Towards Ad-hoc Multimedia Information Exploitation via Web Services and Mobile Agents

Ben Falchuk

Telcordia Technologies, One Telcordia Dr., Piscataway, NJ 08854
email: bfalchuk@research.telcordia.com

Abstract

In typical distributed systems it is not unusual to find Web Services (or more generally, Service Oriented Architectures) supporting information exchange, transactions, and tasking. Similarly, it is not unusual in domains such as manufacturing, aerospace, and military, to find an infrastructure for software agents allowing flexible, scalable, autonomous computing and tasking. This paper describes a framework which begins to enable the best characteristics of both of the above scenarios. Applicable in distributed systems such as enterprises and Internet, the framework enables Web Services to be deployed in ad-hoc fashion on networked computing devices by taskable mobile agents. Once deployed, the Web Service acts as a facade to the node's stored multimedia data (e.g. documents, images, songs) allowing other 'clients' to exploit it, either transparently or quite visibly to the node's user. Exploitation of this multimedia data can be anything from mining business intelligence, to image sharing, to alerting and entertainment. At a later time the ad-hoc service moves off the device or terminates. This paper addresses some of the salient issues with respect to software and open standards for this type of ad hoc services. We also describe results from a working prototype.

Keywords: mobile agents, Web Services, enterprise data mining, multimedia information

1. Introduction and Motivation

There are many realistic scenarios in today's distributed multimedia systems which call out for what we describe in this paper as Ad-Hoc Services (AHS); that is, those that are lightweight, agile, and provide *transient* but important access to stored information via well-established Web Service protocols – i.e. namely the Simple Object Access Protocol (SOAP [2]). Ad hoc services transiently inhabit corporate desktops and mobile devices such as cellular phones; all the while, their lifecycles are mostly transparent to the owner of the device on which they operate[1]. Ad hoc services are desirable when:

- There is no central server indexing distributed multimedia contents of remote devices
- There is some competitive advantage that can be gained from remote data exploitation
- A "thin" transparent approach is desirable

In the peer-to-peer world, file-sharers make up what some call the transient Web. Unlike 'permanent Web' nodes, transient ones come and go in unpredictable fashion with variable network addresses. In the mobile agents realm, a special-purpose software infrastructure in combination with a communications protocol allow autonomous software entities (each with some goal, a plan, and policies admitting interaction) to migrate from node to node while coordinating with other resources to achieve a goal.

The AHS approach (explored in this paper), in which transient Web Services are transported and 'remotely controlled' by mobile agents, enables powerful new information exploitation opportunities. Such an approach is also realistic; PC and cellular phone OS's are now capable of running (small-footprint) agent-environments and Web Servers. Furthermore, most of these computing devices ship with several built-in multimedia applications such as calendars, IM and cameras. Ad hoc services, therefore, have a rich 'playground' of untapped multimedia service data.

AHS has followed from our past experience with Agent-based NGN Services integration [1]. Our principle design requirements included:

- Make information exploitation more systematic (and transparent).
- Serve as a multimedia data exploitation tool in deployments where no central server indexes device multimedia data.
- Exploit Web Services (e.g. SOAP) and agent software and standards [3]
- Lightweight installation, in terms of memory footprint and complexity

[1]The term 'ad hoc applications' has been used to different effect by other researchers

