Asian Journal of Information Technology 13 (12): 739-748, 2014
ISSN: 1682-3915

Enhanced Distributed Service Model for Video on Demand

[1]R. Lavanya and [2]V. Ramachandran
[1]Department of Media Sciences, [2]Department of Information Science and Technology,
Anna University, Chennai, India

Abstract: The conceptual Web Service Architectural Model for the VoD System that consists of policies, resources, services and messages is presented by including the life cycle for video service interaction and the relationship between the key modules of the video service. A performance enhancement for the innovative SOAP Communication Model with modified message exchange pattern is proposed where a MulticastMessageContext with MulticastMessageReceiver is introduced in the SOAP Communication Model to support unicast as well as multicast of video data as per the client requirements. This model makes the existing "one request-one response" message exchange pattern as "one request-many responses" message exchange pattern and inherently reduces the bandwidth requirements. In the implemented VoD Systems, the policies are handled by authentication service provider. The services are handled by either MessageContext or MulticastMessageContext. The resources are handled using download and play approach. The messages are handled using SOAP communication with modified message exchange patterns.

Key words: Web service, SOAP communication, message exchange pattern, video on demand, download

INTRODUCTION

The tremendous growth in the production and use of multimedia contents is due to the availability of large storage space, decreasing hardware cost, rapid increase in computing power and bandwidth that have contributed to numerous multimedia applications made available online which remained inaccessible in the past. The usage of web has entered into a new dimension and equipped with contents of social media, educational and entertainment videos, augmented reality and 3D multimedia. Due to the unpredictable growth of the internet and increasing demand for videos including movies and multimedia contents on the web, streaming of video files over the internet has received enormous attention both from academicians and industries.

Video on demand is a technology that provides not only entertainment on demand but also educational video on demand to all the subscribers of the service. Video on demand provides customers with informative and entertaining streams of multimedia and video information. Video on demand technology offers services such as movies, TV shows, e-Learning video materials and Interactive advertising where the customers can interact directly with full motion video advertisements over web. Thus, a Video on Demand System provides viewers with the ability to watch different videos from various service providers at any time. Video contents are down streamed to clients, who receive, decode and play the video information, always expecting a continuous and synchronized visualization during the streaming session. There is always enormous demand for video on the web with best resolution, less round trip time and without any delay. In order to meet the anticipation of the users, the current research works related to multimedia over web are focused to identify and develop the best suited design and architectures for streaming of video files over the Internet.

Ma and Shin (2002) made a remark that the network I/O bandwidth and the server storage I/O are still bottlenecks of VoD service. The VoD performance can be improved by multicasting which offers an efficient means of distributing videos to multiple clients. Wu *et al.* (2006) suggest that the high demand for on-line videos is not fulfilled because of the limited system capacity and the network resources. A scalable and inexpensive video delivery paradigm namely scheduled video delivery has been introduced to meet the QoS and to maximize the utilization of the resources. Lin *et al.* (2007) comments that providing large-scale on-demand video service is a challenging task. Many schemes such as batching, chaining, patching, periodic broadcast and piggybacking could save the media server's capacity by taking the advantages of IP multicast.

However, these approaches are difficult for deployment. The end system multicast is a feasible way to provide large-scale VoD services. Thus, the Advanced Chaining VoD (ACVoD) System which is based on peer to

Corresponding Author: R. Lavanya, Department of Media Sciences, Anna University, Chennai, India

peer technology makes use of the resources of idle users and makes the video chain more stable. Jung *et al.* (2008) explains the VoD environment that frequently groups the video requests to decrease the I/O demand and to increase the throughput. As the users may leave due to long response time, a good video scheduling policy namely hybrid resource sharing has been adopted. This scheduling policy not only takes care of the batch size but also decreases the waiting time. Xie *et al.* (2010) illustrates the patching based multicast technique that has been adopted for VoD in wireless mesh network, a next generation edge technology to provide broadband data access in residential, business and even in city wide networks.

Browsing streaming video quickly through internet with in the available bandwidth is a challenging one and it is also time consuming. The user has to browse, view and listen to the complete video to identify relevant contents. Thyagharajan and Ramachandran (2006) discusses the segmentaiton techniques for creating video summary and proposes a hierarchial scheme which decomposes a video sequence into different content resolution levels to enhance the transmission and user interaction. The handling of video streams through the use of an OODBMS was described by D'Acierno and De Pietro (1999). Scientific computing involves the construction of mathematical models and numerical solution techniques to solve scientific, social scientific and engineering problems. These models often require a huge number of computing resources to perform large scale experiments or to cut down the computational complexity into a reasonable time frame (Vecchiola *et al.*, 2009).

All the users expect that the video has to be fetched from the internet with best resolution, less round trip time without jitter, delay and essentially with high throughput. The distributors at the same time expect high profit, more secure and scalable environment. The attempt to display media files was started from the mid 20th century. Little progress was made for several decades, primarily due to high cost and limited capabilities of computer hardware and standards.

