Diffusion

Master 2 Informatique - UFR S.A.T

Pr. Ousmane THIARE

ousmane.thiare@ugb.edu.sn
http://www.ousmanethiare.com/

16 avril 2020

Chapitre 3 : Diffusion

Diffusion
Introduction

Problématique Etant donné N sites et un réseau de communication, comment assurer une *diffusion fiable* où :

- tous les destinataires, non en pause, reçoivent les mêmes messages ;

A priori :

Mais Si l'ordre de délivrance est identique pour tous les récepteurs (*propriété d'uniformité*) OU si l'ordre de délivrance est identique à l'ordre causal d'émission alors la diffusion est dite *atomique*.

Diffusion
Introduction

Problématique Etant donné N sites et un réseau de communication, comment assurer une *diffusion fiable* où :

- tous les destinataires, non en pause, reçoivent les mêmes messages ;
- si l'émetteur d'un message tombe en panne pendant une diffusion, alors tous les destinataires reçoivent le message OU aucun destinataire ne reçoit le message.

A priori :

Mais Si l'ordre de délivrance est identique pour tous les récepteurs (*propriété d'uniformité*) OU si l'ordre de délivrance est identique à l'ordre causal d'émission alors la diffusion est dite *atomique*.

Diffusion
Introduction

Problématique Etant donné N sites et un réseau de communication, comment assurer une *diffusion fiable* où :

- tous les destinataires, non en pause, reçoivent les mêmes messages ;
- si l'émetteur d'un message tombe en panne pendant une diffusion, alors tous les destinataires reçoivent le message OU aucun destinataire ne reçoit le message.

A priori :

- il n'y a pas d'ordre supposé sur la réception des messages par les destinataires ;

Mais Si l'ordre de délivrance est identique pour tous les récepteurs (*propriété d'uniformité*) OU si l'ordre de délivrance est identique à l'ordre causal d'émission alors la diffusion est dite *atomique*.

Diffusion
Introduction

Problématique Etant donné N sites et un réseau de communication, comment assurer une *diffusion fiable* où :

- tous les destinataires, non en pause, reçoivent les mêmes messages ;
- si l'émetteur d'un message tombe en panne pendant une diffusion, alors tous les destinataires reçoivent le message OU aucun destinataire ne reçoit le message.

A priori :

- il n'y a pas d'ordre supposé sur la réception des messages par les destinataires ;
- il n'y a pas de relation entre l'ordre d'émission et l'ordre de réception.

Mais Si l'ordre de délivrance est identique pour tous les récepteurs (*propriété d'uniformité*) OU si l'ordre de délivrance est identique à l'ordre causal d'émission alors la diffusion est dite *atomique*.

ATTENTION La date de délivrance d'un message peut être différente de celle de sa réception : le processus de contrôle de la diffusion ne transfert un message reçu précédemment que si ce message est le prochain devant être utilisé par le processus de calcul suivant l'algorithme utilisé. On suppose que le processus de contrôle peut stocker des messages.

Diffusion
Cas d'un anneau uni-directionnel

Un jeton tourne dans l'anneau.
$\Longrightarrow$ A la réception du jeton par un site S_i

- S_i traite les messages dans l'ordre donné par le jeton

Diffusion
Cas d'un anneau uni-directionnel

Un jeton tourne dans l'anneau.
$\Longrightarrow$ A la réception du jeton par un site S_i

- S_i traite les messages dans l'ordre donné par le jeton
- Tant que le message de tête correspond à un message émis par S_i, S_i le supprime du jeton

Diffusion
Cas d'un anneau uni-directionnel

Un jeton tourne dans l'anneau.
$\Longrightarrow$ A la réception du jeton par un site S_i

- S_i traite les messages dans l'ordre donné par le jeton
- Tant que le message de tête correspond à un message émis par S_i, S_i le supprime du jeton
- Tant que S_i a un message M à émettre, il rajoute (M,i) en queue du jeton

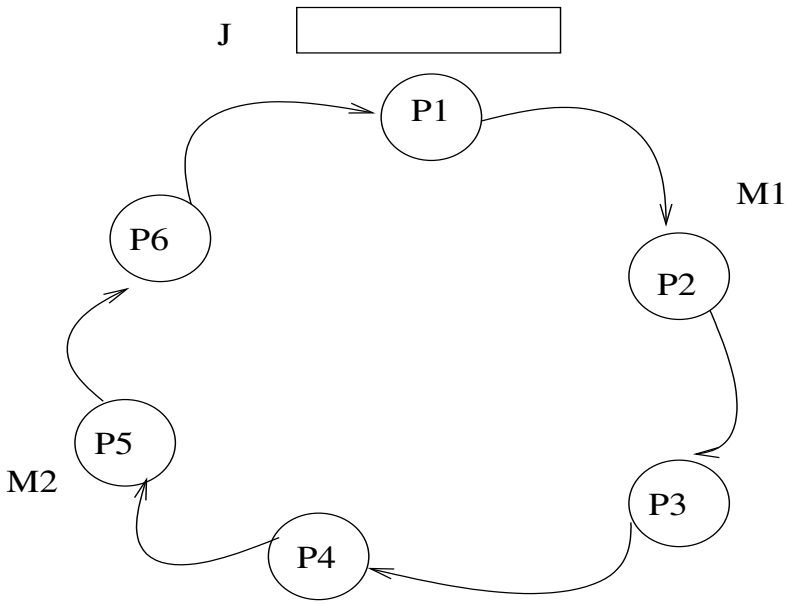
Diffusion
Cas d'un anneau uni-directionnel

Un jeton tourne dans l'anneau.
$\Longrightarrow$ A la réception du jeton par un site S_i

- S_i traite les messages dans l'ordre donné par le jeton
- Tant que le message de tête correspond à un message émis par S_i, S_i le supprime du jeton
- Tant que S_i a un message M à émettre, il rajoute (M,i) en queue du jeton
- S_i émet le jeton vers son suivant sur l'anneau.

