A framework for the design and execution of heterogeneous dynamic distributed applications

Jonathan Bardin

Ph.D Defense,October 2012

Université Joseph Fourier Grenoble 1

Mme Laurence Nigay	Professeur à l' UJF	Présidente
M. Iulian Neamtiu	Assistant Professor at UC Riverside	Rapporteur
M. Philippe Roose	Maître de conférence (HDR) à l'Univ. de Pau	Rapporteur
M. Ye-Qiong Song	Professeur à l'Université de Lorraine / INPL	Examinateur
M. Philippe Lalanda	Professeur à l'UJF	Directeur
M. Clément Escoffier	Arrow Group	Co-Directeur

Outline

Context & Challenges

State of the Art

RoSe

Conception and Implementation Overview

Validation & Evaluation

Conclusion

Outline

Context & Challenges

State of the Art

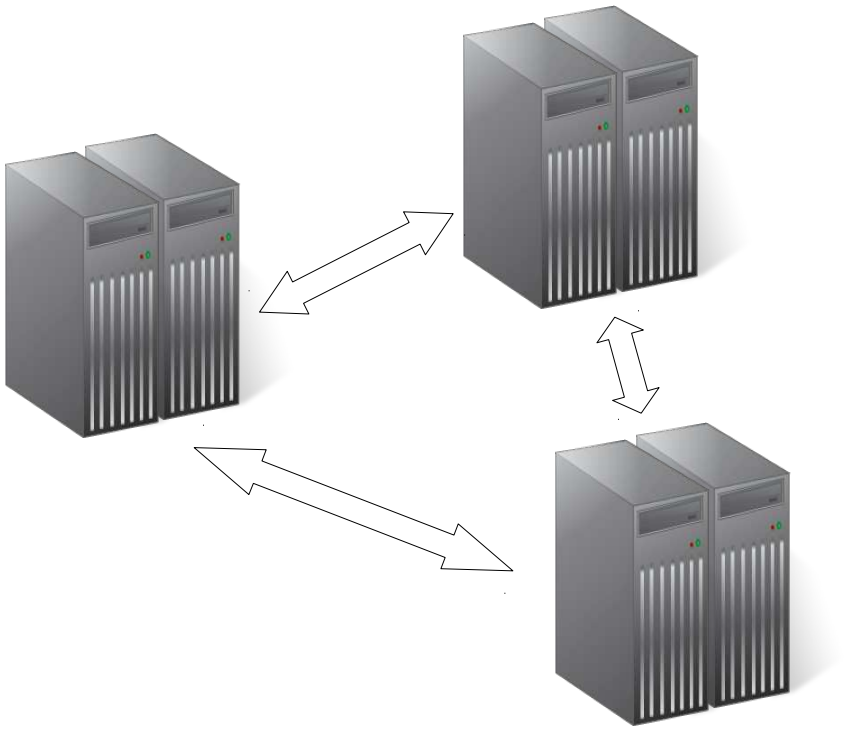
RoSe

Conception and Implementation Overview

Validation & Evaluation

Conclusion

Networked Applications



Networked Applications
Nowadays

Network based

Heterogeneous systems

Autonomous machines

2.7 billion

Internet users worldwide

Ubiquitous

+

Web

+

Dynamic Resources Allocation

Challenges

Distribution

Scalability

Usability

Challenges

Reliability

Heterogeneity

Dynamism

Design and execution
of
heterogeneous dynamic distributed
applications ?

Software
architecture

Dynamic
systems

Outline

Context & Challenges

State of the Art

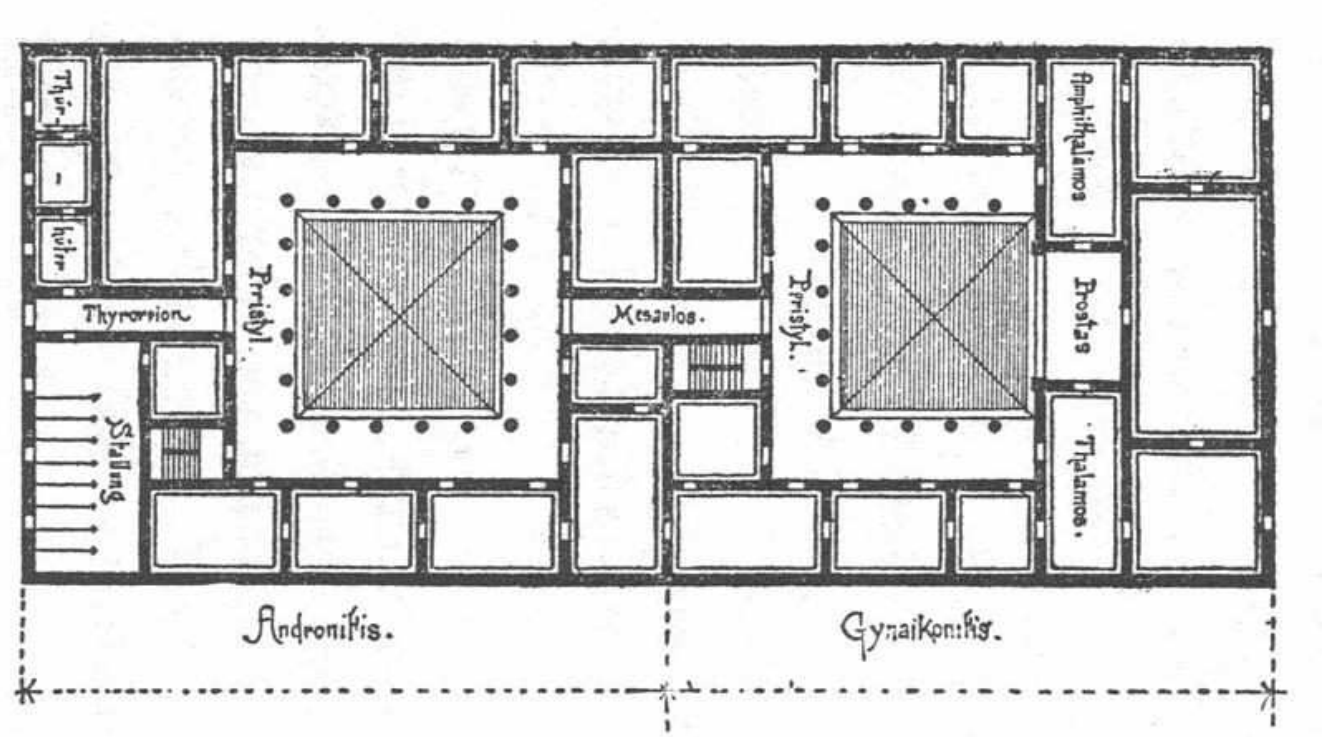
RoSe

Conception and Implementation Overview

Validation & Evaluation

Conclusion

Software architecture



"Abstractly, software architecture involves the description of elements from which systems are built, interactions among those elements, patterns that guide their composition, and constraints on these patterns."

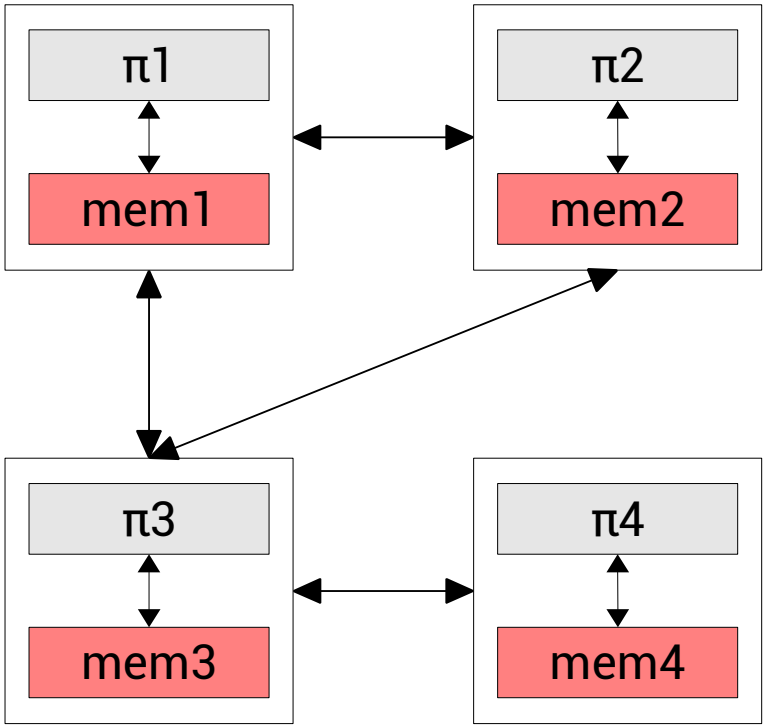
M. Shaw and D. Garlan
(1996)

Architecture Description Languages

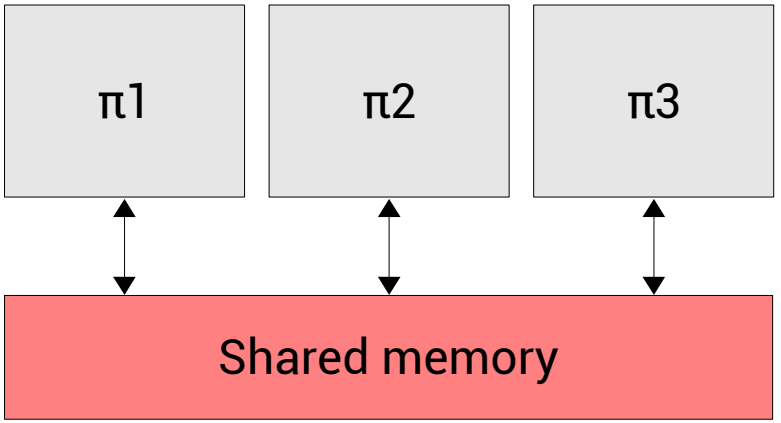
Architectural Styles

Design Patterns

Distributed software architecture

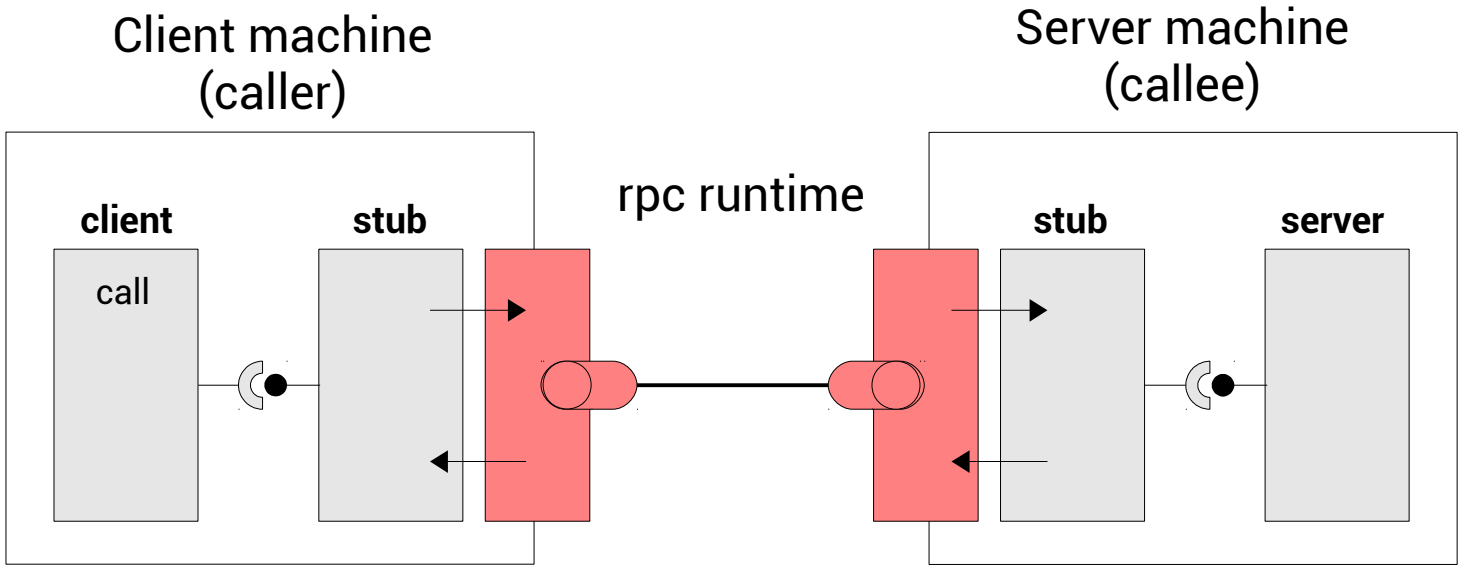


Loosely coupled

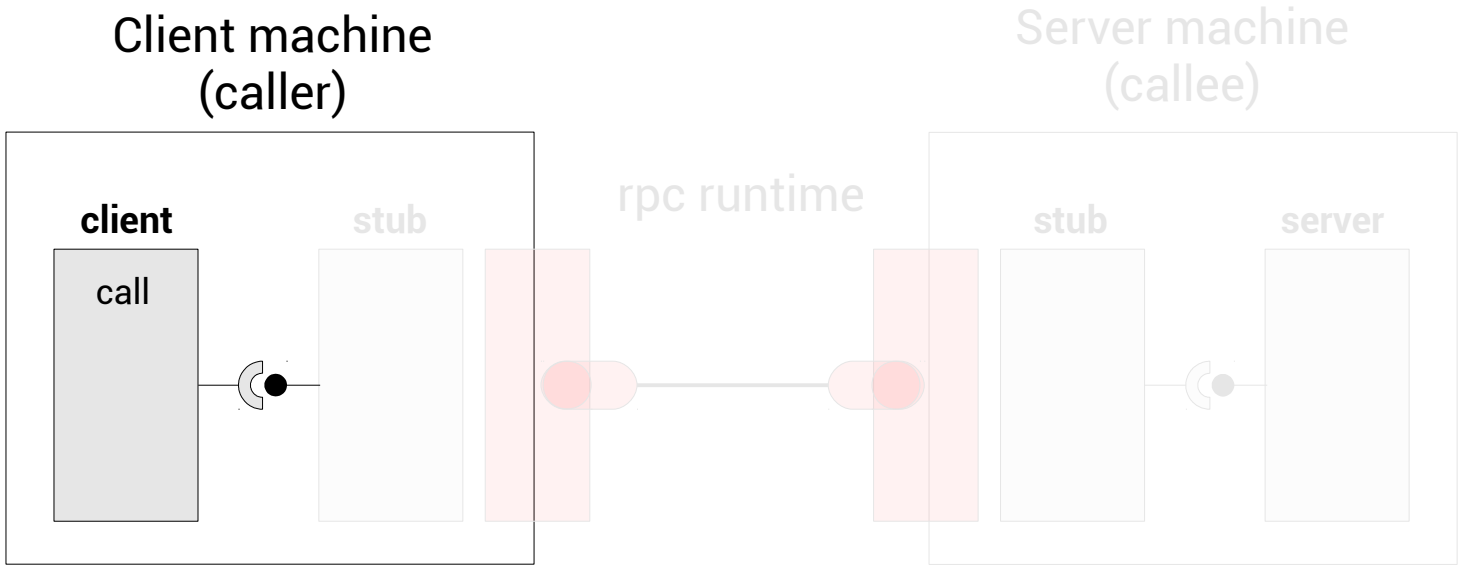


Tightly coupled (parallel systems)

