



Emergent Connectors for
Eternal Software Intensive Networked Systems



Emergent Middleware: Rethinking Interoperability for Complex Systems

Paul Grace
Lancaster University



docomo
DOCOMO Euro-Labs

LANCASTER
UNIVERSITY



THALES



tu technische universität
dortmund



Introducing the Connect Project: Towards emergent middleware

- Resolve interoperability barriers through on the fly CONNECTor synthesis
 - Dynamic Interoperability of Middleware/Application Protocols
 - Case Study: Service Discovery Interoperability – SeDiM from LANCS
- Elicit supporting formal foundation for CONNECTors
- Synthesize CONNECTors that are dependable, unobtrusive, and evolvable



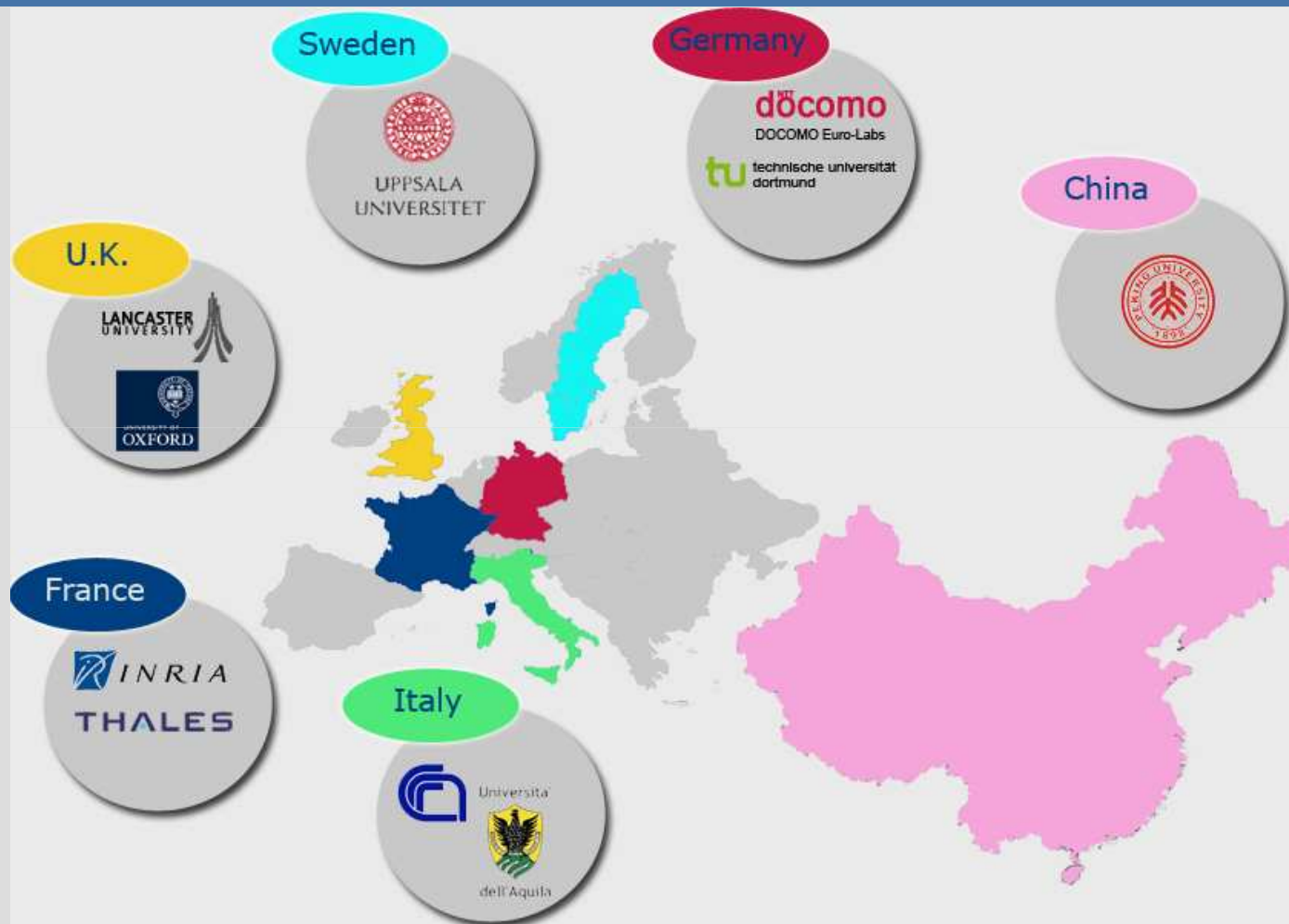
connect-forever.eu



<http://connect-forever.eu/>



The CONNECT Consortium



Interoperability in Complex Systems

- Emergence of multi-faceted ‘*Systems of Systems*’
 - Work together to meet the global aims of associated applications and services
- Characteristics
 - Extreme heterogeneity
 - Technology, network, software, hardware
 - Dynamic composition
 - Which systems interact with which is unknown until runtime
- Interoperability
 - “*Ability of two or more systems to exchange information and to use the information*”





