A Service-Oriented Multimedia Componentization Model[1][2]

Jia Zhang
Department of Computer Science
Northern Illinois University
DeKalb, IL 60115
jiazhang@cs.niu.edu

Liang-Jie Zhang
IBM T.J. Watson Research

Yorktown Heights, NY 10598
zhanglj@us.ibm.com

Francis Quek
Center for Human-Computer Interaction
Virginia Tech
Blacksburg, VA 24061
quek@cs.vt.edu

Jen-Yao Chung
IBM T.J. Watson Research

Yorktown Heights, NY 10598
jychung@us.ibm.com

ABSTRACT:

As Web services become more and more popular, how to manage multimedia Web services that can be composed as value-added service solutions remains challenging. This paper presents a service-oriented multimedia componentization model to support Quality of Service (QoS)-centered, device-independent multimedia Web services, which seamlessly incorporates cutting-edge technologies relating to Web services. A multimedia Web service is divided into control flow and data flow, each can be delivered via different infrastructures and channels. Enhancements are proposed to facilitate Simple Object Access Protocol (SOAP) and Composite Capability/Preference Profiles (CC/PP) protocols to improve their flexibility to serve multimedia Web services. We present a set of experiments that show the viability of our service-oriented componentization model that can support efficient delivery and management of multimedia Web services .

KEY WORDS:

Multimedia Web services, SOAP, CC/PP, Service-oriented componentization model

INTRODUCTION

Simply put, a Web service is a programmable Web application that is universally accessible through standard Internet protocols (Ferris, 2003). The rapidly emerging technology of Web services exhibits the capability of facilitating Business-to-Business (B2B) collaboration in an unprecedented way. By means of each organization exposing its software services on the Internet and making them universally accessible via standard programmatic interfaces, this Web services paradigm enables and facilitates the sharing of heterogeneous data and software resources among collaborating organizations (Benatallah, 2002). In addition, Web services technology provides a

[1] This article is a significant extension to the authors' two papers: (1) "A Router Model for QoS-based Multimedia Web Services" that appeared in the Proceedings of IEEE International Conference on Multimedia & Expo (ICME 2003), Jul.6-9, 2003, Baltimore, MD, USA, pp. 797-800; and (2) "A SOAP-oriented Component-based Framework Supporting Device-independent Multimedia Web Services" that appeared in the Proceedings of IEEE 4th International Symposium on Multimedia Software Engineering (MSE 2002), Dec.11-13, 2002, Newport Beach, CA, USA, pp. 40-47.

[2] The first author is also a Guest Researcher of National Institute of Standard and Technology (NIST).

uniform framework to increase cross-language and cross-platform interoperability for distributed computing and resource sharing over the Internet. Furthermore, this paradigm of Web services opens a new cost-effective way of engineering software to quickly aggregate individually published Web services as components into new services. Therefore, the Web services technology has attained significant momentum in both academia and industry.

If the sharable data to be published by a Web service contain multimedia content, which refers to information that seamlessly integrates multiple media types in a synchronized and interactive presentation, the Web service is considered as a multimedia Web service. Multimedia Web services pose new challenges due to the unique characteristics of multimedia data (Khan, 2002). First, the transport of the multimedia information has to meet some Quality of Service (QoS) requirements, such as the synchronization within and among different multimedia data streams or real-time delivery. For example, let us consider a typical Video on Demand (VoD) service, an Internet Kara OK service. It is critical to provide a significant short-response-time service to a VIP customer. In addition, the audio and video information needs to be synchronized on customer's system. Second, the Simple Object Access Protocol (SOAP) (SOAP), the core transport technique of Web services, does not support massive message transport that is imperative for multimedia content transport, nor multimedia QoS requirements (Khan, 2002). Third, with the advancement of wireless information appliances, Web service interfaces provide a means to enable the content or service to be created once and accessed by multiple SOAP-enabled [4-6] devices, such as wireless phones (NORTEL), Personal Digital Assistance (PDAs), set-top boxes, as well as regular Web browsers. A Web service is thus considered to be device independent if it can be delivered to different devices (Han, 2000). How to deliver a multimedia Web service to users based upon their possessed devices remains challenging.

In summary, the interoperability of Multimedia Web services is not without penalty since the value added by this new Web service paradigm can be largely defeated if a multimedia Web service: (1) cannot guarantee QoS attributes; (2) cannot be transported via the Internet in an organized manner; and (3) cannot be effectively adapted to end devices including mobile devices. In this paper, we aim to present a solution to these existing issues. We accomplish this goal in several ways. First, we propose a separation of control flow and data flow for multimedia Web services, with SOAP is used to transport the control flow. Second, we propose enhancements to SOAP to facilitate its flexibility to serve the transportation of multimedia Web services. Third, we propose enhancements to Composite Capability/Preference Profiles (CC/PP) protocol (CCPP) to provide an easy and flexible way to split and adapt multimedia Web services to appropriate composite devices, and increase the flexibility for users to manage multi-devices. Finally, we propose a service-oriented multimedia componentization model to support device-independent multimedia Web services.

The remainder of this paper is organized as follows. In Section 2, we briefly introduce some core techniques of multimedia Web services. In Section 3, we discuss related work. In Section 4, we present our solution. In Section 5, we present performance analysis. In Section 6, we summarize the contributions and innovations, assess limitations and discuss future work directions.

CORE TECHNIQUES OF MULTIMEDIA WEB SERVICES

In this section, we will first briefly introduce the core techniques and standards of multimedia Web services to provide readers with some background context.

Web services typically adopt a provider/broker/requester architectural model (Roy, 2001). A service provider registers Web services at service brokers; a service broker publishes registered services; and a service requester searches Web services from service brokers. The essential aspect of this model is the concept of dynamic invocation: Web services are hosted by service providers; and service requesters dynamically invoke the Web services over the Internet on an on-demand basis.

In order to enable the communications among service providers, service brokers, and service requesters, the paradigm of Web services mainly embraces three core categories of supporting facilities: communication protocols, service descriptions, and service discovery (Roy, 2001). Each category possesses its own *ad hoc* standard. The Simple Object Access Protocol (SOAP) (SOAP) acts as a simple and lightweight protocol for exchanging structured and typed information among Web services. The Web Service Description Language (WSDL) (WSDL) is an XML-based description language that is used to describe the programmatic interfaces of Web services . The Universal Description, Discovery, and Integration (UDDI) standard (UDDI) provides a mechanism to publish, register, and locate Web services. It should be noted that here we adopt a narrow definition of Web services that refers to an implicit definition of SOAP+WSDL+UDDI for the purpose of simplicity. This implies a focus on the management of stand-alone Web services, instead of the compositions of and the interactions between multiple Web services.

As more and more business organizations have been adopting Web services technology to publish their sharable data to make their services accessible to more other organizations, it is possible that the data to be published include multimedia information, e.g. audio and video. Due to the fact that multimedia data exhibits unique features that require specific handling (Khan, 2002), it is necessary to examine the existing Web services techniques to better support multimedia Web services. When the SOAP+WSDL+UDDI framework is applied to support a multimedia Web service, two major factors influence the success of a multimedia Web service: (1) the transport of multimedia information over the Web, and (2) the management of composite devices for multimedia contents. Multimedia content caching and streaming are two essential solutions to the first issue (Paknikar, 2000). Web caching at a service provider site can largely increase the service provider's ability to support a big amount of service requesters. The streaming paradigm enables a media file be played at the service requester's site while it is being transferred over the Web; therefore, the burden of the service provider can be alleviated. Regarding the latter issue, at the service requester side, multimedia data is normally split over multiple devices with appropriate multimedia capabilities (Han, 2000).

Due to the rapid advancement of wireless information appliances, a Web service will gain more popularity if it is accessible from mobile devices in addition to normal Web browsers, such as wireless phones and Personal Digital Assistance (PDAs) (Pham, 2000). A Web service is thus considered to be device independent if it can be delivered to different devices (Han, 2000). Several techniques are designed to support the device independence. Based on an Extensible Markup Language (XML) and Resource Description Framework (RDF)-based framework, Composite Capability/Preference Profiles (CC/PP) are proposed to define device capabilities and user preferences so as to manage composite devices (CCPP). The combination of XML and Extensible Stylesheet Language (XSL) is usually utilized to realize device independence (Kirda, 2001).

