter ces erreurs, l'émetteur ajoute, aux données transmises par l'utilisateur, une information redondante de contrôle appelée code. Ce code, typiquement calculé à partir des données, va permettre au récepteur de détecter si les données ou le code ont été modifiés lors de leur transfert entre l'émetteur et le récepteur. Pour cela, il peut réaliser la même opération et calculer si le code qu'il a calculé sur les données reçues est le même que le code reçu. Si c'est le cas, il considère que les données sont correctes. Sinon, il en conclut qu'une erreur s'est produite.



Un exemple simple de code de détection d'erreur est le bit de parité. Il permet de détecter un nombre impair de bits erronés lors d'une transmission. Un bit de parité est un bit ajouté à la suite de bits transmise par l'utilisateur émetteur, et qui portera la valeur '1' si le nombre de bits à '1' dans cette suite est impair. Dans le cas contraire, il portera la valeur '0'. Lorsque le récepteur reçoit l'unité de données, il compte le nombre de bits à '1'. Si le compte est impair, il en déduit qu'au moins une erreur binaire s'est produite. Si nous prenons par exemple la suite binaire 1010100, celle-ci contient 3 bits à '1'. Le bit de parité associé est donc '1', de telle sorte que le total des bits à '1' soit 4, un nombre pair. L'unité de données transmise sera alors 1010100 1. Imaginons que le récepteur reçoive la suite 1010100 0 ; le bit de parité étant erroné. La suite de bits contenant 3 bits à '1', un nombre impair, il en déduit qu'une erreur de transmission s'est produite. On voit ici que le récepteur est, dans cette technique, incapable de savoir où l'erreur se trouve. Notons qu'un nombre pair d'erreurs binaires ne peut être détecté puisque deux erreurs binaires vont se compenser. Ainsi, ici, l'erreur transformant les deux bits '1' en '0', le nombre total de bits à '1' reste pair.

Il existe des codes de détection plus complexes tels que les codes de redondance cyclique (ou CRC) qui permettent de détecter un plus grand nombre d'erreurs en contrepartie d'une taille plus importante (typiquement 16 ou 32 bits) et d'une méthode de calcul plus complexe. Il existe également des codes de correction d'erreur permettant, comme leur nom l'indique, de corriger les bits erronés. Pour une capacité de correction égale à une capacité de détection (par exemple la capacité de détecter 4 bits erronés et la capacité de les corriger), le code de correction utilisera typiquement un code de taille plus importante. On aura donc plutôt tendance à utiliser ces codes lorsque le BER est élevé. Il existe un grand nombre de codes. Chacun a des caractéristiques propres en termes de détection. On choisit un code par rapport à un autre afin que la probabilité de ne pas détecter une erreur pour un environnement donné soit acceptable pour une application donnée et que son coût le soit également.

Petit exemple pour terminer cette séance : avec le bit de parité, l'émetteur envoie la suite binaire 11010 1, et le récepteur reçoit la suite 11010 0. Que peut faire le récepteur ? Le récepteur voit que la suite reçue est bonne et peut l'utiliser. Le récepteur voit que la suite reçue est erronée mais il peut l'utiliser car l'erreur porte sur le bit de parité. Enfin, le récepteur voit que la suite reçue est erronée et il ne peut pas l'utiliser. La réponse est que, lorsqu'il reçoit 11010 0, le récepteur détecte une erreur puisque le nombre de bits à '1' est impair. Cependant, cette erreur pourrait venir de la transformation du 2e bit de '0' en '1', ou de la transformation du 5e bit de '1' en '0'. Il ne peut donc savoir que les données sont correctes. Il ne peut donc pas les utiliser.

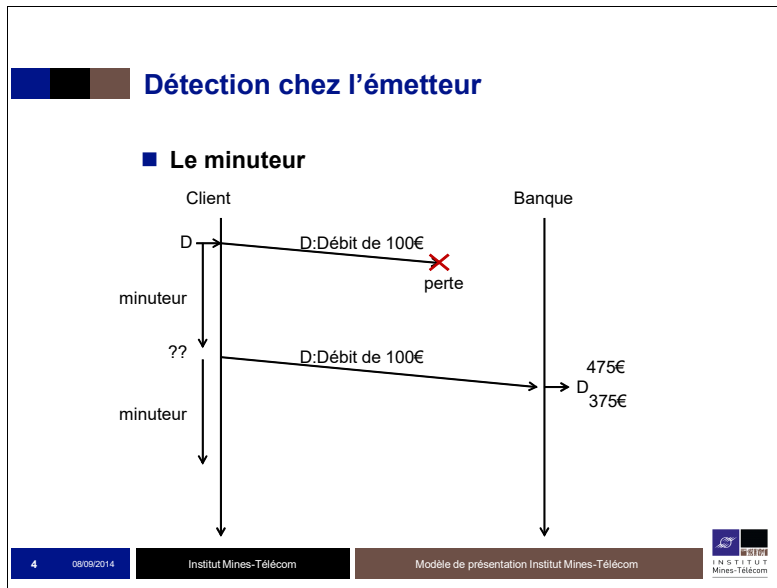
## Leçon 4 : La détection et la reprise sur perte

Nous avons vu précédemment que les unités de données pouvaient arriver erronées à leur destinataire. Si le protocole entre le récepteur et l'émetteur n'utilise pas un code de correction d'erreur, que doit faire le destinataire avec cette unité de données erronée ? Ne pouvant déterminer quelle partie est erronée, il doit faire comme si elle ne lui était jamais arrivée. Elle est considérée comme perdue. L'émetteur et le récepteur doivent mettre en œuvre un protocole permettant de pallier cette perte. C'est ce qui nous intéressera dans ce cours.

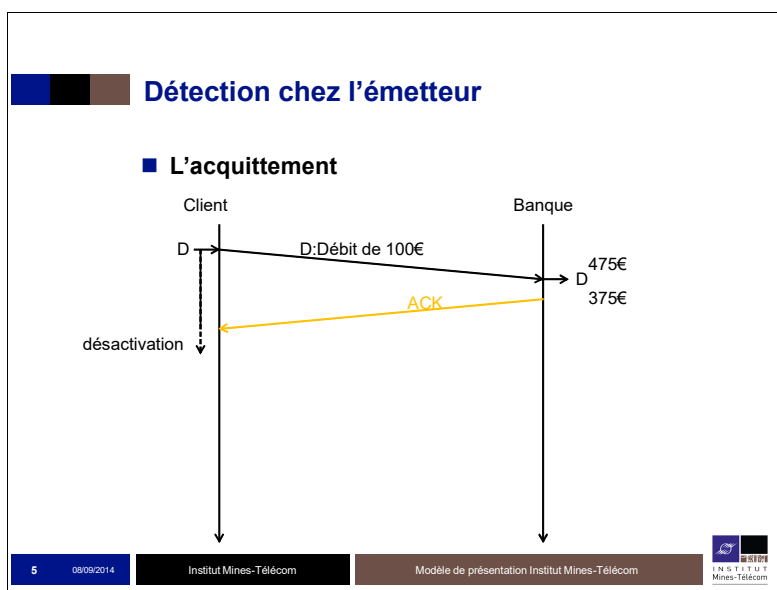
Des raisons, autres que les erreurs, peuvent causer la perte d'unités de données, comme la saturation ou le résultat d'erreurs logicielles au niveau de l'émetteur, du récepteur, ou d'équipements intermédiaires. Les supports physiques peuvent aussi être endommagés ou perturbés au point de ne pas pouvoir transporter le moindre signal entre l'émetteur et le récepteur, sans que l'émetteur ne s'en rende compte. (Vous avez sans doute déjà rencontré ce phénomène lors d'une conversation avec un téléphone portable lorsque votre interlocuteur passe dans un tunnel.)



