

Quelques algorithmes répartis

et leur usage pour les systèmes

Michel RIVEILL

Projet SIRAC

INRIA Rhône-Alpes

655, avenue de l'Europe

38330 Montbonnot

***D'après une première série de transparents réalisée par
Sacha Krakowiak***

Exemple d'utilisation des estampilles : l'exclusion mutuelle répartie

■ Principe : “file d'attente répartie”

- représentation partielle sur chaque site
- ordre global réalisé par estampilles
- Demande d'entrée d'un processus H_i en section critique :
 - demande d'autorisation à tous les autres
 - entrée en section critique après accord de tous
- Réception par p_j d'une demande de p_i
 - si p_j non candidat ou non en s.c. : accord
 - si p_j candidat :
 - accord si demande de p_i antérieure
 - sinon enregistrement de la demande
 - si p_j en section critique :
 - réponse différée
 - en fin de s.c. : accord à toutes les demandes en attente
- Algorithme sans interblocage ni privation
(entrée en s.c. dans l'ordre global des demandes)

■ Problèmes de fiabilité

- retrait et réinsertion d'un site
- résistance aux défaillances

Exclusion mutuelle répartie

Structure des données

Date de la demande
<date>.<pid>

2.1	*	*	*
f	1	3	2

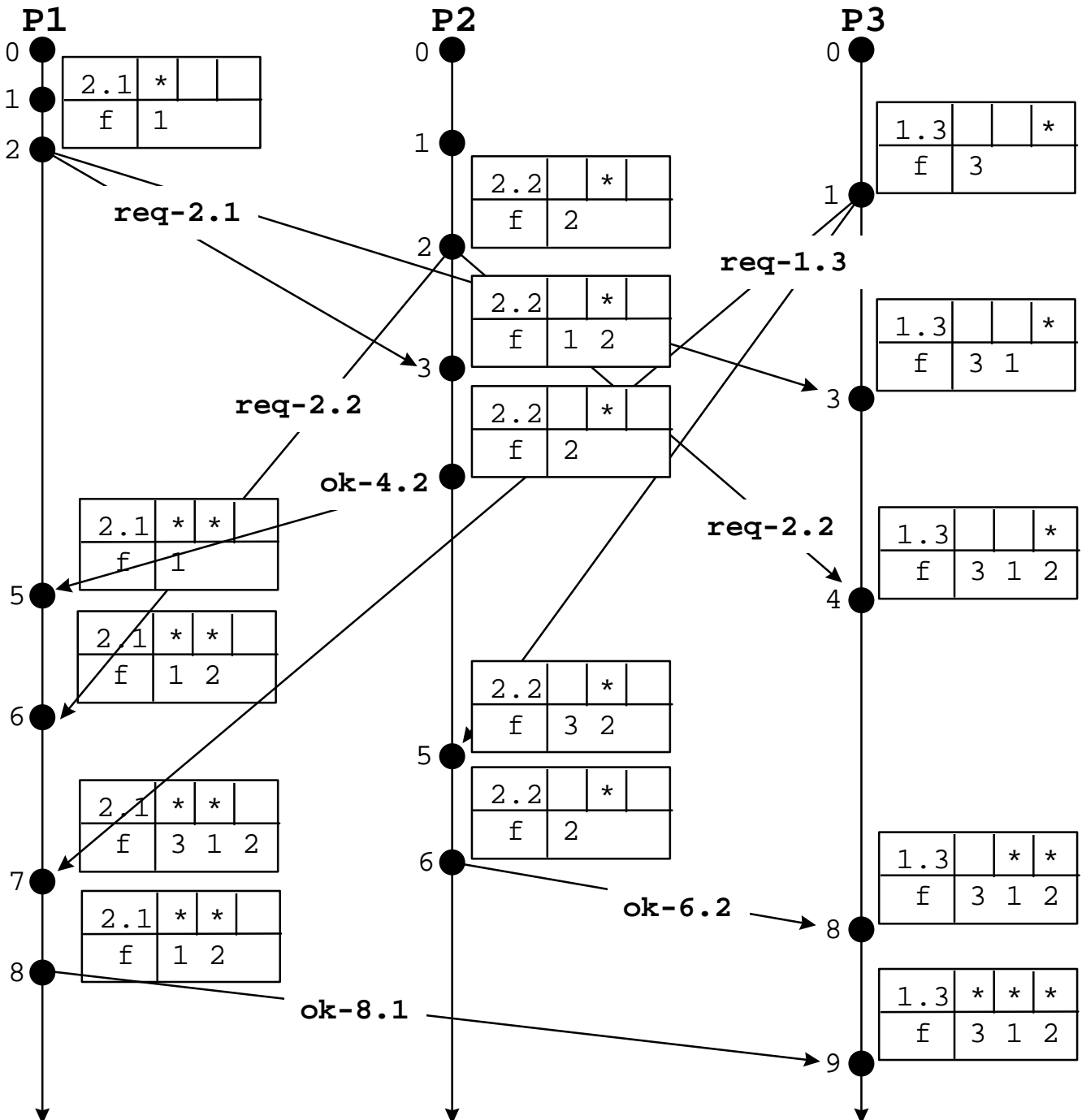
Accord des processus

Ordre des requêtes

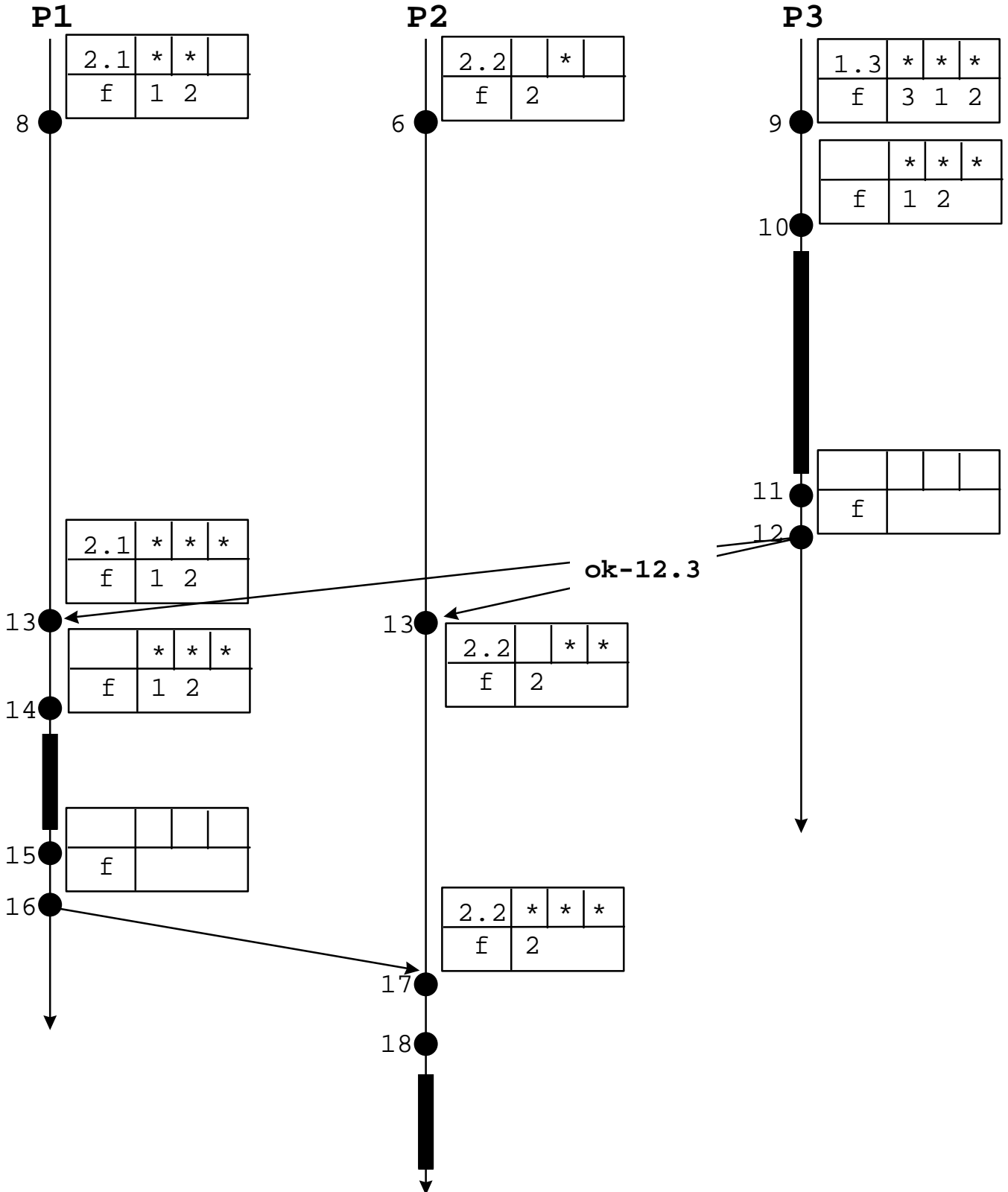
Messages échangés

req-<date>.<pid>

ok-<date>.<pid>



Exclusion mutuelle répartie



Exclusion mutuelle par jeton circulant

■ *Anneau virtuel*

- *définit un ordre (arbitraire) sur un ensemble de sites*

■ *Jeton*

- *message spécial circulant sur l'anneau*
- *le détenteur du jeton peut entrer en section critique*
- *application : système de communication*

