TD6 : Modélisation et implémentation

1. *Une modélisation avec des automates à états finis*

On s'intéresse aux protocoles de liaison implémentés lors du TD4, plus précisément à la version 2.

a) Quels sont les messages qui transitent sur le canal (bidirectionnel) ?

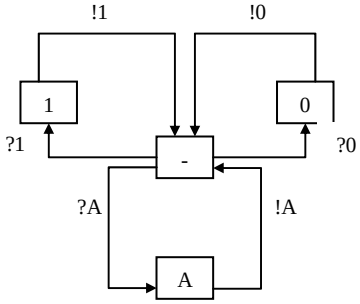
b) Une simplification consiste à considérer que les messages forment un ensemble fini. Que peut-on considérer comme ensemble fini de messages. Appelons M cet ensemble.

On ignore la partie message : M={0,1,A}

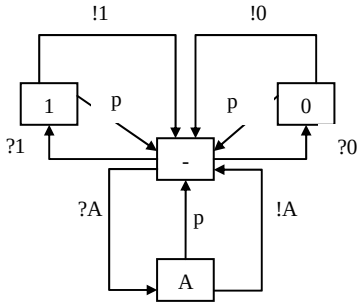
c) Peut-on représenter le comportement du canal par un automate à états fini sur l'alphabet M ?

Si le canal peut avoir un nombre non borné de messages, la réponse est non.
Ici, le canal peut comporter plusieurs messages dans la situation suivante :
message non reçu par le récepteur et le timer expire (côté émetteur).
Si on connaît une borne sur le temps de transit des messages et la valeur du timer, on peut calculer une borne de la taille du canal…

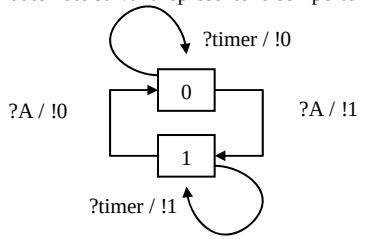
d) On suppose que le canal est borné, (il contient au plus n messages). Pour n=1, donner une modélisation (sous forme d'automate à états fini) d'un canal fiable (en considérant les messages de M).



e) Le canal est maintenant bruité (non fiable). On considère uniquement les pertes : pour n=1, donner une modélisation (sous forme d'automate à états fini) d'un canal avec pertes (en considérant les messages de M).



f) L'automate suivant représente le comportement de l'émetteur :

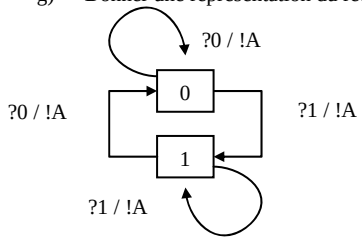


- ?m (resp. !m) désigne la réception (resp. émission) du message ou événement m.
- ?m / !n signifie « après réception du message/événement m, envoyer n ».

Quelles sont les simplifications faites dans cette modélisation ?

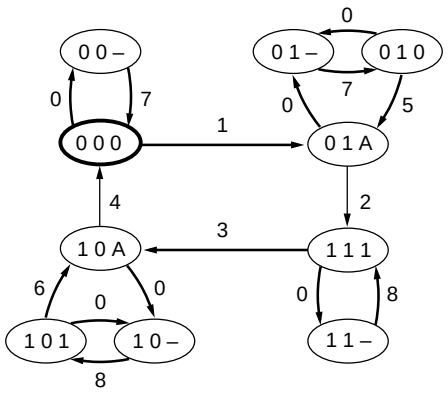
Interaction avec la couche supérieure : on suppose que, à tout moment, un message (paquet) de la couche supérieure est disponible,
Interaction avec la couche inférieure : on agit directement avec le canal

g) Donner une représentation du récepteur sous forme d'automate.



h) Déduire des questions précédentes une représentation du système globale. A-t-on des situations de blocage ? A-t-on des doublons ? Cela ne fait pas apparaître les situations d'erreur rencontrées lors des expérimentations faites avec ce protocole (TD4, version V2). Expliquer. Dans cette modélisation, quelle simplification masque ces situations d'erreur ?

