3 Fernaufruf

❑ Literatur

Nelson, B. J.: Remote Procedure Call. Tech. Rep. CSL-81-9, Xerox Palo Alto Research Center, Palo Alto, Calif., 1981.

Kaiserswerth, M.: Der Fernaufruf als Betriebssystemdienst. Arbeitsberichte des IMMD, Band 22, Nummer 1, Januar 1989.

SUN: Network Programming Guide. 1990.

❑ Untergliederung

3.1 Der Begriff, Anwendungsbeispiele

❑ Fernaufruf ist der synchrone Transfer von Kontrolle und Daten zwischen Teilen eines über disjunkte Adreßräume verteilten Programms.

❑ Um in einer prozeduralen Programmiersprache den gleichförmigen Zugriff auf lokale und nicht-lokale Ressourcen zu ermöglichen, sollte sich der Fernaufruf auf der semantischen (und syntaktischen) Ebene genauso wie ein lokaler Prozeduraufruf verhalten.

❑ Koroutinen, Ausnahmen (exceptions):

- Sind ebenfalls Arten eines synchronen Kontroll- und Datentransfers

❑ Synchroner Transfer und Nebenläufigkeit:

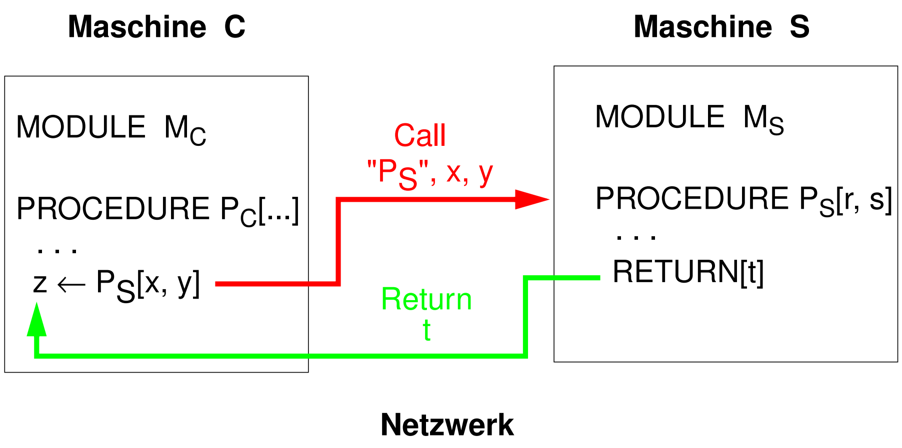
- Unabhängige Konzepte

Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

❑ Anwendungsbeispiele

- Nelson (Fig. 2.1): Netzwerk unsichtbar für Client und Server; bei der Programmierung ohne Bedeutung; übliche Programmentwicklungsumgebung benutzbar; Test kann auf Monoprozessor erfolgen
- File Server: Für Client irrelevant, welche Teile remote sind

