Multimedia Systems (2007) 12:505–520
DOI 10.1007/s00530-006-0071-8

REGULAR PAPER

# Metadata-driven multimedia transcoding for distance learning

**Mazen Almaoui · Azadeh Kushki · Konstantinos N. Plataniotis**

Published online: 9 February 2007


**Abstract** Advances in multimedia processing capabilities of electronic devices and the rapid growth of the Internet have contributed to the proliferation of collaborative applications such as distance learning (DL) webcasting. A key technical challenge in such DL systems is providing access to rich media content to any user regardless of device capabilities, network heterogeneity, and personal preferences. A novel MPEG-21-based adaptation architecture is presented to overcome these challenges by performing (1) application layer transcoding that adapts the presentation format of DL content to match device media decoding capabilities and user-desired modality, (2) bitstream transcoding to adapt multimedia to match device processing capabilities and encoding bit-rate supported by the network. Experimental results indicate that the proposed system delivers personalized DL content to meet end-user environmental restrictions with small transcoding overhead.

This work is partially supported by the Network for Effective Collaboration Technologies through Advanced Research (NECTAR). The authors would like to thank the Knowledge Media Design Institute (KMDI) at the University of Toronto for use of their resources..

M. Almaoui · A. Kushki (✉) · K.N. Plataniotis
Multimedia Laboratory, The Edward S. Rogers Sr. Department of Electrical and Computer Engineering, University of Toronto, 10 King's College Road, Toronto, M5S 3G4, Canada
e-mail: azadeh@comm.utoronto.ca

M. Almaoui
e-mail: mazen.almaoui.@amd.com

K.N. Plataniotis
e-mail: kostas@comm.utoronto.ca

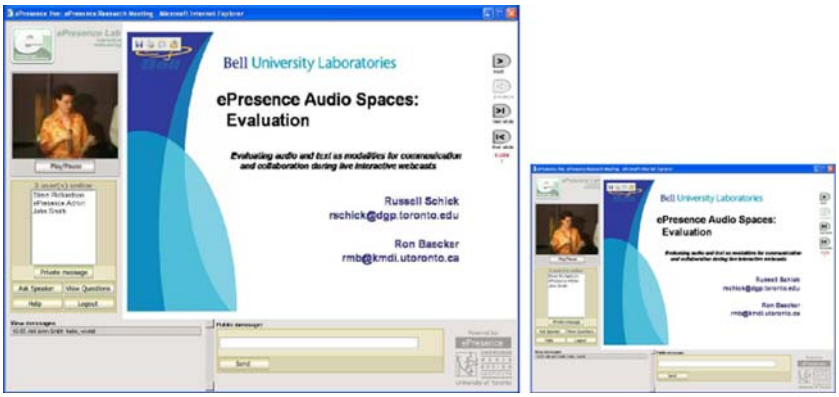
**Keywords** Metadata · MPEG-21 · MPEG-4 · Transcoding · Distance learning · Adaptive content delivery

## 1 Introduction

The maturity of the Internet has given rise to effective collaborative distance learning (DL) webcasting, allowing people from various physical locations to communicate and interact over wired and wireless links. In addition to the traditional video conferencing functionality, DL webcasting applications support user interactions and live streaming of rich media, such as video, audio, slides, and text, to a large number of users [10]. As an example, screen shots of an open source DL webcasting system called ePresence [11] are shown in Fig. 1.

The increasing multimedia processing capability of mobile devices such as Personal Digital Assistants (PDA), Pocket PCs, and Smart Phones as well as advances in wired and wireless networks have added a new dimension to how people and groups collaborate and access media content. Consequently, there is a growing aspiration for a Universal Multimedia Access (UMA) [12] framework that provides seamless delivery of multimedia content to anyone, at anytime, and anywhere. Such a UMA paradigm can be achieved through adaptation of content based on a variety of user needs and preferences. In particular, this work considers three factors that necessitate the need for adaptation. These are unique device capabilities, heterogenous networks, and user-defined Quality of Service (QoS).

The first factor is the diversity in device capabilities, in terms of parameters such as frame rate and screen resolution. This translates to a need for media adaptation




**(a)** PC **(b)** PDA

**Fig. 1** Screen shots of DL content used in the ePresence system. This example demonstrates the need for adaptive delivery

through scaling and transformation as well as termination of media streams in cases where they cannot be processed by the device. This must be performed in real time to avoid significant adaptation delays in delivering DL content to the end user. This real time nature of a DL application is crucial in providing users with an experience similar to that of attending the live event [19]. Figure 1 provides an example demonstrating the effect of display size on the viewing experience for the ePresence system. As seen, displaying complete multimedia content on devices with small screen sizes, renders the content virtually illegible.

The second factor to consider in a UMA framework is the delivery of multimedia over a multitude of heterogeneous and unreliable wired and wireless networks. Clearly, diverse networks exhibit unique behavior in terms of fluctuations in bandwidth capacity. Available bandwidth capacity is time dependant and calculated based on factors such as packet loss, round trip time delay, and jitter [41]. This diversity in bandwidth characteristics of the underlying network gives rise to a need for multimedia adaptation to provide variable bit-rate media encoding.

The third important consideration in UMA is the quality of the end-user experience [32]. Users must be allowed to negotiate a QoS based on their preferred multimedia modality. This must be negotiated, not granted, since not all devices are capable of processing all types of media. For example, video should not be delivered to a user device incapable of processing video.

Unfortunately, conventional DL applications do not support adaptive multimedia delivery. While various adaptive Web content delivery systems [26] have been proposed to address the UMA challenge, they are generally geared towards Web browsing applications and not live streaming of rich multimedia content as needed in DL webcasting [30].

Adaptation of multimedia in a live streaming context can be achieved through a process known as *transcoding*. Multimedia transcoding is the conversion of one media file format to another or the re-encoding of the original media file with new encoding parameters [9].

Traditionally, multimedia transcoding has been limited to *bitstream* transcoding, performed on the actual media to parse, transform, and scale the underlying streams. In addition to this type of transcoding, higher-level *application layer* transcoding [9] can be used to determine a presentation format (browser template) based on device capabilities and user-desired modality [38] and provide adaptation parameters to the underlying bitstream transcoder.

Both application layer and bitstream transcoding require descriptions of the usage environment and user preferences. In this regard, metadata is a powerful tool that can be utilized for describing, indexing, and searching for the properties of diverse devices, networks, user preferences as well as the actual media [38]. This work, therefore, considers a metadata-driven transcoding [38] approach for multimedia adaptation using both application layer and bitstream transcoding.

Metadata-driven transcoding is achieved in this paper through the use of standardized tools provided by MPEG-21 or the ISO/IEC 21000 Multimedia Framework. This framework standardizes interfaces and tools to facilitate seamless exchange and consumption of multimedia resources across a wide range of networks, devices, and users [33]. MPEG-21 packages the actual multimedia and metadata describing the properties of the encapsulated media into an architectural unit known as a Digital Item (DI). Transcoding is performed at the DI level and is known as Digital Item Adaptation (DIA) [6]. The first step in DIA is to perform description adaptation whereby the DI metadata and the user environmental metadata are transformed (based on adaptation rules) to produce parameters for application layer and

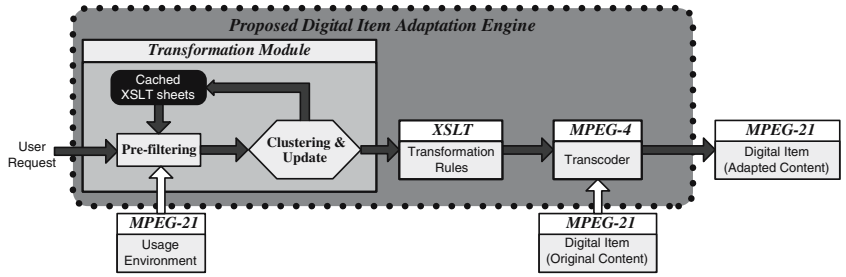



**Fig. 2** System overview: adaptation engine that performs application layer and bitstream transcoding

bitstream transcoding. Consequently, these parameters are used to produce a presentation template and to transform the media bitstreams to meet the end user's environmental needs.

The main contribution of this paper is the design of a novel MPEG-21-based system geared towards achieving UMA in a distance learning application where a combination of multimedia material such as video, audio, images, and text are delivered to the end users. The proposed design, shown in Fig. 2, uses the MPEG-21 adaptation tools to transform digital items based on unique and varying environmental conditions using meta-driven transcoding.

