

DeDiSys: Dependable Distributed Systems

Lorenz Frohofer

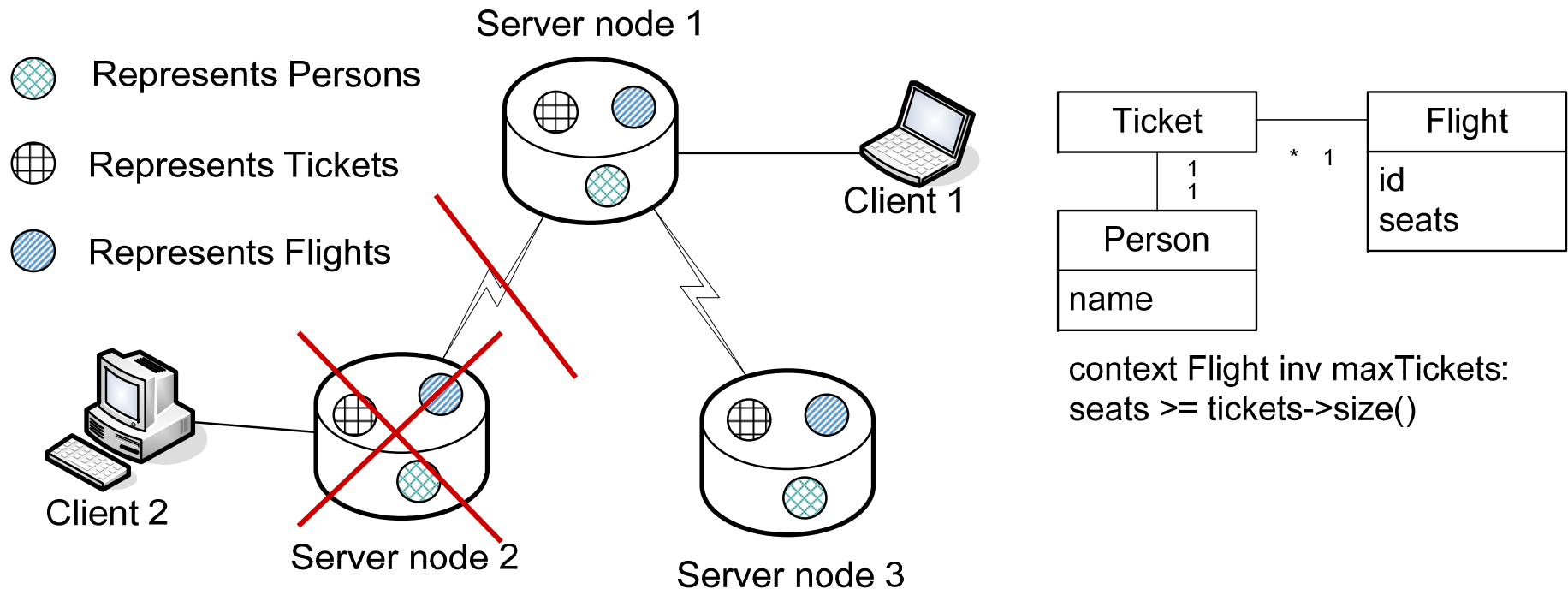
L.Frohofer@infosys.tuwien.ac.at

Vienna University of Technology

Motivation

- ❑ Distributed systems have to be highly available (7x24).
- ❑ With an increasing number of nodes and links, failures become more probable.
- ❑ If a system requires full consistency all the time (e.g. banking applications), the system becomes unavailable during failures.
- ❑ For some systems (e.g. safety or mission critical applications), availability is more important than full consistency.

Motivation: Flight booking system



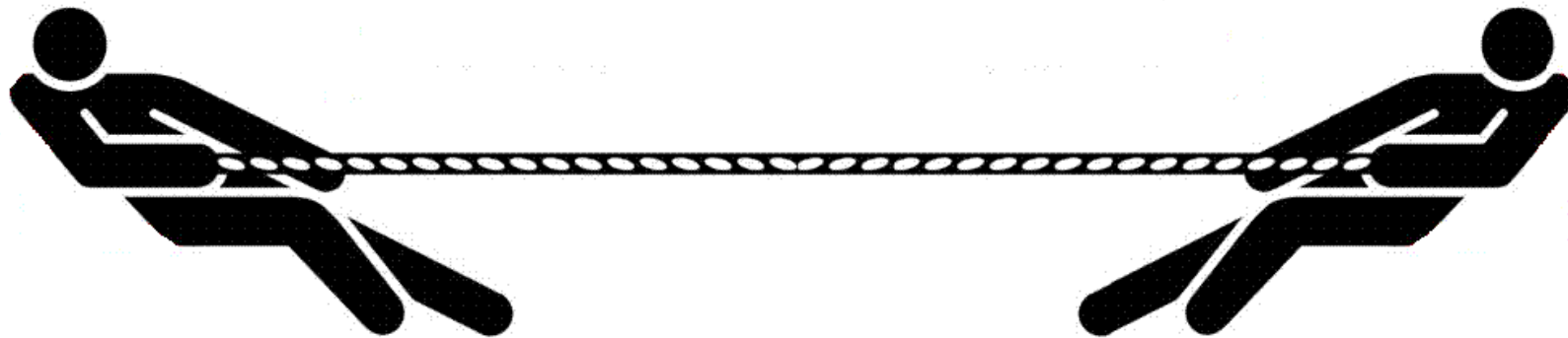
- ❑ Passive replication: primary copies are on server node 1
- ❑ Constraint:
The system must not sell more tickets than available seats for a flight

FP6 IST Project DeDiSys

- ❑ DeDiSys = Dependable Distributed Systems (<http://www.dedisys.org/dedisys/>)
- ❑ Investigates trade-off availability / consistency