❑ Ideale Vorstellung

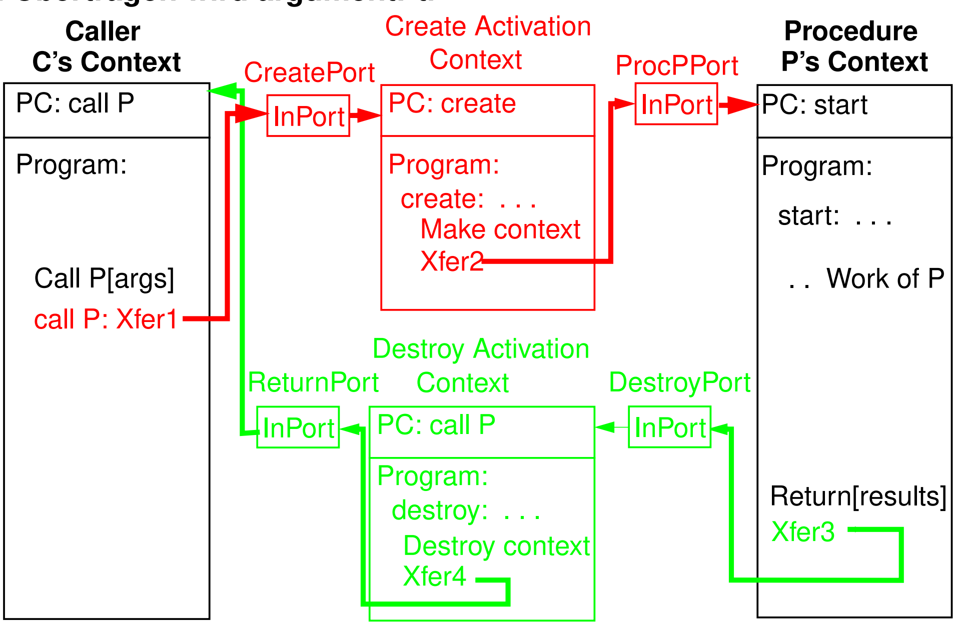


Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann

❑ Abstrakte Grundmechanismen (Lampson e.a.)

◆ Abstrakte Maschinenoperation ’Transfer’

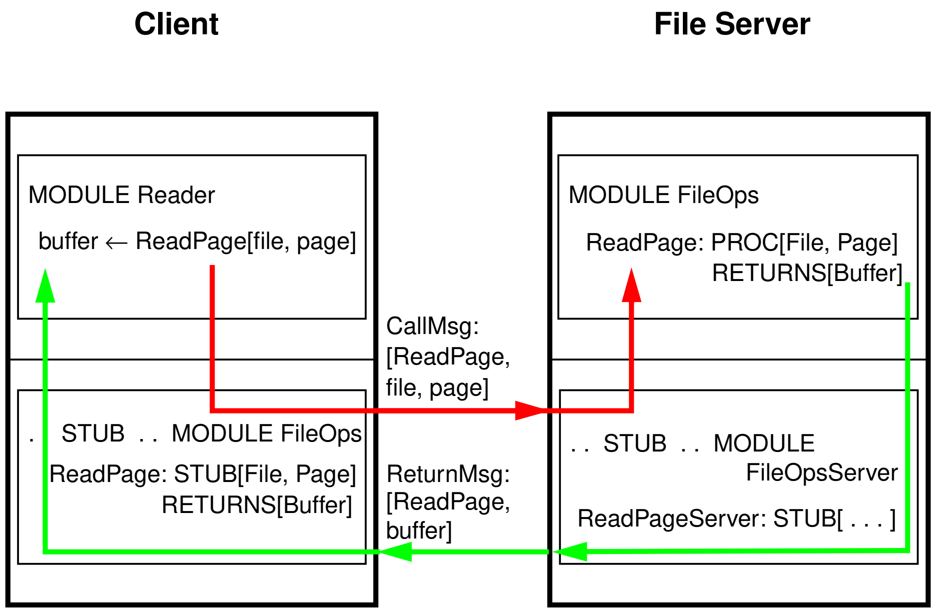
- Transfer[destinationInport, returnOutport, argumentPtr]
- RTransfer: Übertragen wird argumentPtr↑



Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann

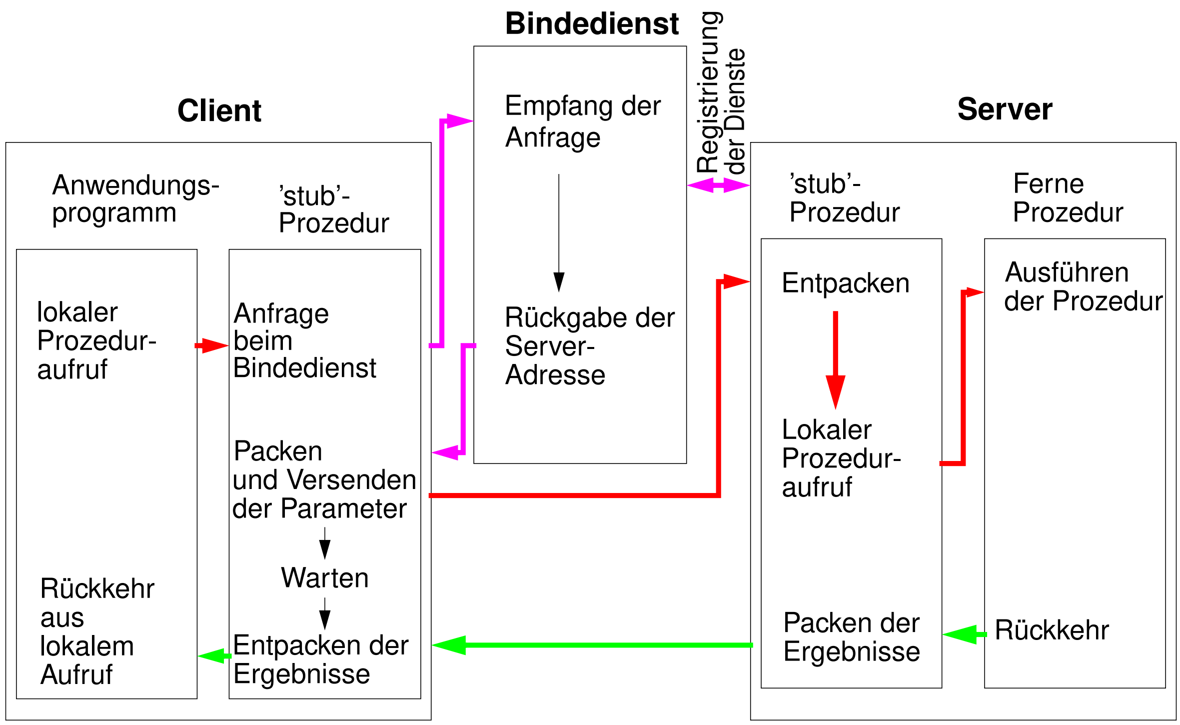
Grundmechanismen in 'Transfer'-losen Sprachen

- **Stubs; Aufrufender und Aufgerufener bedürfen keiner Änderung**



Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann

- **Gesamtstruktur bei 'Transfer'-losen Sprachen**



Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann

❏ Modellierung des physikalischen Kommunikationssystems einschließlich seines tolerierbaren Fehlverhaltens, Spezifikation Send-Receive

◆ Modellierung des Kommunikationssystems, Zustand M

- Nachrichtenverspätung

  M Menge

- Nachrichtenverlust

  {}

  send(m)

  $\{M = {}'M \vee M = {}'M \cup \{m\}\}$

- Nachrichtenverdopplung, Nachrichtenverfälschung

  {}

  [status, m] ← receive()

  $\{$ status = good $\wedge (M = {}'M \vee (M = {}'M - [m] \wedge m \in {}'M)))$

  $\vee$ status = bad $\wedge (M = {}'M \vee \exists\, m_{bad} \in {}'M\ (M = {}'M - \{m_{bad}\})))$

  $\}$

◆ Zu berücksichtigende Ausfälle

- Einzelrechner: Prozessorausfall
  Prozeßzusammenbruch
- Verteiltes System: Zusätzlich Verbindungsausfall

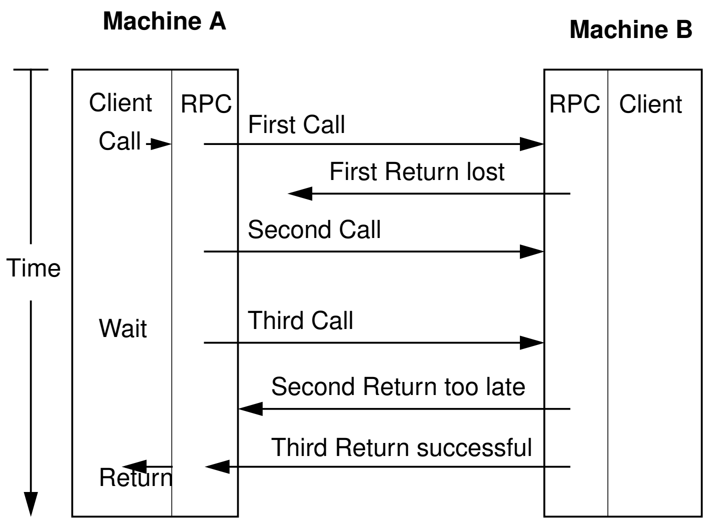
3.2 Überblick über die wesentlichsten Gesichtspunkte