The rest of this paper is organized as follows. Section 2 will provide an overview of the prior work in the area of DL webcasting and metadata-driven transcoding. Sections 3 and 4 discuss the details of the proposed method and experiments and results are presented in Sect. 5. Section 6 concludes the paper and provides directions for future work.

## 2 Prior work

Over the past decade, a plethora of systems have been developed for adaptive content delivery to various devices over the World Wide Web [7,26,28,29,42]. The work of [29] proposes a QoS-aware adaptive Web-based architecture to allow for real time monitoring and modeling of network metrics, content and navigation preferences, as well as perceived performance related to user satisfaction of his/her Web experience. In [26,42], an adaptive delivery framework is presented that consists of three main modules namely, user environment monitoring, decision engine, and content adaptation. The decision engine parallels the application layer transcoder discussed above and is responsible for determining the necessary adaptations. The adaptation module is similar to the bitstream transcoder and operates on the actual media through abstraction (information compression), modality transformation, format conversion, data prioritization, and purpose classification.

The above Web-based systems provide content adaptivity geared towards browsing information on the World Wide Web. Consequently the problem of "on-the-fly" adaptation of live multimedia streams needed in DL is generally not considered [30]. At the same time, various systems [14,37,13,35,31] have been proposed to specifically address Web-based DL but they do not consider the problem of adaptive content delivery. Such systems generally include a combination of video, audio, slides, chat sessions, and whiteboard functionality. They utilize an Apache web server to interact with clients, and an SQL backbone to provide web-based access to media on the server database and a RealNetworks server to stream the media. A similar approach is given in [19] with the exception that an H.263 video compression engine is used to stream an additional video feed (at low bitrates) containing what is being demonstrated on a whiteboard.

Unfortunately, distance learning solutions do not generally provide adaptive content delivery. An exception is the four-tier Electronic Educational System (EES) model proposed in [16] that provides limited data personalization to the end user. Specifically, the top layer in the system, known as the instructional layer, allows educators (not users) to specify which media components to include in the system (video, slides, chatroom, whiteboard, etc.). The lower layers take the instructions from the top layer to create the final presentation format seen through a browser. A similar approach for delivering personalized course material to target specific user learning needs is proposed in [15]. In particular, IEEE LTSC Learning Object Metadata (LOM) is used to create a flexible architecture to allow professors to deliver personalized content. For example, the user can choose the language or difficulty level of the DL content. Although both of the above-mentioned models allow for personalized creation of the presentation content, they too fail to address user personalization in terms of device, network capabilities, and media preference.

As previously mentioned, multimedia adaptation for UMA can be achieved through transcoding. Section 1 motivated the use of metadata-driven transcoding to facilitate the description of user preferences and environmental conditions. In this context, the usage environment, including device capabilities, user preferences, and network conditions are described as metadata. Metadata syntax is commonly represented using Exten-

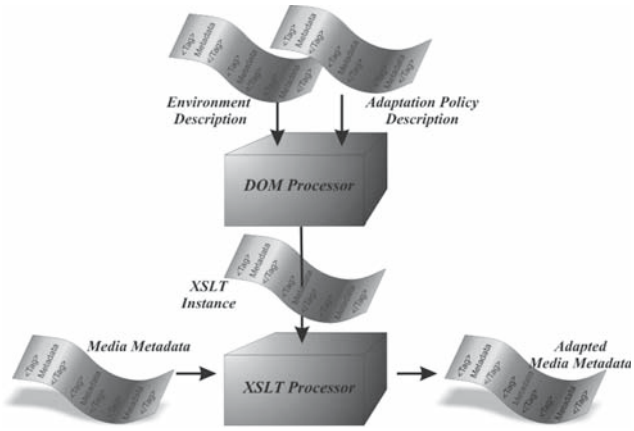



**Fig. 3** Metadata transformation engine

sible Markup Language (XML) [2] and its processing and transformations can be accomplished using Extensible Stylesheet Language Transformation (XSLT) [3]. XSLT is a tool for transforming and parsing an XML document into an alternative format such as HTML or text.

In the context of metadata-driven transcoding, XSLT sheets specify adaptation rules used by both application layer and bitstream transcoders. The application layer transcoder requires transformation rules to process user metadata in order to determine the modality to stream and the stylesheet required to properly display this modality in a browser. The bitstream transcoder requires transformation parameters to process user metadata to physically adapt the media to meet the device limitations and network bit-rate requirement. Bitstream-level multimedia transcoding has been well studied and the interested reader is referred to [22,36, 39] for an overview and examples. The rest of this section provides a brief outline of existing solutions for metadata-driven transcoding.

A generic metadata transformation system is proposed in [23] and similarly in [24] as shown in Fig. 3. These solutions consist of a Document Object Model (DOM) processor that creates XSLT sheets and an XSLT processor that transforms the XML data using the created XSLT sheet. The DOM processor takes in environment and adaptation policy descriptions as its input and produces an XSLT sheet that meets the current adaptation specifications as output. While these implementations are intended to provide generic content adaptation solutions, implementing the DOM processor is application specific and hence requires knowledge of semantics of all applications that will have their contents transformed by this metadata transcoding engine. Moreover, DOM processing requires intensive and computationally expensive conditional programming.

Because a DL application requires live streaming of media content, the computational complexity of DOM processing poses a serious limitation. The proposed method performs DOM processing a priori and caches XSLT sheets to transcode for unique environmental processing conditions. In essence, the proposed method replaces the DOM processor with a selection phase that is more efficient in the context of real time transcoding. Hence, by having a fully cached XSLT solution, metadata-driven transcoding can be accomplished in real time.

## 3 Proposed system

This work proposes a system geared toward the delivery of live and archived DL content. As shown in Fig. 2, this system achieves the UMA objective through the use of MPEG-21-based metadata-driven transcoding for multimedia adaptation. As described in Sect. 3.1, MPEG-21 provides a standard description of environmental and user conditions. Moreover, it allows the packaging of media (audio, video, and slides) and the corresponding metadata into a single entity called a Digital Item (DI). As seen in Fig. 2, the first step in DI adaptation is to transform the metadata pertaining to a DI using an XSLT sheet. Due to the high computational cost of DOM processing needed for creation of such sheets in real time, we propose to create and cache a small set of XSLT sheets a priori to serve various combinations of device capabilities, network conditions, and user preferences. A novel transformation module, described in Sect. 4 is proposed to select one of the cached sheets in a "best effort" manner to match the user's environment description (MPEG-21 usage environment). After an appropriate sheet is chosen, the transformation rules and the MPEG-21 DI associated with the DL content are forwarded to the transcoder. As discussed in Sect. 3.2, the transcoder uses the XSLT rules to produce a new DI. This adapted DI represents the presentation template and DL multimedia that best match the user's environmental description needs. The final step is to stream the content to the end user.

Adaptation can be performed at the content server, a proxy server, or the user terminal [26,27]. In this work, the DIA engine is implemented on the content server in order to create transparent access to devices having minimal processing capabilities.

### 3.1 MPEG-21: usage environment

The MPEG-21 Standard provides various tools to support metadata-driven transcoding [6]. The Usage Environment Description Tool (UEDT) [5] generates

metadata that describes device capabilities, network conditions, and user characteristics as well as the natural environmental characteristics such as user visual impairments. The UEDT provides the proposed system with a standardized description tool that contains all the information needed about the user environment for the purpose of adaptation. These standardized description tools allow for the use of metadata by any system components, external components, or vendors interfacing with the adaptation engine or requiring information about the user's environment. Moreover, MPEG-21 proposes a larger set of parameters [5] for applications that require alternative information about the user's environment. It should be noted that the MPEG-21 Standard does not specify how to obtain information about the environment. This work assumes that the user creates an account on the DL system indicating the usage environment parameters and preferences. There are automated methods of obtaining user environment information such as the Session Description Protocol [41] to obtain device capabilities and Real Time Control Protocol [41] to obtain network information. These protocols are outside the scope of this paper and will not be discussed in detail.

### 3.2 Transcoder

The first step of transcoding is to initiate application layer objectives such as producing the proper template to display the desired modality in a browser. Components of the presentation template such as video, slides, and text chat are described in the DI metadata. This metadata is transformed based on the adaptation requirements into a browser friendly HTML format using the XSLT sheet chosen by the transformation module.