The main problem with successful provision of video on demand to the users is that there are the certain constraints such as bandwidth limitations at the server within the network and at the client side. There are plenty of sites that offer entertainment and educational video on demand services to the users for free of cost and as a paid service. Inspite of these massive services, the online video fails to meet the expectation of the users regarding quality of service, thereby facing the trough of disillusionment. Current web service standards enable publishing of video service descriptions and finding of video services based on method signatures. The video providers upload the videos to the server which can be accessed through distributed video retrieval services from any geographical location, thus made available to requesters. A conceptual web service architectural model which will link the various components of the VoD System is proposed as suggested by the World Wide Web consortium. This model provides clear understanding of the lifecycle behavior of video service interactions and enumerates the relationship between the key modules of the Video on Demand System.

As the use of internet is everywhere and the web provides an environment for all types of applications which include e-Learning, entertainment, etc., the security and scalability are the major concerns while uploading services and data to the web and later to access them. The proposed Web Service Architectural Model for the VoD System consists of four modules namely, policies, resources, messages and services.

The policy module focuses on the security issues such as permission, authentication and authorization that are created by the video distributors which are subjected to the client actions on a video file. The policies or the guidelines are created for accessing the video contents as well as the video service. The resource module focuses on the hardware and network resources and especially on the data resources like video/audio contents. The resource component of the proposed Web Service Model with regard to VoD System maintains all the details regarding the video resources including the ownership and the policies created for those resources.

The message module explains the way in which the messages or video contents are transported within the VoD System. It also concentrates on the structure of the request and response messages and the relationship between video providers and the video requesters. Any communication to the video service is in the form of SOAP request and response and the messages are processed in accordance with the policies.

The service module is on the top of the message module and it extends the same in order to explain the fundamental concepts involved in a service, i.e., the purpose of the communication within the VoD System and it examines the different actions performed by the clients in order to utilize the VoD services. The concept map shown in Fig. 1, depicts the relationship between the various key components of VoD Service Model.

The arrows represent that each component in the VoD System is partially layered on top of the other components.

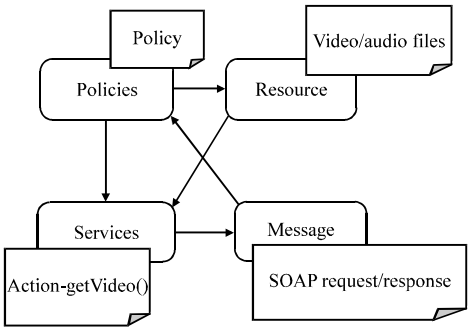

Fig. 1: Concept map of the VoD service

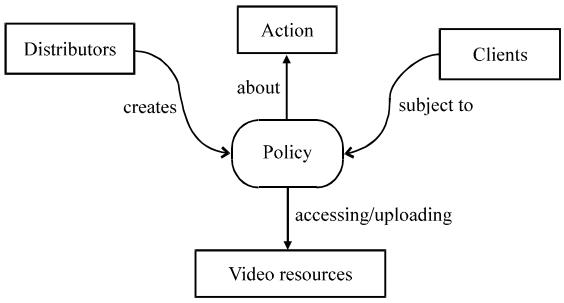

Fig. 2: Policy module for the VoD System

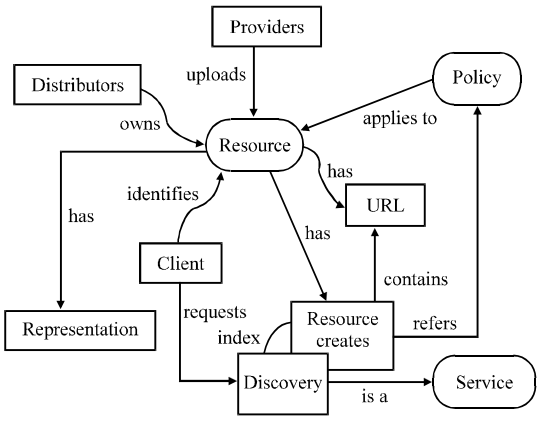

Fig. 3: Resource module for the VoD System

Policy module: The policy module focuses on various aspects related to the video files such as security and QoS in order to provide the VoD services. The distributor creates the policies regarding the video on demand services. The security of the VoD services is maintained by the audit and permission guards. The audit guard in the policy module is used to continuously track and monitor the clients, video resources and the video services are in consistent with the policies created by the distributors. If there is any violation, the permission guard enforces the corresponding action on behalf of the video distributors.

For instance, when the client requests for a video file, the audit guard checks whether the client is a registered user or not and the permission guard depending on the status of the client, specifies the appropriate permission grant whether to access or deny service. Figure 2 shows the policy module of the VoD services.

The security fundamentally emphasizes on the various constraints based on the client's action while accessing the video contents or while utilizing any service in the VoD System. Similarly, the quality of service is about the constraints on the video service. The QoS is associated with the delivery of messages/video contents by the SOAP mechanism. The delivery policy ensures reliable transmission of video data.