Si certains types de communications supportent les pertes, d'autres les supportent moins bien. Ainsi, dans un cadre bancaire, les clients préféreront sans doute que les systèmes communicants soient capables de détecter et de corriger des pertes afin que leurs comptes soient crédités ou débités des montants adéquats. Afin de créer un protocole permettant de lutter contre les pertes, c'est-à-dire de faire de la reprise sur perte, il faut d'abord détecter celles-ci. D'une manière analogue à une conversation humaine, on peut imaginer de faire cette détection chez l'émetteur ou chez le récepteur. Chez l'émetteur : s'il ne reçoit pas un signe que ce qu'il a dit a été entendu par le destinataire. Ou par le récepteur : s'il perçoit un manque entre deux informations reçues. Comment ces mécanismes peuvent ils se traduire dans un environnement numérique ?

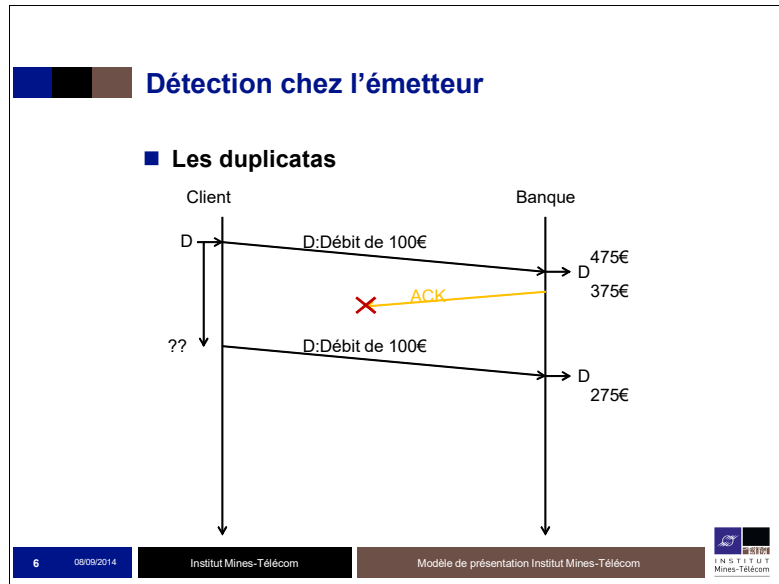


Afin de détecter la perte chez l'émetteur, on va associer à chaque unité de données émise, un minuteur : une sorte de réveil qui va se déclencher au bout d'un temps prédéfini. Imaginons par exemple que l'on considère une communication entre un client et sa banque. Le client désire que son compte soit débité correctement de ses achats. Nous utilisons ici un chronogramme ou diagramme temporel pour représenter le trajet des unités de données. Dans celui-ci, nous ignorons volontairement la durée d'émission. Une première PDU est envoyée et un minuteur est enclenché pour cet envoi. Cette PDU se perd avant d'arriver à la banque. Lorsque le minuteur se déclenche, l'émetteur se rend compte qu'il n'a pas reçu de réponse de la banque et en déduit que la PDU qu'il avait envoyée s'est perdue. Il va alors retransmettre celle-ci afin que celle-ci ait une nouvelle chance d'arriver à son destinataire et enclenche de nouveau un minuteur. Lorsque celle-ci arrive cette fois-ci correctement à la banque, le compte est débité.

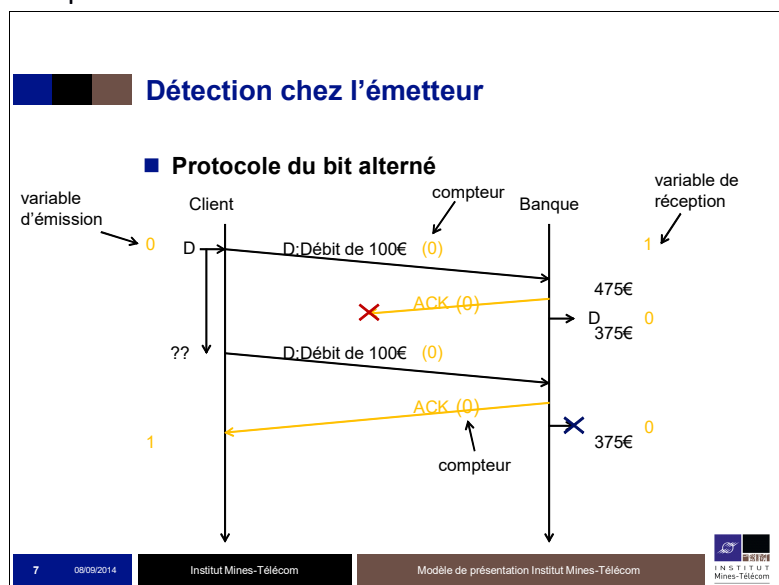


Comme nous venons de le dire, c'est parce que le client ne reçoit pas de message de la banque lui indiquant qu'elle a bien reçu son message, que le client réémet celui-ci. Il

convient donc, pour le récepteur, lors de la réception d'une PDU, d'envoyer à l'émetteur une indication de réception, aussi appelée acquittement. Contrairement à une conversation humaine, cet acquittement ne contient pas nécessairement des informations utiles transmises par la couche supérieure. Il peut ne contenir que des informations de contrôle. Lors de la réception de l'acquittement, l'émetteur désactive le minuteur afin que celui-ci ne provoque pas une retransmission inutile.

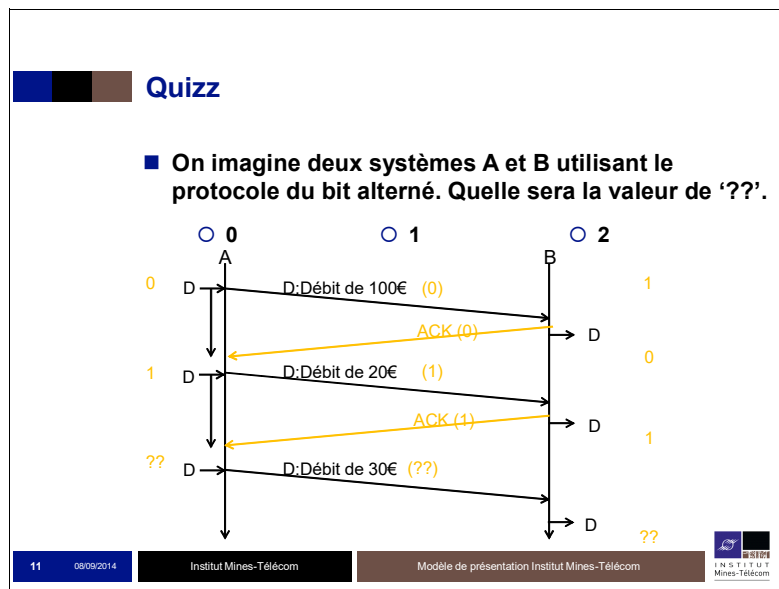


Examinons maintenant un scénario différent, dans lequel c'est l'acquittement qui se perd. L'unité de données est envoyée par le client et le minuteur est enclenché. Cette unité de données parvient au récepteur. Le compte est alors débité correctement. Le récepteur envoie ensuite l'acquittement. Cet acquittement se perd. Le minuteur se déclenche alors au niveau de l'émetteur qui réemet l'unité de données initiale en enclenchant le minuteur. Celle-ci peut alors parvenir de nouveau au récepteur, ce qui provoque le débit du compte une nouvelle fois. Le problème est maintenant que notre protocole peut délivrer les données plusieurs fois au récepteur.



