2nd IEEE International Symposium on
Object-Oriented Real-Time Distributed Computing

Towards Predictable CORBA-based Web-Services

Andreas Polze, Jan Richling, Janek Schwarz and Miroslaw Malek
Department of Computer Science
Humboldt University of Berlin

Towards Predictable CORBA-based Web-Services

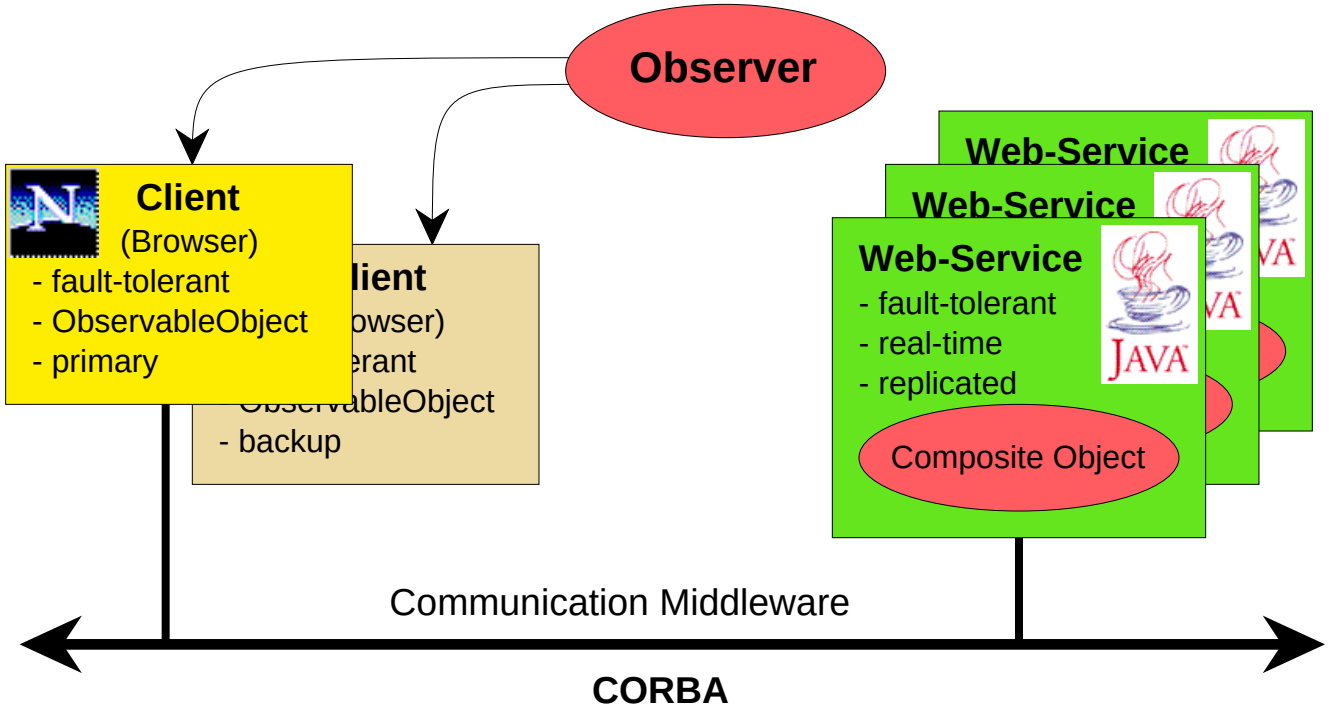
- **Composite Objects**:
  Building Blocks for Predictable CORBA-based Services

  Real-time & Call Admission:
  - Producer/Consumer/Viewer
  - Scheduling Server for predictable execution

  Fault-tolerance:
  - Observer/Observable Objects: FT Netscape
  - Broker+Group Communication: FT Maze

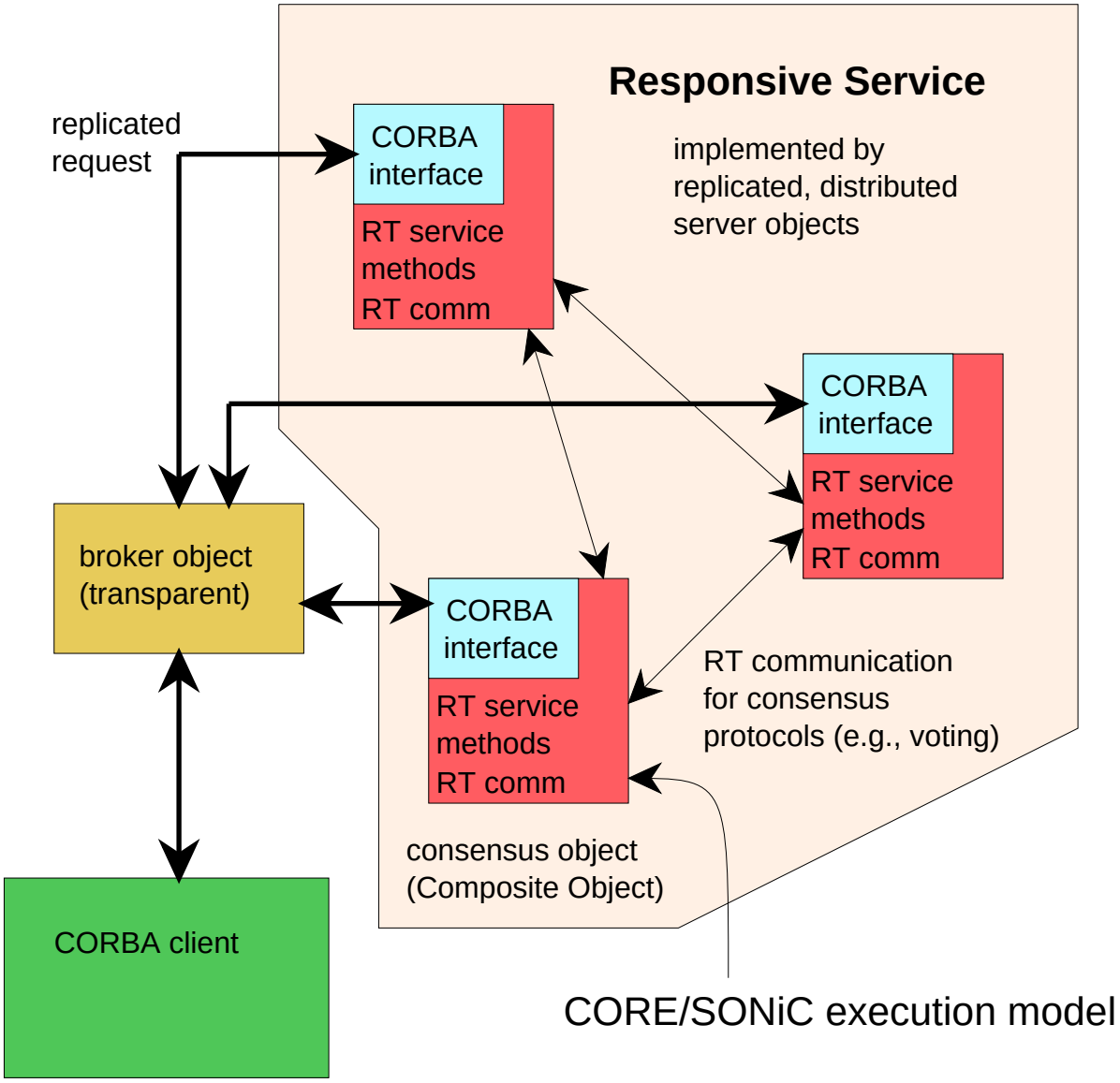
- Conclusions & Future Work
  - FT-DIO: Distributed I/O-Framework

Endsystem Architecture for Predictable Web-Services



- Observer/Observable Objects for fault-tolerant clients
- Composite Objects/Consensus for responsive Web-services

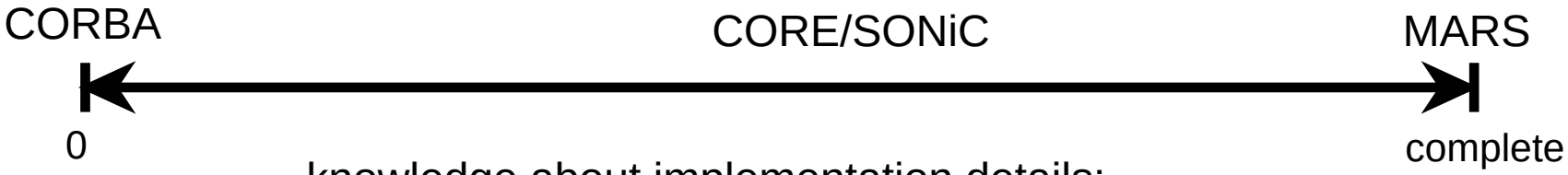
Responsive Services



- Fault-tolerance: redundancy in time and space
- Real-time: guarantees from underlying OS (Mach)
- Method invocation as unit of replication/scheduling

Responsive Computing with CORBA:
Mismatch of system assumptions

Problem:



knowledge about implementation details:
- resource usage,
- timing behavior,
- scheduling policies

Solutions:

1) "Realtime CORBA": quality-of-service guarantees for CORBA
by extending specification

2) "Responsive Services": based on CORE/SONiC and CORBA
connected by Composite Objects

-> **Composite Objects** for predictable integration of CORBA with FT RT computing

CORBA and Real-Time Computing:

1. NONINTERFERENCE:

- we should create an environment in which general purpose computing and real time computing will not burden each other.

2. INTEROPERABILITY:

- the services exported by general purpose computing objects and by real time computing objects can be utilized by each other.

3. ADAPTIVE ABSTRACTION:

- lower level information and scheduling actions needed by real time computing is available for real time objects but transparent to non-real time objects.

Standard CORBA is not sufficient.
Modifications to ORB implementations not desirable.