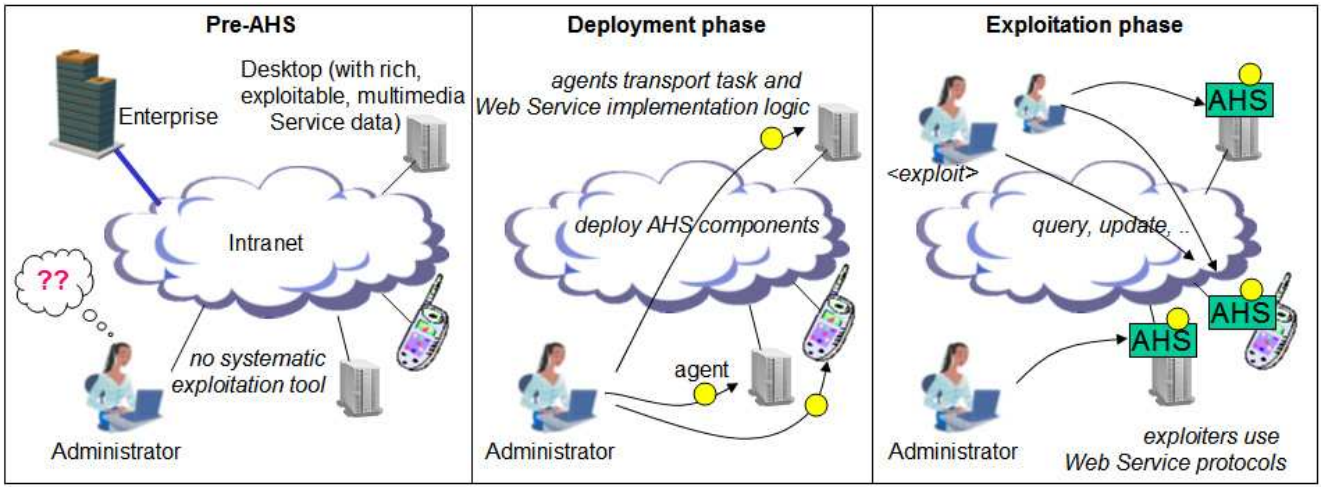

Figure 1. Conceptual idea of AHS ("setup" and "teardown" phases not shown)

2. AHS Architecture

Figure 1 illustrates the AHS paradigm and benefits. Two themes to AHS are: *agility* – meaning easy to run and cross-platform compatible - and *lightweight* – meaning small memory footprint and not CPU-intensive. The AHS Framework architecture supports the following main stages of ad hoc services:

- Service creation[2]
- Deploying the service to a remote Device
- Controlling the service's lifecycle
- Providing access to the service (including controlling security and privacy[3])
- Subsequent exploitation of the Device's data

Service *creation* is the stage in which the AHS Developer or Administrator develops both the API and implementation of the service. For example, if an AHS service called 'Microsoft Office Miner' is to provide SOAP-based access to a remote device's *Microsoft Office* multimedia Service Data, then the developer writes the implementation logic in Java, gathering all required Java (lightweight) libraries. This implementation might exploit technologies such as IBM's *InterfaceTool* [4].

Service *deployment* is the process by which a mobile agent transports the 'Microsoft Office Miner' implementation, along with a Web Services SOAP server and all supporting libraries, to the Device/PC on which it is to run. Devices must be on a TCP/IP network and running a JADE server (which can be setup to run as a system 'service'). Administrators need not write a new mobile agent for each task. Instead, they may use the AHS *Controller GUI* (see Results section) to specify parameters. Figure 2 illustrates the important architecture elements, showing the *Controller* and *Device* on a distributed JADE platform (requiring a Java Runtime Environment and TCP/IP stack). The dark components in the Figure are agent-related, while the dashed ones are Web Service related. Once resident on the device, the *Mobile Agent* instantiates and controls the lifecycle of a *SOAP Server* and a *Service Implementation* (ServiceImpl) which behaves according to the Administrator's design. Multimedia service data resident on the device (e.g. calendar appointments, photos) are accessed natively via the ServiceImpl.

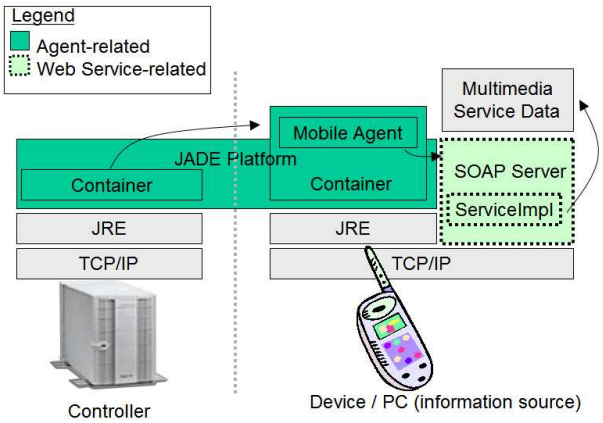

Figure 2 AHS Architecture view

The Service's *lifecycle* can be controlled by the mobile agent on the device, and, accordingly, by the Controller (who can send the mobile agent messages). For example, if a human user at the Controller GUI wishes to immediately tear-down the 'Microsoft