Prenons maintenant un exemple pour comprendre comment ces différents éléments sont utilisés. Imaginons que les compteurs soient initialisés à 0 et à 1 chez l'émetteur et le récepteur. Lors de la réception de données à transmettre, l'émetteur enclenche le minuteur et ajoute aux données utiles le compteur qui prend la valeur de variable d'émission : 0. À la réception de la PDU, le récepteur envoie un acquittement portant la même valeur de compteur. Le récepteur vérifie ensuite si le compteur a une valeur différente de la variable de

réception. Comme 0 est différent de 1, il délivre les données à l'utilisateur et donne à la variable de réception la valeur du compteur, c'est-à-dire 0. L'acquittement se perd ; ce qui provoque le déclenchement du minuteur chez l'émetteur. Celui-ci retransmet alors la dernière unité de données transmise et réenclenche le minuteur. Lorsque celle-ci est reçue, le récepteur envoie un acquittement comme précédemment. Il compare ensuite la valeur du compteur et celui de la variable de réception. Ces deux valeurs étant les mêmes (0 et 0), il ignore les données reçues et laisse la variable inchangée. Lorsque l'émetteur reçoit l'acquittement, il vérifie que la valeur de compteur correspond à celle contenue dans la variable d'émission. Comme c'est le cas, il incrémente la valeur de la variable d'émission, qui passe de 0 à 1, et désactive le minuteur associé à la transmission précédente.



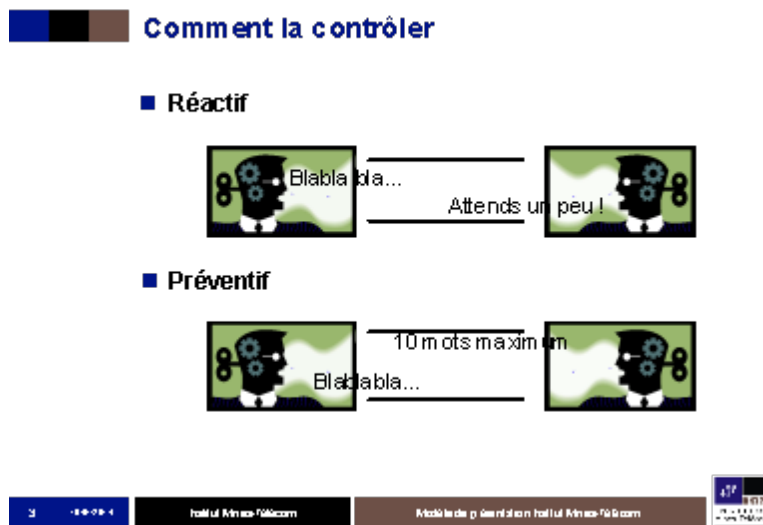
Pour continuer, voici un petit exercice. Imaginons deux systèmes, A et B, utilisant le protocole du bit alterné. Le chronogramme, que nous voyons ici, représente le transport de trois unités de données entre les deux. Quelles seront, selon vous, la valeur du compteur associé à la troisième unité de données, et les valeurs des deux variables chez l'émetteur et le récepteur ? 0, 1 ou 2 ?

La réponse est zéro. En effet, lorsque A reçoit ACK 1, la valeur du compteur est égale à celle de la variable d'émission. Celle-ci est donc incrémentée de 1. Cependant, comme cette variable est codée sur un bit, sa valeur revient à 0. Cette variable est utilisée pour la valeur du compteur de la PDU suivante (« débit de 30 euros ») qui prend donc la valeur 0. Lorsque cette PDU est reçue par B, celui-ci constate que la valeur du compteur est différente de la valeur de la variable de réception. La variable prend alors la valeur du compteur et les données sont livrées à l'utilisateur.

## Leçon 5 : Le contrôle de flux

Dans une conversation, on a tous un jour été confronté à la situation où notre interlocuteur nous parle de façon continue sans nous laisser le temps de réagir. Au bout d'un certain temps, on ne se rappelle plus du début de sa phrase. Une partie de l'information a été transmise en vain. Dans les environnements numériques, un problème similaire peut se poser. Afin d'éviter de transmettre des unités de données pour rien, on peut mettre en place un service permettant à un récepteur de contrôler le rythme d'émission d'un émetteur. Ce service s'appelle le contrôle de flux.

Dans les réseaux, l'émission trop rapide de données par un émetteur vers un récepteur peut provoquer une saturation de celui-ci qui ne sera pas capable de stocker les unités de données lui parvenant. Les données contenues dans celles-ci sont alors perdues. Ce problème est fréquemment dû à une différence de ressources entre l'émetteur et le récepteur, en termes de puissance de calcul, de mémoire ou de bande passante disponible. Par exemple, si l'un d'entre eux est un superordinateur et l'autre un téléphone portable.



On peut imaginer plusieurs mécanismes pour implanter le service de contrôle de flux. Si la saturation est un événement relativement rare, on peut utiliser un mécanisme réactif dans lequel le récepteur indique sa saturation et demande à l'émetteur d'arrêter d'émettre. (L'équivalent du « attends un peu » humain.) Il utilisera par la suite une invitation à émettre afin de relancer la communication lorsque des ressources seront de nouveau disponibles. Du fait du temps d'acheminement du message entre le récepteur et l'émetteur, plusieurs unités de données circulant dans l'autre sens peuvent être perdues. Ce type de mécanisme est utilisé dans des protocoles tels que HDLC ou Ethernet. Si la saturation est un événement plus fréquent, le récepteur peut, au contraire, indiquer par avance à l'émetteur combien d'unité



de données ou quelle quantité de données il est prêt à accepter. Cette stratégie est adoptée dans les protocoles TCP et HDLC.

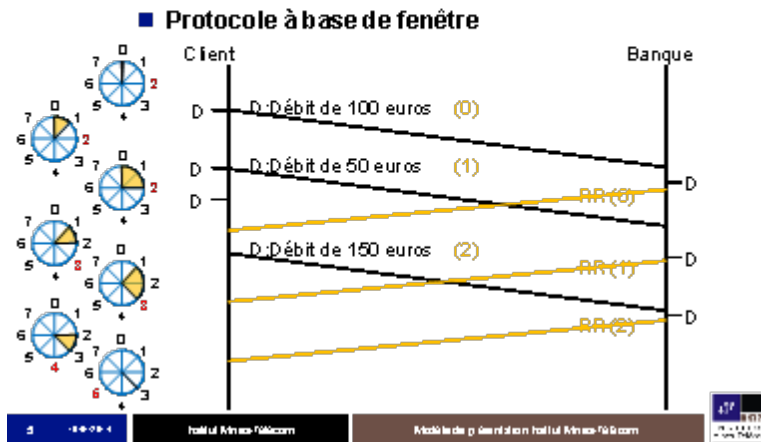
## Contrôle préventif

### ■ Protocole à base de fenêtre

- Les numéros de séquence.
- Borne inférieure (0).
- Borne supérieure (2).
- Taille de la fenêtre ( $2-0=2$ )
- Taille maximale de la fenêtre (3)



Dans ces protocoles, ceci se traduit par l'utilisation d'un objet appelé fenêtre et défini par un intervalle. Lors de son utilisation dans le protocole HDLC, chaque unité de données est numérotée. La fenêtre permet à l'émetteur de contrôler les numéros des unités de données qu'il peut émettre sans saturer le récepteur. Lors de chaque émission, la borne supérieure de la fenêtre est incrémentée. Lors de la réception de cette unité de données, le récepteur envoie un acquittement à l'émetteur. Dans HDLC, cet acquittement est appelé RR pour « Receive Ready », « prêt à recevoir ». Lors de sa réception, la borne inférieure de la fenêtre est incrémentée. La taille de la fenêtre, c'est-à-dire la différence entre les bornes supérieure et inférieure, indique le nombre d'unités de données envoyées mais non encore acquittées. Par ailleurs, la taille de la fenêtre est limitée par une taille maximale. Celle-ci correspond au nombre d'unités de données que le récepteur peut recevoir sans être saturé. Dans HDLC, sa valeur initiale est décidée au démarrage de la communication. Elle peut varier dans d'autres protocoles.



