HAL
open science

Emergent middleware: rethinking interoperability for complex pervasive systems

Paul Grace, Carlos Flores, Gordon Blair

To cite this version:

Paul Grace, Carlos Flores, Gordon Blair. Emergent middleware: rethinking interoperability for complex pervasive systems. 10th ACM/IFIP/USENIX International Conference on Middleware Poster Session, Dec 2009, Urbanna, Illinois, United States. inria-00459163

HAL Id: inria-00459163

https://inria.hal.science/inria-00459163

Submitted on 23 Feb 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Emergent Middleware: Rethinking Interoperability for Complex Pervasive Systems

Paul Grace
Computing Department
Lancaster University, UK
p.grace@lancaster.ac.uk

Carlos Flores
Computing Department
Lancaster University, UK
c.florescortes@lancaster.ac.uk

Gordon Blair
Computing Department
Lancaster University, UK
gordon@comp.lancs.ac.uk

ABSTRACT
Complex systems are characterized by extreme heterogeneity and dynamic composition, and hence pose significant challenges to achieve interoperability. For example, where multiple middleware solutions and protocols are employed, these must be connected in order for applications to operate. We propose a new approach to interoperability that focuses of monitoring, learning and synthesis of middleware behaviour.

Categories and Subject Descriptors
C.2.4 [**Computer Communication Networks**]: Distributed Systems – *Distributed applications.*

General Terms
Design.

Keywords
Middleware, Interoperability, Learning, Synthesis.

1. MOTIVATION
Complex pervasive systems are replacing the traditional view of homogenous distributed systems, where domain specific middleware solutions are used to design and deploy distributed applications. For example, enterprise middleware for enterprise systems, or Grid middleware for Grid applications. Instead, pervasive applications are composed from multi-faceted *systems of systems* where subsystems (each of which is a separately designed and developed system e.g. a sensor network) works together to meet the global aims of the application. The following are illustrative of the types of pervasive applications embracing these philosophies:

- In *Environmental Monitoring and Control*, field-deployed sensors connect to high-performance grid and cluster computers to better monitor and predict natural phenomena such as floods, hurricanes, and volcanic eruptions.
- In *Transport*, embedded sensors in cars, vehicular networks and traffic monitoring systems are integrated to improve both traffic safety and traffic flow.
- In *Healthcare*, remote patient monitoring devices are integrated into large-scale healthcare systems to improve standards of patient care.

To provide these application services, heterogeneous systems must interoperate; where interoperability is defined as *"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"*[1]. This is a principal goal of middleware, and indeed, if common standards were agreed and adopted universally then this problem would be largely solved; however, systems-of-systems have two key properties that ensure that current middleware practices are not suitable and we must rethink approaches to achieve interoperability in this domain:

1. *Extreme heterogeneity*. Pervasive sensors, embedded devices, PCs, mobile phones, and supercomputers are connected using a range of networking solutions, protocols and middleware are themselves composed to create complex systems-of-systems.
2. *Dynamic Communication*. Connections between systems are not made until runtime; no design or deployment decision e.g. choice of middleware can inform the interoperability solution.

With such characteristics using common middleware technologies (with or without common standards) is unsuitable in practice, as a number of technologies co-exist for technical, commercial, or legacy reasons. Indeed, interoperation is required to connect heterogeneous middleware platforms and protocols (e.g. SOAP, CORBA, EJB). Importantly, systems cannot also be aware which technologies they need to interoperate with until runtime.

Hence, we require new ideas that go beyond the state-of-the art in middleware that can identify at runtime what the interoperability challenges are, what middleware solutions are required to connect these systems, and generate or synthesize such software on the fly.

2. INTEROPERABILITY: THE CONNECT APPROACH
Rather than create a middleware solution destined to be yet another legacy platform, work on the Connect project (http://www.connect-forever.eu) is based upon the concept of *emergent middleware*; where such middleware provides runtime interoperability between two systems that spontaneously interact on the fly. Connect synthesizes 'connectors' that resolve interoperability at the data (e.g. heterogeneous data formats), application protocol (for example, different instant messaging or printing protocols) and middleware protocol (e.g. different service discovery or RPC protocols) layers.

The creation of interoperability 'connectors' is performed in three distinct phases:

- *Monitoring and discovery*. The operation of the two or more systems required to connect is monitored and/or discovered. This involves the extraction of information about the systems using traditional discovery protocols e.g. that provided by protocols such as SLP, or in description languages such as

WSDL. Monitoring of protocol messages and behaviours is also used to build a picture of how the protocols operate.

- *Learning*. Using the prior information as a starting point, machine learning approaches are employed to learn how one must interoperate with a particular system i.e. how exactly the middleware and application protocols behave.
- *Synthesis*. Based upon the learned models of behaviour, the differences and similarities of protocols can be identified; such information can be used to synthesize a connector that will operate as a mediator or bridge between systems.

