Journal of Mobile Multimedia, Vol. 2, No.4 (2006) 297-310

CLIENT-CENTRIC USAGE ENVIRONMENT ADAPTATION USING MPEG-21

ANASTASIS A. SOFOKLEOUS AND MARIOS C. ANGELIDES
Brunel University
School of Information Systems, Computing and Mathematics
Kingston Lane, Uxbridge, Middlesex UB8 3PH, United Kingdom
{anastasis.sofokleous, marios.angelides}@brunel.ac.uk

Received May 31, 2006
Revised September 20, 2006

Enabling universal multimedia access either focuses research on content adaptation or on the usage environment adaptation. When a client device makes a single video streaming request, either the streaming server adapts the video or the client device adapt its own properties. Where a client device makes multiple concurrent video streaming requests across several streaming servers, if the streaming servers adapt the videos, then concurrent local video adaptation may easily result in a network bottleneck and if the client device adapts its own properties without any consideration of the bandwidth requirements of each video request, of the local usage environment, of the streaming servers' resources and available adaptation solutions then this may easily result in a device bottleneck. In this paper we propose a client-centric usage environment adaptation framework which allows a client device to generate multiple concurrent video streaming requests across several video streaming servers and using MPEG-21 tools describe the client's and servers' usage environment, constraints and resource adaptation policies, determine optimal adaptation solutions and bandwidth requirements for each video request and distribute adaptation across the servers.

Key words: usage environment adaptation, MPEG-21

1. Introduction

Universal Multimedia Access (UMA) defines that any content should be available anytime, anywhere by any device over any type of network [1]. Although UMA has been has been relentlessly pursued, many research challenges arise with regard to a transparent delivery of content and a provision of a high quality of service to end-users. Much of the rich multimedia content cannot be easily handled by client devices because of their limited communication, processing, storage and display capabilities. A number of approaches for enabling universal multimedia access have been advocated by researchers and practitioners. Some researchers focus on content adaptation to suit the usage environment, whilst others focus on the usage environment adaptation to suite the content[2] [3]. For example, when a client device requests video streaming, either the streaming server adapts the video according to the client device properties, e.g. device resolution, and network capacity, e.g. available bandwidth or the client device adapts its own properties, e.g. resolution, to suit the content.

A client device may often request concurrent streaming of multiple videos that are located across multiple connecting streaming servers. If, on the one hand, the streaming servers are responsible for the adaptation process, then concurrent local adaptation may result in a network bottleneck, especially at the "last mile" and on the client device where the bandwidth is usually limited. If, on the other hand, the client device is responsible for the adaptation process, deciding on adaptation solutions and bandwidth for each video request must take into account not only of the local usage environment characteristics but also of the resources and adaptation solutions of each connecting streaming server.

In this paper we focus on the latter, i.e. adaptation on the client device, and propose a client-centric usage environment adaptation framework which enables a client device to generate multiple concurrent video streaming requests from several connecting video streaming servers and using Part 7 MPEG-21 tools, such as the usage environment description (UED), the universal constraints description (UCD) and the Terminal and Network QoS (AQoS), describe the client's and each server's usage environment, constraints and resource adaptation policies, determine optimal adaptation solutions and appropriate levels of bandwidth for each video request and distribute adaptation across the servers. The paper is organised as follows: Section 2 reviews related research, section 3 describes the framework, section 4 evaluates the framework performance and section 5 concludes.

2. Related research review

Usage environment adaptation is either client centric or server centric. The former defines that the adaptation of the usage environment focuses on satisfying the user constraints and preferences instead of various resource sharing issues among many users. On the contrary, the server centric adaptation takes into account, not only the user preferences but also other constraints, such as resources sharing issues on the server (i.e. available bandwidth, memory, CPU). While the client centric adaptation formulates the adaptation strategy without involving other users or streaming sessions, but only what is best for the current user, the server centric approach needs to coordinate the computational and network resources usage and provide average quality of service for more than one user (e.g. differential services on a server).

The usage environment refers to the terminals and network elements utilised at the content retrieval, such as servers, routers, hubs, end user devices. A number of standards (i.e. MPEG-21, 3GPP, CC/PP) provide tools to describe the usage environment. Some usage environment characteristics (i.e. device resolution, bandwidth allocation, network error protection) can be adapted or changed in order to provide better quality of service to a user or to a group of users. The latter differentiates the usage environment adaptation from the content adaptation: the main objective of the usage environment adaptation is to adapt the usage environment resources and characteristics to fit the content requirements and not to change the content characteristics (i.e. content format, content resolution, content bit-rate, frame rate, coefficient dropping) to fit the usage environment properties.

User-centric schemes try to maximise the interests of individual users, while usage environment centric schemes, such as network-centric schemes, optimise collective metrics for all users[4]. The two subclasses use resource management strategies which usually tend to allocate the resource differently and sometimes with very different degrees of fairness (i.e. differentiated services allocate bandwidth to users based on the service and user type). Case studies implement algorithms for adjusting the network capabilities (e.g. bandwidth, delay, error rate) and terminal capabilities (e.g. screen size, terminal power, memory, CPU, decoder) between provider and consumer.

Possible client-centric adaptations include bandwidth adaptation and prioritisation, device properties adaptation (e.g. display resolution, active network interface), memory and CPU management. Some applications can dynamically modify their behaviour based on the remaining power, such as by consuming with reduction in quality when power is scarce or in case of video presentation to change to black and white video presentation so as to save power. This kind of adaptation, adapting the application properties, has been called by some researchers "application-level adaptation" [5]. They report that in some cases, although the application receives the original content it

maybe necessary to change the needs of the application, such as reduce the application window size. [6] propose an MPEG-21 framework for adjusting the QoS of multiple concurrent media streams according to CPU capabilities. The framework focuses on user-level and application-level QoS requirements. User-level QoS includes user-perceived quality and application-level QoS includes the control processes of the system components for a terminal. The system is capable of adapting playback according to system conditions, for example, when a portable device runs on batteries.