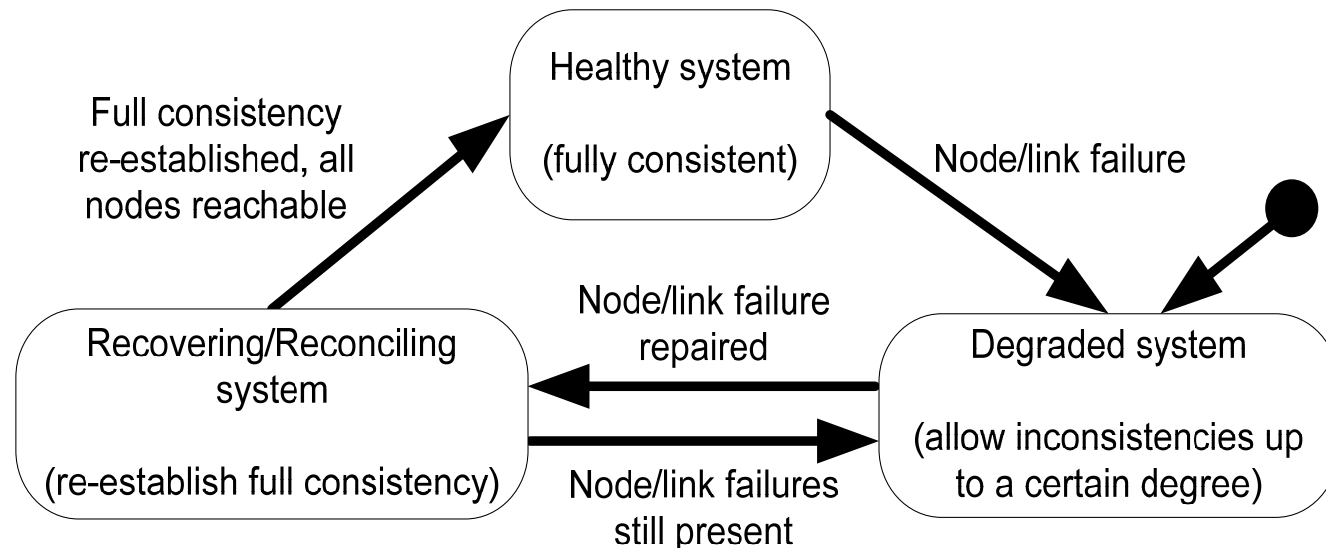
MR. AVAIL ABILITY

MR. CON SISTENCY



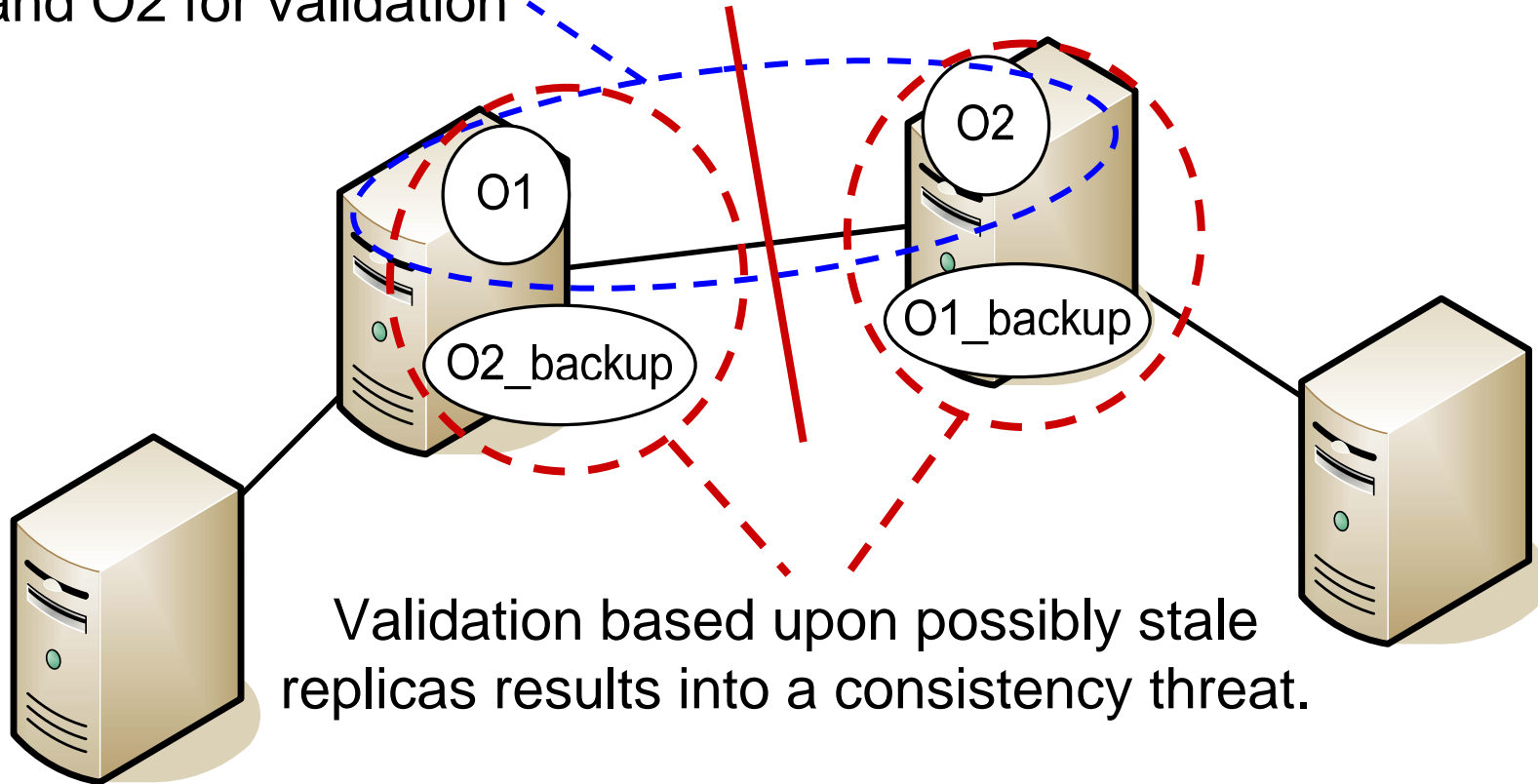
System model

- ❑ Distributed object system
- ❑ Crash failure model for nodes
- ❑ Link failure model: messages might be lost but not duplicated or corrupted
- ❑ Multipartitioning possible but not usual
- ❑ Typically 2 – 10 replicas of a single object

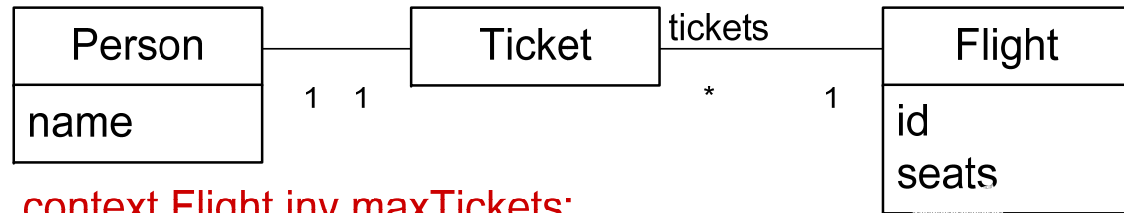


Consistency threat

Constraint C1 needs access to O1 and O2 for validation



Consistency based on integrity constraints



context Flight inv maxTickets:
seats >= tickets->size()

```
public class Flight {
```

```
...
```

```
public Ticket [] sellTickets (Person p, int ticketCount) {  
    if (getSeats() >= getSoldTickets().count() + ticketCount) {
```

```
        // Sell ticket(s)
```

```
    } else {
```

```
        // Do not sell ticket(s)
```

```
    return result;
```

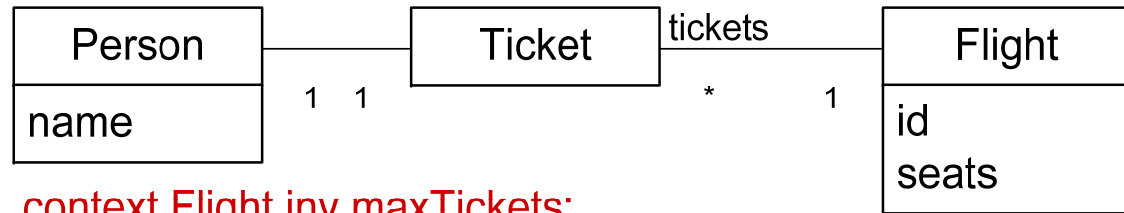
```
}
```

```
...
```

```
}
```

This does not work if inconsistency management is required!

