NEMOS: Service Architecture for Lightweight Mobile Devices

Iulian Radu, Son T. Vuong
Department of Computer Science, University of British Columbia
Vancouver, BC, Canada V6T 1Z4
iulian@interchange.ubc.ca; vuong@cs.ubc.ca

ABSTRACT

This paper presents NEMOS, a semantic framework for deploying service oriented architecture on lightweight mobile devices such as sensor networks and cell phones. Definition and coordination of service compositions is performed using highly compact mobile agents, in an environment tailored for distributed task control. Device contents and capabilities are described using ontological metadata, permitting rich data semantics and facilitating service compositions for easy interoperability of sparse mobile devices.

Keywords: mobile agents, SOA, web services, service ontology, service composition

1. Introduction

Service oriented architectures (SOA) have developed great potential for business interoperation. They allow loose coupling of services available in a heterogeneous network, creating complex reusable added value services which can be accessed on demand. Using this architecture a mobile user can take advantage of processing capabilities of devices in the immediate environment, by representing a high level task through an aggregation of device services.

Most existing service architectures have been built on powerful platforms, and thus developed into complex standards. However, the full scope of most standards is not yet suitable for deployment on mobile devices.

Several difficulties arise in lightweight mobile devices which impede deployment of standard SOA architectures. Low processor capability and power consumption constraints limit processing power. Formatting messages as XML data presents a considerable overhead in processing and transmission of inter-service messaging. Communication bandwidth, availability and frequency are also constrained due to increased device mobility and power constraints. Thus preferred is an architecture in which messages are simple and program execution can be decoupled from unused network entities. A more pragmatic difficulty with such a disconnected infrastructure is the inability of a component to easily know what kind of data and services are offered by its peers. Providing semantic descriptions to device content offers a solution such that network hosts can more easily aggregate content.

We propose a service architecture for lightweight devices which uses a simple mobile agent system to manage services and data semantically described in device ontologies. Mobile agent environments allow programs and associated data to autonomously travel between network hosts while executing a distributed task. The NEMO agent system used for our prototype is a lightweight framework well suited for extension to distributed service control. This minimal system has been previously deployed on Nokia S60 handsets, while another version with similar capabilities is in development for use on sensor networks.

The mobile agent approach reduces communication and processing overhead by transferring the service coordination logic to the execution site, removing the need for centralized coordination and persistent connectivity between hosts. Message overhead due to XML encapsulation is also eliminated by treating service arguments and results as mobile agent data. Service composition can be flexibly expressed through the rich, yet compact, agent control language. Descriptions of data and services through ontological metadata allow the user and agents a semantic level view of network device contents, and permit on-the-fly creation of syntactically correct service compositions. The OWL-S language [6] allows ontological description of web services, and is intended for use by Semantic Web services [7]. Our system uses OWL representation for describing device services and data.

The current developments in service oriented architectures have mostly focused on development of service and process description architectures. The most popular SOA technology is the Web Services architecture [1]. Services from different vendors define their interfaces using a common standard, the Web Service Definition Language (WSDL) [2], and the communication is usually enabled by XML-encapsulated SOAP [3] messages over HTTP. UDDI [4] registries are employed for service discovery. Abstract control flow of business processes is described by the high level BPEL4WS standard [5]. The rich language can define complex composition of services

through a variety of control constructs, but extends beyond the scope of a minimal service architecture to include such concepts as fault recovery operations and policies for partnered interaction.

Combinations of ontological service descriptions and agent environments have been researched, but systems have swayed from interfacing on lightweight devices. Mobile agent systems have previously been employed to communicate with web services in mobile environments [10, 11], and perform BPEL processes [12, 13], however these systems are too heavyweight for deployment on cell phones or sensor networks. The existing mobile agent architectures for sensor networks [16, 17, 18] have not been shown applicable for service composition.

In the following section we describe the architecture of the NEMOS system. Section 2.1 focuses on the mobile agent environment and Section 2.2 describes the service composition layer. Section 2.3 shows our system prototype in operation for search-and-rescue scenario. A summary of our architecture, along with considerations for further research are presented in Section 3.

2. System Architecture

The system is formed by a semantic layer describing device contents whose operation is enabled by the mobile agent framework.

The data and services offered by each network device are described by ontological metadata. Device data is assumed to be classified into ontological objects which hold certain named property values or relationships to other objects. Device services are also represented in the ontologies, having specific object inputs and outputs which forms a typed definition permitting easy visualization and syntactic composition. Figure 1 shows some possible services and objects provided by a cell phone.

A lightweight mobile agent environment is used to manage the execution of dynamic processes within the network. The semantic service layer has been built on top of the NEMO mobile agent system, developed for deployment on Nokia handheld devices.