[2] i.e. the service that façades a device-resident multimedia service

[3] 'ad hoc' does not imply that security will not be enforced

[4] *InterfaceTool* enables Java to COM communications

Office Miner' service on a particular device, the mobile agent resident on the device stops the process implementing the service, stops the process implementing the SOAP server (e.g. tearing down the web service), and cleans up (i.e. garbage-collection).

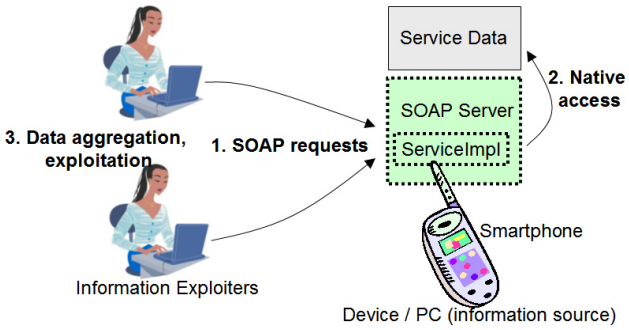

Figure 3. AHS access view

Ad-hoc Service *access* occurs from two fronts. Since the Device and Controller are 'federated' upon a distributed Agent platform, an Agent Communication Language can be used to directly message with the remote agent (only administrators use ACL's). Most exploiters, however, will use Web Services to interact with the AHS since the mobile agent initiates a SOAP Server onto the Device and also brings a Web Service's implementation (Figure 3).

3. Agility Aspects

'Lightweight' (small-as-feasible memory footprint) and 'agile' (mobility, life-cycle control, and cross-platform compatibility) are two attributes of the AHS Framework. To these ends, several software technologies can be exploited (as in our Lab):

- Tiny footprint SOAP Servers – XSOAP, eSOAP, Apache Axis *SimpleServer*
- Small footprint XML Parsers – XMLPULL API (xmlpull.org), MXPI, MinML (14Kb footprint)
- Mobile Agents – JADE agent platform, LEAP is a port of JADE to J2SE, J2ME and PersonalJava allowing agents to run on PDA's

The emergence of such technologies is a good indicator that software component agility is more and more important.

4. Web Services Aspects

Two important Web Services 'protocols' – 1) *Universal Description, Discovery and Integration* (UDDI) and 2) The *Web Services Description Language* (WSDL) – are key parts of the Web Services vision [2]. We have studied if and how these protocols co-exist with the AHS vision and have several findings; a few are summarized below:

- In UDDI, each registered business service has a bindingTemplate which in turn describes the endpoint (one of several types). We have found that it is best if a new `use-type` value called `adHocEndpoint` is defined and used to describe AHS services (i.e. it captures better the semantic of ad hoc services)
- WSDL 2.0 contains no built-in tags to talk about a Web Service's *start-time* and *end-time*, but such concepts are important for AHS ad-hoc services (e.g. they may be torn down quickly). We propose the use of WSDL `feature` and `property` tags to express this directly at the service description level.

4.1. Ontology

A machine parse-able ontology – or information model – is an invisible but key part of our approach. W3C's Ontology Web Language (OWL) is a sophisticated ontology language though lesser degrees of sophistication can be achieved through XML Schema and RDF-based ontologies. At any rate, modeling multimedia content is important in AHS; for example: (1) characterizing multimedia data and determining which native access methods apply, (2) since exploiters may be software agents results from queries such as "get all photos stored in the device home directory" must be self-describing and parse-able. A rich body of research studies multimedia models and meta-data; our ongoing approach is to exploit OWL and semantic Web technology.

