2008 IEEE International Symposium on Service-Oriented System Engineering

A Gateway Design for Message Passing on The SOA Healthcare Platform

Chi-Lu Yang[1, 2], Yeim-Kuan Chang[1], Chih-Ping Chu[1]
[1] *Department of Computer Science and Information Engineering, National Cheng Kung University*
[2] *Networks and Multimedia Institute, Institute for Information Industry*
[1,2]*Tainan, Taiwan R.O.C.*
stwin@iii.org.tw, ykchang@mail.ncku.edu.tw, chucp@csie.ncku.edu.tw

Abstract

Various security and privacy issues have emerged from the SOA healthcare platform. Large amounts of personal data are transmitted daily through the SOA healthcare platform. For security and privacy reasons, the exchanged data should be encapsulated and encoded by a specific standard, a customized standard even. In this study, we first focused on investigating the importance of data exchange and message passing on SOA from the security and privacy viewpoints. Thereafter, we designed a gateway for passing message in the SOA healthcare platform. Subsequently, we initially pointed out the interface utilities on the SOA healthcare platform. Health data format and health packet format were then defined. Finally, the transmission mechanism between the sender and the receiver was exposed. As the practical results in beta-test, the messages could be passed in a secure and reliable routing. These customized messages were able to successfully avoid recognition when they were intercepted by the other P2P streaming tools.

1. Introduction

As Information and Communication Technologies (ICT) have continued to advance, the applications of ICT in different aspects of our lives such as work and entertainment have also evolved. Information Communication Technologies have gradually seeped into our daily lives through network techniques that allow computers to provide many remote services. Interestingly, a large number of innovative service models such as YouTube, WRETCH, Google, and Amazon, among many others, are continuously emerging. The use of ICT to create innovative service models has allowed many innovative service models to extend famous enterprises worldwide.

The application of ICT has become a worldwide trend. Infusing Service-Oriented Architecture (SOA) to provide common activities and interests such as dining, medicine, lifestyle, traffic, education, and entertainment has also become one of the popular methods in different industries. Through the SOA platform, we could integrate individual providers into similar service processes. Modeling distinct business providers could provide services in the specific domain. Undeniably, this is an interesting work. Through this analytic process, researchers would be able to communicate with various stakeholders as well as design systematic platforms. Data exchange and message passing are both key issues in SOA-based systems. In this study, we first focused on investigating their importance on SOA from the security and privacy viewpoints. Thereafter, we designed a gateway for passing message on the SOA healthcare platform.

2. Related works

Service-Oriented Architecture is a software architectural style that is employed for realizing and constructing business processes, which are composed of components as services ([1], [2], [3]). Service-oriented technology could extend ICT to provide various services, which sometimes require a large amount of data exchange. It also separates services into distinct units such as components or modules, which can be deployed over the Internet, and can be combined to be re-used for new applications. Through the SOA platform, services could be delivered to end-users. The typical layers for SOA include the following: business process layer, business service layer, application integration layer, and technology layer.

The general architectural principles point out the ground rules of SOA for its development, deployment, and maintenance. There are three ground rules, namely, usability, compliance to standards, and service modeling ([4], [5]). Moreover, nine specific principles, which are categorized into two types, are defined for components design. The first type includes the specific design guidelines of SOA for service providers. The second type pertains to the interaction between the service consumer and the provider.

From the SOA viewpoint, relationships between the service consumer and provider are not tightly stipulated. Their relations are loose coupling [6]. Thus, consumer services are not forcefully influenced by the changes made by the providers. Second, consumer service interacts with the service provider based on the service contract. Therefore, designing the Service Level Agreement (SLA) is an important task. Moreover, SLA should also satisfy some general and specific principles.

