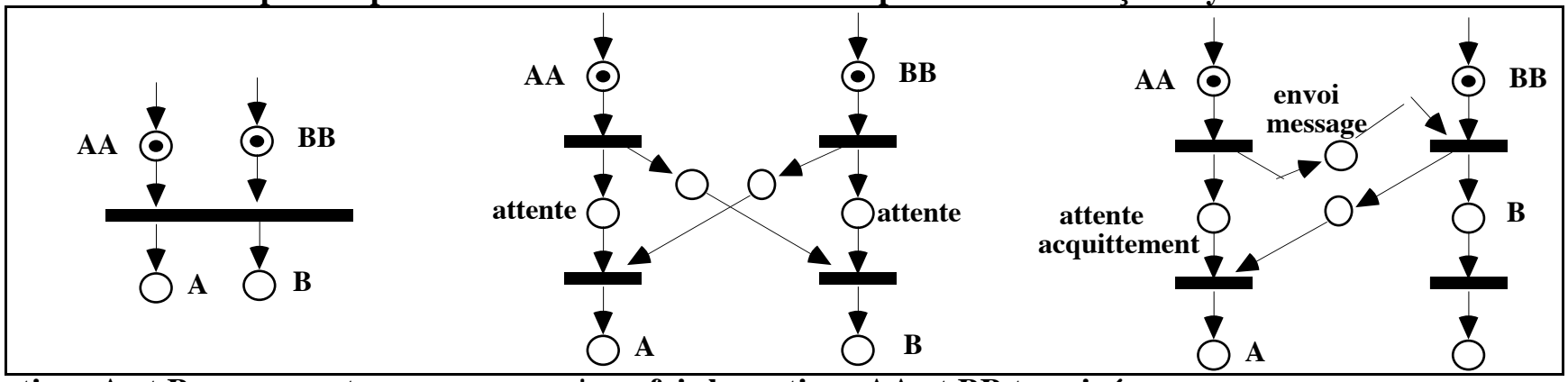
PASSAGE DE TÉMOIN

coopération par division du travail entre les processus

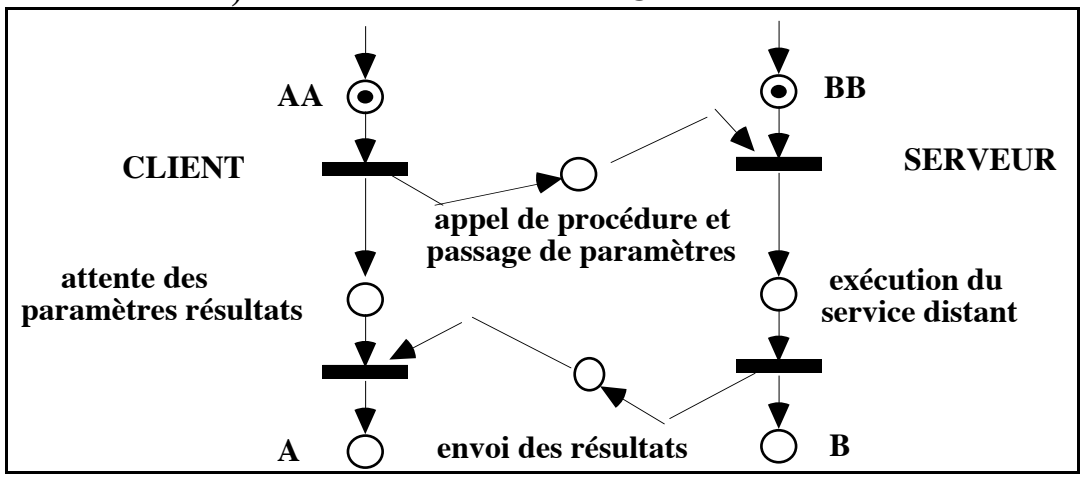
4) Rendez-vous à 2, symétrique, avec ou non une écriture ou un envoi de message (avec acquittement)

• Rendez-vous multiple : m processus au rendez-vous avant de poursuivre de façon asynchrone



les actions A et B ne peuvent commencer qu'une fois les actions AA et BB terminées.