Examinons un exemple dans lequel la taille maximale de la fenêtre est de 2. Le compteur est ici codé sur 3 bits. Les identifiants seront donc codés de 0 à 7. Au démarrage, l'émetteur peut en théorie émettre les unités de données numérotées de 0 à 1. Les valeurs des bornes inférieure et supérieure de la fenêtre sont égales à 0. Lorsqu'il reçoit des données de l'utilisateur, il vérifie que la taille de la fenêtre n'est pas supérieure à sa taille maximale. Comme cette taille est ici de 0, il utilise la borne supérieure pour numéroté l'unité de données. L'unité de données D(0) est alors émise. Il incrémente ensuite la borne supérieure à 1. L'émetteur reçoit ensuite de nouvelles données de la part de l'utilisateur. Il vérifie de nouveau que la taille de la fenêtre n'excède pas sa taille maximale. La valeur étant cette fois-ci de 1, D(1) est envoyée et la borne supérieure de la fenêtre est de nouveau incrémentée à 2. Des données sont reçues de nouveau depuis l'utilisateur. La taille de la fenêtre est maintenant de 2. Comme sa taille maximale est atteinte, l'émission est retardée jusqu'à la réception d'un acquittement. Cet acquittement RR(0) arrive un peu plus tard. La borne inférieure de la fenêtre est alors incrémentée à 1. La taille de la fenêtre est alors de  $2 - 1$ , c'est-à-dire 1. Ceci permet l'émission des données au travers de D(2) et l'incrémement de la borne supérieure qui passe de 2 à 3. Les acquittements RR(1) et RR(2), lorsqu'ils sont reçus par l'émetteur, provoquent de nouveau l'incrémement de la borne inférieure qui passe successivement de 1 à 2 puis de 2 à 3. La taille de la fenêtre est alors de nouveau nulle.

## Quizz

- On suppose que l'on utilise une taille maximale de fenêtre de 3. L'émetteur reçoit une PDU RR 4 alors que sa fenêtre est dans l'état.

Quel est la forme de la fenêtr après ?

A ☐




C ☐



B ☐



D ☐

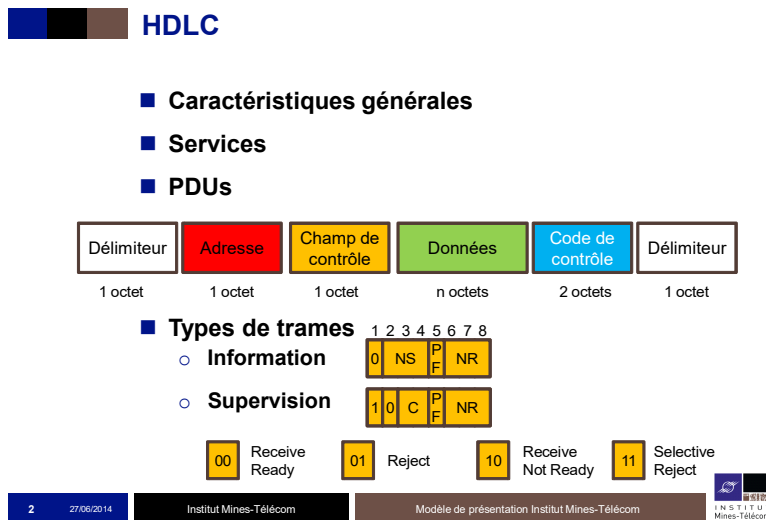


Pour continuer, voici un petit exercice. On suppose que l'on utilise le protocole de contrôle de flux dont nous venons de parler. On suppose, chez l'émetteur, une taille maximale de fenêtre de 3. L'émetteur reçoit une PDU RR(4) alors que sa fenêtre est dans l'état que nous voyons ici à l'écran. Quel est la forme de la fenêtr après cette réception ? Réponse A, réponse B, réponse C ou réponse D ?

La bonne réponse est la réponse D. En effet, à la réception de RR(4), la borne inférieure est incrémentée de 1, passant de 4 à 5. Par ailleurs, la valeur maximale pouvant être prise par la borne supérieure est également incrémentée de 1 et passe donc à 0, les valeurs étant codées sur 3 bits.

## Leçon 6 : HDLC, partie 1

Nous avons vu, dans les séances précédentes, comment différents services pouvaient être rendus au travers de protocoles indépendants les uns des autres. Nous allons voir, lors de cette séance, un protocole complet : le protocole HDLC.



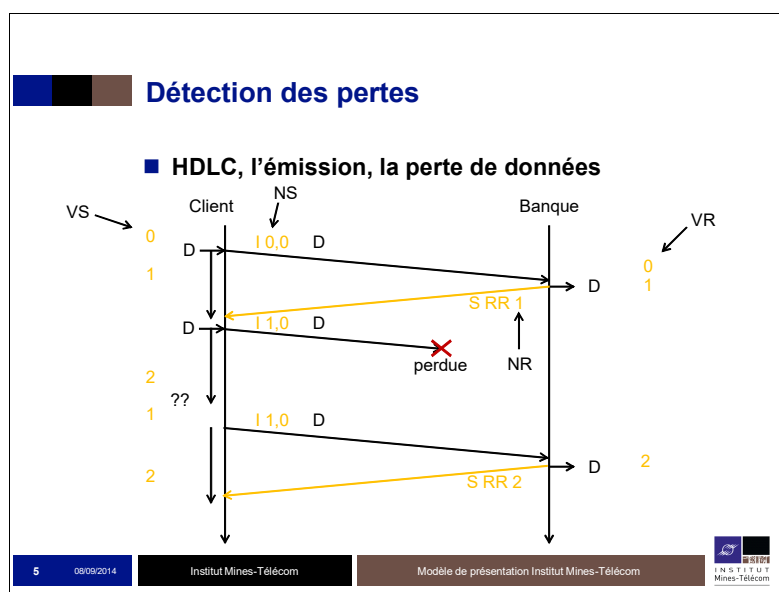
Le protocole HDLC, pour « High-Level Data Link Control », est un protocole de niveau « liaison de données » développé dans les années 70. D'abord utilisé très largement dans les réseaux étendus tels que X25, il a inspiré par la suite de nombreux autres protocoles. Le protocole HDLC peut fonctionner dans plusieurs modes. Celui qui nous intéresse aujourd'hui étant le mode appelé asynchrone équilibré. Ce mode permet la communication entre deux systèmes.

