

# JAIN SLEE

# Technology Overview

Version 1.1 – 2005-04-12

Michael Marezke  
michael@maretzke.de

# Goal



---

Understand the value of JAIN SLEE for  
mobile operators' networks  
and how a combination of J2EE and  
JSLEE provides even more value.

---

# Michael Marezke

- Michael works for an international operating mobile network company as Technology Manager.  
He concentrates on service architectures in mobile operator domains. There, Service architectures improve the capabilities of existing networks and increase the possibilities to offer services in a timely and efficient manner.
- Beside the interest in architectures he leads the prototyping development in the german R&D centre and profits from his experience on Java based technologies like J2SE and J2EE and comparable technologies.
- Michael is part of the JSR-22 Expert Group and the JSR-240 Expert Group specifying JAIN SLEE version 1.0 and version 1.1 standards.
- Furthermore, Michael is co-author of „Java Praxisnah“, the book for advanced Java programmers published by JUGM members.



**— What is JAIN?**

**— What is JAIN SLEE?**

**— JAIN SLEE vs. J2EE**

**— Standardisation**

# What is JAIN?

**JAIN** = **J**ava **A**PIs for **I**ntelligent **N**etworks

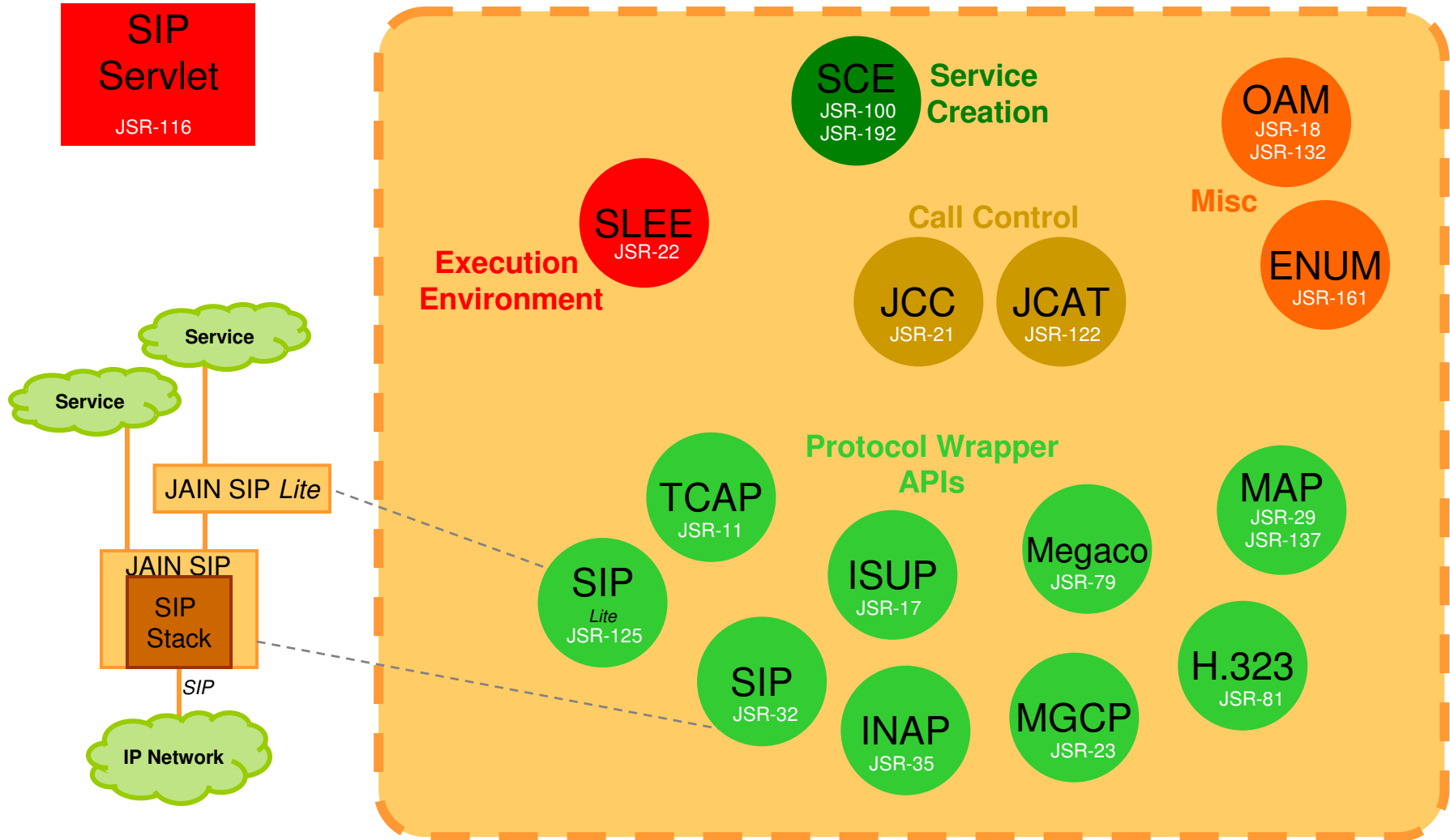
JAIN is an enabling set of Java™ APIs to develop and deploy service driven network applications and services.