- Aufrufsemantik
  - Genau-einmal (exactly-once)
  - Letzte-Ausführung (last-one)
- Nebenläufigkeit, Ausnahmebehandlung
  - Ausnahmen wie üblich behandeln
  - Ergänzung um netzwerkspezifische Ausnahmeursachen
- Gesichtspunkte der Transparenz auf Sprachebene
  - Binden und Konfigurieren
    - gesonderte Konfigurierungsbeschreibung
    - Schnittstellenbeschreibung
  - Typprüfung
    - heterogene Systeme
  - Parameterfunktionalität
    - Marshaling

3.3 Ideale Eigenschaften eines transparenten Mechanismus

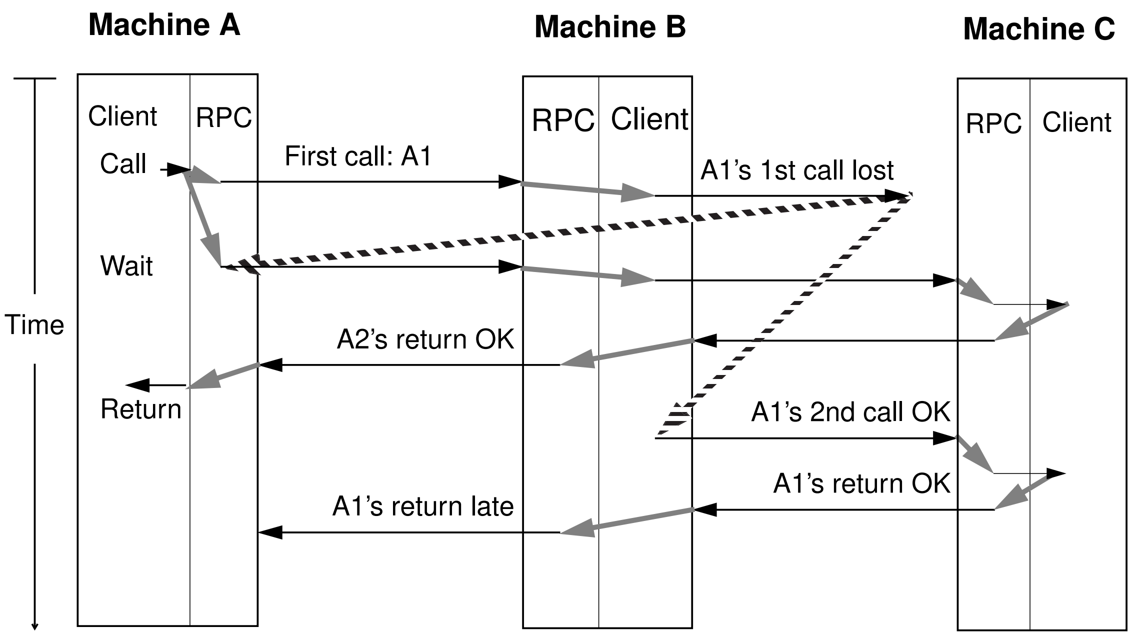
- Aufrufsemantik
  - exactly-once
  - at-least-once
  - last-of-many

