ALGORITHMIQUE REPARTIE

LES COMMUNICATIONS

Chargée de cours: Lélia Blin
Transparents:http://www-npa.lip6.fr/~blin/Enseignements.html
Email: lelia.blin@lip6.fr

DIFFUSION

- Nous allons nous intéresser dans ce cours
  - Aux protocoles de communications
  - En particulier la diffusion
    - Un site donné envoie un message
    - vers tous les autres sites

DIFFUSION

- La diffusion est très largement utilisée
  - mise à jour de caches
  - envoie de résultat partiel
  - envoie d'information

PROTOCOLE DE DIFFUSION

- Sert à fournir à une application un service fiable
- Il doit gérer
  - les réceptions
  - et envois de messages avec le réseau
- Il sert de filtre entre le réseau et l'application

DIFFUSION

L’application veut les messages avec certaines garanties

↑ Délivrer

Le protocole est chargé de délivrer les messages à l’application avec ces garanties

↑ Recevoir

Réseau

PROTOCOLE DE DIFFUSION

- Lorsque le protocole
  - est sûr qu'un message donné est «bon»
  - il utilise la primitive **Délivrer**
    - pour faire passer l'information à l'application
  - Il peut y avoir un décalage entre
    - le moment où le site reçoit un message
    - et où l'application prend en compte le message

Diffusion asynchrone en cas de pannes de sites

Lélia Blin

Université d'Evry

PROTOCOLE DE DIFFUSION

- A partir d'un site p donné
- Dans un réseau physique complet
- Dans lequel les sites peuvent tomber en panne
- Soit V l'ensemble de tous les sites
- et $S_0=\{q_1,q_2,\ldots,q_t\}$ un sous-ensembles de sites ($S_0 \subset V$)
- Ce sous-ensemble de sites est nommé **sites relais**

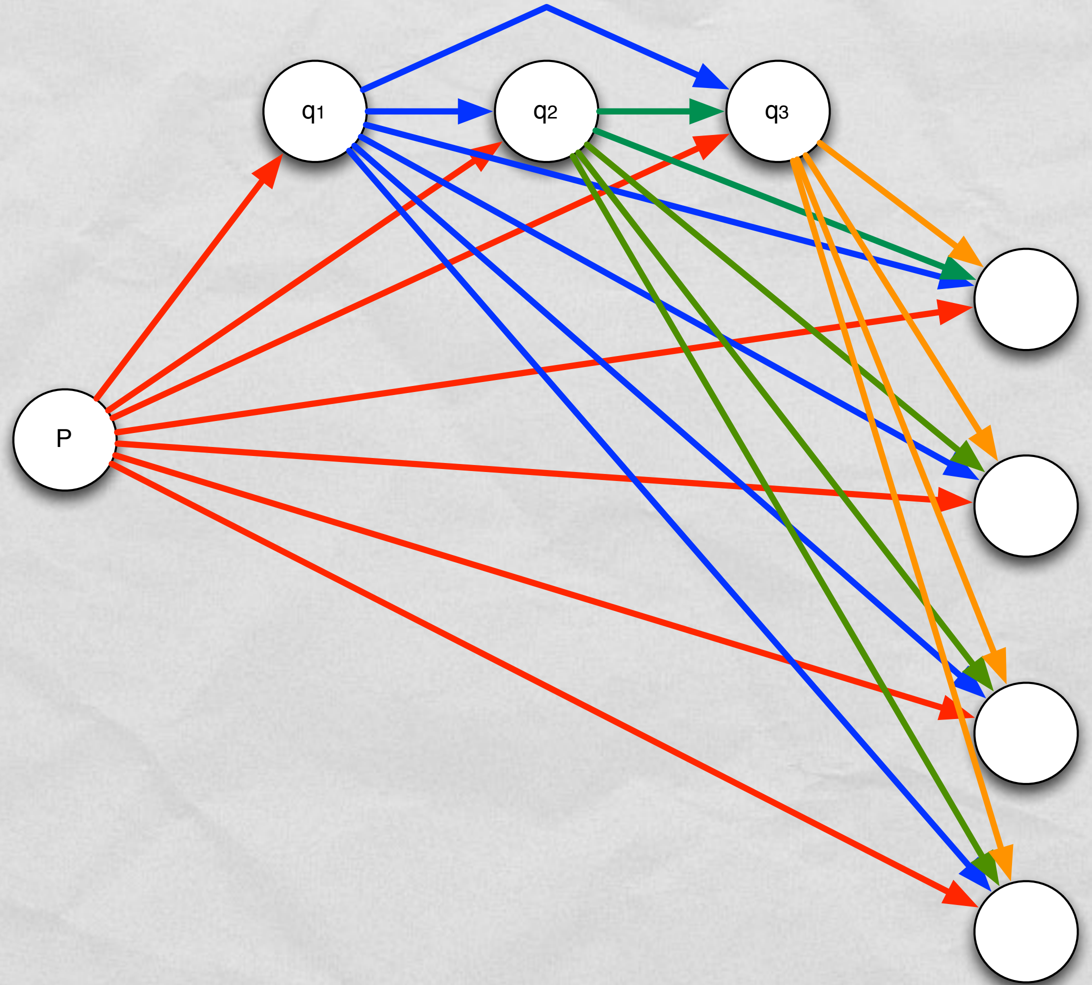
LA PROCÉDURE POUR $P \notin S_0$

```
Procédure diffuser (M)
    Envoyer (<M>) à q1,q2,...,qt dans cet ordre
    Pour tout (q∈V-S0-{p}) faire
        Envoyer (<M>) à q
```