5) Appel procédural (local ou distant). Invocation à distance. Client-serveur



Exemple de rendez-vous dissymétrique (en Ada) entre un CLIENT appelant et un SERVEUR appelé

COOPERATION : LES PRODUCTEURS ET LES CONSOMMATEURS

EXEMPLE 1: des processus utilisateurs préparent des fichiers de texte et déposent des requêtes au service d'impression. Ces requêtes sont satisfaites par l'un des processus de ce service, à un moment où une des imprimantes est disponible, ni en panne, ni en maintenance.
EXEMPLE 2: un processeur d'entrée-sortie, un canal, un organe d'accès direct à la mémoire, reçoivent des données externes, caractère par caractère, et les engrangent à la vitesse des organes périphériques dans une zône de mémoire tampon. Quand le tampon est plein, un processus spécialisé est alerté pour qu'il puisse les exploiter .
EXEMPLE 3: un processus d'un site émetteur dans un réseau envoie des messages à destination d'un processus d'un site récepteur; celui-ci doit être prêt à recevoir ces messages; toutefois l'émetteur ne doit pas les envoyer plus vite que le récepteur ne peut les recevoir.
EXEMPLE 4: un service de messagerie permet aux abonnés d'un réseau d'envoyer du courrier électronique; un abonné édite son courrier et le dépose dans la boite aux lettres du destinataire, que celui-ci soit en session ou non; un abonné lit son courrier, que l'émetteur soit ou non en session.
EXEMPLE 5: une cascade de relations de coopération peut aider à structurer une application. Ainsi dans la gestion d'un atelier de fabrication, un processus d'acquisition de messages reçoit les données des ponts roulants, des stocks et des machines outils et recueille des messages en provenance d'autres calculateurs de l'atelier. Ces données et ces messages sont déposés dans une file d'entrée; un processus analyseur les en extrait pour les filtrer, les mettre en forme et les déposer dans un tampon de requêtes; un processus interpréteur se charge de faire traiter le service demandé en appelant les sous-programmes adéquats; il prépare des réponses qu'il dépose dans une file de sortie ou bien il génère de nouvelles requêtes qu'il se dépose à lui-même pour plus tard dans le tampon des requêtes. Enfin un quatrième processus met en forme les réponses et regroupe dans un message toutes celles qui concernent le même destinataire.

LA TERMINOLOGIE

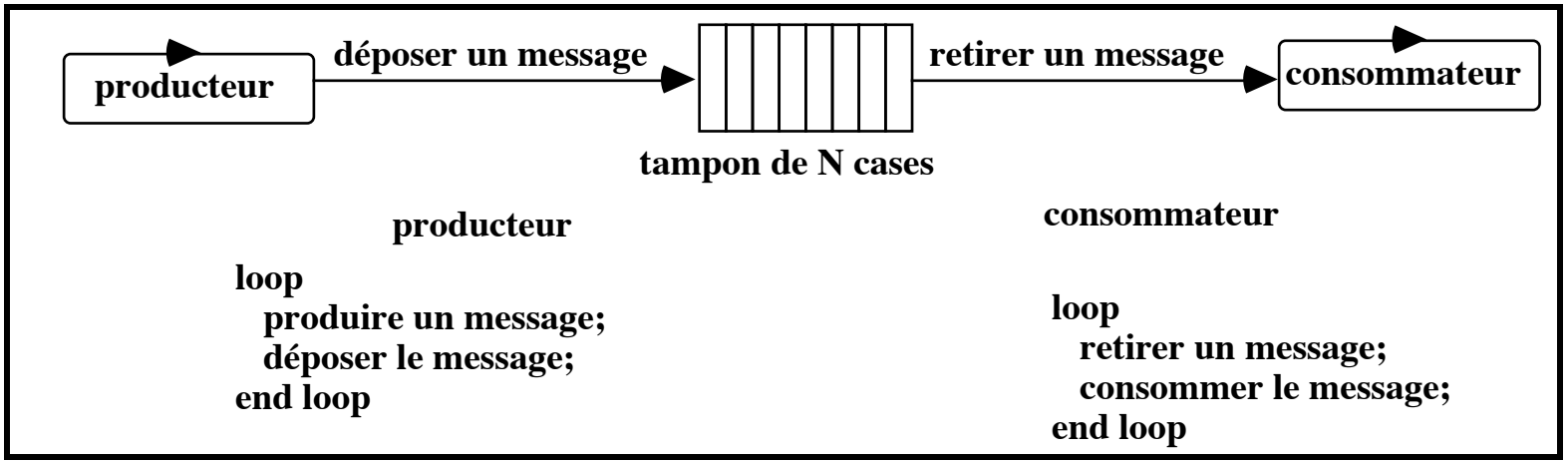
La coopération comprend des producteurs et des consommateurs qui se partagent un tampon ou une voie de communication, avec un nombre maximal, taille du tampon ou crédit de messages. Le contrôle de flux freine les producteurs s'ils vont trop vite pour les consommateurs.

LES PRODUCTEURS ET LES CONSOMMATEURS

HYPOTHÈSES GÉNÉRALES

H1: Les vitesses relatives des processus sont quelconques.
H2: Les priorités des processus sont quelconques.
H3: Tout processus met un temps fini pour déposer ou retirer un message; en particulier ni panne ni blocage perpétuel pendant ces opérations.