-> Composite Objects

Composite Objects: filtering bridge

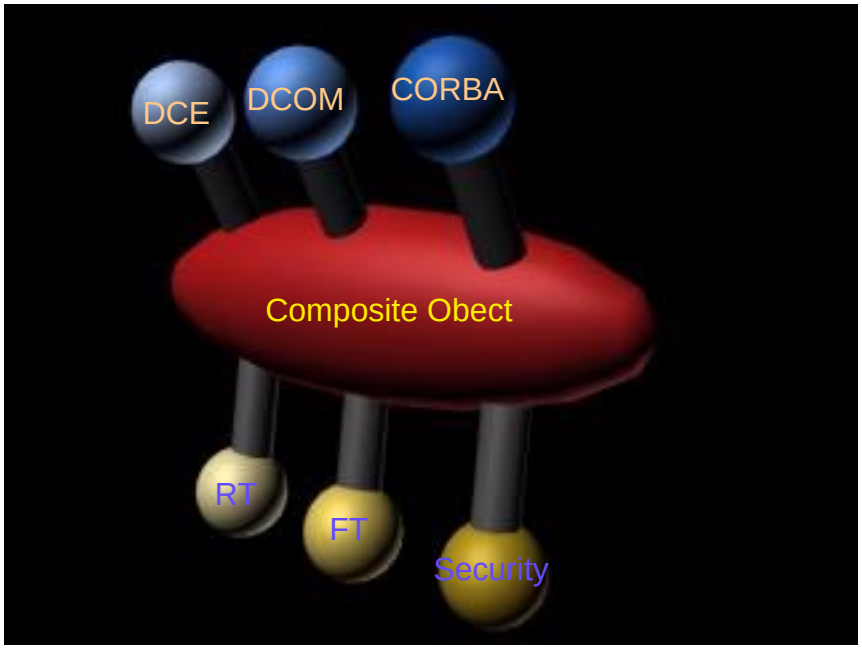
predictable integration of RT and non-RT (CORBA) functionality

ideally: multiprocessor to separate RT & non-RT tasks

- use standard scheduling techniques:
  - RMA for RT tasks
  - interactive scheduling/aging for non-RT tasks

now: simulate multiprocessor on uniprocessor

- vertical firewall: time slicing / Scheduling Server



- horizontal firewall: assign different priority levels to RT/non-RT tasks
- Composite Objects provide functions for assignment of priorities to methods and for registration with Scheduling Server

Replicate object's data: RT & non-RT part

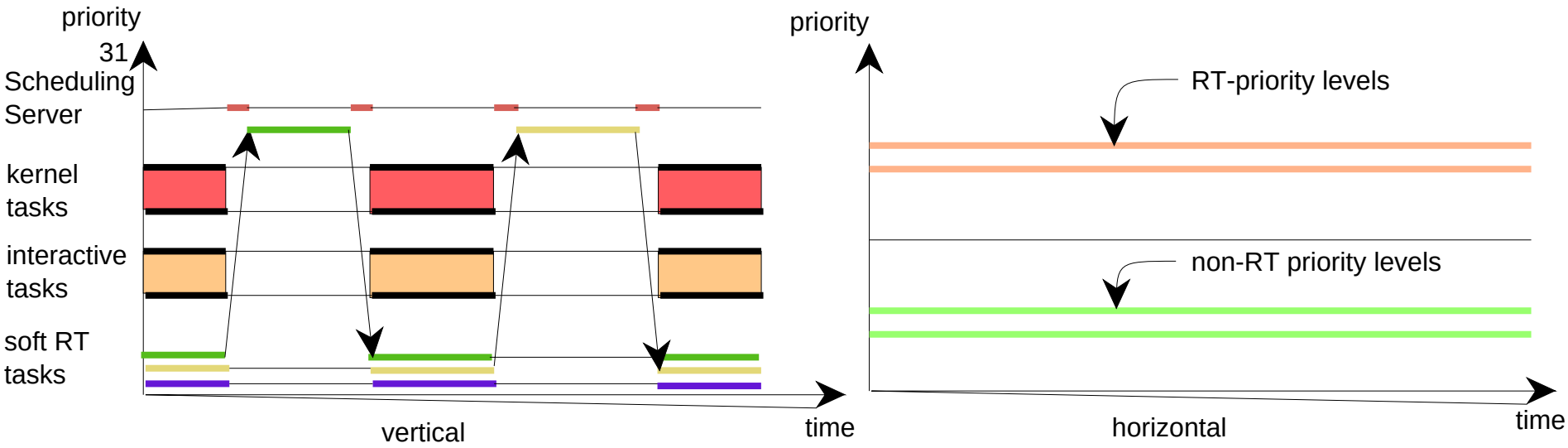
Management of data transfer/mirroring between RT & non-RT part of the Composite Object:

- shadow variables for continous data
- buffers+flush+exceptions+timeouts for discrete data
  (depth of buffer is parameter)
- variables/buffers with different priorities

Memory managment

- memory locking for RT data -> overloaded versions of new/malloc
- paging for non-RT data

CPU Firewalls

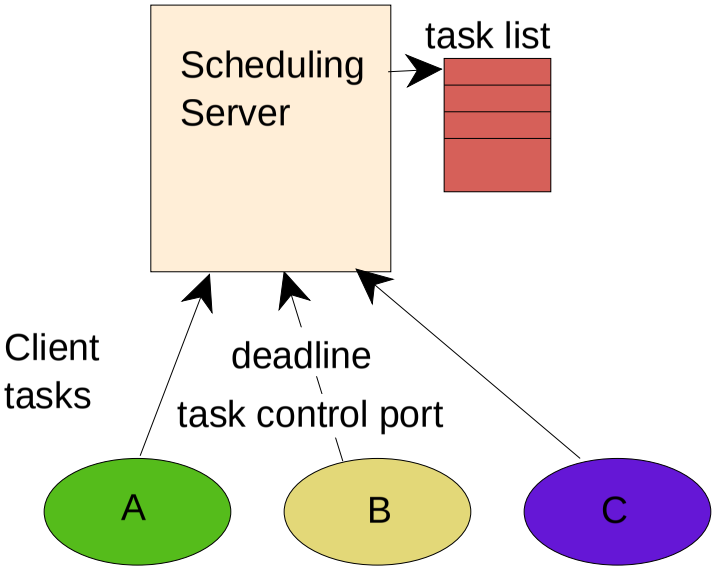


Scheduling Server: restricting CORBA in its CPU usage

- no changes to Object Request Broker required
- without changes to Mach OS kernel (user space server)
- similar work exists for rtLinux, Solaris (URsched)

Scheduling Server Concept

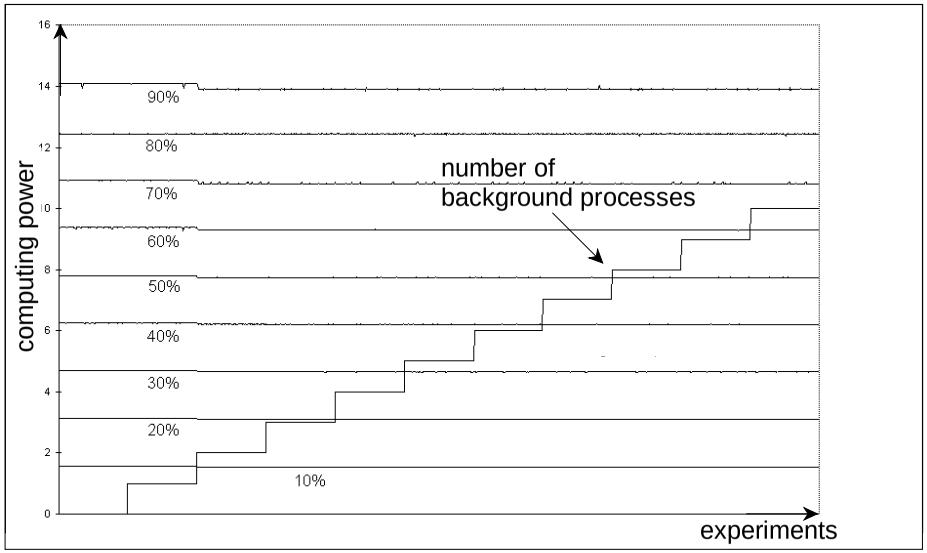
- high priority server manipulates client thread's priority
- fixed priority scheduling policy
- handoff scheduling - hints to the OS' scheduler



Scheduling Server implements:

- Earliest Deadline First (EDF)
- Rate Monotonic Scheduling (RMS)

and ensures **interactive availability** !

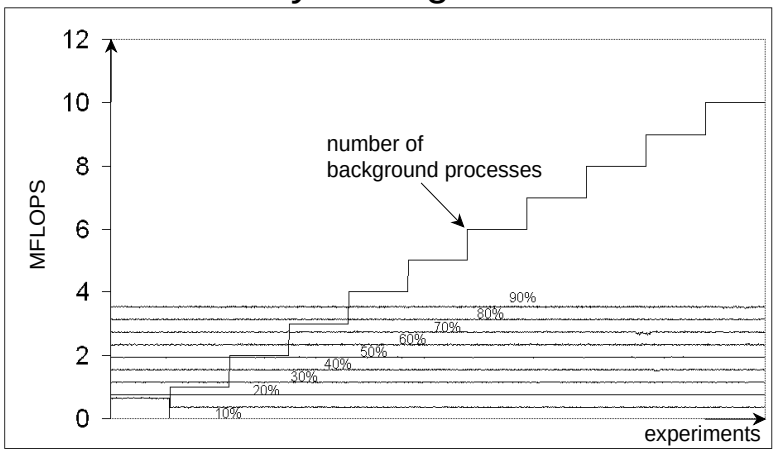


stability of rtLinux version under varying background loads

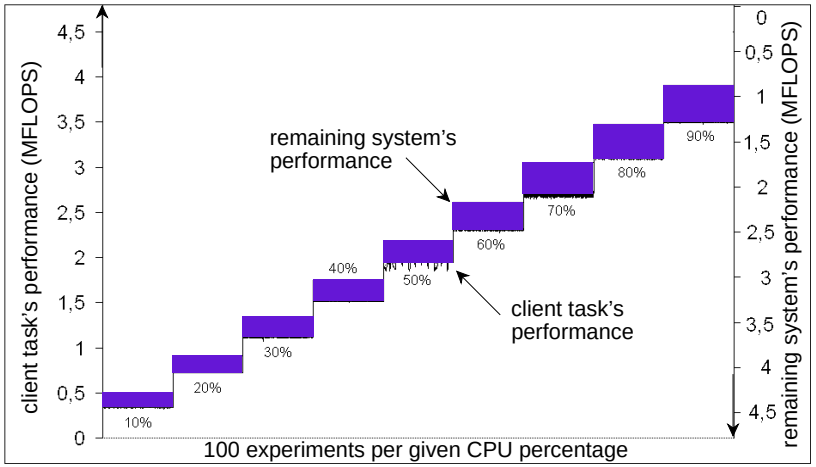
Scheduling Server: overhead and stability

