Chapitre 3 La couche liaison

- 1. Introduction
  - Fonctions :
    - établissement et libération des connexions
      - si service orienté connexion
    - Contrôle d'accès dans le cas des réseaux locaux
      - Medium Access Control (MAC)
    - détection et correction d 'erreurs
    - contrôle de flux
    - structuration des données

1

2. Contrôle des erreurs

- erreurs « sur contenu » des données :
  - Des bits ou suites de bits sont modifiés par le canal
    - Erreur de décodage due aux bruits, parasites,…
  - Un canal a un taux d 'erreur bit (BER)
  - Détection d'erreur basée sur le principe de la redondance
  - Normalement, taux résiduel (après contrôle) < BER
- erreurs « sur arrivée » :
  - pertes, déséquencements, etc.
  - leur détection est basée sur des mécanismes du protocole liaison (ex. numérotation)

2

2.1. Contrôle des erreurs sur contenu

- Codes détecteurs et éventuellement correcteurs d 'erreurs
  - ajout de bits de contrôle (redondance) aux bits d 'information
  - Rendement du code = Nb bits d'information/Nb bits total
  - Compromis entre rendement du code et taux d 'erreurs résiduel
- Exemples simples :
  - codes à répétition
    - on envoie deux fois le bloc d 'information
    - rendement de 50%
  - bit de parité
    - on ajoute à chaque caractère un bit tel que la somme de tous les bits modulo 2 soit 0 (parité paire) ou 1 (parité impaire)
    - rendement de 7/8

3

2.1. Contrôle des erreurs sur contenu

- parité longitudinale et transversale
  - on complète le bit de parité par caractère en réalisant un contrôle de parité longitudinal sur l 'ensemble des caractères
  - utilisé pour les bandes magnétiques
  - permet de corriger une erreur ou de détecter jusqu'à trois erreurs
- Recherche de codes à haut rendement et haut pouvoir détecteur(voire correcteur)
  - => théorie des codes
  - la facilité de codage/décodage par matériel ou logiciel est aussi un critère important
    - Décodage en temps réel des données reçues

4

2.1.1. Codes linéaires

- Blocs d'information à coder
- Mots codés par des vecteurs à n coefficients (bits)
  - Coefficients dans corps fini Z_2 des entiers modulo 2
  - algèbre modulo 2 (addition et multiplication)
- Un code linéaire de longueur n rajoute n-k bits de contrôle aux k bits d 'information
- Un code linéaire est engendré par une **matrice génératrice G** (de taille k,n)
  - X = i . G, avec i= (i_1, …, i_k) bloc d 'information
  - Code engendré par G : C = { X | X = i.G }
  - Si G = (I_k, A) : code systématique : X=(i,i. A)
  - i. A forme les n-k bits de redondance

5

2.1.1. Codes linéaires

- La **matrice de contrôle de parité H** (taille n-k, n):
  - permet de définir chacun des n-k bits de contrôle en fonction des bits d 'information
  - les n-k bits de contrôle sont définis par l 'équation :
  - $H . X^t = 0$
- $H = (A^t, I_{n-k})$
- $G. H^t = H . G^t = 0$
- G est déductible de H et réciproquement
- Tout code linéaire C vérifie les propriétés suivantes :
  - $0 \in C$
  - $\forall c_1, c_2 \in C, \ c_1 + c_2 \in C$ somme bit par bit
  - $\forall c \in C, \ \lambda . c \in C$ ($\lambda = 0$ ou 1) produit par un scalaire

6

2.1.1. Codes linéaires

- Décodage
  - soit Y le vecteur reçu : Y = X + E
  - X est le vecteur émis et E est le vecteur d'erreur
  - la détection d 'erreur est basée sur le syndrome du vecteur Y, défini par :
  - $s(Y) = H. Y^t$
  - $s(Y) = 0 \Leftrightarrow Y \in C$
  - si $s(Y) \neq 0$ alors le décodeur **détecte** qu 'il y a eu au moins une erreur
  - Rem : si $E \in C$ alors $s(E)=0$: erreur **indétectable**
  - si le code est aussi correcteur d 'erreurs alors le décodeur doit être capable, à partir du mot erroné Y, de déterminer quel est le mot X « le plus proche » dans le code
  - => **distance** d 'un code
  - Principe : erreur la plus probable porte sur le moins de bits

7

2.1.2. Distance de Hamming d 'un code

- Poids de Hamming d 'un mot :
  - c 'est le nombre de « 1 » qu'il contient
- Distance de Hamming entre deux mots
  - nombre de bits qui diffèrent :
  - d(X, Y) = Poids (X - Y) = Poids (X + Y)
- Poids minimum d 'un code :
  - c 'est le poids du (des) mot(s) non nul(s) de poids minimum
- Distance d 'un code
  - distance minimale entre deux mots quelconques du code :
  - $\text{Min}_{X \neq Y}$ (d(X,Y)) $\Leftrightarrow$ $\text{Min}_{X \neq Y}$ (Poids(X - Y))
  - = $\text{Min}_{X \neq 0}$ (Poids(X))