5. Results

A distributed prototype version of the AHS Framework has been built in the Telcordia *Applied Research Lab*. It leverages the Sun Java JDK 1.4, JADE 2.6 Agent Environment, and various software toolkits discussed in previous sections. Figure 4 illustrates a simple AHS Controller tool which allows human Administrators to kick-off AHS scenarios. Once the tasks and agents are described in the tool, the agents are 'launched' (launched agents are shown on the right in Figure 4) and begin migrating to remote devices in order to install and manage Web Services (to façade Device multimedia data). This, in turn, enables the exploitation of any device-resident multimedia information. Note that the mobile agents themselves need not have graphical interfaces. Figure 5 illustrates some possible *effects* of having an AHS

ad-hoc service running on a user's Device (e.g. PC, cellular phone). In Figure 5 the AHS service deployed to the user's Device is allowing SOAP-based access to the Device's Microsoft *Outlook* multimedia data. The first message sent to the service from the controller (via a SOAP message) backs up the user's current week appointments into the controller. The next message clears the appointments for the current week. Finally, the next message sends a new set of appointment information to the user. Among other things, this could be a) the appointment set of some other user (e.g. an employee supervised by this user), or b) (slightly more frivolous) a sort of 'banner' message (in the Figure we see the letter 'M' being 'sent' to the Outlook Calendar. In another setup we used several (modest) Windows 2000 machines with 256Mb RAM, and JADE and Java JDK1.4 to test many capabilities of the agent-based AHS in load situations – e.g. one interesting result is that a single node of the AHS can host 50 or more agents, but it is the cumulative message-passing rate that is important. This is acceptable because a) it is Web Services invocations that dominate clients after the deployment and setup phases, and b) we do not expect lifecycle-type requests upon agents to reach 200 req/sec (the rate at which degradation is seen).

Figure 4. Controller spawns two mobile agents; each agent migrates to a Device and allows temporary SOAP-based access to the Device's Microsoft *Outlook* service data

6. Conclusions

Although there is much to be done towards achieving so-called Ad Hoc Services, the combination of computing trends (incl. Web Services) seems to indicate that it is both feasible and merits further research. Our experience with agent-based NGN services integration and with Web Services architectures has solidified this view.

AHS is *not* a Peer-to-Peer system, such as those for file-sharing, nor is it a Grid Computing platform (or single application such as SETI). Rather, AHS aims to provide transient access to multimedia data on devices attached to communications networks (e.g. next-generation networks or Intranets). AHS is much more agile and lightweight than OSGi and component mobility is an integral part of AHS while not permitted in OSGi. AHS could conceivably be built *upon* an OSGi framework. Our Ad-hoc Services Framework is novel because:

- Exploits inherent advantages of mobile code.
- Information gathering occurs transparently to (but authorized by) the device owner / user
- A transparent and flexible way to extract service data (that is not already shared on servers) provides valuable new scenarios. Enterprises achieve 'information awareness'.

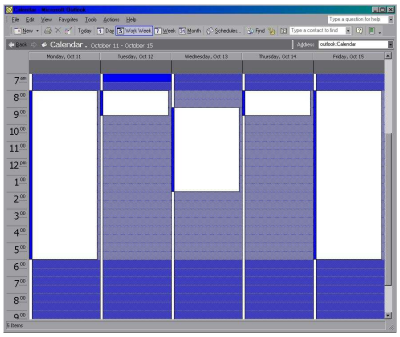

Figure 5. Effects of an AHS taking temporary control of a user's Calendar, clearing it, and then using the week-view as a 'banner' page (the letter 'M'). Later, the AHS 'restores' the original calendar appointments.

Ongoing AHS work includes the introduction of WS-Security (and related) capabilities as well as UDDI and WSDL extensions – and W3C's Ontology Web Language - to describe multimedia meta-data, ad-hoc services and their capabilities. Further study of OSGi technologies is also commencing.

7. References

[1] K.Cheng, B.Falchuk, et al, "An Agile Server for Multi-provider Service Peering and Aggregation", *IEEE Communications*, 41(3), pp. 126-136, 2003

[2] W3C Web Services, http://www.w3.org/2002/ws/

[3] M.Huhns, "Agents As Web Services", Internet Computing, 6(4), pp.93-95, 2002