    Verwendung von Sequenznummern



Fernaufruf: Ideale Eigenschaften eines transparenten Mechanismus

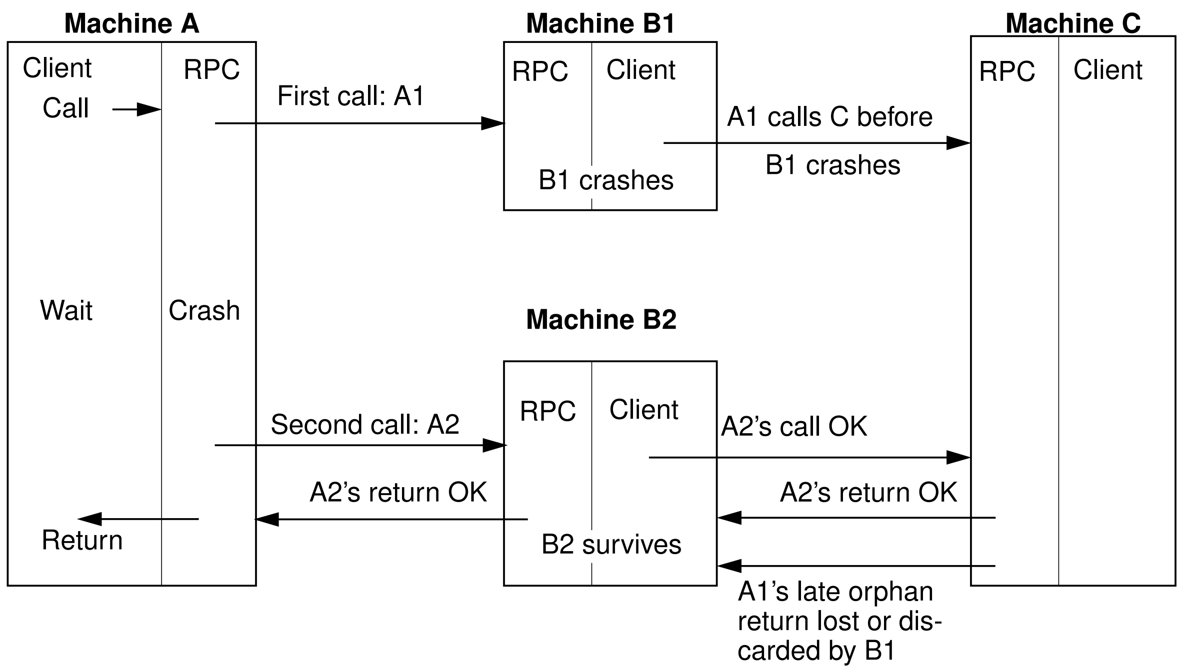
Diese einfache Lösung ist nicht transitiv



Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann

Fernaufruf: Ideale Eigenschaften eines transparenten Mechanismus

- **Numerierung der Aufrufe löst Problem, wenn keine Zusammenbrüche auftreten.**
- **Gegenbeispiel im anderen Fall:**



Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Fernaufruf: Ideale Eigenschaften eines transparenten Mechanismus

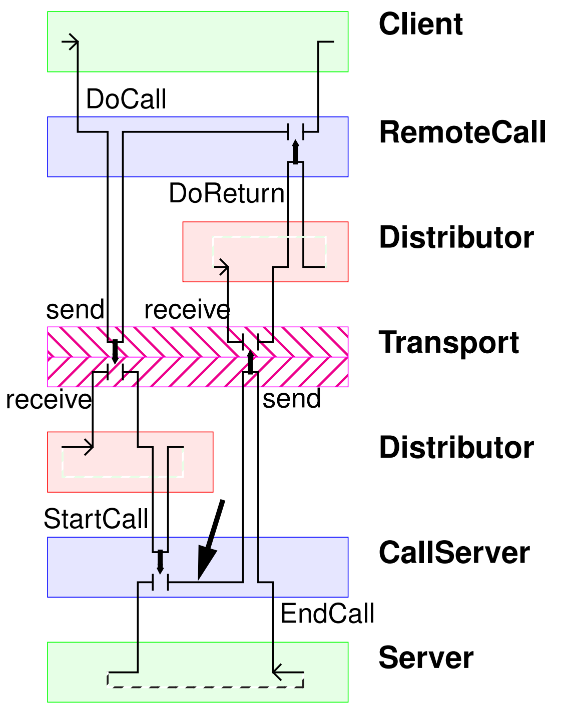
❑ Ausführungsschema

- **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

Fernaufruf: Ideale Eigenschaften eines transparenten Mechanismus

❑ Algorithmus von Lampson



Fernaufruf: Ideale Eigenschaften eines transparenten Mechanismus

❑ **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

Fernaufruf: Ideale Eigenschaften eines transparenten Mechanismus

◆ **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);
            }
         }
      }
   }
```


Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann

Fernaufruf: Ideale Eigenschaften eines transparenten Mechanismus

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


Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Fernaufruf: Ideale Eigenschaften eines transparenten Mechanismus