■ *Problèmes de fiabilité*

- *nécessité de garantir*
 - l'intégrité de l'anneau
 - l'existence et l'unicité du jeton
- *mécanismes d'administration*
 - retrait-réinsertion d'un site
- *mécanismes de détection de défaillances*
 - panne d'un site quelconque
 - perte du jeton entre deux sites
 - panne du détenteur du jeton
 - détection par délai de garde + mécanismes de régénération
 - garantie d'unicité ("élection") [traité plus loin]

■ Définition

- *Election = choix d'un processus et d'un seul dans un ensemble de processus équivalents*
- *L'élection peut être déclenchée par plusieurs processus "simultanément"*
- *En pratique :*
 - le nombre des processus est connu a priori
 - les processus sont identifiés par un nombre (n° de processus, de site, etc)
 - on cherche l'extremum (max ou min)

■ Usage

- *Algorithmique répartie : "brisure de la symétrie"*
- *Choix d'un processus maître (notamment après défaillance)*
 - pour la coordination d'un ensemble de processus
 - la régénération d'une information perdue (ex : jeton)
 - le contrôle de concurrence (séquencement)

Algorithme utilisant un anneau [Chang-Roberts 79]

■ *Principe*

- *Extinction progressive des messages par filtrage*

■ *Réalisation*

- *Anneau virtuel unidirectionnel*
- *Programme du site i :*

détection de la perte du jeton :

 candidat := true ;

 envoyer_suivant (élection, i) :

réception du message (élection, j) :

 select

$j > i$: envoyer_suivant (élection, j) ;

$j < i$: if \neg candidat then

 candidat := true;

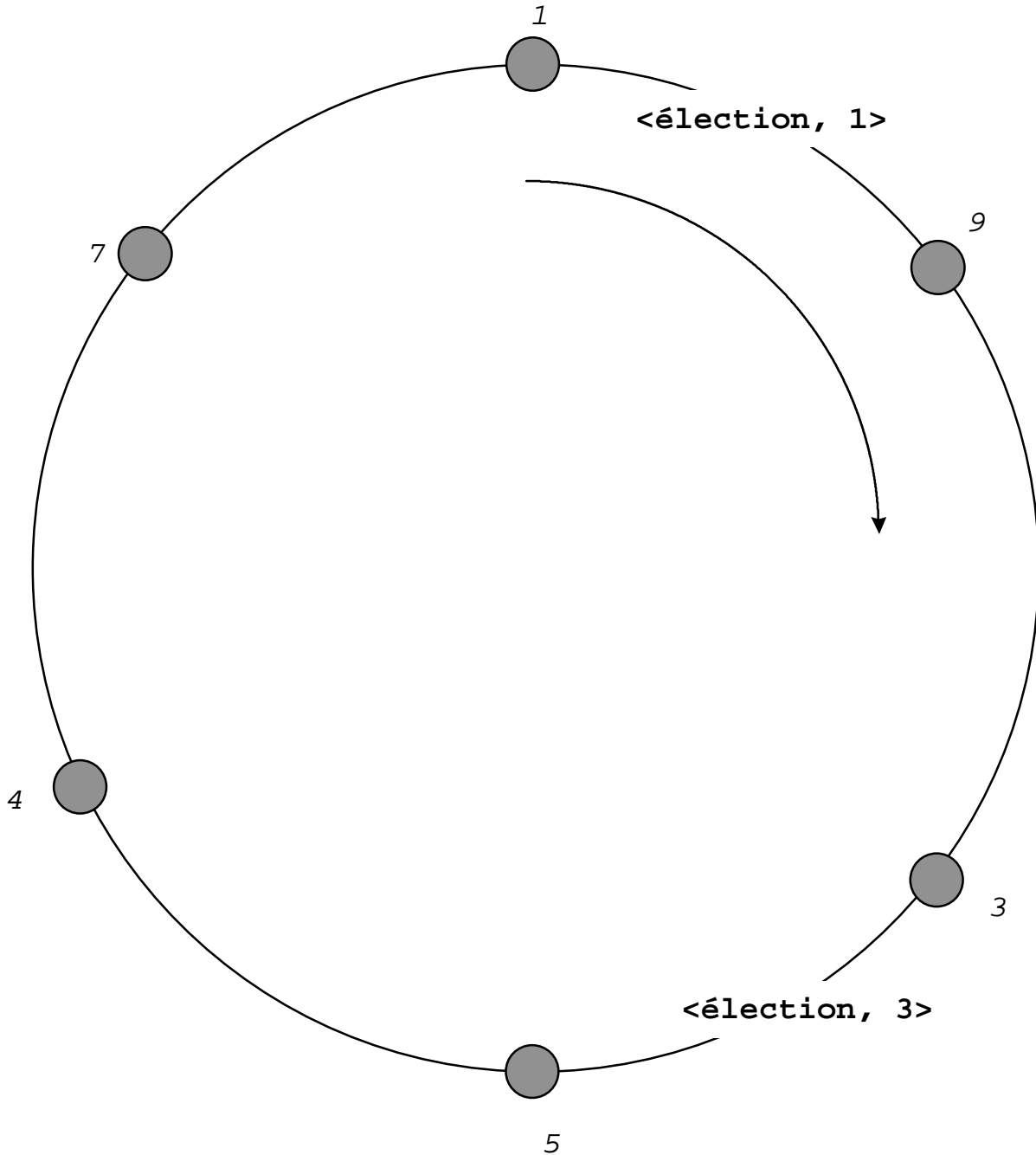
 envoyer_suivant (élection, i)

 end ;

