29 - Entrées privées et instruction requeue

Programmation Concurrente - LI330
Université P. & M. Curie - année scolaire 2013/2014

PrC

Limites des objets protégés

- La clause de garde ne peut dépendre des paramètres passés lors de l'appel d'un point entrée
  - Raison : un coût prohibitif d'implémentation
- Palliatif
  - Accepter l'appel d'entrée puis en fonction des paramètres d'appel à remettre en attente la tâche appelante
  - L'instruction `requeue` permet de faire passer une tâche d'une file d'attente d'entrée à une autre file d'attente d'entrée de même signature
- Exemple, un système de réservation de ressources avec libération «non atomique»
  - On réserve un certain nombre de ressources
    - Bloquant si le nombre de ressources supérieur à ce qui est disponible
      - Le nombre de ressources demandé est en paramètrel
    - On a besoin d'une condition au sens des moniteurs
      - Pour suspendre l'exécution en cas de besoin...

Exemple : le parking

- Réservation de places «par lots»
- Libération de places «par lots» (potentiellement inférieurs aux lots réservés)
- Réservation bloquante lorsqu'elle ne peut être fournie
- On se rappelle des réservations bloquées
  - Traitement prioritaire lorsque des places sont relâchées...

Exemple (1)

```
with Ada.Text_Io; use Ada.Text_Io;

procedure Prott_Ex5 is
   protected type Parking_Lots (Length : Natural) is
      -- Réserve Length places et bloque tant qu'elles ne sont pas disponibles
      entry Get_Lots (
            Id, Nb : Positive );
      -- Rendre des places de parking
      procedure Release_Lots (
            Id, Nb : Positive );
   private
      -- entree privee (utilisée pour  stocker des requetes non satisfiables)
      entry Wait (
            Id, Nb : Positive );
      -- le nombre de ressources
      Avail_Lots : Natural := Length;
      -- Une condition au sens des moniteurs bloque les nouvelles demandes
      -- pour traiter celles qui sont en attente
      Updating : Boolean := False;
   end Parking_Lots;
```

../..

Exemple (2)

```
protected body Parking_Lots is
   entry Get_Lots (
         Id,
         Nb : Positive ) when not Updating is
   begin
      Put_Line ("Get_lots pour" & Integer'Image (Id) & " de" &
                Integer'Image (Nb) & " places");
      if Nb <= Avail_Lots then
         -- On peut donner...
         Avail_Lots := Avail_Lots - Nb;
      else
         -- equivalent d'un condition.attendre (moniteur)
         Put_Line ("        Get_lots, requeue de" &
                   Integer'Image (Id));
         requeue Wait;
      end if;
   end Get_Lots;
```

../..

Exemple (3)


```
procedure Release_Lots (
      Id,
      Nb : Positive ) is
begin
   Put_Line ("       Release_lots pour" & Integer'Image (Id) &
             " avec" & Integer'Image (Nb));
   -- Rendre les ressources
   Avail_Lots := Avail_Lots + Nb;
   -- Equivalent d'un condition.vide
   if Wait'Count > 0 then
      -- Inhibe Get_lots et déclenche wait (equivalent d'un condition.signaler)
      Put_Line ("       Release_lots, reprise");
      Updating := True;
   end if;
end Release_Lots;
```

../..

Exemple (4)

```
   entry Wait (
         Id,
         Nb : Positive ) when Updating is
   begin
      Put_Line ("                Wait pour" & Integer'Image (Id) &
                " avec" & Integer'Image (Nb));
      -- Remet la requête suspendue dans
      -- la file des requêtes à traiter...
      if Wait'Count = 0 then
         Updating := False;
      end if;
      requeue Get_Lots;
   end Wait;
end Parking_Lots;

R : Parking_Lots (9);

task type Users (Id : Positive);
```

../..

