Master Informatique
Cours protocoles réseaux
Université Paris-Diderot

Examen final
(8 janvier 2010)

Aucun document n'est autorisé. Le barème est donné à titre indicatif.

1 Codes (14pts)

On considère le $(5, 3)$ code linéaire sur l'alphabet $\{0, 1\}$ décrit par le tableau ci-dessous :(2pts)

mots source	mots code
000	x00xx
001	00101
0xx	010xx
011	xx1x1
100	1001x
101	101x1
110	1100x
111	111xx

1. En utilisant l'hypothèse de linéarité, remplacer les x par les éléments binaires manquant. (2pts)
2. Quelle est la matrice génératrice du code? (2pts)
3. Les 3 premières lignes de la matrice correspondent-elle à la matrice identité? Si ce n'est pas le cas, donner une matrice génératrice équivalente dans laquelle les 3 premières lignes correspondent à l'identité. (2pts)

   On utilisera dans la suite des questions une matrice génératrice dont les 3 premières lignes correspondent à l'identité.
4. Dans le codage d'un mot à quoi correspondent les 3 premières lettres? (1pt) les 2 dernières? (1pt)
5. Donner la matrice de contrôle. (2pts)
6. Quelle est la distance minimale du code? Combien ce code corrige-t-il d'erreurs ? (2pts)
7. Comment utilise-t-on la matrice de contrôle pour corriger le(s) erreur(s)? (2pts)

2 Questions diverses (16pts)

1. Donner au moins deux protocoles de la couche réseau d'internet. Les décrire succinctement. (2pts)
2. Donner des exemples d'applications qui utilise UDP justifier pourquoi ces applications utilisent UDP et non TCP. (1pt)

3. Qu'est ce que le multicast? Existe-t-il dans les couches réseau ou transport d'internet? (1pt)

4. Dans quelle(s) couche(s) d'internet le contrôle de la congestion est-il réalisé? Quel est l'objet du contrôle de flot? Est-ce différent du contrôle de la congestion? Le protocole UDP assure-t-il un contrôle de flot? (2pts)

   Décrire les grands principes du contrôle de la congestion réalisé par TCP. (2pts)

5. Que signifie RTT? (1pt)

6. Combien d'échanges entre le client et le serveur sont réalisés pour établir une connexion dans le protocole TCP? Décrivez ces échanges (2pts)

7. A quelle valeur est initialisée le numéro de séquence (sequence number dans un segment TCP) d'un client se connectant à un serveur dans TCP? Pourquoi? (2pts)

8. Que signifie dans le protocole TCP la réception d'un segment avec un ack (acknowledgment number) valant n? (1pt)

9. Dans quel(s) cas, l'émetteur est-il amené à réémettre un datagramme dans le protocole UDP? et un segment dans le protocole TCP? (2pts)

3 Protocole du bit alterné (18pts)

3.1 Description

La Figure 1 est le code du bit alterné tel qu'il a été décrit en cours: on a un *sender* s et un *receiver* r qui communiquent par envois réceptions de messages par deux canaux (Q_s et Q_r). Les canaux de communication sont FIFO et peuvent perdre des messages mais ils ne dupliquent ni n'altèrent les messages. in est un tableau infini correspondant à la séquence (infinie) des paquets à transmettre du sender vers le receiver. out est la séquence des messages délivrés par le receiver. On suppose qu'initialement, pour tout j $in[j] \neq \perp$ et $out[j] = \perp$.

Le "Timeout" est un mécanisme qui se déclenche au bout d'un temps déterminé δ après $reset(Timer)$: si aucun $reset(Timer)$ n'est exécuté après δ, la condition "Timer has expired" devient vraie et la partie de code correspondante sera exécutée.

Dans la suite, si v est une variable du processus p (s pour sender et r pour receiver) v_p^t représente la valeur de v à l'instant t.

On a vu en cours que ce protocole assurait une communication fiable entre le sender et le receiver. Plus précisément, le protocole assure les deux propriétés: $\forall k(0 \leq k \leq i_r) out[k] = in[k]$ (*Sûreté*) et $\forall k \exists t : i_r^t \geq k$ (*Vivacité*).

On va s'intéresser ici à ce qui se passe si les variables (i_r, i_s, bit_r et bit_s) du sender et du receiver ne sont pas initialisées correctement et si initialement les canaux de communications peuvent contenir des valeurs binaires quelconques. (Cela pourrait correspondre à une erreur transitoire qui modifie l'état du système). On va montrer que dans ce cas le protocole garantit quand même qu'au plus nombre fini de messages seront perdus.

Pour cela représentons une configuration S du système comme des séquences $S = bit_s, Q_r, bit_r, Q_s$. Par exemple $0, 111000, 1, 00$ correspond à une configuration pour laquelle $bit_s = 0$, Q_r contient 111000 (le premier élément qui sera extrait est 0) , $bit_r = 1$ et Q_s contient 00. Dans la suite, on omettra parfois les ',' : la configuration précédente sera représentée par 0111000100. En particulier on dira qu'une configuration S est *sûre* si elle peut s'écrire sans ','comme un mot de $0^*1^* \vee 1^*0^*$.