$j = i$: régénérer_jeton -- site i élu

 end

Election (exemple)



Compléter l'exemple

Maintien de l'intégrité d'un anneau virtuel

■ *Retrait d'un site*

- *Message aux voisins*
- *Accord entre voisins pour mise à jour*

■ *Réinsertion d'un site*

- *Comme retrait*
- *Diffusion + élection si anciens voisins en panne*

■ *Panne d'un site*

- *Echange de message périodique entre voisins*
- *Réaction après délai de garde*

Diffusion fiable - Utilisations

- ***Diffusion = envoi d'un message à un ensemble de processus spécifié***
- ***Coordination***
 - *Coopération d'un ensemble de processus (groupe) à une tâche commune*
 - Exemple : réalisation d'un serveur par un groupe de processus
- ***Entretien d'un état partagé***
 - *Copies multiples d'une information*
 - Motivation : efficacité, disponibilité
 - Contraintes de cohérence plus ou moins fortes
- ***Synchronisation***
 - *Gestion du temps physique*
 - *Synchronisation d'applications (synchronisme virtuel)*
- ***Réalisation d'algorithmes répartis***
 - *Election*
 - *Terminaison*
- ***Administration***
 - *Connexion-déconnexion de sites*

Diffusion fiable - Spécifications

■ *Fiabilité*

- *Un message diffusé à un groupe est reçu par tous les processus du groupe en état de le recevoir (non en panne), ou par aucun.*

■ *Séquencement des messages*

- *Deux types de contraintes peuvent être imposées :*
- *Ordre de réception identique pour tous les processus récepteurs*
 - Pour un émetteur unique et un seul groupe récepteur
 - Pour un ensemble d'émetteurs et un seul groupe récepteur
 - Pour un ensemble d'émetteurs et plusieurs groupes récepteurs ayant des membres communs
- *Préservation de l'ordre d'émission*
 - pas de contraintes
 - ordre global
 - ordre causal

■ *Principe des algorithmes*

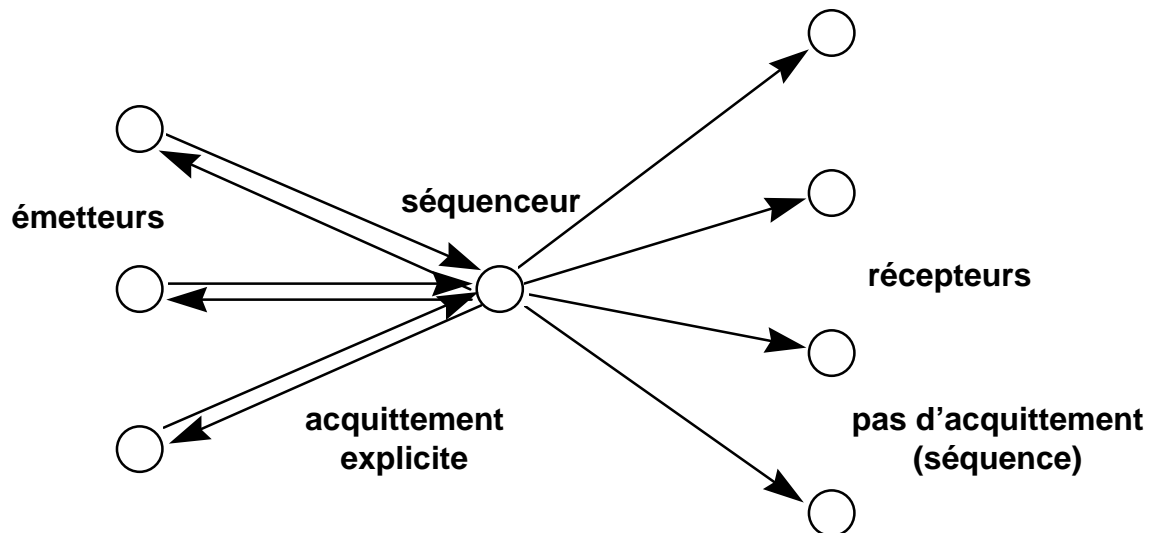
- *les messages sont estampillés (ordre total ou partiel)*
- *les messages sont acceptés dans l'ordre des numéros*
- *le séquencement peut être assuré par le système de communication*
- *détection des messages manquants*

