❑ **Ausführungsschema**

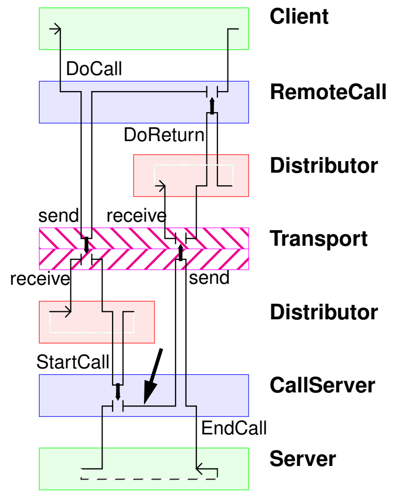
◆ **seriell**

- **Einfach zu realisieren durch sogenannten 'Server'-Prozeß**
- **Kann in nebenläufigen Systemen Verklemmungen einführen!**

◆ **nebenläufig**

- **Bei jedem Fernaufruf neuer Prozeß auf der 'Server'-Seite**
- **Fester Vorrat an 'Server'-Prozessen zur Verringerung des dynamischen Aufwands**
- **Variabler Vorrat an 'Server'-Prozessen mit unterer und oberer Schwelle**

❑ **Algorithmus von Lampson**



❑ **Koordinierung durch Monitore**

◆ **Konzept A (Hoare)**

- **Datenstruktur mit Prozeduren (Funktionen)**
  - **Daten des Monitors können nur durch die Prozeduren (Funktionen) des Monitors manipuliert werden**
- **Prozeduren (Funktionen) laufen unter gegenseitigem Ausschluß**
- **zusätzlicher Datentyp condition**
  - **repräsentiert Warteschlange von blockierten Prozessen**
  - **zugehörige Operationen**

wait	zum Blockieren und Einreihen eines Prozesses in die Warteschlange
signal	zum Deblockieren und Entfernen eines Prozesses aus der Warteschlange

Problem: Geschachtelte Monitore

◆ **Konzept B (Java)**

- **Jedes Objekt besitzt**
  - **einen Belegungszustand (frei, belegt)**
  - **eine Belegt-Warteschlange**
  - **eine Koordinierungswarteschlange**
- **Frei wählbare Prozedurabschnitte (Funktionen) laufen unter gegenseitigem Ausschluß**
- **Belegt-Warteschlange**
  - **Einreihung**
    - bei Betreten eines koordinierten Abschnitts, falls Objekt schon belegt, sonst lediglich belegen des Objektes
  - **Ausreihung**
    - bei Verlassen eines koordinierten Abschnitts; war die Belegt-Warteschlange leer wird das Objekt frei
    - durch Aufruf von wait(); war die Belegt-Warteschlange leer wird das Objekt frei

Fernaufruf: Ideale Eigenschaften eines transparenten Mechanismus

- **Koordinierungs-Warteschlange**
  - **Einreihung**
    - durch Aufruf von wait();
  - **Ausreihung**
    - eines Fadens durch die Operation notify() angewandt auf das koordinierte Objekt. War das Objekt belegt, wird der Faden in die Belegt-Warteschlange eingereiht. Ansonsten belegt er das Objekt.
    - aller Wartenden durch die Operation notifyAll() angewandt auf das koordinierte Objekt.
      War das Objekt belegt, werden diese Fäden in die Belegt-Warteschlange eingereiht.
      Ansonsten belegt einer das Objekt, die anderen werden in die Belegt-Warteschlange eingereiht
- **Lösung des Problems geschachtelter Monitore**
  - **vorzeitige Belegung**

Fernaufruf: Ideale Eigenschaften eines transparenten Mechanismus

```
class RPC {
   public static final int CALL = 1;
   public static final int RETURN = 2;
   ...
}

class ID {
   long value;
   ...
}

class Message {
   int state;                    // state is CALL or RETURN
   Processor source = null, dest = null;
   ID id, request;
   Object obj;                   // Object whose method should be called
   Method action;                // Method to be called
   Object[] val = new Object[1]; // Argument
   ...
}
```


Fernaufruf: Ideale Eigenschaften eines transparenten Mechanismus

```
class Distributor extends Process {
   HashSet connections = new HashSet();
   // The one Process which receives and distributes messages
   public void run() {
      ResultOfReceive r;
      while (true) {
         r = receive();
         if (r.state) {
            if ((r.message.state == RPC.CALL)
                 && okToAccept(r.message)) {
                myCallServer.startCall(r.message);
         } else {
            if (r.message.state == RPC.RETURN) {
               myRemoteCall.doReturn(r.message);
            }
         }
      }
   }
```


Fernaufruf: Ideale Eigenschaften eines transparenten Mechanismus

```
// Make calls
class RemoteCall {
  class CallOut {
    Message m = new Message(); Condition received;
  }
  HashSet callsOut = new HashSet();
```