Explicit constraint classes



```
public class Flight {
```

```
    ..
```

```
    public Ticket [] sellTickets (Person p , int ticketCount ) {  
        // Sell ticket(s)
```

```
    }  
}
```

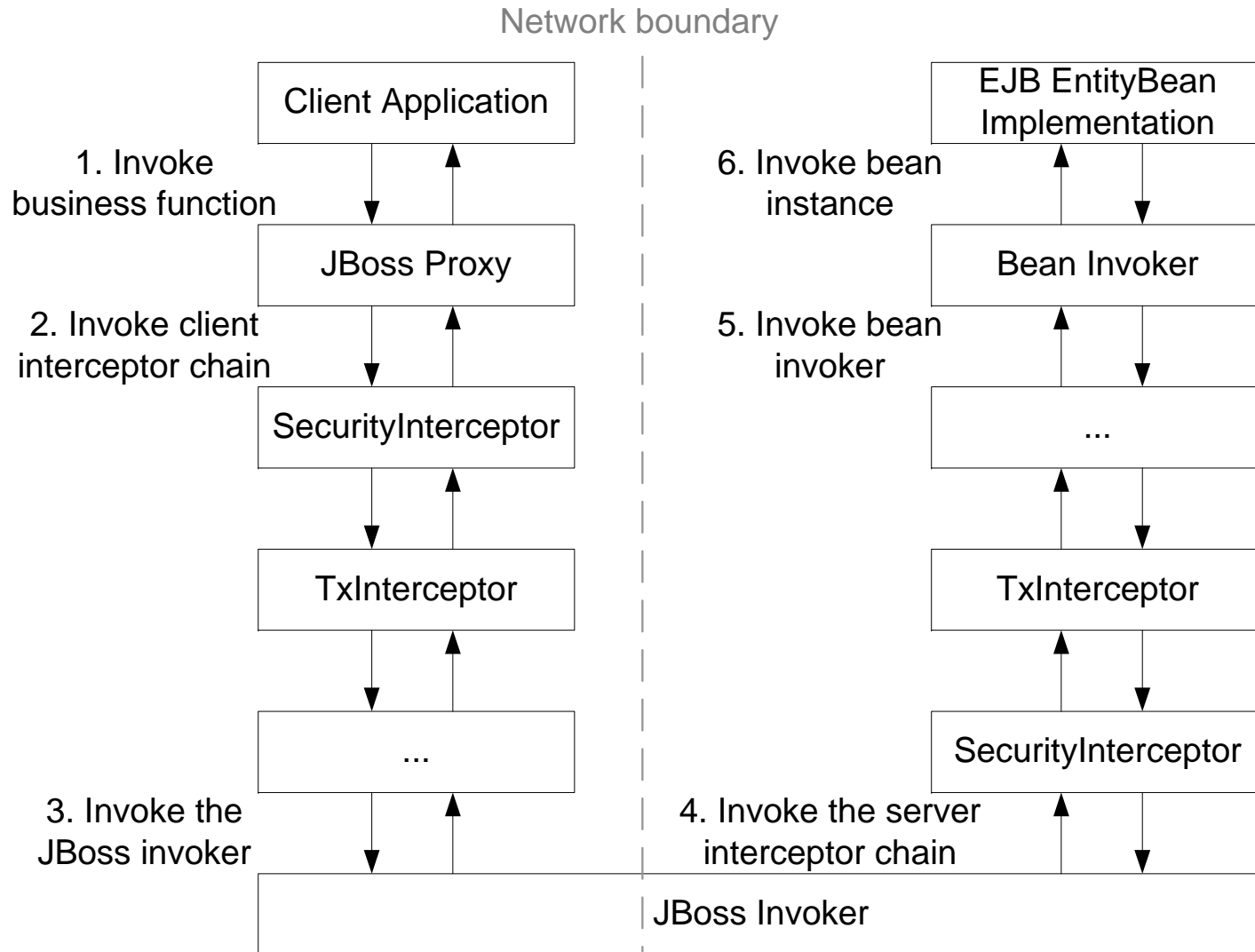
```
public class TicketConstraint {
```

```
    public boolean validate (ConstraintValidationContext ctx) {  
        Flight flight = ((Flight)ctx.getContextObject());  
        return flight.getSoldTickets().count() <= flight.getSeats();
```