Service Portability - *Write Once, Run Anywhere*  
Network Convergence - *Any Network*  
Service Provider Access - *By Anyone*

JAIN is an industry framework designed and specified collaboratively by groups of industry partners and experts.

# JAIN and JAIN SLEE?

## JAIN in context of JCP



— What is JAIN?

— **What is JAIN SLEE?**

— JAIN SLEE vs. J2EE

— Standardisation

# JAIN SLEE – Why?

- \_ Trend towards **component based architectures** – even in telecommunication industries
- \_ Trend towards **open, standardized** off-the-shelf platforms
  - JAIN SLEE is standardised in Java Community Process
  - JAIN SLEE is Java based „**Write once – Run anywhere**“
- \_ JAIN SLEE is a network abstraction layer from application developers point of view and network point of view as well
- \_ JAIN SLEE is designed for **low latency** and **high throughput** environments  
(10-20 calls per second per cpu; ~10 events per call; <200ms RTT)
- \_ **Reduce time to market** and **development cost** by utilizing standards
- \_ Enable **multi-vendor** environments

# JAIN SLEE at a glance



- Based on the **Java** Language
  - Industry standard
  - Huge developer community
  
- Modularised, component and transaction based Execution Environment for service logic components
  
- Provides a framework for ‚portable‘ services
  
- Abstracts underlying infrastructure through utilization of Resource Adaptors
  
- Designed to meet the requirements of the telecommunication market for low latency and high throughput environments
  
- Build in support for high availability and scalability

# What is a SLEE?

— **Low latency** and **high throughput** application server for **event processing**

- Latency < 100 ms
- 100's to 1000's of events per second
- Trial results:
  - 10-20 calls per second per cpu
  - ~10 events per call
  - RTT < 200 ms

— Event optimized component model

— Designed for stringent requirements of core network signaling application

— Distributed component model like EJB

## ***SLEE Management***

MBeans, Service Deployment, Service Mgt, Profile Mgt, ...

## ***SLEE Framework Components***

Event Routing, Profile, Facilities, ...

## ***Resource Adaptors and Resource APIs***

JCC (Call Control), SIP, TCAP, JAIN SPA (Parlay/OSA), ...

## ***Component Model***

Lifecycle, Packaging, Lookup, Events, Invocation Semantics, ...

# Architecture

Management SW

JSLEE

## Management

JMX Agent

Deployment,  
MBeans,  
Management,  
Profile  
Management ...