RELATED WORK

With the basic background, we can now take an examination of related work on multimedia Web services.

Paknikar and colleagues (Paknikar, 2000) define a client-side framework for the caching and streaming of Internet multimedia. The architecture consists of a number of caching proxy servers. A central controlling proxy server called a broker handles all of the initial interactions, and then transfers controls to its sibling proxy servers. This hierarchical structure provides scalability for the proxy servers. Their work also defines a layered replacement policy for caching scalably encoded video objects. Paknikar et al. predicted that the Real Time Streaming Protocol (RTSP) would become the *de facto* standard for Internet Audio/Video (A/V) caching and streaming.

Pham and colleagues (Pham, 2000) define a *Small Screen / Composite Device (SS/CD)* architecture that supports small screen device-focused communication systems. The key component of the architecture is a Smart Gateway (SG) that distributes multimedia information to the most appropriate composite devices to ensure reliable performances. The critical component of SG is a set of algorithms associated with a Selection-Device-Assignment-Matrix. FieldWise (Fagrell, 2000) relies on a server engine to adapt multimedia responses according to the capabilities of the client devices and their network connections. However, these two projects do not adopt the most current Web techniques and standards, such as XML/XSL and CC/PP.

WebSplitter (Han, 2000) provides a unified XML framework supporting multi-device Web browsing. The framework defines a XML-based metadata policy file based on the CC/PP protocol to enable users to specify their access privilege groups. With WebSplitter, all Web pages are constructed as XML files, with pre-defined tags describing mappings to the corresponding access privileges. A proxy is then adopted to split a Web page to different devices. Corresponding XSL style sheets are attached to devices to transform the customized XML to the suitable device-understandable languages. MyXML (Kirda, 2001) is an XML/XSL-based template engine to solve the issue of device independence, whose idea is to completely separate the content from its layout information. Similar to the WebSplitter, MyXML utilizes the XML/XSL combination to realize device independence. However, MyXML introduces a whole set of syntax elements that requires learning curve.

All these efforts concentrate on the client site to facilitate multimedia storage and streaming, to assist multimedia distribution, and to support device independence. At this moment, however, it appears that there still lacks a generic infrastructure for considering both client and server sides of multimedia Web services. An additional limitation is that these methods may or may not integrate easily with the most current Web technologies and standards, since Web service is still an emerging paradigm. Here we seek to provide efficient support of delivery and management of multimedia Web services. In contrast with the previous approaches, we accomplish this objective by seamlessly incorporating the cutting-edge techniques of Web services and providing a SOAP-oriented component-based framework to support device-independent multimedia Web services.

SERVICE-ORIENTED COMPONTIZATION MODEL

In this section, we will introduce our solution to support multimedia Web services. We will first discuss our idea of separation of control flow from data flow to utilize SOAP. Second, we will

propose our enhancements to SOAP. Third, we will propose enhancements to CC/PP protocol. Finally, we will propose our service-oriented multimedia componentization model.

Separation of Control Flow and Data Flow

As a core technique, SOAP is used to transport the content of Web services. However, SOAP was not originally designed to support multimedia Web services; therefore, its current version is not appropriate for streaming multimedia content. The reasons are multi-folded: (1) It is usually infeasible to put a large piece of multimedia content (e.g., a video clip) into one message. However, SOAP does not support message boxcarring and batching (SOAP); therefore, its current version cannot be used to transfer streamed multimedia content. (2) Using SOAP to transport data requires enormous network bandwidth due to its eXtensible Markup Language (XML) markup and protocol overhead (Werner, 2004). Therefore, its performance drawback hinders SOAP from transporting multimedia content that is usually associated with QoS requirements such as response time. (3) Current SOAP specification does not provide facility to define multimedia QoS requirements (Khan, 2002) and multimedia management information such as synchronization signals necessary for time dependent media. In summary, current SOAP is not suitable to transport massive amount of multimedia content.

Nevertheless, we do not have to use SOAP to transport multimedia content. There are already a wealth of existing infrastructures, channels, and standards to support transport of different multimedia content (e.g., MPEG-21 (MPEG-21), SMIL (SMIL), JPEG (JPEG), and HotMedia (HotMedia)). We can still utilize these existing techniques to transport the content of media files. But how about the multimedia control information?

Our solution is to separate the control information from the content information. The control information may include (1) meta data that depict the synchronization relationship between multiple media files, (2) QoS requirements, and (3) other control information such as the service provider. Thus transporting a multimedia Web service includes the delivery of both data flow and control flow. The essential advantage is that different transport protocols can be employed to deliver either data flow or control flow, i.e., SOAP can be used to transport the control information while traditional multimedia channels can be used to transport the multimedia content in SOAP-specific environment.

This separation of control flow and data flow promises several merits. First, the separation solves the dilemma of transferring multimedia applications into multimedia Web services. Control information will be delivered by SOAP so that different multimedia Web services can interoperate with each other, while multimedia content information can be delivered by traditional multimedia protocols to achieve acceptable performance. Second, the separation provides a loose coupling between control information and content information; thus, the content information can be reused for different Web services. Third, the separation facilitates the tracking and logging of control information so that we can achiever better management and monitoring of multimedia Web services.

Enhancements of SOAP to Support Multimedia Web Services

SOAP (SOAP) has been considered to be a *de facto* communication standard to deliver Web services; however, it was not originally designed particularly to deliver multimedia Web services. First, the nature of multimedia data (Khan, 2002) requires the fact that the transportation of the multimedia information normally has to meet some QoS requirements, such as the

synchronization within and among different multimedia data streams or real-time delivery. Consequently, a message containing multimedia data should carry over its QoS requirements as the guidance for Internet transportation. Nevertheless, the original SOAP specifications do not support this feature. Therefore, we believe that enhancements to SOAP are compulsory in order to facilitate multimedia Web services.

Second, for simplicity, SOAP does not support boxcarring and batching of messages; also it is a one-way protocol (SOAP). There are cases where it is difficult to embed a large multimedia file into one SOAP message. On the contrary, it may be more practical to load a big chunk of information into multiple SOAP messages. Of course there should be a way to identify the relationships among these related SOAP messages. In exceptional cases, some of the media files may not even be suitable to be transferred in SOAP messages, e.g. a multimedia segment may need to be transferred in one file with specific file extension. Therefore, special attributes should be provided to identify all of these situations. It should be noted that in the last section, we discussed that multimedia information can be transported via different channels other than SOAP. However, for completeness, we still believe that SOAP should be extended with mechanisms to transport multimedia information.

Our enhancements to SOAP can be therefore categorized into two sets: message batching specifications and multimedia QoS specifications. Two sets of attributes are introduced as summarized in Figure 1; and each of them will be discussed in detail in this section. We first propose a simple workaround of boxcarring and batching of messages to bolster large-scaled multimedia data transportation. A service provider can divide a big message into multiple smaller messages; and these smaller messages altogether conceptually constitute a message box. The service provider can then send these messages to the service requester asynchronously.

As illustrated in the first part of Figure 1, several attributes are defined to support this ability: *SenderURL*, id, index, and total. *SenderURL* is defined to specify the unique address of the service provider, so that the service requester knows where to fetch further information if so

Attribute	*Definition*
SenderURL	Unique address of the service provider
id	Uniquely identify the message
index	The index in message box
total	The total number of messages in message box
MustSendBack	Required to be sent back without changes
Metadata	Specify the structure of the message box
component	Specify the messages in the message box

Attribute	*Definition*
reliable	Reliable transportation
realTime	Real-time transportation
unicast	Unicast to specific node
multicast	Multicast to multiple nodes
secure	Secure in the transportation

Figure 1. SOAP Attributes Introduced

desired; id is defined to uniquely identify the message box in the domain of the service provider; index is defined to identify the index of the message in the corresponding message box; and total is defined to identify the total number of messages in the message box. These attributes can be utilized to uniquely identify a SOAP message and the relationships between messages. In addition, a global attribute *MustSendBack* is introduced, which can be utilized to specify that the information is required to be sent back to the service provider without any changes. For example, the service requester's profile needs to be sent back along with the result for the proper multimedia distribution.