Diffusion
Cas d'un anneau uni-directionnel

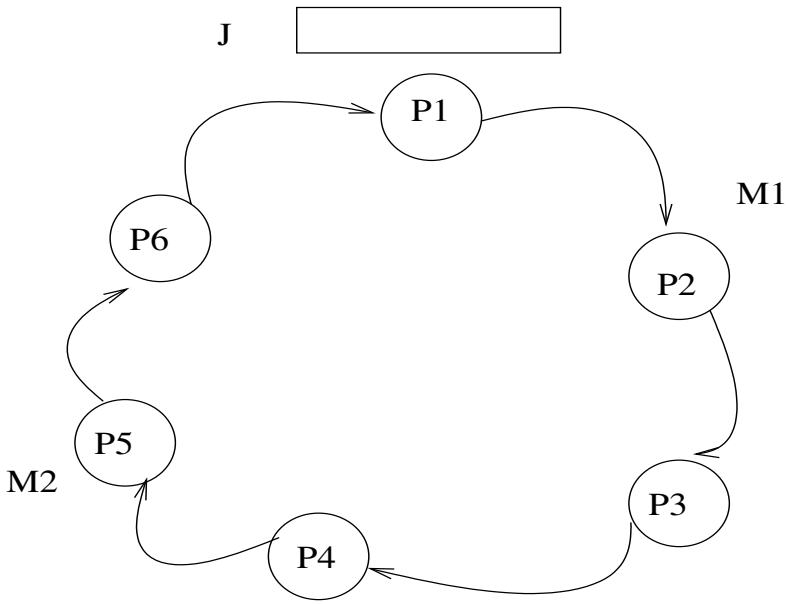
Exemple : sur l'anneau suivant, P_2 et P_5 veulent émettre en diffusion et le jeton J est en P_1.



Lorsque P_2 reçoit le jeton, il y met M1 et le transmet (figure de gauche). Lorsque le jeton passe en P_3 et P_4, ceux-ci récupèrent et traitent les messages du jeton.

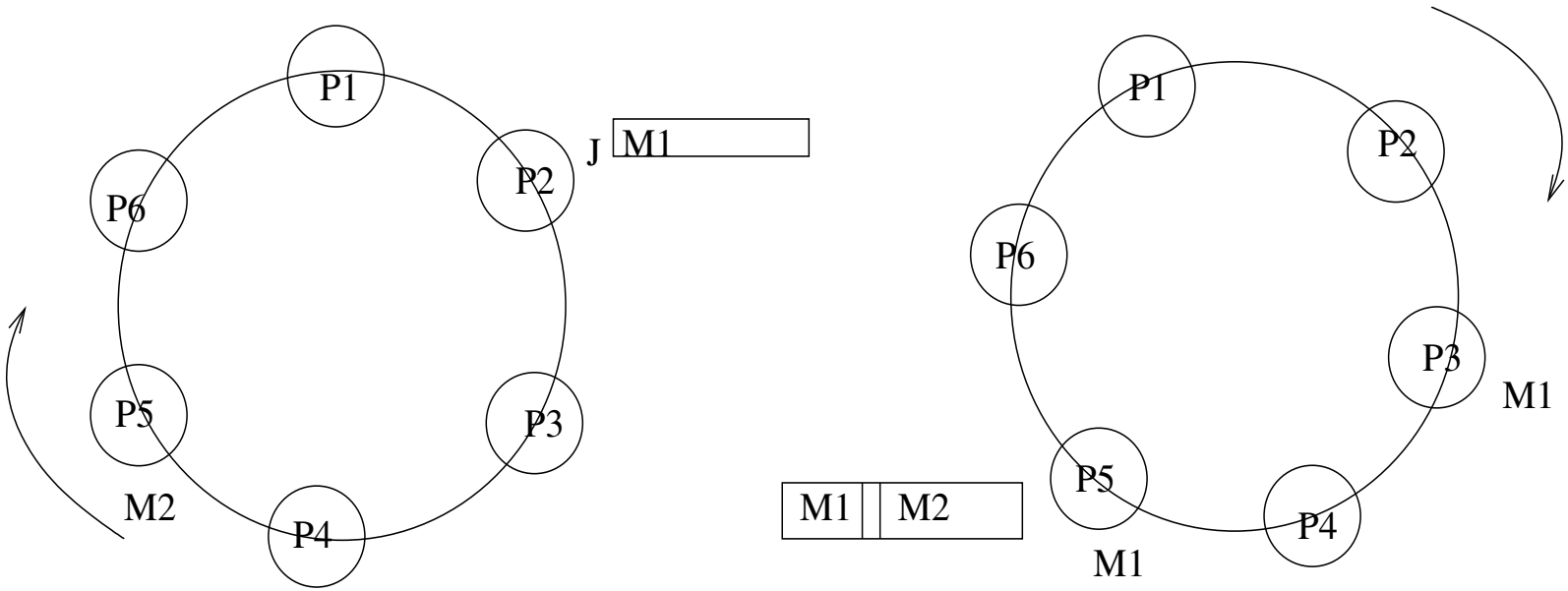
Diffusion
Cas d'un anneau uni-directionnel

Exemple : sur l'anneau suivant, P_2 et P_5 veulent émettre en diffusion et le jeton J est en P_1.



Enfin lorsque le jeton arrive en P_5, celui-ci récupère et traite les messages du jeton, rajoute M2 en queue et transmet le jeton (figure de droite).