After this presentation template has been determined, bitstream adaptation takes place. A commonly used approach for bitstream adaptation is offline scalable coding whereby scaled versions of the same media content are created, stored, and streamed as needed. Such a method, however, is highly storage intensive. Therefore, in cases such as DL content delivery where no limitations on types of devices, user preferences, and network conditions are imposed, realtime adaptation, known as transcoding, is needed [40]. Moreover, the intended application of DL requires streaming of live lecture material as the media are captured. This renders scalable coding unsuitable in this work.

Transcoding is performed through the MPEG-4 Standard scalable tools [25] in the proposed system to provide temporal, spatial, and Signal-to-Noise Ratio (SNR) scalability (i.e., vary the bit-rate encoding). Recall that the set of rules specified in an XSLT sheet together with the user's UEDT are used to transform the DI metadata. Adaptation parameters obtained from this transformation are utilized to perform transcoding on the media streams by providing the proper configuration files for the MPEG-4 encoder. Examples of such parameters include temporal and spatial scaling of video and images as well as audio scaling parameters.

The above transcoding procedure must be performed for XSLT sheets chosen to serve user requests. As multiple user requests can be mapped to the same XSLT sheet, the number of necessary of encodings is determined by the number of such sheets (not users). This highlights the trade-off between the level of adaptation offered by the system and computational complexity. Lastly, it must be noted that there are numerous MPEG-4 encoders that provide real-time transcoding with low processing delay, even for live multimedia streaming. Moreover, [9] proposes modifications to this encoder to further reduce the needed computations.

In addition to parameters related to spatial scalability, the bitstream transcoder requires the knowledge of available bandwidth in order to encode multimedia at a bit-rate that meets the end-user network conditions. The next section will explain how the proposed method estimates the available bandwidth in order to encode and stream DL content at a high level of QoS while dealing with the end users unreliable IP network conditions such as packet loss due to congestion.

### 3.3 Bandwidth estimation

A bandwidth estimation protocol is a critical component for ensuring a high-level QoS multimedia delivery over best effort networks such as the Internet. This is needed to avoid device buffer overflow and multimedia degradation as a result of significant packet loss in congested networks. In this light, the proposed system encodes and streams media according to the available bandwidth of the network connecting the end user to the content server.

Bandwidth estimation is the determination of available bandwidth for data transmission based on dynamic factors such as packet loss and delay in order to cope with congestion points in the underlying network. Existing TCP rate control algorithms are suitable for applications such as HTTP and FTP that can withstand bursty data transmission (i.e., can withstand abrupt bandwidth fluctuations). However, bursty transmission is not appropriate for a DL application due to the high sensitivity of the human observer to visual data bit-rate fluctuations [18], especially in video. Thus, smooth bandwidth rate estimation dynamics are desired in

**Table 1** User environmental parameters used in the proposed system for content adaptation

| Description tool | Parameters |
|---|---|
| Device capabilities | Codec supported, resolution, frame-rate, color depth, frequency range, number of audio channels |
| Network characteristics | Available bandwidth (i.e., bit-rate) |
| User preference | Modality (video, audio, images, text and audio chat) |

multimedia streaming. At the same time, since the occasional loss of packets can be tolerated during video and audio streaming [17], an aggressive rate control algorithm, such as TCP, is not required for our application.

The proposed method utilizes a TCP Friendly Rate Control (TFRC) variant of the TCP congestion control algorithm called Equation-Based Congestion Control (EBCC) [20]. Equation (1) [20] provides one example of calculation of a TFRC response function used to estimate and control the bandwidth rate. This formulation can be used to provide an upper bound on the transmission rate for a given packet size $s$, round-trip-time $R$, calculated packet loss $p$, and the retransmission timeout $t_{\mathrm{RTO}}$.

$$\mathrm{BW}_{\mathrm{available}} = \frac{s}{R\sqrt{\frac{2p}{3}} + t_{\mathrm{RTO}}(3\sqrt{\frac{2p}{3}})p(1+32p^2)} \tag{1}$$

The details of EBCC are outside the scope of this paper, and the reader is referred to [20] for complete details.

## 4 Transformation module

In contrast to previous metadata-driven transcoding approaches [23,24], the transformation module does not create XSLT sheets to serve an incoming user-request. Instead, its objective is to choose a sheet which matches the user-request environment needs from a set of previously cached sheets to provide best effort service in real time. This sheet will then be used by the transcoder to adapt the DL content to the end user environment characteristics. This method avoids the high complexity and delay caused by creation of new sheets on the fly (i.e., DOM processing) as mentioned in Sect. 2. The transformation module also updates the cached sheets to better adapt to changing user devices and environmental conditions by employing a clustering algorithm.

Selection of the required sheet can be done based on various network conditions, device parameters, and user preferences. When choosing such selection criteria, it is important to note that increasing the number of parameters usually adds more complexity to the transcoding process. With this in mind, the proposed system uses the parameters listed in Table 1 to select a sheet most suited to network, device, and user requirements.

Cached sheets are initialized once during an offline step based on expected usage environment conditions. While the initial set of sheets must be chosen to match the anticipated usage environemnt sufficiently well, experiments of Sect. 5 show that the complexity of the system and startup delay increase with the number of cached sheets. For the purposes of this work, the cache is initialized using typical resolution, frame-rate and color depth parameters for three types of devices, namely, personal computers (PC), personal digital assistants (PDA), and smart phones as well as available bitrates for streaming in dial-up, wireless, and high speed networks. Other devices such as High Definition Televisions can readily be taken into account in future developments. Further details on the parameters used are provided in Sect. 5.

After initialization of the cache and upon receiving a new user request, the online XSLT sheet selection process is carried out in three steps. First, the cached sheets are pre-filtered based on device limitations to determine a small set of candidate sheets. Next, one of the sheets from the candidate set is chosen for transcoding and the cache is updated. Finally, the transformation rules are passed to a transcoder to produce the final adapted media for this particular request and stream it to the user as described in [9]. The details of each of these modules are discussed in the following sections.

### 4.1 Pre-filtering

The first step in the proposed approach is to search the cached XSLT sheets and select sheets that satisfy the minimum parameter requirements needed for proper processing of the requested media by the user device. For example, if the device is capable of processing video at 15 frames per second (fps), all sheets with a rate greater than 15 fps are eliminated from the selection process. Parameters used in pre-filtering are those that remain *unchanged* during the webcasting session (assuming the same device). The pre-filtering of XSLT sheets is accomplished by partitioning the set of existing sheets using a tree structure [34] based on parameters such as device capabilities, modality, and available network bandwidth. The advantage of using a tree structure is that XSLT sheets can be organized in a manner that allows efficient insertion, searching, and elimination of

XSLT sheets in the cache. Pre-filtering gives precedence to device and network capabilities over user preference. For example, if a device is not capable of processing video, all XSLT sheets with video are disqualified even if the user indicates high preference for this modality.

The pre-filtering component returns multiple XSLT sheets with the capability to transcode the requested media. Hence, a number of different sheets are available to be used by the transcoding engine for DIA which meet fixed session requirements such as the presentation format of the DL content and the multimedia modality that the device is capable of processing. Having multiple sheets allows the transcoder to switch between adaptation rules to account for dynamically fluctuating parameters such as available bandwidth. The next section describes how a unique XSLT sheet is chosen from the set of pre-filtered sheets and how the cached sheets are updated.

### 4.2 Clustering and cache updates

The pre-filtering component discussed above selects a set of potential sheets to serve an incoming user request. The next step is to choose one XSLT sheet from this set based on the MPEG-21 UEDT for transcoding. As the number of candidate cached sheets are limited, the user request can only be served in a best effort manner. In other words, matching of the user request to one of the cached sheets incurs some loss. In this light, we propose a sheet update procedure that allows the system to minimize this loss over multiple requests, by adapting to changing user needs over time. The cache update is carried out in an automatic manner in parallel with sheet selection and streaming and thus does not lead to additional delay.