LA PROCÉDURE POUR TOUT SITE Q

```
Lors de la réception de <M>
    si (q∈S_0) alors /* Supposons que q=q_k */
      si (k<t) alors
        Envoyer (<M>) à q_{k+1},...,q_t dans cet ordre
      Pour tout (r∈V-S_0-{p}) faire
        Envoyer (<M>) à r
      Accepter(M);
    Sinon Accepter (M);
```

EXEMPLE



REDONDANCE

- On voit dans cet exemple qu'un site va recevoir plusieurs fois le message
  - $|S_0|+|\{p\}|$ fois si il n'y a pas de panne
- Pour simplifier on supposera ici que
  - la fonction **Accepter** ne délivre à l'application qu'un seul exemplaire de chaque message diffusé par p

PANNE(S)

- On appelle ici panne d'un site
  - l'arrêt soudain de se site
  - lorsque un site est en panne il ne fait plus rien
  - cet arrêt est définitif

GARANTIE

- Comment se déroule l'algo lorsqu'il y a une panne?
- Lorsqu'il y a s pannes?
- Que peut-on garantir?

POUR UNE SEULE PANNE

- Si p qui tombe en panne
  - si il a eu le temps d'envoyer au moins un message
  - alors tous les autres sites le recevront
  - sinon personne ne le reçoit
- C'est le principe du tout ou rien
- Si $q \in S_0$ tombe en panne tous les sites vont recevoir le message

POUR PLUSIEURS PANNES

Lemme: *Le protocole de diffusion proposé ne peut vérifier le principe du tout ou rien avec la présence de plus de t pannes*

- ***Preuve****:* Examinons la situation suivante
  - p envoie son message à q_1 et tombe en panne
- q_1 envoie son message à q_2 et tombe en panne
- q_2 envoie son message à q_3 et tombe en panne
- ...
- q_{t-1} envoie son message à q_t et tombe en panne
- q_t envoie son message à $r \in V - S_0 - \{p\}$ et tombe en panne
- D'après l'algo *r* accepte le message et ne fait plus rien
- Ainsi avec *t+1* pannes il y a un site valide qui accepté le message mais pas les autres
- Le principe du tout ou rien n'est pas respecté

COMPLEXITÉ EN NOMBRE DE MESSAGES

- Complexité lorsqu'il n'y a pas de panne
  - le site *p* envoie n-1
  - chaque site q_i envoie
    - $t-i+n-(t+1)=n-i-1$ messages
  - Les sites de $V\text{-}S_0$-{p} n'envoient pas de messages
  - Il y a donc
    - $n-1+\sum_{(i=1 \text{ à } t)} n-i-1$
    - $= n-1+t(n-1)-(t(t+1)/2)$
    - $= (t+1)(n-1-t/2)$ messages

Diffusion respectant l'ordre FIFO des messages

ORDRE FIFO DES MESSAGES

- Réseau asynchrone à *n* sites
- On supposera que les communications
  - entre deux sites
  - ne respectent pas l'ordre FIFO
  - que le temps d'acheminement est fini mais quelconque
  - On considère la diffusion par un site *p*
  - On veut créer un protocole de diffusion respectant l'ordre FIFO