Motivating Scenario

Service Discovery Protocol Heterogeneity

Revisiting Interoperability

- Interoperability is defined by Tanenbaum & van Steen¹
 - *“the extent by which two implementations of systems or components from different manufacturers can co-exist and work together by merely relying on each other’s services as specified by a common standard”.*
- If common standards were agreed and adopted universally then this problem would be largely solved. However, different standards exist:
 - Emerging Technologies
 - Technology development is too fast compared to the standards process, particularly at the application level
 - “One size” standards do not cater for extreme heterogeneity
 - Legacy standards remain useful in practice

1. A. Tanenbaum and M. van Steen, “Distributed Systems: Principles and Paradigms”, Prentice-Hall

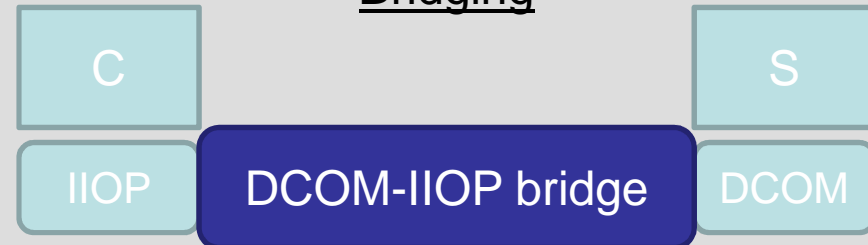
Middleware Approaches to Interoperability

The Middleware Approach



No interoperation with other standards and technologies

Bridging



Static deployment; bridge type must be available; significant effort to develop each bridge

Dynamic Bridges & Service Buses



Tied to interaction styles (e.g. RPC or messaging);

Analysis

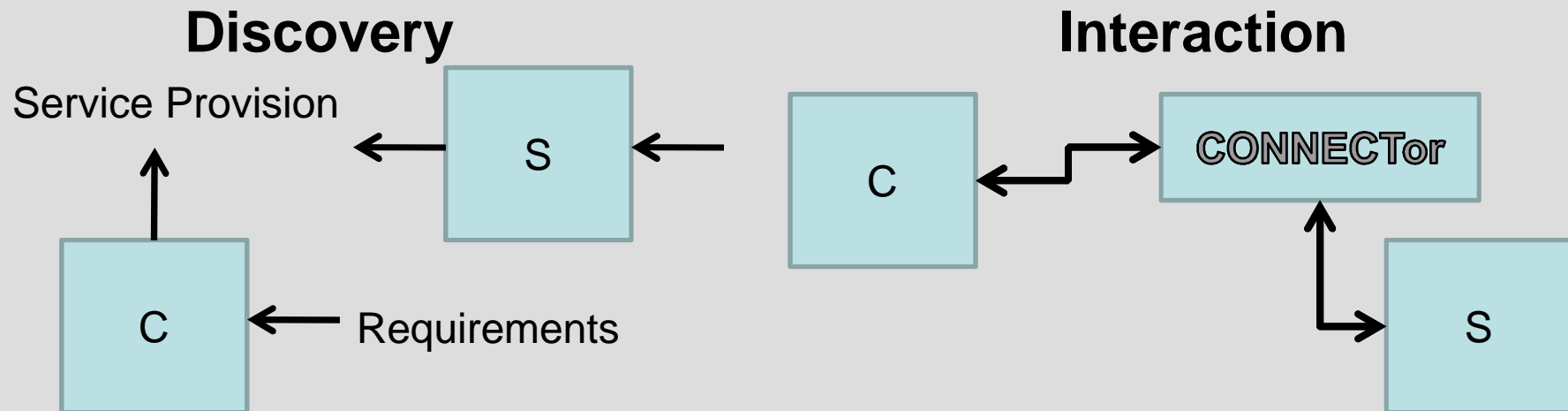
- Focus on Communication protocols
 - Applications, semantic interoperability?
 - Cross abstraction interoperability?
 - RMI to P/S to Tuples to ...
- Known protocols aware of each other
 - New protocol to interoperate with at runtime?