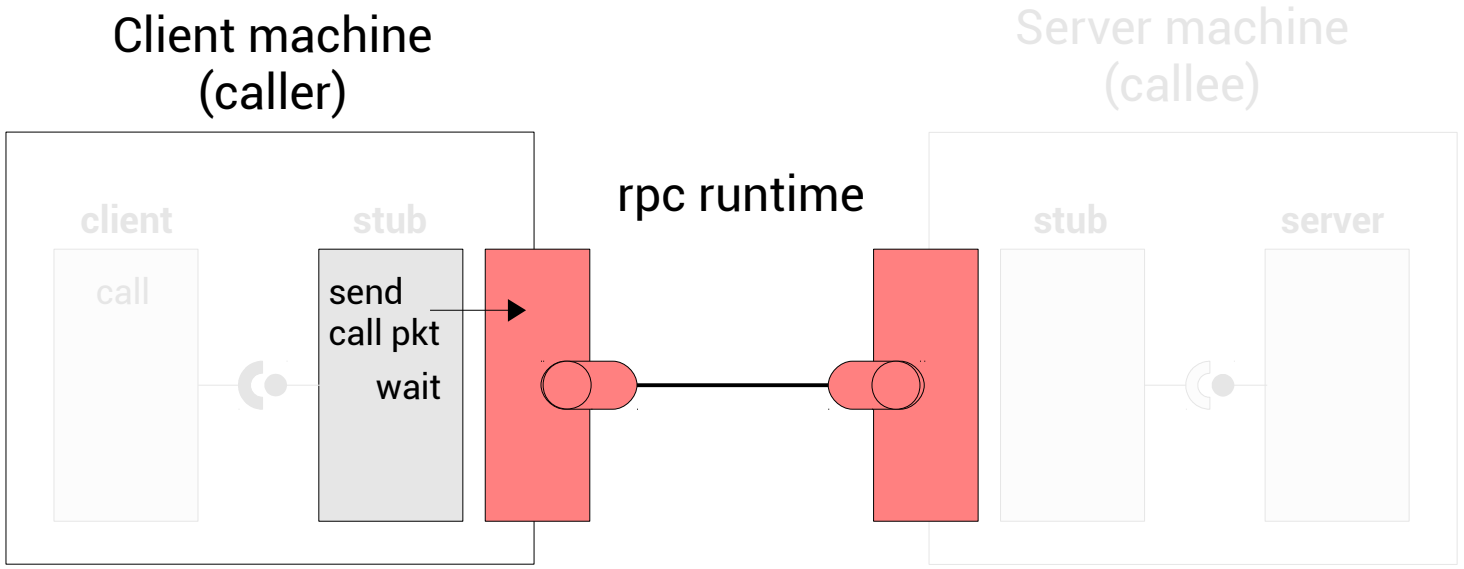
Remote Procedure Call



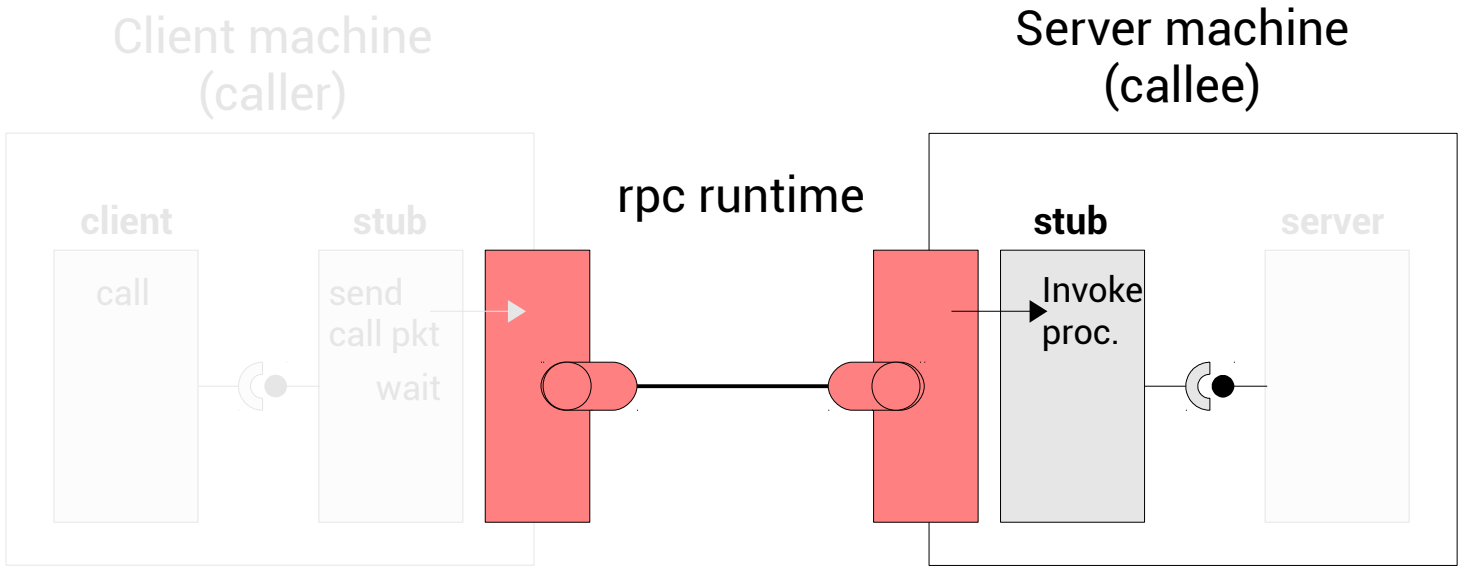
Remote Procedure Call



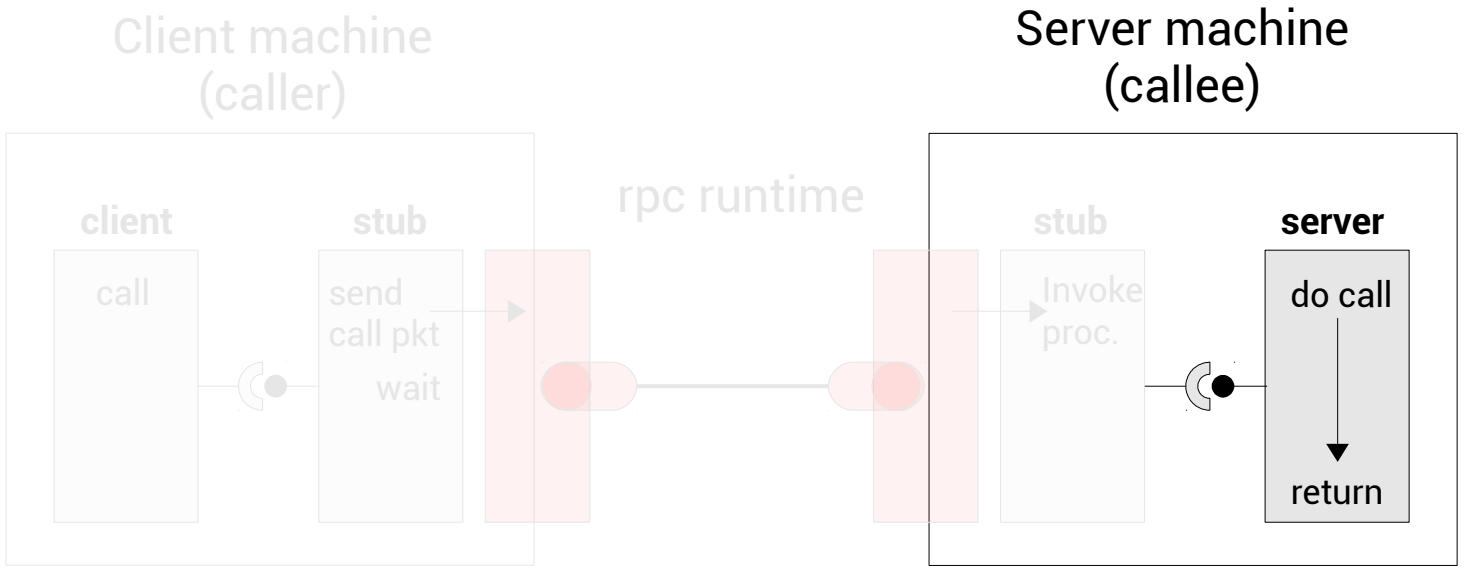
Remote Procedure Call



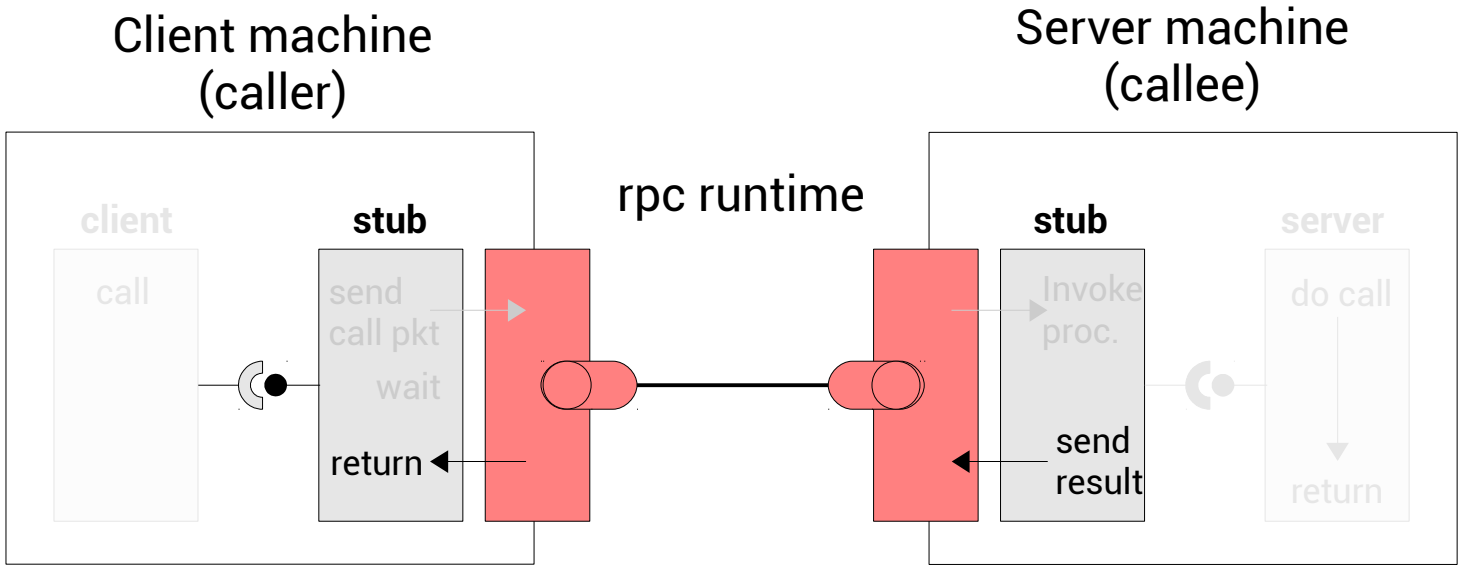
Remote Procedure Call



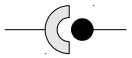
Remote Procedure Call



Remote Procedure Call



Remote Procedure Call



Interface description

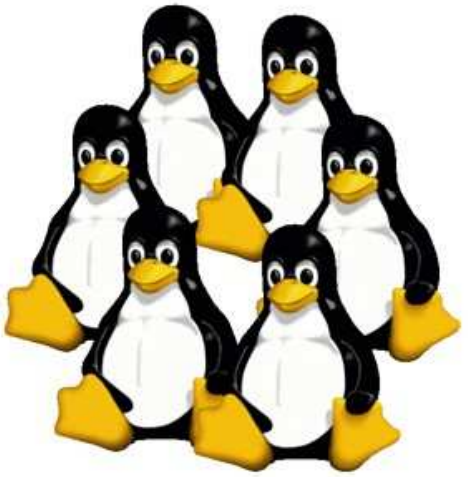
Discovery ?