Voir page 247 du Tanenbaum (édition 4), figure 3.21 [attention, il y a une erreur : l'état destination de la transition 8 est (1,0,1)]



(a)

Transition	Who runs?	Frame accepted	Frame emitted	To network layer
0	–	(frame lost)		–
1	R	0	A	Yes
2	S	A	1	–
3	R	1	A	Yes
4	S	A	0	–
5	R	0	A	No
6	R	1	A	No
7	S	(timeout)	0	–
8	S	(timeout)	1	–

(b)

2. *Une modélisation de CSMA/CD avec des automates temporisés*

La modélisation suivante est basée sur le modèle des automates temporisés. Le système est composé de deux *Sender* (Figure 1) et un *Bus (*Figure 2*). Lambda* et *Sig* représentent respectivement la durée d'émission d'une trame et le temps maximum de propagation entre deux stations du réseau. *x* et *y* sont des horloges. L'état *0* est initial, et les transitions sont de la forme : *garde*/*événement*/*action*. Par exemple, la transition *3* du *Sender* :

- est gardée par l'expression booléenne *x<Sig*,
- correspond à l'événement réception de *CD*,
- effectue l'action *x'=0* (la nouvelle valeur de l'horloge *x* est *0*).

La synchronisation des événement est de type *Rendez-Vous* : en même temps ont lieu les événements *!begin* et *?begin*.

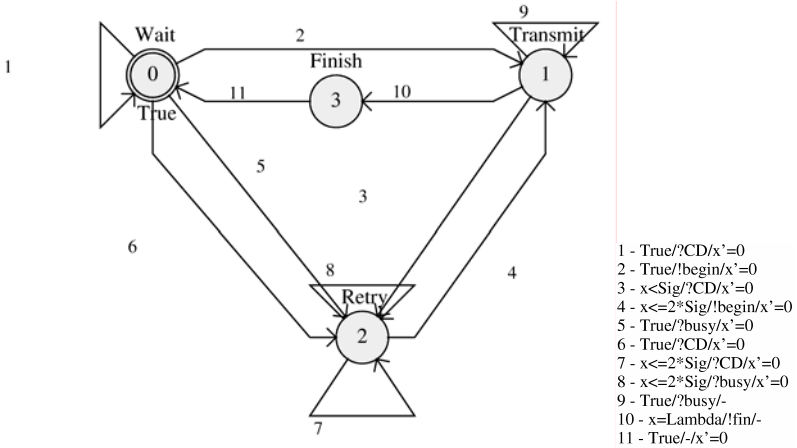


Figure 1 Un Sender

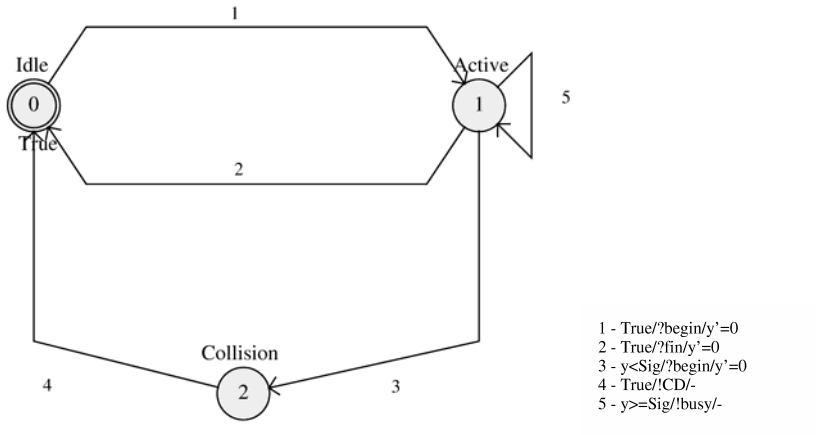


Figure 2 Le Bus

a) Donner les simplifications effectuées sur la modélisation proposée.

Pas d'aspect probabiliste (les réémissions sont non-déterministes),
Pas de réception de message : on s'intéresse juste à l'émission (pas de consommateur)
Les couches au-dessus sont ignorées : pas de producteur ou plutôt on suppose qu'il y a toujours un message à envoyer.
On prend un nombre particulier de Sender pour faire une étude : par exemple 2 Sender avec 1 Bus.

b) Donner un scénario avec 2 Senders qui émettent chacun un message sans collision.