Le protocole HDLC fournit un certain nombre de services : la délimitation et la transparence, la détection des erreurs, la détection et la reprise sur perte, et le contrôle de flux. Pour cela, HDLC utilise un format d'unité de données composé de plusieurs champs : les délimiteurs ayant une valeur binaire de 01111110 sont placés en début et en fin d'unité de données. Un champ « adresse » qui n'a pas de réelle utilité dans le mode du protocole que nous étudions. Un champ de contrôle, que nous détaillerons plus tard, qui permet, entre autres, de distinguer trois types de trames : les trames d'information, les trames de supervision et les trames non numérotées. Comme leur nom l'indique, les trames d'information sont utilisées pour transporter les informations utiles (SDU) reçues depuis la couche supérieure. Ces informations sont alors placées dans le champ de données qui suit le champ de contrôle. Ce champ de données n'existe pas pour les autres types de trames. On trouve enfin un code de contrôle de type CRC calculé sur tous les champs, à l'exception des fanions, et permettant de rendre le service de détection d'erreur. Intéressons-nous maintenant au champ de contrôle. Le type de trame est codé au travers des deux premiers bits du champ de contrôle. Le

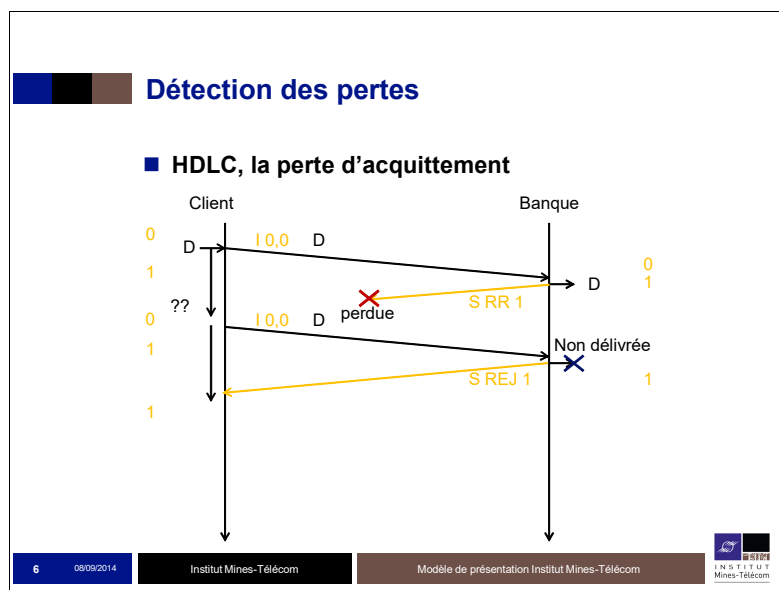
premier bit à 0 indique une trame d'information notée I ; les bits 10, une trame de supervision notée S ; les bits 11, une trame non numérotée notée U. Comme nous l'avons dit précédemment, les trames d'information permettent de transporter les informations reçues depuis la couche supérieure. Les trames non numérotées permettent principalement de gérer la communication entre les deux systèmes. Enfin, les trames de supervision permettent de rendre les services de détection de perte, de reprise sur perte et de contrôle de flux. Afin de mettre en œuvre les services dont nous venons de parler, chaque type de trame possède un champ de contrôle avec un format propre.

Regardons maintenant les règles de fonctionnement du protocole. Dans HDLC, les éléments les plus originaux sont liés à la détection et au traitement des pertes. Les unités de données (PDU) contenant des informations utiles sont numérotées, comme dans le cadre du protocole du bit alterné. Ce numéro, associé au compteur NS des trames I, est codé sur 3 bits. Afin de savoir quel est le prochain numéro à utiliser, l'émetteur maintient une variable, appelée VS, qui est incrémentée à chaque nouvelle émission de trame I. Lorsqu'une trame est reçue correctement par le récepteur, il acquitte sa réception au travers d'une trame de supervision appelée RR pour « Receive Ready ». Dans cette trame, le récepteur indique au travers du champ NR, dans le champ de contrôle, le numéro de la prochaine trame attendue.

Afin de savoir quel est ce numéro, le récepteur maintient une variable appelée VR qui est incrémentée à chaque fois qu'une trame est reçue correctement. On considère qu'une trame est reçue correctement lorsque son numéro correspond au prochain numéro de trame attendu. C'est-à-dire que l'on considère que les trames doivent être reçues dans l'ordre. Comme dans le cadre du bit alterné, la perte de trame peut être détectée en associant un minuteur à l'émission de chaque trame. Lors de l'expiration du minuteur, la variable d'émission VS reprend la valeur qu'elle avait avant l'émission de la trame perdue, et celle-ci est réémise.



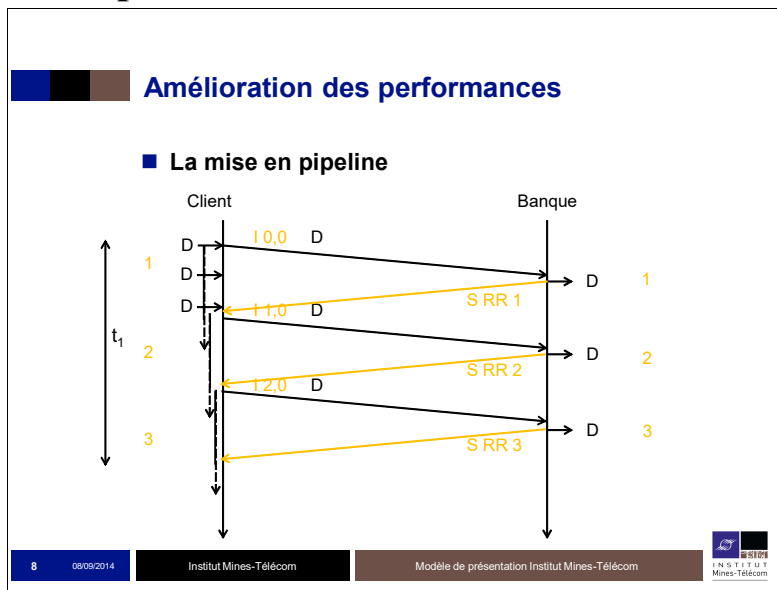
Nous voyons ici un exemple montrant un échange qui illustre ces éléments du protocole. Lorsqu'il reçoit des données de la couche supérieure, l'émetteur enclenche le minuteur et envoie la trame I 0,0. C'est-à-dire une trame d'information dans laquelle la valeur du champ NS représenté par le premier chiffre a pris la valeur de VS, et est donc de 0. Nous expliquerons plus tard le rôle du deuxième zéro présent ici. Il incrémente ensuite la variable VS. Lors de la réception de cette trame, le récepteur vérifie que la valeur de NS est égale à VR. Comme c'est le cas, la trame est considérée comme reçue correctement et le récepteur incrémente son compteur VR puis envoie un acquittement au travers de la trame de supervision RR 1. Dans cette trame, la valeur du champ NR a pris la valeur de VR. Enfin, son contenu est délivré à la couche supérieure. L'acquittement est reçu par l'émetteur. Celui-ci reçoit alors de nouvelles données depuis la couche supérieure. Comme précédemment, il enclenche le minuteur et envoie la trame I 1,0 et incrémente ensuite la variable VS à 2. Cette deuxième trame d'information se perd. Le minuteur se déclenche donc au bout d'un certain temps. La valeur de VS reprend alors la valeur avant l'émission de la trame associée au minuteur, c'est-à-dire 1. Le minuteur est de nouveau enclenché puis la trame est réémise. Enfin, le compteur VS est incrémenté, passant de 1 à 2. Cette fois, la trame arrive à destination. Comme précédemment, le récepteur compare la valeur du champ VR au champ NS de la trame reçue. Les valeurs étant les mêmes, il incrémente VR puis envoie un acquittement au travers de la trame de supervision RR 2 puis délivre le contenu de la trame à la couche supérieure. RR 2 est ensuite reçue par l'émetteur.