## Framework

Trace

Alarm

Timer

Profile

Event Router

## Component Model

Lifecycle

Invocation Semantics

Events

Packaging

Deployment Format

Lookup

SBB

SBB

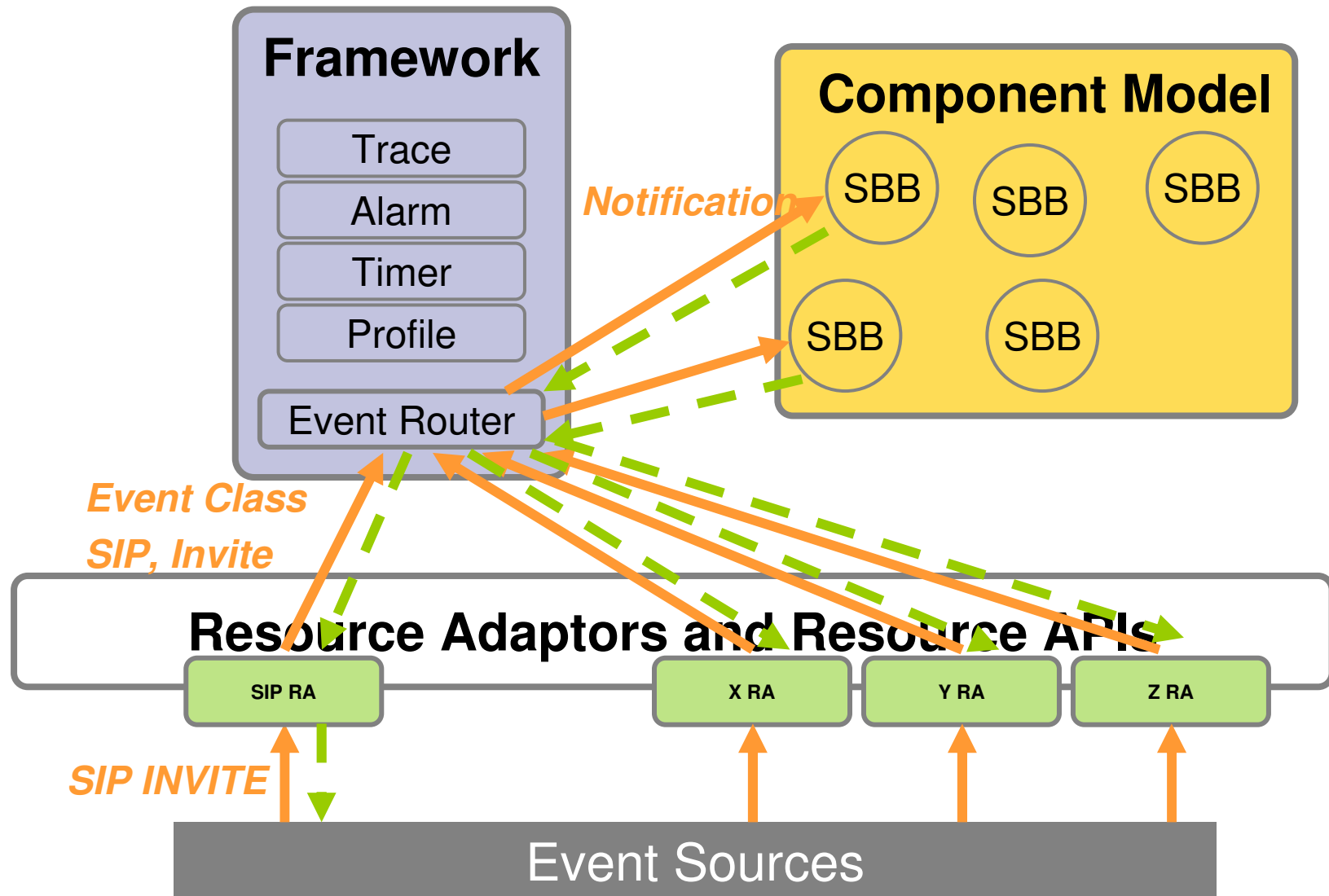
SBB

## Resource Adaptors and Resource APIs

JCC (Call Control), SIP, TCAP, JAIN SPA (Parlay/OSA), ...

Event Sources

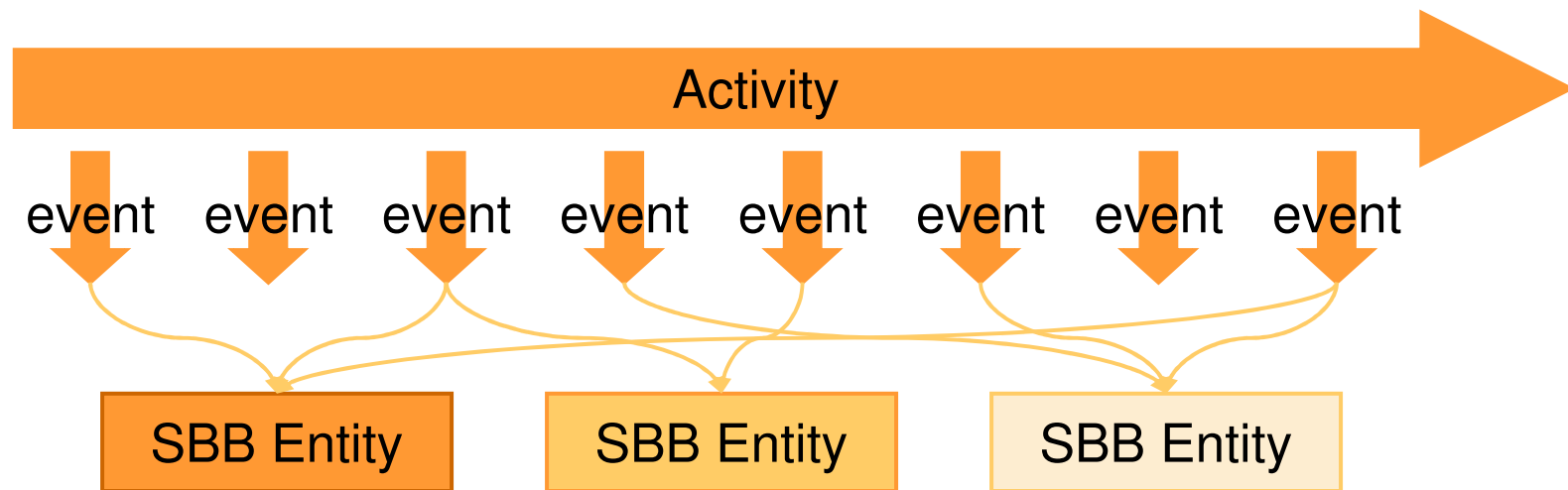
# Network Abstraction - How?