SPÉCIFICATION DE COMPORTEMENT



OBJECTIF

asservir la vitesse moyenne de production à la vitesse moyenne de consommation en ralentissant le moins possible le producteur

HYPOTHÈSES
DU PARADIGME DES PRODUCTEURS ET DES CONSOMMATEURS

1) vitesses relatives quelconques, débit irrégulier
2) tampon de taille fixe : N cases (une case = un message) et tampon vide initialement
3) tout message déposé est lu une fois et une seule
4) communication fiable

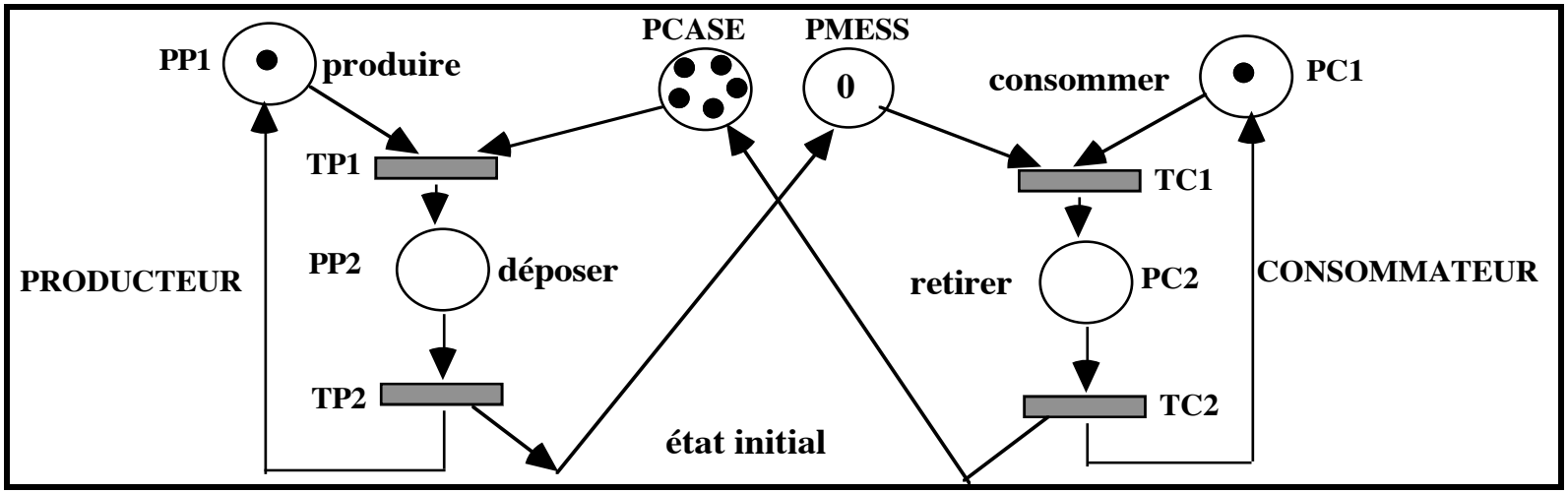
CONSÉQUENCES

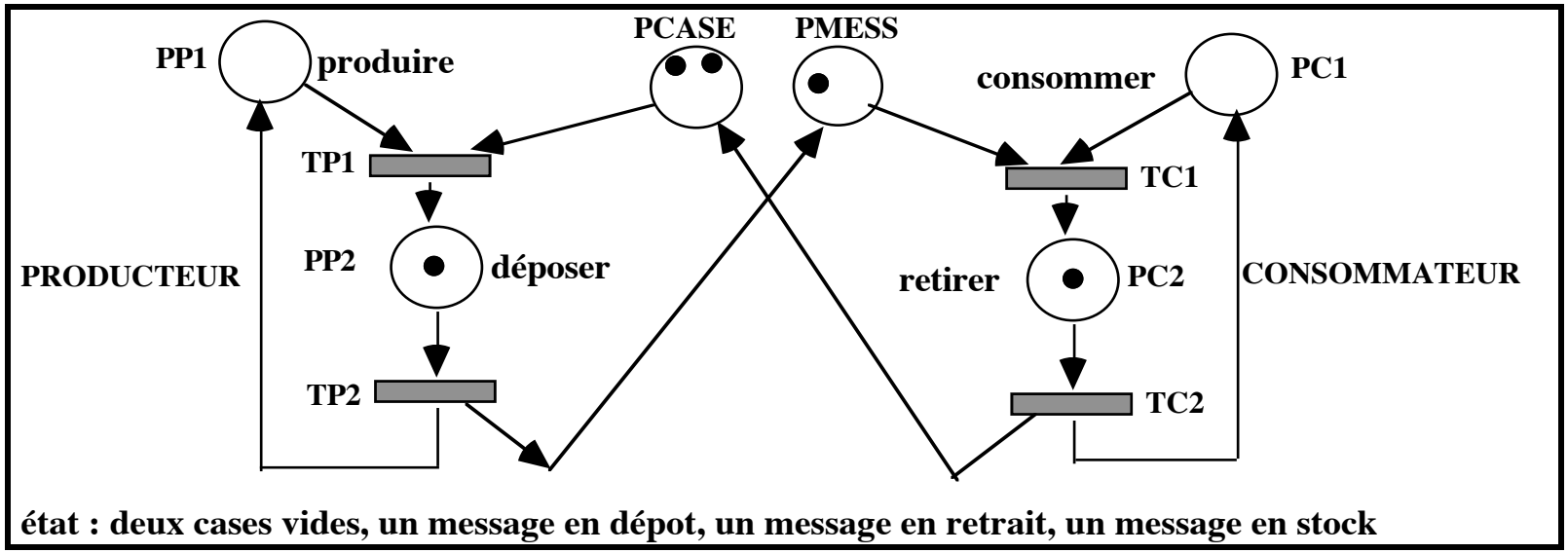
1) exclusion mutuelle d'accès aux messages (le message peut être lu ou écrit en plusieurs opérations
2) le producteur s'arrête s'il veut déposer alors que le tampon est plein. Il est réveillé quand cette condition n'est plus vraie.
3) le consommateur s'arrête s'il veut retirer alors que le tampon est vide. Il est réveillé quand cette condition n'est plus vraie.
4) pas d'interblocage

PARAMÈTRES D'IMPLANTATION

• nombre de producteurs et de consommateurs se partageant le tampon
• ordre de prélèvement des messages (premier déposé, dernier déposé, selon nature du message, ...)
structuration du tampon (tableau, liste chaînée, fichier,...)

RÉSEAU DE PETRI DU PARADIGME PRODUCTEUR-CONSOMMATEUR





COMPÉTITION D'ACCÈS : LES LECTEURS ET LES RÉDACTEURS

LA SITUATION CONCRETE

EXEMPLE1: une entreprise veut gérer en temps réel une base d'informations; certains processus modifient cette base ou y font des adjonctions; d'autres processus ne font que la lire, sans modifications. On souhaite avoir le maximum de concurrence pendant ces lectures.