8

2.1.2. Distance de Hamming d 'un code

- la distance de Hamming d 'un code détermine sa capacité de détection et de correction
- Pour pouvoir détecter k erreurs (au plus) :
  - la distance d d 'un code doit vérifier : $d \geq k + 1$
  - k erreurs simples donnent un mot non autorisé
  - $\Leftrightarrow$ sphère de rayon k centrée sur un mot du code ne contient pas d'autre mot du code
- Pour pouvoir corriger k erreurs (au plus) :
  - la distance d d 'un code doit vérifier : $d \geq 2k + 1$
  - si k erreurs se produisent, le mot erroné reste plus proche du mot du code originel que de tout autre mot du code
  - $\Leftrightarrow$ les sphères de rayon k centrées sur les mots du code ont une intersection vide

9

2.1.2. Distance de Hamming d 'un code

- Cas général pour les codes correcteurs simples (1 erreur) :
  - soit un code dont les mots sont de longueur n
  - n = k + r (k bits d 'information, r bits de contrôle)
  - pour chacun des 2^k mots du code :
  - $\exists$ n mots non autorisés situés à une unité du mot du code
  - à chacune des 2^k combinaisons de bits d 'information on associe n + 1 mots de n bits
  - Le nombre total de mots possibles (appartenant ou non au code) est : 2^n
  - D 'où : $(n + 1). 2^k \leq 2^n \Leftrightarrow (k + r + 1). 2^k \leq 2^{k+r}$
  - $(k + r + 1) \leq 2^r$ ou $n + 1 \leq 2^{n-k}$

10

2.1.3. Distance de Hamming pour les codes linéaires correcteurs simples

- $Y = X + E$ et $s (Y) = H. Y^t = H. X^t + H. E^t = s (E)$
- si E = 0…1…0 (une seule erreur en position i)
- alors $s (E) = H. E^t = H_i$ (i-ème colonne de H)
- Donc pour corriger 1 erreur il faut et il suffit que toutes les colonnes de H soient non nulles et différentes entre elles
- Cette propriété de H (r, n) se traduit par :
- sachant qu 'il y a $2^r - 1$ combinaisons possibles pour les colonnes (toutes sauf 0) et que le nombre de colonnes est n = k + r, on obtient :
- $k + r \leq 2^r - 1 \Leftrightarrow (k + r + 1) \leq 2^r \Leftrightarrow (n + 1) \leq 2^{n-k}$

11

Codes de Hamming

- H matrice (n-k, n)
- Si H composé de toutes colonnes non nulles :
- => corrige une erreur
- Un code de Hamming (n,k) pour $2^{n-k}-1 = n$
- Exemples :
  - n-k=2 : code (3,1) répétition triple
  - n-k=3 : code (7,4)
  - n-k=4 : code (15,11)
- Remarque : d'après la relation $(n + 1) \leq 2^{n-k}$ les codes de Hamming ont le meilleur rendement pour n fixé.

12

2.1.4. Codes polynomiaux

- sous-ensemble des codes linéaires
- les vecteurs binaires sont identifiés à des polynômes à coefficients dans Z_2 (0,1)
- $i(x) = i_k.x^{k-1} + ... + i_1.x + i_0$
- un code polynomial est engendré par un polynôme g(x) : C = {p(x)/ p(x) = q(x). g(x), d° p ≤ n-1}
- la détection d 'erreurs est basée sur l 'utilisation du reste de la division polynomiale par g(x)
- soit i(x) le polynôme correspondant aux bits d'info
- on multiplie i(x) par x^r, avec r = d° g(x)
- x^r. i(x) = g(x). q(x) + R(x)
- r bits de contrôle

13

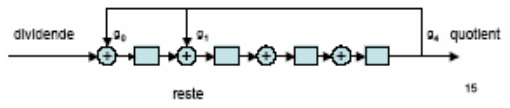
2.1.4. Codes polynomiaux

- On transmet le polynôme $T(x) = x^r$. i(x) + R(x)
  - Bits de i puis r bits de R(x) : code systématique
- Par construction, T(x) est divisible par g(x)
- Le décodage consiste à diviser le polynôme p(x) reçu par g(x)
  - p(x) ∈ C ⇔ p(x) modulo g(x) = 0 => pas d'erreur détectée
- Exemple de la division
- Ces codes sont appelés CRC (Cyclic Redundant Codes) :
  - avis V.41 de l 'UIT-T, $g(x) = x^{16} + x^{12} + x^5 + 1$
  - Trames Ethernet : CRC32 = $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
- Avantage des codes polynomiaux :
  - les opérations de multiplication et de division par un polynôme fixe sont facilement implémentables par des circuits logiques

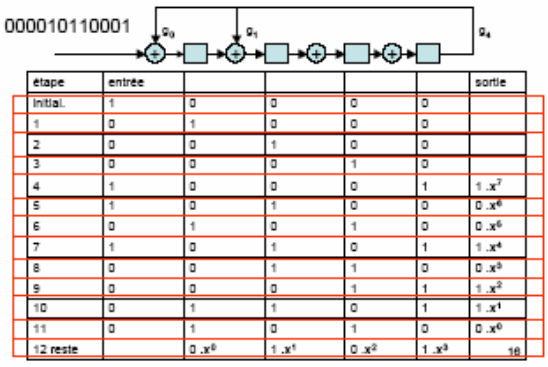
14

Circuit division polynômiale

- Registre à décalage avec retour (feedback)
  - circuit synchrone par top horloge
- le « retour » représente diviseur g(x)
  - ici g(x) = x4 + x + 1
- entrée séquentielle dividende à gauche
  - poids fort en tête
- sortie séquentielle du quotient à droite
- reste dans le registre à la fin