Transparency

Remote Procedure Call

AndrewFS
NFS

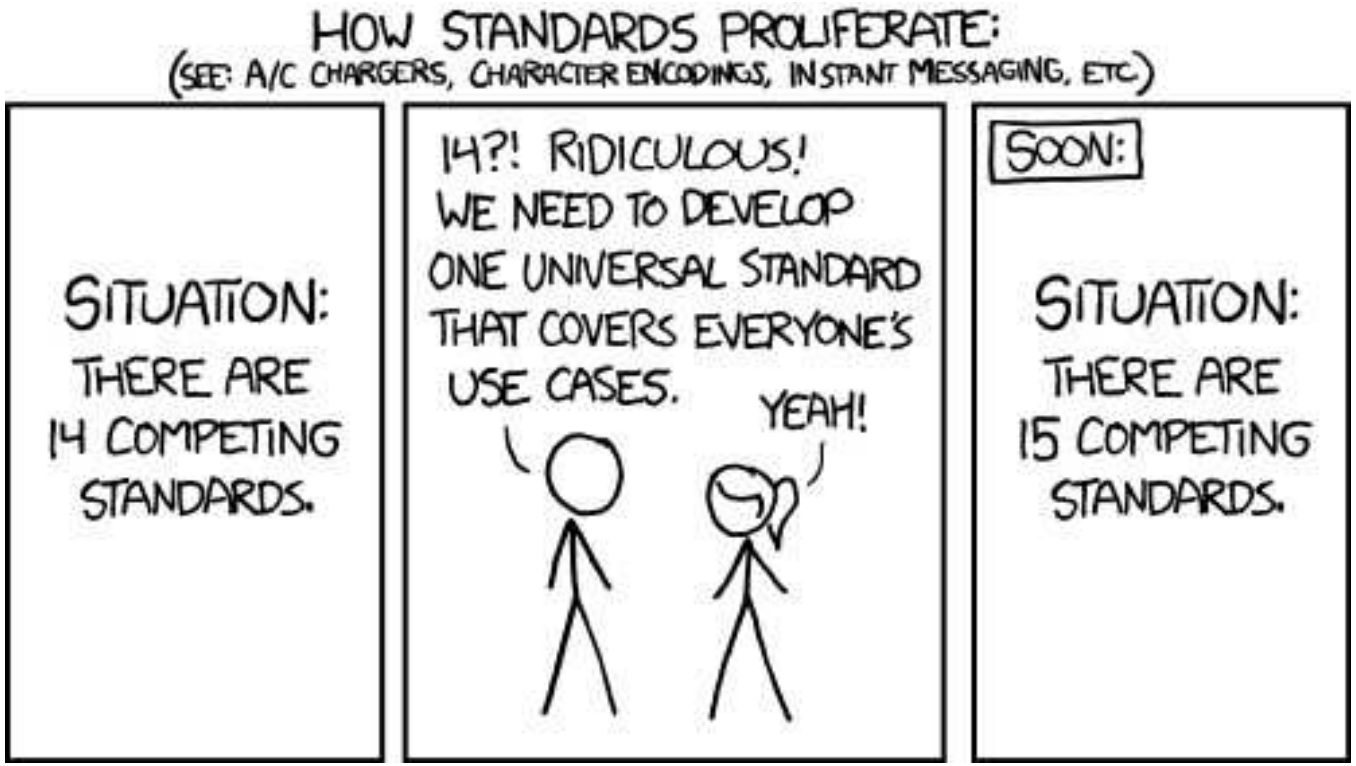
Ninf-G
NetSolve



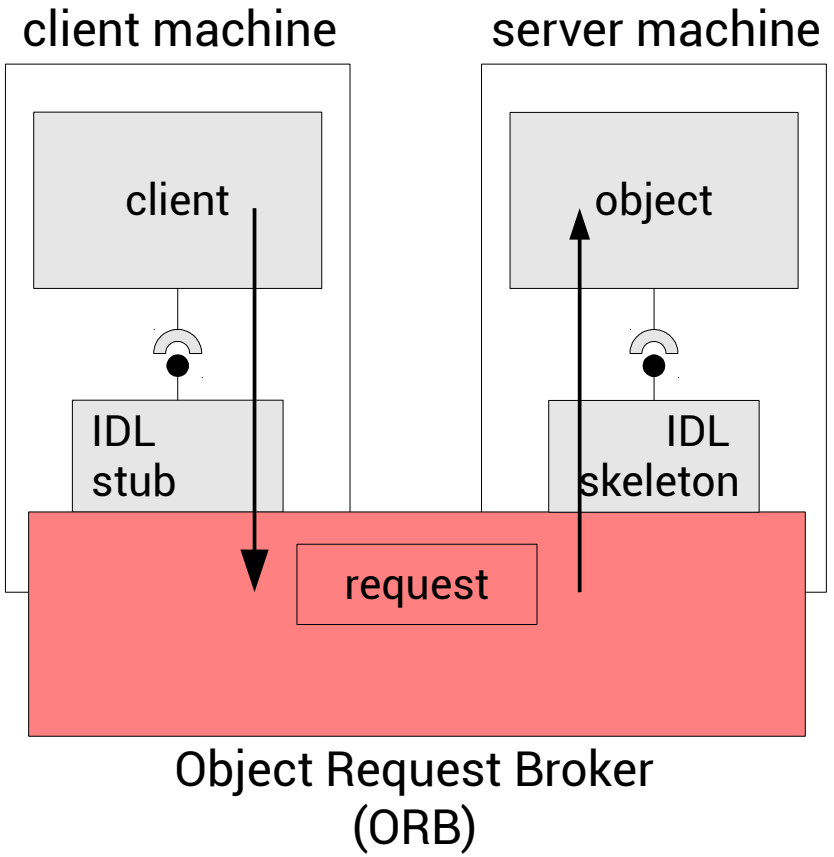
XML-RPC
JSON-RPC

Android IPC

RPC Standard



CORBA

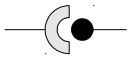

client machine
server machine
client
object
IDL
stub
IDL
skeleton
request
Object Request Broker
(ORB)

Interface Definition Language

```
1  module HelloWorldIDL
2  {
3      interface HelloWorld
4      {
5          string sayHello();
6      };
7  };
```

mapping { C++ Java (98)

CORBA



IDL

ORB + TORBA

Implementation dependent

CORBA

WEB divergence

Complex architecture

Versioning

Security

REST

HTTP

Resources centric

Resource is information

1 resource = 1 global ID

REST Constraints

Client-Server

Stateless

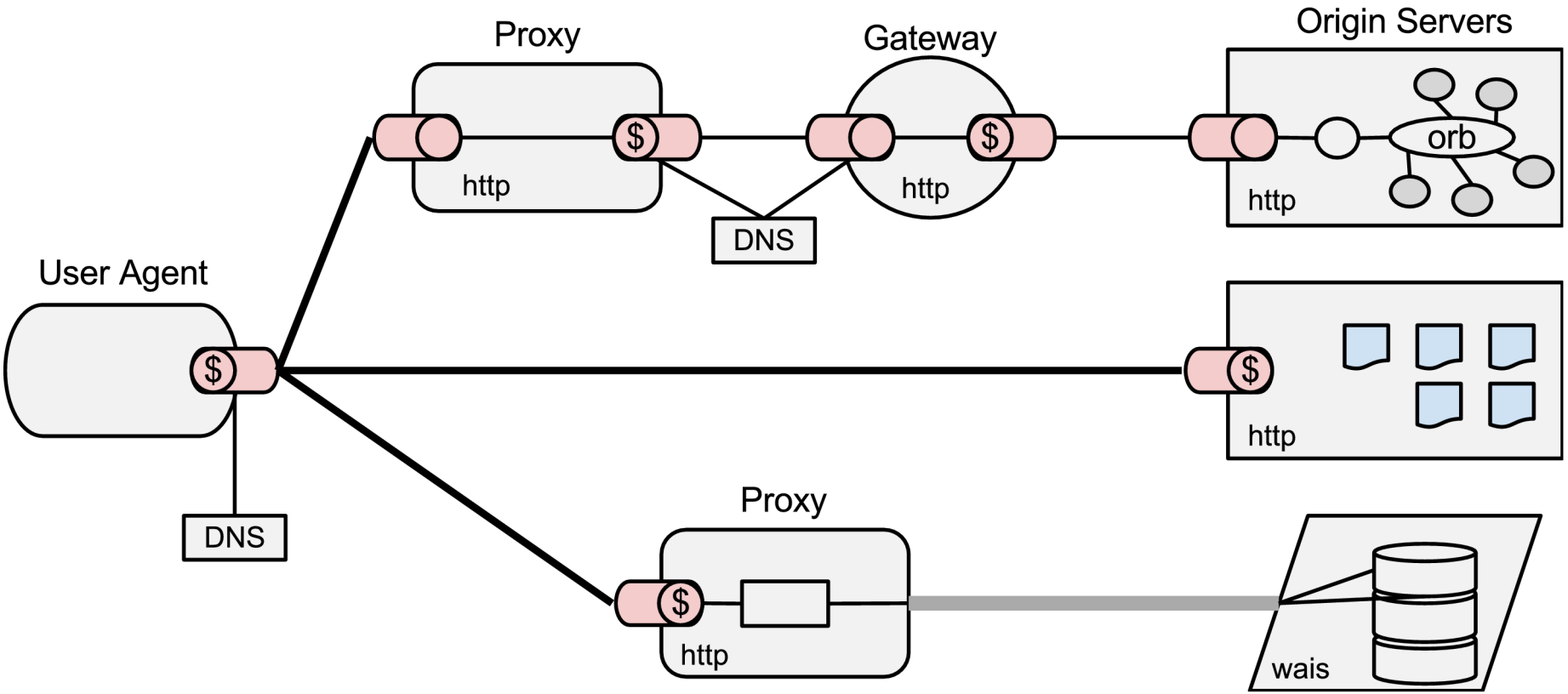
Cacheable

Layered system

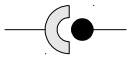
Code on demand

Uniform interface

REST Application



REST



Data based

DNS, HATHEOS

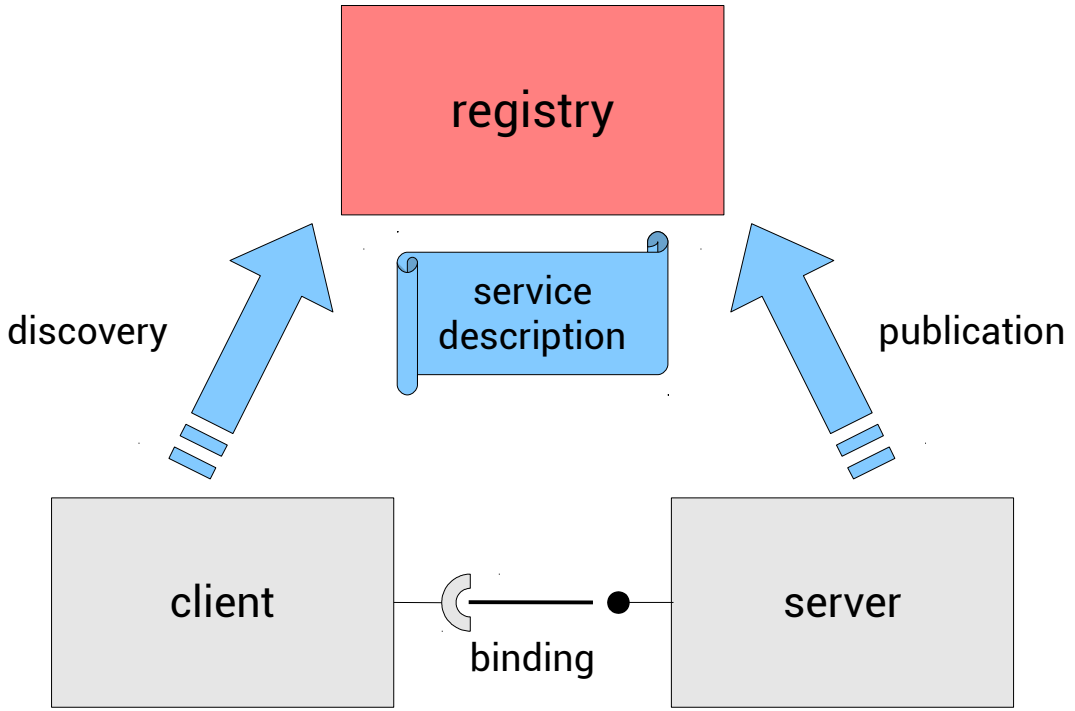
No transparency

Service Oriented Architecture

"Services are self-describing, platform-agnostic computational elements that support rapid, low-cost composition of distributed application."

M. Papazoglou

SOA Interaction

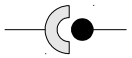


SOA Principles

Loosely coupled

Localization transparency

SOA



Interfaces (XML, Java...)

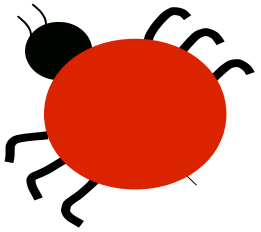
Implementation dependent registries (UDDI, OSGi...)

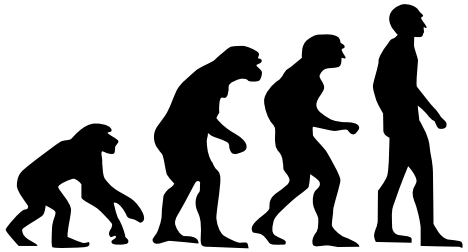
No transparency

How to deal with dynamism?

Adaptation in software

Bugs fix





Evolution

Change in the environment

Dynamic adaptation

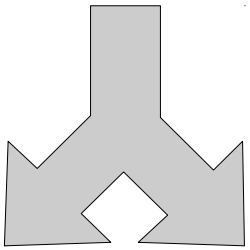
In operating systems

Hot software installation and update

Still requires some reboots

In programing languages

Dynamic linking



Indirection code rewriting

Dynamic loading in C

```
1   void *handle;
2   int *iptr, (*fptr)(int);
3
4   handle = dlopen("/usr/home/me/my_lib.so", ...);
5
6   *(void **)(&fptr) = dlsym(handle, "my_function");
7   iptr = (int *)dlsym(handle, "my_data");
8
9   *(*fptr)(*iptr);
10
11  dlclose(handle);
```

Dynamic loading in C

No verification

No scope

Dangling pointers

Dynamic loading in JAVA

```
1   ClassLoader loader;
2   [..]
3   Class klass = loader.loadClass(name);
4   Object obj = type.newInstance();
```

Dynamic loading in JAVA

Unloading

Versioning

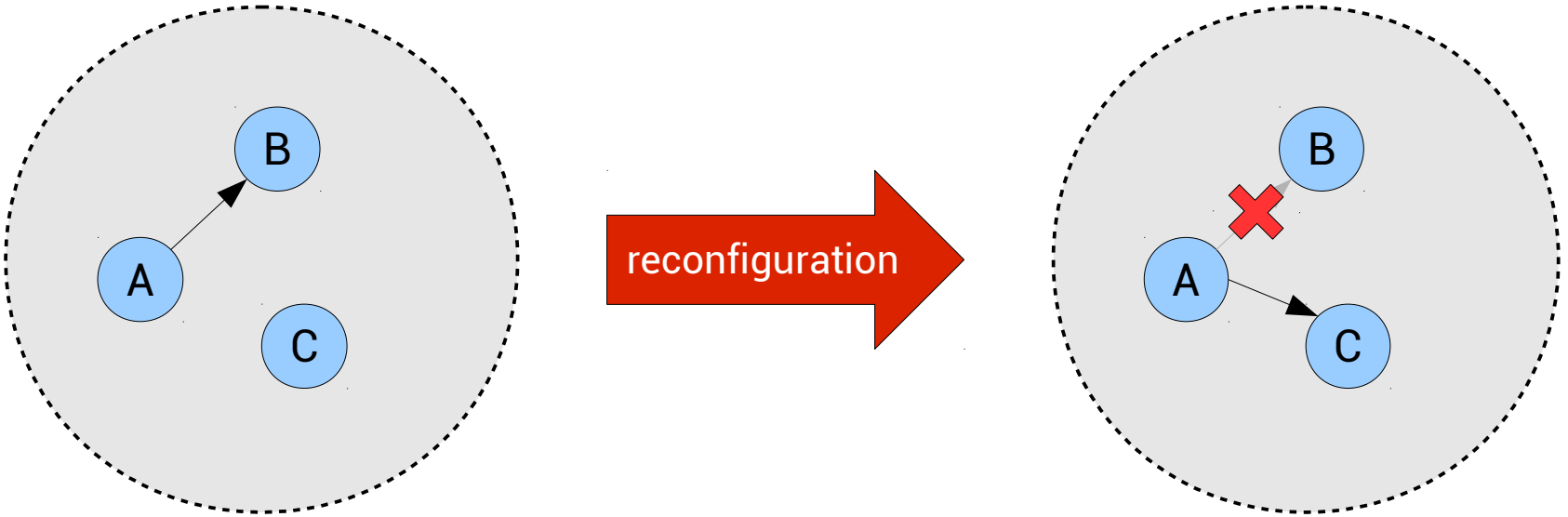
Classpath hell

Architectural adaptation

"A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties."

