UNIVERSITE D'EVRY VAL D'ESSONNE

Fonctions de la couche liaison

- Offrir une interface clairement définie à la couche réseau
- Traiter les erreurs de transmission
- Réguler le flux des données échangées

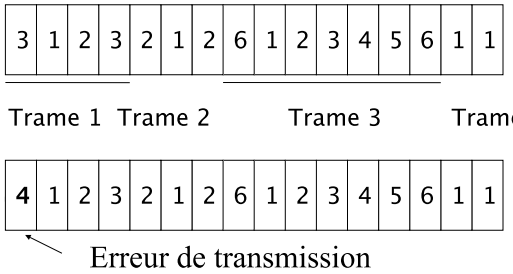
- Moyens utilisés:
  - Découpage en trames
    - Les données de la couche réseau sont encapsulées dans une trame contenant
      - Un entête (header)
      - Les données (LSDU = RPDU)
      - Un enqueue (trailer)
  - Contrôle de flux
  - Détection et correction des erreurs

UNIVERSITE D'EVRY VAL D'ESSONNE

Notion de trame

- Couche physique:
  - Transport d'un flux de bits
- Pb: détection des erreurs: bits et nombre de bits erronés
  - Solution: découpage des données en trames contenant une somme de contrôle (checksum)
- Découpage:
  - séparer les trames par un temps de silence (impossible à garantir à la réception)
  - Compter les caractères

Découpage en trames: ajout d'un nombre de caractères



- L'entête contient le nombre de caractère de la trame
- On peut donc prévoir le début de la trame suivante
- Pb: en cas d'erreur sur le nombre de caractères : on ne peut plus retrouver le début des trâmes suivantes

- L'entête contient le nombre de caractères de la trame.
- En lisant cette information, la couche liaison de la station destination sait quand commence la trame suivante
- En cas d'erreur de transmission corrompant le nombre de caractères d'une trame, on n'a aucun moyen de retrouver le début des trames suivantes

Utilisation d'indicateur de début/fin de trame

- Chaque trame commence par : 01111110

Exemple:

- NPDU (=LSDU) contient : 01101111110110
- Transmis sur la ligne:
  - ***01111110*** 01101111101011***01111110***
- Transmis à la couche réseau destination:
  - 01101111110110

- On utilise un indicateur de début et de fin de trame: 01111110
- Pb: que faire si les données contiennent une séquence identique à l'indicateur. Risque de confusion
- Solution: modifier les données à l'envoi et à la réception:
  - ➔ A l'émission :si on rencontre 5 fois '1', on le complète par un zéro
  - ➔ À la réception: si on rencontre 5 fois '1' suivi d'un zéro, on supprime le zéro
- En cas d'erreur de transmission sur l'indicateur, il suffit de rechercher l'indicateur suivant pour se resynchroniser sur la trame suivante.

UNIVERSITE D'EVRY VAL D'ESSONNE

Détection/correction d'erreurs

- idée: de la redondance pour permettre
  - de détecter les erreurs
  - de corriger les erreurs
- une trame est alors composée:
  - de m bits de données
  - de r bits supplémentaires (redondance)
  - et forme un ensemble de m+r=n bit
- Pb: déterminer r et les r bits supplémentaire s en fonction de notre but (détection/correction)

UNIVERSITE D'EVRY VAL D'ESSONNE

Distance de Hamming

- distance de Hamming: nombre de bits différents entre deux trames
- Comment: en faisant un OU exclusif entre les deux trames
- Ex.: 10001010 XOR 11010010=01011000 : 3 bits différents
- 2^m possibilités pour les données mais par construction, on s'interdit d'utiliser 2^r possibilités pour les bits de contrôle
- distance de Hamming du code complet = min d(t1, t2) pour tous t1, t2 trames valides différentes du code complet

Si la distance de Hamming de deux mots est d, alors, il faut modifier d bits pour passer de l'un à l'autre. Dit autrement : il faut d erreurs simples pour passer de l'un à l'autre.