Algorithmes à séquenceur

[Chang & Maxemchuk 84]

■ Principe

- Protocole réalisé sur un système à diffusion (tous les sites reçoivent tous les messages)
- On se ramène à des communications $n \rightarrow 1$ ou $1 \rightarrow n$ (au lieu de $n \rightarrow p$)
- Schéma logique :



■ Réalisation

Problème : résistance aux défaillances

- séquenceur fixe \Rightarrow délai non borné si communication non fiable
- tampon “infini” pour conserver copies
- problème de la défaillance du séquenceur

1 solution : séquenceur circulant

Diffusion fiable par séquenceur (suite)

Tolérance aux défaillances

■ *Hypothèses*

- *défaillance des sites détectable et franche (“fail-stop” : comportement normal ou arrêt)*
- *perte de messages détectée*

■ *Principe : protocole à 2 phases*

- *phase 1 : réception du message par tous les sites (préparation)*
- *phase 2 : remise du message à tous les destinataires (validation)*
- *circulation du site séquenceur (jeton sur anneau) entre tous les sites*
- *jeton = acquittement d’un message m*
 - *un site n’accepte le jeton que s’il a tous les messages précédant m . Après 1 tour, tous les sites ont tous les messages*
 - *un site ne délivre un message que si L autres sites ont accepté le jeton (tolère $L+1$ défaillances). Après 2 tours, le message a été délivré partout si L sites dans l’anneau*

■ *Protocole de restructuration pour changer l’anneau virtuel*

Diffusion fiable par séquenceur (suite)

■ Evaluation

- Famille de protocoles paramétrée par nb de diffusions avant passage du jeton : compromis entre
 - taille du tampon et nombre de messages de service
 - tolérance aux défaillances et efficacité

<i>Transfert du jeton</i>	<i>Nb messages service</i>	<i>Taille tampon</i>
<i>1 par message</i>	$1 + P0$ (€ 2)	$N - 1$
<i>1 par K messages</i>	$1 + P0$ (fi 1 si Kfi ¥)	$K (N - 1)$
<i>K par message</i>	$P0 + K$ (fi N si Kfi N - 1)	$(N - 1)/K$

$P0$ = Probabilité {file de messages en attente vide}

(mesure du degré de charge : $P0$ faible => charge forte)

Diffusion fiable par séquenceur

[Kaashoek & al. 89, 91]

■ *Hypothèses*

- *Séquenceur = site fixe*
- *Groupes de diffusion (gestion dynamique)*
- *Objectif : efficacité sur réseau Ethernet (voie à diffusion) pour diffusion fréquente de messages brefs*
- *Réception FIFO pour un même émetteur, ordre uniforme*

■ *Fonctionnement*

- *Tampon “historique” maintenu par le site séquenceur*
 - *numéro de séquence attribué par le séquenceur*
 - *remise aux destinataires dans l’ordre de séquence*
 - *détection des “trous” de numérotation par les destinataires -> redemande*
- *Pb : tampon potentiellement non borné (défaillances des communications)*
 - *tampon plein -> resynchronisation et renvoi puis vidage du tampon*

■ *Tolérance aux défaillances*

- *Défaillances des communications : détection des “trous” et demande de réémission*
- *Défaillance du séquenceur :*
 - *élection d’un nouveau séquenceur (avec le n° maximal de séquence)*
 - *conservation d’un double de l’historique*

Séquencement par estampilles

[Birman & Joseph 87]

■ *Principe*

- *Distribution à chaque site dans le même ordre (ordre total)*
- *Accord par protocole à 2 phases*
- *Tolérance aux défaillances*

■ *Réalisation*

- *Les messages sont estampillés (estampille “provisoire” = {date d’émission, n° site émetteur})*
- *Les messages sont mis en attente sur le site de réception*
- *Phase 1 : préparation.*
 - *Chaque site propose à l’émetteur une nouvelle estampille pour chaque message en attente (supérieure à toutes celles connues sur le site)*
- *Phase 2 : validation.*
 - *L’émetteur attribue une estampille définitive (max) à chaque message et la rediffuse*
 - *Chaque site peut alors délivrer les messages, dans l’ordre des estampilles définitives (ordre global uniforme)*
 - *Coût élevé : 3n messages*

■ *Tolérance aux fautes*

- *Défaillance d’un récepteur : rien*
- *Défaillance de l’émetteur : élection d’un remplaçant. Pour messages en attente : interroge les autres récepteurs ; si prêt, estampille définitive ; sinon, rediffuse*

Diffusion fiable avec estampilles

ABCAST [Birman & Joseph 87]