Adaptation of applications has been also addressed in distributed multimedia systems[7]. Adaptive-applications are designed to be more tolerant to inevitable fluctuations in the performance of the environment. Client applications can adapt their policies and attributes to their hardware and operating system environment (e.g. to the available memory size and I/O environment). However, the optimal results (i.e. end-user experience, quality of service) in distributed multimedia systems involve adaptation of all the ingredients of the entire system (i.e. server resources, client resources, network characteristics, content).

In [8], the authors discuss the adaptation and mobility in wireless information systems. They point that the wireless services must be able to adapt in changing environments and have "situation awareness" (the exploit of current situation knowledge). An example of "situation awareness" system can be found in [9]. The system, called Odyssey, is characterized as application-aware adaptation for mobile information access. The Odyssey supports adaptation for mobile clients which wish to access a broad range of mobile information applications. The data accessed by the Odyssey system are stored in servers. Applications and the data (i.e. axes of adaptations) define the adaptation strategy. They discuss the fidelity, the decree (consistency and adaptation axes based on the resource) that data presented to the client matched the data stored in server. They identify three types of adaptation strategies:

- laissez-faire which leaves responsibility of the adaptation entirely to the individual applications and user. Furthermore, there is no centralised support, which means it is very difficult to achieve and synchronise the concurrency.
- application-aware is a hybrid approach with its best representative the Odyssey system. The particular approach has system support and concurrency.
- application-transparent which assumes that adaptation and resource management is controlled by the system. Also it has limited support for diversity and applications do no need to be modified.

The authors define the notion of agility as the speed and accuracy used by mobile systems in order to detect and respond to changes in resource availability. Their system monitors resource levels, notifies applications of relevant changes and enforces resource allocation decisions. Each application independently decides how best to adapt when notified, which enable the concurrency. Their system balances the performance and fidelity since both are equally important. The resources of the Odyssey include Network Bandwidth, Network Latency, Disk Cache Space, CPU, Battery Power etc. Users may observe changes in application fidelity but they don't need to direct such changes themselves. The system monitors resource levels, notifies applications of relevant changes and enforces resource allocation decision. Each application independently decides how to respond to notifications. A Resource Refinement Adaptation Control system for mobile devices is proposed in [10]. The system monitors network traffic and device resources, e.g. the CPU and memory utilization, and adjusts either media quality to suit bandwidth requirements or releases system resources that are not in use in order to satisfy the demands of the streaming application.

Application adaptation can be also combined with hardware management for saving power (e.g. energy aware displays) when its full functionality is not used without compromising the usability[11]. In [12], the authors investigate the various methods for varying the clock speed of CPU dynamically under control of the Linux operating system in order to save the CPU energy (power saving) with limited impact on performance. Specifically, they discuss an algorithm which provides power saving by scheduling different jobs to work at different clock rates. They address that with a job workload prediction algorithm, the power can be saved by adjusting the processor to work at a fine grain so it is fast enough to accommodate the workload without missing deadlines that possibly are specified by the human factor. The challenge of saving the battery power of mobile devices using MPEG-21 is addressed in [13]. The constraints vary based on the device's power and user's mobility characteristics, which are described in the usage environment description. The authors introduce a number of power saving techniques at hardware and application levels: (i) turning on power-saving, (ii) dynamic luminance scaling, (iii) dynamic contrast enhancement, (iv) backlight auto-regulation, (v) variable video frame rate, (vi) variable video resolution, and (vii) LCD refresh rate control.

In [14] the authors investigate the dynamic adaptation of operating system policies which aims to improve application performance, e.g. memory and caching management, or system performance, e.g. I/O scheduling. The authors address the challenge of customising adaptation policies for multiple concurrent applications and shared resources. Furthermore, they argue that it may be necessary to improve system constraints in terms of fairness, response time and latency. In [15] the authors present a system which sits on the client side and allows the constraining throughput of certain low-priority flows in order to provide additional bandwidth, for higher-priority flows based on user preferences. Their algorithm, called BWSS (receiver-based bandwidth sharing system), aims to achieve a desired weighted bandwidth partition so as to be able to satisfy user preferences regarding how the bottleneck bandwidth should be shared. Also they point out that the system is able to adjust to congestion as well. They use a TCP Flow Control System (FCS), which achieves a particular target bit-rate for a given TCP connection, by controlling the receiver's advertised window. In [16], a cross-layer adaptation framework called GRACE 1, utilises user preferences to coordinate the adaptation of hardware, OS and application layer on mobile devices. The prototype system aims to maximise quality whilst minimising energy and is capable of adapting the CPU speed at the hardware layer, the CPU scheduling at the OS layer, and the multimedia quality at the application layer.

Although more user centric mechanisms sit on client device, other techniques placed elsewhere can still work for the best user interest as long as the share resources are enough so quality of other users is not affected. However, when the resources start to vanish, the competition among share resources forces client centric adaptation schemes to change to server centric adaptation schemes.

Server and intermediate nodes, such as proxies, can adapt the usage environment for maximising the end-user experience. However, some shared resources, such as the server memory and network bandwidth between server and client, introduce various management issues, which need to be handled fairly across a number of concurrent requests. Thus, usage environment centric schemes attempt to optimise shared resources for the best interest of the server and the network rather than maximising only the experience of individual users. Inappropriate management of these resources can lead to undesirable results, which can affect both the client and the server. For instance, bandwidth misuse which can result in a network bottleneck and bad end-user experience.