C. Szyperski

Architectural reconfiguration



Component

Binding

Mixing component and service

Service Oriented Component

A service is provided functionality

A service is characterized by a contract

Components implement a contract

Service Oriented Component

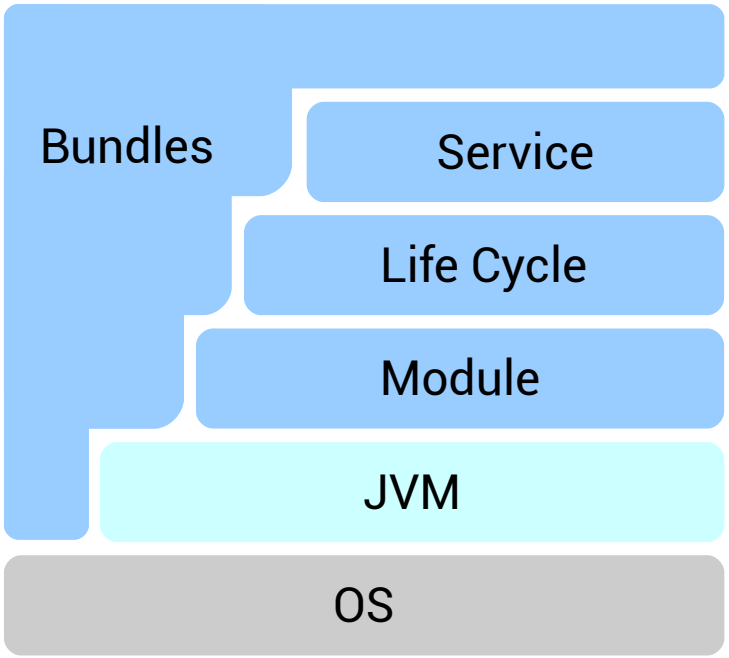
The service-oriented interaction pattern is used to resolved dependency

Compositions are described in terms of contracts

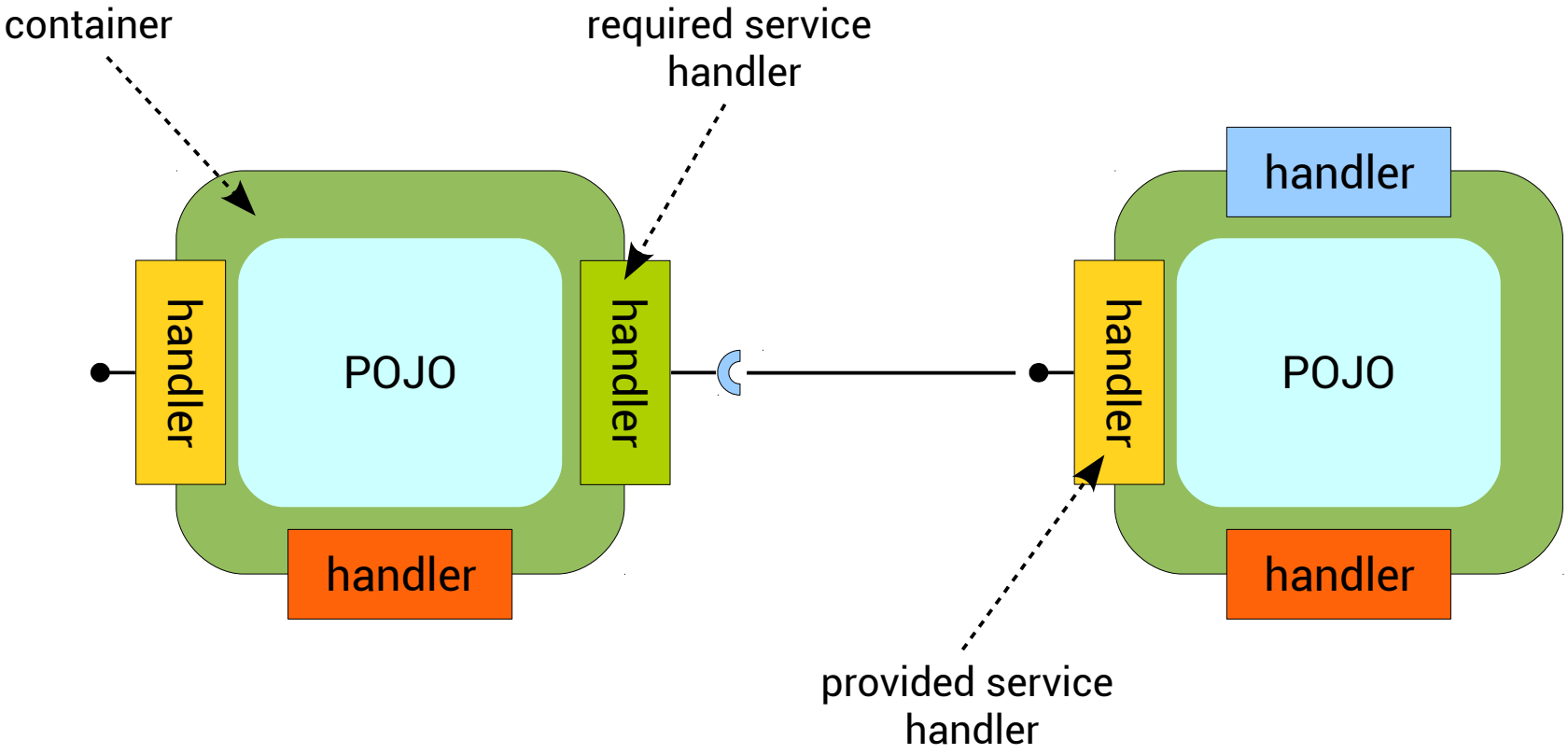
Contracts are the basis for substitutability

OSGi

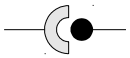
A module system & dynamic service platform for Java.



iPOJO


container
required service handler
handler
POJO
handler
handler
handler
POJO
handler
provided service handler

No Silver Bullet



Several formats

No discovery, active discovery and passive discovery

Different levels of transparency

Integration

Outline

Context & Challenges

State of the Art

RoSe

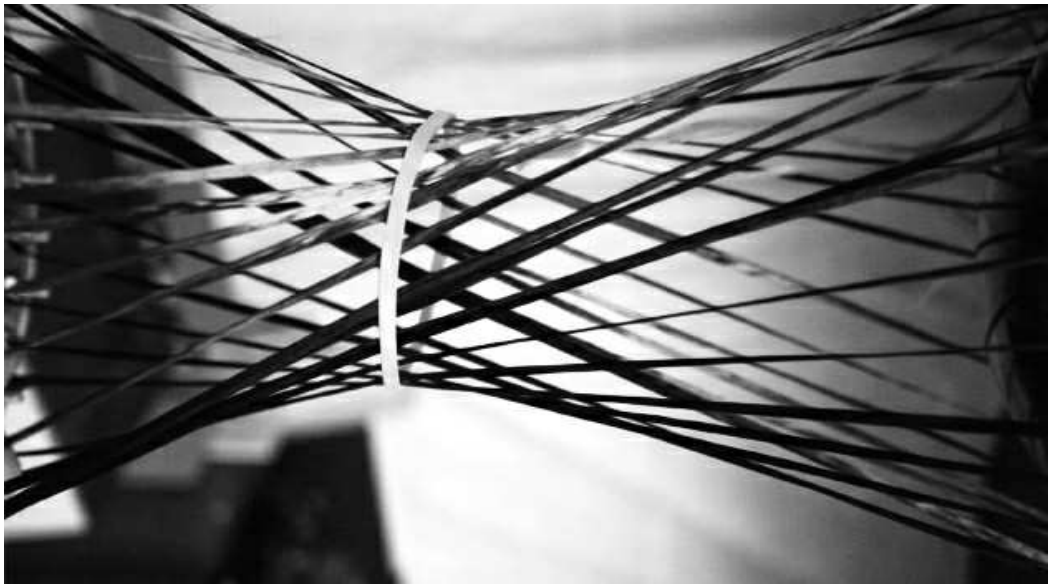
Conception and Implementation Overview

Validation & Evaluation

Conclusion

How to ease
the design and execution of
heterogeneous dynamic distributed
applications ?

Software framework



"A framework is a reusable design of an application or subsystem represented as a set of abstract classes and the way their instances collaborate. […]"

Joseph W. Yoder

The RoSe Framework

7 Principles

Transparent distribution

Separation of concerns

Automation

Flexibility

Anarchic scalability

Simplicity

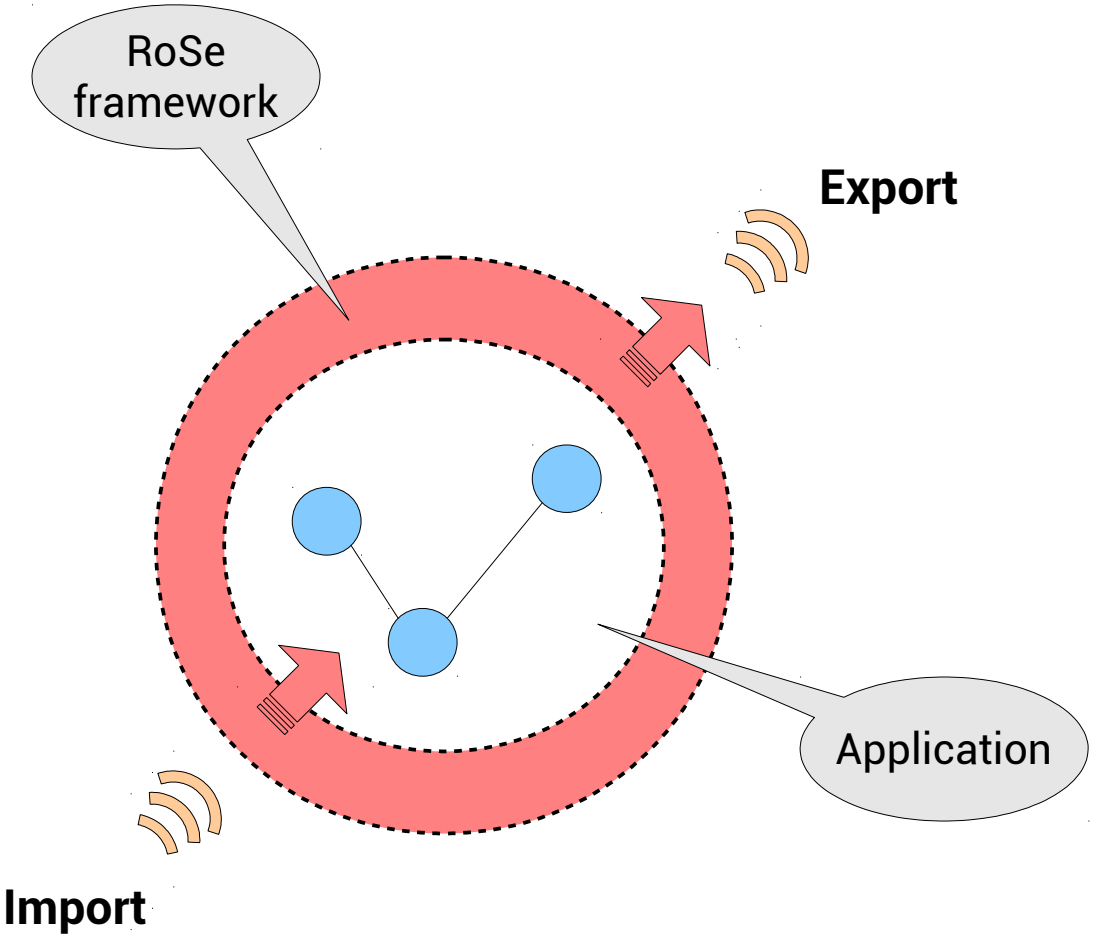
Diversity

Concepts

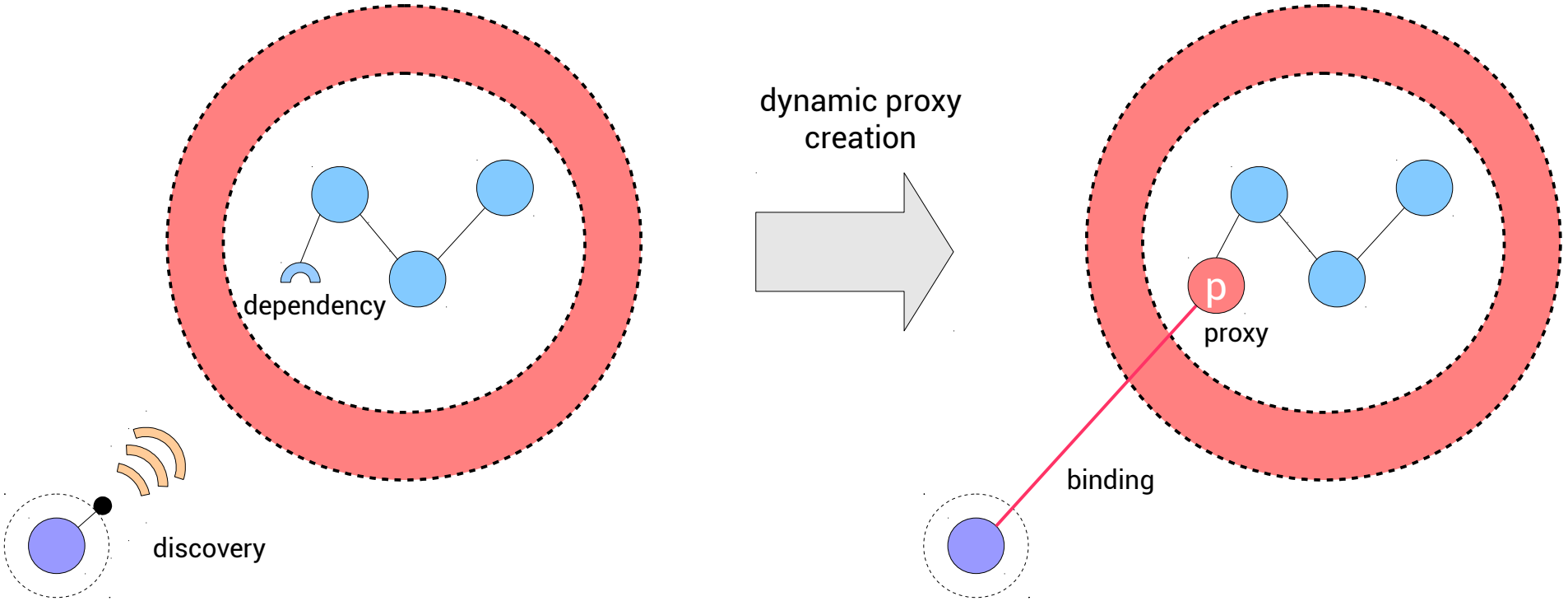
Applications are based on service oriented component models