EXEMPLE 2: dans tout système informatique, on maintient de nombreux catalogues pour répertorier les clients, les abonnés, les groupes d'usagers, les utilisateurs en session, les autorisations diverses, les processus, les ressources disponibles, les ressources en maintenance,....; des processus mettent ces catalogues à jour, en général à fréquence faible; d'autres processus les lisent, en général assez fréquemment; pour une meilleure efficacité, il est souhaitable de faire les lectures en parallèle.

EXEMPLE 3: tout système informatique comprend une horloge qui sert à dater l'arrivée et le départ d'un utilisateur, à mesurer son temps d'utilisation des ressources du système, à dater les versions successives d'un même fichier,...; cette horloge est incrémentée par un compteur à quartz, toutes les dix millisecondes par exemple; ce compteur active un processus qui fait progresser des indicateurs de année, mois, jour, heure, minute, seconde,1/100e de seconde; d'autres processus lisent ces valeurs; l'efficacité prône une lecture concurrente; il faut toutefois éviter de lire pendant la progression des indicateurs, en particulier quand l'heure incrémentée est 31 jours, 23 heures, 59 minutes, 59 secondes, 99/100e, car comment garantir qu'on ne lira pas de valeur intermediaire du calcul, comme 31jours, 23 heures, 0 minute, 0 seconde, 0/100e.

LA TERMINOLOGIE

La compétition d'accès met en présence des processus lecteurs et des processus rédacteurs qui se partagent des données composées (accès non atomique). La synchronisation appelée aussi contrôle de concurrence garantit la cohérence des données écrites ou lues par les processus concurrents.

HYPOTHÈSES

H1: Les vitesses relatives des processus sont quelconques.
H2: Les priorités ou les droits des processus sont quelconques.
H3: Tout processus termine son écriture et/ou sa lecture au bout d'un temps fini

.