In the following discussion, each sheet is treated as a vector of transcoding parameters (all have numerical values) found in Table 1 denoted by $\mathbf{S}_j = [s_1, \ldots, s_m]^{\mathrm{T}}$. In other words, the transformation rules of sheet $j$ adapt the DL content to the environmental parameters represented by vector $\mathbf{S}_j$. Similarly, the incoming user request $i$, $\mathbf{Q}_i = [q_1, \ldots, q_m]^{\mathrm{T}}$, is a vector with the same dimensionality as the sheet vectors. Each incoming request vector is then compared against the set of pre-filtered sheets and the sheet most "similar" to the request is chosen for transcoding. The similarity comparison algorithm is discussed next.

#### *4.2.1 Sheet selection for transcoding*

The objective of the sheet selection algorithm is to map an incoming user request to an XSLT sheet in the cache containing the adaptation rules for transcoding. This is done by clustering the incoming requests based on the sheet they map to. The number of clusters is equal to the number of cached sheet (20 in our case) and these same sheets will serve as the initial cluster centers. The set of requests served by a given sheet $\mathbf{S}_j$ generates a request cluster $c_j$ which will be used for cache selection and update. It is appropriate that each sheet has its own cluster so that each sheet is individually updated according to the requests mapped to it. We denote the set of all clusters (corresponding to each cached sheet) after $n$ requests have been received by the DL system as $\mathcal{C}_N^{(n)} = \{c_1^{(n)}, \ldots, c_N^{(n)}\}$, where $N$ is the total number of cached sheets and $c_i^{(n)}$ denotes the cluster request for sheet $i$ after $n$ requests are received.

After the pre-filtering stage where $M$ sheets are disqualified, $N - M$ sheets are available that match the end-user environmental restrictions. Denote this set as $\mathcal{S}' = \{\mathbf{S}_1, \ldots, \mathbf{S}_{N-M}\}$. The clustering algorithm selects one sheet, $\mathbf{S}_i \in \mathcal{S}'$ for transcoding according to the current environmental conditions. Clearly, finding a cached sheet that *exactly* matches all the environmental criterions of a user request is unlikely. Thus the system must map each user request to a cluster (XSLT adaptation rule) that produces the lowest error. Consequently, a distance measure is needed in order to map an incoming data sample (i.e., user request) to a cluster (i.e., XSLT sheet). In choosing such a measure for clustering, the unique nature of the parameters in Table 1 (in terms of properties such as dynamic range and probability distribution) must be taken into account. This is a difficult task to accomplish because parameters such as frame-rate change continuously due to rapidly advances in technology. In light of these difficulties, the proposed clustering algorithm uses a normalized Weighted Absolute Error (WAE) [8] calculated as follows:

$$d(\mathbf{Q}_i, \mathbf{S}_j) = \sum_{k=1}^{m} w_k \frac{|q_k - s_k|}{\alpha_k} \tag{2}$$

where $\alpha_k$ is used to normalize the parameter distances to the range $[0, 1]$, and $w_k \in [0, \beta]$ indicates the importance of parameter $k$ determined a priori by the user with 0 representing the lowest priority. The WAE offers the following features:

- The absolute error can be calculated efficiently. This is important for real time DL systems, specially when the number of clusters is large.
- All parameters in Table 1 have numerical values that should be normalized because each parameter has a unique dynamic range. Normalization is essential during the aggregation of distances to ensure that

parameters with large dynamic ranges do not mask the effect of parameters with small dynamic ranges.
- The WAE utilizes the user preferences to weigh parameters according to user-perceived importance. For example, if the user preference states that the frame-rate is more important than display resolution, then more emphasis will be placed on finding a closely matching frame-rate request, hence providing the user with a degree of QoS personalization. The number of weights chosen and their values will vary based on adaptation system design.

Sheet selection is accomplished by minimizing the selection loss over all clusters by mapping a request to a cluster that produces the lowest WAE. This loss is denoted as $e$ and is calculated as follows:

$$e = \sum_{\mathbf{S}_j \in \mathcal{S}'} \sum_{\mathbf{Q}_i \in c_j^{(n)}} d(\mathbf{Q}_i, \mathbf{S}_j) \tag{3}$$

where $d(\mathbf{Q}_i, \mathbf{S}_j)$ is the WAE of (2) and $\mathcal{S}'$ and $c_j^{(n)}$ represent the set of sheets after pre-filtering and cluster of requests for sheet $j$ defined previously.

This process is repeated throughout the transcoding session. In other words, a user request is mapped to a particular sheet according to the following condition:

$$\mathbf{Q}_i \in c_j^{(n)} \quad \text{iff} \quad d(\mathbf{Q}_i, \mathbf{S}_j) = \min_{\mathbf{S}_j \in \mathcal{S}'} d(\mathbf{Q}_i, \mathbf{S}_j). \tag{4}$$

A summary of the sheet selection algorithm is shown in Fig.4.

Maintaining the UEDT metadata of every request poses two burdens on the content server. The first concern is that each metadata description can be 50 kB in size [6]; assuming a large number of user requests, significant storage space may be required to store these descriptions. The second is the need for an efficient and organized method of caching and accessing the information from these files. In order to reduce the storage and indexing requirements on the server, the actual clusters, which include the request metadata, are not stored. Instead, only cluster centers $\mathbf{S}_i,\ i = 1, \ldots, N$ are stored. The next section will explain how the user UEDT is used to update the clusters without the need to store the metadata files on the content server.

### 4.2.2 Updating the cached sheets

The proposed system offers a real-time adaptation framework by using a set of cached XSLT sheets as opposed to generating sheets on the fly. The limited

**Inputs:**
$\mathbf{S}_i$: $i^{th}$ cached sheet;
$\mathbf{Q}_i$: $i^{th}$ user request.
**Output:**
$Selection$: XSLT sheet index to be used in transcoding.
**Algorithm:**

```
for each new request
  Pre-filter based on UEDT parameters to obtain S';
  Select sheet j:
    d_min = ∞
    for j = 1 : N − M
      if d(Q_i, S_j) ≤ d_min
        d_min = d(Q_i, S_j)
        Selection = j
    end
end
```

**Fig. 4** Sheet selection algorithm

number of cached sheets are generated based on a priori expectation of user needs by the system designer. Such a scheme results in an adaptation service that inevitably leads to some compromise in matching user requirements. With this in mind, we propose to update the set of cached sheets during the operation of the system to adapt to user requests over time and minimize the overall loss incurred due to limitations of a priori parameter selection during the initialization of the cache, and to cope with technological advances.

Two methods can be used to update the cached sheets. The first method is manually creating new sheets to take into account new devices and network conditions. For example, if new devices such as High Definition Television (HDTV) or Large Displays [21] require access to the DL content, new rules and hence XSLT sheets will be required. The second method explained in this section updates the cached sheets based on environment parameters from incoming requests of devices recognized by our system.

A cluster structure similar to the one presented for sheet selection is used to update the XSLT sheets' parameters based on the characteristics of users accessing the system. This section will give details of how the multimedia bit-rate encoding parameter will be updated based on the bandwidth characteristics of the heterogenous network connecting the end user to the content server. Updating can also be applied to other parameters such as frame-rate and resolution.

Note that the sheet selection clustering algorithm operates only on pre-filtered sheets. This is done because a given request can be served only with XSLT rules that produce a presentation template and adapted

multimedia that the user's device is capable of processing. The pre-filtering operation ensures that this scenario of over-estimating capabilities does not occur. Thus, the error $e'$ taken over *all* sheets is generally smaller than the error taken over the pre-filtered set. This is because a request with slightly higher quality parameters than sheet $\mathbf{S}_j$ will be assigned to another sheet with possibly larger loss. In order to remedy this situation, the cache updating strategy aims to minimize the loss over all cached sheets. Hence a sheet updating algorithm is introduced by considering the modified loss function:

$$e' = \sum_{\mathbf{S}_j \in \mathcal{S}} \sum_{\mathbf{Q}_i \in c_j^{(n)}} d(\mathbf{Q}_i, \mathbf{S}_j) \tag{5}$$

where $\mathcal{S}$ is the set of all cached sheets. The loss function of (5) is essentially the sum of within cluster distances. Using this loss function and the entire set of cached sheets, the algorithm in the previous section is used to assign an incoming request to a particular cluster. In the case of updating the bit-rate encoding, the cluster representative, or sheet bit-rate is the new sample mean evaluated using all elements belonging to the cluster and is updated incrementally using the new request:

$$R_{\mathbf{S}_j}^{(n)} = \frac{n-1}{n} R_{\mathbf{S}_j}^{(n-1)} + \frac{1}{n} R_{\mathbf{Q}_i} \tag{6}$$

where $R_{S_j}^{(n)}$ denotes the bitrate at time $n$ for sheet $\mathbf{S}_j$ and the request $\mathbf{Q}_i$ is assigned to $\mathbf{S}_j$. Due to the incremental update policy utilized here, previous requests do not need to be stored on the system. The cache updating algorithm is outlined in Fig. 5 for the example of bit-rate. Note that not all the sheets are updated. The system reserves one sheet for every type of device supported that adapts DL content assuming minimal processing capabilities. This guarantees that users with outdated technology can still access the system. The experimentation reported in this work will highlight the converging properties of the clustering algorithm and will demonstrate the bandwidth utilization gains by the proposed design.

**Definitions:**
$\mathbf{S}_j^{(n)}$: mean bit-rate of cluster $j$ after $n$ requests.
$R_{\mathbf{Q}_i}$: $i^{th}$ user requested bit-rate.

**Algorithm:**
while new request exists
Assign request to sheet $j$ to minimize loss function $e'$:
$\mathbf{Q}_i \in c'_j$ iff $|\mathbf{Q}_i - \mathbf{S}_j^{(n)}| = \min_{\mathbf{S}_j \in \mathcal{S}} |\mathbf{Q}_i - \mathbf{S}_j^{(n)}|$
Updatecachedsheetbit-rateparameter:
$R_{\mathbf{S}_j}^{(n)} = \frac{n-1}{n} R_{\mathbf{S}_j}^{(n-1)} + \frac{1}{n} R_{\mathbf{Q}_i}$,
end

**Fig. 5** Sheet update algorithm for adaptively changing the bit-rate encoding of sheets

## 5 Experiments and results

This section provides details on experimental evaluation of the proposed methods using Monte Carlo simulations as well as real streaming scenarios. In particular, four points are addressed in this section. First, the effectiveness of application layer transcoding in producing suitable presentation templates for given device specifications is demonstrated. Second, the validity of the clustering algorithm in selecting adaptation rules as well as its convergence properties are investigated. Third, we examine the impact of encoding rate on incurred delay as this contributes directly to the startup delay associated with buffering. Fourth, the effect of various bandwidth estimation and congestion control mechanisms on number of dropped frames is studied as high packet loss ratios can lead to significant errors during video decoding.

### 5.1 Experiment setup

Experiments of this section are based on real streaming results as well as simulated scenarios. The setup for each of these scenarios is described below.

#### *5.1.1 Environment*

Experimental results are reported for a setup similar to the ePresence system as the proposed solution will function on top of such a system to provide adaptive transcoding and streaming. In the real streaming scenario, the DL content server is set up in Toronto, Canada and the content is viewed approximately 600 Km away in Montreal, Canada.

In order to test the proposed solutions with a large number of users and large range of parameters, Monte Carlo-based experimental results on a simulated setup are reported. An IP network emulator ns-2 is used to simulate a fluctuating heterogenous network for realistic interactive webcasting scenarios with the client and server connected by nodes (gateway and routers) and with duplex connections between all nodes. Figure 6 shows the IP network configuration used with the Real Time Streaming Protocol (RTSP) [41] and Real Time Control Protocol (RTCP) [41] plus TCP server streams.

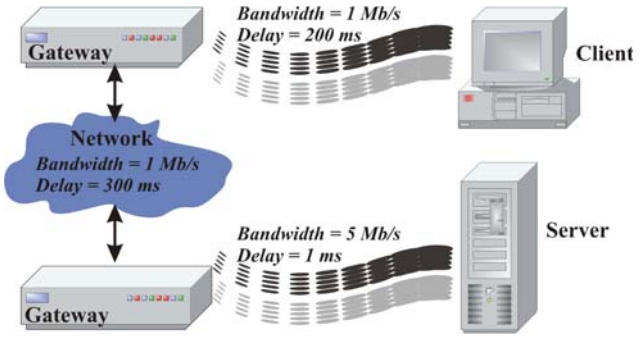



**Fig. 6** Client/Server network configuration for our experiments assuming a 600 km streaming distance. *Dark*-and *light-colored* waves represent RTSP and RTCP plus TCP server streams, respectively

The RTSP connection between the server and the client is for the actual multimedia transportation and the RTCP plus the TCP connections are used for bandwidth estimation and for exchanging other client/server control information. The client is connected to the Internet through a wireless link with a 1 Mb/s streaming capacity. The server is located on a Local Area Network at a distance of 600 Km from this client and has a bitrate of 5 Mb/s for streaming. This server is equipped with a 665MHz Pentium III processor with 128 MB of RAM.

#### *5.1.2 Media, encoding and streaming*

The transcoded event stream is comprised of video, audio, and slides in JPEG format. The proposed architecture uses MPEG-4 as a means to encode video and audio. An FFMPEG[1] encoder is used as the physical transcoder capable of providing temporal, spatial, and SNR scalability. The MPEG-4 and 3GP (MPEG-4 format for mobile devices) content are streamed to the end users by the open source Darwin Streaming Server [1].

#### *5.1.3 Devices*

Two types of devices, PCs and PDAs, are considered for the experiments of this section. The parameters used herein are reflective of typical device capabilities at the time this paper is written and are shown in Table 2.

### 5.2 Application layer transcoding

Application layer transcoding is used to adapt the DL material based on user modality preferences and to deliver the requested multimedia layout. This step is crucial in determining the device capabilities in processing the requested modality through XSLT transformation rules.

Figure 7 shows screen captures of adapted DL content displayed on both a PC and a PDA. Because the PC is capable of displaying the full lecture content (video, slides, and text chat), application layer transcoding produces a browser template that includes all three modalities. Due to screen limitations of the PDA, the presentation template is adapted such that only a subset of modalities is streamed based on the user preference. Moreover, examples of an adapted digital item as well as a browser template are shown in Figs. 8 and 9, respectively. These figures illustrate how device capabilities are considered in generation of the adapted layout.

### 5.3 Clustering algorithm

This subsection will investigate three aspects of the clustering algorithm: (1) sheet selection process via an example based on actual devices, (2) delay incurred due to sheet selection and metadata transformations, (3) capability to update cached XSLT sheets.

#### *5.3.1 Pre-filtering and sheet selection*

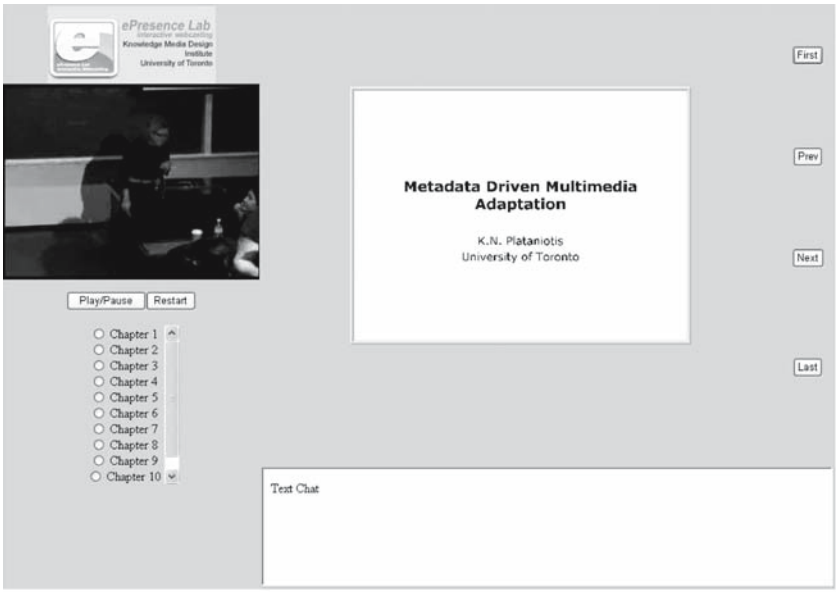
The effectiveness of the sheet selection algorithm is elucidated through an example involving three cached sheets and two requests based on the devices in Table 2. For simplicity, it is assumed that all parameters shown in Table 1, except video resolution, frame-rate, and color depth, are matched perfectly between the selected sheet and user requests. We further assume that these parameters are equally important to the user.

Table 3 shows parameters of the cached sheets and the corresponding WAE for the two requests. It is seen that the request coming from the PC cannot be perfectly matched by any of the sheets. Sheet $S_1$ produces the smallest WAE for this request and is selected for transcoding despite the fact that the adapted resolution is lower than that requested. For the PDA, sheet $S_2$ perfectly matches the requirements of the incoming request from the PDA. As expected this sheet produces the minimum WAE of 0 and is selected for transcoding. Note also that sheet $S_1$ is pre-filtered for this request as the resolution exceeds display capabilities of the device. This example shows how best effort personalization is provided to the end user with the guarantee of matching environmental restrictions to deliver DL content that can be properly processed.

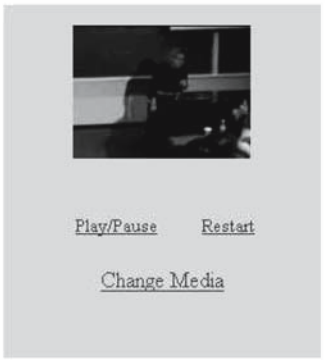
[1] http://sourceforge.net/projects/ffmpeg/

**Table 2** Specifications for devices used in experiments

| Device | Processor | Memory | Video resolution | Frame-rate | Color depth |
|---|---|---|---|---|---|
| PC | Intel Pentium 4 (2.8 GHz) | 512 MB | 360 × 240 | 20 | 24 |
| HP hx2400 (PDA) | Intel PXA270 (520 MHz) | 64 MB | 176 × 120 | 10 | 16 |




**(a)** DL content displayed on a PC screen.




**(b)** PDA display: video only.


**(c)** PDA display: slides only.

**Fig. 7** (a) DL content displayed on a PC screen. (b). PDA display: video only. (c). PDA display: slides only. Example of application layer transcoding: ePresence lecture displayed on **a** PC screen and **b**, **c** PDA

```
<DIA>
 <Description xsi:type="UsageEnvironmentType">
  <UsageEnvironment xsi:type="UserCharacteristicsType">
   <UserCharacteristics xsi:type="UserInfoType">
    <UserInfo xsi:type="mpeg7:PersonType"> ... </UserInfo></UserCharacteristics>
   <UserCharacteristics xsi:type="PresentationPreferenceType">
    <PresentationPriority>
     <GeneralResourcePriorities>
      <ModalityPriorities>
      <Modality href="urn:mpeg:mpeg21:2003:01-DIA-ModalityCS-NS:1 " priorityLevel="3">
      <mpeg7:Name>Video</mpeg7:Name> </Modality>
      <Modality href="urn:mpeg:mpeg21:2003:01-DIA-ModalityCS-NS:1 " priorityLevel="2">
      <mpeg7:Name>Audio</mpeg7:Name> </Modality>
     </ModalityPriorities> </GeneralResourcePriorities>
   </PresentationPriority> </UserCharacteristics> </UsageEnvironment>
  <UsageEnvironment xsi:type="TerminalCapabilitiesType">
   <TerminalCapabilities xsi:type="InputOutputCapabilitiesType">
     <Display bitsPerPixel="24" refreshRate="15">
     <Resolution horisontal="32 0" vertical="240"/> </Display>
     <AudioOut highFrequency="5" lowFrequency="200".../>
  </TerminalCapabilities></UsageEnvironment>
  <UsageEnvironment xsi:type="NetworkCharacteristicsType">
   <NetworkCharacteristics maxCapacity="100" minGuaranteed="100".../>
  </UsageEnvironment> </Description>
</DIA>
```

**Fig. 8** Example of DI adapted to match the PC specifications

**Fig. 9** Example of browser template generation

```
<xsl:stylesheet version="1.0"><xsl:template match="/">
 <html><head><title>
  <xsl:value-of select="//didl:DESCRIPTOR[@ID='LECTURE_TITLE']/didl:STATEMENT"/>
 </title><script type="text/javascript">...</script></head>
 <body>...
  <!-- video -->
  <object classid="clsid:..." height="240" id="movie" width="320">
  ...</object>...
  <!-- slides -->
  <img id="slide" src="1.bmp" width="420">
  <xsl:attribute name="src"><xsl:for-each select="//didl:ANCHOR">
  <xsl:if test="didl:DESCRIPTOR[@ID='SLIDE_NUMBER']/didl:STATEMENT = 1">
  <xsl:value-of
   select="didl:DESCRIPTOR[@ID='SLIDE_LOCATION']/didl:COMPONENT/didl:RESOURCE/@REF"/
  </xsl:if></xsl:for-each></xsl:attribute></img>...
 </body></html>
</xsl:template></xsl:stylesheet>
```

**Table 3** Cached sheets and the corresponding WAE for requests based on devices of Table 2 ($\mathbf{Q}_1$: PC, $\mathbf{Q}_2$:PDA)

| Sheet | Video resolution | Frame-rate | Color depth | $d(\mathbf{S}_i, \mathbf{Q}_1)$ | $d(\mathbf{S}_i, \mathbf{Q}_2)$ |
|---|---|---|---|---|---|
| $S_1$ (generic PC) | 320×240 | 15 | 16 | **0.69** | Pre-filtered |
| $S_2$ (generic PDA) | 176×120 | 10 | 16 | 1.59 | **0** |
| $S_3$ (generic Smart Phone) | 128 × 96 | 10 | 12 | 1.86 | 0.67 |

### *5.3.2 Selection delay*

One of the appealing aspects of the proposed metadata-driven transcoding solution lies in its efficiency in determining adaptation rules to meet the end users' environmental needs and performing metadata transformations. This is accomplished by having a fully cached XSLT solution that does not require DOM processing. Figure 10a shows the delay incurred by this algorithm due to sheet selection and metadata transformation only. Our proposed method currently supports 20 cached sheets and utilizes 8 parameters (Table 1). The results were generated for various number of parameters by applying the sheet selection algorithm based on only a subset of the parameters (for the five-parameter case) and by adding four "dummy" parameters to the sheets and user requests (in order to analyze the algorithm for 12 parameters). Since the objective of this experiment was to quantify the delay incurred by the system, the actual parameter values were irrelevant.

As Fig. 10a shows, the proposed algorithm experiences a negligible delay of approximately 1.5 s. This plot also displays the trade-offs between delay and caching more XSLT rules and the repercussions of taking more environmental parameters into account. By caching more sheets, the DL system can provide more detailed adaptation at the cost of having a higher delay.

### *5.3.3 Convergence of updating algorithm*

As mentioned in Sect. 4.2, device and network parameters in the XSLT sheets are updated according to incoming requests to improve bandwidth utilization and to comply to changing demands in device parameters. By updating bitrate values, the system is able to provide variable encoding to match current network conditions. This should be contrasted against a fixed rate encoding scheme that uses worst-case bitrate conditions. With this in mind, Fig. 10b shows the improvement in bandwidth utilization as a function of the number of requests used to update the clusters. The results are obtained using the simulated setup described in Sect. 5.1.1. The requests were randomly generated by varying user-server distance, type of network (wired or wireless), and download bandwidth using the ns-2 simulator.

The results indicate that after approximately 1,000 requests, the clustering algorithm converges at a local minimum of 10% unutilized bandwidth. This is expected as 1,000 requests are sufficient to expose the system to a wide array of network conditions.

### *5.3.4 Bitstream transcoding*

Real-time applications, such as DL systems, that deliver video and audio to the end user must stream content with small delay in order to produce a live lecture experience. The combination of RTSP/RTCP control mechanisms allow for the delivery of a continuous video stream with low overhead. This is accomplished by pre-buffering a sufficient portion of the video at the user device to maintain a steady flow of media to playback while the rest of the video is streamed. Although RTSP can ensure a steady flow of media to the user device buffer, the effect of bitstream transcoding delay must be addressed. As previously mentioned, the existing works in the area

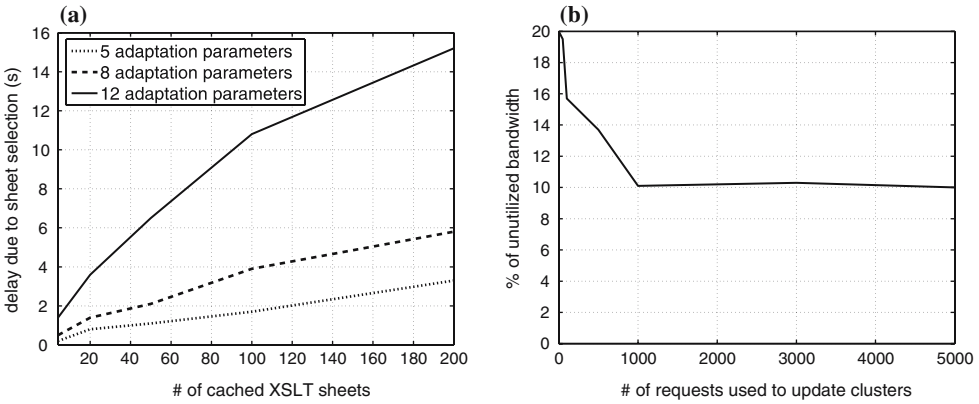



**Fig. 10** Start-up delays for the proposed system

Pre-filtering & sheet selection delay.

