The Design of an XMPP-based Service Integration Scheme

Feng-Cheng Chang and Duen-Kai Chen
Department of Innovative Information and Technology, Tamkang University, TAIWAN
Email: {135170, 134330}@mail.tku.edu.tw

***Abstract*—The advances in network-enabled software development raises the demands on the service oriented architecture (SOA). Popular implementations of SOA uses XML-based representations, such as SOAP and WSDL, to enable remote function invocation and service discovery. Our previous experiences show that a sophisticated XML representation is often over-complex for a controlled environment. We also found that the application composition is a high-level operation. Therefore, a proper infrastructure for hosting and delivering various kinds of services is required. We then proposed an XMPP-based solution to overcome these issues. A service is an XMPP client, and focus on how to interact with the other clients; the management of the service can be flexibly arranged through the proposed format of its full JID; and the fundamental security are handled by the XMPP server and the publisher-subscriber pattern.**

***Keywords*-extensible messaging and presence protocol; XMPP; service-oriented architecture; SOA**

I. Introduction

The advances in network-enabled software development raises the demands on the service oriented architecture (SOA). In essence, a service is a software component that provides a specific functionality to a range of service users. A common principles of service design is to be self-contained and independent of specific applications. Therefore, we can combine different services to form our application flexibly. I. Sommerville defines SOA in [1] as *"Service-oriented architectures (SOAs) are a way of developing distributed systems where the system components are stand-alone services, executing on geographically distributed computers."* The additional "distributed" property implies the using of standardized protocols. A common implementation approach of SOAs is to adopt standard XML-based protocols, such as SOAP and WSDL, to support platform and language-independent information exchange. Because an application is constructed by composing the required services, there are two major tasks to be accomplished before the application is made available. One is to discover the necessary services locally and externally, and the other is to establish the connections among the services. Because services are published with well-defined APIs and the interactions of an application is implicitly specified in the main algorithm, the latter (communication among services) is not difficult to achieve. On the contrary, the run-time discovery of a service poses several issues. Although there are many standards for service description, it still depends on the underlying protocols and architectures to obtain the descriptions. In this paper, we will design a scheme of hosting services on an XMPP network, which meets the requirements of simple application composition.

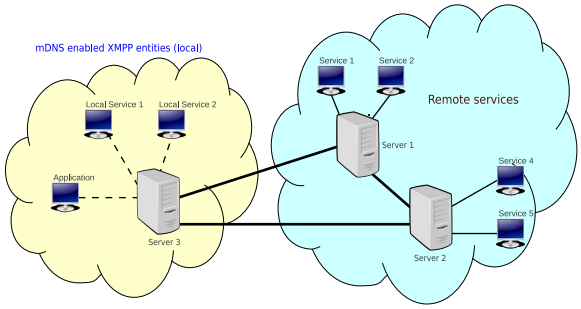

Figure 1. An XMPP network

This paper is organized as follows. In Sec. II, we briefly overview our previous works that address the high-level service integration of a future classroom. In Sec. III, we describe the service discovery issues in a controlled environment. In Sec. IV, we design the detailed scheme to achieve the requirements. Then we conclude our work in Sec. V.

II. Our Previous Work

We will briefly describe our previous work in this section. The first is the study of the Extensible Messaging and Presence Protocol (XMPP) and its usages. The second is a service-based scheme for integrating several hardware devices to form an effective whiteboard system.

A. XMPP

The *extensible messaging and presence protocol (XMPP)* [2][3] is a technology specified by the IETF for real-time communication. It is based on the XML format for exchanging information. More specifically, it provides a stream-like method to send small pieces of XML from one entity to another [4]. In our previous study [5], we proposed that XMPP would be adopted as the high-level communication protocol of a software service.

As shown in Fig. 1, a typical XMPP network is organized as a *federation*. Each client is identified in a format like E-Mail addresses. A server is responsible for authenticating its clients. Once a client is authenticated, it is ready to obtain the presence of its buddies and exchange messages with them.