LES LECTEURS ET LES RÉDACTEURS

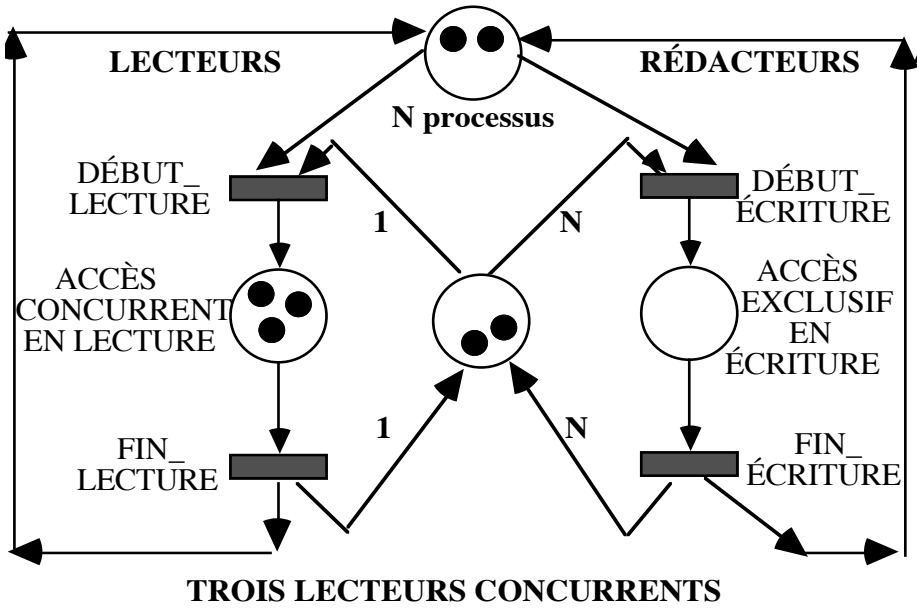
SPÉCIFICATION DE COMPORTEMENT

1. Les lectures sont concurrentes et cohérentes ; les écritures sont en exclusion mutuelle
2. Priorité aux lecteurs ou aux rédacteurs. Équité entre lecteurs et rédacteurs. Service à l'ancienneté.

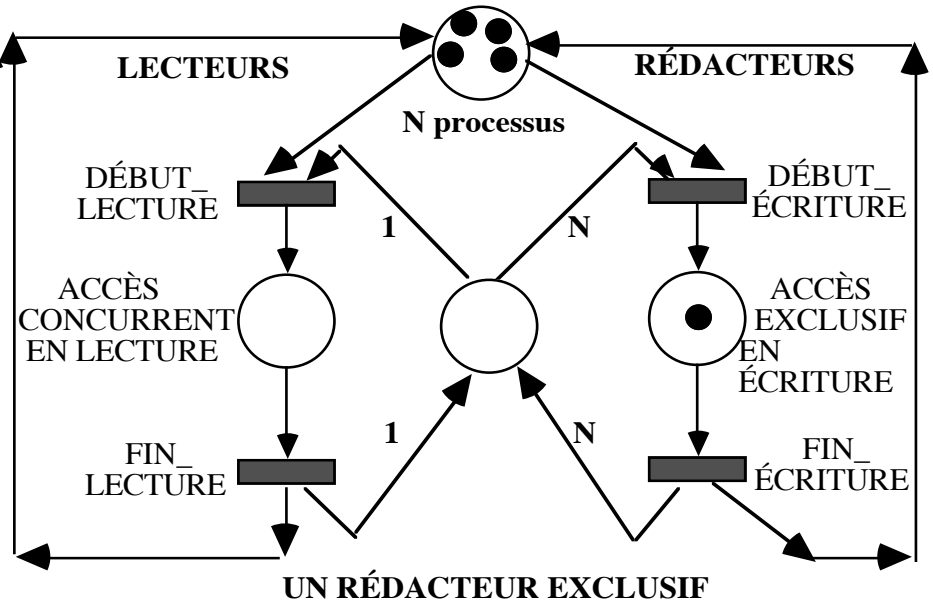
SCHÉMA D'EXECUTION

Protocole d'initialisation
procédures de contrôle : DÉBUT_ÉCRITURE; écriture en exclusion mutuelle; FIN_ÉCRITURE;
procédures de contrôle : DÉBUT_LECTURE; lectures concurrentes; FIN_LECTURE;

RESEAU DE PETRI POUR UN SYSTÈME AVEC CINQ PROCESSUS (N =5)



TROIS LECTEURS CONCURRENTS



UN RÉDACTEUR EXCLUSIF

LE REPAS DES PHILOSOPHES

LA SITUATION CONCRÊTE

Partage de plusieurs classes de ressources entre des processus concurrents asynchrones