Distance de hamming du code complet: on peut lister toutes les trames possibles et calculer la plus petite distance entre trames. Cette plus petite distance est la distance de Hamming du code complet.

Exemple: considérons le code suivant :

00000000, 00001111, 11110000, 11111111.

La distance entre mots vaut 4 ou 8 suivant les cas. La distance de Hamming du code est donc 4.

UNIVERSITE D'EVRY VAL D'ESSONNE

Distance de Hamming

- distance de Hamming du code complet = min d(t1, t2) pour tous t1, t2 trames valides différentes du code complet
- détecter d erreur => distance de Hamming du code > d
- corriger d erreur => distance de Hamming du code > 2*d

Si la distance de Hamming de deux mots est d, alors, il faut modifier d bits pour passer de l'un à l'autre.

détecter une erreur, c'est détecter que la version modifiée d'un mot n'appartient pas à notre code. Si un mot subit de 1 à d erreurs, alors sa distance de Hamming au mot d'origine est au plus de d. Si la distance de Hamming du code complet est strictement supérieure à d,le mot obtenu ne peut pas être un mot du code.

Pour la correction, on souhaite retrouver le mot d'origine. Dans un code de distance de Hamming strictement supérieure à 2*D : Si un mot a subi au plus d erreur alors sa distance au mot d'origine est au plus de d et sa distance à tous les autres mots est strictement supérieure à d. Soit mi le mot initial, me le mot erroné et m un mot du code différent de mi. On a 2*d+1 <=d(mi, m) et d(mi, m) <= d(mi, me) + d(me, m). On en déduit 2*d+1 – d(mi, me) <= d(me, m). De d(mi, me) <=d, on déduit d+1 <= d(me, m) puis d(mi, me) < d(me, m) qui est le résultat cherché.

Donc si la distance du code est supérieure ou égale à 2*d+1, le mot le plus proche du mot erroné est le mot correct. On peut corriger jusqu'à d erreurs.

Dans notre exemple: La distance de Hamming du code est 4.

00000000, 00001111, 11110000, 11111111.

On peut détecter 3 erreurs (d+1=4) et corriger les erreurs simples (2*d+1=3 donc d=1).

Si le mot reçu est : 00001011. On détecte que c'est une erreur car il ne fait pas partie de notre code. Le mot donc il est le plus proche est 00001111 qui est donc le mot correct. On a corrigé une erreur.

UNIVERSITE D'EVRY VAL D'ESSONNE

bits de parité/CRC

- bits de parité:
  - méthode de détection d'erreur: détecte une erreur
  - on ajoute un bit que l'on positionne de façon à ce que la parité de l'ensemble soit paire
  - exemple: 10001010: trois 1 pour avoir une parité paire, on doit ajouter un 1: 100010101
  - exemple: 10001000: deux 1 pour avoir une parité paire, on doit ajouter un 0: 100010000
- CRC (code polynomial ou contrôle de redondance cyclique):
  - les bits sont vus comme les coefficients d'un polynome

la description détaillée du fonctionnement des CRC sort du cadre de ce cours.

UNIVERSITE D'EVRY VAL D'ESSONNE

Exemples de protocoles élémentaires

- Modèle
  - Les couches réseau, liaison et physique sont indépendantes et communiquent par messages
  - Les transmissions de données se font de A vers B
  - Les seules erreurs sont des erreurs d e transmission (pas de panne des ordinateur notamment)

UNIVERSITE D'EVRY VAL D'ESSONNE

Protocole simplex non restreint

- Pas d'erreur de transmission
- Les données sont traitées sans délai par les couches réseau
- Pas de problème de contrôle de flux: on dispose d'autant de mémoire tampon que nécessaire

- Algo d'émission (coucher liaison de A)
  - While (true) {
    - Récupérer des données à envoyer de la couche réseau de 1
    - Créer une trame contenant ces données
    - La transmettre à la couche physique pour émission
  - }
- Algo de réception (couche liaison de B):
  - While (true) {/* boucle infinie)
    - Attendre un événement /* seul choix :arrivée d'une trame */
    - Récupérer le LPDU auprès de la couche physique de B
    - Transmettre le LSDU à la couche réseau
  - }

