On Balancing Between Transcoding Overhead and Spatial Consumption in Content Adaptation

Wai Yip Lum and Francis C.M. Lau
Department of Computer Science &
Information Systems
The University of Hong Kong
Hong Kong

{wylum,fcmlau}@csis.hku.hk

ABSTRACT

We propose a method that can find the optimal tradeoff point between transcoding overhead (CPU cost) and storage needed for the various pre-processed content variants (I/O cost). The method selectively pre-adapts a subset of content variants and leaves the generation of the residue to dynamic content adaptation with this pre-adapted subset as an input. We prove bounds regarding the optimality of the algorithm employed. The proposed model creates a collaborative environment across the components of client, proxy and server, based on which we study the distribution of adaptation complexity across these components. We use simulation to verify the projected benefits. The method has been successfully implemented in a trial PDF document content adaptation system.

Categories and Subject Descriptors

H.4.m [**Information Systems**]: Miscellaneous; I.6.m [**Simulation and Modeling**]: Miscellaneous

General Terms

Design, Performance, Algorithms, Experimentation

Keywords

Content adaptation, pre-adaptation, mobile computing, pervasive computing, performance optimization

1. INTRODUCTION

Mobile computing is creating a new class of applications and new massive markets combining mobile services and hand-held device electronics. The limited computational power of client devices and the constrained cellular connectivity give rise to many opportunities for the design of new patterns of computation for effective presentation of contents available from the Internet. The problem is nontrivial as web pages are becoming increasingly rich in content and varied in format and style, and client devices are getting diversified in their rendering capabilities.

While web users complain about the "World Wide Wait" problem due partly to slow last-mile speeds, cellular networks are working at even lower data rates, which can work fine for plain text, but far from adequate for web pages [8]. The mismatch between rich multimedia contents and the constrained client capability presents a research challenge. *Content adaptation* has emerged as a potential technology to tackle some of the problems arising from the mismatch. Content adaptation generally involves a series of transcoding phases which could be very time consuming, and is therefore unacceptable to mobile device users [4]. There is a need to make content adaptation work much more efficiently—this is the subject of this paper.

1.1 Related work

Content adaptation involves creating different versions suitable for rendering by mobile devices from some original content such as a web page. Content adaptation can be classified into two main types according to when these different content variants are created [18]. In *static adaptation*, different "pre-adapted" versions are created and stored in the server at content creation time. During runtime, a version that best matches the desirable version in a request will be returned. The Odyssey system [26] uses pre-adapted versions; so does InfoPyramid [24], where the content variants are arranged in a pyramid-like representation scheme.

To create a pre-adapted content version, a human designer can be involved to hand-tailor a version for some specific rendering requirement. The models on semantic equivalence in [2] support a strategy for augmentation of multimedia documents by semantically equivalent presentation alternatives. To help the authoring process, some helper tools [16, 25] have been developed. For example, the video annotation feature in [25] allows linking of video segments with corresponding text segments as alternatives for the constrained environment.

With the static adaptation approach, the content provider can have a tight control over what is transcoded and how the result is presented [3]. The primary shortcoming of static adaptation however is that the management and maintainence of a number of content variants in the server could

MOBICOM'02, September 23–26, 2002, Atlanta, Georgia, USA.

result in considerable storage and other I/O costs. Although storage costs are rapidly coming down nowadays, given the huge volume and variety of multimedia contents and the diversity of rendering capabilities of the devices, pre-computing a sufficient number of content variants to cover all possible client requests is impractical. If only a limited set is stored, there would arise the awkward situation where the most optimal version of content falls between two pre-adapted ones. Increasing the number in the set, however, would increase the I/O costs.

To achieve a good match for specific devices, the original content can be pre-adapted along the device dimension—*i.e.*, the matched content version would be a best fit for the device's capabilities. Yahoo.com [31] and e-bay.com [10] for instance have created multiple versions of contents for different types of devices including pagers, PDA's and web-enabled phones. This static adaptation approach requires upgrading whenever a new device needs to be supported. Given the increasing number of devices, this could be an issue. Furthermore, "device" is only one of several possible dimensions which the adaptation process can take into account, one of them being the user's own preference.

In *dynamic adaptation*, the desired content is synthesized on the fly, according to dynamic requirement presented to the server, which could be based on the current characteristics of the client environment [1, 5, 6, 11, 12]. This approach presents no storage problem to the server, but the transcoders could be overwhelmed by heavy workload as transcoding is computationally intensive [6, 14], and the delay incurred can be orders of magnitude larger than that of accessing just the original content.

In fact, adaptation operations can take place in the server, the client or the intermediary proxy. The end-to-end approach suggests that all the manipulations of a web object should be done at the end points of the network and not at any intermediary node. The argument for this is that the end-points could carry out more optimized and complex transformations based on client needs [22]. These operations should be well within the capabilities of a typical server, and the approach is referred to as *server-based adaptation* [26]. In client-based adaptation [35], the required transcoding is performed by the client device taking into account the device's capabilities which might not otherwise be available to the remote server. The transcoding however may require a considerable amount of resources on the client side, which cannot be assumed to always exist. The idea of "thin client" [11, 28] comes in here, which suggests that only some minimal amount of rendering should be done at the client device, leaving the bulk of the operation to some nearby proxy server. The idea was tested in [13] where experiments were designed to compare some image processing functions being performed entirely by a standard PDA versus letting a proxy to help with computing some of the computationally intensive tasks. The results clearly showed that the response time could be much improved when the process is assisted by a proxy server.