A consequence of the constraint is that SOA applications almost always have to be operated in a distributed environment ([7], [8]). Consequently, the end-users and service providers are geographically distributed. Services in SOA platform are delivered via the Internet. Unfortunately, SOA provides an environment that is convenient for hackers and intruders ([9], [10]). There are three reasons for these. The first

978-0-7695-3499-2/08 $25.00 © 2008 IEEE
DOI 10.1109/SOSE.2008.23

reason is precisely that SOA closely ties with the Internet. The second reason is that there are large amounts of remote procedure calls and exchange messages in SOA. The third reason emerges from the distributed providers' systems.

In point-to-point transmission, confidential data can be enforced on web services through the use of Transport Layer Security / Secure Sockets Layer (TLS/SSL). A challenge in using TLS would be if messages needed to go through a proxy server, as it would need to be able to see the request for routing. However, as the proxy is operating on the message, it does not ensure end to end security, but only ensures point-to-point security. Although, TLS/SSL can add tunnel to complete the network secure stack for such as Virtual Private Network (VPN). However, it hides client's network beyond the Internet and it is not convenient for daily transmission.

In end-to-end transmission, confidential data is encrypted by all the ways and transmitted from one client application to another client. Rather than relying on TLS/SSL, end-to-end security puts the power of strong encryption in the user's hands, through a simple interface. The confidential data was previously decrypted at the gateway. SOAP is a protocol for exchanging XML-based messages over the Internet ([11], [12]). The SOAP forms the foundation layer of the web services protocol stack providing a basic messaging framework on SOA platform. In fact, WS-Security incorporates security features in the header of a SOAP message, working in the application layer. However, tunneling over an inappropriate protocol such as SOAP is disingenuous. A firewall attempts to enforce security policy. If the policy states that SOAP is OK but unknown protocols are not, then layering a remote procedure call mechanism through this is not secure and against the security policy. The choices are to use a new, well-known port and to request it be opened for SOAP and other remote-invoked protocols, or the firewall is forced to toughen about packet inspection.

Security and privacy issues have emerged from the SOA healthcare platform. Large amounts of personal data are transmitted daily through the SOA healthcare platform. For security and privacy reasons, the exchanged data should be encapsulated and encoded by a specific standard, a customized standard even. We have to build a reliable and secure transmission interface for the SOA healthcare platform. In the next section, more requirements of the interfaces are described. In Section 4, the interface design is discussed by following a customized standard. The interface utilities with its health data format and health packet format are carefully discussed. The transmission mechanism of health packet is introduced in Section 5. The summary is given in Section 6.

3. Interface requirement on SOA platform

In order to deploy the SOA platform and its systems, we would first classify the relative stakeholders by their geographical locations. All roles of services scenarios on the SOA healthcare platform were divided into three groups by their locations into the following:

- Group A - Home: patient, family members
- Group B - Care center: center staff, care givers and part-time workers
- Group C - Service provider: intensive doctors, druggists, welfare workers, ambulance drivers

To satisfy the message exchange or the service composition between the above roles' systems through the SOA platform, we classified the whole system architecture into three parts: (1) home devices, (2) platform and applications, and (3) collaborative providers' systems. The system architecture overview is shown in Figure 1. The care center holds the platform and plays a kernel part in the system architecture. The center is responsible for communicating and coordinating with the front-end home devices and back-end services systems by various interfaces. We would then be able to layout what interfaces for various external systems are necessary through this overview. Distinct gateways or adapters for the interfaces should be drawn out in the detail design phase.

Home Gateway: To exchange messages between the home devices and the SOA platform, it is necessary to construct an interface called the home gateway for collecting and transferring the device status and the patient's bio-signals. The gateway should be a reliable software module installed in both the homebox and the platform. In the platform, the module is used to receive the patient's bio-signals and the status of the homebox. The status analysis is performed by another analysis module. In the homebox, the module is used to collect and send messages of home devices to the platform.

Adapters: The established platform should receive and analyze not only a patient's bio-signals but also satisfy the information needs of the center staffs, care givers, part-time workers, and intensive doctors, among others. Therefore, it is necessary to develop various adapters for communicating with providers' systems, which are the external systems in this architecture. Among its examples are PDAAdapter, HISAdapter, and EMSAdapter for external systems.