Employing the metaphor of the envelope in postal delivery, we insert these identification information of the multimedia content into the corresponding SOAP envelope as the first part. Meanwhile, we propose to always have the first message contain the metadata in its body block, which identifies the structure of the message box, in other words, the order of the messages in the box. *Attributes Metadata* and component are defined to specify the metadata information. Let us take a piece of a SOAP message as an example.

```
<SOAP-ENV:Envelope
….
      SOAP-ENV:MustSendBack=”RequestId0001, Profile001”
      SOAP-ENV:id=”Msg00001”
      SOAP-ENV:index=”1”
      SOAP-ENV:total=”5”/>
      <SOAP-ENV:Body>
         <m:Metadata>
              <component>Msg00001”</component>
              <component>Msg00002”</component>
              <component>Msg00003”</component>
              <component>Msg00004”</component>
              <component>Msg00005”</component>
         </m:Metadata>
       </SOAP-ENV:Body>
….
</SOAP-ENV:Envelope>
```

This SOAP message is the response message to the request “RequestId0001”. This request id “RequestId0001” and user profile information “Profile001” must be sent back. We can see that the message is identified by its unique id “Msg00001”; and it is the first of the total five messages in the message box. Metadata records the message ids of all five messages: “Msg0001”, “Msg0002”, “Msg0003”, “Msg0004”, and “Msg0005”. Therefore, the requester that receives this message would be aware of the remaining four messages, then could schedule to pre-fetch them with specified ids if so desired.

Second, we introduce five global attributes to the SOAP definition in order to enable a SOAP message to carry on its QoS requirements: *reliable*, *realTime*, *unicast*, *multicast*, and *secure*. These five attributes are summarized in the second part of Figure 1. For an attribute to be set, its keyword-value pair must be present in this envelop block. The *reliable* attribute can be used to indicate whether the SOAP message requires reliable transportation or not. The value of the *reliable* attribute is either “1” or “0”. The absence of the *reliable* attribute is semantically equivalent to its presence with a value “0”. If a header element is tagged with a *reliable* attribute

with a value of "1", the recipient of that header entry either MUST find a reliable transportation protocol, or MUST fail processing the message.
The *realTime* attribute can be used to indicate whether the SOAP message requires real-time transportation or not. The value of the *realTime* attribute is either "1" or "0". The absence of the *realTime* attribute is semantically equivalent to its presence with a value "0". If a header element is tagged with a *realTime* attribute with a value of "1", the recipient of that header entry either MUST find enough resources to transfer the message right away, or MUST fail processing the message.

The *unicast* attribute can be used to indicate whether the SOAP message is to be unicasted to a specific end point. The value of the *unicast* attribute is either "1" or "0". The absence of the *unicast* attribute is semantically equivalent to its presence with a value "0". If a header element is tagged with a *unicast* attribute with a value of "1", the recipient of that header entry either MUST find a available path to the specified end point node, or MUST fail processing the message.

The *multicast* attribute can be used to indicate whether the SOAP message is to be multicast to multiple users. The value of the multicast attribute is either "1" or "0". The absence of the *multicast* attribute is semantically equivalent to its presence with a value "0". If a header element is tagged with a *multicast* attribute with a value of "1", the recipient of that header entry either MUST find available paths to all specified end-point users, or MUST fail processing the message.
The *secure* attribute can be used to indicate whether the SOAP message has to be kept secure in the process of transportation. The value of the *secure* attribute is either "1" or "0". The absence of the *secure* attribute is semantically equivalent to its presence with a value "0". If a header element is tagged with a secure attribute with a value of "1", the recipient of that header entry either MUST find secure paths to the destination, or MUST fail processing the message.

Following is another example of a piece of a SOAP message containing some multimedia information. This message needs to be transferred reliably, real-time, and only transferred to one specific end user. This enhancement provides a way for a SOAP message to delineate its QoS requirements, which can be utilized as a guidance of Internet transportation tools to select different paths and transportation protocols so as to increase performance.

```
<SOAP-ENV:Envelope  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
      SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      SOAP-ENV:reliable="1"
      SOAP-ENV:realTime="1"
      SOAP-ENV:unicast="1"
…
      <SOAP-ENV:Body>
…
      </SOAP-ENV:Body>
…
</SOAP-ENV:Envelope>
```

Enhancement of Cc/Pp Supporting Multimedia Web Services

Multimedia Web services normally need to split multimedia data over multiple devices with appropriate multimedia capabilities (Han, 2000). CC/PP specifies an XML and RDF based framework to help define device capabilities and user preferences so as to manage composite devices (CCPP). We adopt CC/PP to support multimedia Web services on two reasons. One is

that CC/PP was designed particularly for describing device capabilities and user preferences. The other one is that CC/PP is built on top of popular XML technology. Utilizing the CC/PP format, every user can create his/her own profile that declares the list of the user's available resources (devices) and preferences about how each resource would be utilized. However, we found that CC/PP definition is too rigid for device capabilities. For instance, one device may be capable of accepting one kind of media information with some simple transformations; and CC/PP does not support this specification. Therefore we extend CC/PP by enabling transformation description to be added to devices.

With our extension, in a user's profile, every resource is declared as a separate item, such as a WAP phone. Each item comprises of two parts: one is the declaration of the hardware; and the other one is the declaration of the service, or more directly multimedia resource files that the device can receive. Following is an example of a piece of an extended-CC/PP profile for a WAP phone.

```
<ccpp:component>
    <rdf:Description rdf:about="TerminalHardware">
        <rdf:type rdf:resource="HardwarePlatform"/>
        <DeviceName>Nokia-3360</DeviceName>
        <screen>30X23mm</screen>
        <display>101X52Pixels</display>
        <PixelStretch>1.24</PixelStretch>
    </rdf:Description>
</ccpp:component>
…
<ccpp:component>
    <rdf:Description rdf:about="Services">
        <rdf:type rdf:resource="SupportedServices"/>
        <rdf:li>HTML</rdf:li>
        <rdf:transform>HTML2WML.xsl</rdf:transform>
    </rdf:Description>
</ccpp:component>
```

As shown, the declaration of this WAP phone contains two parts. The first part defines the hardware, such as the device name, the size of the screen, resolution, etc. The second part defines its acceptable multimedia information, such as image, text, WML, etc. However, it can be noticed that the normal HTML code cannot be directly retrieved on this WAP phone. A stylesheet named HTML2WML.xsl needs to be used to transform the receiving HTML code to the WAP-enabled WML code. With this extension, CC/PP protocol becomes more powerful of describing device capabilities.

Proposed Model Supporting Multimedia Web Services

As Roy and Ramanujan (Roy, 2001) summarized, a three-role model is broadly adopted in the field of Web services: service providers, service brokers, and service requesters. From the perspective of a service requester, there are two key questions: how to find a demanded service, and how to get the service effectively. This paper focuses on the solution to the second question; therefore service brokers and related issues will not be discussed. The starting point of this paper is that, with the help of service brokers, service requesters have already located the service providers that offer the requested services. Meanwhile, as high-speed local area networks (LANs)

have been deployed extensively over the world (Paknikar, 2000), users on such LANs normally access Internet through a proxy server that provides caching facility. Therefore, we make another assumption on such a concept, that users always invoke Web services through their proxy server. In addition, due to the fact that service requesters are normally separated from service providers by Internet, it is reasonable to assume that a message has to pass through multiple networks between them. Based on these three assumptions, the problem can be refined as: how to effectively and efficiently support multiple service requesters, who rely on the same proxy server and require the same Web service from the same service provider that is several networks away.