```
public Object[] doCall(Processor d, Object o, Method a, Object[] args) {
   CallOut c = new CallOut();
   c.m.source = myMachine(); c.m.request = myMachine().uniqueID();
   c.m.dest = d; c.m.action = a; c.m.obj = o; c.m.val = args;
   c.m.state = RPC.CALL; c.received = new Condition();
   synchronized (this) {
      callsOut.add(c);
      c.m.id = myMachine().uniqueID(); c.m.dest.send(c.m);
   }
   do {
      c.received.synchronizeTimed();
      synchronized (this) {
         if (c.m.state == RPC.RETURN) break;
         c.m.id = myMachine().uniqueID(); c.m.dest.send(c.m);
      }
   } while (true);
   synchronized (this) {
      val = c.m.val; return val;
   }
}
```


Fernaufruf: Ideale Eigenschaften eines transparenten Mechanismus

```
   synchronized public void doReturn(Message m) {
      CallOut c;
      Iterator i = callsOut.iterator();
      while (i.hasNext()) {
         c = (CallOut)i.next();
         if (c.m.id == m.id) {
            c.m = m;
            callsOut.remove(c);
            synchronized (c.received) {
               c.received.notify();
            }
            return;
         }
      }
   }
}
```


Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Fernaufruf: Ideale Eigenschaften eines transparenten Mechanismus

```
class CallIn {
  Message m = new Message();
  Condition work = new Condition();
}
```


Fernaufruf: Ideale Eigenschaften eines transparenten Mechanismus

```
// Serialize calls for each process, and assign work to worker processes
class CallServer {
   HashSet callsIn = new HashSet(), pool = new HashSet(initNumberOfWorkers);
   public void startCall(Message m) {
      CallIn w, c;
      synchronized (this) {
         Iterator i; i = callsIn.iterator();
         while (i.hasNext()) {
            c = (CallIn)i.next();
            if (c.m.request == m.request) { c.m.id = m.id; return; }
         }
         i = pool.iterator();
         // wait if the pool is empty
         while (!i.hasNext()) {
            try { wait(); } catch (InterruptedException e) { }
         }
         w = (CallIn)i.next(); pool.remove(w); callsIn.add(w); w.m = m;
      }
      w.work.signal();
   }
```

Fernaufruf: Ideale Eigenschaften eines transparenten Mechanismus

```
      public void endCall(CallIn w) {
         synchronized (this) {
            if (w.m.source != null) {
               w.m.source.send(w.m);
               callsIn.remove(w);
            }
            pool.add(w);
         }
         w.work.synchronize();
      }
   }
```


Fernaufruf: Ideale Eigenschaften eines transparenten Mechanismus

```
   class Worker extends Process {
      CallServer callServer = ...;

      public void run() {
         CallIn c = new CallIn();
         c.m.source = null;
         callServer.endCall(c);
         while (true) {
            try {
               c.m.val = new Object[1];
               c.m.val[0] = c.m.action.invoke(c.m.obj, null);
            } catch (IllegalAccessException e) {}
              catch (InvocationTargetException e) {}
            c.m.state = RPC.RETURN;
            callServer.endCall(c);
         }
      }
   }
```


Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Fernaufruf: Ideale Eigenschaften eines transparenten Mechanismus

```
class Connection {
    Processor from;
    ID lastID;
}

// The following method is part of the class Distributor

    public boolean okToAccept(Message m) {
        Connection c; Iterator i;
        i = connections.iterator();
        while (i.hasNext()) {
            c = (Connection)i.next();
            if (c.from.myName.equals(m.source.myName)) {
                if (c.lastID != null) {
                    if (m.id.value <= c.lastID.value) return false;
                    c.lastID = m.id; return true;
                } else return false;
            }
        }
        // No record of this processor. Establish connection.
```


Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann

Fernaufruf: Ideale Eigenschaften eines transparenten Mechanismus