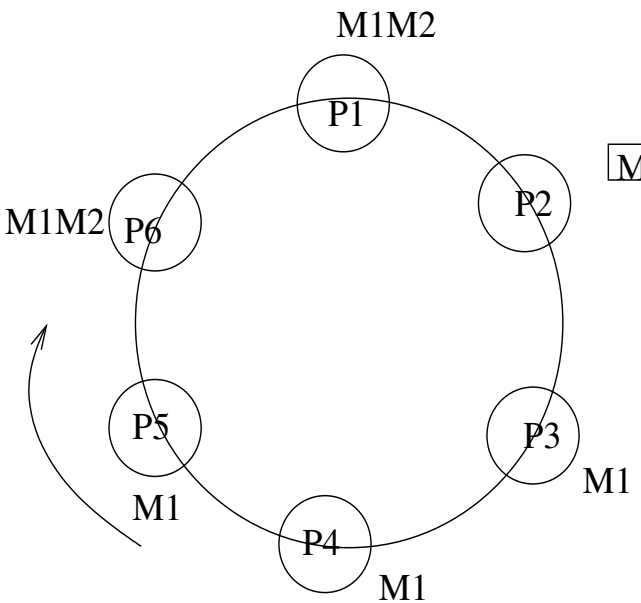
Diffusion
Cas d'un anneau uni-directionnel

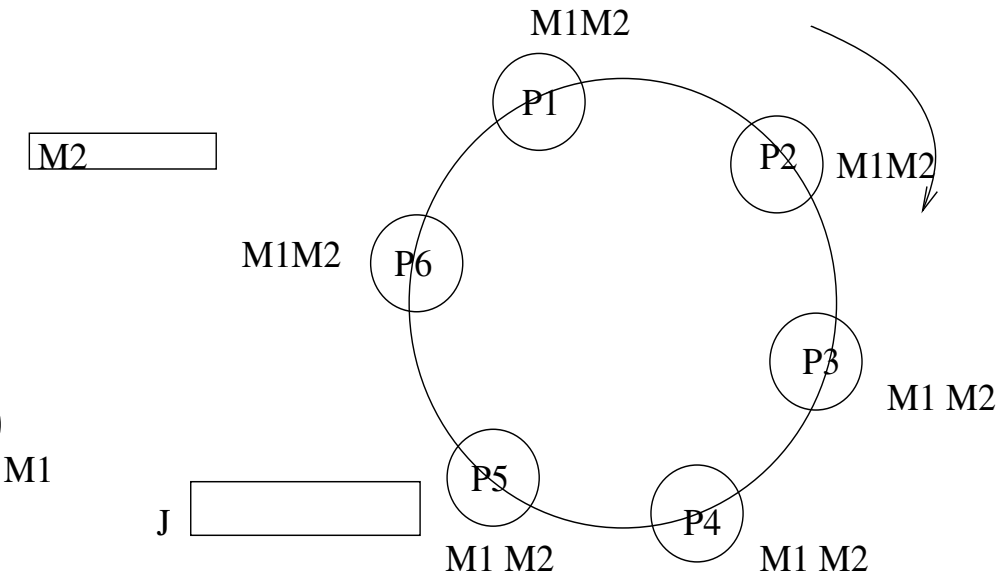


Lorsque le jeton passe en P_6 et P_1, ceux-ci récupèrent et traitent les messages du jeton. Enfin lorsque le jeton arrive en P_2, celui-ci récupère et traite les messages du jeton, enlève M1 et transmet le jeton (figure de gauche).

Diffusion
Cas d'un anneau uni-directionnel

Lorsque le jeton passe en P_3 et P_4, ceux-ci récupèrent et traitent les messages du jeton. Enfin lorsque le jeton arrive en P_5, celui-ci récupère et traite les messages du jeton, enlève M2 et transmet le jeton (figure de droite).





Kaashoeck 1989

Les émetteurs transmettent leurs messages au serveur qui les numérote et assure la diffusion. Ce protocole utilise très souvent des fenêtres d'anticipation.

Le serveur S dispose d'un tampon d'émission qui lui permet de stocker T messages, d'une variable A qui donne le numéro du dernier message reçu à diffuser et d'une variable d'acquittement V qui correspond au dernier message acquitté par TOUS les destinataires. Au tampon est associé le tableau ACK de $T \times N$ bits.

Chaque site S_i dispose d'une variable R_i qui donne le numéro du dernier message reçu et acquitté.

$\Longrightarrow$ Lorsque S reçoit de S_i un message M à diffuser :

- soit le tampon n'est pas plein $A - V < T$:
  - il numérote le message par $d_M = A + 1$ et envoie (M, d_M, i)à tous les destinataires

- soit le tampon est plein : il prévient l'émetteur et refuse le message.

$\Longrightarrow$ Après l'envoi du message à tous les sites, S arme une time-out. A l'expiration de celui-ci, S remet le message vers les sites qui ne l'ont pas encore acquitté et réarme le time-out si nécessaire.
$\Longrightarrow$ Lorsqu'un destinataire S_k reçoit un message (M,d,i) :

Diffusion
Protocole utilisant un serveur

- soit le tampon n'est pas plein $A - V < T$:
  - il numérote le message par $d_M = A + 1$ et envoie (M, d_M, i)à tous les destinataires
  - il incrémente A

- soit le tampon est plein : il prévient l'émetteur et refuse le message.

$\Longrightarrow$ Après l'envoi du message à tous les sites, S arme une time-out. A l'expiration de celui-ci, S remet le message vers les sites qui ne l'ont pas encore acquitté et réarme le time-out si nécessaire.
$\Longrightarrow$ Lorsqu'un destinataire S_k reçoit un message (M,d,i) :

Diffusion
Protocole utilisant un serveur

- soit le tampon n'est pas plein $A - V < T$:
  - il numérote le message par $d_M = A + 1$ et envoie (M, d_M, i)à tous les destinataires
  - il incrémente A
  - il stocke le message dans la case $c = d_M mod\ T$ du tampon T

- soit le tampon est plein : il prévient l'émetteur et refuse le message.

$\Longrightarrow$ Après l'envoi du message à tous les sites, S arme une time-out. A l'expiration de celui-ci, S remet le message vers les sites qui ne l'ont pas encore acquitté et réarme le time-out si nécessaire.
$\Longrightarrow$ Lorsqu'un destinataire S_k reçoit un message (M,d,i) :

Diffusion
Protocole utilisant un serveur

- soit le tampon n'est pas plein $A - V < T$:
  - il numérote le message par $d_M = A + 1$ et envoie (M, d_M, i)à tous les destinataires
  - il incrémente A
  - il stocke le message dans la case $c = d_M mod\ T$ du tampon T
  - il metà0 tous les cases $ACK[c][x], 1 \leq x \leq N$.
- soit le tampon est plein : il prévient l'émetteur et refuse le message.