ORDRE FIFO DES MESSAGES

- p va numéroter chaque message créé
- p va diffuser ce message avec ce numéro
  - *numéro d'envoi* ou *numéro de séquence*
- p a une variable locale num_envoi_p
  - c'est le numéro du dernier message diffusé par p

ORDRE FIFO DES MESSAGES

- Tous les autres sites *i*
  - ont une variable d'attente
  - dont la valeur doit correspondre au numéro d'envoi
  - pour que le message soit délivrer

PSEUDO-CODE POUR *P*

```
Initialisation:
    num_envoi_p:=0
Procédure diffuser(M)
    num_envoi_p:= num_envoi_p+1
    pour tout (x_p∈V-{p}) faire
      Envoyer (<M, num_envoi_p>) à x_p
```

CODE DU SITE *I*

```
Initialisation:
    seq_i:=1
 Lors de la reception de <M,num_envoi_M> de p
    Stoker(M)
    Attendre(num_envoi_M=seq_i)
    Délivrer (M)
    seq_i=seq_i+1
    Détruire(M)
```

INCONVÉNIENTS DU PROTOCOLE

- Un site i
  - peut avoir à stocker beaucoup de messages
  - avant de pouvoir les délivrer
  - utilisation de beaucoup d'espace mémoire
  - Ex: si les messages arrivent dans l'ordre inverse d'envoi

INCONVÉNIENTS DU PROTOCOLE

- Si il y a perte d'un seul message
  - le protocole peut être bloqué
- Le numéro de séquence des messages
  - croît au delà de toute limite raisonnable
  - si *p* diffuse beaucoup de message
- problème de la taille des messages

SOLUTIONS

- On peut mettre en place
  - un système d'accusé de réception
  - ou d'acquittement dans une fenêtre t fixée

SOLUTIONS

- Dans ce système le site *p*
  - fait au plus t diffusion de suite
  - avant de recevoir des acquittements
  - chaque fois qu'un site i
    - a pu délivrer un message
    - il envoie un acquittement à p
    - comprenant le numéro de message acquitté
  - Lorsque p a reçu les acquittement de tous les sites
    - pour les derniers messages envoyés non encore acquittés
    - il peut continuer à diffuser

Diffusion respectant l'ordre causal

Lélia Blin

Université d'Evry

DIFFUSION RESPECTANT L'ORDRE CAUSAL

- Le système est composé de n sites
- Qui font des diffusions à n'importe quel moment
- On supposera le modèle
  - Sans panne
  - Avec un délais non bornés d'acheminement des messages
  - Des messages qui ne respecte pas forcément l'ordre FIFO

BUT

- Le but est de proposé
  - Un protocole qui permet à une application de
    - Recevoir les messages diffusés
    - Avec la garantie que les messages soient délivrées
      - Dans l'ordre induit par l'ordre causal.

PROTOCOLE

- On va construire ce protocole avec
  - Envoyer
  - Délivrer
  - Recevoir
- On va supposer qu'envoyer n-1 message
  - se fait en une seule opération atomique

PROTOCOLE

- Prenons m_1 et m_2 deux messages délivrés à P_i
- Si on a Envoyer(m_1)$\rightarrow$Envoyer(m_2)
- Alors on veut qu'en P_i on ait
  - Délivrer(m_1)$\rightarrow$Délivrer(m_2)

PRÉCÉDENCE IMMÉDIATE DE MESSAGES

- si m_1 et m_2 sont deux messages, on dira
  - m_1 précède immédiatement m_2
  - ou $m_1 \rightsquigarrow m_2$
- si
  - $Envoyer(m_1) \rightarrow Envoyer(m_2)$
  - $\nexists$ m_3 tel que
    - $Envoyer(m_1) \rightarrow Envoyer(m_3)$
    - et $Envoyer(m_3) \rightarrow Envoyer(m_2)$