15



étape	entrée					sortie
initial.	1	0	0	0	0	
1	0	1	0	0	0	
2	0	0	1	0	0	
3	0	0	0	1	0	
4	1	0	0	0	1	$1.x^7$
5	1	0	1	0	0	$0.x^6$
6	0	1	0	1	0	$0.x^5$
7	1	0	1	0	1	$1.x^4$
8	0	0	1	1	0	$0.x^3$
9	0	0	0	1	1	$1.x^2$
10	0	1	1	0	1	$1.x^1$
11	0	1	0	1	0	$0.x^0$
12 reste		$0.x^0$	$1.x^1$	$0.x^2$	$1.x^3$	16

16

2.2. Contrôle des erreurs sur arrivée

- limites des codes détecteurs/correcteurs
  - les erreurs sur contenu ne sont pas toutes corrigées
  - les codes correcteurs sont coûteux en codage/décodage
  - => limite des solution purement **FEC** (Forward Error Correction)
- technique de retransmission
  - **ARQ** : Automatic Repeat reQuest
  - mise en oeuvre le protocole de liaison
  - permet de corriger :
    - une erreur « sur contenu » détectée (ex. CRC invalide) mais non corrigée assimilée à une perte
    - une erreur « sur arrivée » (ex. déséquencement, trou dans la numérotation)

17

2.2.1. Correction par retransmission avec arrêt et attente (« stop and wait »)

- L 'émetteur envoie un bloc d 'information et se met en attente d 'un **accusé de réception** positif **(ACK)** ou négatif (**NAK**).
- Si l 'accusé est négatif il réémet le bloc, sinon il émet le bloc suivant
- Si l 'acquittement n 'arrive pas au bout d 'un certain délai, une temporisation arrive à échéance (« **timeout** ») et le bloc est réémis
- Méthode simple, bien adaptée à une liaison half duplex
- Performance ?

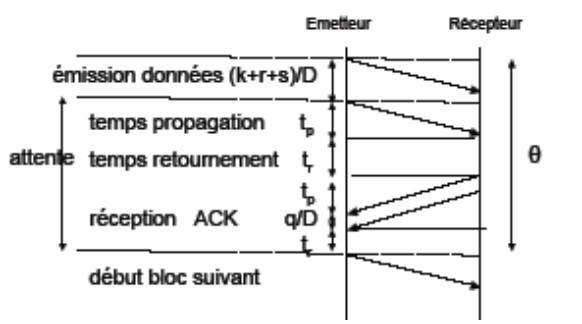
18

2.2.1. Correction par retransmission avec arrêt et attente (« stop and wait »)

- **2.2.2.1. Analyse quantitative**
- Calcul du rendement obtenu par un transfert d 'information entre deux stations A et B :
  - D : débit binaire de la liaison
  - k : nombre de bits d 'information utile d 'un bloc
  - r : nombre de bits de contrôle d 'erreur
  - s : nombre de bits de service
  - τ_e : taux d 'erreur par bit
  - t_p : temps de propagation du signal entre A et B
  - t_r : temps de retournement des modems
  - q : taille de l'accusé de réception
- En l'absence d'erreur :
  - durée de transmission d 'un bloc :
  - $\theta = (k + r + s + q) / D + 2. (t_p + t_r)$
  - débit utile $d = k / \theta$

19

Schéma temporel Arrêt et Attente

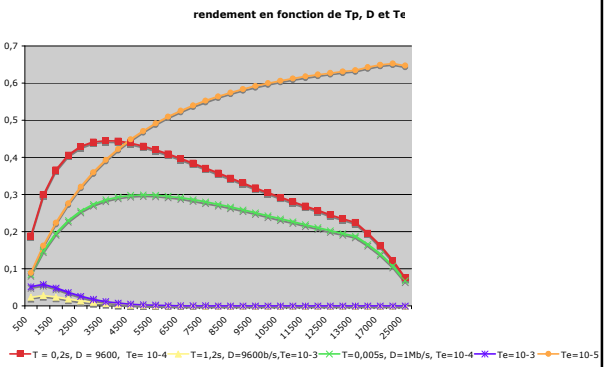


20

2.2.2.1. Analyse quantitative arrêt et attente

- en l 'absence d 'erreurs, le rendement est :
  - $d / D = k / (\theta . D)$
- Hypothèses : erreurs indépendantes et toutes détectées (et pas d'erreur sur les acquittements) :
  - probabilité de transmission d 'un bloc sans erreurs :
  - $P_c = (1 - \tau_e)^{k+r+s}$
  - probabilité pour qu 'un bloc soit transmis avec erreur pendant (i-1) essais, puis correctement au i^e essai :
  - $(1 - P_c)^{i-1} . P_c$
  - → probabilité d 'avoir une durée de transmission de i. θ
  - durée moyenne θ_m nécessaire au transfert correct d 'un bloc :
  - $\theta_m = \Sigma\, i. \theta. P_c. (1 - P_c)^{i-1} = \theta / P_c = \theta / (1 - \tau_e)^{k+r+s}$
- débit efficace moyen :
  - $d = k / \theta_m = (k / \theta). (1 - \tau_e)^{k+r+s}$
- le débit efficace varie avec la longueur des blocs
  - Pour une valeur donnée de τ_e ($\tau_e > 0$), il existe une valeur k* représentant la taille maximum des blocs à partir de laquelle le débit efficace diminue