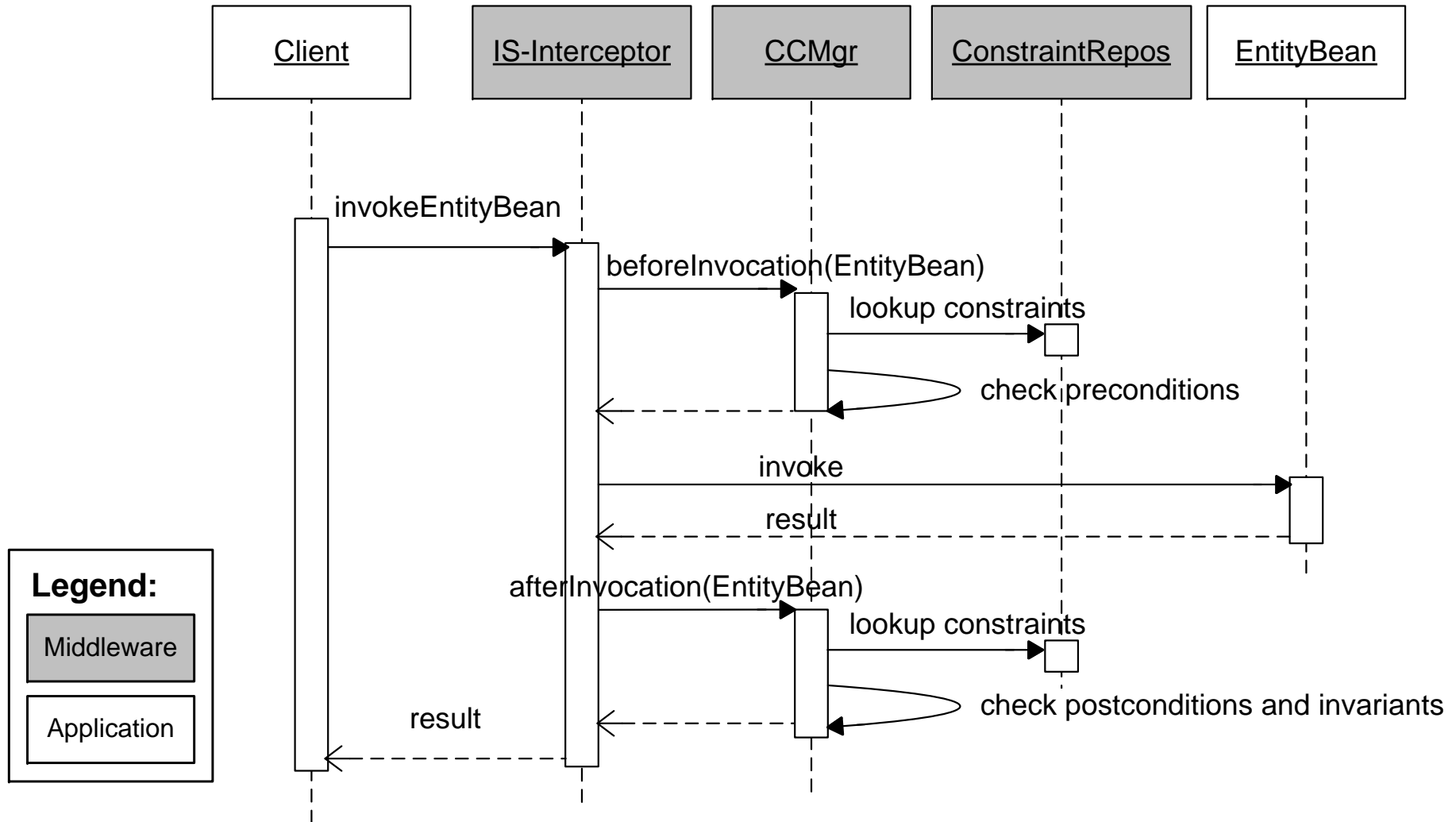
```
    }  
}
```

```
context Flight inv maxTickets:  
seats >= tickets->size()
```