■ Hypothèses

- chaque site i diffuse le message m_i
- estampille provision : $\langle \text{date émission.numéro site} \rangle$
- nouvelle estampille à la réception du message

Site 1		
m_3	m_1	m_2
15.1	16.1	17.1
A	A	A

Site 2		
m_2	m_1	m_3
16.2	17.2	18.2
A	A	A

Site 3		
m_1	m_3	m_2
17.3	18.3	19.3
A	A	A

Etape 1 : diffusion des messages

17.2 17.3

Etape 2 : proposition d'une estampille

$$E = \max(16.1; 17.2; 17.3) = 17.3$$

m_3	m_2	m_1
15.1	17.1	17.3
A	A	P

Etape 3 : attribution de l'estampille définitive

m_3	m_2	m_1
15.1	17.1	17.3
A	A	P

m_2	m_1	m_3
16.2	17.3	18.2
A	P	A

m_1	m_3	m_2
17.3	18.3	19.3
P	A	A

Etape 4 : diffusion de l'estampille définitive

m_3	m_2	m_1
15.1	17.1	17.3
A	A	P

m_2	m_1	m_3
16.2	17.3	18.2
A	P	A

m_3	m_2	
18.3	19.3	
A	A	

**Etape 5 : message délivré si en-tête
et estampille définitive attribuée**

ABCAST [Birman & Joseph 87]

Site 1

<i>m3</i>	<i>m1</i>	<i>m2</i>
15.1	16.1	17.1
A	A	A

Site 2

<i>m2</i>	<i>m1</i>	<i>m3</i>
16.2	17.2	18.2
A	A	A

Site 3

<i>m1</i>	<i>m3</i>	<i>m2</i>
17.3	18.3	19.3
A	A	A

<i>m3</i>	<i>m2</i>	<i>m1</i>
15.1	17.1	17.3
A	A	P

<i>m2</i>	<i>m1</i>	<i>m3</i>
16.2	17.3	18.2
A	P	A

<i>m1</i>	<i>m3</i>	<i>m2</i>
17.3	18.3	19.3
P	A	A

<i>m3</i>	<i>m1</i>	<i>m2</i>
15.1	17.3	19.3
A	P	P

<i>m1</i>	<i>m3</i>	<i>m2</i>
17.3	18.2	19.3
P	A	P

<i>m3</i>	<i>m2</i>	
18.3	19.3	
A	P	

<i>m1</i>	<i>m3</i>	<i>m2</i>
17.3	18.3	19.3
P	P	P

<i>m3</i>	<i>m2</i>	
18.3	19.3	
P	P	

<i>m3</i>	<i>m2</i>	
18.3	19.3	
P	P	

Diffusion fiable avec ordre causal

[Birman, Schiper, Stephenson 90]

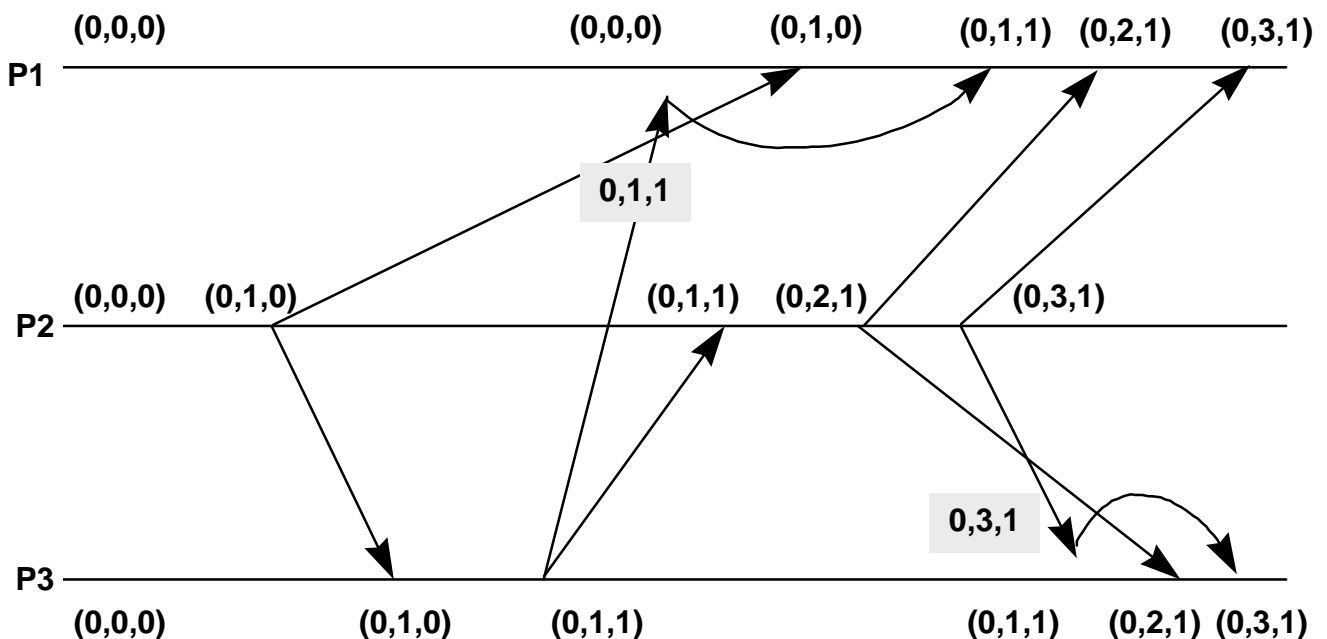
- *Avantage recherché : réduction du délai de diffusion (un message peut être délivré dès réception)*
- *Ordre partiel au lieu de l'ordre total*
- *Utilisation des horloges vectorielles (1 horloge V_i par processus p_i , ne compte que les émissions)*