978-0-7695-4517-2/11 $26.00 © 2011 IEEE
DOI 10.1109/IIHMSP.2011.45

The presence information and the message delivery rely on the server-to-server communications. Presence and instant messaging are the two fundamental functions that XMPP is designed to achieve. In XMPP, two mutually authenticated users automatically subscribe to each other's presence. Each user can update his own presence status, and the servers implicitly handle the propagation of the change. According to the publisher-subscriber pattern, all the subscribers will be notified with this change. There are two approaches for exchanging data between two users. A *normal message* is a one-way data submission from one user to the other. If bi-directional data exchange is required, an initial *chat message* would create a session for further related messages. A multi-party communication can be established based on the *groupchat message* (or the more modern *multi-user chat (MUC)* extension) and the corresponding server components.

In addition to text messages, XMPP support binary data transfer through extensions. We may use *bits of binary (BOB)* or *in-band bytestream (IBB)* to transfer small amount of binary data. For bulk data, it is better to use one of the out-of-band protocol extensions, such as *Jingle*, to efficiently encode and transfer the stream.

In the study of [5], we found that XMPP is suitable for integrating high-level services, especially when the application requires ad-hoc services. In such a dynamic environment, a service should be allowed to publish its availability to the users, and the composition of an application should mostly rely on run-time discovery.

Based on the above concepts, we developed two prototypes. In each prototype, we implemented a finite-state service for reserving and scheduling a resource. We learned a few points from the implementation experiences:

- Although there is a service discovery extension of XMPP, it is not very useful when we implement a service as an XMPP client instead of a server component. We can directly identify the necessary service by its Jabber ID (JID).
- Although a sophisticated service definition language is applicable to a wide variety of service descriptions, a simple string is enough for identifying the capability of a service in a controlled environment. The reason is that we have some *a priori* knowledge about the application and the services.

B. Digital Whiteboard

The design described in this paper is directly motivated by the construction of the interactive whiteboard application [6]. A whiteboard (or blackboard) is an important tool for a teacher to broadcast the teaching materials to the students in the classroom. It enforces the students to pay attention to the location of the teacher, and the teacher can see the reaction of the students. Due to the implicit two-way communications, it is not appropriate to replace a whiteboard with personal displays.

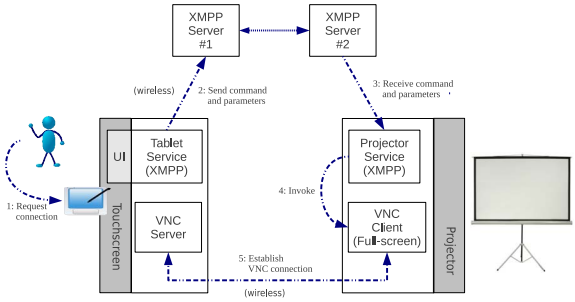

Figure 2. The scenario of the interactive whiteboard

The idea of "a future classroom" [7] has attracted wide attention from academia and practitioner. Based on the advances in various computer technology, it usually refers to a classroom equipped with computers, projectors and wire/wireless broadband connections, aiming at providing computer generated presentations and Internet content. A future classroom thus provides enhanced experiences of receiving spoken, graphical, audio and video content. Because a whiteboard is an indispensable component in a classroom, an enhanced version is usually defined as a large-sized multimedia touchscreen in a future classroom. It is capable of displaying multimedia content and receiving interactive events from the students.

To implement an interactive whiteboard (IWB), the expensive approach is based on a large-sized touchscreen. In [6], we proposed an alternative approach. It is based on the commonly available devices in a multimedia classroom, such as the controlling computer, the projector, and the wireless network. With an additional tablet computer, we can construct an effective IWB. The integration scheme is shown in Fig. 2. In this scheme, we wrote an XMPP client as the wrapper of the projector service. The teacher's tablet is the actual device for display and interaction. The open-source VNC solution is used to synchronously export the tablet display to the projector system. The role of XMPP in the scheme is to carry the control parameters sent from the tablet to the projector wrapper. The wrapper then invokes the VNC client to establish the connection.

By constructing this IWB application, we learned that:

- It is not difficult to provide an XMPP service for representing a hardware device.
- The controlling commands can be easily delivered as XMPP messages.
- More devices can be integrated to enable advanced classroom applications. For example, audio/video recorders can be incorporated to provide real-time multicasting service or video-on-demand service.
- The actual data exchange among devices does not necessarily implemented using XMPP.
- We need a platform for hosting services (at least the controlling parts), so that we can lookup the necessary

devices and integrate them to form the application.