PRÉCÉDENCE IMMÉDIATE DES MESSAGES

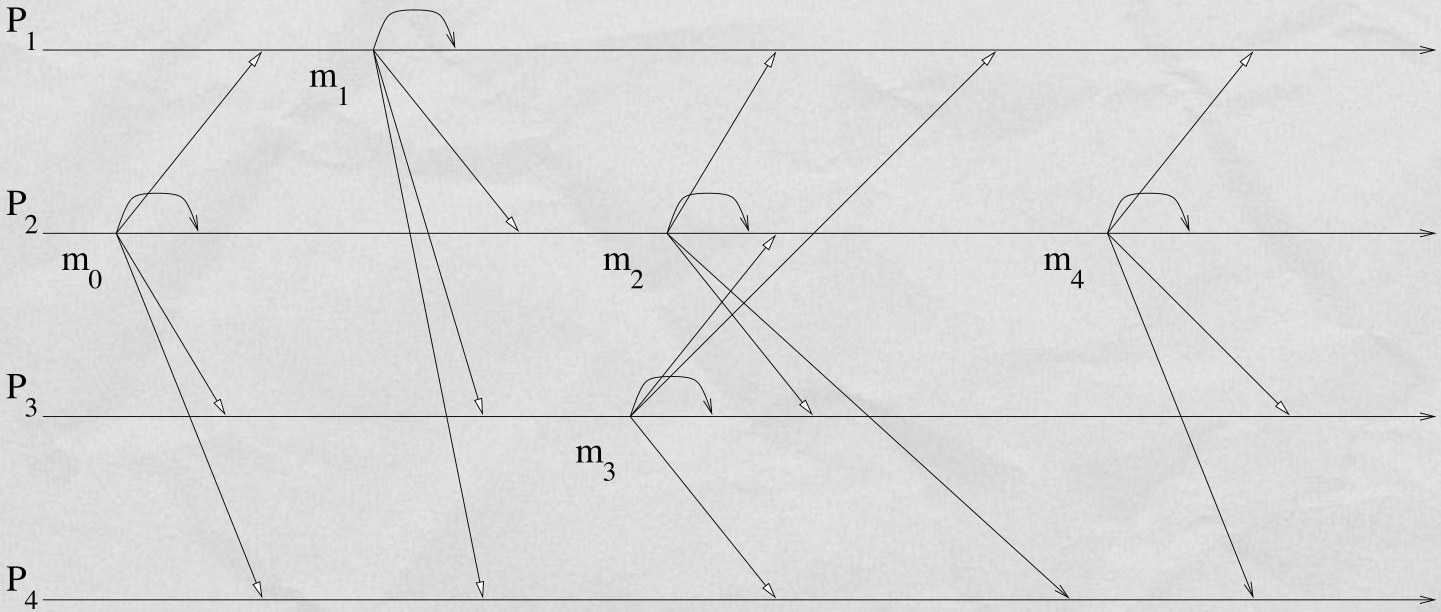


FIG. 3.3 – *5 diffusions.*

GRAPHE DE PRÉCÉDENCE IMMÉDIATE

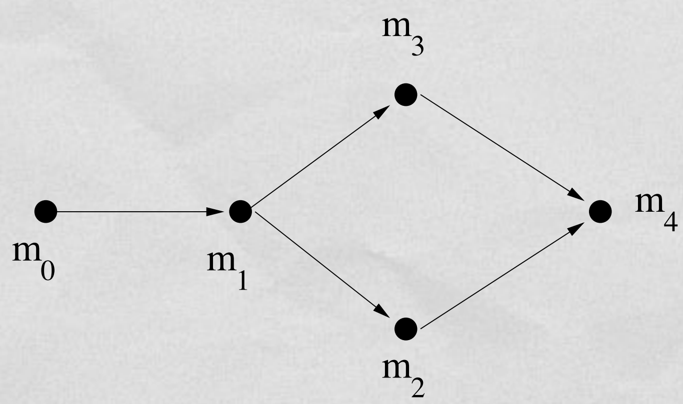


Fig. 3.4 – *Graphe de précédence immédiate des messages de la figure 3.3.*

PROTOCOLE

- On numérotera chaque site de 1 à n
- Chaque site P_i a
  - une variable locale:
    - *num_envoi_i*
  - *Un tableau à n cases*
    - *$DEL_i[j]$*
  - *$DEL_i[j]=d$*
    - le dernier message diffusé à partir de P_j et délivré à P_i avait le numéro d

PROTOCOLE SUITE

- Chaque message M sera identifié par la paire
  - (id,num)
    - id est l'identifiant de l'envoyeur
    - num le numéro du message
- Chaque message transportera aussi
  - un ensemble CB_M (Barrière causale)
  - composé des identificateurs des messages
    - qui précédent immédiatement M

PRISE EN COMPTE DES CONTRAINTES DE PRÉCÉDENCE

- Lorsqu'un site P_i reçoit un message M avec de telles données
- Sous quelles condition peut-il délivrer M
- En prenant en compte les contraintes de précédence immédiates?