On s'intéresse dans cette question et la suivante au produit synchronisé Sender1//Sender2//Bus. On pourrait prendre les valeurs réelles de Lamba et Sig, Ici on va prendre pour simplifier Sig=1 et Lambda=2 (il suffit d'avoir 0< Sig < Lambda).

```
================================================
Time: 0.00
Location: Wait_1.Wait_2.Idle
x1 = 0 & x2 = 0 & y = 0
--------------------------------
VIA: begin1
--------------------------------
Time: 0.00
Location: Transmit_1.Wait_2.Active
x1 = 0 & x2 = 0 & y = 0
---------------
VIA 2.00 time units
---------------
Time: 2.00
Location: Transmit_1.Wait_2.Active
x1 = 2 & x2 = 2 & y = 2
--------------------------------
VIA: fin1
--------------------------------
Time: 2.00
Location: Finish_1.Wait_2.Idle
x1 = 2 & x2 = 2 & y = 0
--------------------------------
```

```
VIA: begin2
------------------------------
Time: 2.00
Location: Finish_1.Transmit_2.Active
x1 = 2 & x2 = 0 & y = 0
------------------------------
VIA: fin2
------------------------------
Time: 2.00
Location: Finish_1.Finish_2.Active
x1 = 2 & x2 = 0 & y = 0
================================================
```

c) Donner un scénario avec 2 Sender qui émettent chacun leur message après avoir provoqué une collision.

```
================================================
Time: 0.00
Location: Wait_1.Wait_2.Idle
x1 = 0 & x2 = 0 & y = 0
------------------------------
VIA: begin2
------------------------------
Time: 0.00
Location: Wait_1.Transmit_2.Active
x1 = 0 & x2 = 0 & y = 0
------------------------------
VIA: begin1
------------------------------
Time: 0.00
Location: Transmit_1.Transmit_2.Collision
x1 = 0 & x2 = 0 & y = 0
------------------------------
VIA: CD2
------------------------------
Time: 0.00
Location: Retry_1.Retry_2.Idle
x1 = 0 & x2 = 0 & y = 0
---------------
VIA 2.00 time units
---------------
Time: 2.00
Location: Retry_1.Retry_2.Idle
x1 = 2 & x2 = 2 & y = 2
```

et après ça se passe un peu comme le scénario précédent…

```
================================================
```

3. *Une implémentation de CSMA/CD : Ethernet*

Câblage

a) Donnez les différents types de support utilisés dans un réseau Ethernet.
b) Que se passe-t-il dans un réseau Ethernet câblé en 10Base2 s'il n'y a pas de bouchon ‘terminateur’ ?

Aucune transmission ne sera possible : les signaux ne seraient pas absorbés en arrivant aux extrémités du câble. Conséquence : une réflexion du signal qui serait source de bruits et perturberait toutes les transmissions.

c) Donnez la définition des équipements réseau suivants : répéteur, pont, hub, switch.
d) D’après vous, comment sont câblées les salles machines que vous utilisez ? Donner un schéma.

Diffusion

e) Sur un réseau Ethernet, comment peut-on envoyer une trame à destination de toutes les machines du réseau ?

Adresse de broadcast : FF:FF:FF:FF:FF:FF.

f) Est-ce que cela charge plus le réseau que d'envoyer une trame à destination d'une seule machine ?

1 seule trame émise pour faire de la diffusion.

Une variante d'Ethernet.

g) Rappeler pourquoi les trames Ethernet doivent être au minimum de 64 octets de longueur.
h) Une variante d'Ethernet 10Base2 impose la même contrainte bien que le débit soit 10 fois plus élevé. Comment est-il possible de maintenir la taille minimale à 64 octets et garantir le bon fonctionnement du réseau ?

D = debit (bps), P = delai max propagation (s), dmax = distance max entre 2 stations (m), v = vistesse de propagation (m/s). On a P = dmax/v, et Tmin = 64 octets = D x 2 x P. On veut que D augmente (x10) et Tmin constant, donc P diminue (/10), et donc dmax diminue (/10). On peut ainsi raccourcir les longueurs de câble d'un facteur 10.

Analyse de traces Ethernet.

Un analyseur de protocole Ethernet a fourni la trace donnée en annexe 1 (hors préambule, délimiteur, et CRC). On rappelle la structure de trame Ethernet (hors préambule, délimiteur, et CRC) en annexe 2.

i) Quelles sont les adresses Ethernet des machines qui interviennent dans ce dialogue?

00:40:07:03:04:2B et 02:60:8C:E8:02:91.

j) Quelle est la nature des données transportées par ces trames?

Type trame 0800 : paquet IP, protocole de la couche réseau.

k) Peut-on savoir si ces trames contiennent, ou non, des caractères de bourrage?