III. Requirements

Service-oriented architectures has been popular for a few years. Some of the related research topics and standards are also developed. One of the specifications for describing a service is the *Web Service Definition Language (WSDL)*. However, our experiences show that WSDL may be overkill for a simple application, especially when being in a well-organized and controlled environment. Therefore, a string is enough for describing a service in a simplified case. For example, we can define a structural string to describe a service in a classroom: *classroom ID, service location, service type, service enumerator*.

In addition to the service description, we need a negotiation method for discovering a service. Some of the approaches are list below:

- Broadcast a discovery message in the local network, and obtain the response messages from the services.
- Send a discovery request to the registry service, and obtain a list of registered services.
- Delegate the discovery to an agent. The agent will resolve the criteria and find the matched services.

No matter which approach we choose, the conceptual negotiation steps are: obtaining the list of services, matching the criteria for the services, and reporting the candidates for application construction. When both local and remote services are considered, the first step becomes complicated. Sometimes, it is also difficult to discover local services in a complex network configuration.

Similar to public network servers, services should be protected by a certain access control mechanism. Otherwise, malicious users can utilize the services for their own purposes. Depending on the organization-wide policies, different security mechanisms would be enforced. We leave this as an implementation issue and only design a platform that can be adapted to work with different mechanisms.

While we protect a service from misuse, we also would like to keep the flexibility to publish it to the potential users. One of the obstacles is firewalls. A firewall is commonly used for protecting enterprise intra-net from the outside attacks. However, it also restricts the communication to a service hosted in the intra-net.

IV. The Architecture

According to the aforementioned discussions, we would like to focus on the design of an XMPP-based service integration scheme. Due to the variety of services, we mainly use the XMPP mechanism as the hosting platform for services. That is, the design will cover the functionality of publishing and locating a service.

A. The Format of Service Identifier

In a controlled environment, we assume that the location of a service is known and the capability can be expressed as a simple string. The full Jabber ID format is `account@server/resource`. To identify a service, we use the following structure:

- The **service controller** (the owner or the administrator) is represented by the bare Jabber ID part. The `account@server` is similar to an E-Mail address, and it is natural for representing an entity that manages a collection of services.
- The location, capability, and the enumerator is represented by the resource part. An XMPP server allows multiple logins of a user as long as the resource strings are different for different logins. We combine the following string fractions with the given delimiter (say, the underscore character) to form the resource string:
  - The **location** string: A camel-case word to represent the location.
  - The **capability** string: A camel-case word to represent the supported functions.
  - The **enumerator** string: A natural number to distinguish different instances of the same capability.

Sometimes the service controller is correlated to the location or the functionality. How to choose a proper granularity of the service controller is an administrative issue. There is no universal policy to make the "right" decision. For example, we have only 10 classrooms and there are no more than three service devices. In this case, a single service controller is enough for managing all the devices. Therefore, we may have a projector identified as `srvcs@ly.tku.edu.tw/CL323_LcdProjector_1`.

B. The Presence Priority

According to the XMPP specifications, an XMPP client can optionally specify a priority number (-127 to $+128$) when logging into the server. A higher priority means that the resource is more likely to receive a message sent to the bare JID. A negative priority indicate that the resource never receives a message to the bare JID.

In our scheme, this property makes it easy to specify the default service under a service controller. The default service may be the most commonly used service in the location, or a service that requires human interactions. Alternatively, we can implement a service gateway as the default service. It receives the service discovery requests and responds with sophisticated discovery results, e.g., in WSDL. The advantage is that we can use the single-string based identification for convenience while keeping the interoperability with other systems via the service gateway.

C. The Security Concerns

As mentioned in Sec. III, we do not specify the detailed security mechanism in the integration scheme. Basically most of the cryptographic algorithms and data exchanging scenario can be implemented to use XMPP with in-band/out-of-band data transmission channels. In this section, we discuss the fundamental protection provided by the XMPP.