Resource module: The resource module shows the relationship between the video providers/distributors and the video resources. It maintains the details of all the video resources that are uploaded to the server and the information about the video provider. The resource module of the VoD System is given in Fig. 3.

It also ensures that whenever the resources are accessed, either by the video providers or video requesters, the corresponding tasks/events are carried out according to the predefined policies assigned by the distributors. The video requester identifies the resource location by requesting the discovery service which has been catalogued in the resource description. The resource description refers the policies that apply to the requested video resource and depending on which it returns the corresponding URL of the resource to the client for accessing the video. In dynamic discovery, the video requester may interact directly with the discovery service to find the appropriate resource. Discovery is the act of locating a machine-processable description of web service-related resources. The video requester entities will find the service description during development for static binding or during the execution for dynamic binding. A discovery service is a service that identifies the required resource for the client and also used to publish the service descriptors. The video provider uses this module to publish the video service descriptors. The uploaded resources are owned by the distributors. The client request for uploading or downloading a video resource can be either through HTTP GET or through HTTP POST Method which is a standard action associated with web resources.

Service module: The 'Service' module focuses on the action that can be performed by the users like either

requesting for a video or playing the video. The user can either be a video provider or video requester. An action is typically expressed in terms of executing some fragment of a program. Service is a set of actions that form a coherent group from the point of view of service providers and service requesters. The service as shown in Fig. 4 provides the video files (resource) to the client.

The video provider uploads the video and the video requester uses the video service. Services are mediated by means of the messages exchanged between video requester and video provider. When the user requests for a service, it looks at the video service description that describes all the information regarding the service interface. The service interface declares the method signature depending on which the client can request the service. An endpoint is the realization of a video service interface. A VoD endpoint is a network location at which an implementation of a service interface is made. The messages are created according to the choreography which defines the sequence and condition through which the clients can exchange the messages in order to perform a task. For the user to complete the registration task, the sequence of operation would start with the request by the client for "signing up" to the service. The different form elements are sent as response to the user. The user fills up all the form elements and sends the filled in form. The services checks for the validation and if satisfied, the user is successfully registered with the service.

Message module: The message module focuses on the aspects of architecture that relates to processing of messages. A message represents the data structure passed from the sender to its recipients. The structure of the message is described in the video service descriptors. This module concentrates on the relationship between the message sender and the message receiver and also deals with the way in which the messages are transmitted. The message module does not much emphasize on the semantics of the message. The messages are transported by means of SOAP which represents the information needed to invoke a service or reflect the results of a service invocation and contains the information specified in the service interface description. The message module for VoD System is shown in Fig. 5.

For instance, to invoke a video service, the client has to provide the filename of the video which is attached within the SOAP body of the request message. Distributed video service communicates via message exchanges. A Message Exchange Pattern (MEP) is a template, devoid of application semantics, that describes a generic pattern for the exchange of messages between clients and servers. The MEP describes the relationship

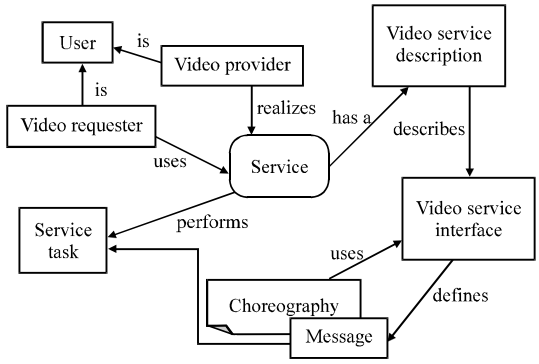

Fig. 4: Service module for the VoD System

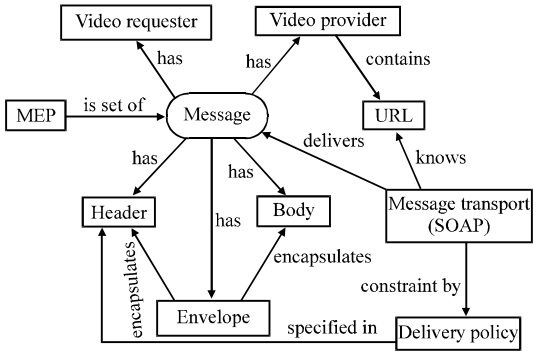

Fig. 5: Message module for the VoD System

of multiple messages exchanged in conformance with the pattern as well as the normal and abnormal termination of any message exchange. The message module defines the structure of a message that mainly contains an envelope which encapsulates both message header and body. The header holds the information regarding the message and the message body contains the content and the URL of the requested video data. All these modules form an integral part of the web service architecture for Video on Demand System.