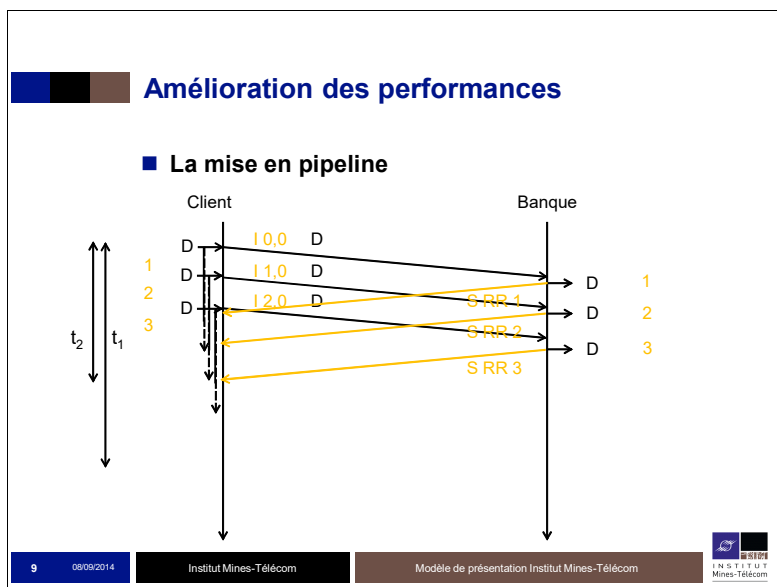
Le compteur VR présent chez le récepteur, et dont nous avons déjà parlé, permet de régler le problème des duplicatas. Ainsi, si l'émetteur réemet une trame déjà reçue par le récepteur, le récepteur se rend compte, en comparant le numéro de la trame NS à son compteur VR, que ce n'est pas la trame qu'il attend. Dans cette situation, le récepteur envoie une trame d'indication d'erreur de type REJ dans laquelle le champ NR prend la valeur de VR ; c'est-à-dire indique le prochain numéro de trame attendu et ignore par la suite les trames d'information ne portant pas le numéro attendu.

Voyons, au travers d'un exemple, comment cela fonctionne. Comme dans notre exemple précédent, la trame initiale parvient au récepteur qui l'acquitte. Les variables sont incrémentées de part et d'autre. Cependant, l'acquittement RR 1 se perd. Le minuteur se déclenche donc chez l'émetteur qui remet la variable VS à 0, réenclenche le minuteur, et retransmet I 0,0. Lors de la réception de cette unité de données, le récepteur compare VR à NS et se rend compte que les deux valeurs 0 et 1 sont différentes. Il laisse donc VR inchangé, ne délivre pas les données à la couche supérieure, et renvoie un message de rejet REJ 1.

## Leçon 6 : HDLC, partie 2



Une des améliorations apportées par HDLC est la mise en pipeline. Dans le protocole du bit alterné, une unité de données ne circule de l'émetteur vers le récepteur que lorsque la précédente a été acquittée. HDLC peut également fonctionner selon ce principe, comme nous le voyons ici. Cependant, ceci limite le débit effectif entre l'émetteur et le récepteur puisque l'émetteur passe du temps à attendre les acquittements sans rien faire. Ainsi, nous voyons ici que la transmission des trois trames d'information utilisant ce principe prend le temps  $t_1$ .

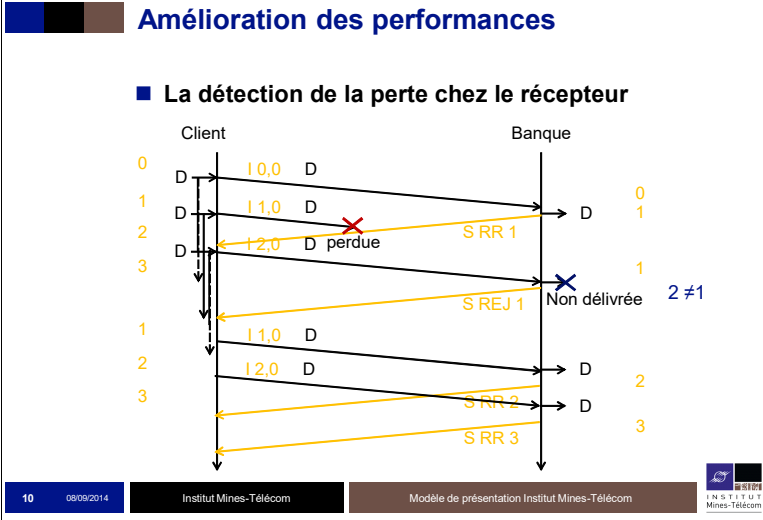


Dans le cas d'une mise en pipeline, les trames d'information peuvent être émises sans attendre ; ce qui réduit le temps total de transmission<sup>1</sup> à  $t_2$ ,  $t_2$  plus court que  $t_1$ . L'existence

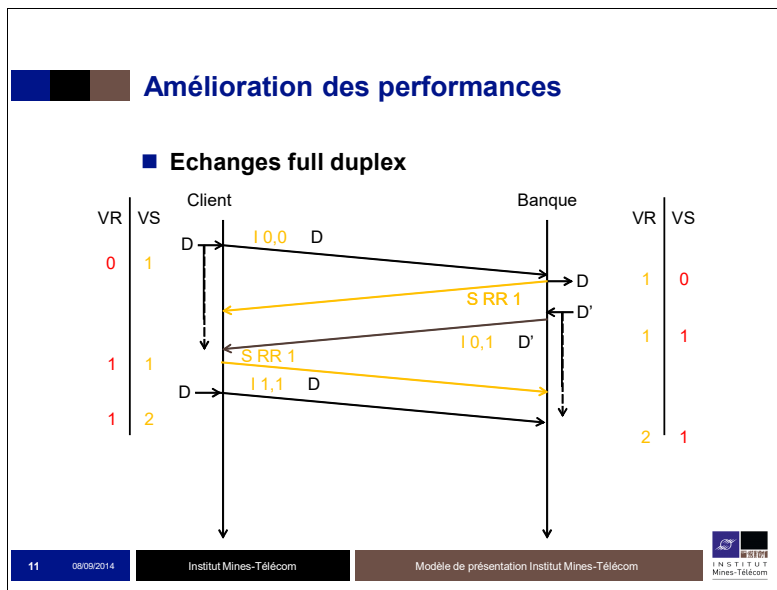
<sup>1</sup> Le temps de transmission désigne ici le temps nécessaire à l'acheminement des trois unités de données et à leur acquittement.



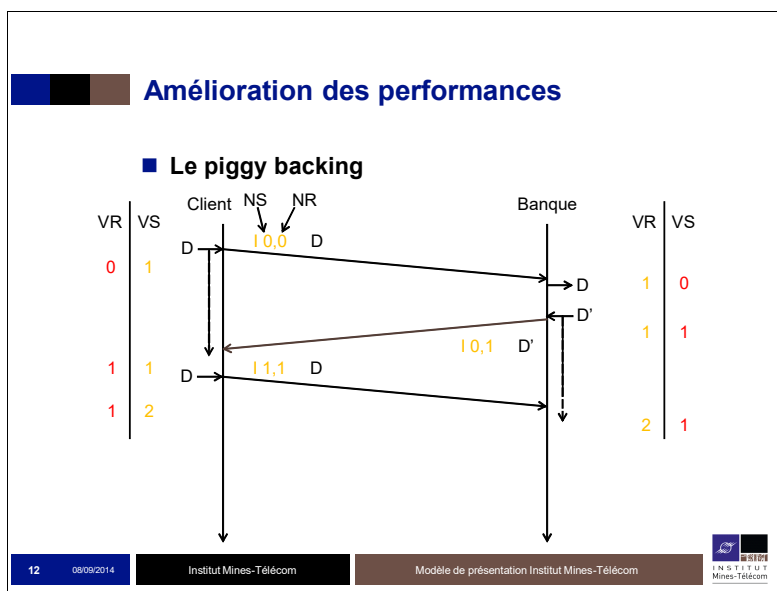
\_\_\_\_\_



La possibilité d'envoyer plusieurs trames d'information au récepteur sans attendre leur acquittement pose cependant un problème. Que faire lorsqu'une trame d'information parmi l'ensemble des trames envoyées se perd ? Dans la version du protocole HDLC que nous étudions, cela se traduit par l'arrivée d'une trame portant un numéro NS plus grand que le numéro attendu VR (2 par rapport à 1 dans notre exemple). Le récepteur considère alors la trame comme non reçue correctement, ignore son contenu, et envoie, comme nous l'avons vu précédemment, un message de rejet REJ associé au numéro de trame attendu. Lorsque ce message arrive à l'émetteur, il donne à la variable VS la valeur contenue dans le champ NR, c'est-à-dire 1. Il désarme les minuteurs en cours et renvoie immédiatement les messages leur étant associés à partir de ceux numérotés VS. Ce mode de reprise sur perte est appelé « go back n » puisque l'émetteur retourne dans l'état où il était avant l'émission de la trame d'information numérotée NR. Ce mode de fonctionnement fait alors porter la détection de la perte chez le récepteur au travers de la découverte d'un « trou » entre deux numéros de séquence de trames reçues consécutivement. L'émetteur est informé de cette découverte au travers de la trame REJ. Ce fonctionnement permet généralement d'accélérer la reprise sur perte par rapport à l'utilisation du minuteur chez l'émetteur.