Rethinking Interoperability

- State of the Art is mostly design-time interoperability
 - A priori knowledge of connected systems
 - Common standards
 - Middleware
 - Mediator/Adaptor/Wrapper
- ➔ Need for run-time interoperability
 - ➔ No a priori knowledge of networked systems
 - ➔ **No reference architecture / middleware / technology → “Next years legacy middleware”**

The Connect Approach

- Synthesize CONNECTors between heterogeneous communication endpoints
 - Generate Middleware and application protocols to create connections that will overcome the interoperability barrier
 - CONNECTors devised and created at **Run Time**



A 3 Phase Autonomic Approach

■ Monitoring

- Discover behaviour of distributed systems
 - Service provisions and requirements
 - Protocol behaviour

■ Learning

- Protocol behaviour
 - Learn to interoperate with a protocol

■ Synthesis

- Generate middleware and application protocols

■ Challenges

- Bootstrapping monitoring
 - Where to start monitoring
- Reusing prior knowledge
 - Including prior information about middleware & middleware code
- Achieving autonomy
 - Criteria of success?
- Synthesis of functional and non-functional interoperability solutions

A Small Step along the Way: Service Discovery Interoperability



Internet

UDDI, INS



Fixed-Wired

SLP, UPnP, Jini, Bonjour



Wireless



Ad-hoc

SSD, GSD, Allia, SLP-B,
Bluetooth, Lanes,, Splendor,
Ariadne

■ In a fully connected world:

- Advertise and/or Find your service irrespective of the:
 - Protocol
 - Description Language
 - Behaviour Model

■ @Lancaster we have developed SeDiM an adaptive middleware to:

- Develop applications transparently from SDP protocols
- Support interoperability between legacy SDP applications

Challenges of Discovery Heterogeneity

Six challenges that must be resolved to achieve SDP interoperability:

1. Heterogeneous description of the behaviour

- Different description languages to describe services e.g. WSDL, SLP templates, ...

2. Heterogeneous Systems Architecture

- e.g. Protocols that use structured networks of directories versus unstructured

3. Heterogeneous Protocol Behaviour

- Active/passive protocols e.g. lookup request and respond, versus periodic announcement