SOAP COMMUNICATION MODEL FOR VIDEO ON DEMAND

The SOAP communication is emerging as a standard way to send XML documents over the internet from different platforms and thus enhancing the interoperability of web services. XML web services are already used as middleware components for interacting internet applications. The SOAP envelope is structured in XML as well and will be delivered by an arbitrary protocol by default HTTP. The SOAP with Attachment API for Java (SAAJ) conforms to the Simple Object Access Protocol specification and the SOAP with Attachments specification.

The XMLised Blob data is encapsulated within the SOAP envelope as well as can be included in the SOAP message as attachment in the proposed model. In the SOAP Communication Model with regular message exchange pattern, the XMLised Blob data is included as an attachment with the SOAP message and in the proposed multicast message exchange pattern the data is kept within the SOAP envelope (Lavanya and Ramachandran, 2014). As video piracy is the major concern, the proposed SOAP Communication Model for Video on Demand System ensures authentication of the client while delivering the video contents. To increase the security of the video contents and to permit different distributed services to run on various platforms, a SOAP Communication Model for the Video on Demand has been proposed with security enhancements.

The video service provider is designed in such a way that it will deliver video contents only to the authenticated SOAP clients. But at the same time, it is de-coupled from the authentication code. The authentication is earmarked to an Authentication Service Provider which uses an extended RPCMessageReceiver that will decide the type of the SOAP clients, i.e., a video provider or a video requester. Based on the type of the client, the read or write permission will be given in the form of a messageID. The video service provider uses this messageID to allow the client either to upload or download the video contents. Both authentication and video service providers use request-response message exchange pattern in which both request and response are XML based SOAP envelopes created using AXIS based Object Model (AXIOM), i.e., OMElement references. The block diagram representation of the proposed SOAP Communication Model is given in Fig. 6.

The RawXMLINOutMessageReceiver associated with the video service provider invokes the getVideo() or uploadVideo() method based on the Message ID as sent by the client. The predefined invokeBusinessLogic() behaviour of RawXMLINOutMessageReceiver invokes findOperation() behaviour based on the information contained in the Message ID. The invokeBusinessLogic() method has two MessageContext parameters, namely, msgContext and newmsgContext where msgContext represents the SOAP request from the client and newmsgContext represents the SOAP response.

The Message ID as provided by client is parsed within the invokeBusinessLogic() in order to find the appropriate service to be invoked. If the operation to be performed is defined within the video service provider, the invokeBusinessLogic() invokes the method using reflective middleware and receives the response as OMElement reference from the method and uses this response to formulate the SOAP envelope of the

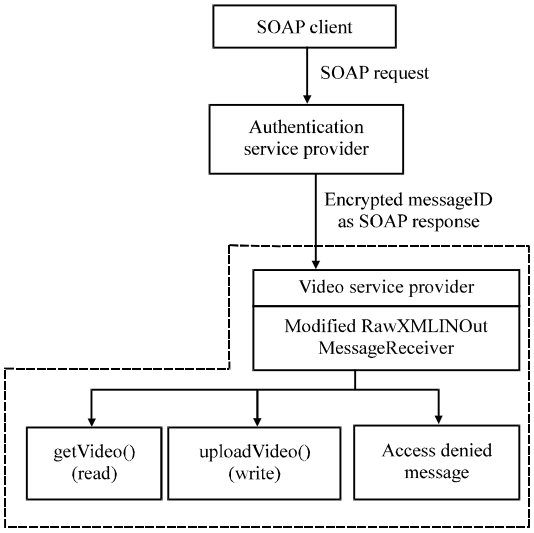

Fig. 6: SOAP Communication Model for VoD service

newmsgContext which is its response parameter. The behaviour of invokeBusinessLogic() is suitably modified which is given as below and the same is incorporated in the RawXMLINOutMessageReceiver.java source code. The modified version of RawXMLINOutMessageReceiver class file is copied into axis2-kernel-1.4.1.jar file which is included in the axis2.war. The modified war file is deployed in the Tomcat Web Server and hence an exclusive message exchange pattern for Video on Demand System is achieved.

In this SOAP Communication Model, the SOAP request is transferred to the message receiver using the OperationClient's execute() Method. The OperationClient will get the target endpoint reference using the options provided the requesters. The execute() method will take the request to the RawXMLINOutMessageReceiver and then calls the executeImpl() on the server side. The whole responsibility of executeImpl() is to implement in-out message exchange by appropriately creating the 'in' and 'out' MessageContexts. The executeImpl() method in turn calls the invokeBusinessLogic() of the message receiver by passing the 'in' and 'out' message contexts as parameters. The 'in' message context actually contains the request and the 'out' message context holds only the socket reference initially. It is the responsibility of the invokeBusinessLogic() Method to call the required service that will return the response as OMElement. The SOAP response is observed within the invokeBusinessLogic() and the same is encapsulated within a SOAP envelope of 'out' message context. The client receives the SOAP response and extracts the video data from the attachment as referred by video ID and will play it in a video player which is plugged-in onto the client application.