Network resources management aims not only to satisfy the application requirements but also to optimize the usage of network recourses. In [17], the authors propose an optimization framework, which enables multiple video senders to coordinate their packet transmission schedules for maximizing the average quality. A distributed packet scheduling algorithm allows the trade off rate and distortion, not only within a single video stream, but also across different streams. Decisions taken are based on the packet size and importance of the packet for the reconstruction quality of the corresponding stream. The performance of the framework was evaluated using non-scalable content while (i) adapting the bandwidth and (ii) adapting the packet loss through prioritized packet retransmissions.

In [18], the authors address the challenge of allocating bandwidth for scalable MPEG format in order to provide deterministic guarantees such as no packet loss and no missed deadlines. They propose a policy, namely the quasi-constant policy, which reduces the quality of transmission of the scalable video when congestion occurs. The Variable Bit Rate (VBR) can be used by (i) a service with stochastic guarantees, which limits the packet loss rate and the percentage of packets that miss the deadline or by (ii) a service with deterministic guarantees, which ensures that no packet is dropped or delayed beyond the deadline. They focus is on resource allocation (buffer and bandwidth) for VBR video traffic-coded according to the MPEG 2 algorithm.

Although the server centric scheme aims to manage and optimise shared resources, some approaches involve user preferences in their adaptation strategies. In [19], the authors propose a mathematical model for the problem of allocating multiple resources to satisfy the QoS needs of multiple sessions and QoS profiles. The model captures the problem of adaptively adjusting the quality of each individual session, in order to maximise some objective functions from a system-level perspective, taking into account the users' preferences and subject to a set of system resource constraints. The model shows that the problem is hard optimised. For solving the problem they propose a heuristic algorithm whose performance and results make it appropriate to be used in real time multimedia applications.

The management of shared resources had been addressed in [20] as well. The authors present an algorithm load balancing algorithm, which dynamically optimizes the disk usage in online video servers. Their algorithm, called Dynamic Segment Replication Policy, balances the load across the disk by replicating segments of files based on their request ratio. Therefore, by adjusting the file system, the system manages to response quickly to all kind of requests.

All algorithms discussed above run on a server. Some other approaches place the algorithms on intermediate nodes, such as proxies. For instance in [21], the authors introduce new cache policies for proxy that assist in providing better QoS for multimedia streaming over the internet. In particular, they propose four new algorithms: (i) a weighted replacement policy to improve the cache hit ratio of mixed media (e.g. image, video), (ii) a cache reallocation algorithm that improves the cache utilization performance by adjusting to network conditions and media characteristics (iii) a QoS-adaptive miss strategy and pre-fetching algorithm that works based on the network conditions (iv) two request schemes (i.e. server-proxy, proxy-client) that allocate appropriately the network resources.

In [22] the authors present a self-adaptive distributed proxy system that provides streaming multimedia service to mobile wireless clients. Their system passively monitors all the nodes (i.e. network, server, and clients) and adapts in order to provide better end-user experience. The system among other utilizes a service, called Automatic Path Creation, which uses the gathered information to create optimal logical and physical paths. The logical path uses DAG (Directed Acyclic Graph) and

matches the search problem with a shortest path graph search problem. The optimization criteria are application-specific, e.g., data throughput, jitter, delay, video/audio quality metrics. On the other hand, the Physical Path aims to determine the optimal placements (physical location) of each operator of the logical path. In [23], the authors propose a distributed architecture, which minimizes the access time of clients with a fault tolerant system that balances the load of video-on-demand requests to multiple distributed servers. The authors employ a retrieval strategy for multiple servers, which are used and coordinated with the aid of the Jini technology. Each client streaming session retrieves different portions of the video from multiple servers based on the server and network available resources. Furthermore an agent, which is registered as a service, is responsible for the coordination of the clients-servers activities, such as how, where and which servers to contact for retrieving different portions of the movies. [24] propose a framework which uses the PICO middleware and supports adaptation for multimedia delivery in heterogeneous wireless networks. By exploiting the usage environment, the system provides transparent access to multimedia data and is able to adapt to changes in the usage environment.

The content adaptation has been used widely for enabling the Universal Multimedia Access. The interoperability between such applications can be achieved with the deployment of standards, such as the MPEG-7 and MPEG-21 [25]. The MPEG-7 Multimedia Description Scheme (MDS) provides the necessary tools for describing, searching, processing and filtering multimedia content [26][27]. Application cases demonstrating the usage of MPEG-7 can be found in[2]. On the other hand, the MPEG-21 standard provides a number of tools, which enable the use of multimedia resources across a wide range of networks and devices[28]. One of the goals of MPEG-21 is to achieve interoperable, transparent and secure access. Part 15 of MPEG-21, Event Reporting, allows control and secure adaptation, distribution and consumption of multimedia content between users by defining the mechanism by which every action can be specified, detected and reported [29]. On receipt of report events, applications can modify their behaviour. A framework, Axmedis, that uses the MPEG-21 Event Reporting and MPEG-21 REL for content production, protection, adaptation, distribution to multiple channels and digital rights management is described in [30]. Axmedis also supports integration with other systems, such as Content Management systems.

3. A development framework for client-centric usage environment adaptation using MPEG-21

The framework enables a client to generate multiple concurrent video streaming requests from several connecting video streaming servers. It employs an adaptation decision engine (ADE) which uses MPEG-21's AQoS, UCD and UED tools to describe a client's and each server's usage environment, constraints and supporting resource adaptation policies in order to determine an optimal adaptation solution and bandwidth allocation for each video request made by the client and to distribute the adaptation process across the connecting streaming servers. Each connecting streaming server hosts a resource adaptation engine (RAE) which adapts the video (Figure 1).