Méthodes d'allocation très variables : globales, à la demande, une à une

Interblocage possible

Famine possible

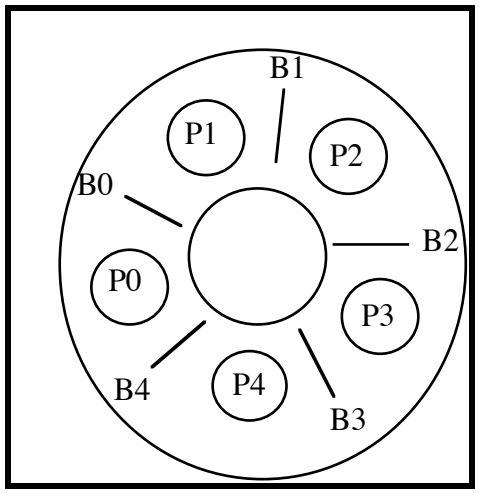
HYPOTHÈSES

H1: Les vitesses relatives des processus sont quelconques.

H2: Les priorités ou les droits des processus sont quelconques.

H3: Tout processus libère ses ressources au bout d'un temps fini.

LE REPAS DES PHILOSOPHES

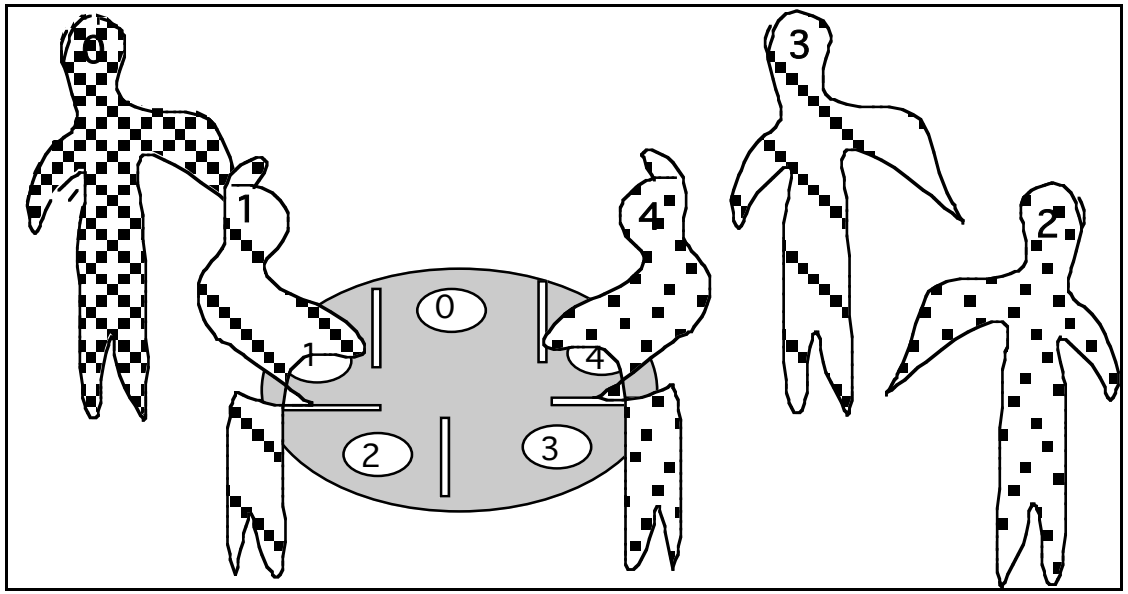


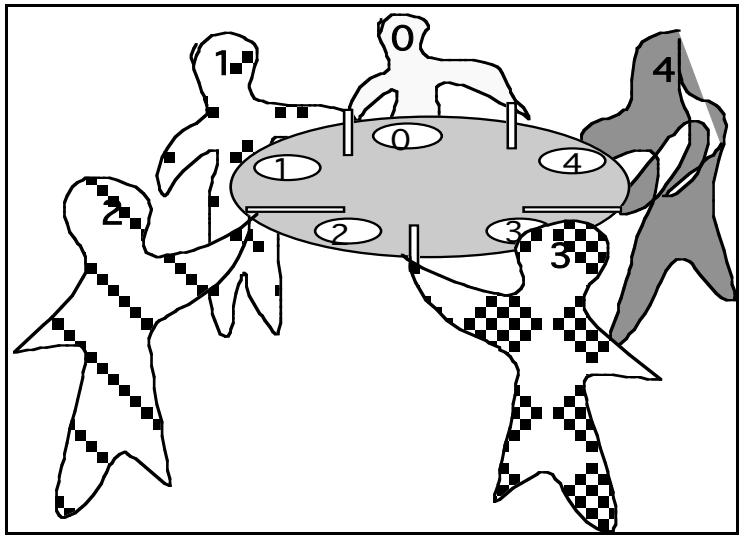
HYPOTHÈSES
1 Chaque assiette a une place fixe
2. Chaque baguette a une place fixe
3. accès à chaque baguette en exclusion mutuelle

PROTOCOLE
1. Pour manger, un philosophe doit utiliser deux baguettes, la sienne et celle de son voisin de droite
2. Tout philosophe ne mange que pendant un temps fini , puis restitue les deux baguettes

PROPRIÉTÉS
1 pas d'interblocage
2 pas de coalition (famine)

REPAS DES PHILOSOPHES





famine : ((1,3)->(1,4)->(2,4)->1,4))* **interblocage**

PROCESSUS ASYNCHRONES ET CONCURRENTS

L'ASYNCHRONISME APPORTE DE LA SOUPLESSE :

• modularité de découpage, extensibilité, localité, indépendance des actions, ...