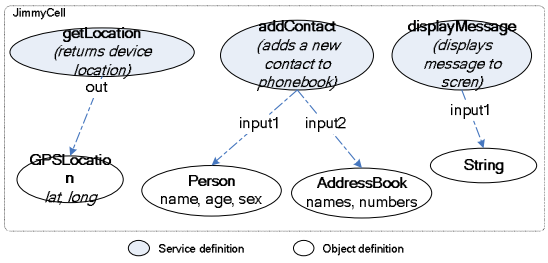


Figure 1. Ontological description of data and services provided by a cell phone.

We will first outline the mobile agent platform then show how a complex semantic layer can be deployed on this infrastructure.

2.1. Mobile Agent Environment

The execution environment follows a typical mobile agent paradigm. An agent is composed of a program description and mobile data, while each network node provides named services and local data.

The agent framework is based on the high level Wave language [13]. The coarse grained language constructs are designed for distributed process control, making this agent system perfectly suitable for use in mobile service coordination applications. The granular nature of the language allows compact agents to describe complex processes, and be deployed under an architecture simple enough to execute on lightweight devices. The NEMO agent system has been developed for the Java Mobile Edition and deployed on Nokia 6620 and 7610 handsets. The service architecture can also be deployed on sensor networks, on top of the MicroWAVE agent system [8] currently under development.

```
Algorithm UpdateMap:
    1.  Repeat
    2.      Read sensor value into MapPoint service argument 1
    3.      Read sensor location into MapPoint service argument 2
    4.      Execute in parallel:
                a.
                        Jump to known mapping device
                        Update the map point risk value.
                        Terminate branch.
                b.
                        Broadcast self to all sensor neighbours
    5.  Continue loop.

Agent code
RP(?gVal. Frma_1=Fr. ?gLoc. Frma_2=Fr.
    (#Fmapper. rma?mapPoint.!, #)).
```

Figure 2. Example of a mapping agent integrating sensor network with mapping device.

The language allows data to be accessed directly through named variables, and services are called through a special system call interface. An example of the language is shown in Figure 2. Miscellaneous services such as list manipulation and mathematical functions can be standard in all the environments and available in all nodes, while specialized services exist only in specific nodes. An agent can store complex objects, whose data properties are directly accessible. Subprograms can also be defined within an agent, and can be invoked at any point during its execution as procedure calls.

The rest of the language constructs handle program flow control and code mobility. An agent may execute instructions sequentially or split its program for parallel execution, as the example illustrates. An

instruction is provided for transferring the agent execution to a different network location. Constructs also exist for conditional execution (if-then-else), repetition of a specific program block, or terminating an agent's existence.

The minimal agent system architecture is outlined in Figure 3. The network interface receives agents and places them in an agent queue. When the interpreter is ready to execute, it parses an agent from the queue, prepares the first instruction for execution and places the rest of the agent back in the queue. The instruction is then executed, an action which can cause application services or data to be accessed, the agent to transfer to another node, execution to expand (such as in the case of a local procedure call), or terminate an agent's execution.

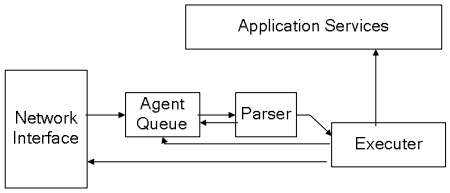


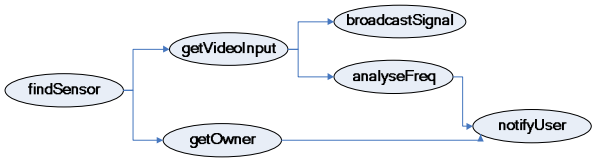
Figure 3. Mobile agent environment architecture.

Due to the richness of language constructs, and simplicity of the design, the architecture creates a scalable, modular lightweight infrastructure upon which complex services may be deployed and coordinated.

2.2. Semantic Layer

The agent framework uses metadata to describe semantics of the contents and capabilities of a device, an approach which has several advantages. The semantic details provide a rich description of content, facilitating complex queries and navigation. This description also creates easy integration of data and services between devices, allowing a human or automated agent to determine relationships between data and create meaningful compositions of services. It also allows autonomous agents to adapt to their environment as they navigate the network.