**Activity** – Abstraction for a related sequence of events

Examples:

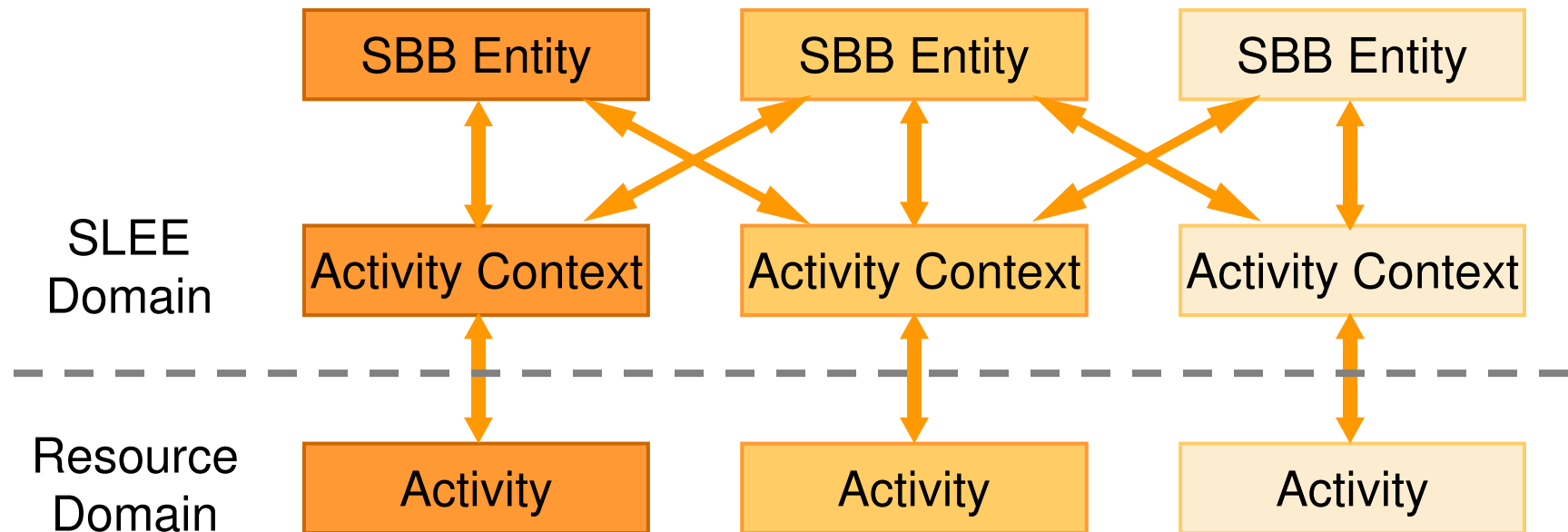
- Call object emitting call establish, terminate, ... events
- Mobile location object emitting location update, ... events



Events are **routed** to SBB entity that are interested in them.  
An SBB entity is an instance of an SBB.

## Activity Context

- A JAIN SLEE defined object that abstracts Activities defined by resources
- An event channel on which events are fired and delivered on
- Keeps shared state regarding the Activity
- Many-to-many attachment relationships
- SBB Entities only receive events from attached Activity Contexts



# High Availability?

## State replication

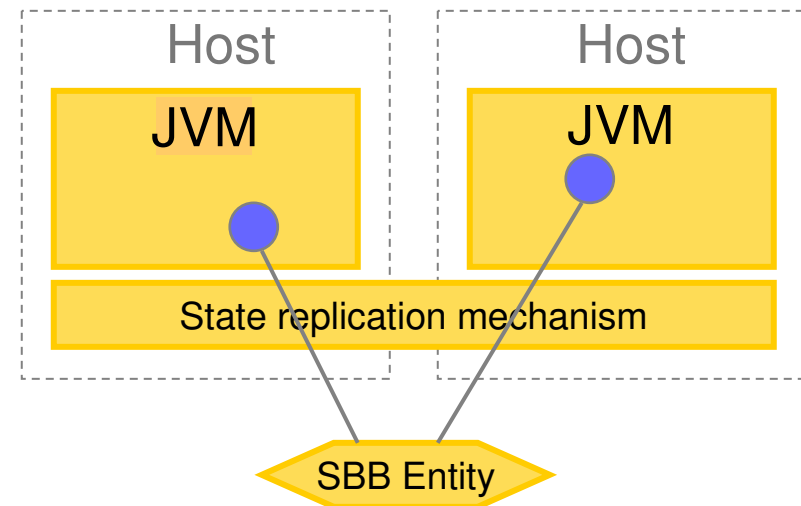
- SBB Entities can restart on other nodes