3.1 The MPEG-21 UED, AQoS and UCD

The MPEG-21 standard provides a number of tools, which enable the use of multimedia resources across a wide range of networks and client devices. Part 7 of the MPEG-21 standard describes various tools pertinent to the Digital Item Adaptation. The standard specifies only the tools that assist the adaptation process but not the adaptation engine. The framework uses three tools from the MPEG-21 standard: UED, UCD and AQoS.

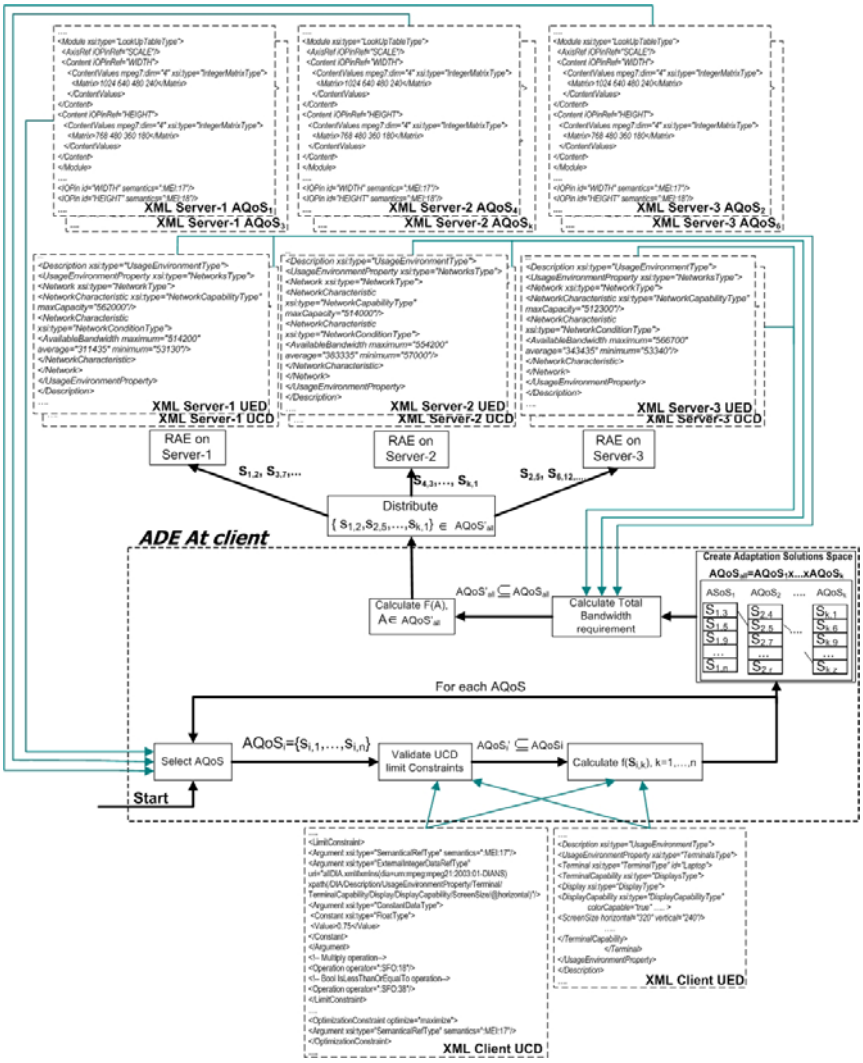

Figure 1: The development framework

The UED describes the terminal, user, network and natural environment characteristics. Terminal characteristics include encoding and decoding, client device properties, such as power and storage, and input-output. User characteristics include user information, user and usage preferences such as audiovisual content and presentation, accessibility, mobility and usage history. Network characteristics include utilisation, delay and error. Natural environment characteristics include information about the audiovisual environment such as noise levels and noise frequency spectrum, location and time. Figure 2 describes the terminal capabilities of a mobile client device using the UED tool.

AQoS assists with maximising the quality of service by selecting optimal adaptation parameters. Specifically, it describes the relationship between constraints, adaptation operations and media resource qualities. The AQoS tool is modular and, therefore, an instance of AQoS can be constructed using the interconnected modules of *UtilityFunction*, *LookUpTable* or *StackFunction*. Figure 3

demonstrates a *LookUpTable* with colour reduction (IS_COLOR) and resolution reduction (SCALE) as axes and file size (FILE_SIZE) as content.

```
<DIA ….>
<Description xsi:type="UsageEnvironmentType">
    <UsageEnvironmentProperty xsi:type="TerminalsType">
        <Terminal xsi:type="TerminalType" id="PocketPC">
        <TerminalCapability xsi:type="DisplaysType">
                <Display xsi:type="DisplayType">
                <DisplayCapability xsi:type="DisplayCapabilityType" colorCapable="true" activeDisplay="true">
                        <Mode>
                                <Resolution horizontal="320" vertical="240"/>
                        </Mode>
                    <ScreenSize horizontal="320" vertical="240"/>
                </DisplayCapability>
                </Display>
        </TerminalCapability>
    <……..>
    </Terminal>
</UsageEnvironmentProperty>
</Description>
</DIA>
```