Dynamic import and export of services according to a specification

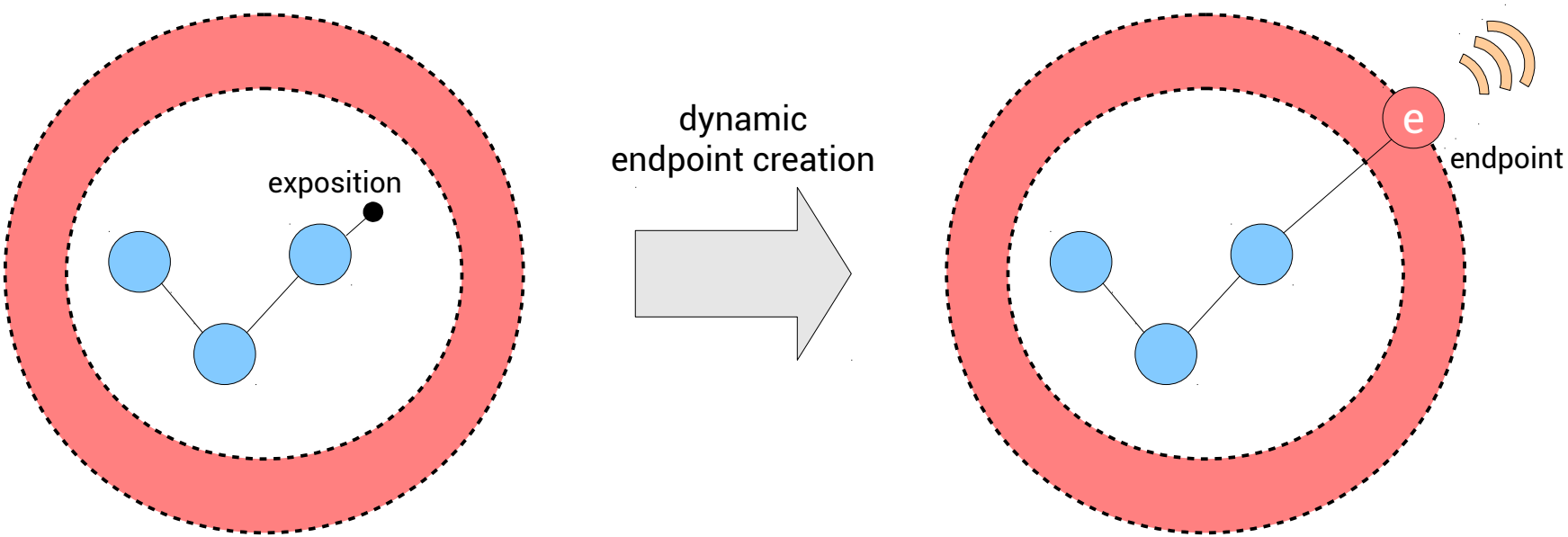
Each remote service has an immutable description


RoSe
framework
Export
Application
Import

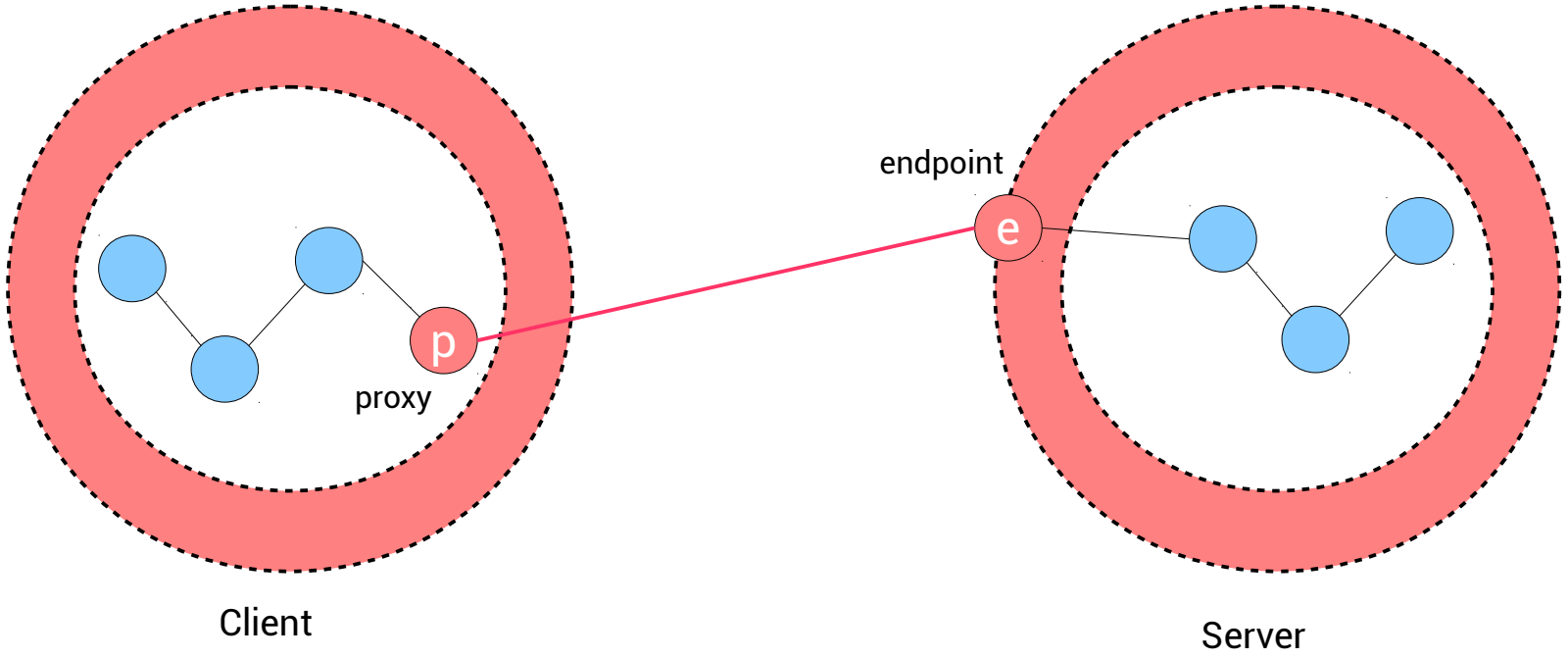
Import


dependency
discovery
dynamic proxy
creation
p
proxy
binding

Export

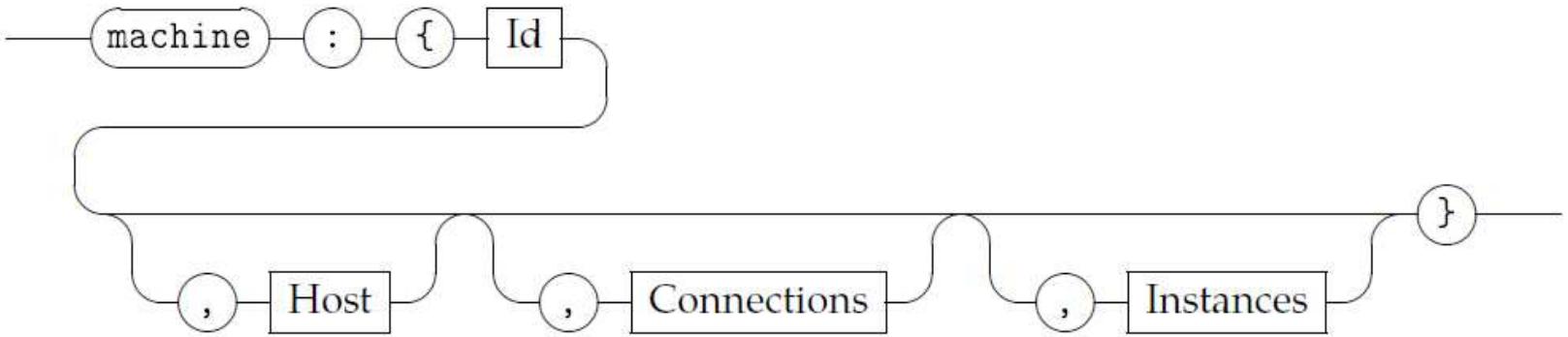


Transparent distribution

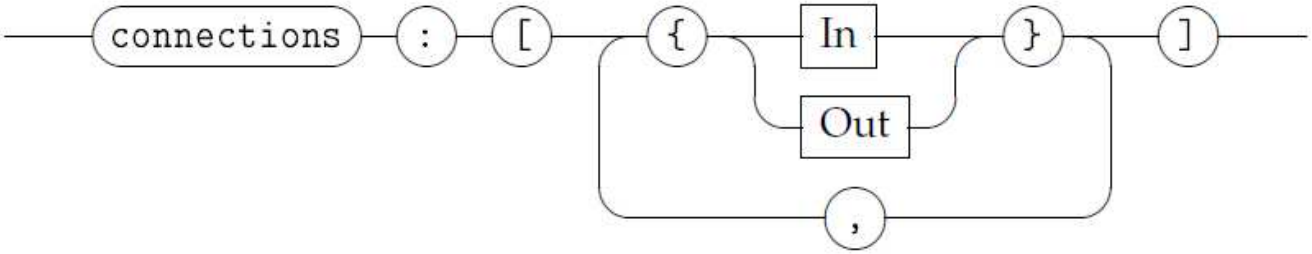


RoSe language

Machine

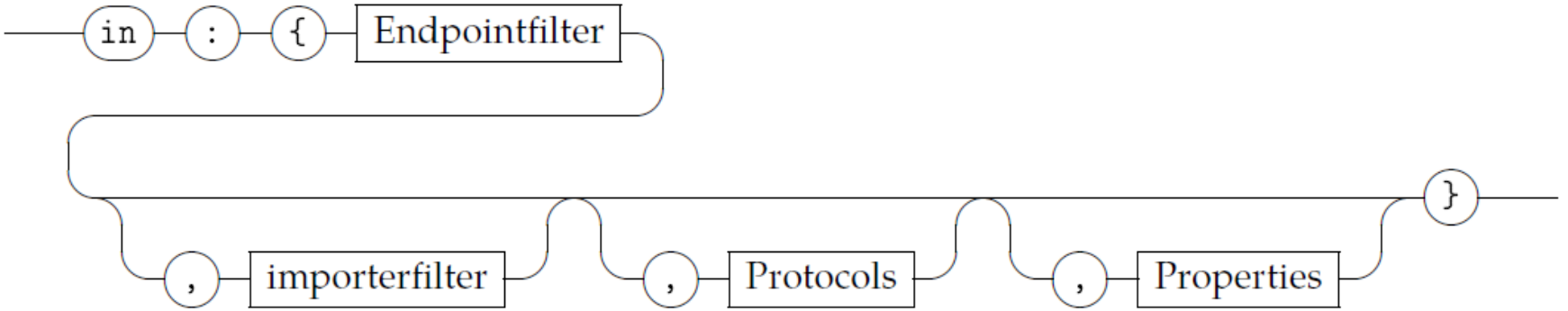


Connections

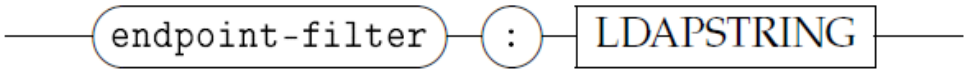


Import

In

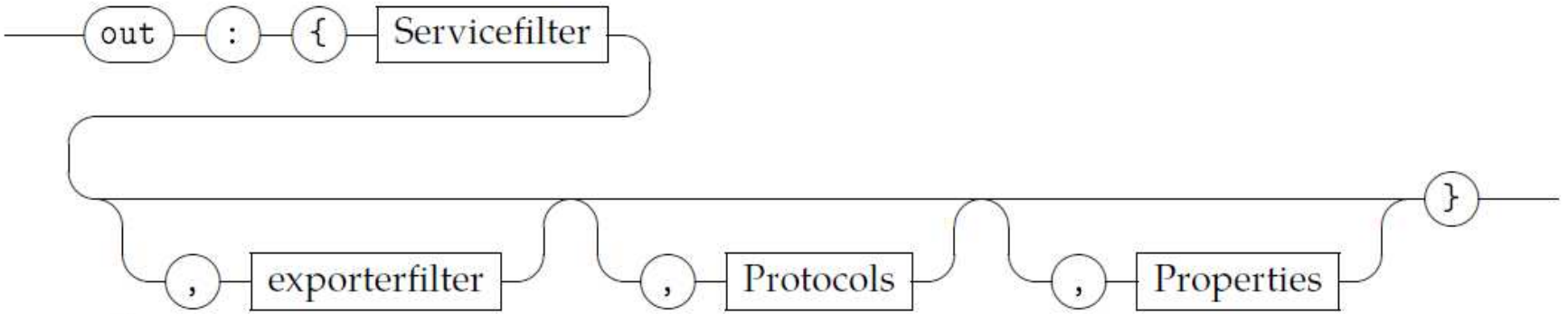


Endpointfilter

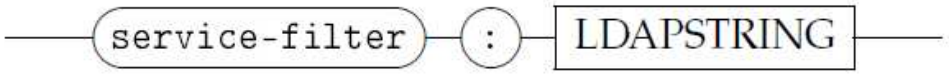


Export

Out



Servicefilter



Example

```
{
 machine : {
  id : "server1",
  host : "localhost",
  connections : [
   { out : {
    service_filter : "(objectClass=toto.contract.HelloWorld)",
    properties : { "tag" : [ "example", "helloworld"] }
   }
  }]
}
```