$\Longrightarrow$ Après l'envoi du message à tous les sites, S arme une time-out. A l'expiration de celui-ci, S remet le message vers les sites qui ne l'ont pas encore acquitté et réarme le time-out si nécessaire.
$\Longrightarrow$ Lorsqu'un destinataire S_k reçoit un message (M,d,i) :

- si $d > R_k + 1$, alors le destinataire stocke (si ce n'est pas déjà fait) ce message et demande au serveur la ré-émission (vers lui) des messages $(M, x, \cdots)$ tel que $d > x > R_k + 1$ et $(M, x, \cdots)$ n'est pas stocké par le destinataire.

Si après une incrémentation de R_k, le destinataire trouve $(M, R_k + 1, \cdots)$ dans son stock, il émet l'acquittement du message $(A, R_k + 1, k)$ vers le serveur, "utilise" M et incrémente R_k.

- si $d > R_k + 1$, alors le destinataire stocke (si ce n'est pas déjà fait) ce message et demande au serveur la ré-émission (vers lui) des messages $(M, x, \cdots)$ tel que $d > x > R_k + 1$ et $(M, x, \cdots)$ n'est pas stocké par le destinataire.
- si $d < R + 1$ alors le destinataire émet l'acquittement du message (A, d, k) vers le serveur

Si après une incrémentation de R_k, le destinataire trouve $(M, R_k + 1, \cdots)$ dans son stock, il émet l'acquittement du message $(A, R_k + 1, k)$ vers le serveur, "utilise" M et incrémente R_k.

Diffusion
Protocole utilisant un serveur

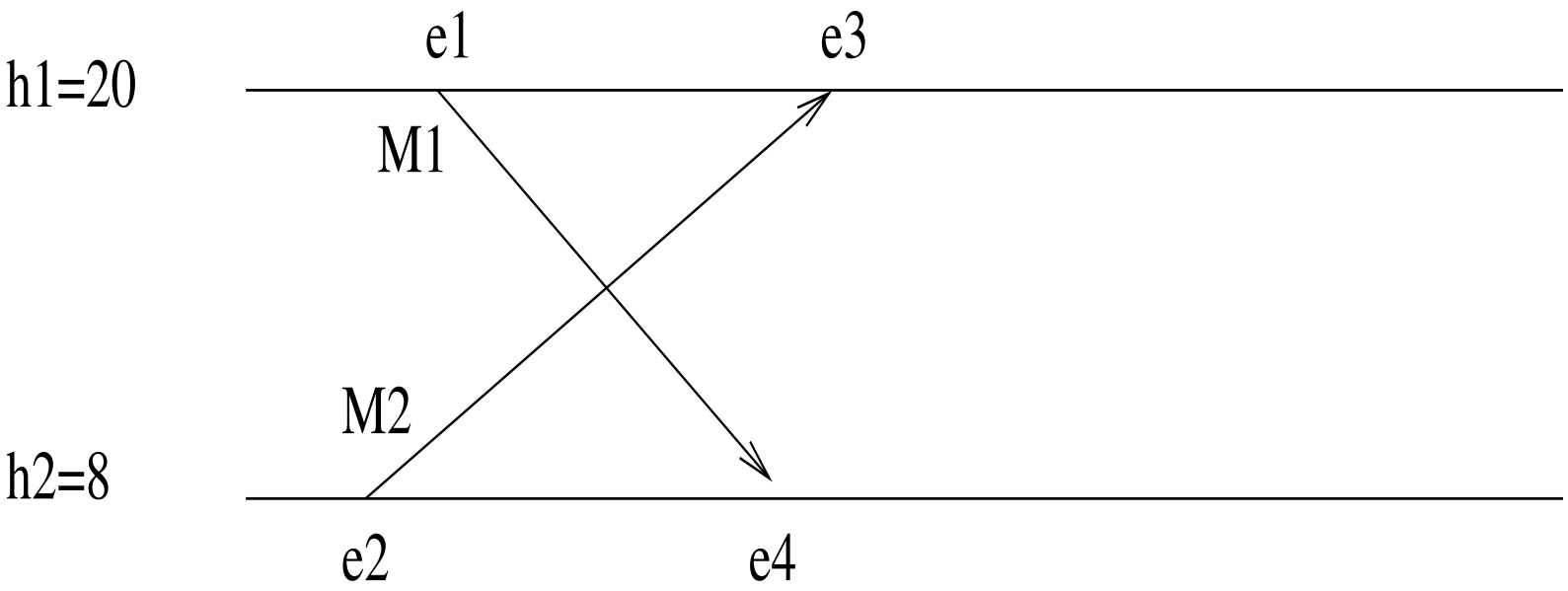
- si $d > R_k + 1$, alors le destinataire stocke (si ce n'est pas déjà fait) ce message et demande au serveur la ré-émission (vers lui) des messages $(M, x, \cdots)$ tel que $d > x > R_k + 1$ et $(M, x, \cdots)$ n'est pas stocké par le destinataire.
- si $d < R + 1$ alors le destinataire émet l'acquittement du message (A, d, k) vers le serveur
- si $d = R_k + 1$ alors le destinataire émet l'acquittement du message (A, d, k) vers le serveur, incrémente Rk et "utilise" M.

Si après une incrémentation de R_k, le destinataire trouve $(M, R_k + 1, \cdots)$ dans son stock, il émet l'acquittement du message $(A, R_k + 1, k)$ vers le serveur, "utilise" M et incrémente R_k.

$\Longrightarrow$ Lorsque S reçoit un acquittement (A, d, k) du message M (qui est stocké dans la case $c = d \bmod T$ du tampon), il met à1 le bit $ACK[c][k]$. Si toutes les cases du tableau $ACK[c][x], 1 \leq x \leq N$ valent 1, le message M a été acquitté par tous les destinataires : il déstocke M. Dès que $V = d - 1$ il incrémente V.
$\Longrightarrow$ Lorsque S reçoit un message de demande de ré-émission d'un message M, il le ré-émet vers le demandeur (s'il l'a encore en stock, sinon il ne fait rien). Cet algorithme qui assure l'uniformité par le réseau ethernet.