Here we propose a component-based framework as a solution. As illustrated in Figure 2, three types of intelligent agents are introduced as the main components of the framework; and SOAP is adopted as the service transportation protocol. The first agent introduced is the Multimedia Web Service Server Agent (MWSSAgent) that locates at service provider; the second one is the Multimedia Web Service Agent (MWSAgent) that locates at proxy server on the clients' LAN; and the third one is the Routing Agent (Ragent) that locates at intermediate network nodes. Figure 2 as well implicates the information path of a multimedia Web service on the basis of this framework. When a service requester submits the request, the MWSSAgent handles the request, invokes the corresponding service backend to get the results, equips return SOAP messages, and sends them to the Internet. The RAgent reads the envelopes of the receiving SOAP messages, analyzes their QoS requirements, and selects the appropriate protocols to send to the next proper RAgent or the proxy server. The MWSAgent finally receives the SOAP messages and propagates to the original requester, as well as distributes the multimedia streams to appropriate media devices, such as Web browsers, audio systems, PDAs, etc. We will discuss each of the three agents in detail in the following sections.

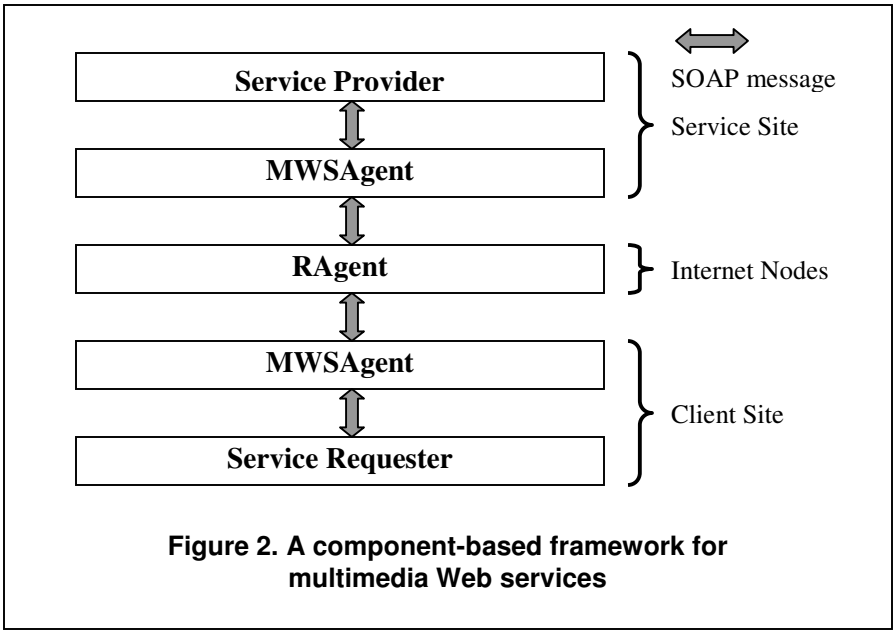

Figure 2. A component-based framework for multimedia Web services

Multimedia Web Services Server Agent (MWSSAgent)

A multimedia Web service may generate a complex result including multiple multimedia files. Disregarding the fact that SOAP currently can only bind to HTTP while HTTP was not designed for streaming media data (Paknikar, 2000), it is not efficient and practical to encapsulate all information in one SOAP response message to pass through the Internet, especially when the

information may get lost in the process of transportation. With our enhancements to SOAP described in the previous section, a big piece of information can be separated into multiple interrelated SOAP messages, with the first message containing the metadata and specifying the identification of each message. The metadata is completely separated from the real content. Here the metadata includes the structure of the set of the sub-messages, and the relationships between messages. It is one of the main responsibilities of the WMSSAgent to generate the SOAP response message box from the result provided by the service provider. To avoid aggravating Internet traffic, the response philosophy that the MWSSAgent adopts is a lazy-driven norm. That is, it does not always send back to the service requester all of the messages at the same time; some messages may stay on the MWSSAgent until they are requested particularly. Therefore, the caching facility is imperative to support this philosophy. In addition, published Web service providers normally expect large amount of requests, thus managing and reusing cached messages become essential.

Therefore we propose the MWSSAgent as an intelligent agent on the service provider site to facilitate multimedia services. Its architecture is illustrated in Figure 3, together with the interactions among its components. The architecture contains three functional components - SOAP message parser (MP), SOAP message generator (MG), SOAP message manager (MM) - and a local cache. MP is in charge of parsing incoming SOAP request messages, interpreting the requests and forwarding them to MM. MM first checks the cache to see whether the result SOAP messages have already been generated and stored. If the results exist, the MM will send back the results through the MG – the MG needs to generate the corresponding return envelopes based on the requests. If the results are not found, the MM will invoke service backend for the service. The MG will then generate the full set of SOAP response messages and store to the local cache, before sending back the first response message that includes the metadata. All of the SOAP response messages will be cached on local disk under the control of the MM. When a SOAP request message arrives, the MP will verify whether the request is the first request for a service, or a subsequent request for a specific response message of a particular service. When the MG generates the set of response messages, all messages will be uniquely numbered for identification purposes. Owing to the fact that a service requester will invoke a specific SOAP message later on, the full set of SOAP messages corresponding to a Web service will either be all stored in the cache of the MWSSAgent, or fully swapped out when storage limitation is encountered. Here we adopt the caching and streaming algorithms introduced in (Paknikar, 2000).

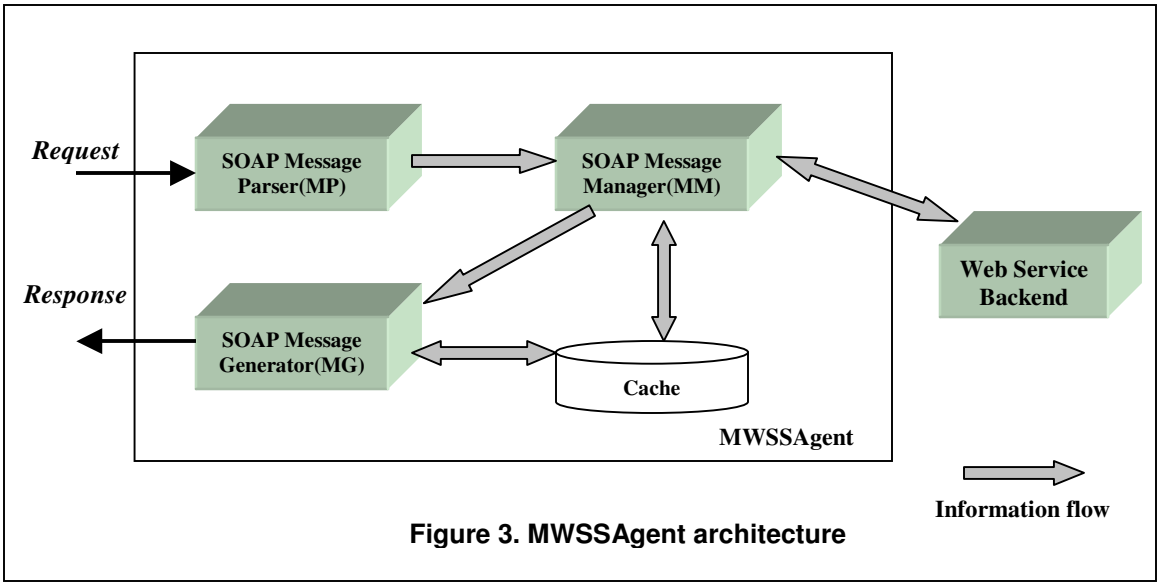

Figure 3. MWSSAgent architecture

Multimedia Web Services Agent (MWSAgent)

We propose a MWSAgent as an intelligent agent on the proxy server at the client site to support multimedia Web services. The component-based architecture is illustrated in Figure 4. The MWSAgent comprises of four functional components: service broker, user profile manager, service delivery manager, and multimedia manager. In addition, there are three caches contained in a MWSAgent: one for user profiles, one for request-profile mapping, and one for multimedia information.