1) Avant diffusion de m , p_i exécute $V_i[i] := V_i[i] + 1$
estampille de m : $V_m = V_i$

2) Réception par p_j de (m, V_m) envoyé par p_i : remise de m retardée jusqu'à ce que :

$$\begin{cases} V_j[i] = V_m[i] - 1 \\ \forall k \neq i : V_j[k] \leq V_m[k] \end{cases}$$

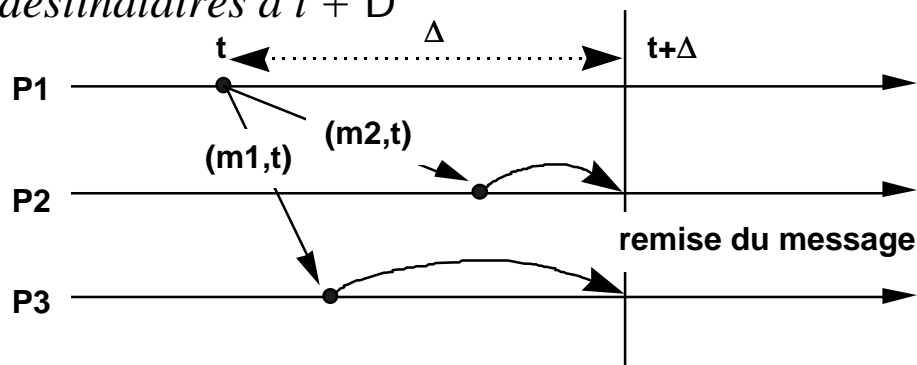
3) Après remise de m : $V_j := \max(V_j, V_m)$



Diffusion fiable avec contraintes de temps réel

- **Problème** : garantir que tous les destinataires d'un message diffusé le reçoivent "simultanément" (ou dans une fourchette spécifiée en durée réelle)
- **Hypothèse 1** : on dispose d'horloges physiques exactes

- Soit D une borne supérieure du délai de diffusion. Un message diffusé au temps t est remis aux destinataires à $t + D$



- **On sait calculer D en cas de pannes**

$$D = (t + d) d \quad t = nb \text{ max de processus fautifs}$$

- δ = délai max entre deux sites

- d = diamètre du réseau "survivant"

- **Hypothèse 2** : horloges physiques à dérive bornée

- Résultat (Cristian & al. 86) :

$$D = t(d + e) \wedge d + e$$

- si la dérive entre horloges est bornée par e

- **Systemes futurs** : connaissance exacte précise du temps physique (temps diffusé par satellite GPS)