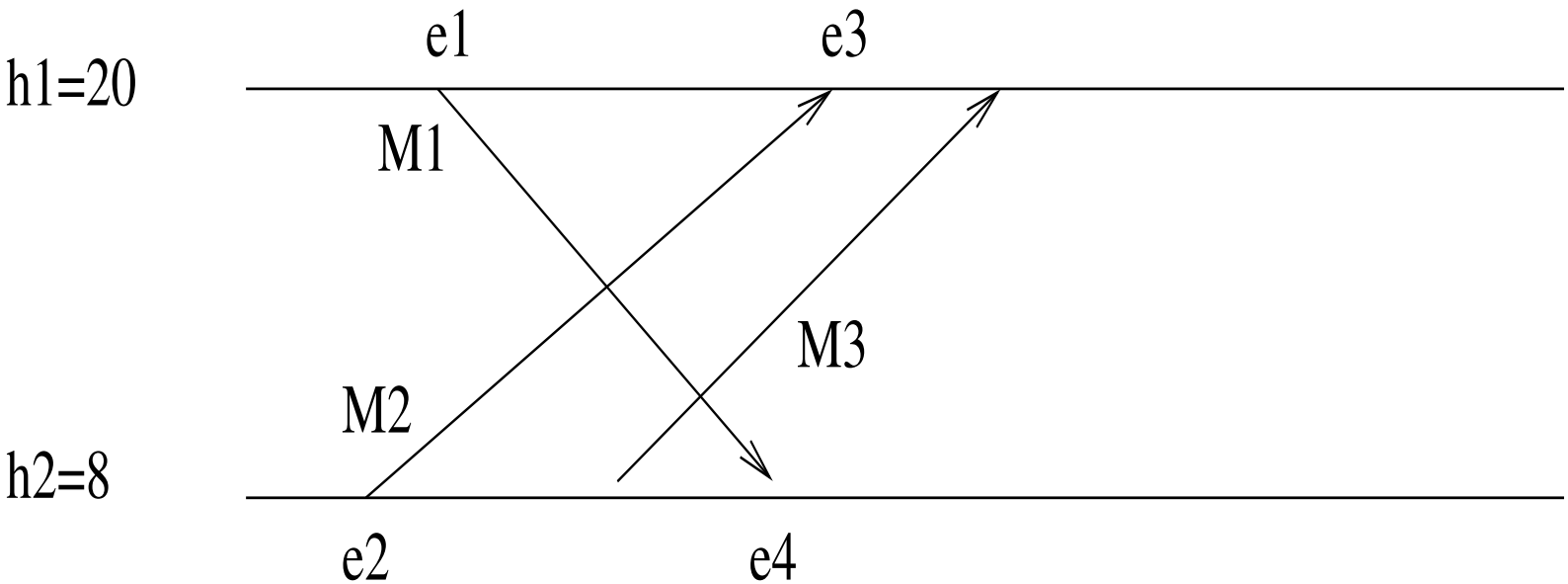
Diffusion
Protocole utilisant des estampilles

On se place dans le cas distribué sur un réseau FIFO.



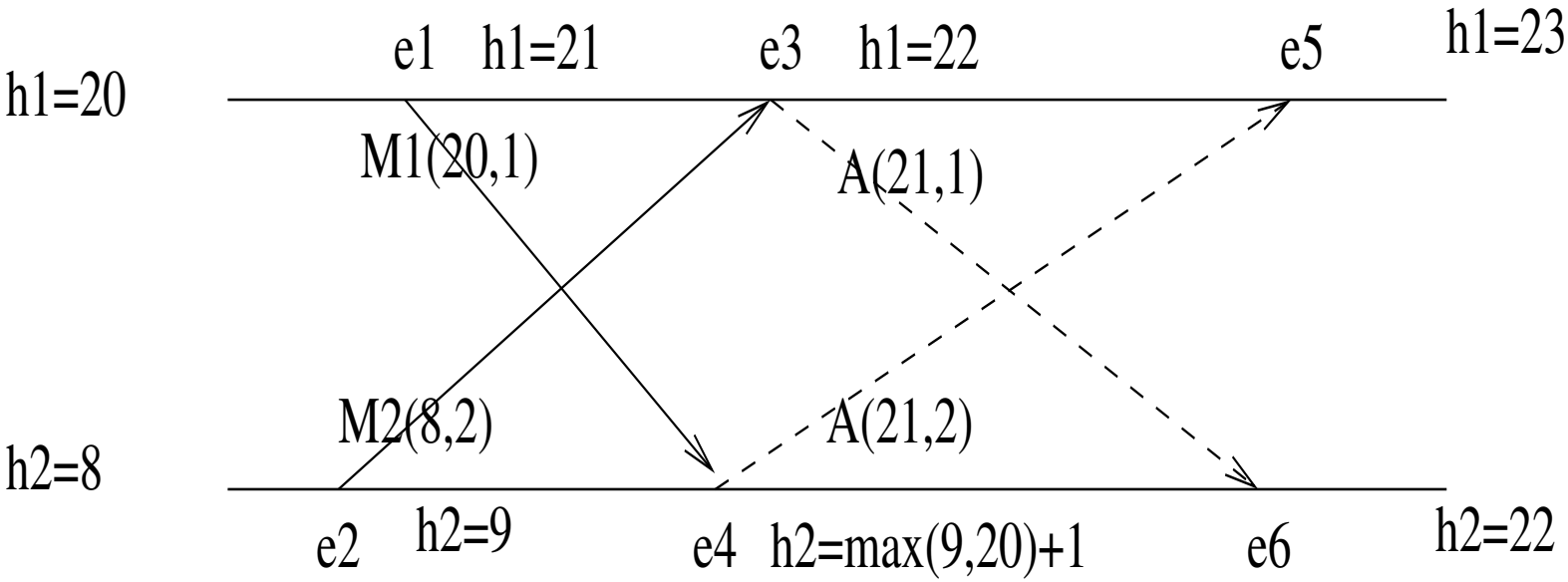
On peut utiliser les dates h_1 et h_2. Les messages en diffusion sont estampillés par la date sur le site émetteur. en *e*4 : P2 reçoit M1 daté 20. Il sait alors qu'il peut traiter M2 et M1 car il sait qu'il n'y a pas de message provenant de P1 de date inférieure à 20 car le FIFO garantit qu'il l'aurait déjà reçu

en *e*3 : P1 reçoit M2, Il sait qu'il devra traiter M2 avant M1 par contre il ne peut toujours pas traiter M1 car dans le cas suivant



rien ne dit que la date de M3 n'est pas inférieure à 20.
$\Longrightarrow$ utilisation d'acquitements datés et récalage d'horloges.