1. Montrer qu'en appliquant les règles de l'algorithme:
   partant de $(0, 000, 1, 1)$ on peut arriver à $(1, 1111, 1, 1)$
   partant de $(1, 0, 0, 0)$ on peut arriver à $(0, 0, 0, 0)$

1 **Initialisations globales**
2 $Q_r := \emptyset$
3 $Q_s := \emptyset$

CODE DU SENDER s:

4 **Initialisation**
5 $i_s := 0$
6 $bit_s := 0$
7 $reset(Timer)$
8 **TIMEOUT:**
9 { **Timeout: Timer has expired** } $\longrightarrow$ $send(bit_s, in[i_s])$ /* timeout */
10 **SEND:**
11 { **un message** (b) **dans File** Q_s} $\longrightarrow$
12 $receive(b)$ /* ack */
13 **if** $b = bit_s$ **then**
14 $bit_s := 1 - bit_s$
15 $i_s := i_s + 1$
16 $send(bit_s, in[i_s])$ /* send */
17 $reset(Timer)$

CODE DU RECEIVER r:

18 **Initialization**
19 $i_r := 0$
20 $bit_r := 1$
21 **RECEIVE:**
22 { **A message** $(b, data)$ **in Queue** Q_r} $\longrightarrow$
23 $receive(b, data)$
24 **if** $b \neq bit_r$ **then**
25 $bit_r := 1 - bit_r$
26 $i_r := i_r + 1$
27 $out[i_r] := data$ /* deliver */
28 $send(bit_r)$

COMPORTEMENT DU CANAL
29 **LOSS** Q_s**:** /* m est perdu */
30 $\{m \in Q_s\} \longrightarrow Q_s := Q_s - \{m\}$
31 **LOSS** Q_r**:** /* m est perdu */
32 $\{m \in Q_r\} \longrightarrow Q_r := Q_r - \{m\}$

Figure 1: Protocole du bit alterné.

partant de $(1,000,1,1)$ on peut arriver à $(0,0,0,0)$
(on rappelle que la règle LOSS permet de perdre des messages) (2pts)

2. Montrer que si les initialisations sont correctement faites alors la configuration initiale est sûre. (1pt)

3. Montrer que si S est sûre, alors S restera sûre. En déduire que si les initialisations sont correctement faites alors toutes les configurations qui peuvent être atteintes sont sûre. (3pts)

4. En remarquant que si S est sûre, alors les messages émis seront bien reçus, montrer que si S est sûre, il existe i il existe j tels que pour tout k, $in[i+k] = in[j+k]$. (2pts)

5. Soit $S = (0, 01010, 1,)$ (ici le Q_s est vide), montrer qu'avec un *deliver*, un *ack* et *send*, un *deliver* et enfin un *ack* et un *send* on peut retourner dans l'état S. En déduire que partant dans une configuration non sûre, on peut rester indéfiniment dans des configurations non sûres (même si plus aucun message n'est perdu). (2pts)

6. Soit $S = s_1 \cdots s_m$ une configuration (écrite sans ',') $V(S)$ est défini par $V(S) = |\{i | s_i \neq s_{i+1}\}|$.

   (a) Montrer que S est sûre si et seulement si $V(S) \leq 1$. (0,5pt)

   (b) On dira que $S = s_1 \cdots s_m$ est *activable* si et seulement si $s_1 = s_m$. Montrer que l'émission du message *suivant* par le sender n'est possible que dans un état S activable. (0,5pt)

   (c) Montrer que: S activable $\wedge S$ sûr $\Leftrightarrow V(S) = 0$. (0,5pt)

   (d) Montrer que: S non activable $\wedge S$ sûr $\Leftrightarrow V(S) = 1$. (0,5pt)

   (e) Étant donné une séquence infinie d'états obtenue à partir de S, $S_0, \ldots, S_k, \ldots$, et soit $S_{i_0}, \ldots, S_{i_k}, \ldots$ la sous séquence des états activables. Soient S_{i_j} et $S_{i_{j+1}}$ deux états activables successifs montrez que $V(S_{i_j}) \leq V(S_{i_{j+1}})$. (2pts)

   (f) Étant donné une séquence infinie d'états obtenue à partir de S, $S_0, \ldots, S_k, \ldots$, et soit $S_{i_0}, \ldots, S_{i_k}, \ldots$ la sous séquence des états activables. Montrer qu'à partir de l'état S, il y aura au plus $V(S)$ erreurs de transmissions (modification ou perte). (2pts)

7. Déduire des questions précédentes que quelque soit l'état des canaux et des variables du sender et du receiver, le protocole du bit alterné assure qu'au plus un nombre fini de messages seront perdus ou modifiés. (2pt)