21



22

2.2.2.2. Analyse qualitative

- Définition incrémentale des mécanismes de protocole nécessaires
- Les acquittements peuvent se perdre, la liaison peut être coupée
- => retransmission sur délai de garde (timeout)

Algorithme Emetteur	**Récepteur**
envoyer B	réception B'
armer T	si B' correct (checksum)
si réception ACK	alors envoyer ACK,
lire	
alors passer au bloc suivant	sinon envoyer NAK
sinonsi réception NAK	finsi
alors recommencer	
sinonsi déclenchement délai T	
alors recommencer	
finsi	

23

2.2.2.2. Analyse qualitative

- Nouveau problème : si perte de ACK
  - => récepteur doit distinguer un nouveau bloc d'un bloc réémis
  - => numérotation modulo 2
- cas du full duplex
- Exemple : protocole du bit alterné

24

Protocole du bit alterné

Initialement i = 0 , des deux côtés

Algorithme Emetteur	Récepteur
envoyer B,i	si réception B',i
armer T	alorssi B' correct
si réception ACK	alors envoyer ACK et lire
alors i := ¬i	i := ¬i
passer au bloc suivant	sinon envoyer NAK
	finsi
sinonsi réception NAK	sinonsi réception B', ¬i
alors recommencer	alors envoyer ACK
sinonsi déclenchement délai T	finsi
alors recommencer	
finsi	

- Remarque : quand le récepteur reçoit un bloc dupliqué B', ¬i, il l'acquitte (débloque l'émetteur), ne le « lit » pas (car déjà lu)

25

2.2.2. Correction par retransmission continue ou sélective

2.2.2.1. Mécanisme d 'anticipation

- Nécessite une liaison full duplex
- quand les sites sont éloignés, le temps de propagation n 'est pas négligeable
- => la source envoie plusieurs blocs de données avant de se bloquer dans l 'attente d 'un acquittement => **anticipation**
- si le degré d 'anticipation est suffisant, en l 'absence d 'erreurs, il y a transmission des blocs sans interruption dans un sens simultanément avec le retour des accusés de réception dans l 'autre sens

- durée de transmission d 'un bloc :
  - $\theta = (k + r + s) / D$
- l 'acquittement arrive au bout d 'un temps :
  - $\theta' = q / D + 2. t_p$
- pour qu 'il n 'y ait pas d 'attente entre les blocs :
  - $\theta' \leq (m - 1). \theta$

26

2.2.2. Correction par retransmission continue ou sélective

Mise en oeuvre du mécanisme d 'anticipation

- la source dispose d 'une séquence de numéros de bloc à émettre appelée **fenêtre d 'émission (FE)**. Les numéros de la fenêtre d 'émission indiquent les messages émis et non encore acquittés
- dès que la source reçoit un acquittement correspondant au premier numéro de la fenêtre, celle-ci est décalée d 'une position (fenêtre glissante = « **sliding window** »)
- le puits de données possède une séquence de numéros de bloc qu'il peut recevoir, appelée **fenêtre de réception (FR)**. Si le puits reçoit un bloc dont le n° ∉ FR, alors le bloc est refusé
- dès que le puits reçoit un bloc dont le numéro est le premier n° de FR, celle-ci est décalée d 'une position
- la taille de FR dépend de la façon dont le puits accepte les données (dans l 'ordre ou le désordre)

27

2.2.2.2. Retransmission continue (« go back N»)

- la retransmission est reprise à partir du bloc erroné
- le puits accepte les données dans l 'ordre :
- taille de FR = 1
- taille de FE optimale : $m \geq E (1 + \theta' / \theta)$
- s 'il se produit une erreur de transmission alors la durée pour transférer correctement le bloc n + 1 est $(m + 1).\theta$
- durée moyenne θ_m nécessaire au transfert correct d 'un bloc :
- $\theta_m = \Sigma (i.m + 1). \theta. P_c. (1 - P_c)^i = \theta . (m. (1 - P_c) / P_c + 1)$
- débit efficace :
- $d = k / \theta_m = D / (k + r + s) . k . (1 + m. (1 - P_c) / P_c)^{-1}$
- Le rendement R = d/D vaut : $(k/(k+r+s)) * P_c /(P_c + m * (1 - P_c))$
- $m = 1 + [(2 t_p * D + q) / (k + r + s)]$
- Plus D. tp est grand plus une erreur vaut cher : « Long Fat Pipe »
- Taille Fenêtre Emission : m trames ≥ 2 tp * D bits
- **Le récepteur est simple : accepte une seule trame et la livre (pas de tampon)**
- **Remarque : TCP de base voisin de ce modèle**