MAIS LA CONCURRENCE PEUT CRÉER DU DÉSORDRE ET CELUI-CI PEUT ÊTRE ACCENTUÉ PAR L'ASYNCHRONISME

• désordre des accès à une mémoire commune partagée en accès direct
(mélanges incongrus, incohérence, calculs faux,...)
• désordre des messages transférés entre processus
(réception dans un ordre différent de celui de l'émission, mélange de parties de différents messages, messages non reçus, messages lus deux fois, ...)

SOLUTION AU NIVEAU DE L'ARCHITECTURE LOGIQUE

• utilisation des bons paradigmes d'accès, de coopération, de compétition
(ils remettent de l'ordre là où il est nécessaire et utile sans perdre la souplesse de l'asynchronisme)

PROPRIÉTÉS GLOBALES D'UNE APPLICATION CONCURRENTE

♦ sûreté ("safety") : garantir que rien de "mauvais" ne peut se produire

• états interdits pour des raisons sécuritaires, en fonctionnement normal ou en cas de panne

• respect de l'exclusion mutuelle, absence d'interblocage (interblocage actif ou passif),...

♦ vivacité ("liveness") : garantir que quelque chose de "bon" se produira

♦ éviter la famine

philosophes : si allocation globale des baguettes

exclusion mutuelle : si file d'attente avec priorité

producteurs consommateurs : si service en pile (LIFO) ou priorités

lecteurs rédacteurs : si priorité aux lecteurs (ou aux rédacteurs)

♦ respecter des priorités

importance (criticité) : critère issu des besoins applicatifs, utile en cas de surcharge

urgence : critère associé à des échéances temporelles

priorité (fixe ou variable): critère local à un processus utilisé comme moyen de gestion de ressource

aspects liés à la priorité fixe : • inversion de priorité, • héritage de priorité