Exemple (5)


```
task body Users is
begin
   Put_Line (Integer'Image (Id) & " reserve" & Integer'Image
             (3 + Id) & " places");
   R.Get_Lots (Id, 3 + Id);
   -- je gare mes vehicules
   delay 0.1 ; -- ...
   -- Trois vehicules partent...
   Put_Line (Integer'Image (Id) & " rend 3 places");
   R.Release_Lots (Id, 3);
   -- J'ai terminé avec le reste des places de parking...
   Put_Line (Integer'Image (Id) & " rend" & Integer'Image (Id) &
             " places");
   R.Release_Lots (Id, Id);
end Users;
```

Exemple (6)


```
   U4 : Users (4);
   U1 : Users (1);
   U3 : Users (3);
   U2 : Users (2);

begin -- Prott_Ex5
   null;
end Prott_Ex5;
```

Exemple 5 (6)


```
$ ./protT_ex5
 4 reserve 7 places
 3 reserve 6 places
Get_lots pour 4 de 7 places
 1 reserve 4 places
Get_lots pour 3 de 6 places
       Get_lots, requeue de 3
Get_lots pour 1 de 4 places
       Get_lots, requeue de 1
 2 reserve 5 places
Get_lots pour 2 de 5 places
       Get_lots, requeue de 2
 4 rend 3 places
       Release_lots pour 4 avec 3
       Release_lots, reprise
              Wait pour 3 avec 6
              Wait pour 1 avec 4
              Wait pour 2 avec 5
Get_lots pour 3 de 6 places
       Get_lots, requeue de 3
Get_lots pour 1 de 4 places
Get_lots pour 2 de 5 places
       Get_lots, requeue de 2
 4 rend 4 places
       Release_lots pour 4 avec 4
       Release_lots, reprise
              Wait pour 3 avec 6
              Wait pour 2 avec 5
Get_lots pour 3 de 6 places
       Get_lots, requeue de 3
Get_lots pour 2 de 5 places
 1 rend 3 places
       Release_lots pour 1 avec 3
       Release_lots, reprise
              Wait pour 3 avec 6
Get_lots pour 3 de 6 places
       Get_lots, requeue de 3
 1 rend 1 places
       Release_lots pour 1 avec 1
       Release_lots, reprise
              Wait pour 3 avec 6
Get_lots pour 3 de 6 places
       Get_lots, requeue de 3
 2 rend 3 places
       Release_lots pour 2 avec 3
       Release_lots, reprise
              Wait pour 3 avec 6
Get_lots pour 3 de 6 places
 2 rend 2 places
       Release_lots pour 2 avec 2
 3 rend 3 places
       Release_lots pour 3 avec 3
 3 rend 3 places
       Release_lots pour 3 avec 3
```

pas assez de ressources

Tentative de reprise (passe pour 1)