- implementation based on MachOS (NeXTSTEP), HP PA-RISC
- little impact of varying background/ disk I/O loads
- overhead less than 10%, typical 5%

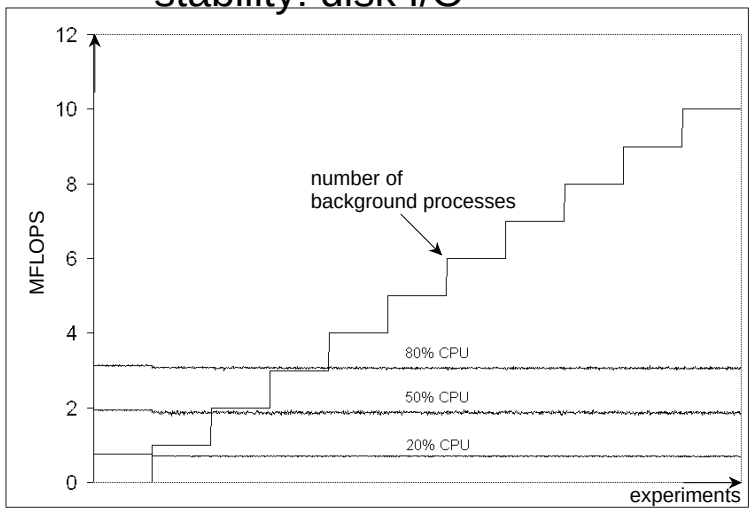
stability: background load



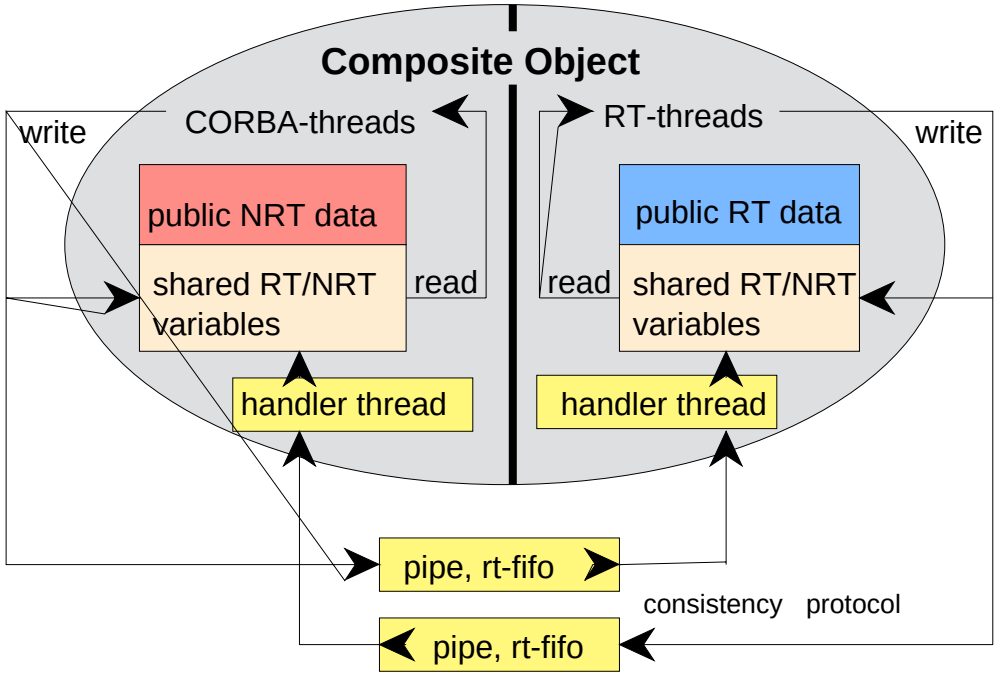
overhead



stability: disk I/O



Communication inside a Composite Object

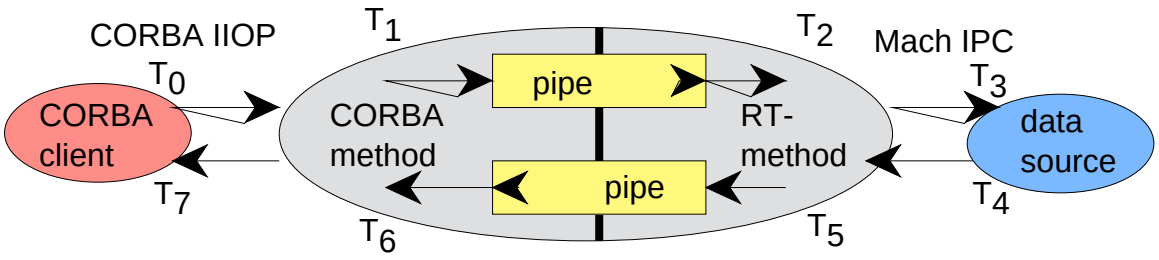


- replicated data: shared Real Time / Non-Real Time variables
- weakly consistent memory management

- handler thread implements data mirroring:
  - periodically, programmable update rate
  - event-triggered

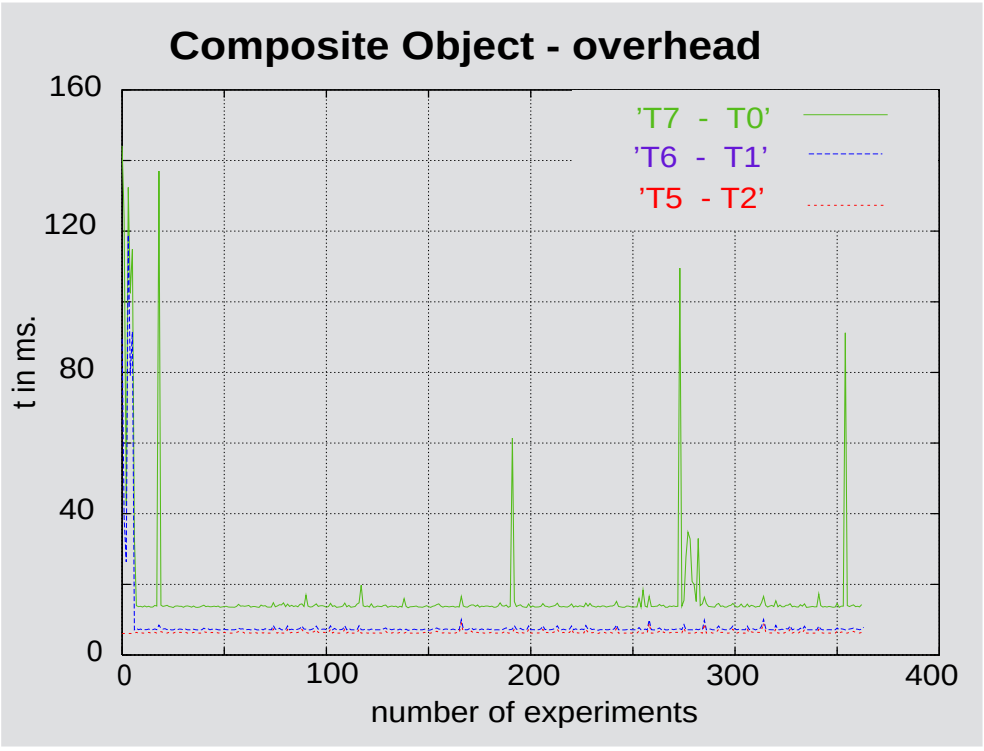
Composite Object´s Overhead

Scenario:



Environment:

- Composite Object´s host:
  HP 715/50, NeXTSTEP 3.3, ILU 2.0 alpha 12
- CORBA client:
  SparcStation 2, Solaris 2.6, OOC OmniBroker 2.02