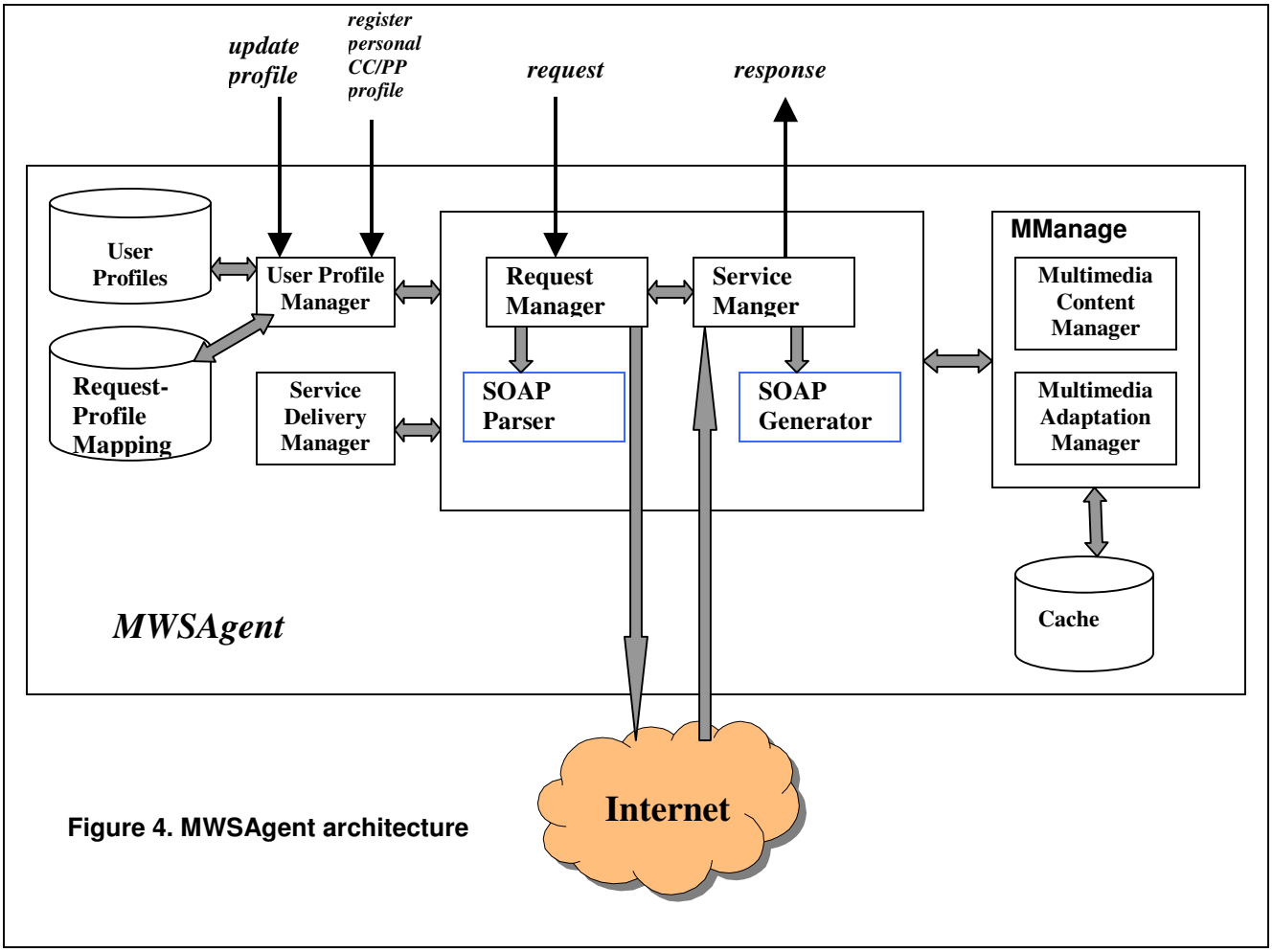

Figure 4. MWSAgent architecture

User Profile Manager (UPM)

User profile manager (UPM) registers user profiles and stores them at the user profiles cache. Adopting our extended CC/PP protocol as discussed in the previous section, a user profile declares the list of the user's available resources and the user's preferences about how each resource would be utilized. UPM assigns a unique id to every registered profile, so that one user can possess multiple profiles for the purposes of different Web services. When a user requests a Web service, he is required to specify which profile he wishes to use to receive the service. The service broker will assign a unique id to each request; and the UPM will store the 2-tuple (request id, profile id) in the cache of request-profile mapping. This 2-tuple will also be included in the

SOAP message of the service request as *MustSendBack*. Therefore, when the response comes in, the UPM will use the sent-back (request id, profile id) pair to pick up the stored corresponding profile from the profile cache. Users are allowed to change their profiles, add new profiles, or delete some profiles. The granularity of the definition of a profile depends on the power of CC/PP and RDF vocabulary.

Multimedia Manager (MManager)

The Multimedia Manager (MManager) contains two sub-components: The Multimedia Content Manager (MCM) and the Multimedia Adaptation Manager (MAM). The MCM manages the caching of multimedia SOAP response messages on the local disk of the proxy server. The MAM handles media adaptation if necessary, in order to achieve device independence. In some cases when the destination device is not capable of handling requested multimedia contents, the MAM may convert the media accordingly. In some other cases, the destination device may announce that it is able to receive some media contents after some adaptations are performed. For instance, a WAP phone could accept HTML code; however it needs to be transformed to WML code. In accordance with our extension to the CC/PP, when a user specifies her profile, she can stipulate the transformation algorithm she prefers.

Service Broker

The service broker consists of four sub-components: request manager, service manager, SOAP generator, and SOAP parser, as shown in Figure 4. The SOAP generator generates SOAP request messages; the SOAP parser parses incoming SOAP response messages; and the request manager is a request broker of user requests for Web services. A user may request a Web service that another user from the same LAN has previously requested from the same Web service. Therefore, the request manager will first check the local disk through the MCM. If the result is a hit, the request manager will pass the control to the service manager to send back the information. As a consequence, not only the remote service broker and the service provider will have less traffic, but also the response time may be largely shortened. Otherwise, the request manager will assign a unique id to the request, ask UPM to store to the request-profile mapping, and then call the SOAP generator to generate a SOAP request message and send it to the service provider over the Internet. The unique request id and the user profile id will be marked as *MustSendBack* and be sent together with the SOAP request.

The service manager is invoked when a SOAP response comes back from Internet. It will first call the SOAP parser to analyze the content of the result message. If the result contains multimedia information not coming together with the first message, the service manager will schedule to pre-fetch the corresponding media files to enhance the streaming of the media data. All the messages will be sent to the MCM to store in local disk before sending back to the original requester, so as to achieve caching at the proxy server level. As a result, the service provider might not be even on-line all the time but its content can still be available at the cache of the MWSAgent. For persistent multimedia data (e.g. a movie), this cached data may be immediately utilized. For time-varying data, however, a mechanism for determining if the data is stale will have to be employed. Considering scalably encoded or layered video information, for example, such objects have a “base” layer containing essential information, and one or more “enhanced” layers containing higher level information (Paknikar, 2000). The service manager will try to download the sub-nodes following the layers. It will attempt to download the lower layers before downloading higher and more enhanced layers. Meanwhile, the service manager will work with the UPM to launch the corresponding user profile. And then the control will be passed to SDM to deliver the result to the original service requester.

Service Delivery Manager (SDM)

The Service Delivery Manager (SDM) decides the priority and the order of the services to be sent back to the requesters, and decides how to split the returning information to appropriate devices if there are multiple devices. A set of criteria is kept at the SDM to be utilized to optimize the order of the delivery, such as request time, request priority, current available resources, etc. The details of the algorithms of the selection and adaptation will not be discussed in this paper.