SOAP WITH MULTICASTING VIDEO ON DEMAND

The recent growth in use of the World Wide Web especially for multimedia applications in the internet has caused a significant increase in the demand placed on web servers. This increased load results in noticeably not only longer response times for users but also increased use of network bandwidth. The key problem in deploying large-scale video on demand applications is economy rather than technology. In order to have an efficient and cost effective Video on Demand System it is important to reduce the bandwidth requirements as much as possible. A typical Video on Demand System uses dedicated channels to provide the service to every user request. This increases the bandwidth requirement as more and more streams are requested. Multicasting is an approach whereby various customers can share a single movie stream resulting in reduced system cost per customer and improved system scalability. Multicasting has some limitations associated with it. For example, in order to share a single channel with a group of customers all the customers have to request the video at the same time and watch it without any interaction (no pause, fast forward, rewind, etc.). This defeats the purpose of a true and interactive Video on Demand System. There are certain multicasting techniques which can be used to provide interactive Video on Demand services.

Batching is one of the multicasting techniques in which the requested video is intentionally delayed by some amount of time, called a batching interval, so that subsequent request for the same video arriving during the current batching interval may be serviced using a single I/O stream. This trade off reduced I/O stream requirements for increased latency. Therefore, large batching intervals would seem to be incompatible with the notion of a true VoD System. Batching can only be used with popular videos since unpopular videos are unlikely to receive multiple requests during the short delay interval (Spicer and Kulkarni, 2001). Patching is another multicasting technique in which the multicasting stream is not static but can expand dynamically to accommodate new requests. For example, if there is a video being streamed through a local video server and there is a new request for the same video, then the video server starts caching the video data in its local disk. The server therefore needs an initial patching stream to transmit the leading portion of the video. When the leading portion of the video is transmitted, the rest of the video is transmitted from the data already buffered in the local disk (Hua *et al*., 1998).

Using the patching technique, channels are used only briefly to broadcast the first few minutes of the video, instead of being held up for the entire duration of the playback. Patching can be seen as a multicasting technique for interactive Video on Demand Systems where a single channel can be shared with multiple users. Patching also saves the storage costs as only a small amount of the video stream has to be buffered at the local disk depending on the requests. If there are new requests for the same video title then patching is used for that time period. Patching is very effective in reducing the bandwidth and storage requirements if the number of interactive requests from the users is within a certain limit. Beyond that, patching loses its efficiency as it results in starting multiple patches of the same video and increases the bandwidth requirements. In piggybacking approach, the playback rates are altered for the on-going video streams by 5% in order to merge the respective video streams into a single stream which can then serve the entire group of requests. The idea is to display the leading stream at a slower rate and the trailing stream at a faster rate. Then, assuming this interval is sufficiently small, the faster stream will eventually catch up with the slower stream. At this point the streams can be piggybacked or merged (Golubchik *et al*., 1995).

There are different piggybacking policies that can be used to achieve maximum savings in bandwidth. The three basic types of piggybacking policies are simple merging policy, the odd-even policy and the greedy policy. The first two are elementary as they involve at most a single change of speed for each video stream. Therefore, they can achieve a maximum of 50% improvement in the number of streams saved because the subsequent streams are paired. The greedy policy changes the speed of the different streams in such a way that the streams are merged at the earliest possible time in order to achieve maximum savings in bandwidth. In volatile storage approach, every single video title that is transferred to a video server is cached in the server's local disk completely. Apart from its local storage, each video server has a volatile storage where the video content is stored for a short period of time. This procedure enables any future requests to be served from the local server without any additional video streams required from the distant server (the only exception is when a future request is for a part of the video which is still not present in the local volatile storage). There are no restrictions in the number of interactive requests as all the requests are satisfied from the local video server. The only drawback with this approach is that it increases the storage cost as additional volatile storage is required at each video server connected to a network.

Popular websites such as You-Tube and Google video use progressive download approach to serve video

over the internet. If the video file is enriched with proper metadata, a HTTP streaming server is capable of streaming from any specific key frame (Zhu, 2008). The progressive download approach cannot be used for streaming live events; it can only be used with stored video files. At the same time, Liddell (2010), CEO of Panther Express, stated that "Most end users cannot tell the difference between the video delivered by progressive download and that delivered via a streaming video server". HTTP progressive download can work with any Content Delivery Network (CDN) service, thereby allowing faster access even for large number of users in disparate locations. The current web services standards are not sufficient to cater for multimedia streaming, the commonly used delivery method for large multimedia objects. Lam and Rossiter (2008) have proposed SOAP based streaming content delivery framework for multimedia web services. The framework uses an extension of existing SOAP standards which allows streaming content to be transmitted between two SOAP nodes over HTTP protocol. The extension includes a new SOAP streaming message exchange pattern and its corresponding SOAP HTTP binding.