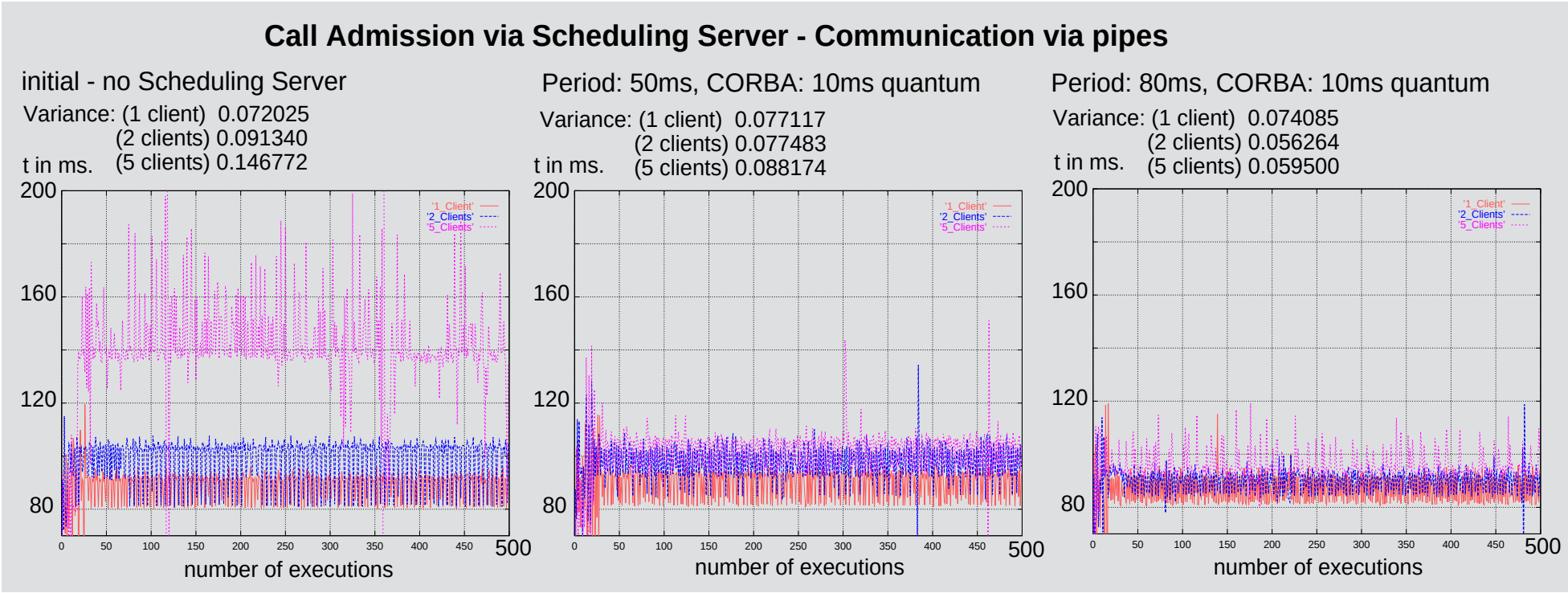
Observations:

Stable timing behaviour inside Composite Object.
Communication latency increased by 1ms.

A.P./M.M. 4/98

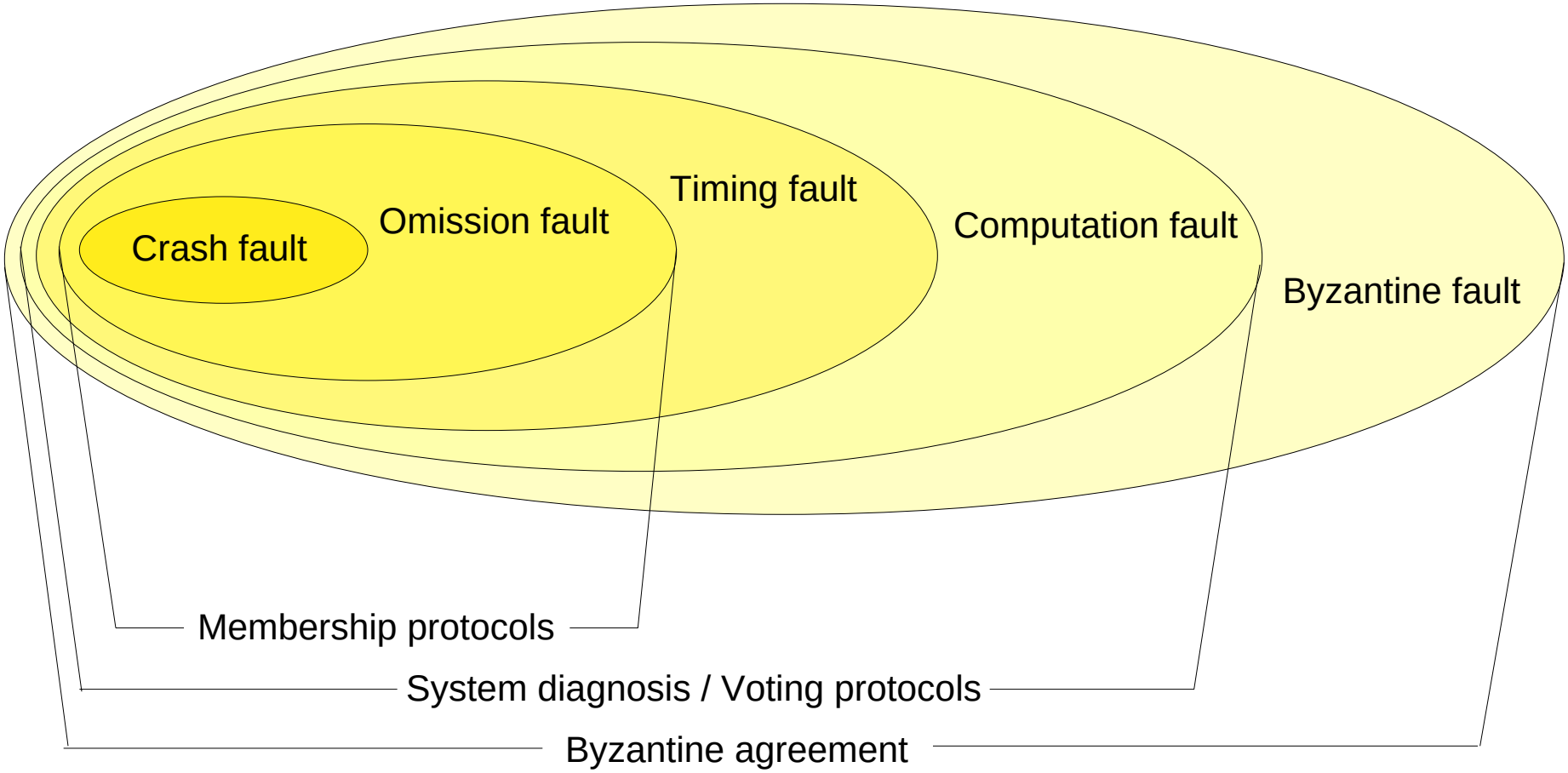
Restricting the ORB:
call admission via Scheduling Server

Version B: (contd.)
Viewer is CORBA client



- Object Request Broker is restricted in its CPU usage:
  - independent of load/number of clients, calls, objects
- tradeoff between predictability and communication latency
- no changes neither to ORB nor OS kernel necessary

Software Fault-tolerance - Fault Model

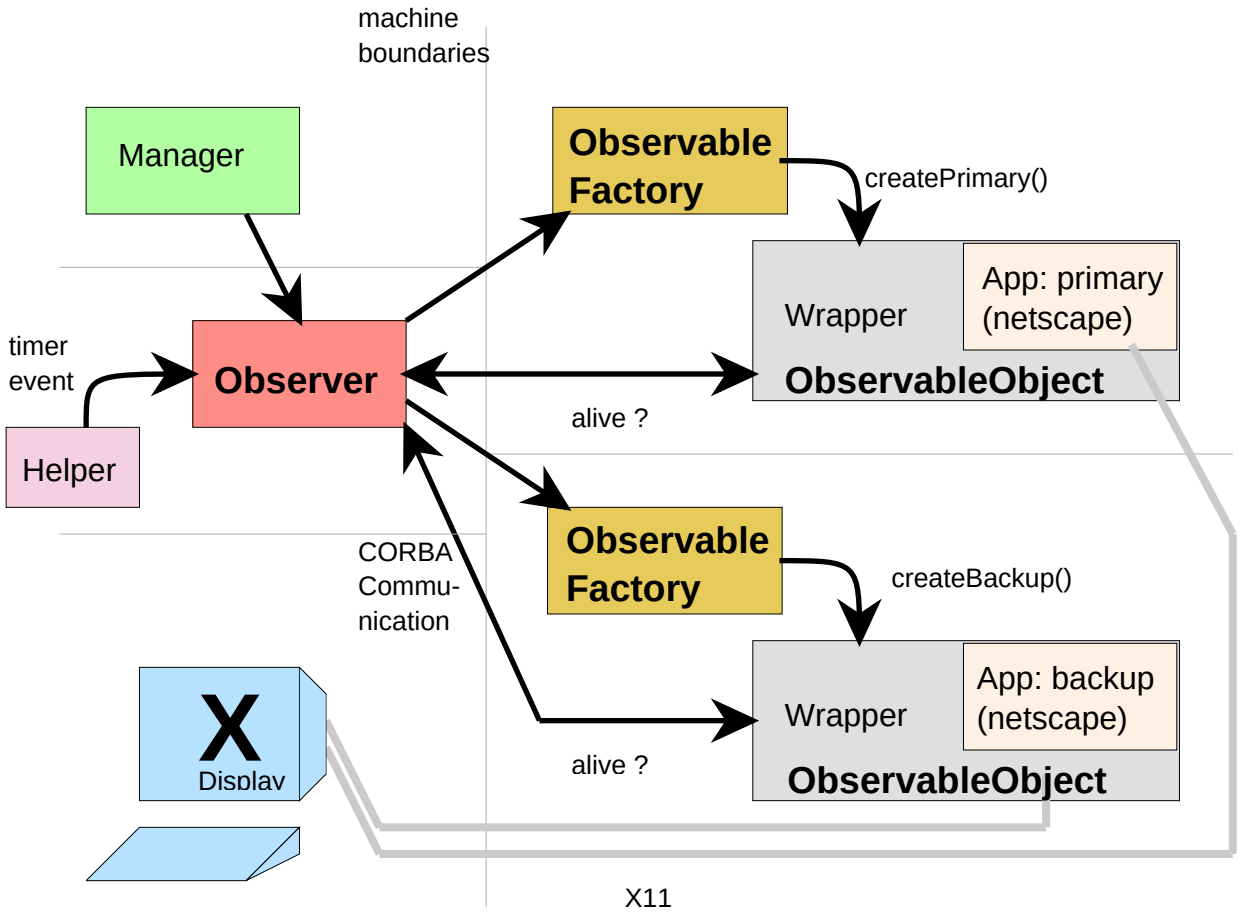


Observer/Observable Object - Software Fault-tolerance

- Recover-and-Retry / primary-backup approach
- Toleration of crash-faults (program, processing node)
- Observer monitors application/wrapper via "alive"-messages
- ObservableObject encapsulates standard UNIX apps.
- Detection of an application’s status / checkpointing:

  - stdin/stout
  - UNIX signals: `kill (pid, 0)`
  - IPC: pipes, shared memory, messages
  - X11 mechanisms: properties
  - program specific API, i.e., CORBA interface
- Exposition of backup on failover;
  backup instance becomes primary
- Start of a new backup instance

Fault-tolerant Netscape - Communication Structure



IDL interfaces - Observer / ObservableObjects

Observer:

```
typedef short Id;
interface ObservableObject;
interface ObservableFactory;

interface ObserverBasics {
    exception BadExec { string why; };
    void connect( in ObservableObject ref,
        in Id id, in Id fid ) raises (BadExec);
    void connectFactory( ObservableFactory ref,
        in Id id, ) raises (BadExec);
    void disconnect( in ObservableObject ref,
        in Id id, in Id fid ) raises (BadExec);
};
interface ObserverManager {
    exception ManagerOnly { string why; };
    exception badExec { string why; };
    void start( in short m_id )
            raises (ManagerOnly, BadExec);
    void stop( in short m_id )
            raises (ManagerOnly, BadExec);
};
```

ObservableObject:

```
interface ObservableObject {
    short state();
    short id();
    void bePrimary( in short o_id );
    void shutdown( in short o_id );
};
```

ObservableFactory:

```
# include "ObservableObject.idl"

interface ObservableFactory:
    ObservableObject {
    void create_primary( in short o_id );
    void create_backup( in short o_id );
};
```

Fault-tolerant Netscape - Fault detection mechanism

- X properties for monitoring / controlling Netscape Navigator
  - `_MOZILLA_URL, _MOZILLA_LOCK,`
  - `_MOZILLA_COMMAND, _MOZILLA_RESPONSE`

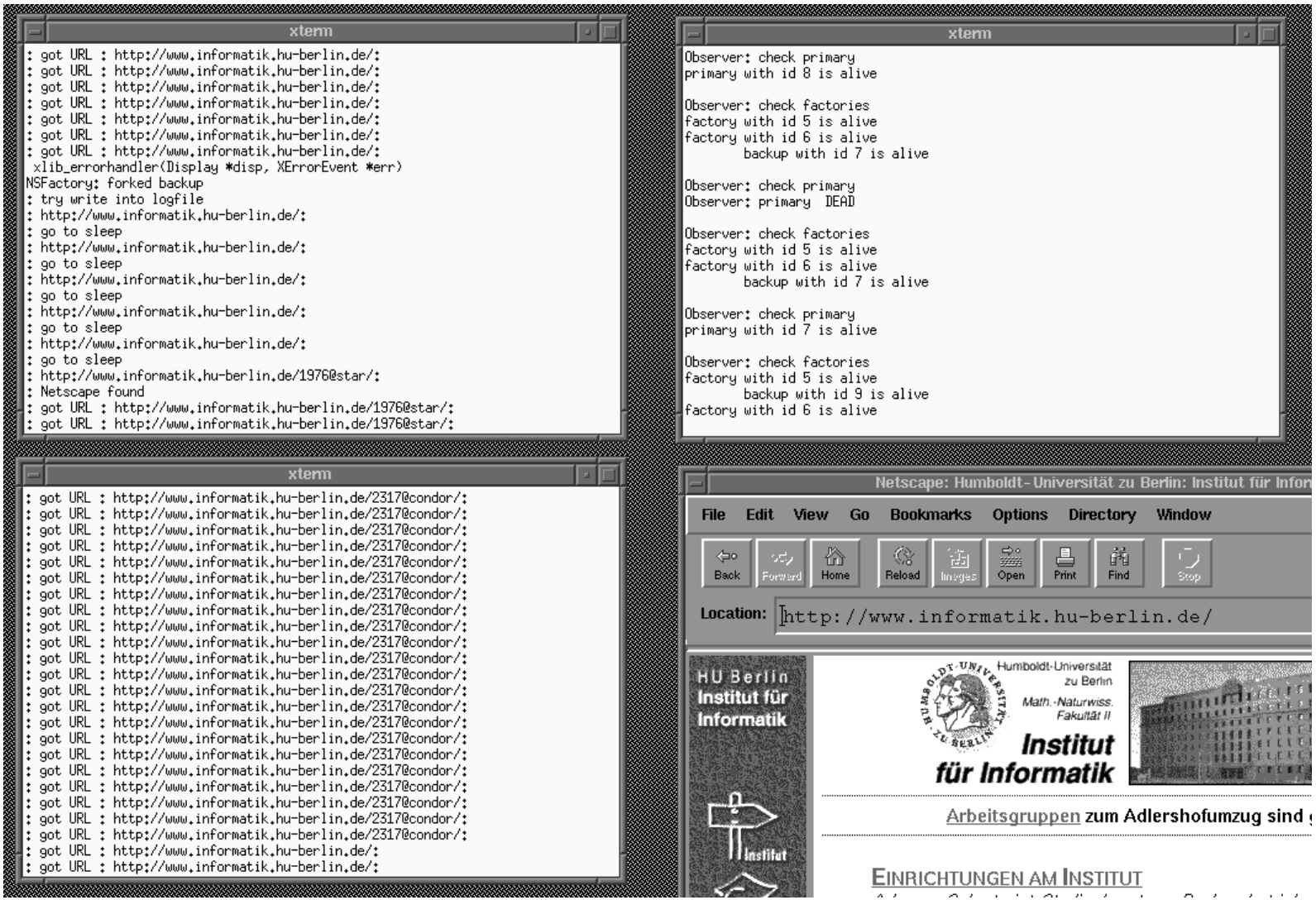
- Window ID obtained through Xlib-functions
  `XQueryTree()` and `XmuClientWindow()`
  problem: atomicity

- periodic checkpointing: store current URL in file

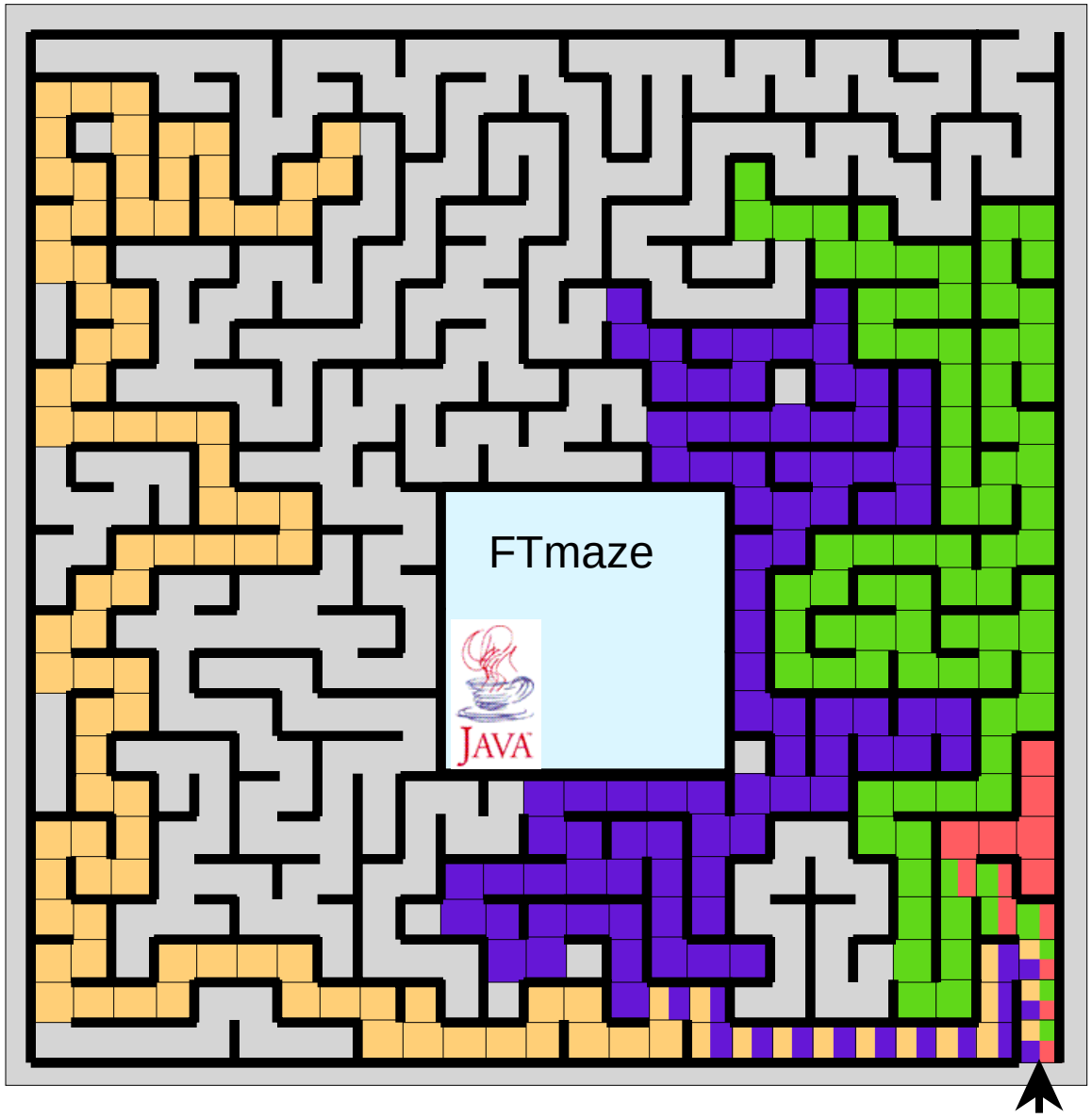
- access to non-existent X property results in X error
  -> wrapper detects application crash (Netscape Navigator)
  -> Observer activates backup as new primary
  (loading of current URL)
  -> Observer starts new backup instance