Toutes les trames ont 46 octets de données (le minimum requis), donc bourrage possible…

1. Sans connaître le format des données transportées par ces trames.

NON : on ne peut pas le savoir lors de l'analyse niveau Ethernet.

2. En connaissant le format des données transportées par ces trames.

OUI : Il faut analyser le paquet IP fourni par la couche réseau, plus précisément le champ longueur du paquet IP

En gras+italique : la taille du datagramme IP.
Barré : les caractères de bourrage Ethernet.

```
==================================
Date: Fri, 28 Oct 2003 13:27:33

Trame n°1
00 40 07 03 04 2b 02 60 8c e8 02 91 08 00 45 00
00 2c 14 ee 00 00 3c 06 85 7a 93 d2 5e 63 93 d2
5e 5c 10 a4 09 e7 42 0c 56 01 00 00 00 00 60 02
40 00 c1 29 00 00 02 04 05 b4 02 80

Trame n°2
02 60 8c e8 02 91 00 40 07 03 04 2b 08 00 45 00
00 2c 8b 46 00 00 40 06 0b 22 93 d2 5e 5c 93 d2
```

5e 63 09 e7 10 a4 4d 91 6c 01 42 0c 56 02 60 12
16 d0 30 b6 00 00 02 04 05 b4 ~~00 00~~

Trame n°3
00 40 07 03 04 2b 02 60 8c e8 02 91 08 00 45 00
00 ***28*** 14 ef 00 00 3c 06 85 7d 93 d2 5e 63 93 d2
5e 5c 10 a4 09 e7 42 0c 56 02 4d 91 6c 02 50 10
3e bc 20 87 00 00 ~~3d 00 00 04 02 80~~

Trame n°4
02 60 8c e8 02 91 00 40 07 03 04 2b 08 00 45 00
00 ***2a*** 8b 47 00 00 40 06 0b 23 93 d2 5e 5c 93 d2
5e 63 09 e7 10 a4 4d 91 6c 02 42 0c 56 02 50 18
16 d0 17 36 00 00 31 33 ~~00 08 00 00~~

Trame n°5
00 40 07 03 04 2b 02 60 8c e8 02 91 08 00 45 00
00 ***28*** 14 f8 00 00 3c 06 85 74 93 d2 5e 63 93 d2
5e 5c 10 a4 09 e7 42 0c 56 02 4d 91 6c 04 50 10
3e bc 20 85 00 00 ~~3d 00 00 04 02 80~~

Trame n°6
02 60 8c e8 02 91 00 40 07 03 04 2b 08 00 45 00
00 ***2c*** 8b 4a 00 00 40 06 0b 1e 93 d2 5e 5c 93 d2
5e 63 09 e7 10 a4 4d 91 6c 04 42 0c 56 02 50 18
16 d0 e2 fb 00 00 32 37 33 32 ~~8c e8~~

Trame n°7
00 40 07 03 04 2b 02 60 8c e8 02 91 08 00 45 00
00 ***28*** 14 fd 00 00 3c 06 85 6f 93 d2 5e 63 93 d2
5e 5c 10 a4 09 e7 42 0c 56 02 4d 91 6c 08 50 11
3e bc 20 80 00 00 ~~00 01 86 a3 00 00~~

Trame n°8
02 60 8c e8 02 91 00 40 07 03 04 2b 08 00 45 00
00 ***28*** 8b 4b 00 00 40 06 0b 21 93 d2 5e 5c 93 d2
5e 63 09 e7 10 a4 4d 91 6c 08 42 0c 56 03 50 10
16 d0 48 6c 00 00 ~~32 37 00 08 00 00~~