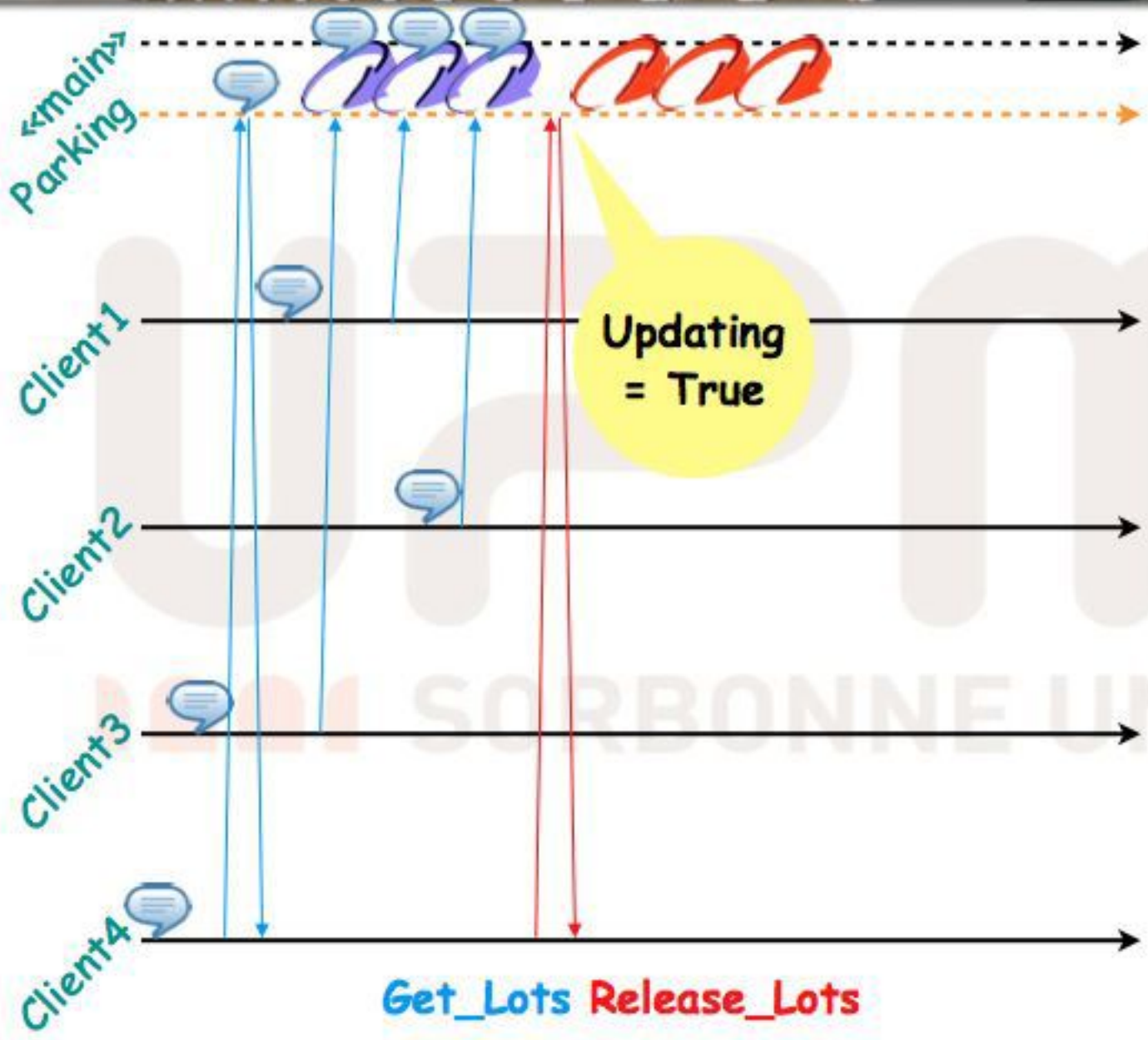
Tentative de reprise (passe pour 2)

Tentative de reprise (sans succès)

Tentative de reprise (san succès)

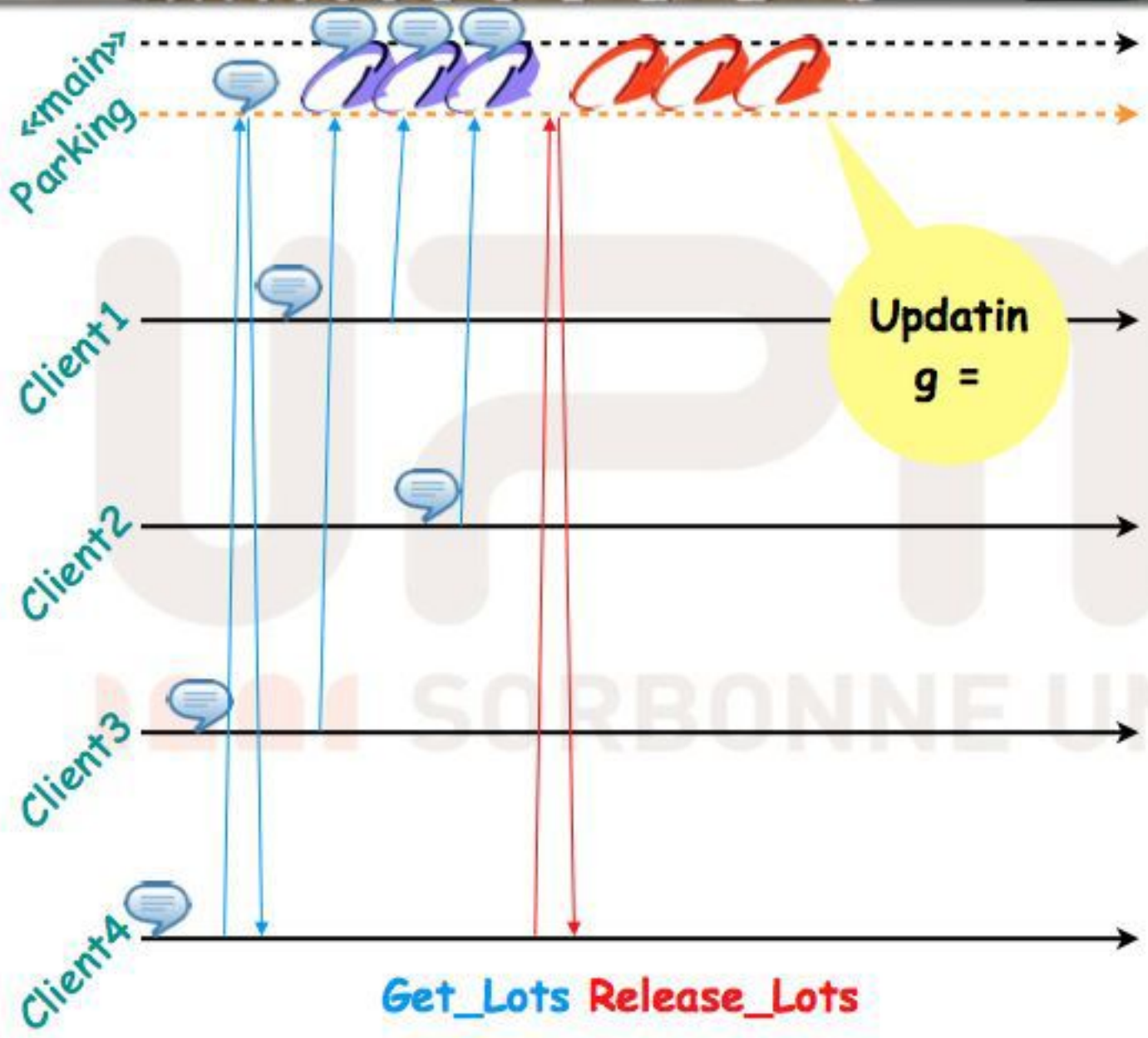
Tentative de reprise (passe pour 3)