- invalid CORBA reference to ObservableFactory indicates
  crash of computer

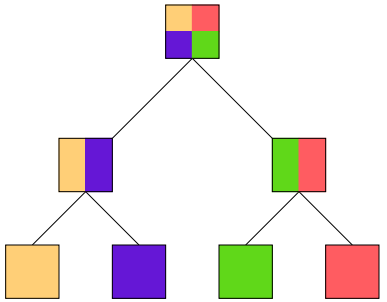
Fault-tolerant Netscape - Screendump

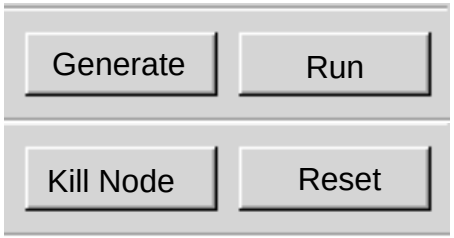


Execution of fault-tolerant labyrinth search

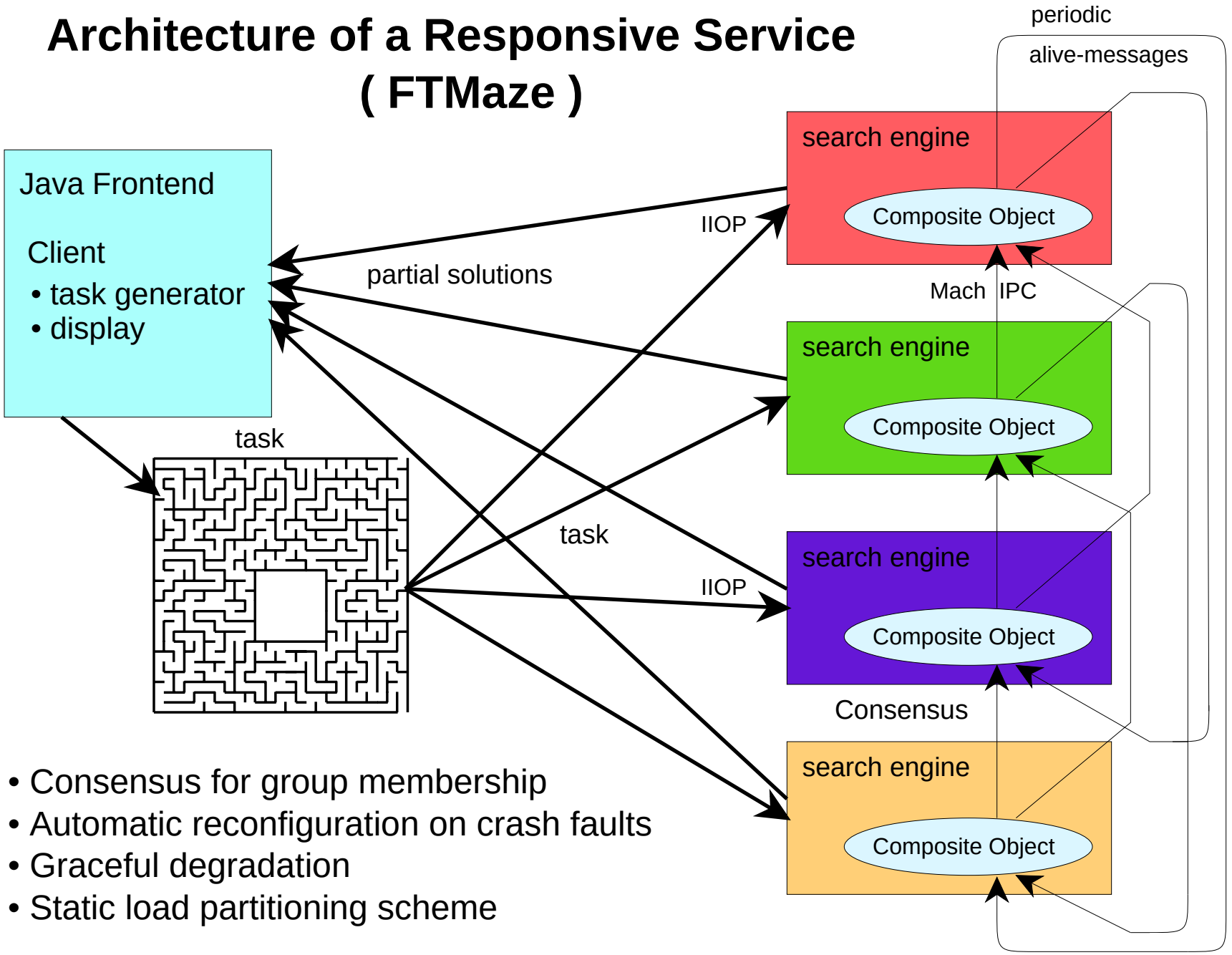


- 4 nodes
- right-hand first search rule
- load partitioning scheme



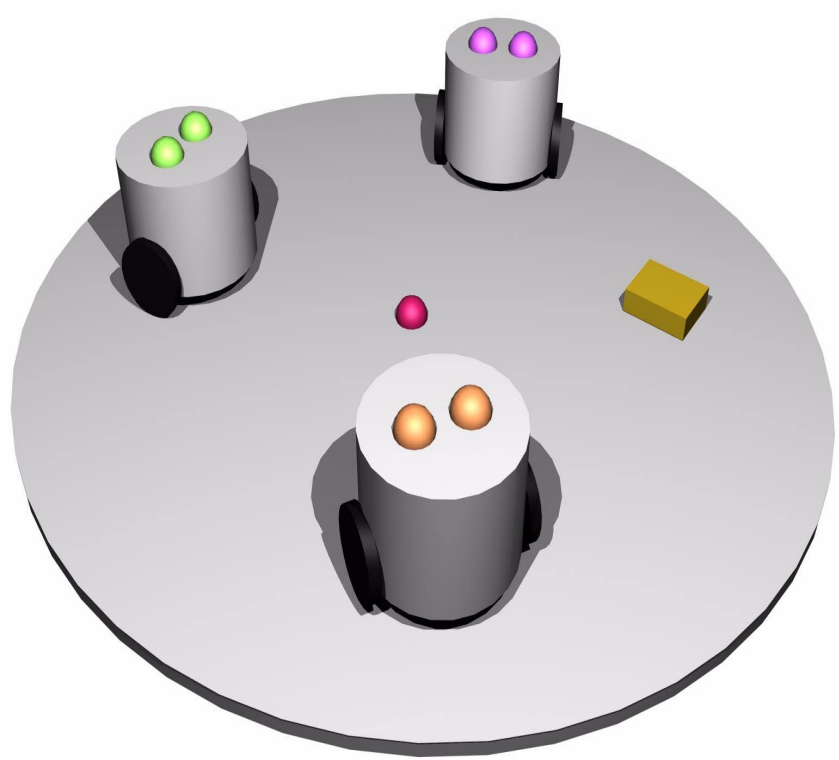


Architecture of a Responsive Service (FTMaze)

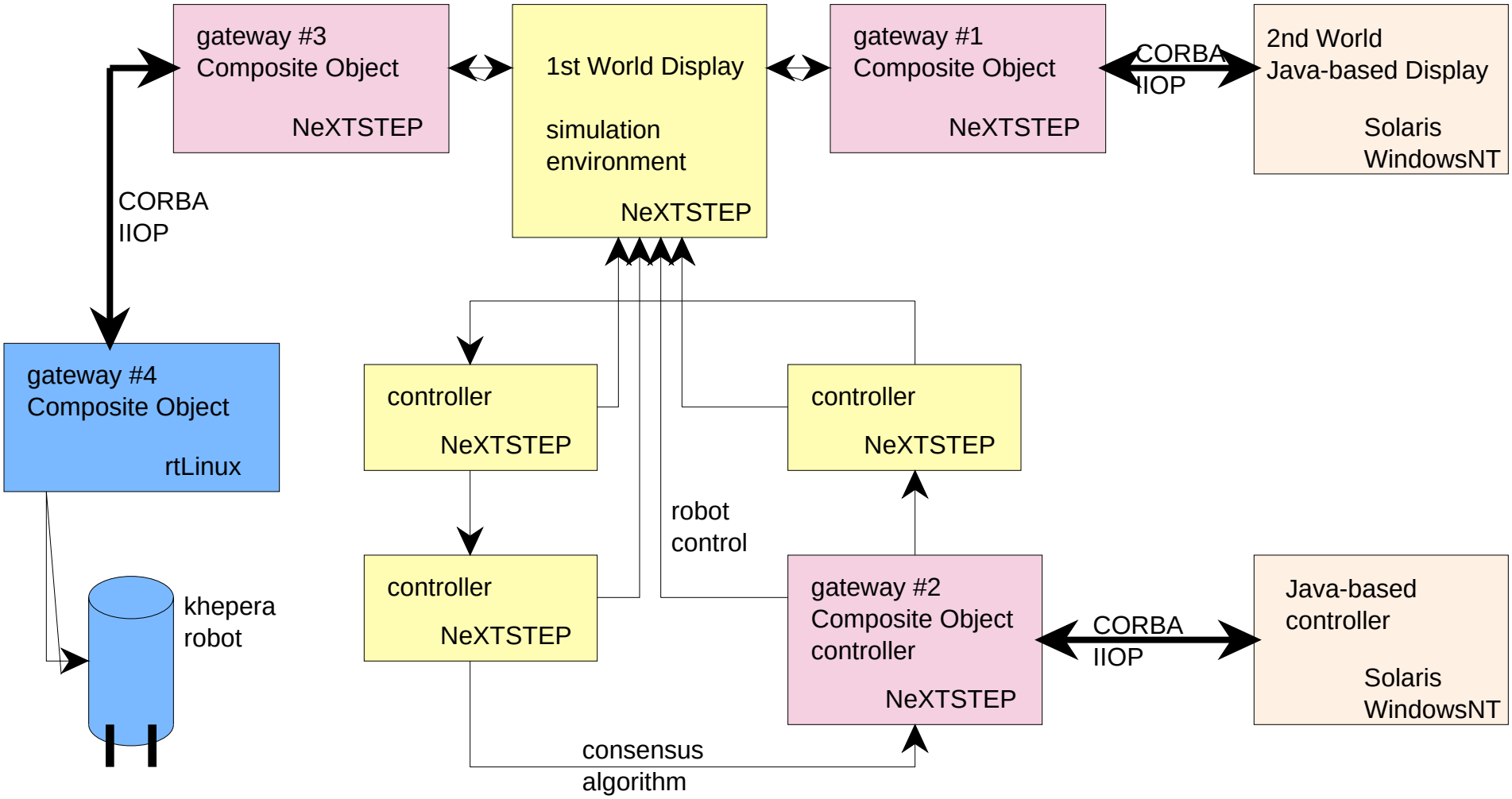


- Consensus for group membership
- Automatic reconfiguration on crash faults
- Graceful degradation
- Static load partitioning scheme