ANNEXE 1

```
Date: Fri, 28 Oct 2003 13:27:33

Trame n°1
00 40 07 03 04 2b 02 60 8c e8 02 91 08 00 45 00
00 2c 14 ee 00 00 3c 06 85 7a 93 d2 5e 63 93 d2
5e 5c 10 a4 09 e7 42 0c 56 01 00 00 00 00 60 02
40 00 c1 29 00 00 02 04 05 b4 02 80
```

```
Trame n°2
02 60 8c e8 02 91 00 40 07 03 04 2b 08 00 45 00
00 2c 8b 46 00 00 40 06 0b 22 93 d2 5e 5c 93 d2
5e 63 09 e7 10 a4 4d 91 6c 01 42 0c 56 02 60 12
16 d0 30 b6 00 00 02 04 05 b4 00 00

Trame n°3
00 40 07 03 04 2b 02 60 8c e8 02 91 08 00 45 00
00 28 14 ef 00 00 3c 06 85 7d 93 d2 5e 63 93 d2
5e 5c 10 a4 09 e7 42 0c 56 02 4d 91 6c 02 50 10
3e bc 20 87 00 00 3d 00 00 04 02 80

Trame n°4
02 60 8c e8 02 91 00 40 07 03 04 2b 08 00 45 00
00 2a 8b 47 00 00 40 06 0b 23 93 d2 5e 5c 93 d2
5e 63 09 e7 10 a4 4d 91 6c 02 42 0c 56 02 50 18
16 d0 17 36 00 00 31 33 00 08 00 00

Trame n°5
00 40 07 03 04 2b 02 60 8c e8 02 91 08 00 45 00
00 28 14 f8 00 00 3c 06 85 74 93 d2 5e 63 93 d2
5e 5c 10 a4 09 e7 42 0c 56 02 4d 91 6c 04 50 10
3e bc 20 85 00 00 3d 00 00 04 02 80

Trame n°6
02 60 8c e8 02 91 00 40 07 03 04 2b 08 00 45 00
00 2c 8b 4a 00 00 40 06 0b 1e 93 d2 5e 5c 93 d2
5e 63 09 e7 10 a4 4d 91 6c 04 42 0c 56 02 50 18
16 d0 e2 fb 00 00 32 37 33 32 8c e8

Trame n°7
00 40 07 03 04 2b 02 60 8c e8 02 91 08 00 45 00
00 28 14 fd 00 00 3c 06 85 6f 93 d2 5e 63 93 d2
5e 5c 10 a4 09 e7 42 0c 56 02 4d 91 6c 08 50 11
3e bc 20 80 00 00 00 01 86 a3 00 00

Trame n°8
02 60 8c e8 02 91 00 40 07 03 04 2b 08 00 45 00
00 28 8b 4b 00 00 40 06 0b 21 93 d2 5e 5c 93 d2
5e 63 09 e7 10 a4 4d 91 6c 08 42 0c 56 03 50 10
```

```
16 d0 48 6c 00 00 32 37 00 08 00 00
```

ANNEXE 2 : Structure des trames ETHERNET.

Adresse Destinataire	Adresse origine	Type de trame	'données'

Signification et taille des différents champs :

Adresse destinataire (6 octets) : Adresse ETHERNET du destinataire,

Adresse origine (6 octets) : Adresse ETHERNET de l'émetteur,

Type de trame (2 octets) : précise à quel protocole s'adresse les données,

0600	XNS
0800	IP
0806	ARP
...	

Données (46-1500 octets) : les données !
Au minimum 46 octets (avec caractères de bourrage si nécessaire).

ANNEXE 3 : Structure des datagrammes IP.

0	8	16	24	31
Version	Taille de l'entête	Type de service	Longueur totale	
Identification		Flags	Offset (fragment)	
Durée de vie	Protocole transporté	Checksum		
Adresse IP source				
Adresse IP destinataire				
Options				
Données (Segment de niveau supérieur)				

Signification et taille des différents champs :

Version (4 bits) : Numéro de version du protocole

Taille de l'entête (4 bits) : Longueur de l'en-tête en mots de 32 bits (les options font partie de l'en-tête),

Type de service (8 bits) : Priorité au délai, au débit, à la fiabilité,

00	pas de service,
08	priorité au délai,
10	priorité au débit,
20	priorité à la fiabilité,
...	

Longueur totale (16 bits) : Longueur totale, en octets, entête et données comprises,

Identification (16 bits) : Identifiant du datagramme

Flags (3 bits) : Gestion de la fragmentation

000	dernier fragment
001	fragment à suivre
010	pas de fragment

Offset (13 bits) : Décalage de la fragmentation: position relative par rapport au début du datagramme initial, si celui-ci a été fragmenté (exprimé en unité de 8 octets),

Durée de vie (8 bits) : Temps écoulé depuis l'émission, exprimé en nombre de sauts,

Protocole transporté (8 bits) : Protocole de niveau supérieur

01_h	ICMP
06_h	TCP
09_h	IGP
11_h	UDP
$1D_h$	Transport ISO Classe 4
...	

Checksum (16 bits) : Total de contrôle,

Adresse IP source (32 bits) Adresse IP de l'émetteur

Adresse IP destinataire (32 bits) Adresse IP du destinataire

Options (n mots de 32 bits) : Facultatives,

Données (n octets) : Les données.