Chronogramme partiel (début)



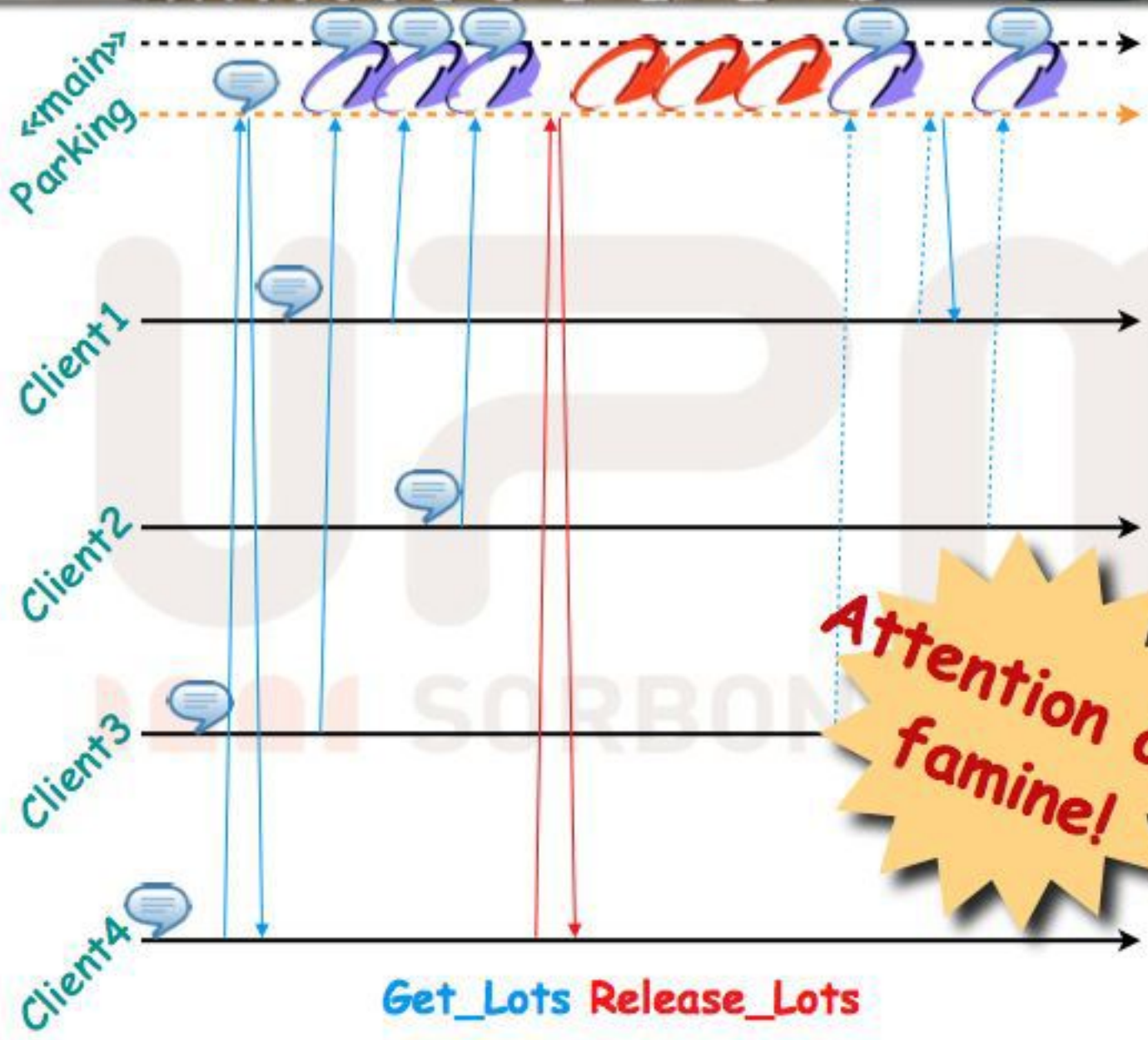
4 reserve 7 places
3 reserve 6 places
Get_lots pour 4 de 7 places
1 reserve 4 places
Get_lots pour 3 de 6 places
Get_lots, requeue de 3
Get_lots pour 1 de 4 places
Get_lots, requeue de 1
2 reserve 5 places
Get_lots pour 2 de 5 places
Get_lots, requeue de 2
4 rend 3 places
Release_lots pour 4 avec 3
Release_lots, reprise
Wait pour 3 avec 6
Wait pour 1 avec 4
Wait pour 2 avec 5
Get_lots pour 3 de 6 places
Get_lots, requeue de 3
Get_lots pour 1 de 4 places
Get_lots pour 2 de 5 places
...