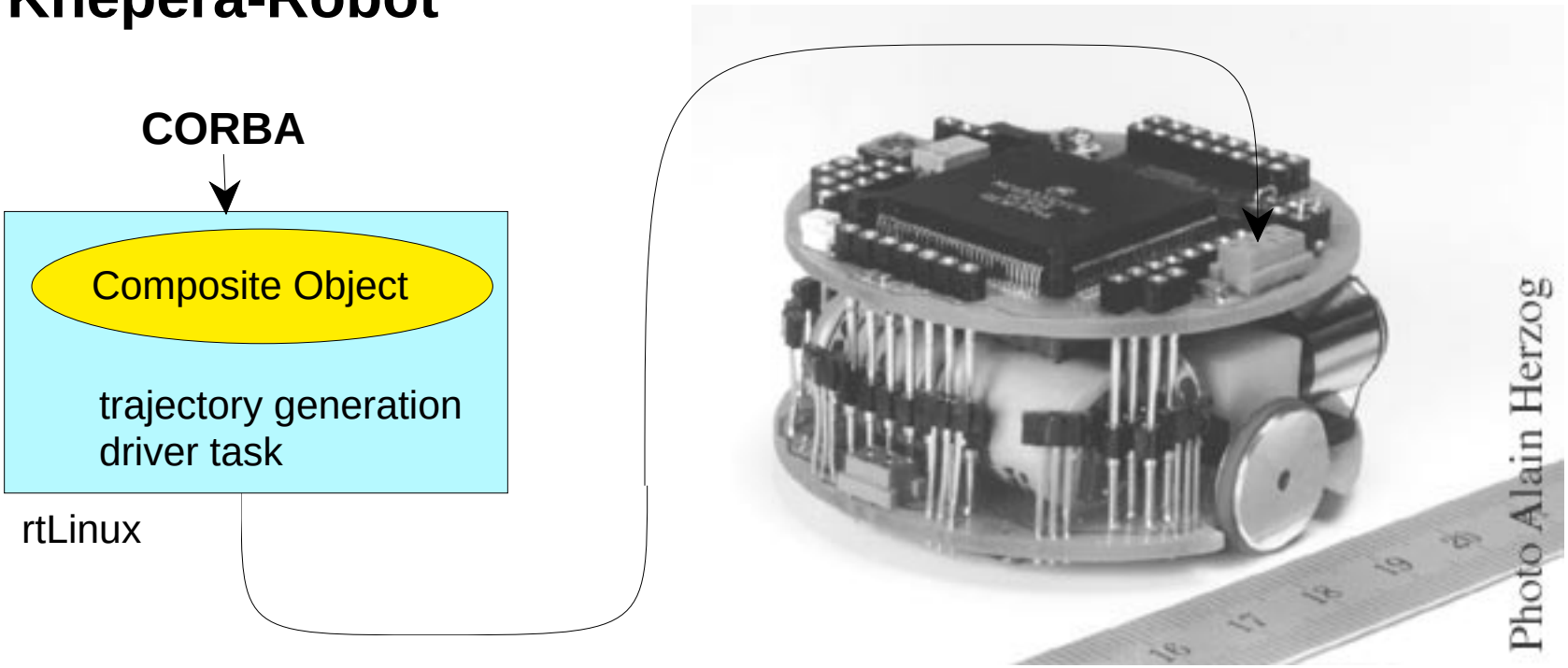
The Unstoppable Robots - a Fault-tolerant Real-time application



- Fault-tolerance: consensus protocol (voting) among replicated controllers
- Real-time: robots use up fuel with constant rate;
  moves have to be computed with 4 Hz frequency

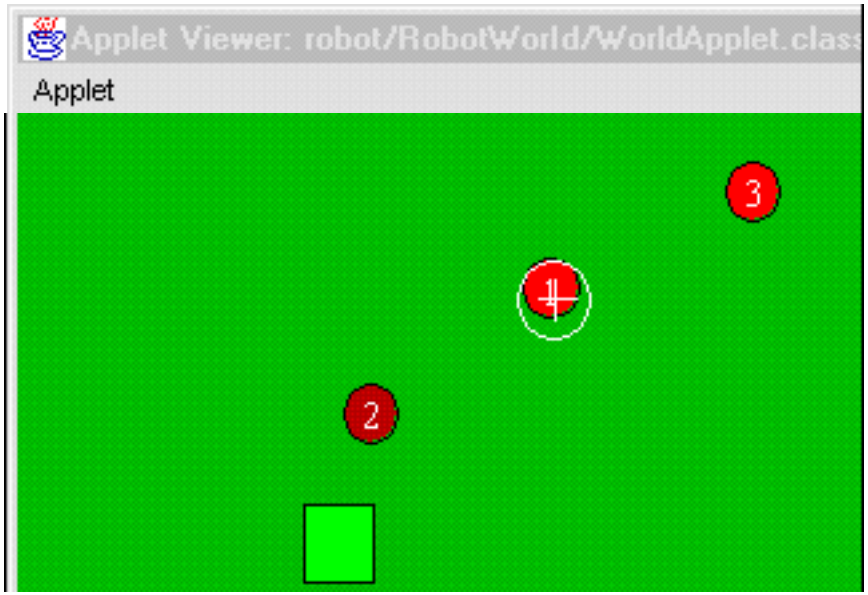

gateway #3
Composite Object
NeXTSTEP
1st World Display
simulation
environment
NeXTSTEP
gateway #1
Composite Object
NeXTSTEP
CORBA
IIOP
2nd World
Java-based Display
Solaris
WindowsNT
CORBA
IIOP
gateway #4
Composite Object
rtLinux
controller
NeXTSTEP
controller
NeXTSTEP
robot
control
controller
NeXTSTEP
gateway #2
Composite Object
controller
NeXTSTEP
CORBA
IIOP
Java-based
controller
Solaris
WindowsNT
khepera
robot
consensus
algorithm

Khepera-Robot



- Composite Object is critical: CORBA's varying communication latency must not disturb rtLinux driver task
- rtLinux driver task performs trajectory generation and outputs fine-grained motion commands with frequency of 800Hz
- simulation sends coarse-grained commands with frequency of 4Hz via CORBA

Web-interface to Unstoppable Robots



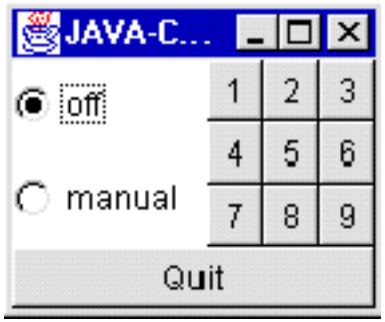
Java Applet showing Unstoppable Robots Simulation

- Open interfaces must not disturb timing behavior of real-time application
- Web-clients may create inacceptable high loads
- read accesses are easy:

Composite Object implements caching and call admission

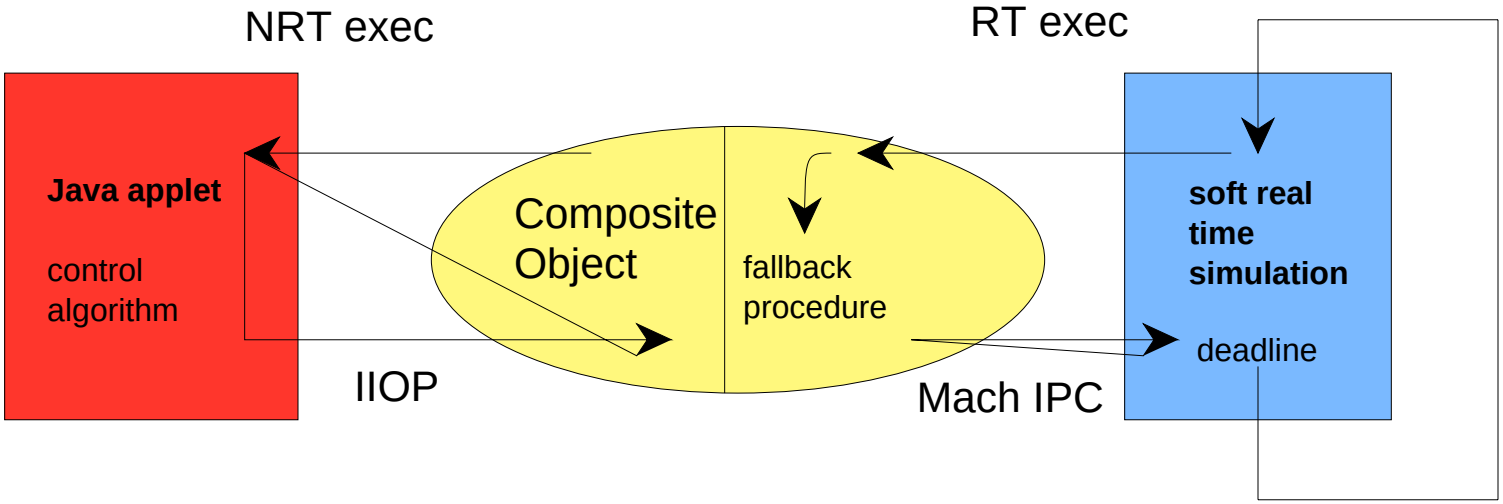
- write accesses may modify RT-data
- RT-app must kept stable even if deadlines are missed