Proposed framework: The proposed framework is capable of delivering quality adaptive multimedia content to users with heterogeneous configurations and requirements. When a multimedia file is requested, the current request-response message exchange pattern is not sufficient to handle one request and multiple responses. The researchers also proposed a service oriented architecture for the delivery of streaming multimedia which follows the standards of web services. The purpose of this model is to unify the e-Learning platform together with streaming multimedia in the web services domain. Hence, in order to reduce the load on the servers and to minimize the bandwidth requirements, the current research in this thesis is focused on the use of multicasting in the delivery of video contents by enhancing the conventional MessageContext of SOAP communication in Apache eXtensible Interaction System (AXIS). The existing "in-out" message exchange pattern is extended to "multicast in-out" message exchange pattern within the RawXMLINOutMessageReceiver of the SOAP Communication System in addition to the regular "in-out" scheme. The current MessageContext creates a new connection for every SOAP request and the latency associated with SOAP communications is huge while handling multimedia data. A multicast SOAP communication would have been a better solution to reduce the load on the server and to minimize the bandwidth requirements.

In the proposed Video on Demand Model using SOAP communication, the requests arriving for a particular video file are grouped but with restriction in the size of the group and the response is transmitted using a multicast connection from the server to the requested clients. The architecture of the proposed Multicast Video on Demand System using SOAP Communication Model is shown in Fig. 7.

The multicast connection is an additional behaviour that has been included in the MessageContext which has already been encapsulating a regular socket connection. This is achieved by introducing a new class MulticastMessageContext which extends MessageContext. Thus, the MulticastMessageContext provides regular HTTP response within the SOAP envelope if the request is for other than multimedia data. The requests are always sent to the server via an HTTP connection. The server classifies the requests as "requests for multimedia data" and "requests for other than multimedia data" using data typing services provided by FileDataSource. Based on the classification, the response is decided either to use socket connection or MulticastSocket connection.

The proposed model is designed in such a way that during a specific period of time, if there is only one request for the multimedia data, the response is sent back using the same HTTP connection that had carried the request. A vector of video/audio file names are maintained in the server and if there are requests for these data, the requests are grouped and a multicast connection is established for each group. If there are multiple requests simultaneously within a specific period of time from various users to the same multimedia data whose file name is included in the vector, the multicast connection is established. The proposed system also supplements the multicast service to the registered clients for a specific type of multimedia data. The overhead while making the decision of using regular socket connection or multicast socket connection is negligible while compared to bandwidth requirement due to single HTTP connection

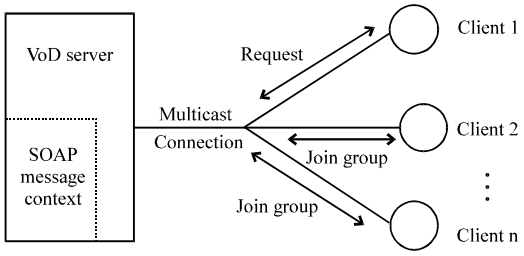

Fig. 7: Proposed multicast SOAP Communication Model for VoD System

for every request and the load on the server while transferring large amount of same data repeatedly. The server is forming a multicast group for those clients requesting the same multimedia data.

A single multicast IP address is used for a multicast group and all the clients need to explicitly join that group. The entire request URLs are mapped to that multicast IP address. The major advantage of the proposed model is that the client can have its own option by sending an indication to the server that it will support a multicast response for its request. The server listens for connection requests and receives HTTP requests once a connection is established. When the requests are for the same multimedia data then a new connection is established with the multicast IP address used to satisfy those requests. If the server receives a request other than multimedia data, the request is satisfied using regular way of unicasting over the request connection.

Implementation of multicast message exchange pattern: web services use SOAP for messaging. SOAP is an XML protocol that provides an envelope which encapsulates XML data for transfer through the web infrastructure. SOAP-over-UDP decouples SOAP from its underlying layers and defines a binding for SOAP envelopes to UDP. It uses video service endpoint address to support the message exchange patterns as SOAP-over-HTTP. In the current web service, the exchange of information is of single request and single response only. The SOAP Message Exchange Pattern (MEP) defines properties for exchange of SOAP messages required by a communication protocol HTTP. The two message exchange patterns namely request-response MEP and response MEP are suitable for RPC representation of web services where for each request there is only one response. This is not suitable for multicasting the video files as the video contents are divided into multiple small segments and there should be multiple responses. The requirements for implementing the multicast MEP are given below:

- The request is given only for joining the group and not for fetching the video. Once the client joins the group, the video has to start playing from the same scene that the other users in the group are currently viewing
- As the messages are sent through UDP, the video files are divided into small chunks of data that the UDP packet can hold. The message exchange should support multiple responses for a single request

A new SOAP message exchange pattern is required for video multicast in order to send multiple SOAP responses in a single HTTP response. This can be achieved by adopting the multipart structure in the Multipurpose Internet Mail Extensions (MIME). Hence, the proposed video multicast uses SOAP for message exchange between the video service provider and video requester by adopting multipart/related MIME type. The video file is divided into small chunks of 48 kb less than the size of the UDP packet can hold. These packets are transmitted one after the other from the server to the clients continuously. The client receives these packets and starts playing the video file. The workflow of the proposed SOAP MEP for video multicast is shown in Fig. 8.