Outline

Context & Challenges

State of the Art

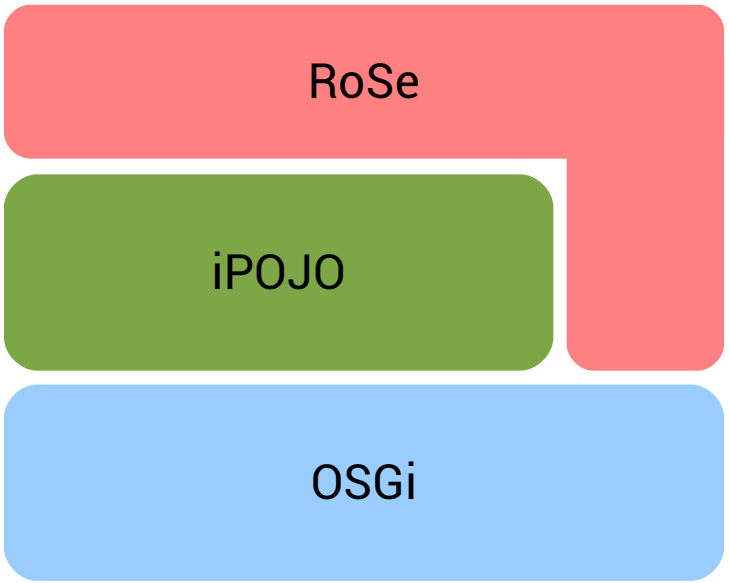
RoSe

Conception and Implementation Overview

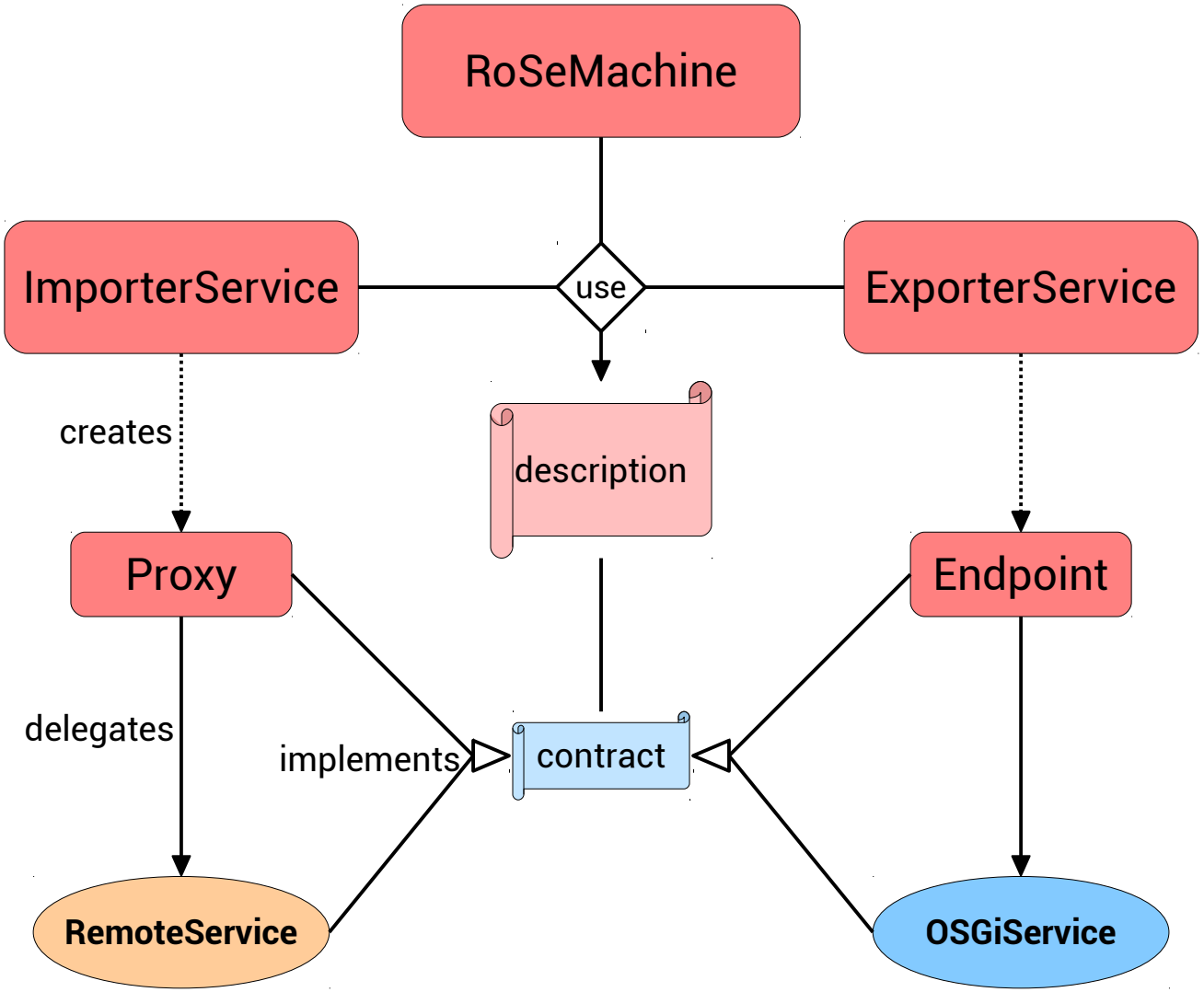
Validation & Evaluation

Conclusion

Implementation



RoSe Elements



RoSeMachine

The mediator between the framework components

Contains the description of each imported and exported services

Relies on the OSGi service registry

Importer & Exporter

The factories that creates proxies and endpoints

One factory per communication protocol

May wrap a library

Description

Represents a remote service

A set of properties

Immutable data that serves as the basis for discovery

Description

Property	Description	Optional
endpoint.id	The Endpoint id	
machine.id	The Machine Id	
service.contract	Contract name	
service.contract.version	Contract version	x
service.id	Id of the associated service	x
endpoint.url	The remote service endpoint address	x
endpoint.config	Protocol name	

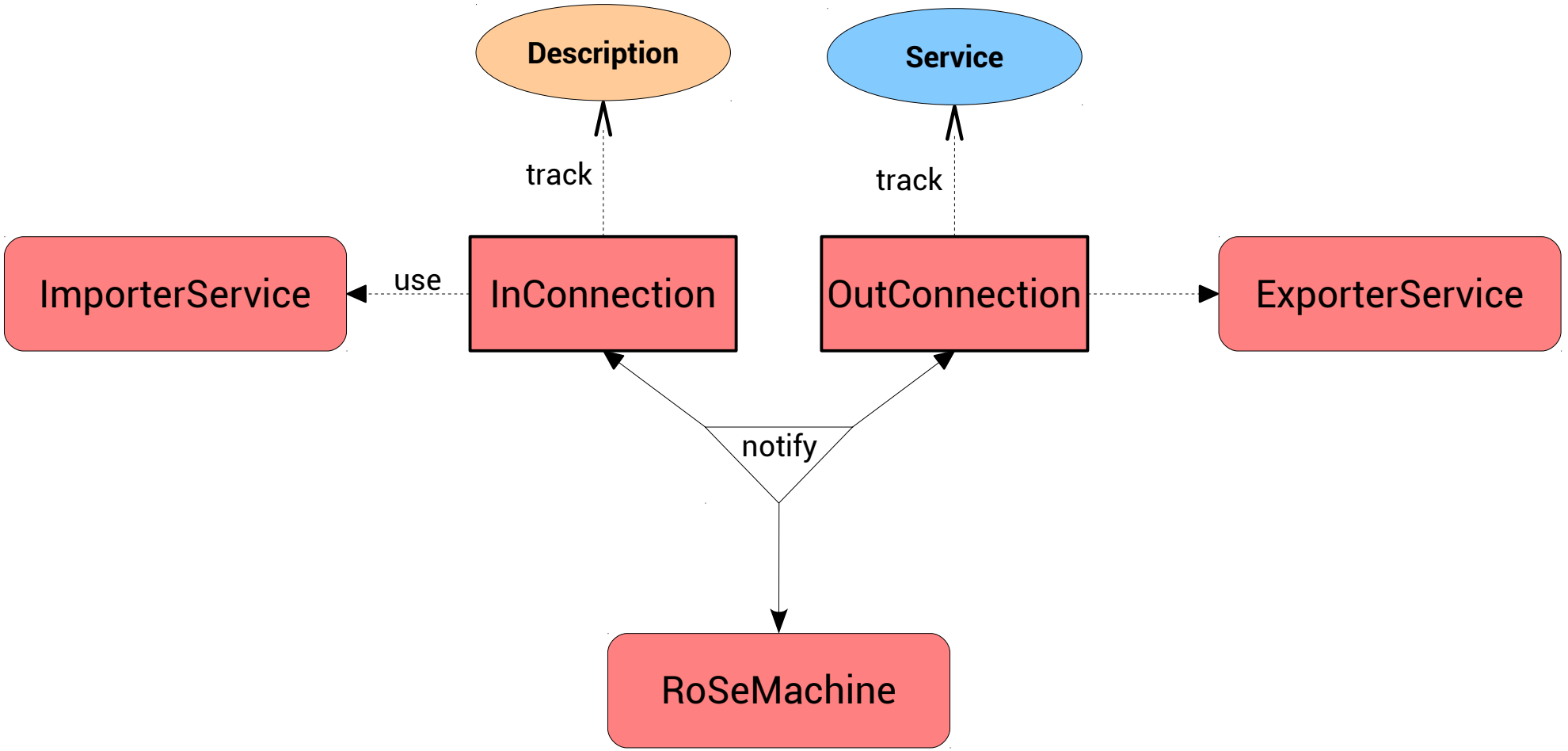
Discovery

Discovered and Published Description

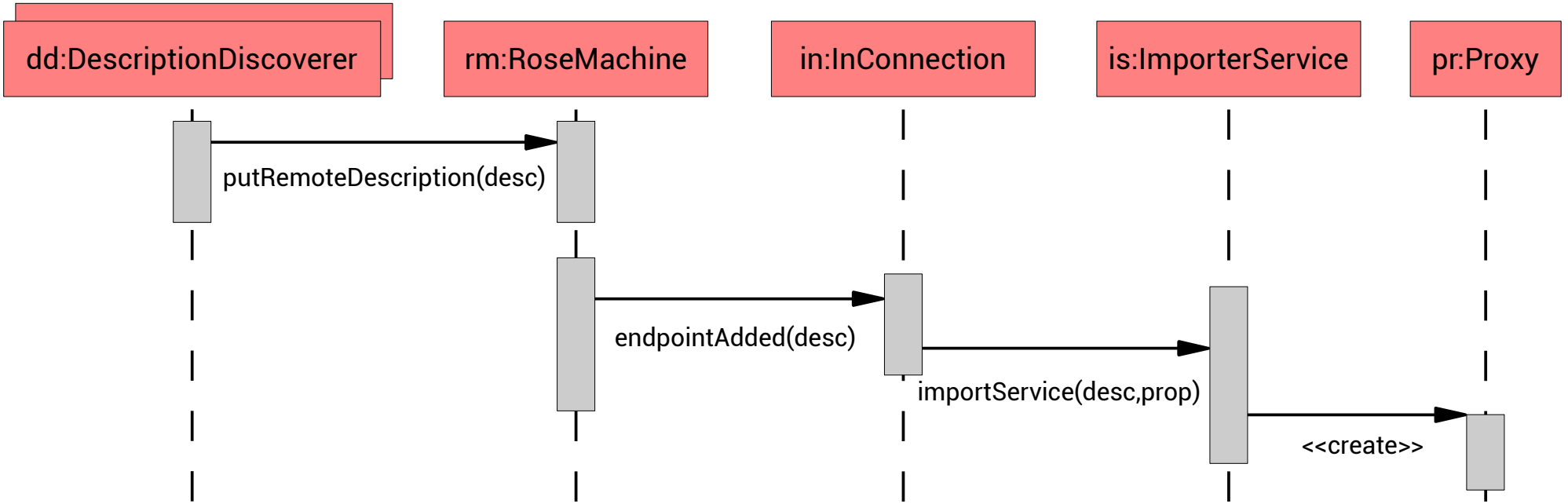
One component per discovery protocol

Optional

Connections



Connections



Example

```
1  {
2   machine : {
3    id : "id",
4    host : "localhost",
5    connections : [
6     { in : {
7      endpoint_filter : "(endpoint.config=ws)",
8      importer_filter : "(instance.name=RoSe_importer.cxf)",
9      protocols : [ "ws" ] }
10    }]
11 }
```

Embedded DSL

```
1 Machine machine = machine(context, "id").host("localhost").create();
2 machine.start();
3
4 machine.in("(endpoint.config=ws)")
5 .withImporter("(instance.name=RoSe_importer.cxf)")
6 .protocols(["ws"])
7 .add();
```

Supported Protocols

Protocol	Style	Library	Import	Export
JSON-RPC	RPC	jabsorb.org	x	x
XML-RPC	RPC	Apache XML-RPC	x	x
JAX-RS	REST	Oracle Jersey		x
JAX-WS	WS	Apache CXF	x	x

Supported Protocols

Protocol	Domain	Pub.	Disc.
UPnP	Pervasive	(x)	x
DPWS	Pervasive	x	x
Dns-sd	Pervasive	x	x
Modbus	Pervasive, industrial		x
Comet-d	Web	x	x
Zookeeper	Web, industrial, cloud	x	x
Pubsubhubbub	Web, cloud	x	x