4. Heterogeneous Message Format

- Each discovery protocol uses a different message format

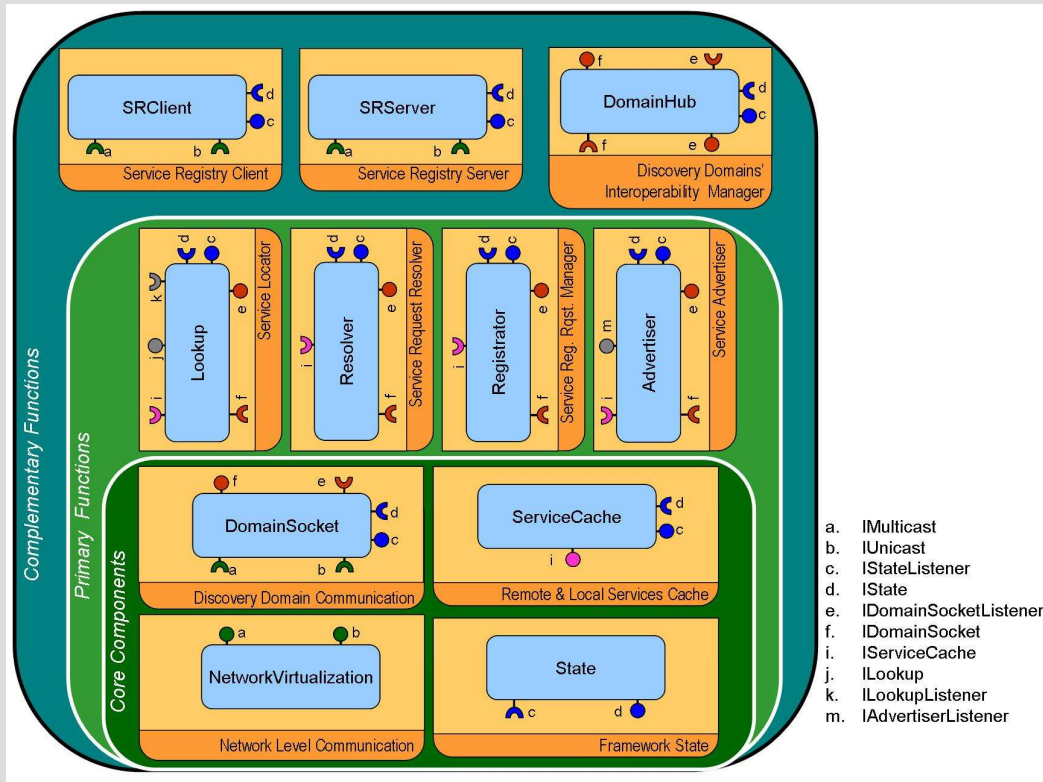
5. Heterogeneous Networks

- Lookup requests and responses are routed over different networks e.g. IP multicast, P2P (Pastry, Gnutella), MANETs

6. Heterogeneous Non-functional properties

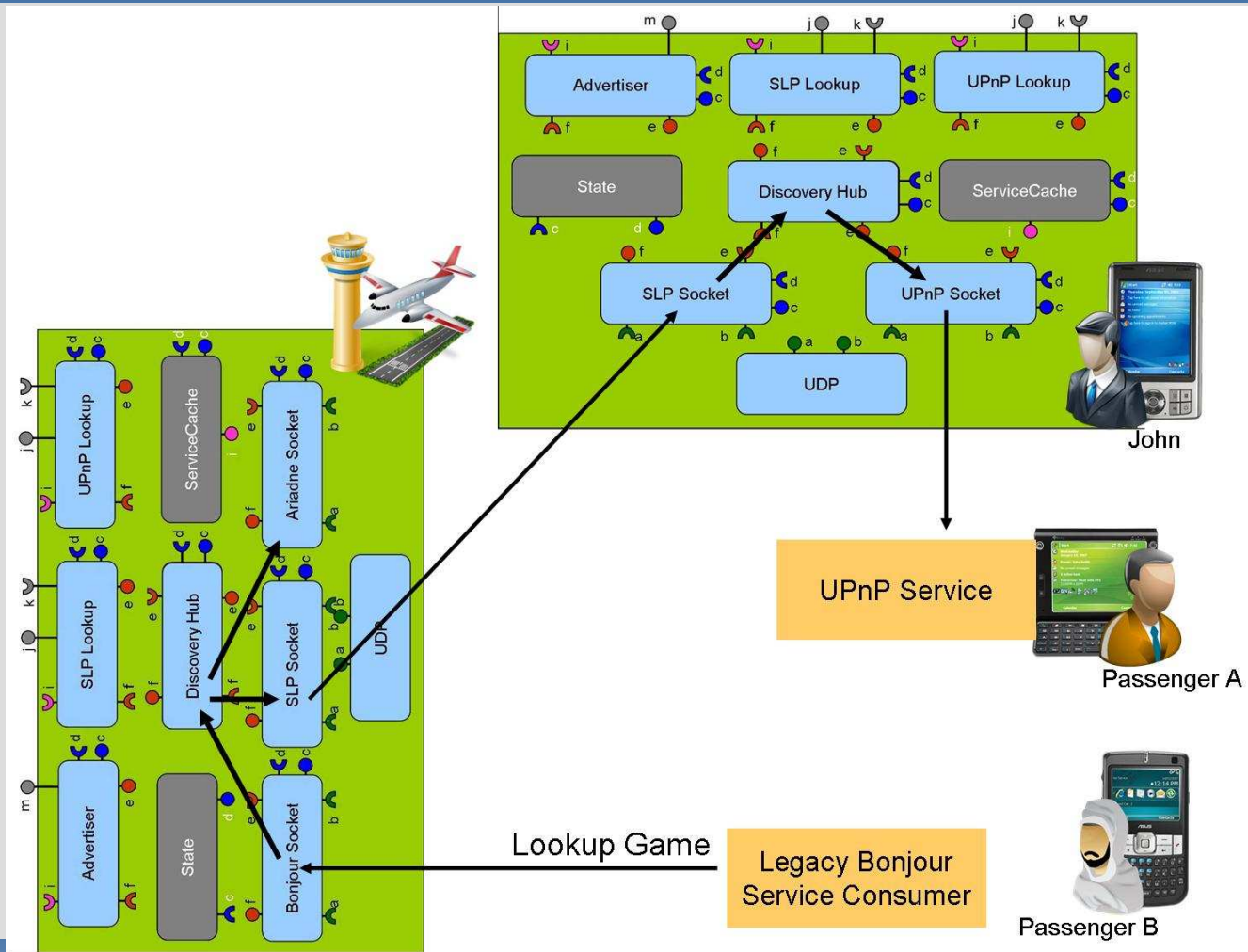
- Security, privacy, location mechanisms in SDPs