The proposed multicast video service provider has two methods whose signatures are given below:

- Public OMElement getVideo(OMElement request)
- Public OMElement multicastVideo(OMElement request)

The input parameter of both methods has <responseType> tag which has the value either “regular” or “multicast”. The RawXMLINOutMessageReceiver handles the request if the client does not support multicast response. The customized in-out message exchange pattern associated with RawXMLINOutMessageReceiver in order to decide unicast or multicast communication is coded as follows:

```
SOAPBody body =
                    msgContext.getEnvelope().getBody();
OMElement element =
  body.getFirstChildWithName(new QName("http://video",
                                        "responseType"));
String resp = element.getText();
if (resp.equals(“regular”))
Method m = findOperation(op, VideoService,”regular”);
```

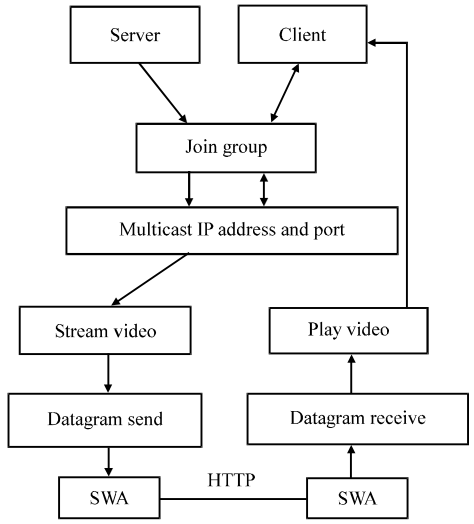

Fig. 8: Workflow of the proposed SOAP MEP for video multicast

If the response type of the request given by user is "multicast", the request is transferred to MulticastMessageContext which is being associated with MulticastMessageReceiver, a new class with the same behaviours as available in the RPCMessageReceiver namely, findOperation() and invokeBusinessLogic(). The modified method signature of invokeBusinessLogic() is public void invokeBusinessLogic (MessageContext msgContext, MulticastMessageContext newmsgContext) throws AxisFault.

The MulticastMessageContext class encapsulates MulticastSocket along with regular Socket as it is extends from SOAP MessageContext. In this case, there will be no change in the in-out message exchange pattern. In case of "multicast" response type, i.e., if the client supports multicasting, the request to the MulicastMessageReceiver is to join the multicast group. The model triggers the mulitcastVideo() service once the specified number of clients joined the group. The server starts the video multicast through the Multicast IPaddress and a specific port.

The clients who wish to view the video have to join the above multicast IP address to receive the video. Once the server starts streaming the video, the video data is divided into multiple datagrams and sent using SOAP with Attachment to the client, who receive the datagram and start playing the video. Thus, the main four operations of the MulticastMessageContext of the video multicast are as follows:

- Clients has to Join the multicast group using JoinGroup()
- The server sends the video to the members of the group
- The clients receives the video from the group
- After the video multicast stops, the client leaves the multicast group

SOAP MessageContext is extended by integrating multicast features for handling viewers of similar interest in a distributed environment. Multicast extends the traditional unicasting with efficient multipoint communications in which data can be sent to a set of destinations simultaneously belonging to a particular group. By sending same video content to multiple users, the better bandwidth utilization can be achieved with less host processing. The MulticastSocket is used to design multicast communication for the video service. The DatagramSocket class is used to create UDP socket connections that sends and receives the video data using DatagramPackets. The video client and server communicate over a UDP connected to a port on their machines. The client requests for a video file through a SOAP message and the corresponding video data is sent over DatagramSocket using DatagramPacket and each DatagramSocket contains a data buffer, the address of the remote host to send the data and the port number the server is listening on. The client request is implemented through a HTTP operation where the required video file name is sent to the server and the transmission of this operation will be request-response message exchange pattern. The SOAP message is sent to the multicast address. For the multicast transmission of video, the multicast service provider sends the SOAP message with the same attachment to the multicast address, thus, all clients will receive the same packet. The SOAP message over UDP restricts the size of the message to be transmitted over the network. Hence, the video data is divided into multiple small packets before transmitted.

CONCLUSION

A SOAP Communication Model is designed in accordance with the conceptual web service framework to implement the VoD System. In this model, policies are defined to authorize the clients and access controls are implemented for VoD services. Services are defined for the VoD System as per the client requirements whether to consume or upload the video contents. An SOAP Ommunication Model for VoD System has been developed in which the chaining of MessageContexts is achieved, i.e., the SOAP response of one service is forwarded as SOAP request for another service. An authentication service provider is introduced in between the video requester and the video service provider to enhance the security of VoD services. The authentication service provider protects the video content which comes under its control by using inherent password and provides appropriate authentication to the users based on their credentials. It generates a SOAP response which contains a messageID and forwards the same to video service provider. The message exchange pattern of video service provider is modified so that it can parse the messageID and based on the permission provided whether "read" or "write" or "fail", it has to initiate the invocation of appropriate method in the video service provider.