Outline

Context & Challenges

State of the Art

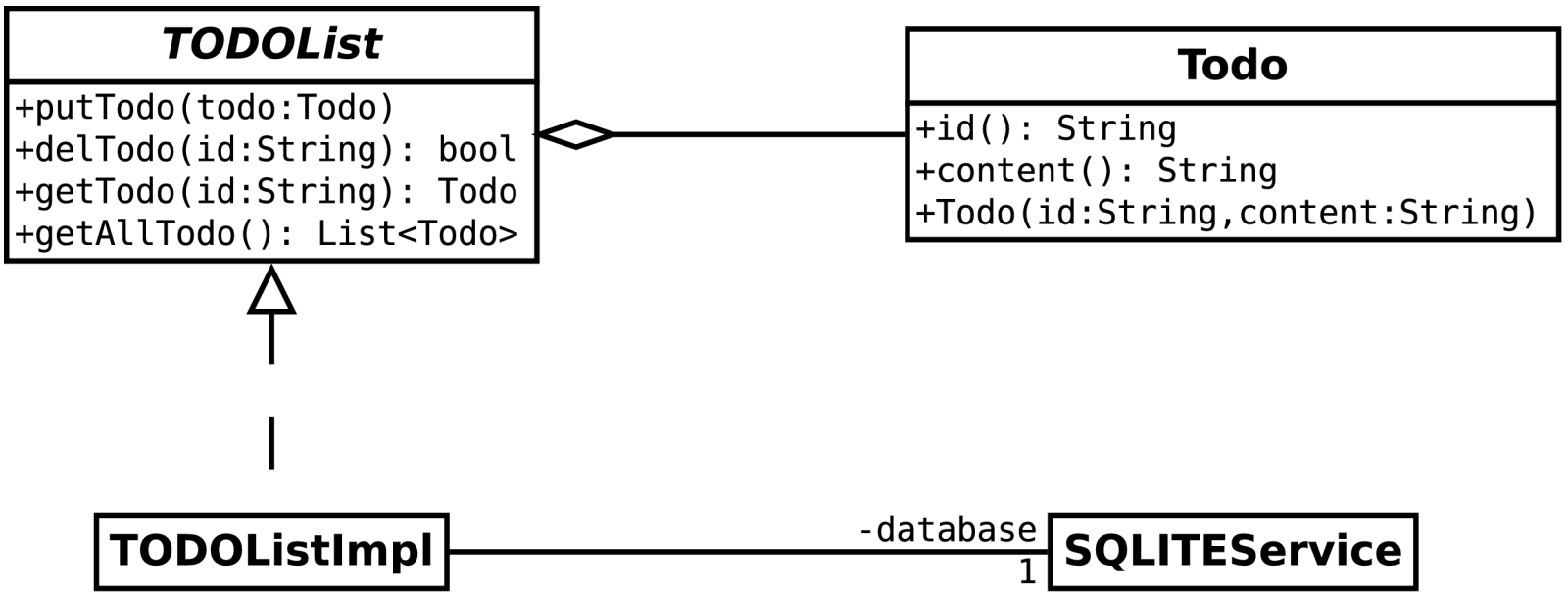
RoSe

Conception and Implementation Overview

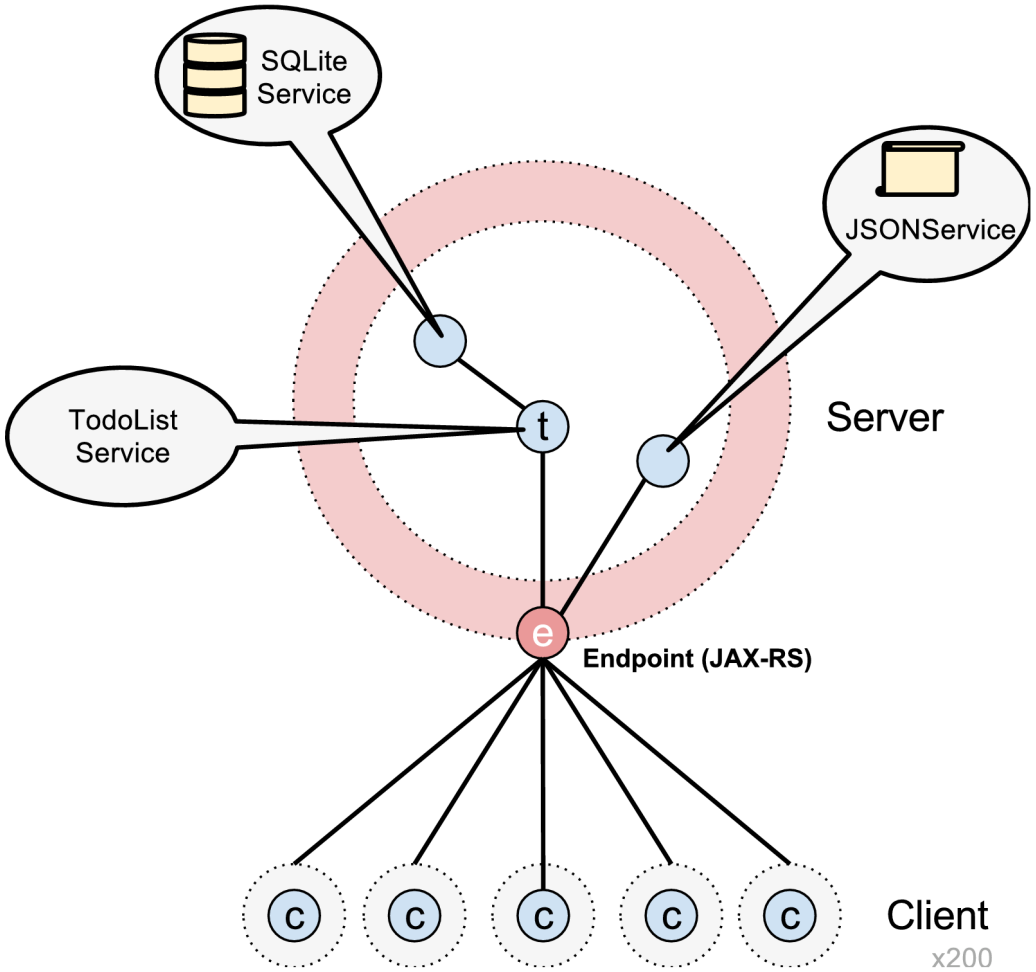
Validation & Evaluation

Conclusion

A simple todolist



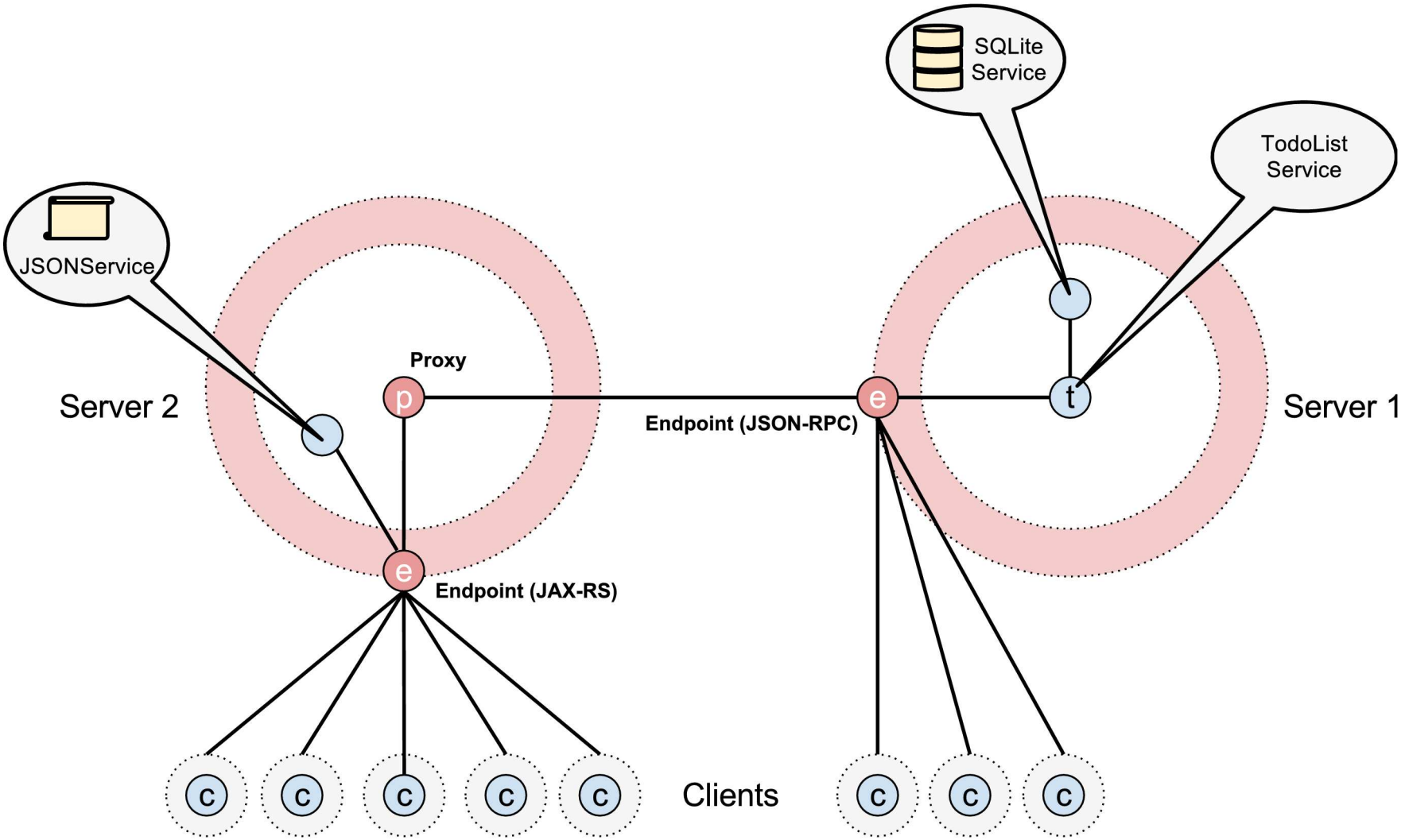
Case 1


SQLite
Service
JSONService
TodoList
Service
t
Server
e
Endpoint (JAX-RS)
c
c
c
c
c
Client
x200

Line of Code and Configuration

Framework	LOC	LOConfig.
Java	22	0
JavaEE	0	21
RoSe	0	15

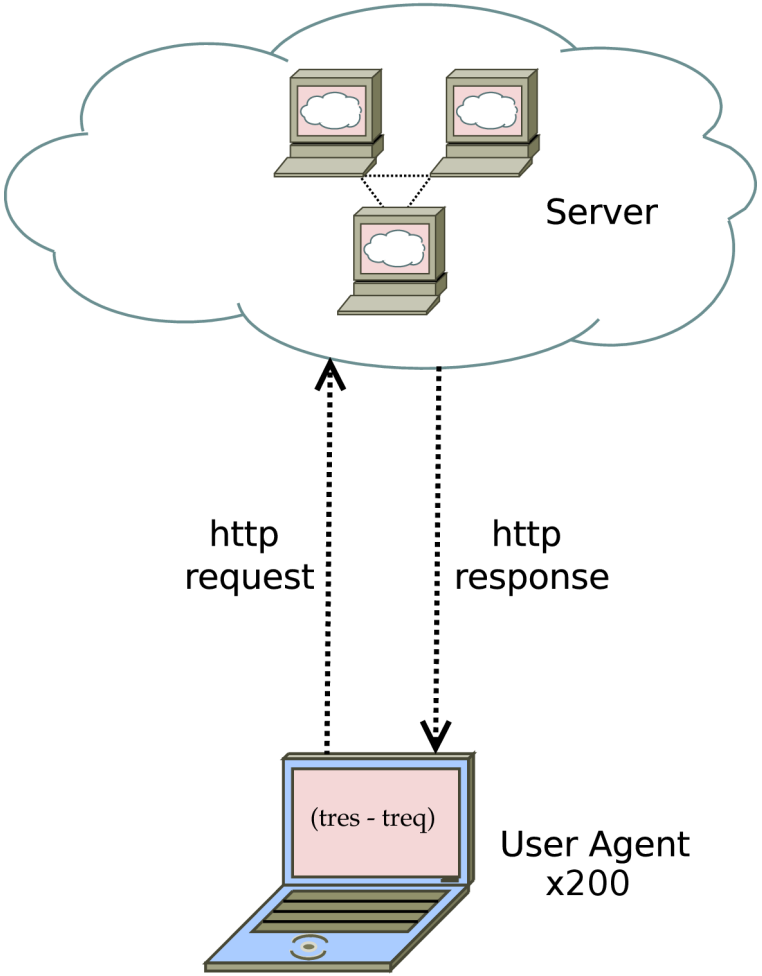
Case 2


SQLite
Service
TodoList
Service
JSONService
Proxy
Server 2
p
e
Endpoint (JSON-RPC)
e
t
Server 1
e
Endpoint (JAX-RS)
c
c
c
c
c
Clients
c
c
c