```
// okToAccept continued

        // No record of this processor. Establish connection.
        if (m.action.getName().equals("uniqueID")) {
            return true;
            // Avoid an infinite loop; OK to duplicate this call.
        }
        c = new Connection();
        c.from = m.source;
        connections.add(c);
        Connector connector = new Connector(myMachine, "connector", c);
        connector.start();
        return false;
    }
}
```


Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Fernaufruf: Ideale Eigenschaften eines transparenten Mechanismus

```
class Connector extends Process {
    Connection connection = ...;
    Processor myMachine = ...;
    public void run() {
        Class cl = Processor.class;
        try {
            connection.lastID
                = (ID)(((myMachine.remoteCall
                        .doCall(connection.from,
                                connection.from,
                                cl.getMethod("uniqueID", null), null))[0])
        } catch (NoSuchMethodException e) {...}
    }
}
```


Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann

Fernaufruf: Ideale Eigenschaften eines transparenten Mechanismus

Übertragung / Prozessoren	keine Fehler	keine Fehler erhöhte Verzögerung	Paketverlust	Paketverlust Serverausfall	Paketverlust Clientausfall	Paketverlust Client- und Serverausfall
Maybe (No Recovery)	OP = 1 Res = 1	OP = 1 RES ≤ 1	OP ≤ 1 RES = 0	OP ≤ 1 RES = 0	OP ≤ 1 RES = 0	OP ≤ 1 RES = 0
at least once (Regular Server)	OP = 1 Res = 1	OP ≥ 1 RES ≥ 1	OP ≥ 1 RES ≥ 1	OP ≥ 0 RES ≥ 0	OP ≥ 0 RES = 0	OP ≥ 0 RES = 0
at least once (Recov. Server)	OP = 1 Res = 1	OP ≥ 1 RES ≥ 1	OP ≥ 1 RES ≥ 1	OP ≥ 1 RES ≥ 1	OP ≥ 0 RES = 0	OP ≥ 0 RES = 0
last of many (Regular Server)	OP = 1 Res = 1	OP ≥ 1 RES ≤ 1	OP ≥ 1 Res = 1	OP ≥ 0 RES ≤ 1	OP ≥ 0 RES = 0	OP ≥ 0 RES = 0
last of many (Recov. Server)	OP = 1 Res = 1	OP ≥ 1 RES ≥ 1	OP ≥ 1 Res = 1	OP ≥ 1 Res = 1	OP ≥ 0 RES = 0	OP ≥ 0 RES = 0