Diffusion
Protocole utilisant des estampilles

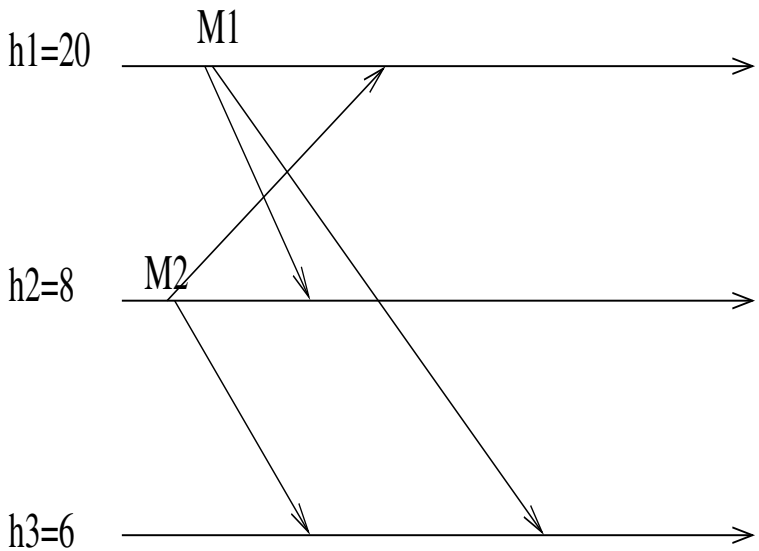


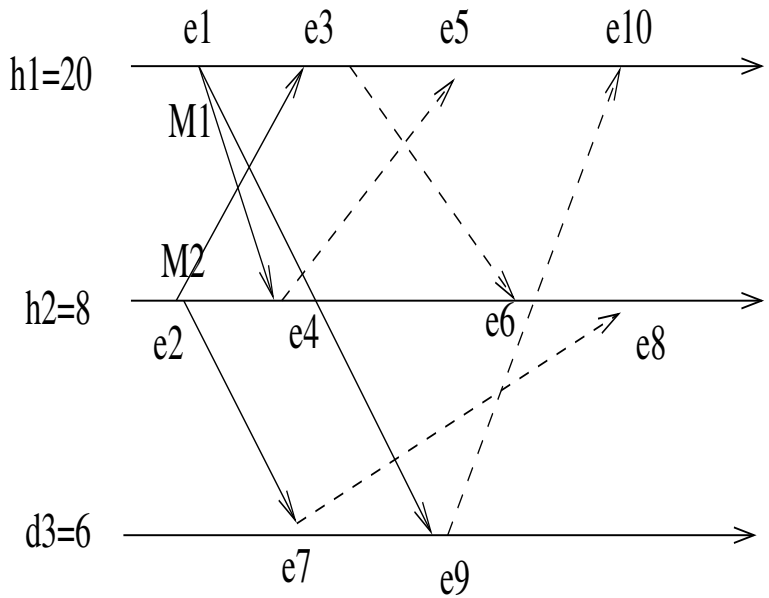
En e5, P1 sait qu'il a reçu tous les messages provenant de P2 de date inférieure à 21 (donné par l'acquittement), donc tout message venant de P2 aura donc une date supérieure à 22 donc à la date de M1, il sait qu'il peut traiter M1.

Mais que ce passe t'il dans ce cas :

Réponse :