ENCORE UN EXEMPLE D'UTILISATION DES PARADIGMES ET DES ARCHÉTYPES

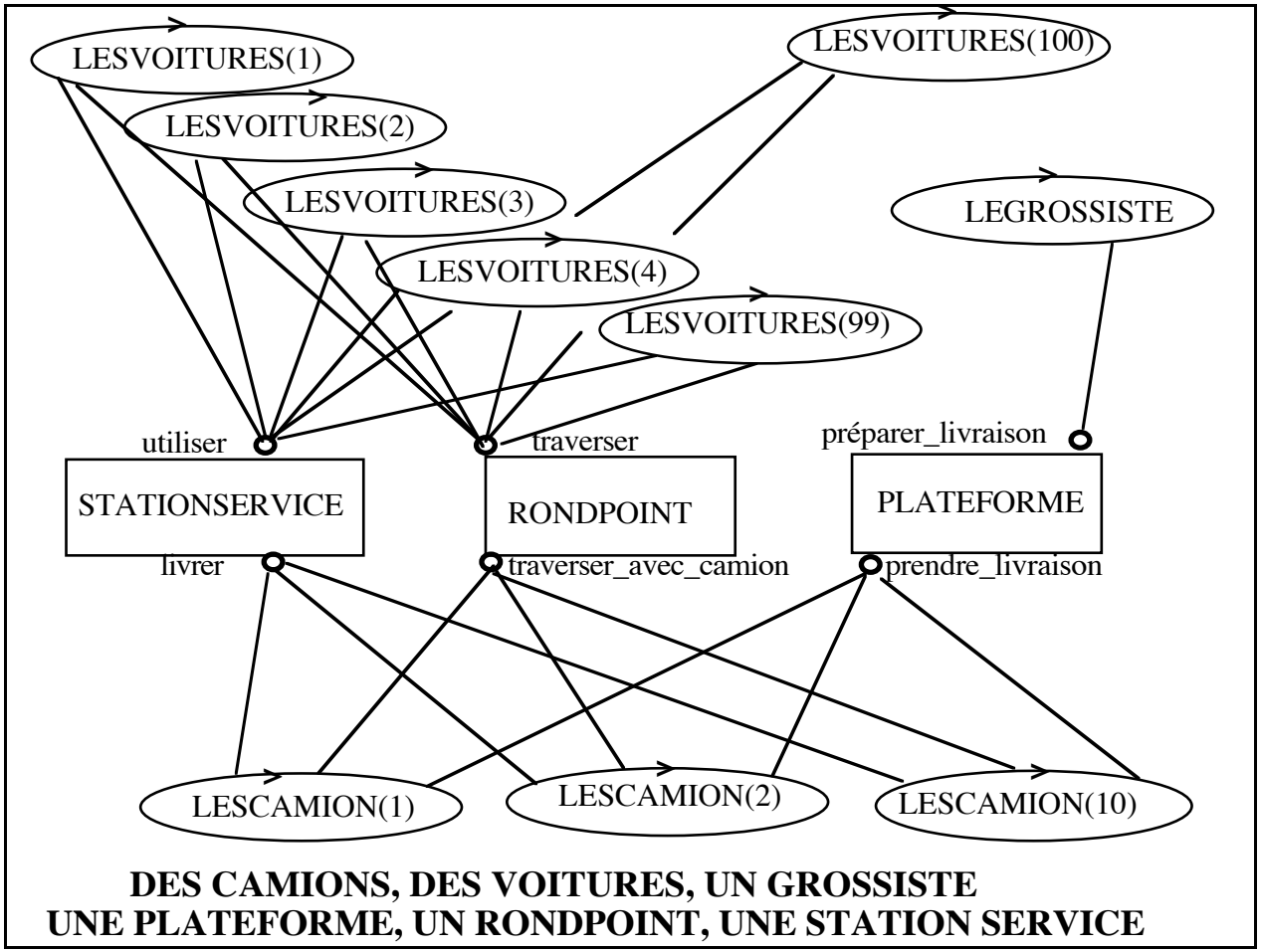


SCHÉMA AVEC PARADIGMES (exclusion mutuelle, producteurs-consommateurs)
COMPLÉMENT DU CODE AVEC OBJETS OU PAQUETAGES (ARCHÉTYPES, COMPOSANTS)

```
procedure MAIN_SIMULATION is                -- c'est le processus principal qui lance l'application
   NBVOITURE: constant INTEGER := 2500; NBCAMION: constant INTEGER :=10;
   task type VOITURE; LESVOITURES: array (1..NBVOITURE) of VOITURE;
   task type CAMION; LESCAMIONS: array (1..NBCAMION) of CAMION;
   task type GROSSISTE; LEGROSSISTE: GROSSISTE; -- déclaration des processus

   package PLATEFORME is                    -- spécification d'un objet partagé entre processus
      procedure PRENDRE_LIVRAISON;                                          -- PARADIGME
      procedure PREPARER_LIVRAISON;                                -- PRODUCTEURS-CONSOMMATEURS
   end PLATEFORME;

   package STATIONSERVICE is                --- spécification d'un objet partagé entre processus
      procedure UTILISER;                                                   -- PARADIGME
      procedure LIVRER;                                                     -- LECTEURS-RÉDACTEURS
   end STATIONSERVICE;

   package RONDPOINT is                     -- spécification d'un objet partagé entre processus
      procedure TRAVERSER;                                                  -- PARADIGME
      procedure TRAVERSER_AVEC_CAMION_PLEIN;          -- EXCLUSION MUTUELLE OU COHORTE
   end RONDPOINT;

   task body VOITURE is                     -- expression des activités du processus
   begin loop              -- boucle infinie car processus cyclique
         RONDPOINT.TRAVERSER; ROULER; RONDPOINT.TRAVERSER; ROULER; STATIONSERVICE.UTILISER;
   end loop; end VOITURE;

   task body CAMION is
   begin loop              -- boucle infinie car processus cyclique
         PLATEFORME.PRENDRE_LIVRAISON; ROULER; RONDPOINT.TRAVERSER_AVEC_CAMION_PLEIN ;
      ROULER; STATIONSERVICE.LIVRER; ROULER; RONDPOINT.TRAVERSER; ROULER;
   end loop; end CAMION;

   task body GROSSISTE is
   begin loop              -- boucle infinie car processus cyclique
         PLATEFORME.PREPARER_LIVRAISON; ENREGISTRER
   end loop; end GROSSISTE;

   package body PLATEFORME is separate;          -- annonce une compilation séparée
   package body STATIONSERVICE is separate; package body RONDPOINT is separate;

begin null; -- les tâches déclarées ci-dessus démarrent à ce point. Il y a (NBVOITURE +NBCAMION+2) processus
end MAIN_SIMULATION;        -- le seul rôle de ce programme principal est de créer les objets et les processus
```