We will approach the details of service and data domains separately. Services may update the data on a device, perform analysis on an object, or interact with the user. Defining services as part of the same ontology as the data domain permits one to indicate the types of input and output objects, allowing the agent to easily identify which type of objects are affected by each service, and how objects may relate to each other through the use of services. Most importantly, the strong syntactic definition allows valid composition of services into complex programs. This facility combined with the mobile agent platform's several constructs for dynamic program control permits for complex service compositions to be defined within the implementation of an agent. Figure 4 shows an example of a constructed process, and its associated agent implementation code. Compositions can be saved as complex services within a host, and reused through a normal service invocation. It is possible to enhance network nodes to learn composite services by broadcasting new service definitions related to their neighbors' interests, or by monitoring agent traffic. Internal details of service composition and choreography will be further detailed in Section 2.3.



Agent code:

?findScanner. (r?getVideoInput. (r?broadcastSignal, r?analyseFreq. notifyUser?sync. r?notifyUser), r?getOwner. notifyUser?sincf).

Figure 4. A simple service composition and associated agent code.

Ontological descriptions of data allow device contents to be represented by objects, as in the object-oriented programming paradigm. Agents have fine grained control over object properties, and inspection or modification of data fields is possible with a single command. Moreover, ontological descriptions allow definition of subclass and superclass relationships between objects, allowing generalized service descriptions and data access.

2.3. Mobility and Composition

Agents in this framework navigate between network hosts and coordinate execution of services as specified in itinerary compositions. An agent holds data related to each service execution in its local variables, which transfer with the agent to subsequent services. To conserve an agent's data space, it is possible to explicitly discard data variables if it is known they will not be used by subsequent services, or automatically if this information can be deduced by analysis of the program itinerary. By analyzing a program itinerary, the system can also exploit the distributed nature of a task. An agent can automatically split execution into multiple parallel paths when inputs to a service can be separate execution branches. When an entity reaches a service whose inputs come from multiple branches, it may either wait for uncompleted branches to provide results or notify failure, or join with the existing results and execute the service.

Due to the strongly typed nature of service definitions, it is possible to create a service composition on demand, as long as contributing services are

discovered in advance. In some situations it is useful to have the capacity to compose programs on the fly, specifically when the user does not know what services will be available in the surrounding area, or what tasks will need to be performed in a specific situation. If intelligent service discovery mechanisms are available, it is even possible for agents to determine their own itinerary based on semantic information.

Consider the scenario of a plane crash in a mountain peak, where a sensor network was previously deployed for avalanche monitoring. Assume a search and rescue team comes on site and wishes to use the capabilities of the sensor network and any survivor's devices that may be available. Ontological description of possible device contents are shown in Figure 5.

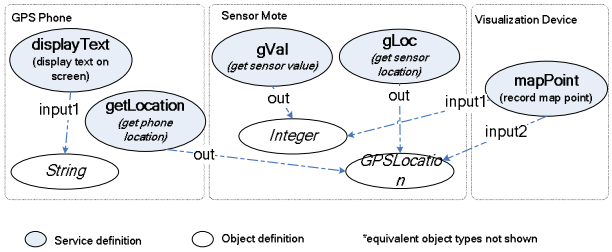


Figure 5. Integrated ontologies in search and rescue scenario.

A simple area mapping agent is built using the ontological description from the sensor network, as was presented in Figure 2. The agent is broadcast to all sensor motes, gathers their readings and locations by invoking appropriate services, then to a visualization device which produces a map for the rescue coordinator. The mapping device can now provide this program as a regular service to the network, along with other compositions such as finding a safe path between two points, or monitoring the area and alerting users if they are entering zones of high risk.

Algorithm RescueWounded:

1. Find a wounded person
2. Find an available medic
3. Using sensor network, compute a safe path between wounded and medic.
4. Find a transport vehicle
5. Using the availability of the vehicle and medic, schedule a transport to the wounded.
6. Execute in parallel:
   a. Reserve space in the hospital and notify the staff.
   b. Notify the person that a medic will be arriving.

Figure 6.a. Algorithm for scheduling medical treatment of wounded person.

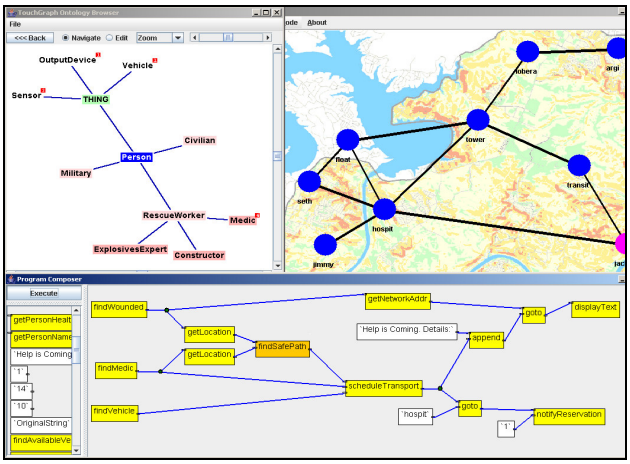


Figure 6.b. The management user interface provides tools for network topology visualization (right), ontology management (left), service composition (bottom).