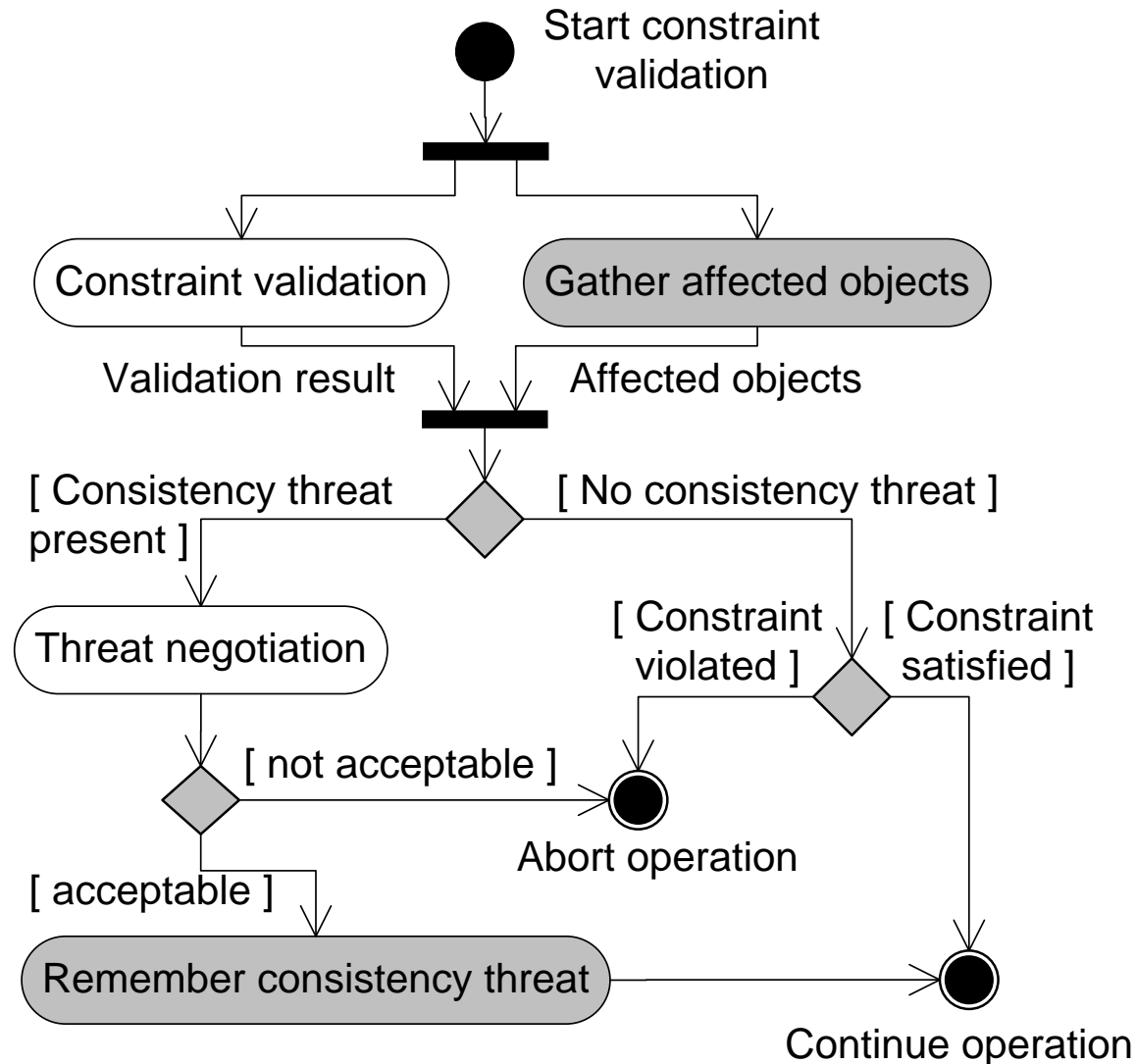
JBoss invocation service



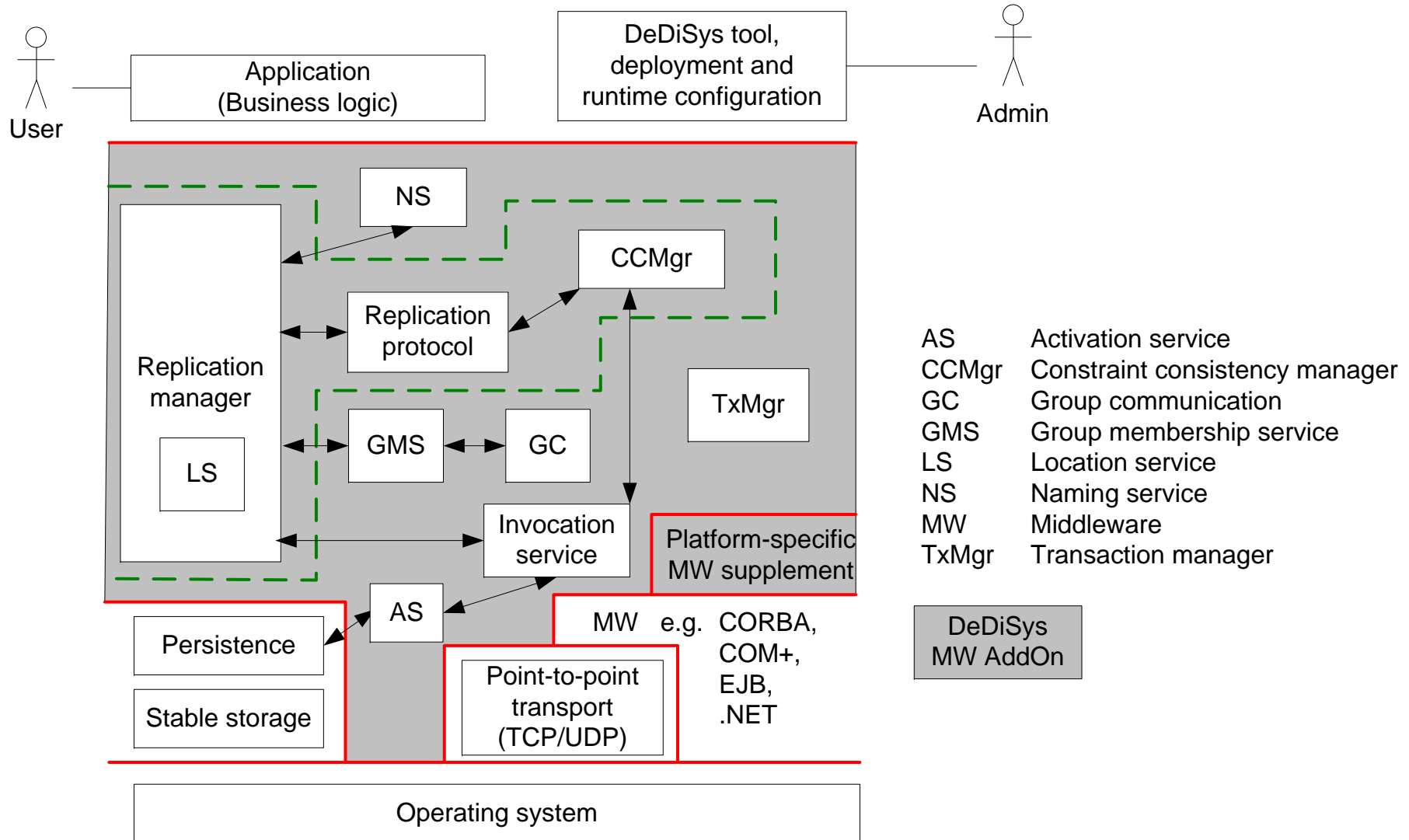
Constraint consistency manager



Negotiation: availability – consistency



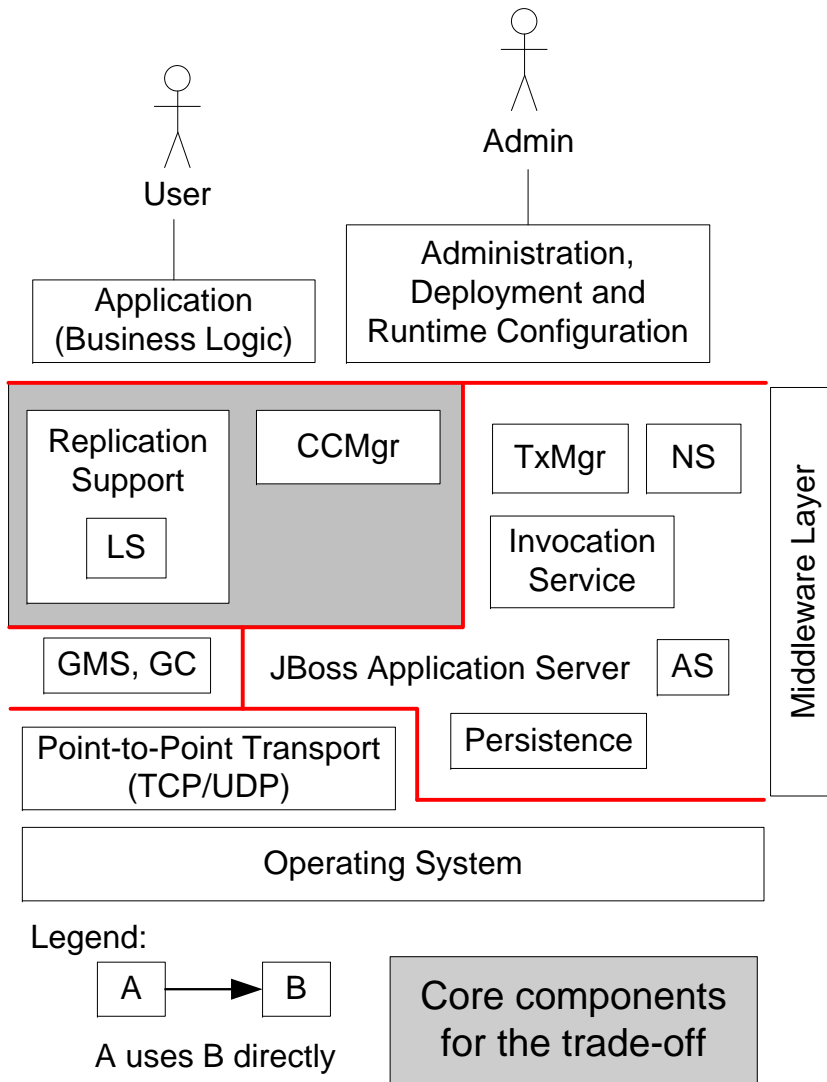
Basic architectural components



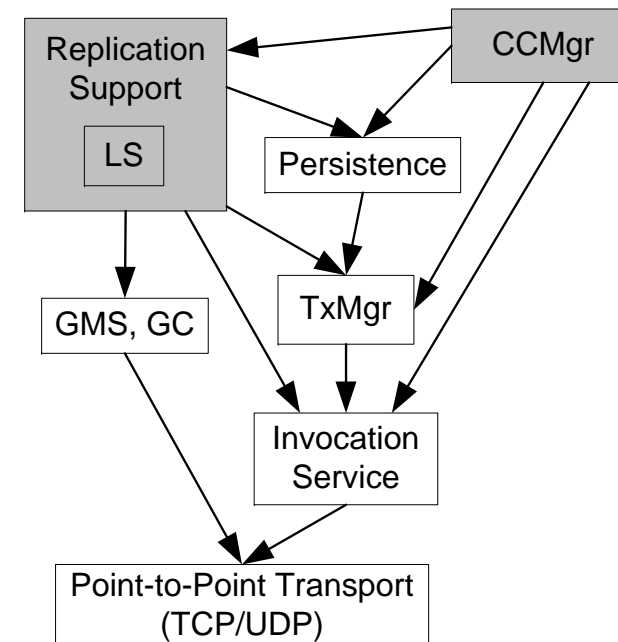
Further details and explanation, see pp. 24-27 of

http://www.dedisys.org/dedisys/images/stories/documents/deliverables/2005-10-07_D2.2.1v1.0_System_Model_FTNS.pdf