Nous avons supposé jusqu'à présent que l'information circulait uniquement dans le sens client vers banque. Dans la pratique, la banque peut évidemment avoir également envie de transmettre de l'information au client. HDLC offre cette possibilité. Afin de permettre la gestion de la communication dans les deux sens, chaque entité maintient deux variables, VS et VR, contrôlant respectivement les trames d'information transmises et reçues. Ces deux nouvelles variables sont représentées ici en rouge. Dans cet exemple, chaque trame d'information est acquittée au travers d'une trame de supervision RR.



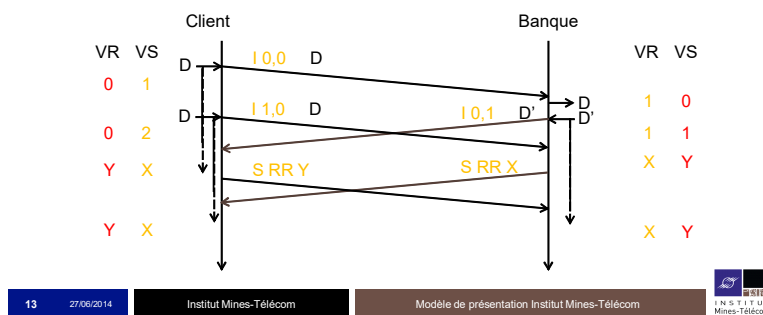
La possibilité de transmettre des trames d'information dans les deux sens est utilisée par HDLC pour une autre amélioration appelée « piggy backing ». L'idée est d'utiliser les trames d'information afin de réaliser les acquittements sans utiliser de trame de contrôle. Ce mécanisme est donc utilisé par le récepteur lorsque celui-ci doit envoyer un acquittement à l'émetteur et doit envoyer à peu près au même moment une trame d'information à celui-ci. A cet effet, les trames d'information possèdent dans le champ de contrôle un champ NR

dans lequel le récepteur indique le numéro de la prochaine trame d'information attendue, c'est-à-dire en y copiant la valeur de la variable VR. La valeur de ce second champ est indiquée dans nos échanges par la deuxième valeur numérique précédant le I. Lorsqu'il reçoit une trame d'information, l'émetteur considère que toutes les trames d'information d'un numéro inférieur à la valeur contenue dans NR ont été reçues par le récepteur et désactive les minuteurs associés.

## Quizz

■ En supposant que l'on ait l'échange suivant entre les deux entités, quelles sont les valeurs de X et Y ?

- ☐ 1 et 1   
 ☐ 2 et 1   
 ☐ 1 et 2   
 ☐ 2 et 2



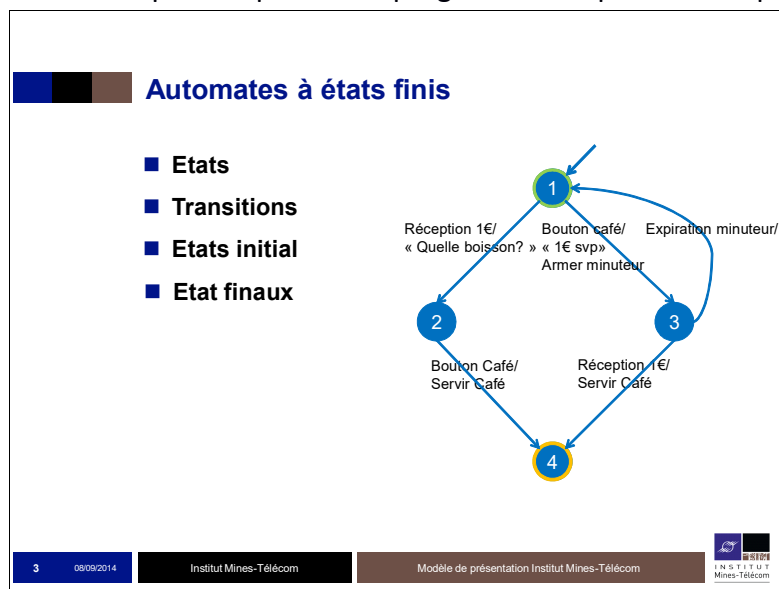
Un petit exercice pour continuer cette séance. En supposant que l'on ait l'échange suivant entre les deux entités, quelles sont, selon vous, les valeurs de X et Y ? 1 et 1, 2 et 1, 1 et 2, ou 2 et 2 ?

La bonne réponse est 2 pour X et 1 pour Y. En effet, la banque, lorsqu'elle reçoit la trame I 1,0, incrémente VR (c'est-à-dire X) de 1 à 2. VS (c'est-à-dire Y) reste inchangé. Du côté du client, VR (c'est-à-dire Y) est incrémentée de 0 à 1 après la réception de I 0,1. VS (c'est-à-dire X) reste inchangé. L'émission et la réception des trames de supervision RR n'a pas d'influence sur les variables.

## Leçon 7: La modélisation des protocoles

Nous l'avons vu, lors des séances précédentes, les protocoles peuvent dans certains cas être relativement complexes. Afin de pouvoir raisonner plus facilement sur ceux-ci, il est nécessaire de les modéliser. C'est ce que nous allons voir lors de cette séance.

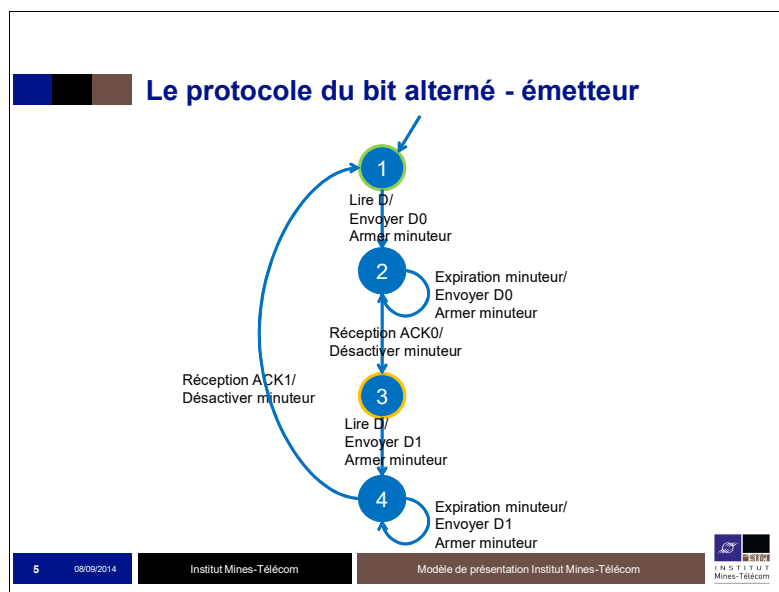
Quelles sont les qualités que l'on attend d'un modèle ? Il doit permettre de représenter certaines caractéristiques des protocoles de manière simple, synthétique et non ambiguë. À quoi cela sert-il ? Cela peut servir pour ceux qui les utilisent à raisonner plus facilement sur les protocoles. Ces modèles peuvent également être utilisés par des programmes d'analyse pour vérifier certaines propriétés protocolaires, comme le fait que le protocole se termine, ou que tel ou tel type d'unité de données est utilisé. Ils peuvent aussi être utilisés afin de créer de manière automatique une partie des programmes implantant les protocoles.