Hence, this approach goes beyond current state of the art in dynamic interoperability; application software does not need to be altered at design time (as with the majority of middleware solutions) nor is there a requirement for a common framework with a priori knowledge (and implemented solutions) of the bridging protocols to be deployed for connection to be achieved e.g. as provided in INDISS [2], ReMMoC [3], and uMiddle [4].

3. CASE STUDY: SERVICE DISCOVERY

A particular example of middleware heterogeneity is in the field of service discovery. Because of the heterogeneity of pervasive environments in terms of network styles (e.g. fixed-infrastructure, mobile ad-hoc) and device capabilities (e.g. resource constrained versus resource rich) there exist a suite of discovery protocols. Protocols have been developed for operating in environments such as sensor networks, enterprise networks (e.g. Bonjour, Jini and UDDI) and mobile ad-hoc networks among others. All these protocols share a same common purpose: to advertise and discover services. However, they differ in a number of distinct concerns that are magnified by the different environments they are employed within: i) the languages used to describe and advertise service's behaviour; ii) the content and format of the protocol messages; iii) the distributed architecture of directories where service advertisements are maintained (indeed some protocols have no directory architecture); iv) the discovery behaviour model, e.g. if the protocol behaves actively to request services, or passively listens for service announcements; v) the network communication protocol employed to route messages, e.g. IP Multicast versus a peer-to-peer overlay; vi) the non-functional features included in service description and discovery, e.g. security, privacy and trust properties.

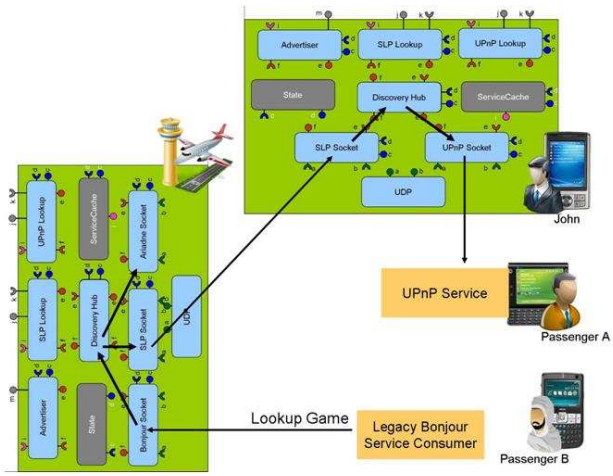

Figure 1. Connecting SLP, UPnP and Bonjour with SeDiM

As an initial approach to ensuring dynamic interoperability between heterogeneous discovery protocols we have developed SeDiM. This is an adaptive middleware that monitors the environment to determine which protocols are in use i.e. what protocols are being used for lookup and which are being used for advertisement; from this information it creates service discovery protocol bridges. This involves the specialisation of a common component software architecture to particular protocol i.e. adding protocol specific code to create the behaviour. SeDiM then uses a translation pattern between instances of the specialised protocol configurations. This process is illustrated in figure 1. Here two legacy applications are deployed atop UPnP and SLP; SeDiM detects this and automatically deploys appropriate component architectures to bridge the two.

SeDiM relies on pre-implementation of the protocol specialisation component at present. However, we are currently investigating approaches to synthesise the discovery protocol connectors.

4. CONCLUSIONS: FUTURE ROADMAP

There has been considerable 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; and automatically synthesize Emergent Middleware.

There are research challenges we hope to address; i.e. how to learn the behaviour of a middleware protocol; how to discover the message format of a protocol at runtime; how to leverage semantics to determine a common understanding between two systems; how to synthesise middleware; how to maintain end-to-end QoS in connectors e.g. dependability and security.

These are big challenges that we believe can revolutionize the state of the art in distributed systems in general, and middleware more specifically.

5. ACKNOWLEDGMENTS

This work is funded under the CONNECT Project (http://connect-forever.eu) (FP7 Theme / ICT-2007.8.6, FET Proactive 6: ICT Forever Yours)

6. REFERENCES

[1] Tanenbaum, A. and van Steen, M. 2007. Distributed Systems: Principles and Paradigms. Prentice-Hall.

[2] Bromberg, Y. and Issarny, V. 2005. INDISS: interoperable discovery system for networked services. In *Proceedings of the ACM/IFIP/USENIX 2005 international Conference on Middleware*. G. Alonso, Ed. Middleware Conference. Springer-Verlag New York, New York, NY, 164-183.

[3] Grace, P., Blair, G. S., and Samuel, S. 2003. ReMMoC: A Reflective Middleware to support Mobile Client Interoperability. *In Proceedings of the Int. Symposium on Distributed Objects and Applications*, 1170-1187

[4] Nakazawa, J., Tokuda, H., Edwards, W. K., and Ramachandran, U. 2006. A Bridging Framework for Universal Interoperability in Pervasive Systems. In *Proceedings of the 26th IEEE international Conference on Distributed Computing Systems*.