Route Agent (RAgent)

SOAP messages are adopted to transfer requests and responses through the Internet between a MWSSAgent and a MWSAgent. A SOAP message in this paper contains multimedia information that normally possesses QoS requirements. As discussed in the previous section, a SOAP messages may have to travel through several heterogeneous intermediate networks before it finally reaches the MWSAgent. Each of these heterogeneous networks could support multiple protocols and the flexibility of selecting protocols dynamically (Banchs, 1998). Therefore, in order to satisfy the performance requirements of the multimedia SOAP message to be transported, each network may need to select the most appropriate network protocol. For example, a NACK-reliable multicast protocol (Floyd, 1997) should be adopted on an ATM network in order to increase performance. We propose the RAgent to achieve this goal. A RAgent resides at intermediate network nodes on the Internet. Generally we suggest that each active network install a RAgent as the high-level director of the network routing. For a legacy system network that does not install RAgent, SOAP messages can be passed without taking a look at the information it carries.

This paper focuses on the selection of protocols to increase the performance of network transport rather than the selection of pathrouting algorithms. The architecture of a RAgent is composed of four sub-components, as illustrated in Figure 5. The SOAP parser is responsible for parsing incoming SOAP messages, determining the QoS requirements and passing them to the Protocol Manager (PM). The PM receives QoS requirements, searches the Protocol Pool (PP), and finds out the appropriate protocol to use. The PP is the interface encapsulating all protocols registered in the corresponding network. The Protocol Registration Manager (PRM) handles the registration of new protocols to the network or removal of some protocols. Under the PP are two sub-

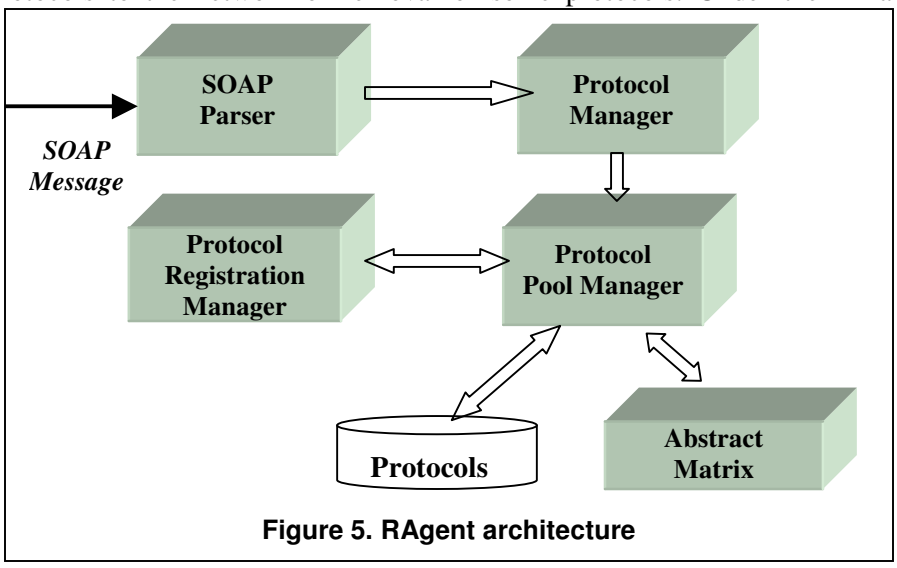

Figure 5. RAgent architecture

components: a cache of the registered protocols, and an abstract matrix that records the protocols and their QoS characteristics, such as reliability, real-time, security. The PRM maintains the abstract matrix when a new protocol is registered to the PP or removed from the PP.

When the PRM registers a new protocol to the protocol pool, it not only records its QoS properties, but also its cost. The cost here refers to the efficiency of the protocol. The more reliable a protocol is, the more costly it will be. For example, TCP is more costly than UDP because of the acknowledgement requirement. Table 1 is an example of the abstract protocol matrix. Three protocols are registered into the protocol pool: TCP, UDP, and NACKRMP. TCP and NACKRMP are reliable protocols; while NACKRMP is a real-time protocol. TCP and UDP can be used for unicasting to single node; while TCP, UDP, and NACKRMP can all be adopted for the purpose of multicasting. From Table 1 we can also conclude that TCP is more costly than UDP; and NACKRMP costs the most.

Table 1. Example of abstract protocol matrix

Protocol	*TCP*	*UDP*	*NACKRMP*
Reliability	**X**		**X**
Security			
Real-time		**X**	**X**
Uni-cast	**X**	**X**	
Multi-cast	**X**	**X**	
Cost	**4**	**1**	**5**

PERFORMANCE EVALUATION

The goal of our experiment is to design and conduct a simulation to evaluate the performance of our framework in a typical multimedia Web service application. The application we selected to implement is a distance-learning environment: multiple students request multimedia course information from servers. In such an application, one may assume that the students will be congregated at a set of proxy servers, and will be accessing a common set of material. Hence, we want to vary the number of service requesters and proxy servers, holding the other system components constant. To realize this performance analysis, we design the experiment based on the following assumptions. First, we assume that there is only one service provided by the system, and there is only one server machine that provides the service. Second, we assume that multiple students who reside on the same LAN request the service through one common proxy server. Third, we assume that there are three networks between the server and the proxy server: Ethernet and ATM and then Ethernet. A message from the server machine has to first pass through these three networks in the order specified before it hits the proxy server. Ethernet has TCP and is NACK RMP (Floyd, 1997) registered; and the ATM network has UDP and is NACK RMP registered. Suppose we require real-time QoS multimedia transportation. Fourth, we assume that the transportation time between a service requester and its corresponding proxy server is ignored due to the fact that they reside on the same LAN. Therefore, the problem is simplified and refined as shown in Figure 6. One server machine stores all of the course information and serves as the service provider; and multiple students request the same course information from this server machine as service requesters. Each student may possess one computer and some multimedia

devices, such as speakers or cell phones. All students reside on the same LAN and communicate to the service provider through a proxy server machine.

We set up the experimental environment as shown in Figure 6. The multimedia Web service is implemented as a normal J2EE-compatible Web application on the JRun application server (JRun). One MWSSAgent is installed on the server; and one MWSAgent is installed on the proxy server. Each of the three networks has one machine that applies RAgent. All of the three types of agents are implemented in Java. Both MWSSAgent and MWSAgent are also implemented as J2EE-compatible server on the JRun application server (JRun). The SOAP parsers and generators on all three agents are implemented by modifying the open source Apache Axis system (Axis). According to our experimental setup, the two RAgents in the Ethernet will choose TCP, and the RAgent in the ATM network will choose UDP to increase the performance.