28

2.2.2.2. Retransmission continue : numérotation

- Généralement blocs numérotés **modulo 2^n** (n bits)
  - Bit alterné : n= 1
- Objectif :
  - déterminer la relation entre la taille de FE (m), la taille de FR (1) et le modulo nécessaire pour la numérotation des données
- Supposons m ≥ 2^n
  - émetteur envoie blocs 0 à 2^n -1
  - scénario1
    - les 2^n blocs se perdent,
    - émetteur renvoie bloc 0 sur timeout
    - récepteur accepte bloc 0 et renvoie ACK0
  - scénario 2
    - les 2^n blocs arrivent sont lus et acquittés par récepteur
    - les 2^n ACK se perdent
    - émetteur renvoie bloc 0 sur timeout
    - récepteur accepte bloc 0 comme bloc 2n
    - et renvoie ACK0
  - => bloc **dupliqué**

29

2.2.2.2. Retransmission continue : numérotation

- Le problème
  - à un instant donné le récepteur peut attendre
    - un des blocs de FE (y compris le premier)
    - le bloc qui suit FE
    - donc m + 1 blocs différents
  - Les numéros de ces m +1 blocs doivent être ≠
  - Si m < 2^n
    - récepteur ne peut attendre un bloc i + 2n (∈ FR)
    - tel que i ∈ FE => pas de duplication

- Go Back N : pas de duplication ssi $m + 1 \leq 2^n$
  - Exemples de scénarios
  - Modulo minimum de la numérotation = m + 1

30

2.2.2.3. Retransmission sélective

- En cas d'erreur (NAK ou timeout)
  - La source ne retransmet que le bloc erroné
  - Le puits stocke les données dans le désordre
    - Tant que les blocs ∈ FR
    - A priori livre dans l'ordre à la couche supérieure
  - $\theta_m = \theta / P_c$ (idem retransmission avec arrêt et attente)
  - $\theta = (k + r + s) / D$
  - Hypothèse : transmission continue :
    - pendant l 'attente de l 'acquittement d'autres blocs sont transmis et acceptés => FE et FR assez grands
  - débit efficace : $d = k / \theta_m = k / (k + r + s) . D . P_c$
  - rendement $d / D = k / (k + r + s) . P_c$
    - => indépendant de D et de t_p
  - Méthode efficace même si D élevé et t_p élevé
  - Nécessite de la mémoire au récepteur (FR)

31

Taille des fenêtres et retransmission sélective

- Pour avoir rendement maximal
  - En l'absence d'erreur, il suffit que :
    - $FE = m = 1 + [(2 t_p * D + q) / (k + r + s)]$
    - FR = 1 (car blocs arrivent dans l'ordre)
  - Si une erreur isolée
    - FE = 2m -1 (émission en attendant acquittement réémission)
    - FR = m (accepter les blocs qui suivent l'erreur)
  - Si deuxième erreur même bloc
    - FE = 3m – 2
    - FR = 2m - 1
  - Si τ_e élevé et blocs longs : retransmissions multiples
    - => rendement élevé seulement si Fenêtres (buffers) grande taille

32

2.2.2.3. Retransmission sélective : numérotation

- Numérotation modulo 2^n (n bits)
  - taille de FE = m
  - taille de FR = k
- Supposons $m + k > 2^n$
- scénario
  - émetteur envoie blocs 0 à m - 1
  - m blocs bien reçus et acceptés
  - Récepteur attend blocs m , … , m + k -1
  - attend numéros m, …, 2n -1, 0, …, (m + k - 1) (mod 2^n)
  - m acquittements se perdent
  - émetteur renvoie les m blocs 0, … , m - 1
  - récepteur reçoit bloc 0 et l'accepte comme bloc $2^n + 1$
  - => **bloc dupliqué**
- **Il n'y a pas duplication ssi $m + k \leq 2^n$**
  - Etude de scénarios
  - modulo numérotation minimal = m + k

33

3. Contrôle de flux

- le puits possède un nombre limité de tampons
  - Blocs reçus et non encore consommés par l'application
- le contrôle de flux permet au puits de réguler le débit de la source
- 3.1. Contrôle par ligne séparée
  - Hardware : exemple liaison parallèle
  - codage des données transparent
- 3.2. Protocole en mode caractères XON / XOFF
  - codage des données non transparent
- 3.3. Mécanisme de fenêtrage (crédit)
  - le crédit est le nombre de données que le puits accepte de recevoir (en fonction des tampons libres)
  - le crédit peut être fixe (HDLC) ou dynamique (TCP)