Integration into Enterprise JavaBeans



- AS Activation Service
- CCMgr Constraint Consistency Manager
- GC Group Communication
- GMS Group Membership Service
- LS Location Service
- NS Naming Service
- TxMgr Transaction Manager



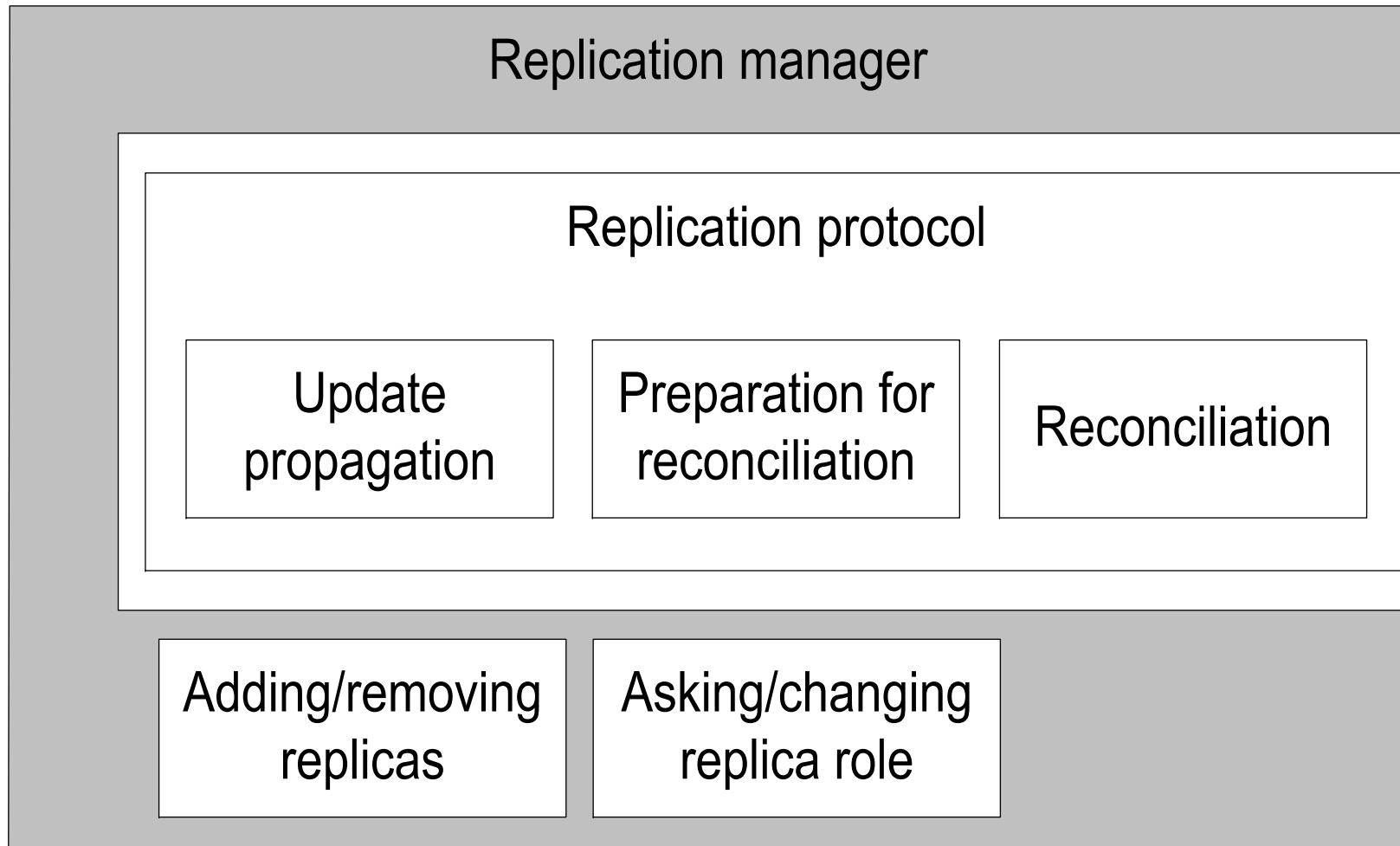
DeDiSys replication model

- ❑ Failure model:
 - *crash* for nodes
 - *link failure model* for communications
- ❑ Group communication system
 - Extended virtual synchrony
 - Partitionable model
- ❑ Passive replication model
 - Requests served by a primary replica
 - Updates sent from primary to its back-ups
 - Default replication protocol: Primary per partition
 - ❑ Once a partition arises, a temporary primary is chosen in all new subgroups
 - ❑ Reconciliation procedure needed when partitions are merged