A Dynamically Reconfigurable Framework

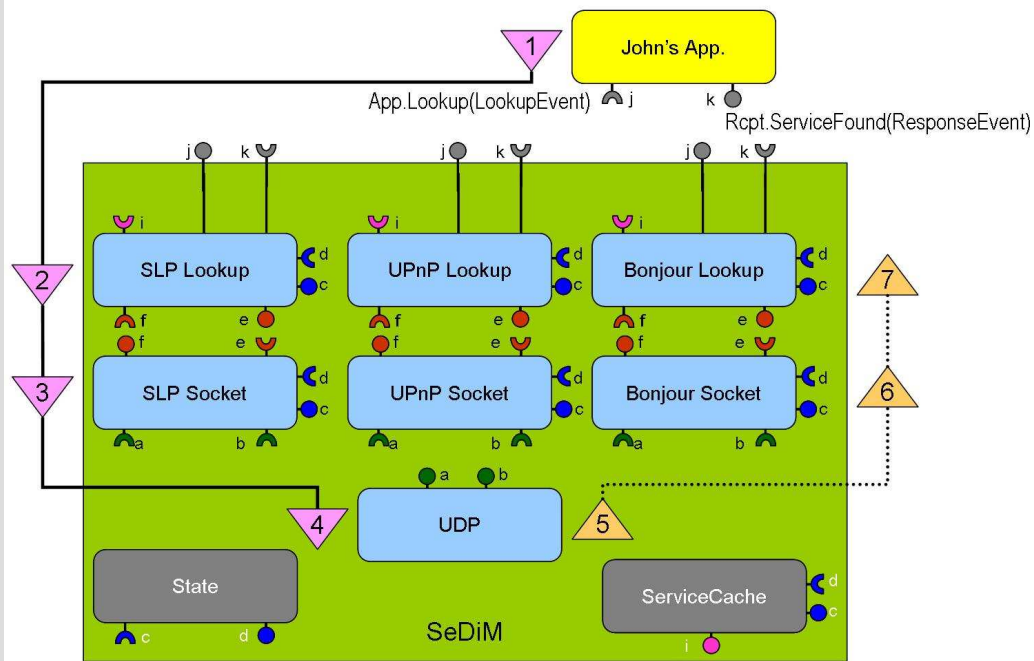


- Configure Single & Multiple protocols e.g. SLP, UPnP and Bonjour by ***specialising the component architecture***
- Network Component
 - Plug-in routing component
- DomainSocket
 - Map protocols e.g. SLP message format to SeDiM internal abstraction
- DomainHub
 - Map between protocols i.e. Achieve interoperability
- Reusable components to create configurations at runtime

SeDiM in Action: Interoperability between SeDiM and Legacy



SeDiM in Action: Transparent Interoperation



```
SeDiM sedimApp = new SeDiM()
```

```
Service printer = new Service("Printer")
```

```
FeatureAttribute attr = new
```

```
    FeatureAttribute("Type", "Laser")
```

```
printer.addAttribute(attr)
```

```
sedimApp.lookup(new LookupEvent(printer))
```

Future Work: Making SeDiM Autonomic

- Introduce a new unknown discovery protocols into SeDiM
 - Develop methods to monitor its behaviour
 - Develop methods to learn the discovery protocols
 - Based upon a common architecture of discovery protocol behaviour
 - Synthesize the components to specialise the SeDiM personalities

Conclusions

- There has been considerable volume of research on interoperability in distributed systems; while progress has been made, the state of the art remains rather patchy, particularly when addressing the complexity of contemporary, highly heterogeneous distributed systems.
- **CONNECT** aims to :
 - identify a common framework for Emergent Middleware
 - covering discovery, interaction and quality of service;
 - Ensure semantically correct discovery and interaction;
 - Automatically synthesize Emergent Middleware.
- These are big challenges that can revolutionize the state of the art in distributed systems in general and middleware more specifically.

Acknowledgements

- Carlos Flores Cortes
 - Primary designer and developer of SeDiM
- Gordon Blair
- Connect Partners
 - Valerie Issarny (Project co-ordinator)

Questions

- More Information at
 - [Http://connect-forever.eu](http://connect-forever.eu)