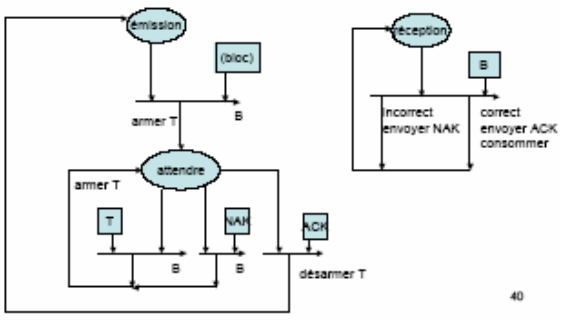
34

4. Représentation des protocoles

- un protocole peut être représenté comme un (ou plusieurs) automate(s) à plusieurs entrées et sorties
- On distingue :
  - les états internes de l'automate
    - y compris compteurs, …
  - les événements en entrée
    - blocs reçus, déclenchement de délais
  - les sorties
    - blocs émis
  - les transitions, qui à un état et à un événement, associent un nouvel état et une sortie (et une action)
- représentation par un réseau de Nutt
- exemple de représentation graphique du protocole avec arrêt et attente

35

Automate simplifié arrêt et attente



40

36

5. Protocoles de liaison

- 5.1. Catégories de protocoles
  - Protocoles orientés caractères
    - unité = caractère
    - dépendent d 'un alphabet (ASCII, EBCDIC)
    - caractères de contrôle => pas de transparence
    - ex : BSC (Binary Synchronous Communication)
    - STX < texte > ETX
    - DLE : permet de réaliser la transparence (code ASCII 16):
      - Pour envoyer C1 C2 C3 ETX C4 DLE C5
      - Coder **STX** C1 C2 C3 **DLE** ETX C4 **DLE** DLE C5 **ETX**
    - => adapté au mode asynchrone par caractère
    - => facile à implanter par logiciel

37

Code ASCII, en rouge codes liaison

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SP	!	«	#	$	%	&	`	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

38

Protocoles orientés bits

- Unité = bit
  - indépendant du codage des données
  - pas d'alphabet spécifique
  - permet de transmettre un nombre de bits quelconque (non multiple longueur caractère)
- indépendance avec les couches supérieures
- adapté au mode synchrone (et donc aux hauts débits)
- implanté au moins partiellement par circuit
- codage spécifique pour synchronisation trame
- ex : HDLC

39

5.2. Famille des protocoles HDLC

HDLC : High-level Data Link Control

- famille de protocoles
  - normalisée par l'ISO et l'UIT-T(protocoles LAP)
  - utilisé pour liaisons séries synchrones
    - modes half-duplex et full-duplex
    - modes asymétriques (primaire/secondaire) ou symétrique
  - nombreuses variantes
    - utilisé pour le protocole PPP (voir RFC 1662)

5.2.1. Délimitation des trames : fanions

- 01111110 < bits de la trame > 01111110
- pour réaliser la transparence on ajoute un bit 0, tous les cinq « 1 » consécutifs entre les 2 fanions
- fanion = resynchronisation trame
- entre deux trames : suite de fanions
- plus de 6 « 1 » consécutifs : annulation trame en cours

40

5.2.2. Format de la trame HDLC

- FCS = Frame Check Sequence (code polynomial V41)
- Adresse : permet de distinguer le primaire
  - aussi liaisons multipoint
- Commande : nécessaire à la signalisation
  - connexion, acquittement, numérotation, …

Fanion 8 bits	Adresse 8 bits	Commande 8 bits	Data n bits n ≥0	FCS 16 bits	Fanion 8 bits

insertion de 0 après 5 « 1 »

41

5.2.3 Fonctions principales d'HDLC

- Le protocole peut être découpé en plusieurs sous-protocoles
  - phase de connexion
  - phase de déconnexion
  - phase de transfert d'information
  - plusieurs rôles
    - rôle de la source de données
    - rôle du puit de données
    - deux rôles possibles simultanément (transfert duplex)

42

5.2.4 Transfert de données

- Les trames d'information I et de supervision : RR, REJ, SREJ, RNR
  - seules trames I contiennent données i(N(R), N(S))
  - N(S) = numéro de la trame de données (mod 8)
  - N(R) = numéro de la trame attendue (mod 8)
  - acquittement cumulatif : acquitte toutes les trames jusqu'à N(R) -1 (mod 8)
  - « piggybacking » : trame I contient N(S) et N(R)
  - transfert de données dans les 2 sens half ou full duplex
- RR(N(R)) : acquittement simple
- REJ(N(R)) : acquittement plus Go Back N(R)
- SREJ(N(R)) : acquittement plus Rej. Select. de N(R)
- RNR(N(R)) : acquittement plus indication problème (Not Ready)

43

5.2.6 Implantation des fenêtres

- source gère les variables d'état :
  - V(S) : n° de la prochaine trame à émettre
  - DN(R) : n° du dernier N(R) reçu
  - trames entre DN(R) et V(S) sont émises et non acquittées
- puits gère :
  - V(R) : n° de la prochaine trame attendue
- chaque trame I émise contient un n° N(S) = V(S)
- une trame reçue avec N(S) = V(R) est consommée
- si N(S) > V(R) alors la trame est :
  - rejetée si le puits accepte dans l'ordre (go back N)
  - mise en attente si le puits accepte dans le désordre
- un accusé de réception contient un n° N(R) = V(R), et acquitte toutes les trames de n° < N(R) (acquittement collectif)