## Transactional semantics

- Atomicity – All updates complete successfully or none completes
- Isolation – Similar to some sequential execution order
- Automatic – JTA not required

## Exception handling

- Unchecked exceptions
- Transaction rollbacks



— What is JAIN?

— What is JAIN SLEE?

— **JAIN SLEE vs. J2EE**

— Standardisation

# The J2EE World



- \_ Mostly synchronous invocations
- \_ Heavy weight data access objects / components
- \_ Databases as primary information stores
- \_ Transactions follow database transactions
- \_ Database access intensive computation
- \_ No Real-Time behaviour
- \_ Small number of nodes in clusters

**→ For Heavy Business Logic!**

# The JSLEE World



- \_ Mostly asynchronous events, messages, triggers
- \_ Light weight objects with short lifetime
- \_ Multiple data sources
- \_ Fast completing light weight transactions
- \_ State transition intensive computation
- \_ Soft-Real-Time behaviour
- \_ Distributed deployment throughout network

**→ For Low Latency and High Throughput!**

# JSLEE relationship to J2EE



- \_ JSLEE is a component model specialized for event driven applications
- \_ JSLEE builds upon the strong concepts tried and tested in the enterprise domain, namely evident in EJB
- \_ JSLEE is not one of the J2EE platform component models
- \_ JSLEE is designed for low-latency and high-throughput event orientated applications

# JSLEE & EJB – Specs

## **FACT:**

- \_ The JSLEE specification **does not** ensure high performance
- \_ The EJB specification **does not** ensure high performance

*Performance is a characteristic of implementation relevant to a specific network environment, which defines the transaction and persistence layer specific to that environment.*

- J2EE implementations by design exhibit high performance in enterprise networks,
  - yet **do not** meet performance characteristics specified by communication networks.
- JSLEE implementations by design exhibit high performance in communication networks,
  - yet **do not** meet performance characteristics specified by enterprise networks.

**Applications characteristics  
drive Container Design!**

# JSLEE Model Value-add



- \_ Powerful component model for event processing
- \_ Asynchronous support
- \_ Event processing components with strongly typed interfaces
- \_ JSLEE container understands relationship between event producer and event consumer
- \_ Remote reference are forbidden to avoid performance overheads and dangling references

— What is JAIN?

— What is JAIN SLEE?

— JAIN SLEE vs. J2EE

— **Standardisation**

## \_ JSR-22 – Specifying JAIN SLEE v1.0

- Finished since 2004

## \_ JSR-240 – Evolving JAIN SLEE to v1.1

- Public Draft available
- Plan to finish in summer 2005

## \_ Hot topics on JSR-240 list:

- Standardization of Resource Adaptor Architecture
- Internal interfaces Resource Adaptor <-> JAIN SLEE
- High Availability for Resource Adaptors
- Transaction context propagation and Resource Adaptors
- Tight interactions between JAIN SLEE and J2EE

# References

## — More information on JAIN

- <http://java.sun.com/products/jain/>
- <http://jcp.org/en/jsr/tech?listBy=2&listByType=tech>

## — More information on JCP

- <http://jcp.org/>

## — More information on JSR-22

- <http://jcp.org/en/jsr/detail?id=22>
- <http://jcp.org/aboutJava/communityprocess/first/jsr022/index.html>

## — More information on JSR-240

- <http://jcp.org/en/jsr/detail?id=240>
- <http://jcp.org/aboutJava/communityprocess/first/jsr240/index.html>

## — More information on JAIN SLEE

- <http://www.jainslee.org>
- [http://java.sun.com/products/jain/article\\_slee\\_principles.html](http://java.sun.com/products/jain/article_slee_principles.html)
- <http://java.sun.com/products/jain/JAIN-SLEE-Tutorial.pdf>

## — Products?

- [http://java.sun.com/products/jain/certprod\\_table.html](http://java.sun.com/products/jain/certprod_table.html)

## — Open Source Project Mobicents

- <http://www.mobicents.org>