4. Interfaces design

4.1. Interface utilities

The gateway is an interface used to transmit messages among systems. Bio-signals, patient's conditions or homebox statuses could be handled as messages through the gateway. This gateway is combined by two separate parts – *sender* and *receiver*. Both the *sender* and *receiver* have their own functionalities. *Sender* is a software module used to encapsulate data into XML file, encode the file into packet, and send out the messages. *Receiver* is a software module used to receive messages, decode the packet to the file, and un-encapsulate XML file to data. The various distinctions are listed in Table 1. *Default folder* is a buffer used to prepare sending or receiving messages.

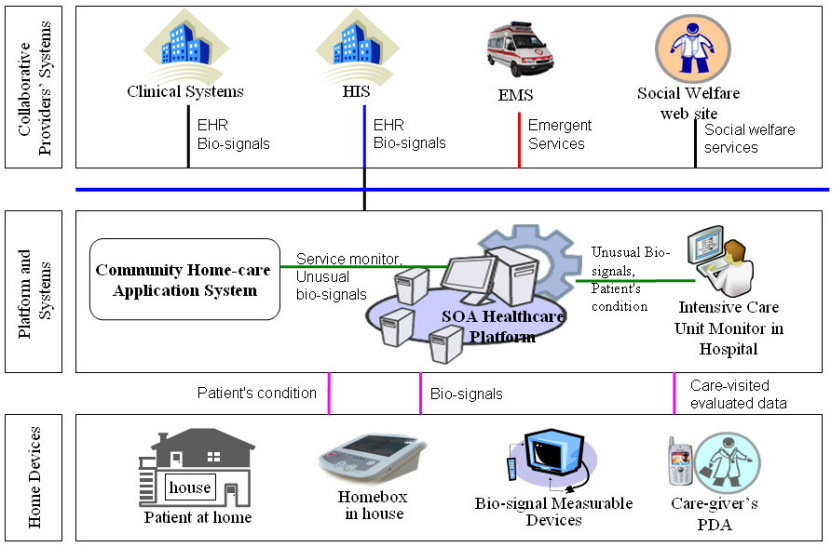

Fig. 1 System Architecture Overview

Table 1 Interface Utilities

	Sender	Receiver
Default Folder	Outgoing	Incoming
Health Data Format	XML File Encapsulation	XML File Un-encapsulation
Health Packet Format	Data Encode	Data Decode
Transmission	Send Packet	Receive Packet
Utilities	Homebox→Platform Platform→Homebox	Platform→Homebox Homebox→Platform Measurable Devices →Homebox

One utility of the *sender* is to send messages from homebox to the platform. If a patient's digital bio-signals are measured, the *sender* will first transform and encapsulate the bio-signals into XML file, which is a temporary file stored in the *outgoing* folder. It will be labeled and encoded into a packet by a *mapping table* before the file is sent out. The *mapping table* is a customized security. The manipulative sequence of the *sender* is shown in Figure 2.

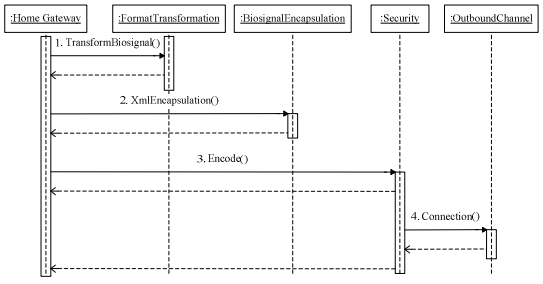

Fig. 2 Manipulative Sequence of the Bio-signals

4.2. Health data format

The file name of XML file is specified in Figure 3. The first field of the file name is used to indicate its priority. If the *priority* field is marked as "A", the message will be sent earlier then which is marked as "B". For example, the priority field of an emergency message file is always marked as "A".

The *label* field, which is reserved from second byte to fifth byte, is used to present message types. Fifteen message types are defined for various messages passing between the homebox and the platform. Seven message types are allowed to be sent from the homebox to the platform. One message type is defined as bidirectional. All message types and descriptions are listed in Table 2.

