A Pluggable Service-to-Service Communication Mechanism for VNA Architecture

Jin Nakazawa  
jin@ht.sfc.keio.ac.jp

Yoshito Tobe  
tobe@mkg.sfc.keio.ac.jp

Hideyuki Tokuda  
hxt@ht.sfc.keio.ac.jp

Keio University  
Graduate School of Media and Governance  
Endo 5322, Fujisawa, Kanagawa, Japan

Abstract

This paper proposes a middleware for home networks, called Virtual Networked Appliance (VNA) architecture, in which the service description method and the Service-to-Service (S2S) communication mechanism are separated in an orthogonal way. Through the separation, VNA architecture solved the following two problems of existing middleware technologies: aspect violation and middleware fragmentation. In this paper, we first clarify the two problems and their relationship. Then, we describe the proposed middleware architecture as a solution from the viewpoint of the overall configuration and the S2S communication mechanism.

1. Introduction

Home networks consist of various information appliances including computers, sensors, and consumer appliances. In this research, we are aiming at increasing usability of home networks, by reducing complexity of information appliance operation.

Many researchers have been investigating middleware which integrate distributed services on home networks. Each middleware provides its original service description method and Service-to-Service (S2S) communication mechanism. Generally, the former is provided as API, and the latter as the modules which implement communication protocols. However, in existing middleware technologies, the service description method and S2S communication mechanism are tightly coupled.

We propose a middleware, called Virtual Networked Appliance (VNA) architecture. VNA architecture facilitates information appliance coordination through an orthogonal separation of the service description method and S2S communication mechanism.

2. Problems in S2S Communication

Fixing the S2S communication protocol in a middleware violates realization of a service specific requirement for messaging: aspect violation problem. Suppose two services which have the aspects “reliable event exchange,” and “real-time stream transfer,” respectively. The former service can be implemented atop any of Jini, UPnP, and ADS[2]. However, the latter service cannot be due to the mismatch between the service’s aspect and their S2S communication mechanisms’ facilities. Programmers can hardly specify any timing constraints for RMI/TCP/IP protocols, nor implement any aspects that are out of semantics of the communication protocols adopted in a middleware.

Services on a home network are fragmented into multiple heterogeneous middleware technologies: middleware fragmentation problem. On event-centric middleware, such as SIENA[1], non-event-oriented services cannot be implemented without the programmers’ effort to build bulk and stream communication mechanisms from scratch. In the same manner, data-centric middleware, such as ADS, cannot host event-oriented services. This phenomenon imposes on users a complex task dealing with multiple middleware technologies simultaneously.

3. VNA Architecture

To cope with the abovementioned problems, we propose a middleware called Virtual Networked Appliance (VNA) architecture[4]. We extracted the S2S communication mechanism from the service description method, and enabled service programmers to specify their own aspects. Our systematic approach for realizing these characteristics is providing various off-the-shelf protocol stack modules and enabling them to be dynamically loaded. Programmers specify aspects as protocol names above the network layer.
Table 1. Summary of supported information appliances

<table>
<thead>
<tr>
<th>name</th>
<th>vendor</th>
<th>Serdget</th>
<th>code</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDP 502M (Plasma Display)</td>
<td>Pioneer</td>
<td>PDP</td>
<td>98 lines</td>
<td>3.55KB</td>
</tr>
<tr>
<td>DSP AX-1 (Amplifier)</td>
<td>YAMAHA</td>
<td>AAVmp</td>
<td>147 lines</td>
<td>4.47KB</td>
</tr>
<tr>
<td>DA100 (Thermometer)</td>
<td>Yokogawa</td>
<td>DA100Thermometer</td>
<td>179 lines</td>
<td>7.75KB</td>
</tr>
<tr>
<td>WV-DR9 (Digital VCR)</td>
<td>Sony</td>
<td>DVdeck</td>
<td>82 lines</td>
<td>3.26KB</td>
</tr>
<tr>
<td>EVI-D30 (Video Camera)</td>
<td>Sony</td>
<td>SonyEVID30Controller</td>
<td>104 lines</td>
<td>4.36KB</td>
</tr>
</tbody>
</table>

We adopted XML for the specification to enable programmers to change aspects without re-compiling service codes. Next, we define the service description (Serdget) API and the S2S communication (Port) API separately. Multiple protocol modules are provided, which are accessed via common Port API. The advantage of our S2S communication mechanism is that we support simultaneous use of multiple protocol modules in a pluggable manner.

Throughout these configurations, the VNA architecture has solved abovementioned problems as follows.

- Since the S2S communication protocol can be selected by service programmers, our architecture does not violate their aspect implementation. Instead, by providing various off-the-shelf protocol modules, VNA architecture facilitates aspect realization.

- Since our architecture does not suffer from the aspect violation problem, services with diverse aspects can be implemented. As a result, heterogeneous services can be provided for our architecture.

VNA architecture, implemented with Java language, consists of four main mechanisms: Serdget instantiation, inter-Serdget communication, Serdget directory, and VNA mapping. The current implementation supports information appliances shown in Table 1. Each row in the table includes the name of the Serdget specific to the corresponding information appliance.

4. Related Work

There are a few implementations of CORBA which are free of the aspect violation problem, such as TAO[6] and IONA[3]. TAO leverages the Pluggable Protocols Framework[5] to support timing constraints for CORBA services. In the framework, a service conveys a specific profile to the CORBA runtime. Service programmers can realize the aspect of the service to implement, by describing an appropriate profile. The protocol modules are for enabling service programmers to choose CORBA transport; i.e. each programmer can define their own upper layer protocol as an interface shared by both stub and skeleton objects. Our S2S communication mechanism, on the other hand, enables service programmers to choose the well-known upper layer protocol, while the network layer protocol is fixed to IP. This difference results in broad aspects of services to be provided for our architecture keeping backward/forward compatibility.

5. Conclusion

In this paper, we proposed the middleware called Virtual Networked Appliance (VNA) architecture, which solves aspect violation and middleware fragmentation problems.

To cope with these problems, we separated the service description method and the S2S communication mechanism in our architecture. Also, we enabled programmers to choose communication protocols for each services. Programmers are encouraged to build heterogeneous services on our architecture, since various dynamically loadable off-the-shelf protocol modules eliminate their effort to implement the service-specific data transmission protocol stacks. Users, then, can utilize information appliances flexibly through well-integrated heterogeneous services on the VNA architecture.

References