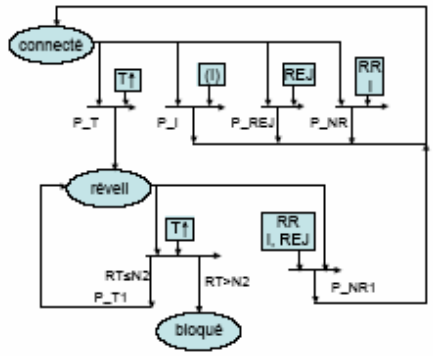
44

5.2.7 Exemple HDLC diagramme de Nutt

- Phase de transfert des données
  - rôle symétrique des deux extrémités, qui peuvent exécuter simultanément les fonctions de source et de puits
  - T = délai avant retransmission
  - RT : compteur de retransmissions

45

Source simplifiée, avec REJ, sans P/F ni RNR



46

procédures émetteur

P_T
envoyer I(V(R),DN(R))
armer T
RT:=1

P_I
envoyer I(V(R),V(S)) *
si V(S) = DN(R)
alors armer T, RT :=0
finsi
V(S) := V(S) +1

***émission ou réémission de trame**
Si réémission, V(S) doit progresser jusqu'à atteindre la borne supérieure de la fenêtre d'émission (retransmission continue)

P_REJ
DN(R) := N(R)
armer T
RT := 0
V(S) := N(R)
/* réémettre trames à partir N(R) */

P_NR
si N(R) = V(S)
alors
désarmer T, RT:=0
sinonsi N(R) > DN(R) armer T, RT :=0
finsi
DN(R) := N(R)

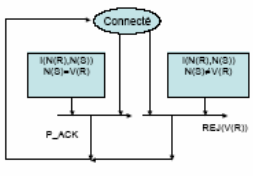
P_NR1
P_NR
V(S) := N(R)

P_T1
envoyer I(V(R), DN(R))
armer T
RT := RT +1

Note +, > : modulo 8

47

Puits simpifié

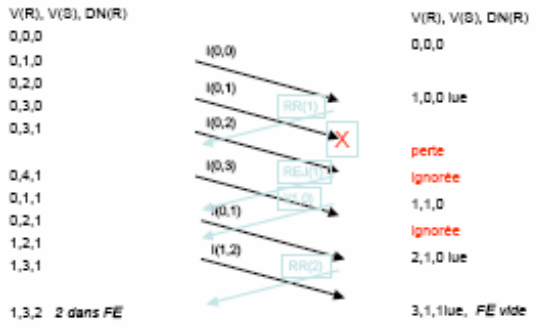


- P_ACK
  consommer I
  V(R) := V(R) + 1
  si aucune émission
  alors envoyer RR(V(R))
  finsi

  Remarque :
  si trame I à émettre
  on inclut V(R) dans le champ N(R) de la trame I
  (« piggybacking »)

48

Exemple d'échange d'information duplex, REJ



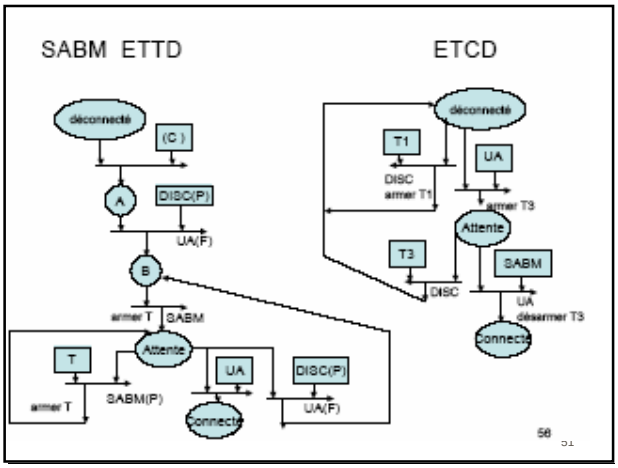
49

5.2.8 HDLC : Contrôle de liaison

HDLC : mode connecté

- plusieurs modes différentiés par
  - trame d'établissement de connexion ex :
    - mode normal (SNRM normal)
    - mode équilibré SABM (autonomous balanced)
    - mode symétrique SARM (autonomous response)
- Exemple mode SABM
  - utilisé entre usager (ETTD) et réseau (ETCD)
  - ETTD prend initiative connexion
  - ETCD attente passive
  - Commandes SABM, UA (Unnumbered Ack), DISC

50



58

6. Spécification et vérification de protocoles

- **6.1. Modèles d'automates à états finis**
  - chaque automate du protocole est dans un état défini à chaque instant
  - un état est composé des valeurs de toutes les variables d'état
    - compteurs de trames/ fenêtres
    - nombre retransmisions …
    - chaque variable doit avoir un nombre fini de valeurs
  - un canal a aussi un état (trames en circulation, …)
    - modèle communication synchrone
    - émission et réception simultanées du message
    - modèle communication asynchrone (plus réaliste)
      - canal = **retard** (mémoire)