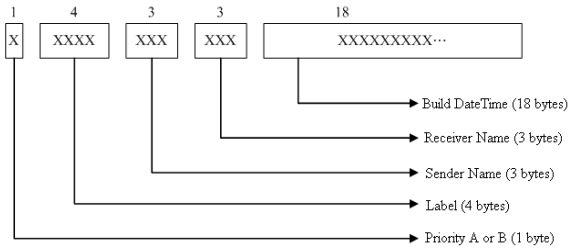

Fig. 3 File Name Format

Table 2 Message Types

Label	Description
Part I (homebox → platform)	
BA01	Send bio-signals
CA01	Send an emergency message
DA01	Respond "Yes" or "No"
DA02	Respond to the selected number
FA01	Respond with its IP address regularly
GA03	Respond with a file, which is requested by platform
IA02	Respond with its status
Part II (platform→homebox)	
DA03	Send an alternative question

DA04	Send a multiselection question
EA01	Notify a message to user via homebox
GA01	Send attached files, such as updated file
GA02	Request a particular file from homebox
HA01	Request a command to execute shell script, such as reboot homebox
IA01	Request to get the status of the homebox
Part III (homebox ←→platform)	
AA01	ACK of successful receipt of the message

The *sender* field is reserved to mark an alias of the sender. The *receiver* field is reserved to mark an alias of the receiver. The *datetime* field is used to indicate when the message is built. The following example means that one emergency message was built by homebox B01 and prepared for sending to platform S01 on September 28, 2006.

File Name Example:
ACA01B01S01200609281120561232

Table 3 Messages Encapsulated into XML File

Label	Example of encapsulated message content of the XML file
AA01	<Ack>200512528</Ack>
BA01	<Biosignal> <ID>A123456789</ID> <DeviceType>BloodPressure</DeviceType> <SignalType>HK97G</SignalType> <SignalData> <Instance No=”1”> <SignalTime>2005-03-04 17:29:26.105</SignalTime> <Level>Abnormal</Level> <H>101.0</H> <L>80.0</L> </Instance> <Instance No=”2”> <SignalTime>2005-03-04 19:21:54.232</SignalTime> <Level>Normal</Level> <H>119.0</H> <L>76.0</L> </Instance> </SignalData> </Biosignal>
CA01	<Text>Emergency</Text>
DA01	<Text>Yes</Text>
DA02	<Text>2</Text>
DA03	<Text>When you go to the hospital tomorrow, do you need someone to accompany with you?</Text> <WaveFileName>sample.wav</WaveFileName> <WaveReplay>5</WaveReplay> <WaveInterval>3</WaveInterval> <TextDisplayTime> 60</TextDisplayTime> <FontSize>5</FontSize>
DA04	<Text>Which number of the lunch would you like to eat today?</Text> <WaveFileName>sample.wav</WaveFileName> <WaveReplay>5</WaveReplay> <WaveInterval>3</WaveInterval> <TextDisplayTime> 60</TextDisplayTime> <FontSize>5</FontSize>
EA01	<Text>It’s time to eat drugs. Please eat drugs on time.</Text> <WaveFileName>sample2.wav</WaveFileName> <WaveReplay>8</WaveReplay> <WaveInterval>5</WaveInterval> <TextDisplayTime>120</TextDisplayTime> <FontSize>4</FontSize>
FA01	<IP>192.168.1.150<IP>
GA01	<Path>./homebox/</Path> <Base64Encoding> A File is encoded by base64</Base64Encoding>
GA02	<Path>./homebox/sample.exe</Path>
GA03	<Path>.\homebox\thejpg1.jpg</Path> <Base64Encoding>A File is encoded by base64</Base64Encoding>
HA01	<Path>./homebox/sample.sh</Path>
IA01	<Msg>HomeboxStatus</Msg>
IA02	<User>Ken is ok</User> <User>Jeff is ok</User>

All exchanged data and passing messages are encapsulated into an XML file by various message labels. The contents of the messages are transformed by specific tags and stored into the XML file. The examples of the encapsulated message contents for the defined message types are listed in Table 3.