Figure 2: UED – terminal capabilities

```
  <Module xsi:type="LookUpTableType">
   <AxisRef iOPinRef="SCALE"/>
   <AxisRef iOPinRef="MEDIA_COLOUR"/>
   <Content iOPinRef="FILE_SIZE">
    <ContentValues xsi:type="IntegerMatrixType" mpeg7:dim="4 2">
     <Matrix>
  38330 38430 15900 23700
  7100 14100 3320 7530
     </Matrix>
    </ContentValues>
   </Content>
  </Module>
  <IOPin id="MEDIA_COLOUR" semantics=":MEI:13">
    <Axis>
  <AxisValues xsi:type="TokenVectorType">
          <Vector>colour greyscale </Vector>
  </AxisValues>
    </Axis>
  </IOPin>
  <!-- File size of the video-->
  <IOPin id="FILE_SIZE" semantics=":MEI:3"/>
  <!-- Scale factor of the video -->
  <IOPin id="SCALE">
    <Axis>
  <AxisValues xsi:type="IntegerVectorType">
          <Vector>0 1 2 3</Vector>
  </AxisValues>
    </Axis>
  </IOPin>
```

Figure 3: AQoS – LookUpTable

```
<DIA ….>
<Description xsi:type="UCDType">
  <AdaptationUnitConstraints>
        <LimitConstraint id="3">
      <Argument xsi:type="SemanticalRefType" semantics=":MEI:17"/>
      <Argument xsi:type="ExternalIntegerDataRefType" uri="deviceUED.xml#xmlns(dia=urn:mpeg:mpeg21:2003:01-
DIANS)(/DIA/Description/UsageEnvironmentProperty/Terminal/TerminalCapability/Display/DisplayCapability/ScreenSize/
@horizontal)"/>
      <Argument xsi:type="ConstantDataType">
        <Constant xsi:type="FloatType">
          <Value>0.75</Value>
        </Constant>
      </Argument>
      <!-- Multiply operation-->
      <Operation operator=":SFO:18"/>
      <!-- Bool IsLessThanOrEqualTo operation-->
      <Operation operator=":SFO:38"/>
     </LimitConstraint>
      <!--……..-->
     <OptimizationConstraint optimize="maximize">
     <Argument iOPinRef="VIDEO_QUALITY" xsi:type="ExternalIOPinRefType"/>
     </OptimizationConstraint>
     <!--……..-->
  </AdaptationUnitConstraints>
</Description>
</DIA>
```

Figure 4: UCD – optimization and limit constraints

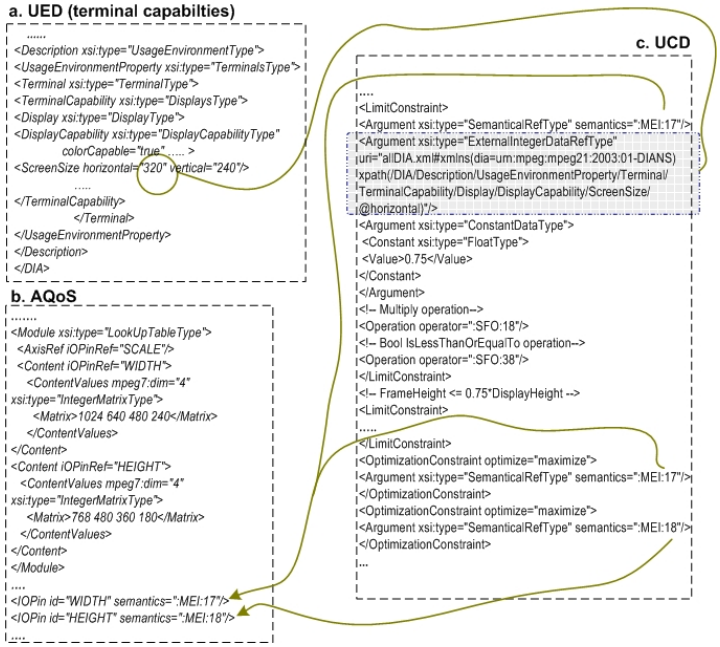

Figure 5: AQoS, UED and UCD linkage

UCD describes limitation and optimisation constraints on those AQoS parts that affect adaptation decisions. The tool consists of arguments and operators. Argument types are variables representing usage environment characteristics or AQoS's IOPins [28]. Whilst AQoS values are obtained dynamically using the IOPIN's, UED values are obtained either using XPath expressions or semantics from classification schemes. Figure 4 shows an example of the UCD.

Figure 5 illustrates how a UCD references dynamically UED and AQoS values. Figure 1a shows a portion of a client device's UED. Figure 1b shows possible adaptation solutions in an AQoS. AQoS IOPins are linked to semantics, such as frame width, with ":MEI:17" defined with *MediaInformationCS*, a classification scheme shared by UCD and AQoS. Semantics allow to specify limit and optimisation constraints without knowing the actual AQoS implementation. Figure 1c shows a portion of the UCD. The first constraint in UCD comprises of three arguments and two operators. The *SemanticalRefType* argument obtains its value from AQoS using ":MEI:17". The *ExternalIntegerDataRefType* argument uses *XPATH* to obtain the value of the horizontal screen size from UED. The *ConstantDataType* argument is a constant value of 0,75. In addition to the three arguments, the limit constraint uses two operators, namely multiply (:SFO:18) and equal or less (:SFO:38). "SFO" is defined with the *StackFunctionOperatorCS* classification scheme.