Recovery Blocks: RT-part of Composite Object implements save fallback-method



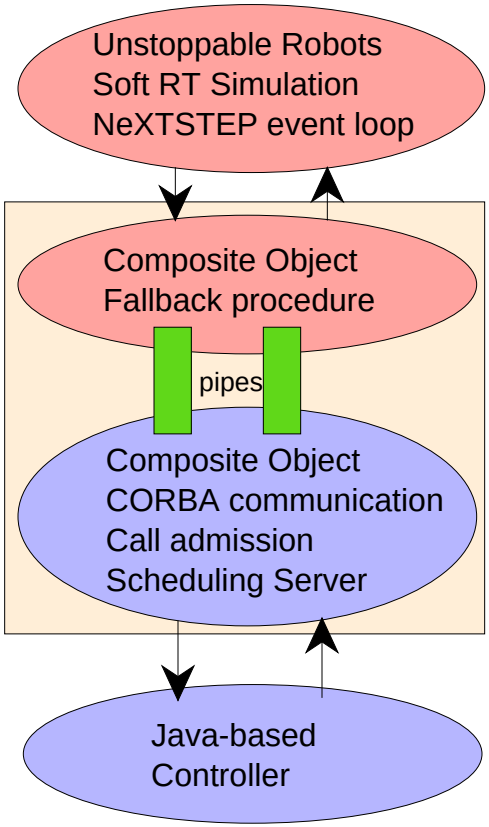
Java Controller
Virtual Joystick for one robot

Analytic Redundancy - RT fallback procedure

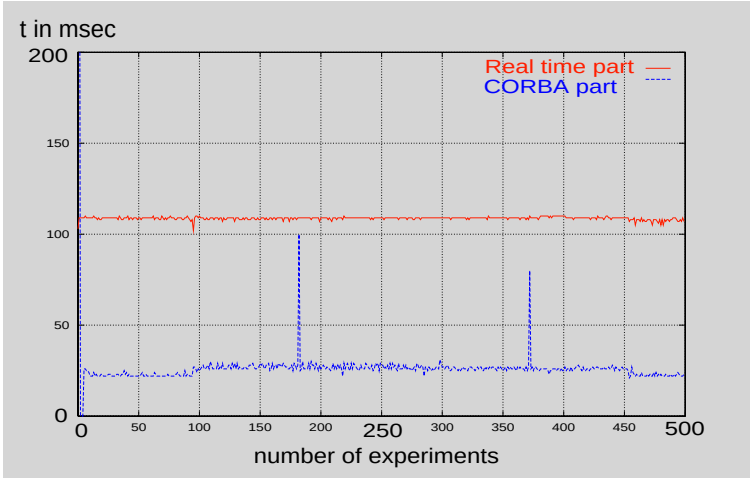


- concept similar to recovery blocks, mutli-version programming

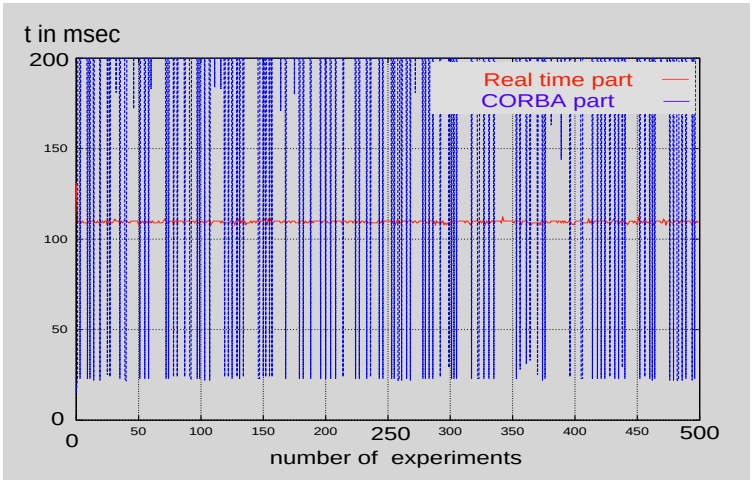
Unstoppable Robots -
CORBA interactions with Java-based external controller



periods of robots simulation



unloaded
Windows NT
system running
external controller

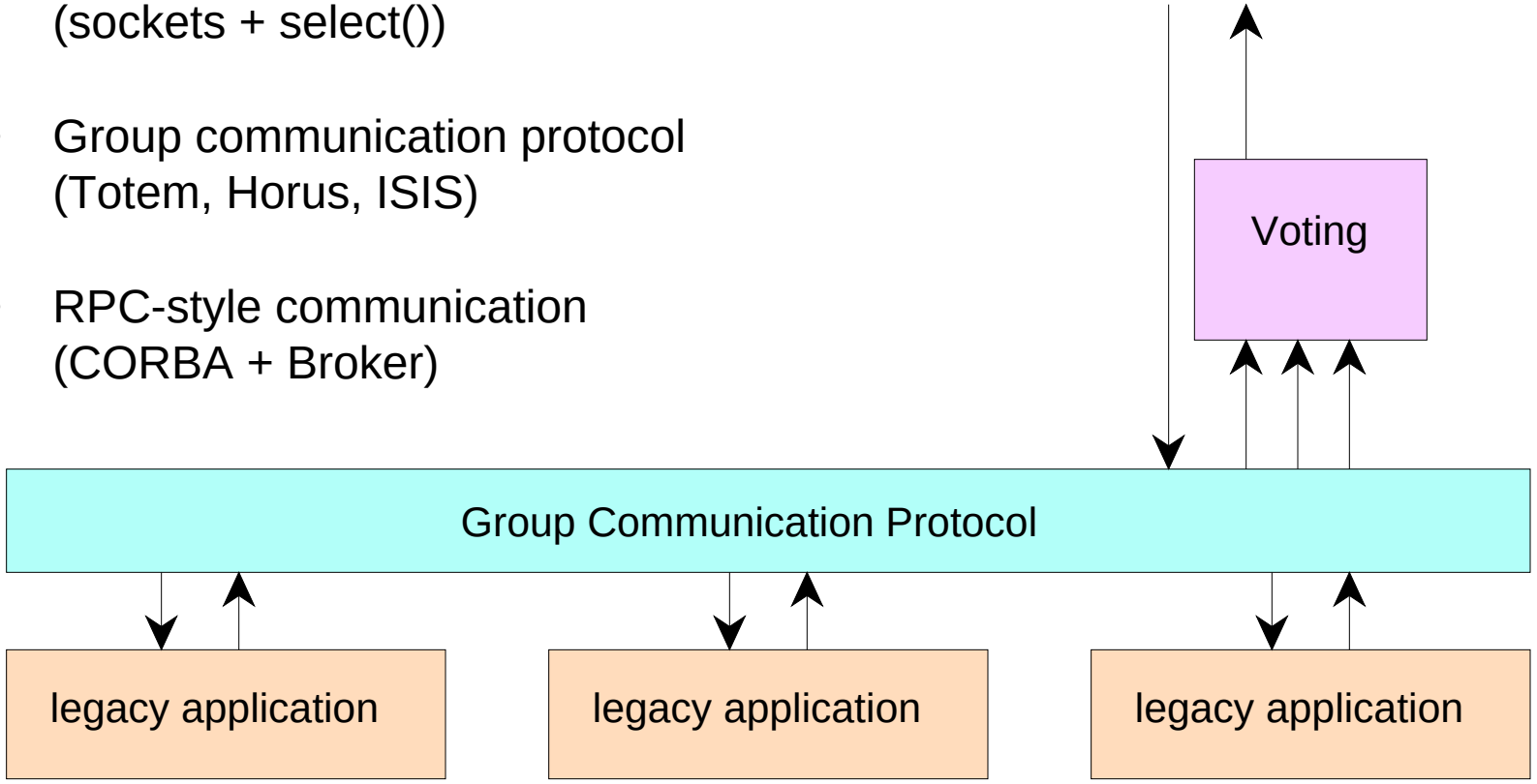


loaded
Windows NT
system running
external controller

Future:

Fault-tolerant Distributed Input/Output (FT-DIO)

- stream-based communication
  (sockets + select())
- Group communication protocol
  (Totem, Horus, ISIS)
- RPC-style communication
  (CORBA + Broker)



Conclusions

- Composite Objects allow for predictable integration of CORBA and responsive computing with small overhead
- Scheduling Server provides basis for true call admission without changes to CORBA (ORB) and OS kernel
- Open Web interfaces for leagcy real-time applications are feasible using Composite Objects technology
- Data replication and weak memory consistency are key concepts for decoupling CORBA and responsive computing
- Concepts can be transformed onto many COTS platforms:
  -> rtLinux, Solaris, Windows NT

Demo Applications - Lessons learned

- Observer/ObservableObject:
  generic interfaces for fault-tolerance based on CORBA
  -> have been successfully reused

- only small subset of CORBA functionality needed for
  implementation of responsive services

  (architectural approach applicable to minimal/embedded CORBA?)

- Composite Objects decouple CORBA and
  real-time consensus protocols

- Java language binding allows Web-access to responsive services

Group, Topics, and Contact Info

- Jan Richling, Ph.D. student
  "Real Time and CORBA: Experiments with Composite Objects" (in german)
- Janek Schwarz, M.S. student
  "Fault-tolerance techniques for CORBA --
  Experiments, Measurements, Evaluation" (in german)
- Oliver Freitag, Mario Schröer, M.S. students
  "Automatic Generation of responsive CORBA-services" (in german)
- Martin Meyka, Thomas Lehmann, M.S. students,
  "Business Object Framework --
  Security- and Consistency-protocols for replicated CORBA-Objects" (in german)

Responsive CORBA Unified Environment (RESCUE)

- Robots on the Web:
  http://www.informatik.hu-berlin.de/~apolze/rescue
- Contact: Dr. Andreas Polze
  Department of Computer Science
  Humboldt University of Berlin
  10099 Berlin, Germany
  apolze@informatik.hu-berlin.de