4.3. Health packet format

For the security and privacy of the patient's information, the XML file should be encoded into a customized health packet if the file in the outgoing folder is ready to be sent out. A customized encoder will encode the XML file into Health Packet Format (HPF), which is shown in Figure 4. The header size of HPF is 64 bytes. There are 11 defined fields in the header. The specifications of the fields are listed in Table 4. Some examples are shown along fields in Table 4. Owing to the customized encoder, the exchanged data will not be easily recognized by the other tools.

The *CertifyID* field is used to validate the legality of the packet. If a packet passes the validation, the *receiver* will manipulate the packet. If a packet fails, the *receiver* will immediately skip the packet. The *Level* field is used to present the priority of a packet. The marked "E" in the *Level* field has higher priority than the one marked "N". The *receiver* always manipulates the packet with higher priority. The *Encode* field is used to indicate if the data should be encoded by a *mapping table*.

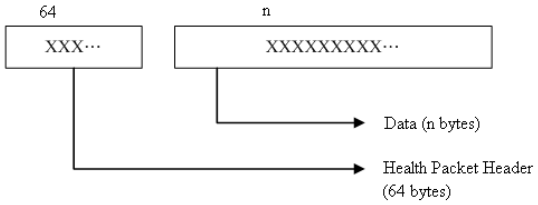

Fig. 4 Health Packet Format

Table 4 Health Packet Header

Field Name	Bytes	Example
CertifyID	7	XLXIDDU
SenderID	3	S01
SenderIP	4	140.114.87.15
ReceiverID	3	B01
ReceiverIP	4	192.168.1.150
SequenceID	4	200512527
ACKID	4	200512528
Level	1	E: Emergent N: Normal
Encode	1	N: None M: Mapping table
FileName	29	File format
PacketSize	4	Packet Size

5. Health packet transmission

The health packet is encoded by obeying its format and sent out by the TCP socket. In the first step, a connection between the *sender* and *receiver* should be built. The connection is prepared to provide secure and private transmission channel. When the *sender* is going to send its packets, the *receiver* should be ready to receive packets in a listen loop. The transmission is executed by observing TCP protocol and using the TCP socket. If a *receiver* gets a final sequence of the transmitting message, it will send a stop ACK to the *sender*. The *sender* thus can close the connection. The sequence of the transmission is shown in Figure 5. The pseudo codes of *sender* and *receiver* are shown in Figure 6 and Figure 7, respectively.

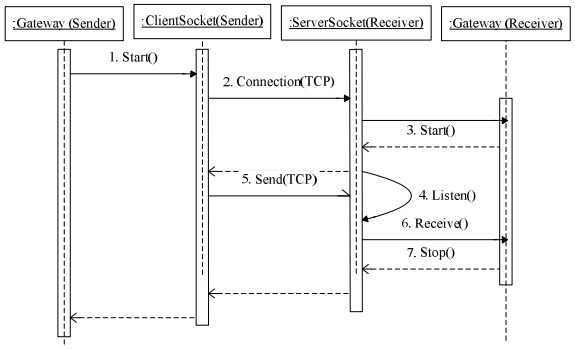

Fig. 5 Health Packet Transmission

6. Conclusions

Infusing Service-Oriented Technology to provide common activities and interests has become one of the popular methods in different industries. Through the SOA platform, we could integrate individual providers into similar service processes. As revealed in this study, data exchange and message passing are both key issues in SOA-based systems. For reasons of security and privacy, the exchanged data should be encapsulated and encoded by a specific standard, a customized standard even. We have to build a reliable and secure transmission interface for the SOA healthcare platform. In this study, we first focused on investigating the importance of data exchange and message passing on SOA. Thereafter, we designed a gateway for message passing on the SOA healthcare platform. We initially pointed out the interface utilities on the platform. Health data format and health packet format, which are customized standards, were then carefully defined. Moreover, the transmission mechanism between the *sender* and the *receiver* were exposed.