$$\begin{bmatrix} O_1(s_{i,1}) & O_2(s_{i,1}) & ..... & O_d(s_{i,1}) \\ O_1(s_{i,2}) & O_2(s_{i,2}) & ..... & O_d(s_{i,2}) \\ ..... & ..... & ..... & ..... \\ O_1(s_{i,k}) & O_2(s_{i,k}) & ..... & O_d(s_{i,k}) \end{bmatrix} \rightarrow \begin{bmatrix} \overline{O}_1(s_{i,1}) & \overline{O}_2(s_{i,1}) & ..... & \overline{O}_d(s_{i,1}) \\ \overline{O}_1(s_{i,2}) & \overline{O}_2(s_{i,2}) & ..... & \overline{O}_d(s_{i,2}) \\ ..... & ..... & ..... & ..... \\ \overline{O}_1(s_{i,k}) & \overline{O}_2(s_{i,k}) & ..... & \overline{O}_d(s_{i,k}) \end{bmatrix} \quad \text{(a)}$$

$$\begin{bmatrix} f(s_{i,1}) \\ f(s_{i,2}) \\ .... \\ f(s_{i,k}) \end{bmatrix} \rightarrow \begin{bmatrix} \overline{f}(s_{i,1}) \\ \overline{f}(s_{i,2}) \\ .... \\ \overline{f}(s_{i,k}) \end{bmatrix} \quad \text{(b)}$$

Figure 6: normalising (a) optimisation values, (b) f values

3.2. The ADE

Initially the client device generates k requests and receives *k* AQoS from more than one connecting video streaming servers. The ADE selects an AQoS, denoted as $AQoS_i$, $0<i\leq k$ and executes the first three steps (**Select AQoS, Validate UCD limit constraints, Calculate** $f(S_{i,k})$).

Select AQoS: Create set $AQoS_i = \{s_{i,1}, s_{i,2}, s_{i,3} \ldots s_{i,n}\}$ where each $s_{i,k}$ incorporates values of adaptation operations and associated results $s_{i,k}$ = {bandwidth value, frame rate value, media colour value, resolution reduction value, PSNR value}. For example, the AQoS in figure 3 gives AQoS = [{0, colour, 38330},{1, colour, 38430},{2, colour, 15900},{3, colour, 23700},{0, greyscale, 7100},{1, greyscale, 14100},{2, greyscale, 3320}, {3, greyscale, 7530}] where each solution $s_{i,k}$ = {SCALE value, MEDIA_COLOUR value, FILE_SIZE value}.

Validate UCD limit constraints: The $AQoS_i$ set is validated against the UCD limit constraints, which may include arguments that reference the UED, e.g. device resolution and current available bandwidth. The result is a new set $AQoS'_i$, where $AQoS'_i \subseteq AQoS_i$. $AQoS'_i$ contains only those solutions that satisfy all limit constraints, $AQoS'_i = \{s_{i,k} \mid s_{i,k} \in AQoS_i \ \& \ L_1(s_{i,k}) \leq 0,..., L_d(s_{i,k}) \leq 0\}$, where $L_z(s_{i,k})$ is the evaluation of the z^{th} limit constraint (L_z) using $s_{i,k}$. A UCD limit constraint may be $L_z(s_{i,k}) = R(S_{i,k}) - D_{resolution} \leq 0$, where $R(S_{i,k})$ is the resolution of the adapted video and $D_{resolution}$ is the device resolution.

Calculate $f(s_{i,k})$: Valid solutions from each set are selected using: $f(s_{i,k}) = \frac{w_1.O_1(s_{i,k}) + w_2.O_2(s_{i,k}) + .. + w_d.O_d(s_{i,k})}{w_1 + w_2 + ... + w_d}$, where $s_{i,k} \in AQoS'_i$, $w_1,..,w_d$ are system weights and $O_d(s_i)$ returns the normalized value of optimisation constraint *d* using the $s_{i,k}$. The ADE calculates all optimisation values before normalisation since it requires the minimum and maximum value of each $O_z(s_i)$, $0<z\leq d$. It uses a *dxk* matrix to store all values, where d is the number of optimization constraints and k the number of solutions. It then determines the min-max values of each column, $O_1(s_{i,1}),...,O_1(s_{i,k})$ and uses these values to normalise each column to $\overline{O}_1(s_{i,1}),...,\overline{O}_1(s_{i,k})$ (figure 6a). It then uses the new matrix to calculate $f(s_{i,k})$ and normalise all f values within the range [0-1] using min-max (figure 6b). The ADE repeats all steps for each AQoS.

Create adaptation solutions space: The ADE creates an adaptation solutions space that contains all sets and their valid solutions: $AQoS_{all} = AQoS'_1 \, x AQoS'_2 ... x AQoS'_i$. For example, solution $A \in AQoS_{all}$ where $A = \{s_{1,a}, s_{2,b}, ...s_{k,z}\}$, $s_{1,a} \in AQoS_1, s_{2,b} \in AQoS_2, ..., s_{k,z} \in AQoS_k$.

Calculate total bandwidth requirement: The framework first ensures that the total bandwidth requirement does not exceed the available bandwidth between the client and the connecting video streaming servers. It then ensures that the total bandwidth requirement of all adaptation solutions that will be distributed to each connecting video streaming server will not exceed the server's available bandwidth capacity. Each server's available bandwidth capacity is described in their XML Server UED. The result is a new set, $AQoS'_{all}$, where $AQoS'_{all} \subseteq AQoS_{all}$. $AQoS'_{all}$ contains only those solutions whose total bandwidth requirement does not exceed the total bandwidth capacity between client and servers, $AQoS'_{all} = \{A \mid A \in AQoS_{all} \ \& \ L(A) \leq 0 \ \& L_1(A_1) \leq 0,..., L_3(A_3) \leq 0\}$, where $A = \{s_{1,a}, s_{2,b}, ...s_{k,z}\}$, $s_{i,a} \in AQoS_i$. The total bandwidth constraint is expressed as $L(A) = \sum_{n=1}^{i} B(A) - N_{Bandwidth} \leq 0$, where $N_{Bandwidth}$ is the total available bandwidth and $B(A)$ is the bandwidth required by solution A. The bandwidth constraint for each connecting video streaming server *i* is expressed as $L_i(A_i)$, where A_i are the set of solutions distributed to a connecting video streaming server *i* and $A_i \subseteq A$.