PRISE EN COMPTE DES CONTRAINTES DE PRÉCÉDENCE

- La condition est:
  - $\forall (k,d) \in CB_M : d \leq DEL_i[k]$
- Cela exprime que
  - Toutes les prédécesseurs immédiats de M
    - ont déjà été délivrer

PSEUDO-CODE DU PROTOCOLE

Initialisation:

$num\_envoi_i := 0$

$CB_i := \emptyset$

Procedure diffuser(M):

num_envoi_i++

Pour tout ($x_i \in V$) faire

Envoyer(<M, num_envoi_i, CB_i) à x_i

$CB_i := (i, num\_envoi_i)$

Lors de la réception de (<M, num_envoi_i, CB_M>) de P_j

Attendre($\forall (k,d) \in CB_M$: $d \leq DEL_i[k]$)

$DEL_i[j] := num_M$

$CB_i := CB_i - CB_M \cup \{(j, num_M)\}$*

Delivrer(M)

Rm: Le stockage et la destruction ne sont pas indiqué ici mais sont implicite
* A faire de façon atomique

LEMME

Soit m_0 identifié par (k_0,d_0) et m_1 tels que $m_0 \rightarrow m_1$ alors $(k_0,d_0) \in CB_{m_1}$

PREUVE

- Soit P_i l'envoyeur de m_1.
- Comme $m_0 \rightarrow m_1$ deux cas doivent être considérés.
  - m_0 a été délivré à P_i
  - m_0 a été envoyé à P_i
- Dans les deux cas
  - P_i met à jour CB_i (voir algorithme) en prenant compte de m_0.
- La délivrance d'un message m' entre m_0 et l'envoie de m_1 soit ne change rien soit est impossible (car $m_0 \rightarrow m_1$).

THÉORÈME

Les délivrances de messages respectent l'ordre causal

PREUVE

- Considérons deux messages m_0 et m_x tels que:
  - **Envoyer**(m_0)$\rightarrow$**Envoyer**(m_x)
  - et m_0 et m_1 sont délivrés à P_i
- Il faut montrer que m_0 est délivré avant m_x en P_i.

PREUVE PAR RÉCURRENCE

- On montre cela par **récurrence**
  - sur le longueur *L* du chemin entre m_0 et m_x
  - dans le graphe des précédence immédiate des messages.
- **Cas de base**:
  - si $L=1$, $m_0 \rightarrow m_x$
  - et d'après le résultat du lemme précédent, $m_0 \in CB_x$.
  - Or P_i va attendre d'avoir délivré m_0
  - avant de délivrer m_x
    - c'est l'attente de l'algorithme lors de la réception de m_x.

PREUVE PAR RÉCURRENCE

- **Hypothèse de récurrence:**
  - lorsque le chemin de causalité est de longueur strictement inférieure à $L \geq 2$,
  - les contraintes de précédence sont respectées.

PREUVE PAR RÉCURRENCE

- Considérons maintenant
  - un chemin de longueur *L* de m_0 à m_x:
    - $m_0 \rightarrow m_1 \rightarrow .... \rightarrow m_{L-1} \rightarrow m_x$
- Ainsi par hypothèse:
  - tous les messages $m_0, m_1, ..., m_{L-1}$ délivrés à P_i
  - le sont suivant l'ordre causal.

PREUVE PAR RÉCURRENCE

- Le message m_L est délivré à P_i.
- Comme $m_{L-1} \rightarrow m_x$,
  - m_{L-1} est délivrer avant m_x (c'est le cas de base).
- De plus, comme par hypothèse de récurrence on a
  - $m_0 \rightsquigarrow m_1 \rightsquigarrow .... \rightsquigarrow m_{L-1}$
- m_0 est délivré avant m_{l-1}.
- Ainsi m_0 est délivré avant m_x en P_i.

$\square$