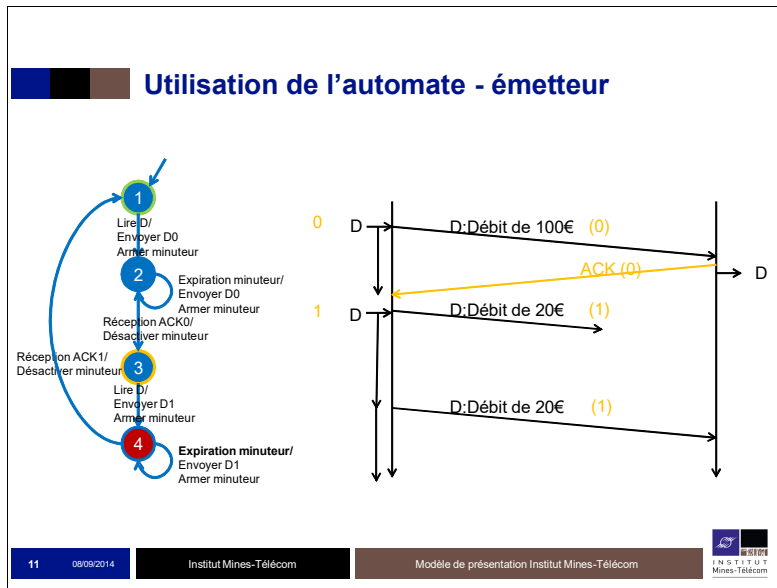
Parmi les modèles existants, nous allons aujourd'hui étudier les automates. Ceux-ci sont utilisés dans de nombreux domaines, parfois différents des réseaux de données. Nous nous intéressons, par exemple ici, à la modélisation du protocole d'utilisation d'un distributeur de café. Un automate à nombre fini d'états se définit au travers de deux éléments principaux : un ensemble d'états souvent représentés graphiquement par un cercle et un ensemble de transitions représentées par des flèches. À chaque transition est associé un événement : réception d'une pièce d'un euro, expiration d'un minuteur, appui sur une touche de la machine pour choisir une boisson... Cet événement permet de passer d'un état à l'autre. On peut également attacher aux transitions, les actions qui sont associées à chaque événement comme, par exemple, l'affichage d'un message, l'armement d'un minuteur, la préparation de la boisson choisie.... Ceux-ci sont notés après l'événement. Parmi les états, on distingue par ailleurs l'état initial qui correspond à l'état dans lequel se trouve le système avant exécution

du protocole, et un ou plusieurs états finaux correspondant aux états dans lesquels le système peut se trouver en fin d'exécution.

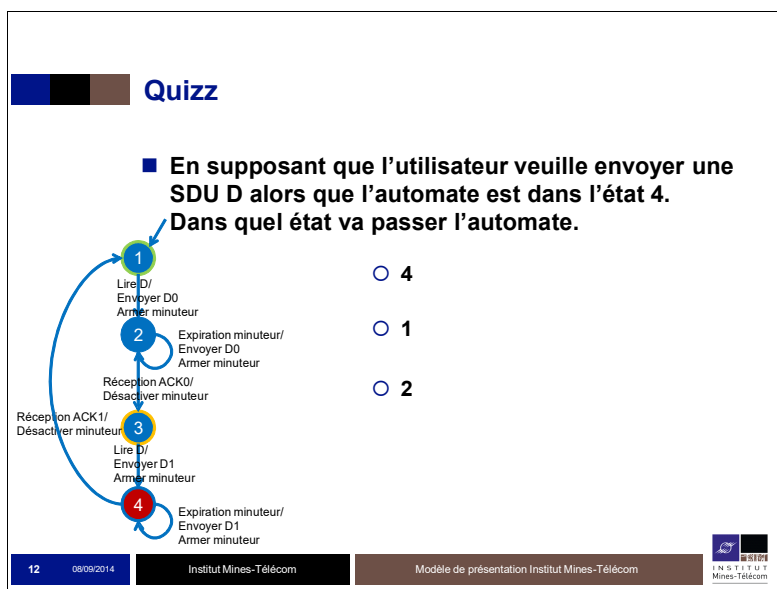
Nous avons vu, il y a peu de temps, le protocole du bit alterné. Essayons de modéliser celui-ci en utilisant ce formalisme. Les deux entités peuvent échanger quatre unités de données : une unité contenant des données associée à un compteur valant '0' ou '1', et une unité contenant un acquittement associée à un compteur valant lui aussi '0' ou '1'. Pour simplifier, nous appellerons ces unités de données : D0, D1, et ACK0, ACK1. L'émetteur peut être soumis à plusieurs événements : la réception d'une unité de donnée D de la couche supérieure, la réception d'un ACK0, la réception d'un ACK1 et enfin l'expiration du minuteur. Il peut par ailleurs lui-même initier des événements en armant un minuteur, en envoyant D0 et en envoyant D1.



Il est possible de représenter le protocole de bit alterné du côté de l'émetteur en utilisant 4 états. Comme on le voit sur notre schéma, dans l'état initial 1, l'entité attend des données de la couche supérieure. Lorsqu'elle en reçoit au travers de l'événement « Lire D », elle envoie D0 au récepteur, arme le minuteur et passe dans l'état 2. Dans cet état, deux événements peuvent se produire. Soit le minuteur expire, auquel cas l'entité estime que D0 s'est perdue. D0 est donc retransmise et le minuteur réarmé. L'entité reste alors dans l'état 2 et cette situation peut se reproduire indéfiniment. La deuxième possibilité est qu'un ACK 0 soit reçu, signifiant que D0 a été reçu par son destinataire. Le minuteur est alors désarmé et le compteur passé à 1. L'entité passe ensuite dans l'état 3. Dans cet état, la situation est similaire à celle rencontrée dans l'état 1, à l'exception du fait que nous utilisons une valeur de compteur de 1. De manière similaire, l'état 4 est semblable à l'état 2 et permet, lors de la réception de ACK 1, de revenir à l'état 1. Les états 1 et 3 peuvent représenter une fin de communication si aucune autre information complémentaire n'est reçue depuis la couche supérieure.



Utilisons maintenant cet automate pour voir comment celui-ci fonctionne. Dans l'état initial 1, une SDU est reçue de la couche supérieure. Elle permet de passer à l'état 2 en provoquant l'envoi de D0 et l'armement du minuteur. Lors de la réception du ACK0, l'automate passe de l'état 2 à l'état 3 et provoque le désarmement du minuteur. Lorsqu'une nouvelle SDU est reçue depuis la couche supérieure, l'automate passe de l'état 3 à l'état 4 ce qui provoque l'envoi de D1 et l'armement du minuteur. Aucun acquittement n'ayant été reçu, le minuteur se déclenche, ce qui provoque la retransmission de D1 et le réarmement du minuteur. L'automate reste ainsi dans l'état 4.



Pour terminer cette séance, un petit exercice. En supposant que l'utilisateur veuille envoyer une SDU D alors que l'automate est dans l'état 4 pour le protocole du bit alterné, dans quel état va passer l'automate selon vous ? L'état 4, l'état 2 ou l'état 1 ? La réponse est l'état 4. En effet, aucune transition ne permet de sortir de 4 au travers de l'événement « Lire D ». D ne sera donc lu que lorsque l'automate sera revenu dans l'état 1.