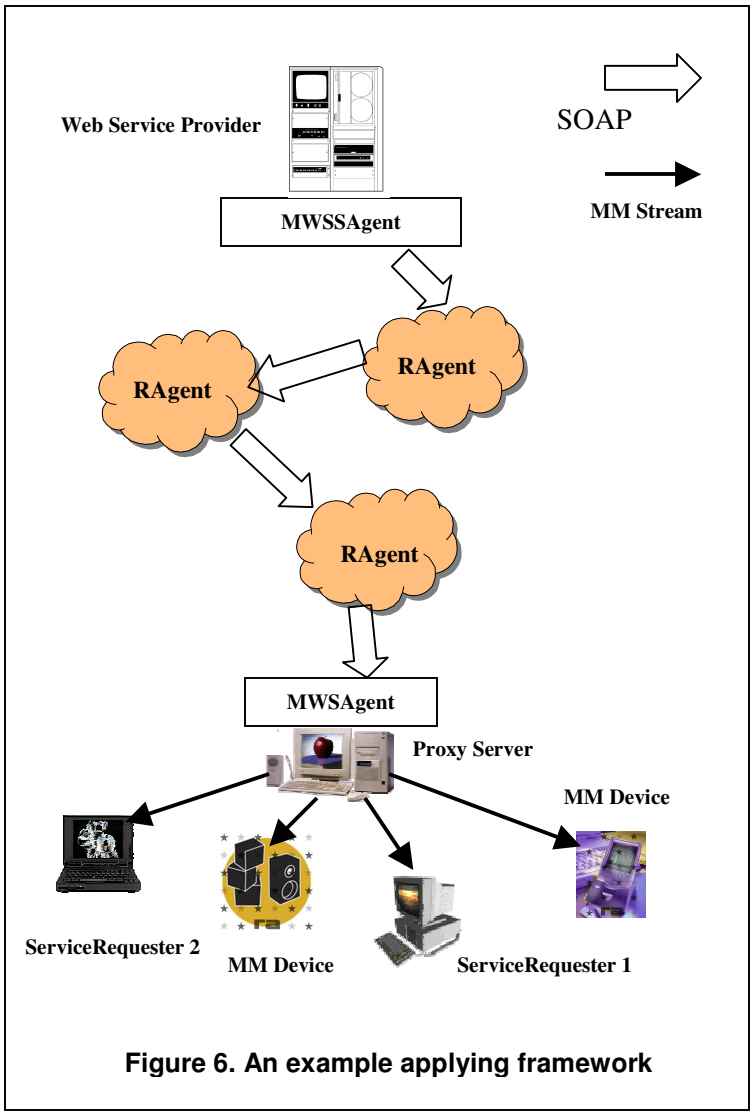

Figure 6. An example applying framework

We design the experiment such that, each client would create threads and then communicate with the server, requesting multimedia services. We perform tests on server machine applying a MWSSAgent and without a MWSSAgent. Moreover, we performed the same set of tests on service with one SOAP message (980KB) and two SOAP messages (980KB each). For our

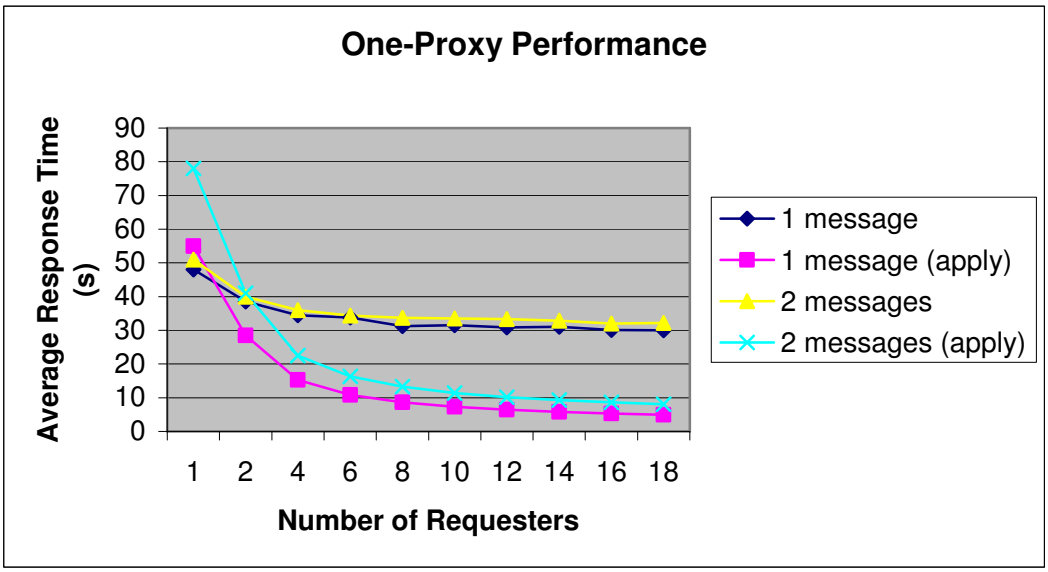

Figure 7. One-proxy performance

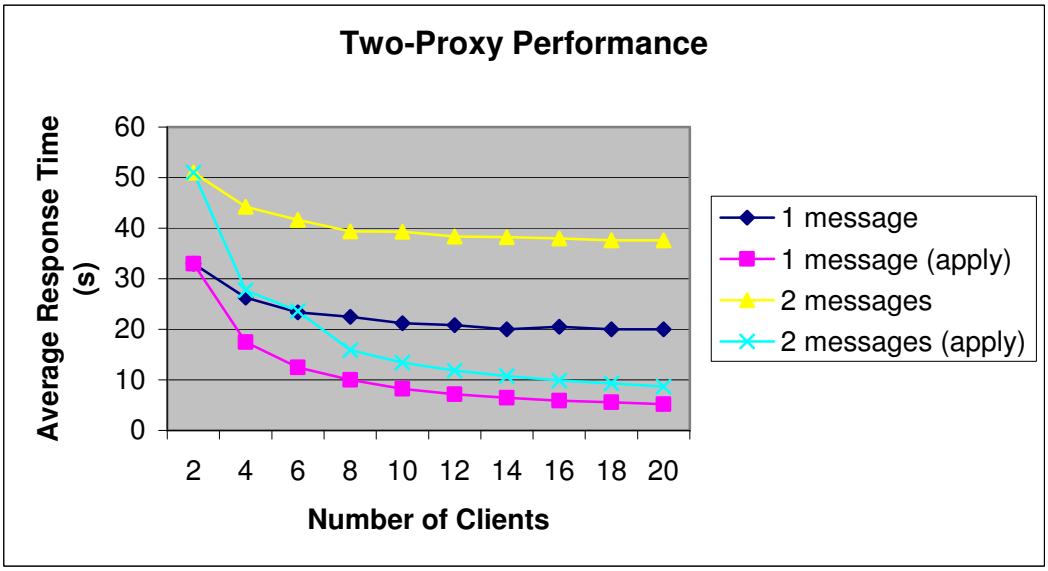

Figure 8. Two-proxy performance

experimentation we performed very low contention (1 client), low contention (2,4 clients), moderately low contention (6,8 clients), moderately high contention (10,12,14 clients), and high contention (16,18 clients). For each client, we record the response time, from the time the thread starts the request, to the time the entire multimedia information is received. Then the response times from all clients were averaged. The results performed on the four sets of situations are shown in Figures 7. Each point in the figure shows the averaged response time measured in seconds when there are specific numbers of clients requesting the service. From the figure, it is apparent that the system applying MWSSAgent outperforms the system without it. The higher number of clients the system supports, the higher the performance the MWSSAgent exhibits. Figure 7 also shows that the more clients a service provider needs to support, the higher efficiency our framework exhibits. Only when only one client requests a service, the system applying the MWSSAgent takes longer because of the extra operation time spent on the

MWSSAgent. However, the benefits justify this extra work as long as there are multiple requesters.

To test the effectiveness and efficiency of applying the MWSAgent, we relax one assumption from the first experiment, so that there are two proxy servers in the experimental system, and clients are averagely distributed behind these two proxy servers. We apply the MWSAgent on each of the proxy server. The results performed on the same four sets of situations are illustrated in Figures 8. The approach to measure the results is the same as that of the first experiment. The figure shows that the system applying MWSAgent outperforms the system without it. Figure 8 shows that the more clients share one proxy server, the higher efficiency the MWSAgent exhibits. It also shows that when there is only one client behind of a proxy server, the system applying the MWSAgent takes longer because of the extra operation time spent on the MWSAgent. However, again, the benefits justify this extra work as long as there are multiple requesters.

These two experiments show that our framework exhibits the distinct advantage of facilitating a multimedia Web service provider to support larger amounts of Internet clients, and shortening the average response time for service requesters.

ASSESSMENTS, INNOVATIONS AND FUTURE WORK

We present in this paper a service-oriented componentization model to support device-independent multimedia Web services. In the current infrastructure, we adopt caching and replacement algorithms introduced in (Paknikar, 2000) for both the MWSAgent and the MWSSAgent. Since these two agents serve different purposes, using the same set of caching and replacement algorithms may not be most efficient. In addition, this paper concentrates on protocol selection and dynamic binding, based on the QoS requirements carried by SOAP messages, on networks to support multimedia QoS requirements. To guarantee multimedia QoS requests, efficient routing algorithms are inevitable. Many challenging issues remain in the realm of multimedia Web service. The framework proposed in this paper is based on a much simplified problem domain.