The video file will only be accessed by the client as per the permission granted by the authentication service provider as well as while accessing the video file, it expects from the user to provide correct password. The user given password is matched with the inherent password associated with the video file for which the existing behaviour of SecurityManager is extended by redefining two of its methods, checkRead() and

checkWrite(). Finally, the permission is based on the system level security permissions as provided in the ".java.policy" for each client.

A multicast SOAP message exchange pattern is implemented which will support one request-multiple responses for the VoD service using SOAP Communication Model. When there is a request for the same video contents from multiple clients, the developed VoD service will provide one response and the same will be dispatched to the clients who have joined in the multicast group. This is achieved by extending the SOAP MessageContext to include MulticastSocket and its associated behaviours. A new class named MulticastMessageContext which extends the MessageContext is introuduced which encapsulates both unicast and multicast behaviours. The user's request is analyzed within the modified message exchange pattern and appropriate services will be invoked to handle unicast or multicast communication.

Based on the decision by the message exchange pattern, the method getVideo() for regular unicast or the method multicastVideo() for multicast will get invoked. The method getVideo() uses regular MessageContext and there is no change in the message exchange pattern associated with it. The method multicastVideo() uses MulticastMessageContext whose message exchange pattern is modified in its business logic which has regular MessageContext for input and MuticastMessageContext for output. The clients' requests are initially grouped and the multicast service provider assigns a multicast IP address and the port to that group. This model is implemented in order to reduce the bandwidth requirements. Hence, in the implemented VoD Systems, the policies are handled by authentication service provider. The services are handled either by MessageContext or MulticastMessageContext. The resources are handled using download and play approach. An efficient strategy has been adopted to download the multimedia data with reduced retrieval time and without memory constraints. The messages are handled using SOAP communication with modified message exchange patterns.

REFERENCES

D'Acierno, A. and G. De Pietro, 1999. A distributed object oriented approach for parallel VoD systems. Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, June 28-July 1, 1999, Las Vegas, NV., USA., pp: 2122-2128.

Golubchik, L., J.C.S. Lui and R. Muntz, 1995. Reducing I/O demand in video-on-demand storage servers. ACM SIGMETRICS Perform. Eval. Rev., 23: 25-36.

Hua, K.A., Y. Cai and S. Sheu, 1998. Patching: A multicast technique for true video-on-demand services. Proceedings of the 6th ACM International Conference on Multimedia, September 13-16, 1998, Bristol, UK., pp: 191-200.

Jung, H.C., Y.T. Chuang, C.T. Guan, Y.C. Luo, K.W. Hu and C.H. Chen, 2008. A hybrid priority-based video-on-demand resource sharing scheme. Comput. Commun., 31: 2231-2241.

Lam, G. and D. Rossiter, 2008. A SOAP-based streaming content delivery framework for multimedia web services. Proceedings of the IEEE Asia-Pacific Services Computing Conference, December 9-12, 2008, Yilan, Taiwan, pp: 1097-1102.

Lavanya, R. and V. Ramachandran, 2014. Soap communication model for video on demand. Int. Rev. Comput. Software, 9: 128-134.

Liddell, S., 2010. Streaming or progressive download. Panther Express Inc., New York, USA., November 11, 2010.

Lin, F., C. Zheng, X. Wang and X. Xue, 2007. ACVoD: A peer-to-peer based video-on-demand scheme in broadband residential access networks. Int. J. Ad Hoc Ubiquitous Comput., 2: 225-231.

Ma, H. and K.G. Shin, 2002. Multicast video-on-demand services. ACM SIGCOMM Comput. Commun. Rev., 32: 31-43.

Spicer, S. and S. Kulkarni, 2001. VOD deployment: A discussion paper. Interactive Media, Telstra Research Labs (TRL), Australia, April 2001.

Thyagharajan, K.K. and V. Ramachandran, 2006. Optimal buffering requirement analysis for jitter-free variable bit rate video streaming. Int. J. Comput. Syst. Sci. Eng., 21: 161-172.

Vecchiola, C., S. Pandey and R. Buyya, 2009. High-performance cloud computing: A view of scientific applications. Proceedings of the 10th International Symposium on Pervasive Systems, Algorithms and Networks, December 14-16, 2009, Kaohsiung, Taiwan, pp: 4-16.

Wu, M.Y., S.J. Ma and W. Shu, 2006. Scheduled video delivery-a scalable on-demand video delivery scheme. IEEE Trans. Multimedia, 8: 179-187.

Xie, F., K.A. Hua and N. Jiang, 2010. Optimizing patching-based multicast for video-on-demand in wireless mesh networks. Int. J. Commun. Syst., 23: 1057-1077.

Zhu, L., 2008. Efficient video delivery over the internet. Digital Web Magazine, May 13, 2008. http://www.digital-web.com/articles/efficient_video_delivery_over_the_internet/.