last one (Recov. C & S)	OP = 1 Res = 1	OP = 1 Res = 1	OP = 1 Res = 1	OP ≥ 1 Res = 1	OP ≥ 1 Res = 1	OP ≥ 1 Res = 1
only once t1 (Regular Server)	OP = 1 Res = 1	OP = 1 Res = 1	OP = 1 Res = 1	OP ≤ 1 RES = 0	OP ≤ 1 RES = 0	OP ≤ 1 RES = 0
only once t2 (Recov. C & S)	OP = 1 Res = 1	OP = 1 Res = 1	OP = 1 Res = 1	OP = 1 Res = 1	OP = 1 Res = 1	OP = 1 Res = 1
all or nothing (Transactions)	OP = 1 Res = 1	OP = 1 Res = 1	OP = 1 Res = 1	OP = 1 Res = 1	OP = 0 RES = 0	OP = 0 RES = 0

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann

Fernaufruf: Ideale Eigenschaften eines transparenten Mechanismus

❑ Verwaiste Aufrufe (orphans)

- Fragestellung
- Extermination

  gezielte Elimination

  Adoption
- Expiration

  Zu kurze Intervalle verursachen auch bei Fehlerfreiheit Abweichen von der exactly-once-Semantik

  Problem: Uhrensynchronisation, um Verweilzeit im Kommunikationssystem berücksichtigen zu können
- Deadlining with postponement
- Reincarnation

  weak last-one

  gentle

Fernaufruf: Ideale Eigenschaften eines transparenten Mechanismus

❑ Weitere Punkte

- wrap around

  16 Bit, jede msec eine Nachricht: 1,1 min

  32 Bit, jede msec eine Nachricht: 1200 Stunden

- Stabilität der Sequenznummern bei Prozessorausfällen

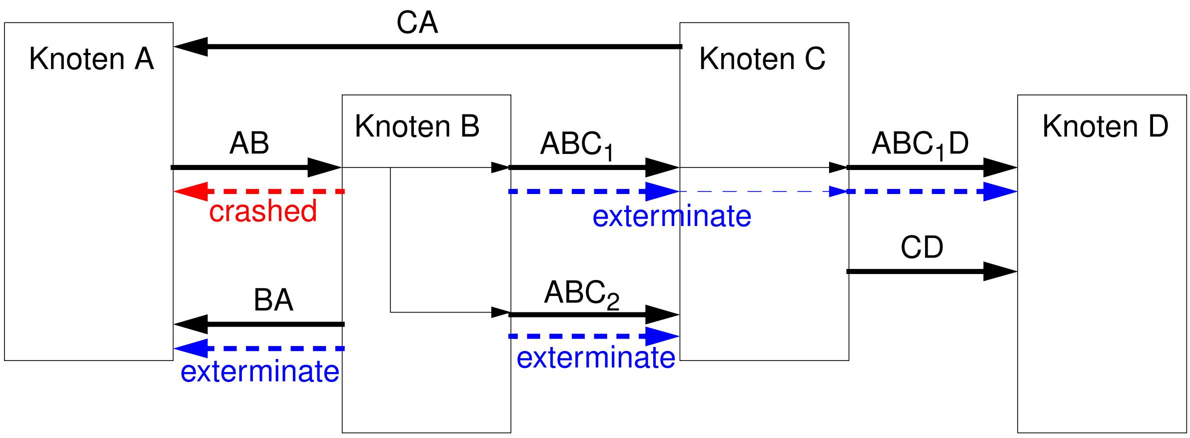
  Generation zusammen mit Sequenznummer;

  Generation nur geändert bei Recovery

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Fernaufruf: Ideale Eigenschaften eines transparenten Mechanismus

Extermination

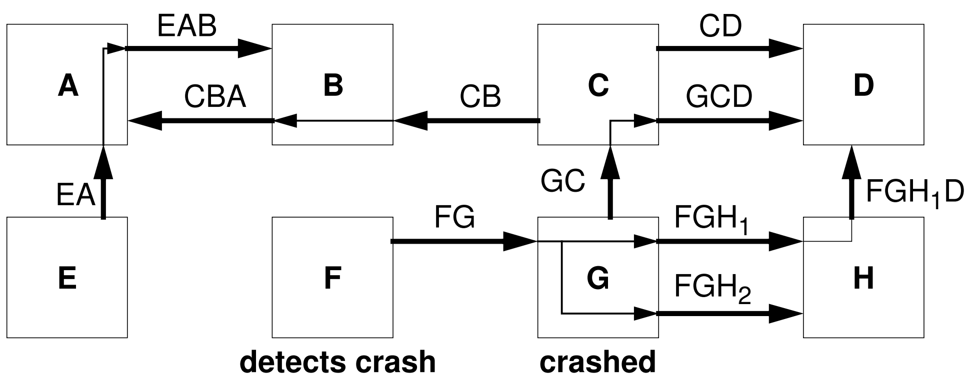


recovering

nodesIn	nodesOut
A	A, C
child crashed	exterminate children

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann

Fernaufruf: Ideale Eigenschaften eines transparenten Mechanismus



Extermination: GC, FGH_1, FGH_2 rekursiv: GCD, FGH_1D crashed: FG

Expiration: GC, GCD, FGH_1, FGH_2, FGH_1D

Reincarnation: regular: alles
gentle: wie bei Expiration

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann