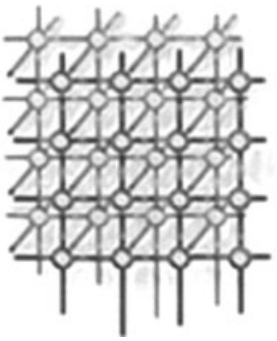
CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. (2009)
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe.1400



A biologically inspired framework for multimedia service management in a ubiquitous environment

M. Shamim Hossain*,†, A. Alamri and Abdulmotaleb El Saddik

Multimedia Communications Research Laboratory (*MCRLab*), *School of Information Technology and Engineering* (*SITE*), *University of Ottawa*, *Ottawa*, *Ont.*, *Canada K1N 6N5*

SUMMARY

This paper addresses several key issues in distributed multimedia services management and composition such as scalability, heterogeneity, and quality of service (QoS). The proposed framework introduces biologically inspired multimedia service management through the composition of basic multimedia services such as streaming services and different transcoding services. The biologically inspired approach is used for collecting the QoS requirements from individual transcoding services in order to select the most suitable services for the desired composition process. A prototype of the proposed framework is designed, implemented, and evaluated in terms of scalability and load balancing. Copyright © 2009 John Wiley & Sons, Ltd.

Received 18 May 2008; Revised 21 August 2008; Accepted 17 September 2008

KEY WORDS: multimedia service management; transcoding service; service-oriented architecture (SOA); biologically inspired approach

1. INTRODUCTION

The emergence of the Service-Oriented Architecture (SOA) paradigm, where numerous services dynamically join and leave the pervasive network environment has resulted in the proliferation of distributed multimedia services (e.g. multimedia transcoding services and streaming services) at any time from any pervasive device through any network. However, the heterogeneity of resources, the quality of service (QoS) demand of distributed rich multimedia services and the changing network characteristics pose a challenge for the management of those services in a ubiquitous environment.

*Correspondence to: M. Shamim Hossain, Multimedia Communications Research Laboratory (MCRLab), School of Information Technology and Engineering (SITE), University of Ottawa, Ottawa, Ont., Canada K1N 6N5.
†E-mail: shamim@mcrlab.uottawa.ca

One of the issues in multimedia service management is to get complex multimedia composite services dynamically from a number of available services and to maintain this composite service in a highly distributed and ever-changing network environment.

It is desired to access multimedia services ubiquitously through heterogeneous devices such as PCs, laptops, cell phones, and PDAs. Each of these devices has different capabilities and resources in terms of processing power, display (e.g. resolution and colors) and storage. Multimedia services are diverse in terms of the different operating systems that they are running on, the limited set of formats they support (M-JPEG, MPEG-4, H.263, H.264, etc.), and finally the incompatibility of supported codecs. QoS is also an important issue in dynamic service composition as well as service management. As each service has a common functionality, albeit with different QoS characteristics, a mechanism to select suitable services is required for a composite multimedia service. Flexibility is the ability to adapt an on demand service composition process that is customized for the users in a dynamically changing ubiquitous environment, where everyday new multimedia services appear and existing services disappear frequently and at any time.

To meet the above challenges, a framework for multimedia service management, which combines SOA [1] with a biologically inspired mechanism for seamless multimedia service composition in a ubiquitous environment, is expected to be a viable solution. In the proposed framework, a collection of services can be assembled to deliver composite multimedia services that are customized for the user's demand. This may include, but is not limited to, a streaming service, transcoding service, conferencing service, and registry service. Each type of multimedia service with similar functionality may have different QoS demands.

Currently, there are numerous existing researches related to web service management [2–4] and web service composition [5–9]; however, only a few have addressed multimedia service management [10,11]. Multimedia service management deals with multimedia service composition [12] and service maintenance (e.g. load balancing). Owing to (a) the rich semantics of multimedia content, (b) the complexity and dynamic characteristics (e.g. synchronization and continuous flow of stream) of multimedia applications and their metadata, and (c) the QoS demand of the user as well as multimedia itself, it is not easy to directly apply some of the existing solutions in web services composition into the multimedia domain [13,14].

The described framework is one of the few attempts toward bringing the combined potential of service-oriented architecture and the foraging behavior of ant colony concepts [15] into multimedia service management research. The proposed framework is evaluated through implementation as well as simulation in terms of scalability, load distribution, and some QoS parameters. The scalability was evaluated in terms of concurrent service requests.

The remainder of this paper is organized as follows. Section 2 describes some related studies and research challenges. Section 3 gives an overview of the system based on the SOA paradigm. It also describes the detailed system architecture. Section 4 presents brief implementation details, the experimental results, and the analysis. Finally, concluding remarks are made in Section 5.

2. RELATED STUDIES

Recently, interdisciplinary research specifically bringing inspiration from the metaphor of a bio-inspired process toward a multimedia system [16,17], distributed system [18], service composition

DOI: 10.1002/cpe

[19,20], service management [21], different communication network services and system [22,23] have been gaining attention in the research community. The reason behind such choice is twofold: (a) bio-inspired system has the natural capability of self-healing and robustness in dynamic and uncertain environments and (b) it is able to scale and adapt to the highly distributed, large scale, and dynamically ever-changing ubiquitous network environment.

SOA based on the realization of service-oriented computing is gaining significant attention in industry, science, and the academic world due to its advantages, which include scalability and the flexible composition of services that ranges from telecommunication to multimedia. SOA has been applied in different application domains such as mobile service [24,25] , Grid services for multimedia streaming in e-learning environment [26], optical network service [27], pervasive and ubiquitous environment[13], web application [28], collaborative service [29], Service management [2,11]; however, little is known about the use of SOA in the multimedia application domain other than the work mentioned in [26,30]. The challenge of SOA for multimedia service composition has been addressed in [12].

In [30], Wu *et al.* proposed a Global Multimedia Collaboration System (Global-MMCS) based on grid computing. This grid computing is parallelization of applications and systematic use of idle resources in the overall network environment. The authors attempted to integrate the SOA paradigm with grid computing, where XML-based General Session Protocol (XGSP) and simple object access protocol (SOAP) are used for creating and controlling VoIP conferencing services. Using XGSP schema, a Global-MMCS is developed, which can support and integrate services such as videoconferencing, instant messaging, and streaming. Mainly audio streaming was extensively used, video transcoding used to a limited extent; scalability was tested using a text chat session. In [11], Jin and Nahrsted proposed a QoS-aware service management framework where scalability was evaluated in terms of network size and client application's population size. This approach used a dijkstra-based algorithm for service path computation and service composition The service management includes the collection of QoS (e.g. bit rate, frame rate, resolution, delay) as well as functional information, composition (path finding), and service maintenance (tree rearrangement or resource adaptation and failure recovery.

Ding *et al.* [19] use bio-entity inspired by the immune behavior of a neuro-endocrine-immune system for web service composition and management. The system is regarded as a web service emergent system (WSES), where service composition is developed through web service emergence. In WSES, in response to a web request, the bio-entities set up the new composite web service through negotiation of the bio-entities. The system was evaluated in terms of energy consumption, response time, and adaptability.

In [21], Chiang *et al.* propose a bio-inspired framework for service management in a ubiquitous computing environment. An Ant Colony Optimization (ACO) meta-heuristics is adopted for the configuration of this network service component. As a proof of concept, they simulate the service configuration process (composition plan) for an e-mail application.

To the best of our knowledge, only a few researchers use biologically inspired concepts in the multimedia domain other than the previous work [16]. In this work, authors focus on a biologically inspired ant colony's foraging behavior in multimedia content repurposing, where authors make use of a biologically inspired service selection algorithm for selecting suitable repurposing services for heterogeneous network environments based on QoS. Our framework is somewhat consistent with [21] in using the ACO-based approach; however, we distinguish our work in many aspects. First, we

DOI: 10.1002/cpe

adopted an AntNet service selection strategy in the context of real time multimedia streaming for ubiquitous users. Second, in order to validate the above we implemented the proposed management framework for live multimedia streaming. Third, through simulation we also present some service management issues such as load management. We also distinguish our proposed framework from the previous work [16] in using the potential and promises of the SOA paradigm in order to fulfill the distributed ubiquitous access of multimedia service by composing a simple multimedia service component.

3. THE SOA-BASED FRAMEWORK FOR MULTIMEDIA SERVICE MANAGEMENT

The proposed multimedia service management system consists of three main components in addition to the client. The three components represent a variety of services that are depicted in Figure 1. These services are the registry service, the streaming service, and the transcoding service. Each service

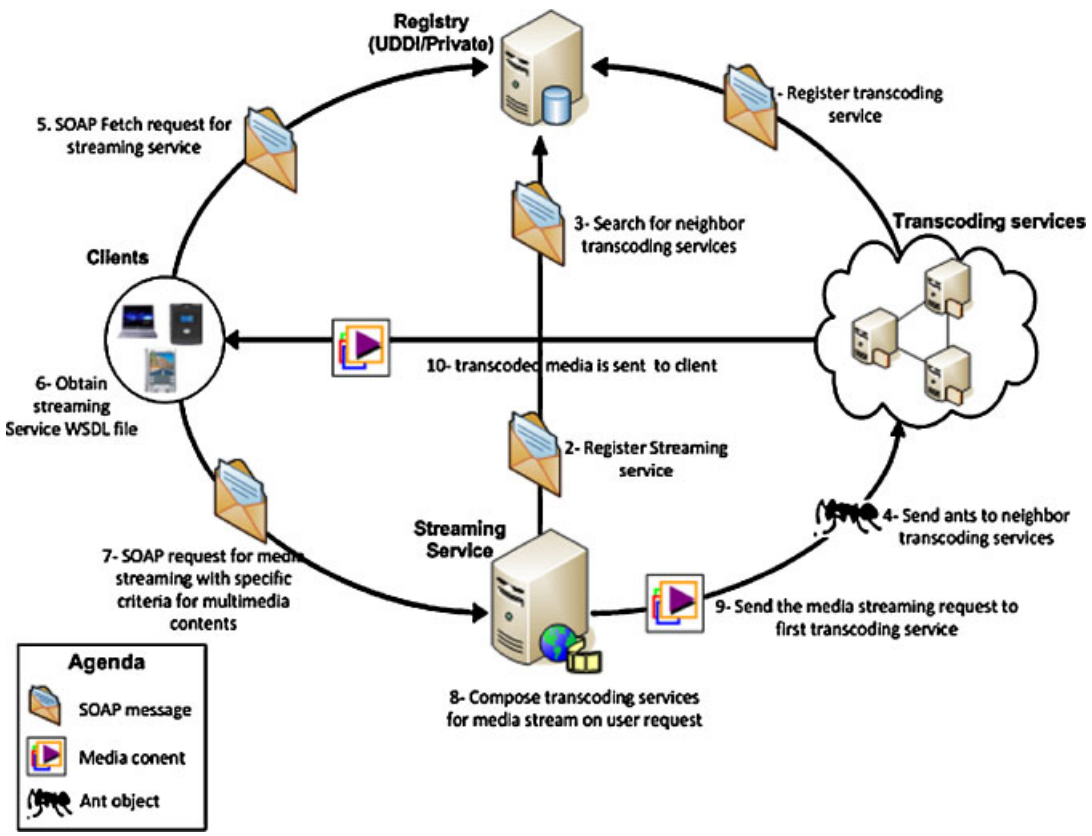


Figure 1. The SOA-based multimedia management system scenario.

DOI: 10.1002/cpe

has a different job or functionality to provide to its requestor. This can guarantee the flexibility, scalability, and extendibility of the architecture to include more services if needed.

The client first queries the registry to find out a streaming service. Then, it sends a request to the streaming service with specific QOS parameters. The streaming service is the service that originates or creates the multimedia stream. The originated stream can either be captured in real time by using a web cam, for example, or it can be an already stored content that the streaming server retrieves. The transcoding service transforms the media stream from one media format to another according to the request. In addition to the incoming stream, the transcoding service should accept a file from the requester. This file corresponds to a composite transcoding service that is constituted by multiple calls to primitive transcoding services in order to produce the final content as a reply to the client request.

Each transcoding service forwards its output stream to the next subsequent service based on the composite service plan until the last service, which forwards its output directly to the client. The framework relies on a registry, which is used for service query purposes. The registry is like a database that contains identification information about the different services in the architecture. After the creation of each service, the first step is to register the service in the registry using the correct keywords and under the proper classification that truthfully describes the service. Multiple transcoding services can reside in the host and usually this network host node is called a proxy server.

When the client requires a media stream, it first searches a streaming service from the registry. The fetch request returns the Web Service Description Language (WSDL) file [31]. The streaming service that geographically resides in the nearest point to the client is returned as a reply to the client fetch request. This is highly required to minimize network resource consumption. Using the WSDL file, the client generates a request to the streaming service using SOAP [32] and sends it to the service. The request consists of a call to the streaming service with parameters such as the format of the stream and the client location.

Upon receiving a request, the streaming service uses already gathered information by the bio-inspired component to create a composite transcoding service to satisfy the client request. The bio-inspired component works independently within each streaming service to dispatch ants periodically based on the information available at the registry to collect information about transcoding services. Then, the streaming service utilizes this information by a service composition system to compose a composite service as previously described.

When the composite transcoding service is created, the streaming service sends a transcoding request to the first transcoding service in the composite file followed by the media stream itself. The transcoding service transforms the stream into another form of media stream and forwards it to the next transcoding service preceded by the composite service file. Before sending the composite service file, each transcoding file eliminates itself from the file to ensure the sequence of execution. When the last transcoding service in the XML file list is reached, it sends its output directly to the client as a result of its request.In the following, we will discuss each of the services in more detail.

3.1. The streaming service

As depicted in Figure 2, the streaming service consists of three components: the bio-inspired service selection component, the composition component, and the execution component.

DOI: 10.1002/cpe

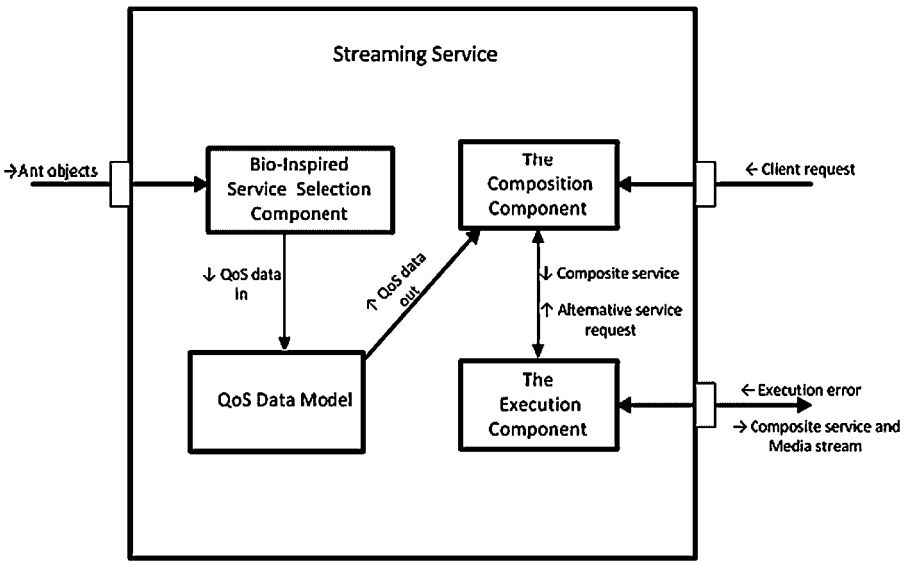


Figure 2. A decomposed view of the streaming service.

3.1.1. The Bio-inspired service selection component

The biologically inspired selection component is an independent system that resides at each streaming service in this SOA-based architecture. It is motivated from the foraging behavior of an ant colony metaphor [33] of AntNet. Using the main registry of the architecture, this system discovers other transcoding services and periodically dispatches ants to these services. Through this component, the streaming service is able to discover and collect QoS information about other transcoding services by dispatching ants, called forward ants, toward a targeted transcoding service. The collected QoS data are used to facilitate service discovery and selection, and eventually to optimize the composed process. To do so, the ant should know other neighbor transcoding services, and this can be achieved by inquiring the registry. When a targeted transcoding service fails to find other neighbor transcoding services, the forward ant dies and a backward ant is created and sent back toward the original streaming service. Upon receiving the backward ant at the original streaming service, the system strips the ant object and stores its collected data in a separate QoS Data Model to be used by the composition component.

3.1.2. The composition component

The composition component builds up a composition plan of multiple transcoding services to satisfy a client query. This component accepts a client request of a media stream with a specific format and size. Then, it un-marshals the request and identifies the criteria of the request that will be used to create the composition plan. The plan contains all the information needed to identify the transcoding services to be used and the client. After that the system utilizes the information available in the QoS

DOI: 10.1002/cpe

Data Model to find out possible composition plans that satisfy the client request. In case more than one plan is found, the component shall select the one with the highest QoS score. The composition plan is a file that plans the sequence of execution for the media transcoding services until the final transcoded media streams reach the client. It is comparable to the service configuration stated in [34]. Some of the examples of this composition plans are as follows:
$SC_n : MS \rightarrow TS_1 \rightarrow TS_2 \rightarrow TS_3 \cdots TS_n \rightarrow Clients(PDA, Cell\ Phone, Laptop\ etc.)$
For instance,
$SC_1 : MS \rightarrow TS_1 \cdots TS_n \rightarrow Cell\ Phone$
$SC_2 : MS \rightarrow TS_1 \rightarrow TS_2 \rightarrow TS_3 \cdots TS_n \rightarrow Laptop$
$SC_3 : MS \rightarrow TS_1 \rightarrow TS_2 \cdots TS_n \rightarrow PDA$
Where MS represents a multimedia streaming service, TS_n represents a transcoding service, and SC_n represents a composition service plan. Once the composition plan is created and finalized, the component hands the composite service to the execution component.

3.1.3. The execution component

The execution component runs the composition service plan, Upon receiving the plan, this component establishes and handles a communication link with the first transcoding service in the plan. Then, it starts the communication by sending the first composite service to that service followed by the media stream. Once the media stream ends, the components close the connection channel with the remote service. During communication, the component shall monitor and handle any exception or error that occurs in the execution. In case of execution failure, the system requests an alternative composition service plan from the composition component and tries to execute it.

3.2. The transcoding service

Figure 3 shows an architectural view of the main components that constitute the transcoding service: the dispatcher component and the transcoding component. Both components are further explained in the following.

3.2.1. The dispatcher component

The dispatcher component represents the interface for the transcoding service that mainly handles the functionality of the ant objects. When an ant is dispatched by the streaming service, it is usually sent as a streamed object toward the targeted transcoding service. This component is responsible for accepting incoming ant objects and forwards them toward their final destinations. Simply, it accepts the incoming ant object, performs some testing to recognize the final destination node for this ant, and forces the ant to jump to the next node if the current node is not the final destination. To do so, the ant should know other neighbor transcoding services, and this can be achieved by inquiring the registry. In addition, it allows the ant object to access the transcoding service properties usually fixed and kept in permanent storage.

In addition to the above, this component also acts as a load balancer, which is invisible to the users as they do not need to know the service from which they can be serviced. This component monitors the availability of appropriate services, dispatches the incoming multimedia composition

DOI: 10.1002/cpe

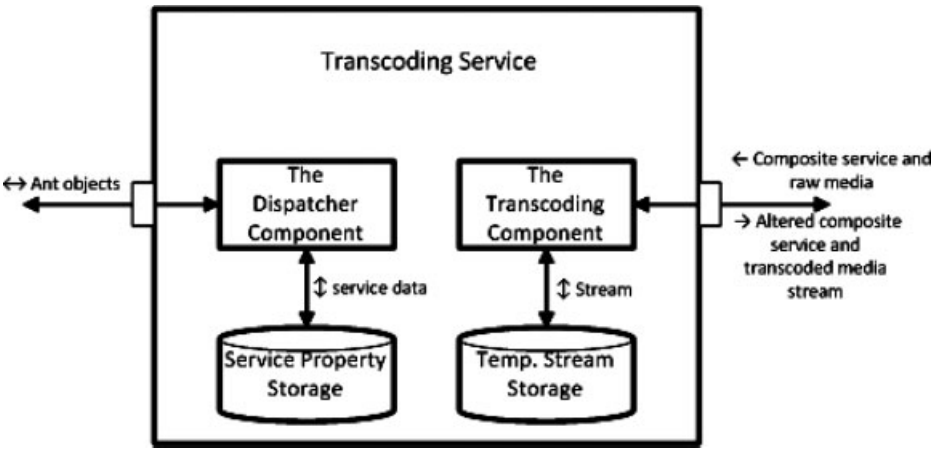


Figure 3. A decomposed view of the transcoding service.

request to the proxy server where transcoding services are running. It facilitates the selection of appropriate transcoding services and distributes the media server load based on the QoS collected by the bio-inspired selection component.

3.2.2. The transcoding component

The transcoding component is responsible for receiving a raw media stream and converting it into another media stream format. This component accepts a composite service plan followed by a raw media stream. Upon receiving the request, it first updates the composite service by removing the entry in the plan corresponding to its service node. Then, it initiates a connection to the next transcoding service in the plan and sends the composite service first to that remote transcoding service. Once it starts receiving the raw media stream, it initiates the conversion process and sends the converted stream immediately to the remote subsequent service using the previously established connection. In order to do that, the component utilizes temporary storage to store the converted stream frames before the sending process begins.

3.3. The client

The client is a viewer that allows the user to select and view a media stream of their choosing based on what is available in the registry. The end-user will only have access to the client and the client will only display what is available in the registry. That is to say that the client will only allow the user to select a media stream of a particular format and resolution given the registry states that a particular media stream is available and that it can be viewed at a particular resolution in a particular format. However, one should note that the content of the media stream and its format and resolution are independent of one another since the media content is determined by the streaming server and the format as well as the resolution is determined by the available transcoders. The client

DOI: 10.1002/cpe

is designed to be lightweight and can thus be installed on any machine in order to allow it to view the chosen stream.

3.4. The registry service

The registry is a permanent storage where the available media services, streaming services, and available transcoding services register themselves so that they may be discovered by the other services that need to use them. The streamers and transcoders are responsible for registering information about the video (e.g. QoS, format, and resolution) and they are able to output information pertaining to their particular 'contact' information (i.e. their address). The registry maintains a reference to all the available media (e.g. video) streams, proxy nodes, and transcoders, and where transcoders that have the same address are assumed to be on the same proxy node.

4. SYSTEM IMPLEMENTATION AND RESULTS

We implemented the proposed multimedia service management framework by following the SOA paradigm, where services are registered, queried (searched), and used. In this framework, we did not use UDDI as in [35]; however, we used a private directory as the registry. In our case, the registry is a MySQL database that contains three tables; the streaming services table, proxy nodes table, and the transcoding services table. Each table holds information regarding the type and quality of the stream provided by the corresponding service. The streaming service is implemented using Java 6.0 and the Java Media Framework 2.1.1e.

The streaming web service publishes, among others, a method that allows the users to request a multimedia (e.g. video) stream of a particular format and resolution through SOAP messages. The bio-inspired service selection component uses Java's capability to serialize and transmit ants to different transcoding services in the system responsible for collecting QoS information.

Using SOAP over HTTP is not suitable for delivering multimedia streamed content because SOAP messages introduce unnecessary overhead for the streamed media content This overhead includes among others, processing cost, message parsing, and a marshalling or unmarshalling process [6]. Consequently, we divided the implementation of our multimedia web services into two parts. The first uses XML-based SOAP messages over HTTP in order to gather all relevant information (such as port number, type of services, QoS, etc.). The second part uses socket communication in order to stream the media content.

In order to validate our framework; we deployed five transcoding services for real-time multimedia streaming for different ubiquitous clients. Figure 4 shows the top-level implementation of the proposed system described above for a video transcoding example. Each of the transcoding services belongs to a different composition plan, as stated in Section 3.1.2. In this test, an Intel Pentium 4 3.6 GHz Windows XP Pro SP2 with 1GB RAM PC was used as the media server. Five Intel Pentium 4 3.6 GHz Windows XP Pro SP2 with 1GB RAM PC were used as transcoding proxies where transcoding services were running. An Intel Pentium 4 3.5 GHz Windows XP Pro SP2 with 512 RAM desktop PC, a PDA (blackberry), one cell phone, and notebook computers were used as clients. Among these, some of the clients were connected through Wireless LAN 802.11 and some of these were connected through GPRS.

DOI: 10.1002/cpe

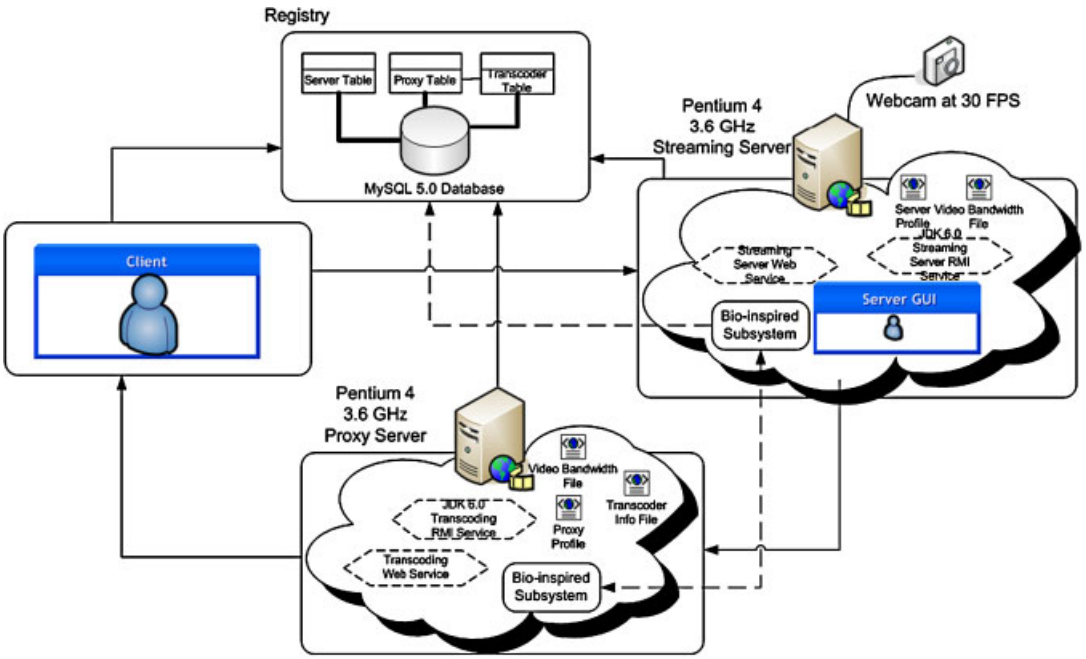


Figure 4. Detailed top level implementation.

Table I. Transcoding services with QoS (bit rate, frame rate, delay, etc.)

Service no.	Possible transcoding services(TS)	Average bit rate (Kbps)	Average frame rate (fps)	Average transcoding delay (ms)	Target client for the service
TS_1	Input = MJPEG 320 × 240, 30 fps Output = MJPEG 160 × 120	48	8.6	92	Cell phone
TS_2	Input = MJPEG 320 × 240,30 fps output = H.263 352 × 288	320	15.3	100.01	Laptop
TS_3	Input = MJPEG 320 × 240, 30 fps Output = H.263 176 × 144	96	14.5	98.7	PDA
TS_4	Input = H.263 352 × 288, 30 fps Output = H.263 176 × 144	64	9.2	99.1	PDA
TS_5	H.263 176 × 144, 15 fps H.263 128 × 96	28	8.8	93.5	PDA

4.1. Multimedia service composition

During the validation tests, we consider a number of transcoding services as shown in Table I. In this table, we list some transcoding services along with their input or output capabilities and different QoS attributes (transcoding delay, bit rate, and frame rate). For example, in row 1 we show that transcoding service 1 (TS_1) is capable of taking MJPEG 320 × 240, 30 fps as input and deliver

DOI: 10.1002/cpe

MJPEG 160×120 as output, which is suitable for rendering to the client (e.g. cell phone) at a bit rate of 48 kbps, frame rate of 8.6 fps, and a transcoding delay of 92 ms. It should be mentioned that we measured the transcoding delay as the time it takes for the captured video stream to go through the individual transcoders. Now, we describe two composition scenarios for the validation of the proposed framework.

At first, we consider an application scenario, where a wireless client (e.g a laptop connected to a Wireless LAN) is accessing a live video stream. The stream is originally captured from a webcam as MJPEG 320×240, at 30 fps and delivered to the client as H.263 352×288 at 15–20 fps with a bit rate of 245–330 kbps due to its capability of accepting such a media format. In this particular scenario, the client (Laptop) cannot play any MJPEG stream; it can only render an H.263 352×288 format, which requires the source content to be transcoded to the appropriate format. Given the list of transcoding services in Table I, the composition plan we used in this scenario is: $SC_1 : MS \rightarrow TS_2 \rightarrow Laptop$.

Here, *MS* is the multimedia streaming services and *TS* is the transcoding service used. Note that only one transcoding service is used in this presented case; the service with the identification number 2. Therefore, we can call this an atomic composition or simple composition.

We now take a complex composition scenario, where a wireless user (e.g PDA) has a GPRS connection with a maximum bandwidth of 28.8 kbps and can accept videos with a frame rate of around 9 frames/s for SQCIF (128×96) resolution. Similar to the above simple composition scenario, the video stream is originally captured from a webcam as MJPEG 320×240, at 30 fps. In this particular case, the user is unable to get service according to his QoS demands from the available transcoders' list. Now, the user (PDA) has to search for and select the appropriate transcoding service in order to render the multimedia stream that satisfies his QoS requirements, namely the format (H.263), the PDA's small display (128×96), as well as the low network bandwidth (28 kbps) and frame rate (8.1–9.4 fps). Given the list of transcoding services in Table I, the composition plan we used in this scenario is as follows: $SC_2 : MS \rightarrow TS_3 \rightarrow TS_5 \rightarrow PDA$. Note that two transcoding services (TS_3 and TS_5) are selected and composed with the streaming service. Therefore, we call this composition a complex composition.

4.2. Scalability

Scalability refers to the capability and flexibility of a system to adapt under increased load (e.g. increasing number of concurrent request) without sacrificing performance degradation [36]. Scalability should be evaluated to see how well the system works with the increased number of concurrent composition requests while meeting the user's demand by maintaining the performance of the system. The experimental results demonstrate our proposed framework's capability to scale under (a) an increased number of services in the system (i.e. service registry) and (b) an increased number of concurrent composition service requests by the users. Our SOA-based system provides linear scalability, until a certain extent, service load distribution, and increased throughput.

4.2.1. Scalability: average response time vs concurrent requests

To conduct the scalability tests, we ran the system for 1000 concurrent composition requests. As shown in Figure 5, the response time gradually increases in response to the increased number of

DOI: 10.1002/cpe

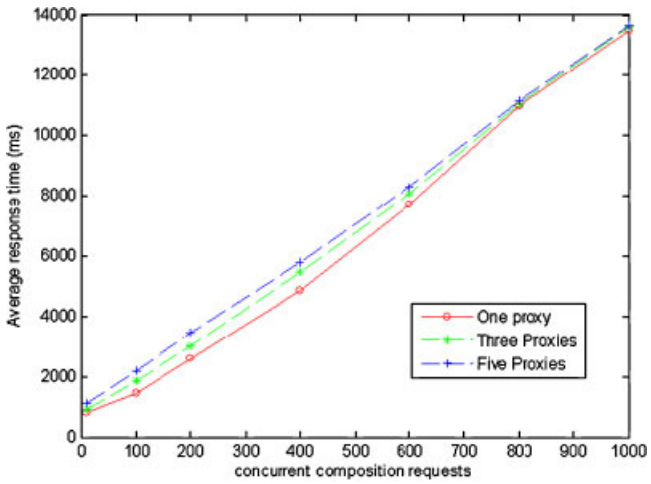


Figure 5. Scalability over increasing number of concurrent request with varying proxy servers.

concurrent requests. With a higher number of concurrent requests, the curve follows an exponential trend, where the average response time increases linearly with the number of concurrent requests. We also find that by adding the number of proxies (where multiple transcoding services are running) the response time does not increase dramatically. As seen from Figure 5, at 800 concurrent requests the average response time slightly increases in the system with three transcoding proxies (around 11 050 ms) compared with the system with five transcoding proxies (around 11 150 ms). From the results, it seems that the system is linearly scalable as the average response time increases linearly with the increasing number of concurrent requests and even with an increase in proxy servers, while serving the same rate of request per second (throughput).

4.2.2. *Scalability*: *the system's throughput vs concurrent requests*

In order to understand the predicted throughput in terms of service requests per second in our proposed system, we apply Little's law, which tells us that the average number of concurrent service composition requests (N) in the system is equal to the average arrival rate (λ) of the service request to that system, times the average time (T) spent in that system. This is expressed as follows: For instance,

$$N = \lambda T \tag{1}$$

$$\lambda = N/T \tag{2}$$

where, N is the number of concurrent request, λ is the throughput of N requests in terms of served request per second in the system, and T is the response time in seconds.

We used the PushToTest Testmaker 5.1 [37] for testing scalability over the system's throughput. PushToTest Testmaker 5.1 is suitable for testing SOA-based applications and web applications. In order to test the scalability, TestThread is required to send a request to the service. We ran one

DOI: 10.1002/cpe

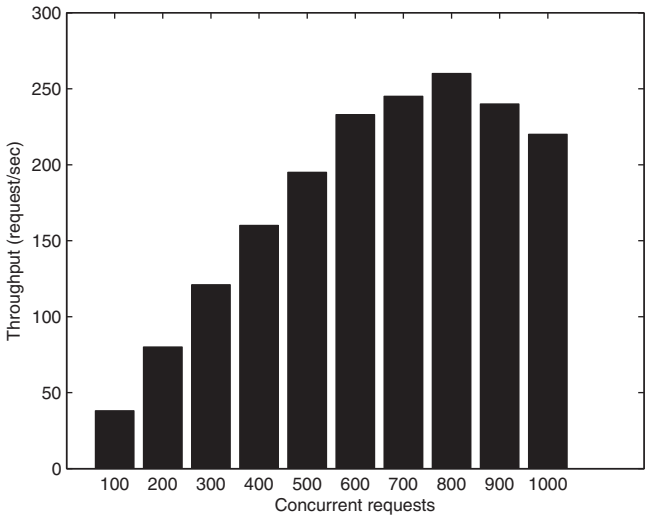


Figure 6. Scalability over systems' throughput (request per second).

TestThread for each concurrent request, record, and saved the test results to the log file. We ran the test for 1000 concurrent requests. For this test, we used the same media server as a web server for accepting requests, and the desktop PC as the test client where a number of concurrent TestTthreads were running.

Figure 6 shows that the increased throughput is directly proportional to the increased number of concurrent requests, i.e. throughput increases linearly with the number of received requests while keeping the average system response time constant regardless of the number of concurrent requests. As shown in Figure 6, at 200 concurrent requests, the proposed system handles 1600 requests in 20 s, which results in a throughput of 80 ($\lambda = \frac{1600}{20}$) requests per second with a 2.5 ($T = N/\lambda = \frac{200}{80} = 2.5$) second response time. In order to keep the response time at 2.5 s, the same system at 400 concurrent request handles 3200 requests in 20 s, which results in a throughput of 160 ($\lambda = \frac{3200}{20}$) requests per second. Therefore, we find that with the increase in concurrent requests, the throughput also increases in the same manner, which demonstrates perfect scalability. In other words, the increasing number of requests does not affect the system's response time. In this test, perfect scalability was achieved as long as we had about 800 concurrent requests. After that, it starts to decrease. In order to have the desired throughput or to solve this problem, the media server load has to be controlled or distributed to another serving site. In this case, the load is distributed to different proxy servers where transcoding services are running. A bio-inspired service selection component facilitates selecting the appropriate transcoding services and distributing the load. In the next section, this is described.

4.3. Load distribution

We tested the system for load balancing through simulation. We used multithreaded clients in Java to emulate the same number of concurrent requests as above (1000) and sent them to the media

DOI: 10.1002/cpe

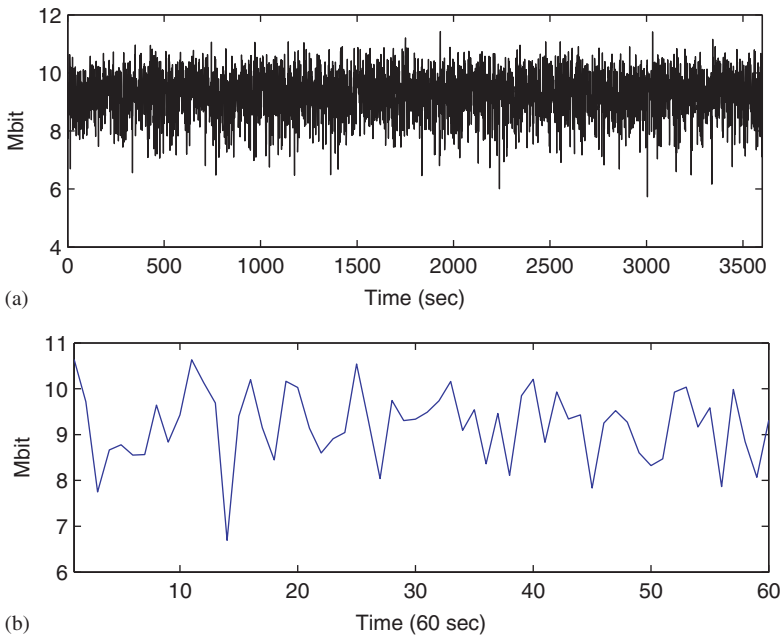


Figure 7. The media server's outbound traffic for the concurrent requests: (a) over a period of one hour (3600 s) and (b) over a period of 1 min (60 s).

server. The simulation was 1 h long and was repeated 5 times under the same computational power conditions. We configured our media servers to stream data at a maximum capacity of 10 Mbps.

Figure 7(a) shows that the server load can exceed 10 Mbps at some instances in time. To clearly present this fact, Figure 7(b) zooms a 60-s segment. Taking into consideration that the media servers' stream videos with a maximum bandwidth of 10 Mbps, it is obvious that there is a load balancing problem since the load is not constant. In order to solve this problem, the load should be distributed among other proxy servers. We use bio-inspired service selection algorithms as a distributed solution.

Figure 8(a) shows that our proposed bio-inspired approach described in Section 3 results in fewer overloads as the transcoding computation load is distributed among the different proxies. This is due to the dispatcher component (Section 3.2.1) that can dynamically distribute computational tasks to transcoding services. We compare the results of our proposed method with a traditional algorithm based on the distributed Bellman Ford [38,39]. As shown in Figure 8(b), our proposed method outperforms the traditional approach; on average, the overload is reduced by approximately 1.5 times.

5. CONCLUSION

The combination of SOA and a biologically inspired mechanism has the potential to fulfill the requirements and challenges of scalability, flexibility, and reusability of distributed multimedia

DOI: 10.1002/cpe

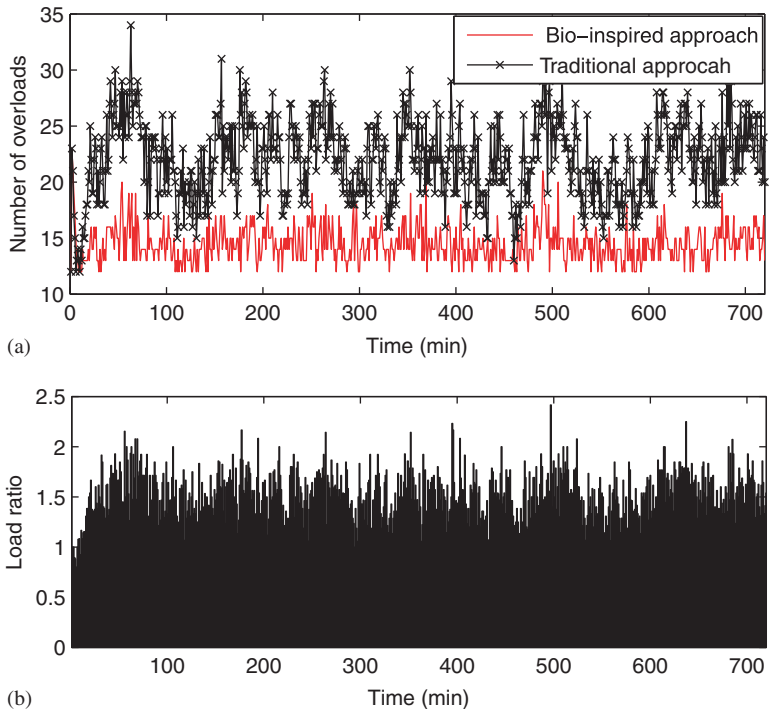


Figure 8. The comparison of our bio-inspired approach against the traditional load distribution approach.

delivery. To the best of our knowledge, this is one of the first ever works that brings together the SOA paradigm as well as a biologically inspired mechanism into the multimedia domain, and more specifically into multimedia service management for a ubiquitous environment. We evaluated and compared our proposed system through implementation and simulation. We discussed how our system scales under an increased number of composition requests. In the future we intend to explore robustness of the proposed framework, particularly under a hostile ubiquitous environment.

REFERENCES

1. Papazoglou MP, van den Heuvel WJ. Service oriented architectures: Approaches, technologies and research issues. *The International Journal on Very Large Data Bases* (*VLDB*) 2007; **16**(3):389–415.
2. Papazoglou MP, van den Heuvel WJ. Web services management: A survey. *IEEE Internet Computing* 2005; **9**(6):58–64.
3. Erradi A, Padmanabhuni S, Varadharajan N. Differential QoS support in web services management. *Proceedings of the International Conference on Web Services* (*ICWS '06*), Chicago, U.S.A., September 2006; 781–788.
4. Nowlan MF, Blake MB. Intelligent agent communication and collaboration for web services management. *Proceedings of the 16th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises* (*WETICE'07*), Paris, France, 18–20 June 2007; 18–23.
5. Alamry A, Eid M, El Saddik A. Classification of the state of the art dynamic web services composition techniques. *International Journal of Web and Grid Services* 2006; **2**(2):148–166.

DOI: 10.1002/cpe

6. Younas M, Awan IU, Duce D. An efficient composition of web services with active network support. *Elsevier Expert Systems with Applications* 2006; **31**(4):859–869.
7. Gu X, Nahrstedt K, Yu B. Spidernet: An integrated peer-to-peer service composition framework. *Proceedings of the 13th IEEE International Symposium on High performance Distributed Computing*, Honolulu, HI, U.S.A., 4–6 June 2004; 110–119.
8. Ponnekanti SR, Fox A. SWORD: A developer toolkit for web service composition. *Proceedings of the 11th World Wide Web Conference*, Honolulu, HI, U.S.A., 4–6 June 2004.
9. Jureta I, Faulkner S, Achbany Y, Saerens M. Dynamic web service composition within a service-oriented architecture. *IEEE International Conference on Web Services* (*ICWS 2007*), Salt Lake City, UT, U.S.A., July 2007; 304–311.
10. Bruneo D, Guarnera M, Zaia A, Puliafito A. Grid based architecture for multimedia services management. *Proceedings of the First European Across Grids Conference*, Antiago de Compostela, Spain, February 2003.
11. Jin J, Nahrstedt K. QoS-aware service management for component-based distributed applications. *ACM Transactions on Internet Technology* 2008; **8**(3):1–31.
12. Nahrstedt K, Balke WT. Towards building large scale multimedia systems and applications: Challenges and status. *Proceedings of the First ACM International Workshop on Multimedia Service Composition*, Hilton, Singapore, 2005; 3–10.
13. Kalasapur S, Kumar M, Shirazi B. Seamless service composition (SeSCo) in pervasive environments. *Proceedings of the First ACM International Workshop on Multimedia Service Composition*, Hilton, Singapore, 2005; 11–20.
14. Nahrstedt K, Balke WT. A taxonomy for multimedia service composition. *Proceedings of the 12th ACM International Conference on Multimedia*, New York, NY, U.S.A., 10–16 October 2004; 88–95.
15. Dorigo M, Di Caro G, Gambardella LM. Ant algorithms for discrete optimization. *Artificial Life* 1999; **5**(2):137–172.
16. Hossain MS, El Saddik A. A biologically inspired multimedia content repurposing system in heterogeneous network environments. *Springer/ACM Multimedia Systems Journal* 2008; **14**(3):135–143.
17. Hossain MS, Alamri A, El Saddik A. A framework for QoS-aware multimedia service selection for wireless clients. *Proceedings of the 3rd ACM Workshop on Wireless Multimedia Networking and Performance Modeling* (*WMuNeP'2007*), Chania, Greece, 2007.
18. Babaoglu Ö, Canright G, Deutsch A, Di Caro G, Ducatelle F, Gambardella LM, Ganguly N, Jelasity M, Montemanni R, Montresor A, Urnes T. Design patterns from biology for distributed computing. *ACM Transactions on Autonomous and Adaptive Systems* (*TAAS*) 2006; **1**(1):26–66.
19. Ding Y, Sun H, Hao K. A bio-inspired emergent system for intelligent web service composition and management. *Elsevier Knowledge-Based Systems* 2007; **20**(5):457–465.
20. Musunoori SB, Horn G. Ant-based approach to the quality aware application service partitioning in a grid environment. *Proceedings of IEEE Congress on Evolutionary Computation*, Vancouver, BC, Canada, 16–21 July 2006; 2604–2611.
21. Chiang F, Braun R, Agbinya JI. Self-configuration of network services with biologically inspired learning and adaptation. *Springer Journal of Network and Systems Management* 2007; **15**(1):87–116.
22. Carreras I, Chlamtac I, De Pellegrini F, Miorandi D. BIONETS: Bio-inspired networking for pervasive communication environments. *IEEE Transactions on Vehicular Technology* 2007; **56**(1):218–229.
23. Boonma P, Suzuki J. BiSNET: A biologically inspired middleware architecture for self-managing wireless sensor networks. *Elsevier Computer Networks* 2007; **51**(16):4599–4616.
24. Thanh DV, Jørstad I. A service-oriented architecture framework for mobile services. *Proceedings of the Advanced Industrial Conference on Telecommunications Workshop* (*AICT2005*), Lisbon, Portugal, July 2005; 65–70.
25. Sanchez-Nielsen E, Martin-Ruiz S, Rodriguez-Pedrianes J. An open and dynamical service oriented architecture for supporting mobile services. *Proceedings of the 6th International Conference on Web Engineering* (*ICWE '06*), ACM: New York, NY, U.S.A., 2006; 121–128.
26. Amoretti M, Bertolazzi R, Reggiani M, Zanichelli F, Conte G. Designing grid services for multimedia streaming in an e-learning environment. *Concurrency and Computation*: *Practice and Experience* 2006; **18**(8):911–923.
27. Verdi FL, Magalhães MF, Cardozo E, Madeira ERM, Welin A. A service oriented architecture-based approach for interdomain optical network services. *Springer Journal of Network and Systems Management* 2007; **15**(2):141–170.
28. Agrawal R, Bayardo RJ, Gruhl D, Papadimitriou S. Vinci: A service-oriented architecture for rapid development of web applications. *Elsevier Computer Networks* 2002; **39**(5):523–539.
29. Jørstad I, Dustdar S, Thanh DV. A service oriented architecture framework for collaborative services. *Proceedings of the 14th IEEE International Workshops on Enabling Technologies* (*WETICE 2005*), Linköping, Sweden, June 2005; 121–125.
30. Wu W, Fox G, Bulut H, Uyar A, Huang T. Service oriented architecture for VoIP conferencing. *International Journal of Communication Systems* 2006; **19**(4):445–461.
31. Chinnici R, Moreau JJ, Ryman A, Weerawarana S. Web services description language (WSDL) version 2.0 part 1: Core language. Online: http://www.w3.org/TR/wsdl20/ [December 2007].
32. Gudgin M, Hadley M, Mendelsohn N, Moreau J, Nielsen H, Karmakar A, Lafon Y. SOAP version 1.2 part 1. Online: http://www.w3.org/TR/2007/REC-soap12-part1-20070427/ [December 2007].

DOI: 10.1002/cpe

33. Di Caro G, Dorigo M. AntNet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research* (*JAIR*) 1998; **9**:317–365.
34. Xu D, Wichadakul D, Nahrstedt K. Resource-aware configuration of ubiquitous multimedia services. *Proceedings of the IEEE International Conference on Multimedia and Expo* (*ICME-2000*), New York, NY, U.S.A., vol. 2, July 30–August 2, 2000; 851–854.
35. Clement L, Hately A, Riegen CV, Rogers T. *UDDI Version 3.0.2*, *UDDI Spec Technical Committee Draft*. OASIS UDDI Spec TC, 2004.
36. Cohen F. *FastSOA*: *The Way to Use Native XML Technology to Achieve Service Oriented Architecture Governance*, *Scalability*, *and Performance* (1st edn). Morgan Kaufmann (Elsevier): San Fransisco, CA, November 2000.
37. Pushtotest testmaker 5.1. Online: http://docs.pushtotest.com/docs/index.html [April 2008].
38. Bertsekas D, Gallager R. *Data Networks* (2nd edn). Prentice-Hall: Englewood Cliffs, NJ, U.S.A., December 1991.
39. Hossain MS, El Saddik A. Scalability measurement for multimedia repurposing system. *International Journal of Advanced Media and Communication* 2008; **2**(3):267–287.

DOI: 10.1002/cpe