As more devices are discovered, a variety of data and services become available for integration. Equipped with a visualization tool and flexible program composition capabilities, a rescue coordinator can immediately take advantage of the network information. Figure 6 outlines an algorithm for finding and scheduling transport and medical care for a wounded person, while the corresponding user interface shows the resulting program composition. The user interface pictured was developed for a high-end coordination device, and is intended for ontology visualization and manipulation, service composition, and network monitoring. The composition module tracks available network services and provides a trivial click-and-drag interface for programming service coordination agents.

3. Conclusions and Future work

We have presented the NEMOS system architecture and prototype. The system combines the existing NEMO mobile agent technology with ontological descriptions of network device data and services, to enable a service oriented architecture for lightweight devices such as handheld phones and sensor networks.

Further research will be focused on adding semantic data to services and implementing a lightweight overlay, such that agents can autonomously discover and compose an itinerary based on predefined inputs and outputs. An agent will then be able to navigate through areas unknown at program composition, or modify its behavior without user reprogramming.

A future difficulty in deploying over a large variety of hosts is that increased heterogeneity of

ontologies will make it challenging to integrate network data and services. If ontologies become diverse, the GODIS [14] system provides a mechanism of aggregating objects from different ontologies into a single unified entity without standardizing on a single unified ontology.

Optimizations to the agent system, such as communication compression, agent conversion to bytecode, and a lighter system footprint can also be investigated if necessary. Deployment of the system in a large scale network and measuring processing speeds for various tasks will be done in the near future.

4. References

[1] Web Services Architecture: http://www.w3.org/TR/ws-arch/

[2] Web Services Description Language: http://www.w3.org/TR/wsdl

[3] SOAP V1.2 Specification: http://www.w3.org/TR/2003/REC-soap12-part0-20030624/

[4] UDDI Specification: http://www.oasisopen.org/committees/tc_home.php?wg_abbrev=uddi-spec

[5] Business Process Exection Language Specification: http://www-128.ibm.com/developerworks/webservices/library/specification/ws-bpel/

[6] OWL-S Specification: http://www.daml.org/services/owl-s/

[7] Semantic Web Services: http://www.daml.org/services/

[8] S. Gonzales-Valenzuela, S. Vuong, V. C. M. Leung. "A Mobile Code Platform for Distributed Task Control in Wireless Sensor Networks," MobiDE 06, pp 83-86.

[9] Watts, D.J., Strogztz,S,: Collective dynamics of 'small-world' networks. Nature 393 (1998).

[10] V. Baousis, E. Zavitsanos, V. Spilipoulos. "Wireless Web Services using Mobile Agents and Ontologies," ACS/IEEE International Conference on Pervasive Services, June 2006, pp 69-77.

[11] F. Ishikawa, N. Yoshioka, Y. Tahara, S. Hondien. "Mobile Agent System for Web Services Integration in Pervasive Networks". http://honidenlab.ex.nii.ac.jp/record/2004/W8_1173_Ishikawa.pdf

[12] Z. L. Liang, K. W. Raymond. "A Lightweight Mobile Platform for Business Service Networks," ACM International Conference Proceeding Series 2005; Vol. 87, pp 12.

[13] G. Hackmann, M. Haitjema, C. Gill, G. Roman. "Silver: A BPEL Workflow Process Execution Engine for Mobile Devices". http://cse.seas.wustl.edu/techreportfiles/getreport.asp?525

[14] J. Li, I. Radu, S. T. Vuong. "GODIS: Ontological Resource Discovery and Integration in Grids". IASTED PDCS, 2006.

[15] Sapaty, P. "Mobile Processing in Distributed and Open Environments". *John Willey & Sons*, 2000.

[16] Fok, C.-L., Roman, G.-C. and Lu, C. "Rapid development and flexible deployment of adaptive wireless sensor network applications," Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'05), Columbus, USA 2005.

[17] Liu, T. and Martonosi, M. "Impala: A middleware system for managing autonomic, parallel sensor systems," *Proceedings* of ACM SIGPLAN Symposium on Principles and Practice of *Parallel Programming*, San Diego, USA 2003.

[18] Boulis, A., Han, C.-C. and Srivastava, M. "Design and implementation of a framework for efficient and programmable sensor networks," *Proceedings of the First* International ACM Conference on Mobile Systems, Applications and Services, San Francisco, USA 2003.

[19] Kang, P. et. al. "Smart messages: A distributed computing platform for networks of embedded systems," *The Computer* Journal, Special Issue on Mobile and Pervasive Computing, Oxford Journals, 47(4):475-494, 2004.