Improvements in band width utilization after updating sheets

utilize DOM processing that is computationally intensive and is proposed as an offline solution. For this reason the direct comparison of our online technique to that method is untenable. In this light, the delay incurred by the system with no transcoding is used as the benchmark in the following results.

Figure11 shows the start-up delays for the real streaming scenario described in Sect. 5.1.1. In particular, the delay value for streaming of live transcoded media, archived transcoded media, and media without any transcoding are shown for both a PC and a PDA. For live DL sessions, the encoder requires the aid of a capturing device to transfer and synchronize live video and audio from devices (e.g., video camera). Moreover, transcoding causes additional delay as video and audio must be converted from one format to another (e.g., raw video to MPEG-4 format) or scaled (e.g., change frame-rate). For archived content, the media is stored in MPEG-4 format and transcoding delay is due to scaling only. As Fig. 11 shows, the delay increases proportional to the bit-rate encoding. This is due to the computational intensity and memory usage that the transcoder experiences as the target encoding bit-rate increases. Clearly, transcoding live session incurs the highest delay. This additional overhead is due to two factors: (1) delay from the capturing device, (2) computational intensity of transcoding raw video and audio to MPEG-4 bitstream format.

With the application of DL webcasting in mind, our system uses the MPEG-4 Simple Scalable Profile (SSP) [4] that offers a maximum bit-rate of 128 kb/s. Due to the low motion activity present in DL lectures, this provides sufficient video viewing quality for the users. The results indicate that with the SSP, the combination of buffering and transcoding delay for streaming live lecture content is less than 45 s. for both PC and PDA users at a distance of 600 Km to the content server. As seen here, adaptation comes at the cost of additional delay of 25–30 s. While this is larger than the delay incurred when no transcoding is performed, the overhead is small relative to the total duration of a lecture (generally over 1 h). The transcoding delay can be further reduced by using a more powerful server (recall that our server is a Pentium III with 128 MB of RAM) as well as optimized

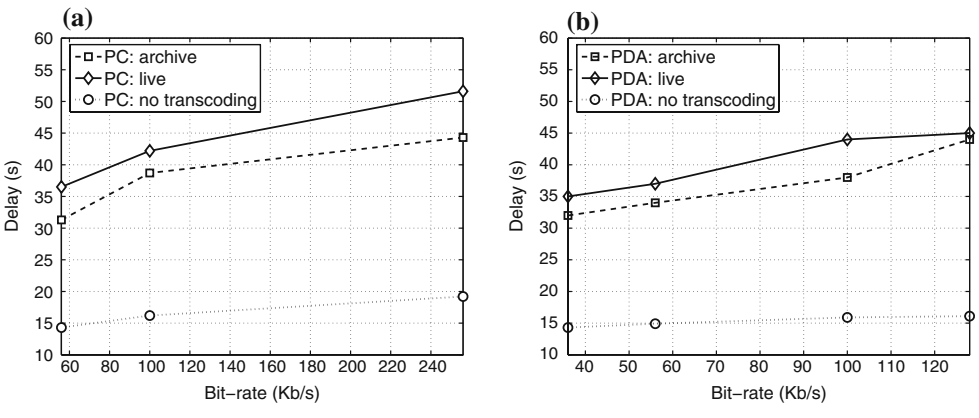



**Fig. 11** Bitstream transcoding delay as a function of encoding bit-rate

PC

PDA

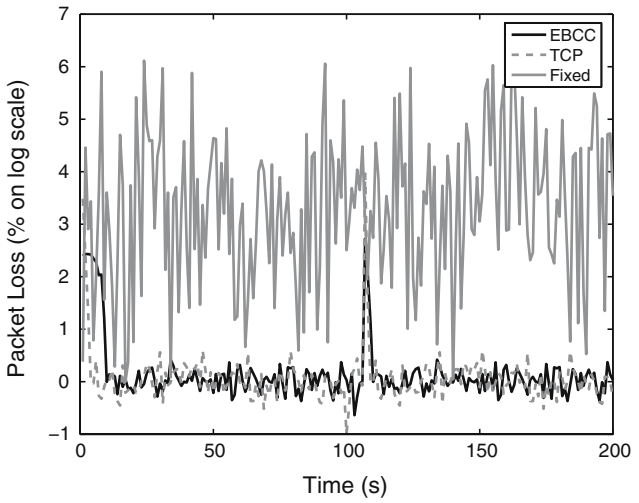



**Fig. 12** Ratio of packets lost over a period of 200 s at a congested network node

encoding and decoding procedures such as those suggested by [9].

### *5.3.5 Bandwidth estimation and congestion control*

As mentioned in Sect. 3.3, a low packet loss ratio is desirable in video. This is because significant errors in one frame can leave multiple frames undecodable as the construction of prediction (P) and bi-directional (B) frames used in MPEG video rely on information from previous or future frames.

Figure 12 compares the percentage of packets lost over a 200 s period using fixed-rate encoding as well as variable encoding using TCP and EBCC bandwidth estimation and control. The results are obtained using the ns-2 simulator with the network topology of Fig. 6. As seen, variable-rate encoding is clearly advantageous to its fixed-rate counterpart. This is due to the fact that overshooting in the fixed-rate case leads to packet loss in a best effort IP network when congestion is evident. However, due to the adaptive nature of TCP and EBCC, congestion control allows both methods to maintain a low level of packet loss (which can be a QoS parameter specified by the client or server). This plot also shows that EBCC can compete respectably with TCP algorithms in terms of packet loss ratio while providing the smooth dynamics desirable in video streaming.

## 6 Conclusions

This paper has addressed the problem of Universal Multimedia Access in the context of DL webcasting. More specifically, seamless delivery of multimedia content to diverse end users in unreliable network conditions has been considered. The proposed solution is a real-time application level and bitstream transcoding solution performed through caching of XSLT transformation sheets, bandwidth estimation, and transmission rate control. The experimental results indicate that the proposed solution is effective in achieving DL content personalization while achieving a small delay overhead. An interesting avenue for future research is the incorporation of multimedia content analysis into the proposed design. This would allow personalization of the actual media content in terms of color, audio, and other preferences.

## References

1. Apple's open-source darwin streaming server, http://developer.apple.com/darwin/projects/streaming/
2. Extensible markup language (xml) 1.0 (second edition) (October 2000). URL http://www.w3.org/TR/REC-xml, W3C Recommendation
3. Extensible stylesheet language (xsl) version 1.0 (October 2001). http://www.w3.org/TR/xsl
4. (2003), Information technology-generic coding of moving pictures and associated audio information = ISO/IEC, Tech. Rep. 13818-7:2003
5. (2004), Information technology-multimedia framework MPEG-21-part1: Vision, technology and strategy, Tech. Rep. TR 21000-1:2004, ISO/IEC
6. (2004), Information technology-multimedia framework MPEG-21- part7: Digit item adaptation, Tech. Rep. TR 21000-7:2004, ISO/IEC
7. Abdelzaher, T.F., Bhatti, N.: Web server QoS management by adaptive content delivery. In: Proceedings of the 7th International Workshop on Quality of Service, pp. 216–225 (1999)
8. Aggarwal, C.C., Hinneburg, A., Keim, D.A.: On the surprising behavior of distance metrics in high dimensional space. In: Lecture Notes in Computer Science, vol. 1973 pp. 420–434 (2001)
9. Almaoui, M.Y.: Metadata driven multimedia transcoding. Master's thesis, University of Toronto, Canada (2005)
10. Baecker, R.: A principled design for scalable internet visual communications with rich media, interactivity, and structured archives. In: Proceedings of the Conference of the Centre for Advanced Studies Conference on Collaborative Research, pp. 16–29 (2003)
11. Baecker, R., Moore, G., Zijdemans, A.: Reinventing the lecture: webcasting made interactive. In: Proceedings of HCI International, vol. 1, pp. 896–900 (2003)
12. Bormans, J., Gelissen, J., Perkis, A.: MPEG-21: the 21st century multimedia framework. IEEE Signal Process. Mag. **20**(2), 53–62 (2003)
13. Bouras, C., Destounis, P., Garofalakis, J. et. al.: Efficient web-based open and distance learning services. J. Telemat. Informat. **17**(3), 213–237 (2000)
14. Brotherton, J., Bhalodia, J., Abowd, G.: Automated capture, integration, and visualization of multiple media streams. In: Proceedings of the IEEE International Conference on Multimedia Computing and Systems, pp. 54–63 (1998)
15. Cirillo, F., Cozzolino, A., Santo, M.D., Marsella, M., Salerno, S.: A metadata based distance learning platform. In:

Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Vol. 1, pp. 44–48 (2000)
16. Cloete, E.: Electronic education system model. Comput. Educ. **36**(2), 171–182 (2001)
17. Cranle, N., Perry, P., Murphy, L.: Perceptual quality adaptation (PQA) algorithm for 3GP and multi-tracked MPEG-4 content over wireless ip networks. In: Proceedings of the IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (2004)
18. Cranley, N., Murphy, L., Perry, P.: Video adaptation: user-perceived quality-aware adaptive delivery of MPEG-4 content, In: Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video, pp. 42–49 (2002)
19. Deshpande, S., Hwang, J.: A real-time interactive virtual classroom multimedia distance learning system. IEEE Trans. Multimedia **3**(4), 432–444 (2001)
20. Floyd, S., Handley, M., Padhye, J., Widmer, J.: Equation-based congestion control for unicast applications. In: SIGCOMM (2000)
21. Friesen, J., Tarman, T.: Remote high-performance visualization and collaboration. IEEE Comput. Graph. Appl. **20**(4), 45–49 (2000)
22. Keesman, G., Hellinghuizen, R., Hoeksema, F., Heideman, G.: Transcoding of MPEG bitstreams. Signal Process. Image Commun. **8**(6), 481–500 (1996)
23. Kinno, A., Yonemoto, Y., Morioka, M., Etoh, M., Environment adaptive XML transformation and its application to content delivery. In: Proceedings of the Symposium of Applications and the Internet, pp. 31–36 (2003)
24. Lemlouma, T., Layaida, M.: Adapted content delivery for different contexts. In: Proceedings of the Symposium of Applications and the Internet (2003)
25. Lepold, K., Quality controlled temporal video adaptation. Dissertation, Institut fur Informaionstechnologie, Universitat Klagenfurt, Germany (2003)
26. Ma, W., Bedner, I., Chang, G., Kuchinsky, A., Zhang, H.: A framework for adaptive content delivery in heterogeneous network environments. In: Proceedings of SPIE/ACM Conference on Multimedia Computing and Networking, vol. 3969, pp. 86–100 (2000)
27. Magalhes, J., Pereira, F.: Using MPEG standards for multimedia customization. Signal Process. Image Commun. **19**(5), 437–456 (2004)
28. Mohan, R., Smith, J., Li, C., Adapting multimedia internet content for universal access. IEE Trans. Multimedia **1**(1), 104–114 (1999)
29. Muntean, C.H., McManis, J., A QoS-aware adaptive Web-based system. In: Proceedings of the IEEE International Conference on Communications, vol. 4, pp. 2204–2208 (2004)
30. Muntean, C.H.: Quality of experience-aware adaptive hypermedia system. Master's thesis, Dublin City University (2005)
31. Padhye, J., Kurose, J.: Continuous-media courseware server: a study of client interactions. IEEE Internet Comput. **3**(2), 65–73 (1999)
32. Pereira, F., Burnett, I., Universal multimedia experiences for tomorrow, IEEE Signal Process. Mag. **20**(2), 63–73 (2003)
33. Pereira, F., Smith, J., Vetro, A.: Introduction to the special section on MPEG-21. IEEE Trans. Multimedia **7**(3) 397–399 (2005)
34. Preiss, B.: Data Structures and Algorithms with Object-Oriented Design Patterns in Java, Wiley, University of Waterloo, Waterloo (1999)
35. Rowe, L., Harley, D., Pletcher, P., Lawrence, S.: Bibs: a lecture webcasting system. Technical Report FCD 21000-1:2007 (2001)
36. Shanableh, T., Ghanbari, M.: Heterogeneous video transcoding to lower spatio-temporal resolutions and different encoding formats. IEEE Trans. Multimedia **2**(2), 101–110 (2000)
37. Tsang, H., Hung, L., Ng, S.: A mulitmedia distance learning system on the internet. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, vol. 2, pp. 243–246 (1999)
38. van Beek, P.: Metadata-driven multimedia access. IEEE Signal Process. Mag. **20**(2), 40–52 (2003)
39. Vetro, A., Christopoulos, C., Sun, H.: Video transcoding architectures and techniques: an overview. IEEE Signal Process. Mag. **20**(2), 18–29 (2003)
40. Vetro, A., Timmerer, C.: Digital item adaptation: overview of standardization and research activities. IEEE Trans. Multimedia **7**(3), 418–426 (2005)
41. Wu, D., Hou, Y., Zhu, W., Zhang, Y.-Q., Peha, J.: Streaming video over the Internet: approaches and directions. IEEE Trans. Circuits Syst. Video Technol. **11**(3) 282–300 (2001)
42. Zhang, H.: Adaptive content delivery: A new research area in media computing. In: Proceedings of the 2000 International Workshop on Multimedia Data Storage, Retrieval, Integration and Applications (2000)

## Authors' biographies

**Mazen Y. Almaoui** received the B.A.Sc. and M.A.Sc. degrees from the Department of Electrical and Computer Engineering at the University of Toronto in 2003 and 2005. He has since been working as a Senior Software Engineer with AMD in the area of video decoding and rendering.

**Azadeh Kushki** received the B.A.Sc. and M.A.Sc. degrees from the Department of Electrical and Computer Engineering at the University of Toronto in 2002 and 2003, respectively. She is currently working toward the Ph.D. degree with particular emphasis on radio and video positioning, multimedia systems, adaptive signal processing, and data fusion methods.

**Konstantinos N. (Kostas) Plataniotis** received his B. Eng. degree in Computer Engineering and Informatics from University of Patras, Greece in 1988 and his M.S. and Ph.D. degrees in Electrical Engineering from Florida Institute of Technology (Florida Tech) in Melbourne, Florida, in 1992 and 1994, respectively. He was with the Computer Technology Institute (C.T.I.) in Patras, Greece from 1989 to 1991. He was a Postdoctoral Fellow at the Digital Signal and Image Processing Laboratory, Department of Electrical and Computer Engineering University of Toronto, Toronto, Canada from November 1994 to May 1997. From September 1997 to June 1999 he was an Assistant Professor with the School of Computer Science at Ryerson University, Toronto, Ontario. He is now an Associate Professor with The Edward S. Rogers Sr. Department of Electrical and Computer Engineering at the University of Toronto in Toronto, Ontario, Canada. His research interests include multimedia systems, image processing, signal processing, communications

systems, biometrics, stochastic estimation, target tracking and pattern recognition.

Kostas Plataniotis is a registered professional engineer in the province of Ontario, and a member of the Technical Chamber of Greece. He is a Senior Member of IEEE, and he served as the IEEE Toronto Section 2004/05 Chair. Kostas is an Associate Editor for the IEEE Transactions on Neural Networks and the IEEE Signal Processing Letters. He is the image processing Area Editor for the IEEE Signal Processing Society's e-letter, the Technical Program Co-Chair for ICME 2006 and the Vice-Chair of ITSC 2006.

Dr. Plataniotis is the 2005 recipient of IEEE Canada's Outstanding Engineering Educator Award "for contributions to engineering education and inspirational guidance of graduate students" and the co-recipient of the 2006 IEEE Transactions on Neural Networks Outstanding Paper Award for the published in 2003 paper "Face Recognition Using Kernel Direct Discriminant Analysis Algorithms" (Juwei Lu, Konstantinos N. Plataniotis and Anastasios N. Venetsanopoulos, IEEE TNN, Vol. 14, no. 1, 2003).