Chronogramme partiel (début)



4 reserve 7 places
3 reserve 6 places
Get_lots pour 4 de 7 places
1 reserve 4 places
Get_lots pour 3 de 6 places
Get_lots, requeue de 3
Get_lots pour 1 de 4 places
Get_lots, requeue de 1
2 reserve 5 places
Get_lots pour 2 de 5 places
Get_lots, requeue de 2
4 rend 3 places
Release_lots pour 4 avec 3
Release_lots, reprise
Wait pour 3 avec 6
Wait pour 1 avec 4
Wait pour 2 avec 5
Get_lots pour 3 de 6 places
Get_lots, requeue de 3
Get_lots pour 1 de 4 places
Get_lots pour 2 de 5 places
...

Chronogramme partiel (début)



4 reserve 7 places
3 reserve 6 places
Get_lots pour 4 de 7 places
1 reserve 4 places
Get_lots pour 3 de 6 places
Get_lots, requeue de 3
Get_lots pour 1 de 4 places
Get_lots, requeue de 1
2 reserve 5 places
Get_lots pour 2 de 5 places
Get_lots, requeue de 2
4 rend 3 places
Release_lots pour 4 avec 3
Release_lots, reprise
Wait pour 3 avec 6
Wait pour 1 avec 4
Wait pour 2 avec 5
Get_lots pour 3 de 6 places
Get_lots, requeue de 3
Get_lots pour 1 de 4 places
Get_lots pour 2 de 5 places
...

Conclusion sur le parallélisme en Ada

- Le modèle de parallélisme d'Ada est extrêmement riche
  - Java, il faut «assembler» des éléments de base (ou bibliothèques)
  - C and Co, il faut faire appel au système d'exploitation
- Il existe encore un niveau de parallélisme : la répartition
  - Annexe des systèmes répartis (Ada/DSA)
  - Plus proche d'un intergiciel (middleware)
    - Équivalent de Java/RMI
    - Équivalent de CORBA
    - Objectif: cacher les appel système (y compris la gestion du réseau)
  - On peut combiner...
    - Exécution répartie + processus légers
    - La programmation devient délicate;-)