Calculate $F(s_{1,a},...s_{i,z})$**:** Adaptations solutions that will be distributed to the servers are selected:

$$F(s_{1,a},...s_{i,z}) = \frac{w_1 . f(s_{1,a}) \ + ... + w_i . f(s_{i,z})}{w_1 + ... + w_i}, \qquad \text{where } s_{1,a} \in AQoS_1, ..., s_{i,z} \in AQoS_i .$$

$f(s_{i,k})$ values have already been normalised.

Distribute $A = \{s_{1,a}, s_{2,b}, ...s_{k,z}\}$**:** The combination of adaptations solutions with the maximum fitness function value F(A) will be distributed to the connecting video streaming servers to direct each servers' RAE to adapt the requested videos. Adapted videos are sent to client device for consumption.

4. Experiments and results

We tested the proposed framework in a controlled network environment consisting of several streaming video servers and client devices. Each server hosted media files encoded at different format, quality and bit rate settings. Using a bandwidth control system, the network bandwidth was adjusted to 512kbits. The AQoS of each media file used *utilityfunction* to describe possible adaptation solutions in relation to bandwidth and video quality (PSNR). Adaptation operations used were resolution, frame rate and colour reduction. UEDs described the usage environments and UCDs described limit and optimisation constraints of both client devices and servers. In the experiment device resolution was set at 1024x768pixels. In our experiment, we set the following constraints:

- L_1:Limit constraint: video resolution ≤ device resolution
- L_2:Limit constraint: video bandwidth ≤ network bandwidth
- O_1: Optimization constraint: Maximize video quality
- O_2:Optimization constraint: maximize video resolution

In figure 7, a client device generated 3 concurrent video streaming requests to 3 servers. The ADE received 3 AQoSs, denoted as $AQoS_1$, $AQoS_2$ and $AQoS_3$. The solutions of each AQoS were validated and evaluated using the above limit constraints (L_1,L_2) and optimisation constraints (O_1,O_2) respectively. The final adaptation solutions were validated against the total bandwidth requirement. Figure 7 shows the initial bandwidth required by the three videos and the average bandwidth per video allocated during adaptation. Figure 8 shows the bandwidth consumed over 10 seconds by each video.

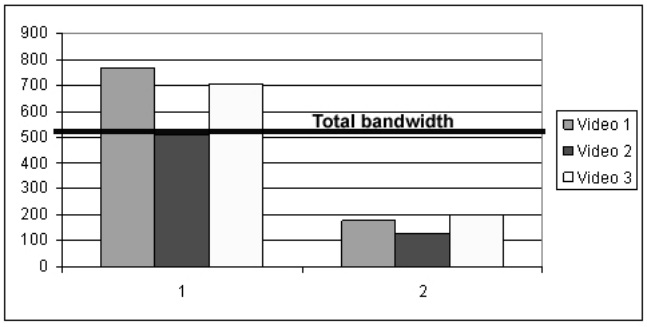

Figure 7: Initial bandwidth requirement and allocated bandwidth

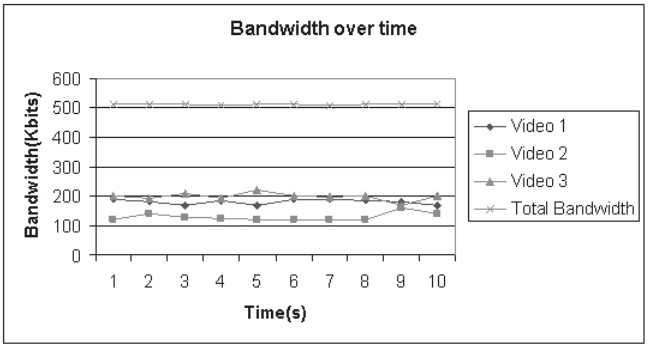

Figure 8: Bandwidth over time

5. Conclusions

This paper addresses several ephemeral issues in the context of universal multimedia access. Several approaches use content adaptation to enable multimedia access from devices with limited communication, processing, storage and display capabilities. In this paper we address the problem of the "last mile" bottleneck, in which the client has requested multiple media files concurrently from more than one servers and possibly over the available bandwidth. Conventional approaches place the ADE and RAE on the same node and use shared bandwidth competitively which is a main cause of such a network bottleneck. To address this problem, we propose a usage environment adaptation framework to reside on the client that determines optimal adaptation solutions and manages the shared bandwidth of the multiple flows. This is achieved by retrieving AQoSs from connecting video streaming servers, and using the UED and UCD. Optimal solutions are distributed back to the connecting video streaming servers which use these to adapt the requested video and communicate it to the client device. In order to calculate optimal solutions the current work uses system weights, which we accept may cause bias towards the final adaptation strategy. In future work we will consider mechanisms that will allow the framework to work without system weights.