Replication manager

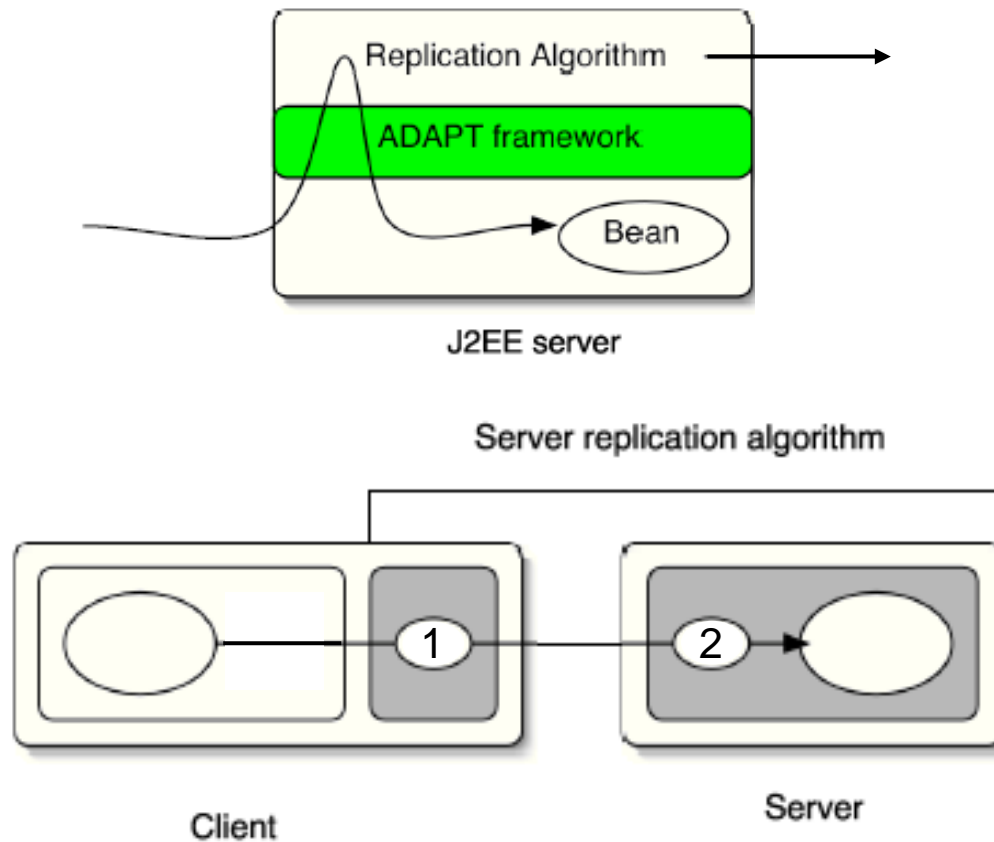
- The replication manager uses two kinds of references:
 - *Logical object reference*
 - Refers to a logical object – it is only a “dummy” object
 - Needed by the replication manager that translates it into a replica reference in order to complete an invocation
 - *Replica reference*
 - Refers to a particular replica
 - The logical object reference has to be translated into a reference of this kind to proceed with the invocation
 - Only visible on the servers
- Translation made by the replication manager when called by the interceptors