Diffusion
Protocole utilisant des estampilles

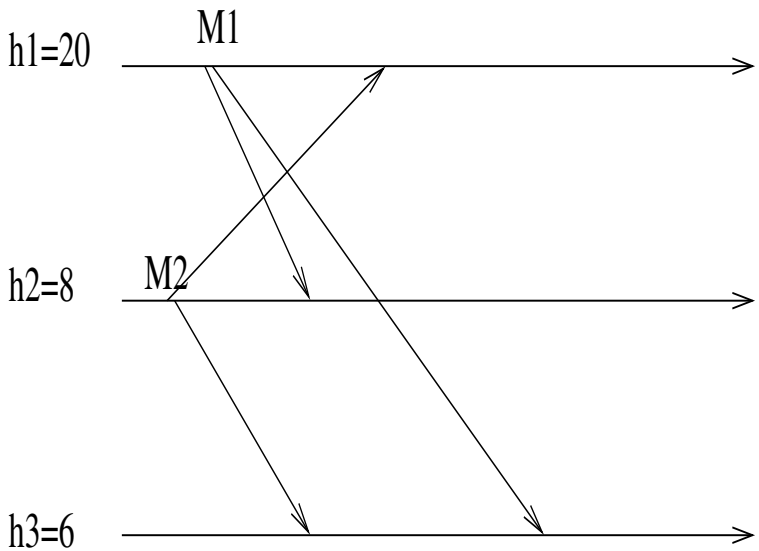


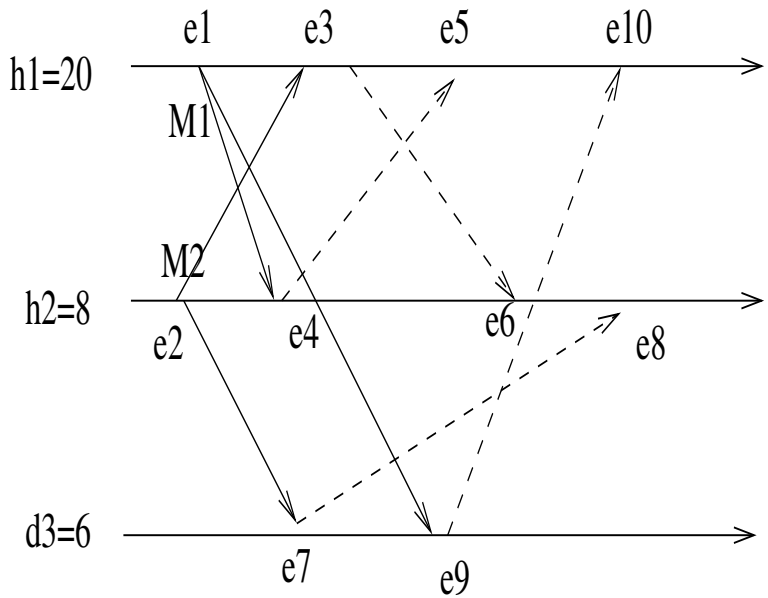


Deux problèmes :

- P1 et P2 sont obligés d'attendre les acquittements de P3

Diffusion
Protocole utilisant des estampilles

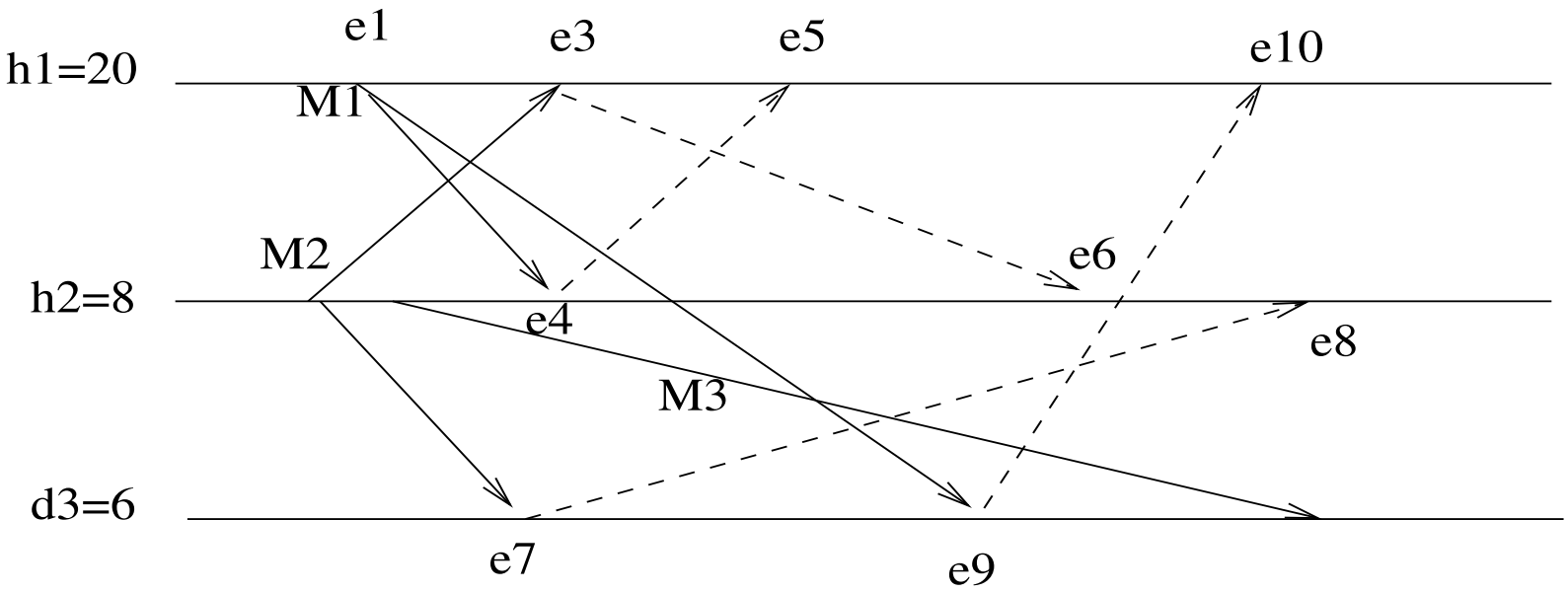




Deux problèmes :

- P1 et P2 sont obligés d'attendre les acquittements de P3
- ne sait pas quand il peut traiter les deux messages.