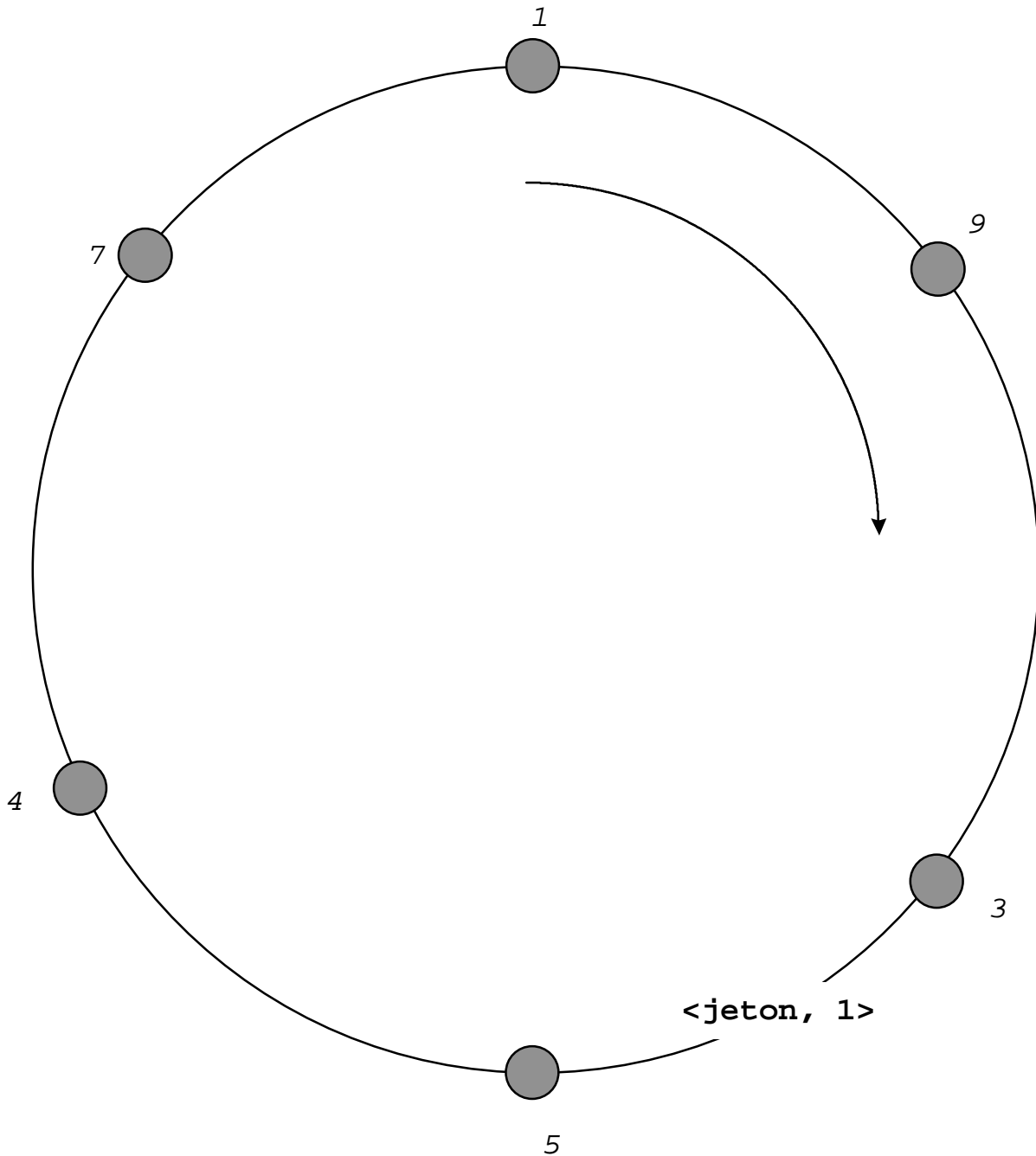
Terminaison

- ***Problème : détecter qu'un calcul réparti est bien terminé***
 - *tous les processus sont au repos*
 - *aucun message n'est en transit*
- ***C'est une propriété stable (détectable par examen de l'état global)***
- ***Utilisation***
 - *Tout algorithme réparti. Exemples :*
 - *calcul d'état global*
 - *ramasse-miettes réparti*
- ***Réalisation***
 - *Anneau virtuel, messages reçus dans l'ordre FIFO*
 - *Visite des sites par un jeton*
 - *Condition de terminaison : le jeton a trouvé tout site passif au cours de 2 visites successives*

Terminaison (suite)

- **Chaque site a deux états, blanc ou noir (noir = actif)**
- **Le jeton porte un compteur des sites trouvés passifs**
- **Programme d'un site :**
 - *attente d'un message :*
état := passif ;
 - *réception d'un message :*
état := actif ;
couleur := noir ;
 - *réception du jeton (valeur = j) :*
jeton_présent := true ;
nb := j ;
if nb = N and couleur = blanc then
 terminaison détectée
end ;
 - *émission du jeton :*
attendre (jeton_présent and etat=passif) ;
if couleur = blanc then
 envoyer_suivant(jeton, nb + 1)
else
 envoyer_suivant(jeton, 1)
end ;
couleur := blanc ;
jeton_présent := false;

Terminaison (exemple)



Compléter l'exemple

Conclusion sur les algorithmes répartis

■ *Méthodes de base*

- *Ordonnancement global*
 - horloges logiques, scalaires ou vectorielles
 - anneau virtuel
- *Antériorité : marqueur (si messages FIFO)*
- *Utilisation de graphes de processus*
 - cas particuliers : anneaux, arbres

■ *Tolérance aux fautes*

- *Types de fautes tolérables*
 - défaillances franches (“fail-stop”)
 - défaillances par omission
 - comportement byzantin
 - détection par signal / essais infructueux
- *Protocole d’accord réparti*
 - validation à 2 [ou 3] phases
 1. recherche de l’accord
 2. confirmation
 - [3. confirmation de la confirmation....]

■ *Compromis temps / place*

- (nb de messages / taille d’information conservée)

■ *Compromis degré de cohérence / complexité*