References

1. Kasutani, E., New Frontiers in Universal Multimedia Access. Tech. Rep. ITS Rept.04.22 (2004)
2. van Beek, P., Smith, J. R., Ebrahimi, T. et al., Metadata-Driven Multimedia Access. IEEE Signal Processing Magazine, 20(2). (2003) 40-52
3. Vetro, A., Timmerer, C., Devillers, S., Digital item adaptation. In: I. S. Burnett, F. Pereira, Van de Walle, R. et al., The MPEG-21 Book. Wiley, Hoboken, NJ (2006)
4. Feng, N., Mau, S. C., Mandayam, N. B., Pricing and Power Control for Joint Network-Centric and User-Centric Radio Resource Management. IEEE Trans. Commun., 52(9). (2004) 1547
5. Boszormenyi, L., Hellwagner, H., Kosch, H. et al., Metadata Driven Adaptation in the ADMITS Project. Signal Process Image Commun, 18(8). (2003) 749-766
6. Di Cagno, G., Concolato, C., Claude Dufourd, J., Multimedia Adaptation in End-User Terminals. Signal Process Image Commun, 21(3). (2006) 200-216
7. Gecsei, J., Adaptation in Distributed Multimedia Systems. Multimedia, IEEE, 4(2) (1997) 58-66
8. Katz, R. H., Adaptation and Mobility in Wireless Information Systems. Communications Magazine, IEEE, 40(5). (2002) 102-114
9. Noble, B. D., Satyanarayanan, M., Narayanan, D. et al., Agile Application-Aware Adaptation for Mobility. in Proc. of 16th ACM Sympo. on Operating Systems Principles (1997) 276-287
10. Kung, H., Hua, J., Chang, Y. et al., Seamless QoS Adaptation Control for Embedded Multimedia Communications. IEEE Trans. On Consumer Electronics, 52(1). (2006) 240

11. Ranganathan, P., Geelhoed, E., Manahan, M. et al., Energy-Aware User Interfaces and Energy-Adaptive Displays. IEEE Computer, 39(3). (2006) 31-38
12. Weiser, M., Welch, B., Demers, A. et al., Scheduling for Reduced CPU Energy. ," in Proceedings of the First Symposium on Operating Systems Design and Implementation (OSDI) (1994) 13-23
13. Shim, H., Cho, Y., Chang, N., Power Saving in Hand-Held Multimedia Systems using MPEG-21 Digital Item Adaptation. 2nd Workshop on Embedded Systems for Real-Time Multimedia, ESTImedia 2004, (2004) 13-18
14. Teller, P. J., & Seelam, S. R., Insights into Providing Dynamic Adaptation of Operating System Policies. ACM SIGOPS Operating Systems Review, 40(2). (2006) 83-89
15. Mehra, P., De Vleeschouwer, C., Zakhor, A., Receiver-Driven Bandwidth Sharing for TCP and its Application to Video Streaming. IEEE Transactions on Multimedia, 7(4). (2005) 740-752
16. Yuan, W., Nahrstedt, K., Adve, S. V. et al.: GRACE-1, Cross-Layer Adaptation for Multimedia Quality and Battery Energy. IEEE Transactions on Mobile Computing, 5(7). (2006) 799-815
17. Chakareski, J., & Frossard, P., Rate-Distortion Optimized Distributed Packet Scheduling of Multiple Video Streams Over Shared Communication Resources. IEEE Transactions on Multimedia, 8(2). (2006) 207-218
18. Conti, M., & Gregori, E., Bandwidth Allocation for the Transmission of Scalable MPEG Video Traffic with Deterministic Guarantees. Real Time Imaging, 7(3). (2001) 237-253
19. Lawabni, A. E., & Tewfik, A. H., Resource Management and Quality Adaptation in Distributed Multimedia Networks. in Proceedings of the 10th IEEE Symposium on Computers and Communications (ISCC 2005) (2005) 604-610
20. Dan, A., Kienzle, M., Sitaram, D.: A Dynamic Policy of Segment Replication for Load-Balancing in Video-on-Demand Servers. Multimedia Systems, 3(3) (1995) 93-103
21. Yu, F., Zhang, Q., Zhu, W., Zhang, Y., QoS-Adaptive Proxy Caching for Multimedia Streaming Over the Internet. IEEE Transactions on Circuits and Systems for Video Technology, 13(3) (2003) 257-269
22. Mao, Z. M., So, H. W., Kang, B., Network Support for Mobile Multimedia using a Self-Adaptive Distributed Proxy. in Proceedings of the 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video (2001) 107-116
23. Veeravalli, B., Chen, L., Kwoon, H. Y. et al., Design, Analysis, and Implementation of an Agent Driven Pull-Based Distributed Video-on-Demand System. Multimedia Tools and Applications, 28(1) (2006) 89-118
24. Kalasapur, S., Kumar, M., Shirazi, B., Personalized Service Composition for Ubiquitous Multimedia Delivery. in Proceedings of the Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM'05) (2005) 258-263
25. Vetro, A., & Timmerer, C., Digital Item Adaptation: Overview of Standardization and Research Activities. IEEE Transactions on Multimedia, 7(3). (2005) 418-416
26. Martínez, J. M., MPEG-7 Overview of MPEG-7 Description Tools, Part 2. IEEE Multimedia, 9(3) (2002) 83-93
27. Martínez, J. M., Koenen, R., Pereira, F., MPEG-7 the Generic Multimedia Content Description Standard, Part 1. IEEE Multimedia, 9(2). (2002) 78-87
28. ISO/IEC 21000-7:2004, Information Technology - Multimedia Framework- Part 7: Digital Item Adaptation.
29. Tokmakoff, A., Nuttall, F., Ji, K., MPEG-21 Event Reporting, Enabling Multimedia E-Commerce. Multimedia, IEEE, 12(4). (2005) 50-59
30. Chellini, S., Martini, T., Nesi, P., AXMEDIS: An MPEG-21 Based Solution for Protected Cross Media Content Production and Distribution. in Proceedings of the 8th IEEE International Conference on E-Commerce Technology and the 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE'06) (2006) 52-55