Protocole simplex arrêt et attente

- Canal bidirectionnel
- Pas d'erreur de transmission
- Les données sont traitées sans délai par les couches réseau
- contrôle de flux pour éviter de saturer le récepteur :
  - Le réception acquittera chaque trame reçue
  - L'émetteur attendra l'accusé de réception pour émettre la trame suivante
- Si l'aller/retour est long, on perd beaucoup en performances car rien n'est émis pendant l'attente de l'accusé de réception. Exemple: ligne rapide longue distance: vitesse élevée mais latence importante

- Algo d'émission (coucher liaison de A)
- While (true) {
  - Récupérer des données à envoyer de la couche réseau de 1
  - Créer une trame contenant ces données
  - La transmettre à la couche physique pour émission vers B
  - Attendre un événement /* on attend le ack */

  → }
- Algo de réception (couche liaison de B):
  - → While (true) {/* boucle infinie)
    - Attendre un événement /* seul choix :arrivée d'une trame */
    - Récupérer le LPDU auprès de la couche physique de B
    - Transmettre le LSDU à la couche réseau
    - Créer l'accusé de réception
    - Le transmettre à la couche physique pour émission vers A

Protocole simplex sur canal bruité

- Canal bidirectionnel
- Canal bruité: erreurs de transmission
- Les données sont traitées sans délai par les couches réseau
- contrôle de flux pour éviter de saturer le récepteur :
  - Le réception acquittera chaque trame reçue
  - L'émetteur attendra l'accusé de réception pour émettre la trame suivante
  - Utilisation d'un numéro de séquence pour détecter les réémissions
  - Utilisation d'un timer pour réémettre une trame sans ack

- Solution incorrecte:
  - → La machine A déclenche un timer à l'émission d'une trame
  - → En cas d'erreur d'émission, la machine B n'émet de ack
  - → Si le timer de A expire, A réémet la trame
- Pb: si l'émission est correcte et si l'acquittement se perd, la machine B reçoit deux fois la trame sans le savoir et la transmet deux fois à sa couche réseau
- Solution: un numéro de séquence sur les trames pour détecter les réémissions de trame.
- Pb: quelle taille pour le numéro de séquence ? Le protocole arrêt et attente fait que la confusion ne peut porter que sur une trame et la suivante. Un numéro de séquence à deux valeurs (1 bit) suffit

Protocole simplex sur canal bruité

Algo d'émission (coucher liaison de A)

- NoTrameSuivante=0;
- Récupérer des données à envoyer de la couche réseau de A
- While (true) {
  - Construire une trame contenant les données et le NoTrameSuivante
  - La transmettre à la couche physique pour émission vers B
  - Lancer un timer
  - Attendre un événement /* on attend le ack ou la fin du timer */
  - Si l'événement est l'arrivée d'une trame:
    - Récupérer le LPDU auprès de la couche physique de A
    - Si le NoAck du LPDU est égal à NoTrameSuivante+1
      - Désactiver le timer
      - Récupérer des données à envoyer de la couche réseau de A
      - Incrémenter(NoTrameSuivante)
    - finSi
  - FinSi
- }

Protocole simplex sur canal bruité

- Algo de réception (couche liaison de B):
  - NoTrameAttendue=0
  - While (true) {/* boucle infinie)
    - Attendre un événement /* arrivée d'une trame ou erreur de checksum*/
    - Si l'événement est l'arrivée d'une trame
      - Récupérer le LPDU auprès de la couche physique de B
      - Si le No de séquence du LPDU est égal à NoTrameAttendue
        - Transmettre le LSDU à la couche réseau
        - Incrémenter NoTrameAttendue
    - Créer l'accusé de réception avec NoTrameAttendue comme No de séquence
    - Le transmettre à la couche physique pour émission vers A
  - }

➔ Le timer doit être suffisamment long pour permettre

- le transfert d'une trame à destination
- Son traitement
- Le transfert et la réception de l'acquittement