Interoperating with heterogeneous Mobile Services

Paul Grace and Gordon S. Blair
Computing Department
Lancaster University

Mobile applications are now developed upon a wide range of service development platforms, commonly referred to as middleware. However, the diversity of those available presents a problem for mobile client development. How can a single client implementation interoperate with heterogeneous service implementations?

The emergence of mobile computing has created new classes of applications dependent upon the user's location, context and interaction with their current environment. However, given the constraints of the wireless environment (i.e. weak connection, poor network quality of service and mobile devices with limited resources) developing distributed applications within this domain is a complex task. Therefore, new middleware has emerged to mask such problems from the user. These encompass synchronous (e.g. remote method invocation) and asynchronous (e.g. publish-subscribe and tuple spaces) communication paradigms. However, it is the heterogeneous nature of the solutions that has created a new problem. Only applications and mobile services developed upon the same middleware style can interoperate with one another. For example, a tourist guide client application interacts with services available at the user's current location that provide tourist information. If the client is implemented upon a particular middleware platform (for example, SOAP) it will only function in locations that offer tourist services across SOAP.

Similarly, services are advertised using one of the contrasting service discovery protocols. At present, four main service discovery protocols exist: Jini, Service Location Protocol (SLP), Universal Plug and Play (UPnP) and Salutation. In addition, new protocols are emerging to better support the discovery of services in mobile environments and across ad-hoc wireless networks (e.g. Service Discovery Protocol in Bluetooth and Salutation Lite). Mobile clients using a single discovery protocol will miss services advertised by alternative protocols.

Hence, we argue that adaptive middleware is required to support the interoperation of mobile clients with heterogeneous services. Using this approach, middleware dynamically alters its behaviour to: i) find the required mobile services irrespective of the service discovery protocol, and ii) interoperates with services implemented by different types of middleware.

The ReMMoC (Reflective Middleware for Mobile Computing) project, which began in October 2000 and concludes in September 2003, is being carried out at Lancaster University in collaboration with Lucent Technologies UK. The aim of the project is to design and implement such an adaptive middleware, which allows mobile clients to be developed independently from the underlying middleware implementations that may be encountered at different locations.

For applications to dynamically operate across different middleware they must be programmed independently from them. Hence, abstract definitions of mobile services

are required. The client application, which requests this service, can then be developed using this "interface" in the style of IDL programming. A request of the abstract service is mapped at run-time to the corresponding concrete request of the middleware implementation. The emerging Web Services Architecture includes a Web Services Description Language (WSDL) that provides such an approach based on abstract and concrete service definitions. We employ WSDL as the basis of service description and programming model.

To build the adaptive middleware, we utilise three key techniques: components, component frameworks and reflection. We use OpenCOM (developed at Lancaster University) as our component model to develop components for Windows CE platforms. OpenCOM is a lightweight, efficient and reflective component model, built atop a subset of Microsoft's COM. The motivation behind component frameworks is then to constrain the scope for evolution, ensuring all dynamic changes between component configurations are legitimate. Finally, reflection is a principled method that supports introspection of the underlying middleware and provides techniques to dynamically adapt component architectures.

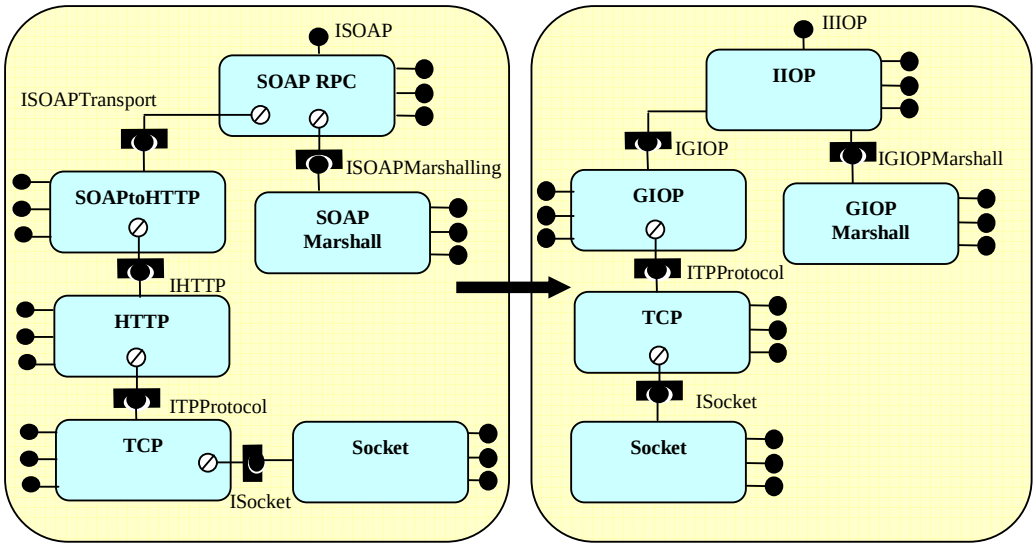

Figure 1. Dynamic change from SOAP client configuration to IIOP client configuration of OpenCOM components

The ReMMoC platform consists of two key component frameworks for binding and service discovery, whose behaviour can be dynamically altered using reflection. The binding framework re-configures between different interaction protocols (we have implemented IIOP, SOAP and publish-subscribe bindings); figure 1 illustrates a dynamic change of personalities within this framework. The service discovery framework changes between one or more of the different service discovery technologies (we have implemented the SLP and UPnP protocols).

ReMMoC has been fully developed and tested using simple applications e.g. chat, news and stock quote clients. Ongoing work includes an evaluation of this method on larger, complex applications (e.g. ubiquitous computing and intelligent home environments) and across a range of further middleware bindings including data

sharing and tuple spaces. An evaluation of memory use has concluded that each device cannot store every possible middleware component. Therefore, methods for dynamically downloading components when needed are required. Furthermore, techniques to ensure components are available to start-up before they are used, e.g. predictive caching based upon context information, are an interesting option.

Useful Links:
OpenCOM and ReMMoC Web Page -
http://www.comp.lancs.ac.uk/computing/research/mpg/projects/opencom/

Please Contact:

Paul Grace
Computing Department, Lancaster University, Lancaster, LA1 4YR
Tel: 01524 593801 E-mail: p.grace@lancaster.ac.uk

Gordon Blair (Project Co-ordinator)
Computing Department, Lancaster University, Lancaster, LA1 4YR
Tel: 01524 593801 E-mail: gordon@comp.lancs.ac.uk