Replication system: manager & protocol

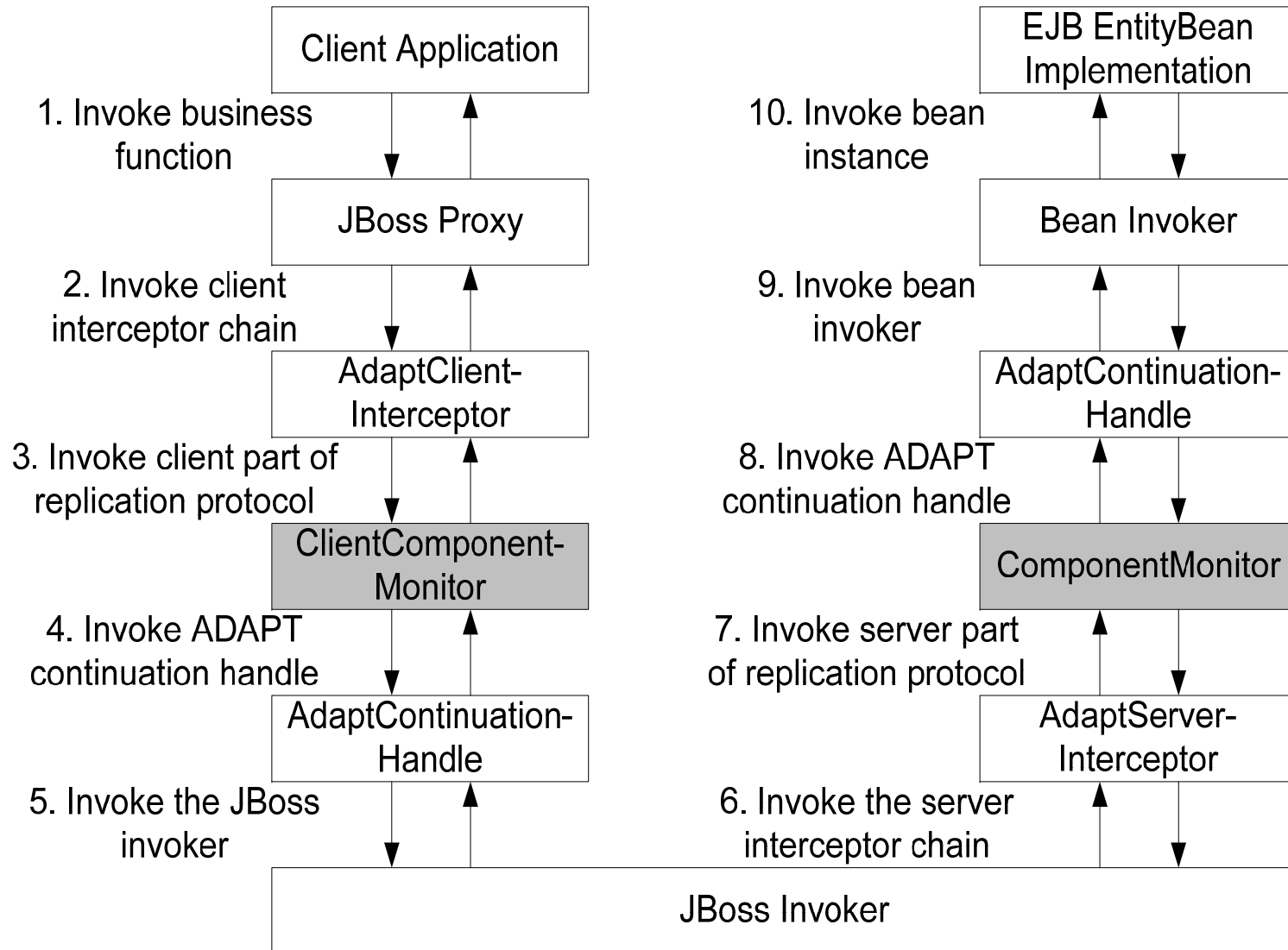


Replication support

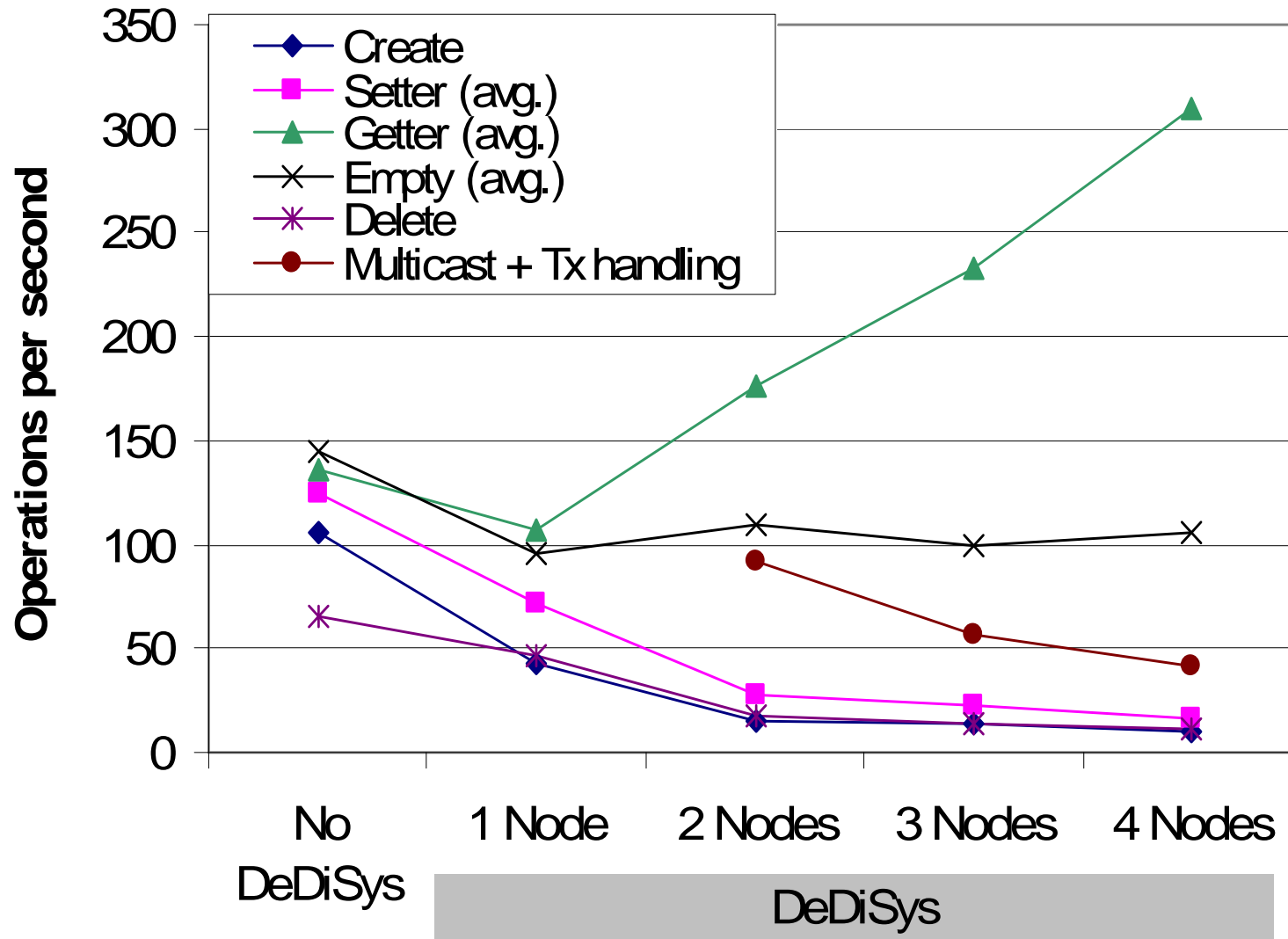
- ❑ Implementation based on the ADAPT framework



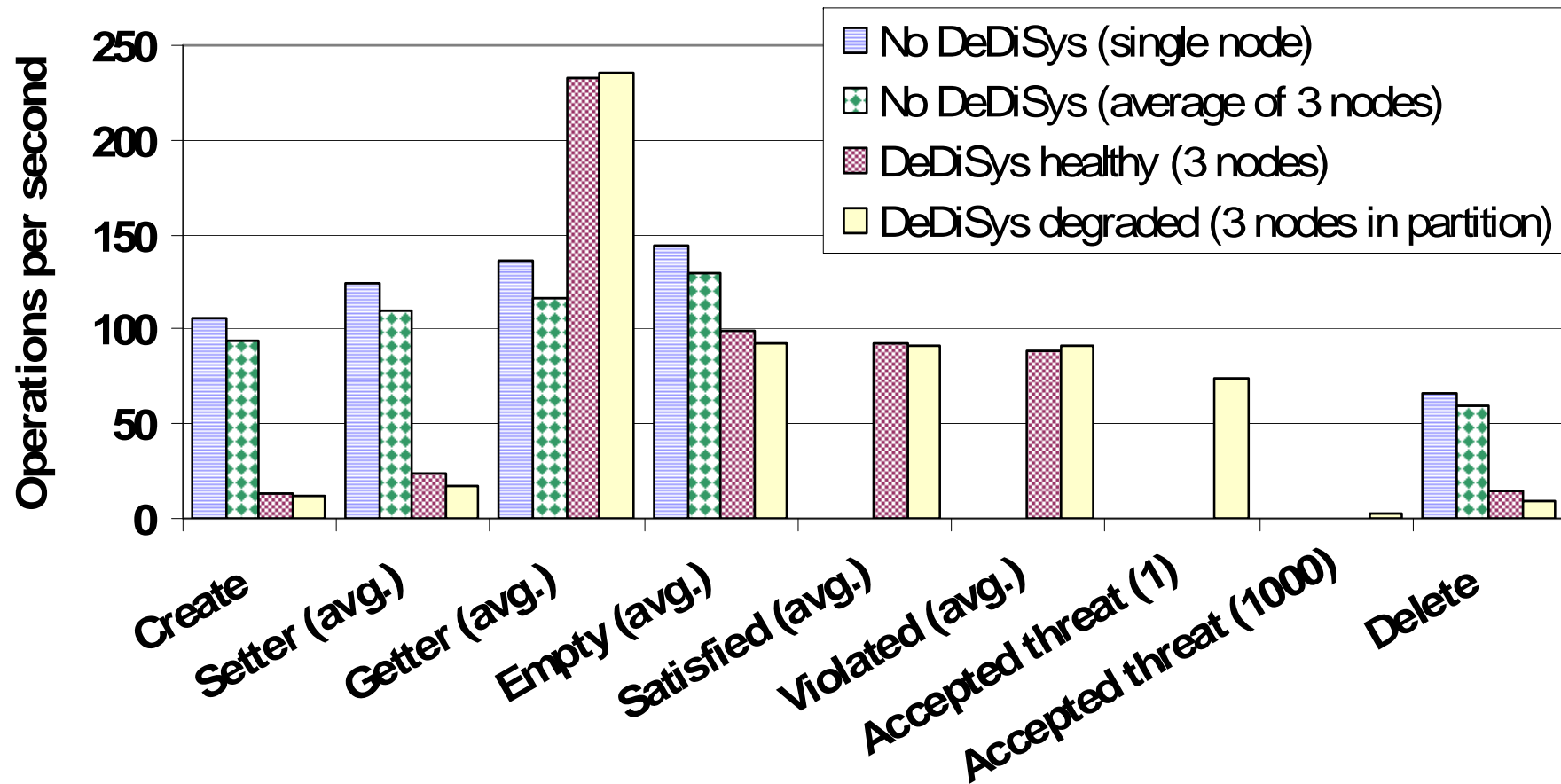
JBoss invocation service and replication



Replication effects on operations



Performance of operations



Thank you for your attention!

Lorenz Froihofer

L.Froihofer@infosys.tuwien.ac.at

Vienna University of Technology