Despite these limitations that could be improved or resolved by further work, our model extends research on multimedia Web services in several ways. First, the SOAP enhancements provide a simple way to improve the ability and flexibility of the ad hoc standard SOAP protocol to serve for multimedia Web services, by supporting batch facilities and QoS requirements. Second, the CC/PP enhancements increase the flexibility of the system for users to manage multi-devices and ensure device independency. Third, three types of intelligent agents are synergistically integrated to form a framework to support efficient service-oriented multimedia Web services. This model also facilitates caching and streaming of multimedia transport. In addition, our model seamlessly incorporates cutting-edge technologies relating to Web services: SOAP, XML/XSL, and the CC/PP. The result of this research can be applied to the software industry and serves as an architectural design to construct multimedia Web service applications.

We intend to continue our research work in the following directions. First, we will explore efficient caching, replacement, and pre-fetching algorithms so as to improve QoS performance. Second, we will attempt to bind SOAP to other more multimedia-oriented transportation protocols, such as Real Time Streaming Protocol (RTSP). Third, we will investigate QoS routing algorithms on routers to support SOAP QoS requirements. Fourth, we will pursue a formal description language to facilitate protocols to be published on the Web and dynamically

registered to networks. Finally, we intend to implement a mechanism to monitor multimedia QoS performances over different networks.

REFERENCES

Ferris, C., Farrell, J. (2003), What Are Web Services? *Communications of the ACM*, June 2003, 46(6), 31.

Benatallah, B., Sheng, Q.Z., Ngu, A.H.H. (2002), Declarative Composition and Peer-to-Peer Provisioning of Dynamic Web Services, Proceedings of the 18th International Conference on Data Engineering (ICDE'02). February 26-March 01, 2002, San Jose, CA, USA, 297-308.

Khan, M.F., Ghafoor, H., Paul. R. (2002) QoS-Based Synchronization of Multimedia Document Streams, Proceedings of IEEE 4th International Symposium on Multimedia Software Engineering (MSE'02), December 11-13, 2002, Newport Beach, CA, USA, 320-327.

SOAP (2004), Simple Object Access Protocol (SOAP) 1.1.

NORTEL (2004), http://www.nortelnetworks.com/products/01/mcs52/collateral/nn105360-091103.pdf.

Han, R., Perret, V., Naghshineh, M. (2000), WebSplitter: a Unified XML Framework for Multi-Device Collaborative Web Browsing. Proceedings of the ACM 2000 Conference on Computer Supported Cooperative Work (CSCW'00), 2000, Philadelphia, PA, USA, 221-230.

CCPP (2001), W3C Composite Capability/Preference Profile, W3C Working Draft, March 15, 2001.

Roy, J., Ramanujan, A. (2001), Understanding Web Services, *IEEE IT Professional*, November 2001, 69-73.

WSDL (2004), http://www.w3.org/TR/wsdl.

UDDI (2004), http://www.uddi.org.

Paknikar, S., Kankanhalli, M.S., Ramakrishnan, K.R., Srinivasan, S.H., Ngoh, L.H. (2000), A Caching and Streaming Framework for Mulitmedia, Proceedings of the 8th ACM International Conference on Multimedia, 2000, Marina del Rey, CA, USA, 13-20.

Pham, T., Schneider, G., Goose, S. (2000), A Situated Computing Framework for Mobile and Ubiquitous Multimedia Access Using Small Screen and Composite Devices, Proceedings of the 8th ACM International Conference on Multimedia, 2000, Marina del Rey, CA, USA, 323-331.

Kirda, E. (2001), Web Engineering Device Independent Web Services, Proceedings of the 23rd International Conference on Software Engineering (ICSE'01), 2001, Toronto, Ontario, Canada, 795-796.

Fagrell, H., Forsberg, K., Sanneblad, J. (2000), FieldWise: A Mobile Knowledge Management Architecture, Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW'00), 2000, Philadelphia, PA, USA, 211-220.

Werner, C., Buschmann, C., Fischer, S. (2004), Compressing SOAP Messages by using Differential Encoding, Proceedings of IEEE International Conference on Web Services (ICWS'04), July 6-9, 2004, San Diego, CA, USA, 540-547.

MPEG-21 (2004), http://xml.coverpages.org/ni2002-08-26-b.html.

SMIL (2004), W3C Synchronized Multimedia Activity Statement, http://www.w3c.org/AudioVideo/Activity.html.

JPEG (2004), http://www.jpeg.org.

HotMedia (2004), http://www-306.ibm.com/software/awdtools/hotmedia.

Banchs, A., Effelsberg, W., Tschudin, C., Turau, V. (1998), Multicasting Multimedia Streams with Active Networks, Proceedings of the 23rd Annual Conference on Local Computer Networks, October 11-14, 1998, Boston, MA, USA, 150-154.

Floyd, S., Jacobson, V., Liu, C.G., McCanne, S., Zhang, L. (1997), A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing, *IEEE/ACM Transactions on Networking*, 1997, 5(6), 784-803.

JRun (2004), http://www.macromedia.com/software/jrun.

Axis (2004), Apache AXIS Project, SOAP Protocol Implementation, http://ws.apache.org/axis.

ABOUT THE AUTHORS

Jia Zhang, Ph.D. Jia Zhang is currently an Assistant Professor of Department of Computer Science at Northern Illinois University, and also a Guest Researcher of National Institute of Standards and Technology (NIST). Her current research interests center around software trustworthiness in the domain of Web services, with a focus on reliability, integrity, security, and interoperability. She also has seven years of industrial experience as software technical lead in Web application development. Zhang received a Ph.D. in computer science from University of Illinois at Chicago in 2000. She is a member of the IEEE and ACM. Contact her at jiazhang@cs.niu.edu.

Liang-Jie Zhang, Ph.D. Liang-Jie Zhang is a research staff member and the founding chair of the Services Computing Professional Interest Community (PIC) at the IBM T. J. Watson Research Center. He is part of the Business Informatics research team with a focus on SOA and Web services for industry solutions and business performance management services. He has filed more than 30 patent applications in the areas of e-commerce, Web services, rich media, data management, and information appliances, and he has published more than 80 technical papers in journals, book chapters and conference proceedings. Dr. Zhang is an IEEE Senior Member and the chair of the IEEE Computer Society's Technical Steering Committee for Services Computing (TSC-SC). He was the general chair of the 2004 IEEE International Conference on Web Services (ICWS 2004) and the general co-chair of the 2004 IEEE Conference on E-Commerce Technology (CEC 2004). He is the Editor-in-Chief of the International Journal of Web Services Research (JWSR) and general co-chair of 2004 IEEE International Conference on Services Computing (SCC 2004). Liang-Jie received a B.S. in Electrical Engineering at Xidian University in 1990, an M.S. in Electrical Engineering at Xi'an Jiaotong University in 1992, and a Ph.D. in Computer Engineering at Tsinghua University in 1996.

Francis Quek, Ph.D. Francis Quek is currently a Professor of Department of Computer Science at Virginia Tech. He is also the director of Center for Human-Computer Interaction. His current research interests center around computer vision (dynamic vision, color, object recognition), computational multimodal language analysis, human computer interaction (HCI), medical imaging and visualization, video analysis for multi-media database access, and robot navigation. Quek received a Ph.D. in computer science in engineering from The University of Michigan, Ann Arbor in 1990. He is a senior member of the IEEE and ACM. Contact him at quek@cs.vt.edu.

Jen-Yao Chung, Ph.D. Jen-Yao Chung is the Chief Technology Officer for IBM Global Electronics Industry, where he is responsible for identifying and growing new technologies into future businesses for IBM. Before that, he was senior manager of the electronic commerce and supply chain department, and program director for the IBM Institute for Advanced Commerce Technology office. He has been involved in research on electronic commerce, electronic marketplaces, and Web application systems. Dr. Chung's current research is in the area of business process integration and management. Dr. Chung is the co-chair for IEEE technical

committee on e-Commerce. He has published 120 technical papers. He was awarded an IEEE Outstanding Paper award, two IBM Outstanding Technical Achievement awards, and an IBM Outstanding Contribution award. He is a senior member of the IEEE and a member of ACM. Dr. Chung received the M.S. and Ph.D. degrees in computer science from the University of Illinois at Urbana-Champaign. (http://www.research.ibm.com/people/j/jychung/)