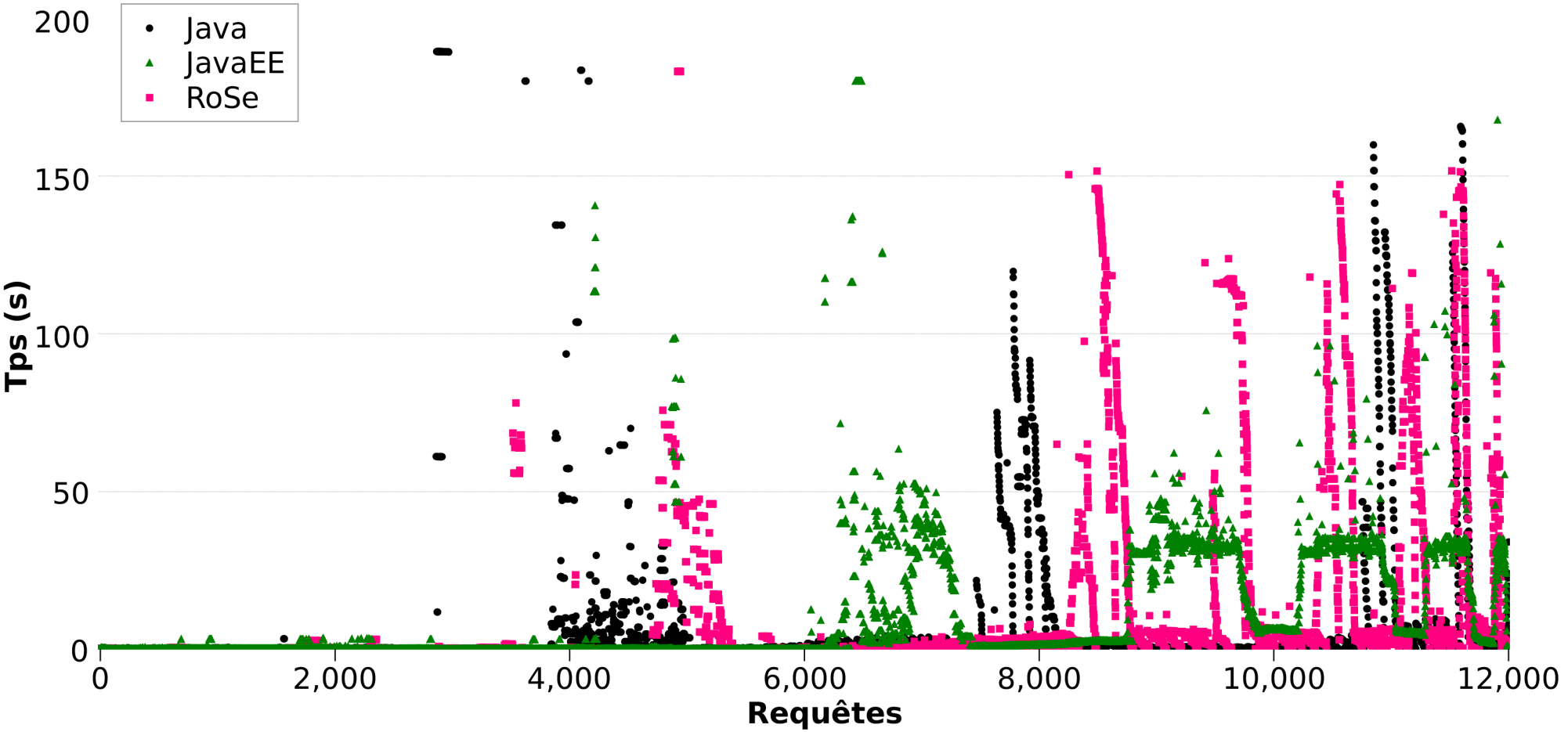
Line of Code and Configuration

Framework	LOC	LOConfig.
Java	56	0
JavaEE	17	43
RoSe	0	34

Performance



Performance



Performance

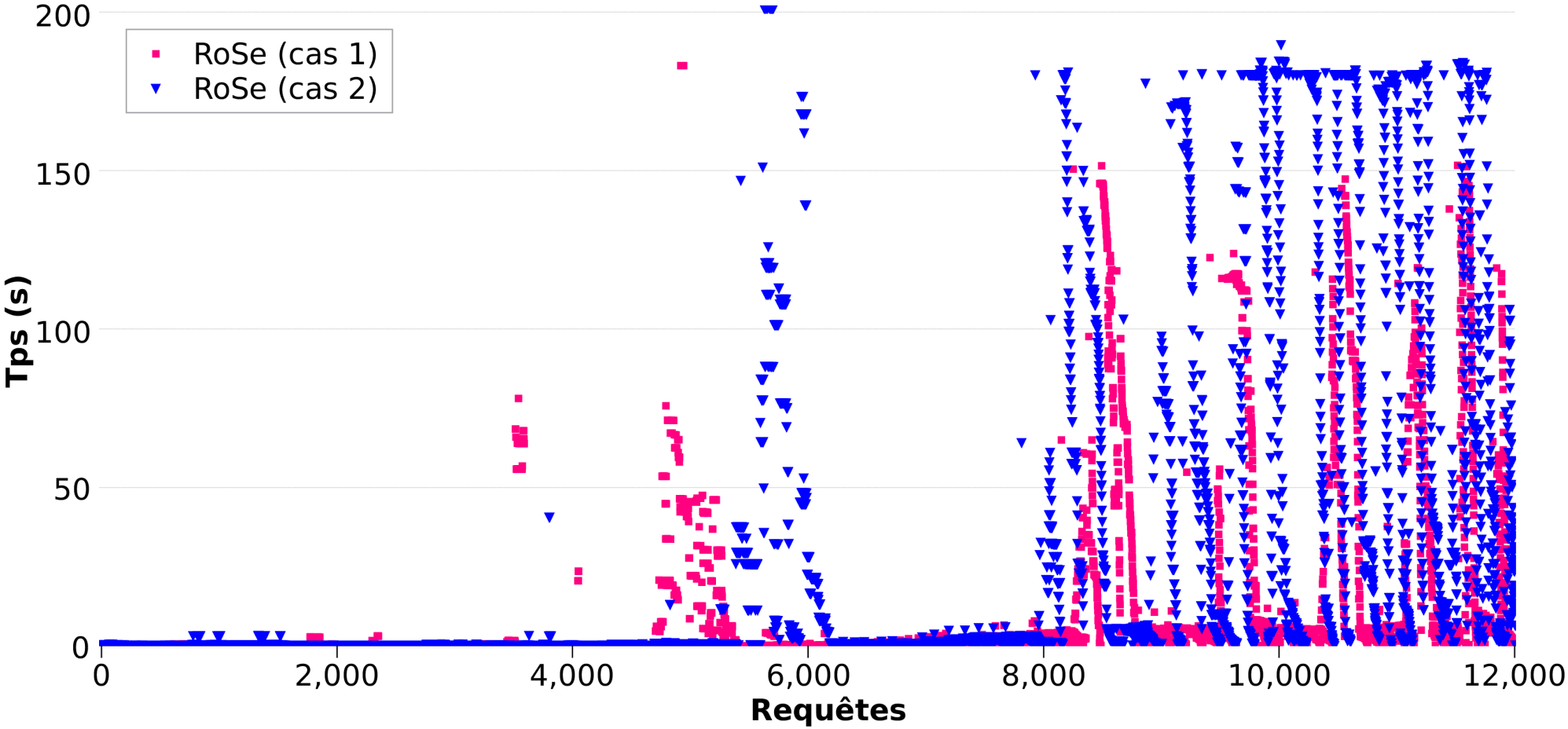
Framework	Mem. max (mb)	CPU av. (%)	CPU max. (%)
Java	**30.15**	54.53	99
JavaEE	58.57	**52.91**	99
RoSe	37.24	56.57	99

Framework	Av. time (s)	Median time (s)
Java	**8.116**	0.655
JavaEE	10.219	0.616
RoSe	10.749	**0.477**

Flexibility at runtime

1. LogService update
2. Jax-RS exporter update
3. Add an out connection to export the TodoList as a web-service.
4. Change configuration file.

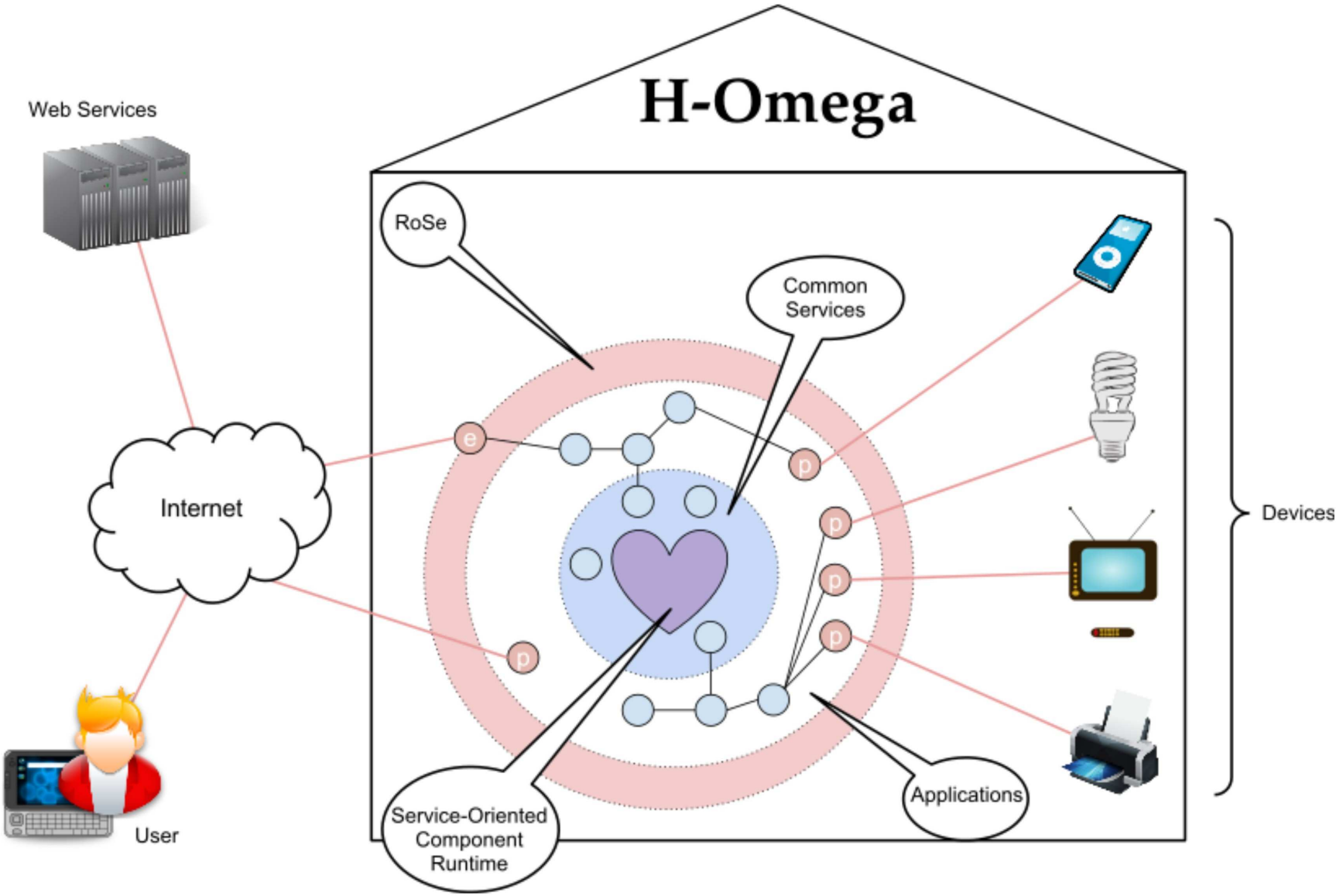
Flexibility at runtime



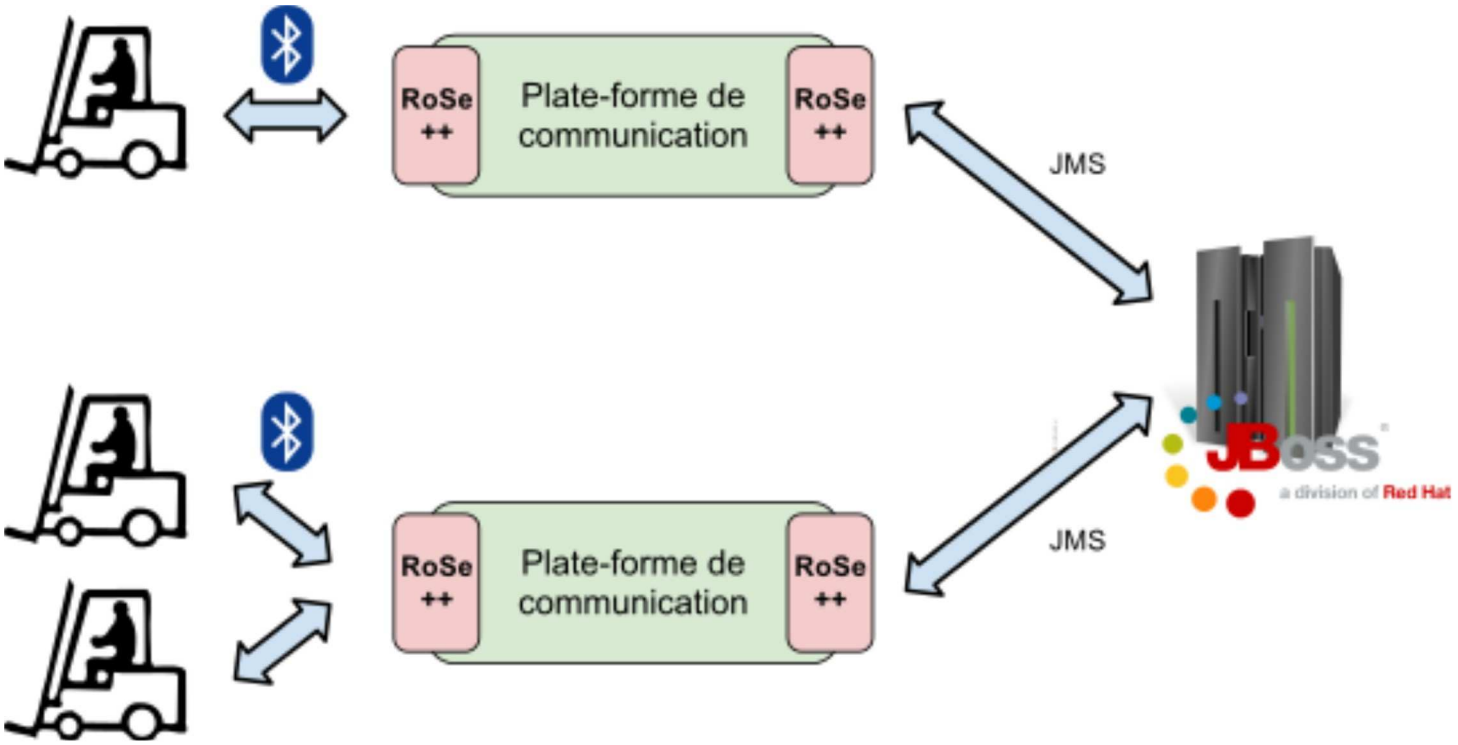
Flexibility at runtime

Framework	Av. time (s)	Median time (s)
RoSe (1)	**10.749**	**0.477**
RoSe (2)	26	0.54

Use of RoSe


H-Omega
Web Services
RoSe
Common Services
Internet
e
p
p
p
p
p
Devices
User
Service-Oriented Component Runtime
Applications

A Communication Platform



Outline

Context & Challenges

State of the Art

RoSe

Conception and Implementation Overview

Validation & Evaluation

Conclusion

Conclusion

Transparent distribution

Separation of concerns

Automation

Flexibility

Anarchic scalability

Simplicity

Diversity

Perspectives

Towards an autonomic distribution management

A new RoSe framework for the web

Towards new discovery protocols

Thanks!

Pervasive middleware