The SOA healthcare platform and its applications were deployed in a real environment based on the architecture overview. In Kaohsiung city, the systems were installed in a community care center in October 2007. As the practical results in beta-test, the messages could be passed in a secure and reliable routing. These customized messages could successfully avoid being recognized and transmit in a secure way, when they were intercepted by the other P2P streaming tools, such as Ethereal (wireshark), Winpcap, and PPStream, among others.

```
Sender() {
   LOOP () {
      CALL CheckOutgoing()
      IF (XML file existed in outgoing) {
         CALL CreateClientSocket
         IF (Connected) {
            try {
               CALL Encoder
               CALL Open XML file
               CALL Read XML file
               CALL Write into HPF
               CALL SendHPF
            } catch {} finally {
               CALL DeletedClientSocket
            }
         }
      }
   }
}
```

Fig. 6 Sender Method

```
Receiver() {
   CALL InitialSocket
   CALL Listen
   LOOP () {
      IF (CertifyID is TRUE) {
         ReceiveHPF in Buffer
       IF (End of HPF) {
          Try {
             CALL Decoder
             CALL OpenXML file
             CALL ReadPHF
             CALL WriteXML file
             CALL PutXML file in Incoming
             CALL Leave LOOP
          } catch {} finally { … }
       }
      }
  }
   CALL DeletedClientSocket
}
```

Fig. 7 Receiver Method

7. Acknowledgements

This research was supported by the Applied Information Services Development and Integration project of the Institute for Information Industry (III) and sponsored by MOEA, Taiwan R.O.C.

8. References

[1] Thomas Erl, Service-oriented Architecture: Concepts, Technology, and Design, Prentice Hall PTR., July, 2005.

[2] Dave Hornford, Definition of SOA, The Open Group, October, 2006.

[3] Choudhary, V., “Software as a Service: implications for investment in software development”, Proceedings of the 40th Annual Hawaii International Conference on System Science (HICSS’07), IEEE Computer Society, Los Alamitos, CA, USA, 2007, pp. 209-218.

[4] Yvonne Balzer, Improve your SOA project plans, IBM Global Services, July 2004.

[5] Chi-Lu Yang, Yeim-Kuan Chang and Chih-Ping Chu, ”Modeling Services to Construct Service-Oriented Healthcare Architecture for Digital Home-Care Business”, Proceedings of the 20th International Conference on Software Engineering and Knowledge Engineering (SEKE’08), San Francisco, CA, USA, July, 2008.

[6] Eric Newcomer and Greg Lomow, Understanding SOA with Web Services, Addison Wesley, January, 2005.

[7] Asit Dan, Robert Johnson and Ali Arsanjani, “Information as a service: modeling and realization”, Proceedings of International Workshop on Systems Development in SOA environments (SDSOA’07), IEEE Computer Society, Los Alamitos, CA, USA, May, 2007.

[8] Gennaro Cuomo, “IBM SOA on the Edge”, Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, ACM Press, New York, NY, USA, 2005, pp 840-843.

[9] Ned Chapin, “Service Granularity Effects in SOA”, Proceedings of the 20th International Conference on Software Engineering and Knowledge Engineering (SEKE’08), San Francisco, USA, July, 2008.

[10] H. Xu, M. Ayachit and A. Reddyreddy, ”Formal Modeling and Analysis of XML Firewall for Service Oriented Systems“, International Journal of Security and Networks (IJSN), Vol. 3, No. 3, 2008.

[11] Anthony Nadalin, Chris Kaler, Phillip Hallam-Bakerand and Ronald Monzillo, Web Services Security: SOAP Message Security 1.0, Organization for the Advancement of Structured Information Standards (OASIS), March, 2004.

[12] Tim Bray; Paoli Jean, C. M. Sperberg-McQueen, Eve Maler and François Yergeau, Extensible Markup Language (XML) 1.0 (Fourth Edition), World Wide Web Consortium (W3C), Sep., 2006.