52

6.1. Modèles d'automates à états finis

- la combinaison des états des automates du protocole (émetteur + récepteur) et du canal forme **l'état global**
- on peut construire le graphe des états globaux, où les nœuds représentent les états, et les arcs les transitions entre états
- les transitions correspondent à un événement (émission ou réception de message, timeout, etc
- à partir de **l'état initial** on peut construire le **graphe des états accessibles**, qui permet d'analyser le fonctionnement du protocole
- le nombre d'état accessibles peut être très grand
  - produit nombre états chaque automate, chaque canal
  - explosion combinatoire

53

6.1. Modèles d'automates à états finis

- Définition : automate = (S, M, I T)
  - S : ensemble des états des processus et du canal
  - M : ensemble des messages échangés à travers le canal
  - I : ensemble des états initiaux des processus
  - T : ensemble des transitions entre états
- L'analyse du graphe d'accessibilité des états permet de détecter des erreurs ou anomalies telles que :
  - blocage (« deadlock ») : états puits
  - spécification incomplète : événement non attendu
  - transition non pertinente
  - états non atteints
  - états non accessibles depuis un autre
- Exemple :
  - langage SDL (Specification and Description Language)
    - langage normalisé par ITU
    - MSC Message Sequence Charts : représentation graphique

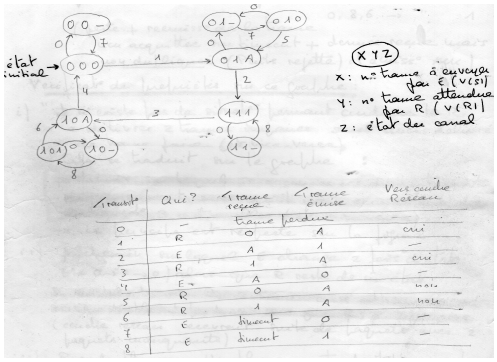
54

Bit alterné : automates

- Exemple : protocole du bit alterné avec acquittements non numérotés en « half duplex »
  - Emetteur
    - 2 états : V(S) = 0,1
  - Récepteur
    - 2 états : V(R) = 0,1
  - Canal
    - 4 états : vide, B0, B1, Ack (_,0,1,A)
  - => 2*2*4 = 16 états globaux
    - état global initial (0,0,0)
    - Tous les états ne sont pas accessibles, ex : (0,0,A)
- Graphe des états accessibles
  - Modélisation des pertes et des retransmissions
  - Vérification de propriétés :
    - Il n'existe pas de situations pouvant conduire le récepteur à accepter deux trames paires (resp. impaires) consécutives
    - L'émetteur ne peut pas changer deux fois d'état alors que le récepteur reste dans le même état
    - Pas de blocage

55

Exemple : diagramme d'états du bit alterné sans numérotation des ACK en half duplex



56

Bit alterné : automates

- Exemple : protocole du bit alterné avec acquittements non numérotés en « full duplex »
  - Emetteur
    - 2 états : V(S) = 0,1
  - Récepteur
    - 2 états : V(R) = 0,1
  - Canal « aller »
    - 3 états : vide, B0, B1 (_,0,1)
  - Canal « retour »
    - 2 états : vide, Ack (_,A)
  - => 2*2*3*2 = 24 états globaux
- Graphe des états accessibles
  - Les transitions du graphe sont les mêmes que dans le graphe précédent, sauf dans le cas de **croisement** entre une trame de données et un ACK

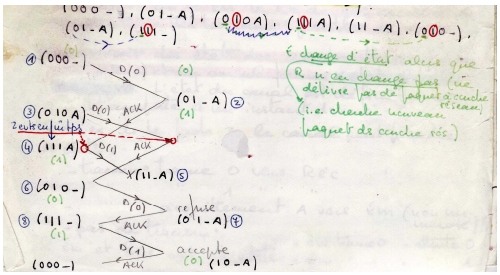
57

Modélisation du croisement entre une donnée et un acquittement

- Quand une trame de données et un ACK se croisent :
  - Le récepteur ne peut pas enlever la trame du canal « aller »
    - => il y aurait 2 ACK simultanément dans le canal « retour »
  - L'émetteur ne peut pas enlever l'ACK du canal « retour »
    - => il y aurait 2 trames de données simultanément dans le canal « aller »
  - Les deux événements (enlever trame + enlever ACK) se produisent simultanément
- Ex : (000A) → (111A)
  - trame 0 + ACK => pris pour ACK(0)
  - trame 1 + ACK => pris pour ACK(1)

58

Scénario mettant à défaut le protocole du bit alterné sans numérotation des ACK en full duplex



59

6.2. Le langage Promela

- approche « langage »
  - modèle de validation : définition d'un ensemble de règles complet et consistant définissant les interactions entre processus d'un système distribué
- Objectif d'un modèle de validation
  - modéliser les protocoles afin de vérifier leur complétude et leur consistance logique
  - description partielle des protocoles => abstraction des détails d'implémentation
- langage PROMELA :
  - permet de décrire des modèles de validation de protocoles
- Spin : permet d'exécuter une spécification Promela
  - simulation du protocole
  - construction exhaustive du graphe d'accessibilité
  - aide à la preuve de propriétés
- description du langage PROMELA et exemples : TD-TP

60