1) XMPP-Level: XMPP supports negotiating the stream options. In most of the implementations, the `<starttls>` element enables the *Transport Layer Security (TLS)* [8] encryption of the stream. With this transport-layer encryption, an XMPP application transparently sends and receives data without explicit cryptographic functions calls. Of course, you may disable this feature to achieve better communication efficiency when working in a trusted network. We may also use the `<mechanism>` element to specify the authentication method. One of the popular mechanisms is the *Simple Authentication and Security Layer (SASL)* [9]. It supports various authentication methods: PLAIN, DIGEST-MD5, SCRAM, EXTERNAL, GSSAPI, ANONYMOUS, etc.

2) Service-Level: When composing XMPP-based services in our scheme, the publisher-subscriber pattern enforced by the XMPP server provides the fundamental service-level security. A service publishes itself on a server after the authentication. It may control whether to serve a specific user or not by accept or decline the subscription request. A service user may discover services after authentication. Once it discovers a new service, it sends a subscription request to the service and waits for the authorization. The design pattern provides not only the efficiency of event communication, but also the extensible service access scenario. A service actively controls to whom it provides the functions; and an application actively controls whose function is included in the composition.

V. Conclusion

In this paper, we reviewed our previous work and found a few application design issues based on SOA. The first point is that it is not necessary to use a sophisticated definition language to describe a service in a controlled environment. The second point is that the application composition is a very high-level operation. We need a platform for hosting services and exchanging various kind of data. The third point is that the platform should be extensible to meet security mechanisms and application protocols.

We then proposed that XMPP network could be adopted as the infrastructure of our platform. The XMPP provides the stream encryption and the default authentication mechanism as the fundamental protection. The publisher-subscriber pattern provides the basic service-level authorization. To implement a service, we write an XMPP client which calls the corresponding local function to process the input messages and send the results as the output messages. To lookup a service, we can use the full JID to locate the desired service. The JID can be broken into several components: the bare JID part represents the service controller and the hosting server; the resource part represents the location of the service, the capability of the service, and the enumeration number. Alternatively, we may construct a service gateway to responds to the external requests (e.g., WSDL). The service gateway is logged in as the highest non-negative priority client. Requests sent to the bare JID will be redirected to the service gateway implicitly, and this increases the level of interoperability to external systems.

Acknowledgment

This work was partially supported by the NSC, Taiwan, under Grants NSC 99-2221-E-032-050. We also appreciate the efforts of the following students in their XMPP-based senior projects: Wen-Hsin Chiang, Shih-Han Lin, Shan-Shan Hsu, Szu-Wen Tsao, Yi-Cheng Tsai, Feng-Yi Hsiao, Ching-Hua Li, Hsiang-Chi Tsai, and Chin-Yuan Chang.

References

[1] I. Sommerville, *Software Engineering*, september ed. Addison-Wesley, 2011.

[2] P. Saint-Andre, "Extensible messaging and presence protocol (XMPP): Core," Internet Engineering Task Force (IETF), Request for Comments 6120, March 2011.

[3] ——, "Extensible messaging and presence protocol (XMPP): Instant messaging and presence," Internet Engineering Task Force (IETF), Request for Comments 6121, March 2011.

[4] P. Saint-Andre, K. Smith, and R. Tronçon, *XMPP: The Definitive Guide*. O'Reilly, April 2009.

[5] F.-C. Chang, "A scheme for network service integration by XMPP," in *2010 E-Tourism*, Yilan, Taiwan, June 2010, pp. 55–64.

[6] F.-C. Chang and D.-K. Chen, "An open-source enabled scheme for improved interactive whiteboard," in *The 4th International Conference on Ubi-media Computing (ADET Workshop)*, Sao Paulo, Brazil, July 2011, accepted.

[7] L. R. Winer and J. Cooperstock, "The "intelligent classroom": changing teaching and learning with an evolving technological environment," *Computers & Education*, vol. 38, no. 1-3, pp. 253–266, January-April 2002.

[8] T. Dierks and E. Rescorla, "The transport layer security (TLS) protocol version 1.2," Internet Engineering Task Force (IETF), Request for Comments 5246, August 2008.

[9] A. Melnikov and K. Zeilenga, "Simple authentication and security layer (SASL)," Internet Engineering Task Force (IETF), Request for Comments 4422, June 2006.