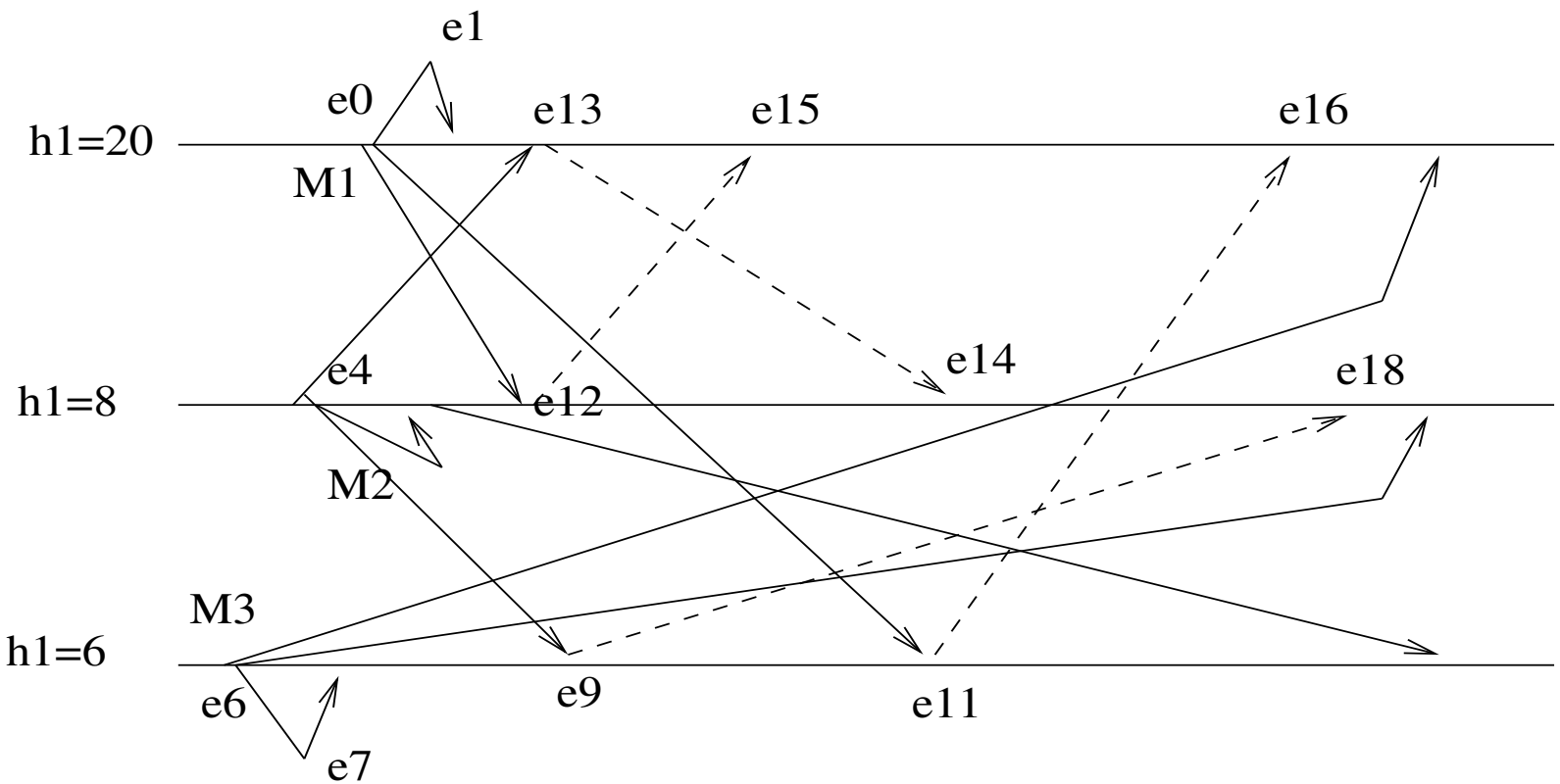
En effet, il peut être dans le cas suivant :



Il faut traiter M3 avant. Et on peut continuer cela longtemps. Une solution serait que les sites acquittent vers tous les autres sites et non uniquement vers l'émetteur.

Mais, cela ne résout pas le cas où **le réseau n'est pas FIFO (sauf perte de message uniquement**

Diffusion
Protocole utilisant des estampilles



Pour P1 et P2, M3 n'existe pas et ils n'ont aucune raison de l'attendre !

Donc, en e18, P2 croit pouvoir utiliser M2 et M1.

Or si on conserve la date d'émission pour ordonner les message alors il faudra traiter M3 avant tous les autres messages.
D'où comment garantir que les messages seront traités dans le même ordre sur toutes les machines sans perdre trop de temps ?
$\Longrightarrow$ on va utiliser les dates de validation c'est-à-dire les dates où un message a été acquitté (et donc reçu) par tous les sites.

Chaque site S_i entretient une horloge vectorielle $H_i[N]$. Chaque envoi de message sera daté par celle-ci : la relation de précédence causale entre les messages pourra donc être retrouvée (propriété des horloges vectorielles).

- avant de diffuser un message M, S_i traite M puis S_i exécute $H_i[i]++$ et estampille M par $V_M = H_i$

Diffusion
Protocole respectant l'ordre causal

Chaque site S_i entretient une horloge vectorielle $H_i[N]$. Chaque envoi de message sera daté par celle-ci : la relation de précédence causale entre les messages pourra donc être retrouvée (propriété des horloges vectorielles).

- avant de diffuser un message M, S_i traite M puis S_i exécute $H_i[i]++$ et estampille M par $V_M = H_i$
- à la réception d'un message M envoyé par S_i et estampillé par V_M, S_j le met en attente jusqu'à ce que :

Chaque site S_i entretient une horloge vectorielle $H_i[N]$. Chaque envoi de message sera daté par celle-ci : la relation de précédence causale entre les messages pourra donc être retrouvée (propriété des horloges vectorielles).

- avant de diffuser un message M, S_i traite M puis S_i exécute $H_i[i]++$ et estampille M par $V_M = H_i$
- à la réception d'un message M envoyé par S_i et estampillé par V_M, S_j le met en attente jusqu'à ce que :
  - $V_M[i] = H_j[i] + 1$

Chaque site S_i entretient une horloge vectorielle $H_i[N]$. Chaque envoi de message sera daté par celle-ci : la relation de précédence causale entre les messages pourra donc être retrouvée (propriété des horloges vectorielles).

- avant de diffuser un message M, S_i traite M puis S_i exécute $H_i[i]++$ et estampille M par $V_M = H_i$
- à la réception d'un message M envoyé par S_i et estampillé par V_M, S_j le met en attente jusqu'à ce que :
  - $V_M[i] = H_j[i] + 1$
  - et $\forall k \neq i,\ V_M[k] \leq H_j[k]$

Diffusion
Protocole respectant l'ordre causal

Chaque site S_i entretient une horloge vectorielle $H_i[N]$. Chaque envoi de message sera daté par celle-ci : la relation de précédence causale entre les messages pourra donc être retrouvée (propriété des horloges vectorielles).

- avant de diffuser un message M, S_i traite M puis S_i exécute $H_i[i]++$ et estampille M par $V_M = H_i$
- à la réception d'un message M envoyé par S_i et estampillé par V_M, S_j le met en attente jusqu'à ce que :
  - $V_M[i] = H_j[i] + 1$
  - et $\forall k \neq i,\ V_M[k] \leq H_j[k]$
- s après traitement de M, S_j exécute : $H_j[i]++$

Exemple :