Alternatively, we could rely on the proxy server to take up the entire task of producing the needed content version, without having to involve the client at all. This is the *intermediary-based approach* where the proxy server is the intermediary situated in between the client and the content-providing server [1, 5, 6, 11, 12]. The advantage of this approach is that it simplifies the design of the provider server and the client device by pushing the transformation complexity to the intermediary infrastructure. The potential problem is that when the intermediary performs transcoding, it might not have access to the necessary semantic information needed for a satisfactory operation. Context-aware adaptation [18, 34, 33, 32, 9] aims to solve this problem, where context information containing implicit or explicit guidance can be provided to the intermediary. This information could be provided by the server as in *server-directed transcoding* [23]. The intermediary-based approach is generally not very scalable since transcoding is a resource intensive process.

We can build a model for a mixed approach, which combines the advantages of the various approaches just discussed but tries to avoid their problems.

1.2 Motivations

In this paper, we propose a method to achieve an optimal tradeoff between the real-time transcoding overhead (CPU cost) of the dynamic approach and the storage overhead of pre-adaptation (I/O cost) of the static approach. Our hybrid approach selectively pre-adapts a subset of content variants and leaves the generation of the residue to a dynamic algorithm that uses the pre-adapted subset as a base. The algorithm considers the following in coming up with a solution.

1. When: Whether the adaptation should be performed at presentation time as in the dynamic adaptation approach or at content creation time as in the static approach.

2. Where: The model should create a collaborative environment between the proxy and the server and give insight on how the adaptation complexity should be shared between the two components.

3. What: Original untranscoded content versions need to be always stored. The remaining space can be allocated to storing transcoded content versions, some of which could be hand-tailored versions to compensate for the lossy nature of transcoding.

4. Which: To decide which content versions should be pre-adapted and stored in the server. Storing versions with rich content will give a greater coverage of future requests while storing versions with less rich content will favor online adaptation since these versions are closer to the desired results.

Section 2 gives an overview of the proposed content adaptation framework and discusses some essential characteristics of the transcoding process. Section 3 presents the requirements for the design of our algorithm. In Section 4, we introduce the transcoding relation graph for mapping the actual application scenario to this representation. The representation is used in Section 5 where we discuss the details of the pre-adaptation selection algorithm. Section 6 addresses the performance of the algorithm, and Section 7 gives some simulation results. Section 8 presents a prototype content adaptation system built based on the proposed framework.

2. A CONTENT ADAPTATION FRAMEWORK

2.1 Content negotiation and realization

There are two modules that make up our proposed content adaptation framework: the *content negotiation module* [19, 20] and the *content realization module*. The content negotiation module consists of a context-aware decision engine that would automatically negotiate for the appropriate adaptation configuration to be used by the content realization module to produce the desired adapted version. The negotiation happens between an internal data structure representing all possible solutions and the requirement coming from the context. The negotiation follows a QoS-sensitive approach which can complement the lossy nature of the transcoding operations. The decision engine will look for the best tradeoff among various parameters in order to reduce the loss of quality in various domains. Scoring methods were suggested to measure the QoS of the content versions in various quality domains, and based on the particular user requirement in these quality domains and other context information on the client's capabilities, the negotiation algorithm can determine a content version with a good aggregate score.

The decision from the content negotiation module is then passed on to the *content realization module* where the real content will be generated. This paper focuses on the design of this module. Our proposed model for this module has a selection logic that can select a subset of "significant" content variants to be pre-adapted in the web server, which could help reduce the overhead of on-the-fly transcoding as well as the transmission overhead. The latter is reduced because the web server no long needs to always send to the proxy the original version of the requested content. The *Similarity Algorithm* in the model will determine the most suitable pre-adapted content version in the web server from which to dynamically generate the desired content version based on the recommendation from the negotiation module. The selected pre-adapted version is either an exact match, or a version closest to the desired version in terms of transcoding steps. The cost-effectiveness of the transcoding process is guaranteed by the Similarity Algorithm. With handshaking happening between the two models, the framework presents a good collaborative environment in which to perform content adaptation.

The various content versions can be synthesized according to settings in different adaptation "axes". An example is the axis of "device", for which the number of possible content variants is proportional to the variety of the device types. With a device-specific authoring approach [1], the introduction of a new device would trigger updating of the content variants. Alternatively, we could view a device as a combination of value settings along such axes as fidelity (quality or rendering requirement) and modality (datatype) [24]. This structure of axes should be extensible so that other related attributes or requirements to define a device or a context can be included. With this approach, the content adaptation procedures can become device-independent.

The settings for synthesizing the desired content version are provided by the decision engine of the content negotiation module prior to the execution of the content adaptation and realization procedures. The decision engine will take into account the different types of context such as device capability profile, networking characteristics, user preference, and metadata about the content itself to provide the necessary input to the transcoding operations that can guarantee the quality of service as outlined in Figure 2.

2.2 Characteristics of transcoding

The decision on a particular content version so generated will be applied to the actual content adaptation procedure to realize the generation of a real content variant. The realization of content requires a series of transcodings towards the desired content. The followings are some of the characteristics of these transcoding processes.

In general, a high-fidelity content variant (with richer content) can be converted into a lower-fidelity variant, but not vice versa. This is true for most images and HTML data [22]. Hence, content versions with a high fidelity can be used to fulfill requests for lower-fidelity contents by going through transcoding. An object from one transcoding may be further transcoded to yield yet a lower-quality object. Each content variant can be transcoded from any one of a subset of higher-quality versions. A *transcoding relation graph* (TRG) [4, 7] can be used to represent the scenario, which will be discussed in Section 4.

The adaptation process is combinatorial [30], consisting of a number of transcoding processes each dealing with a particular feature of the content being adapted. For example, to adapt a 256-color, 100×100-pixel JPEG image to a 2-color, 50×50-pixel GIF image, the process is composed of several atomic transcoding processes such as color-depth reduction, dimensional scaling, and format transformation for the image. The transcoding path through these processes can follow different possible sequences and the same synthesized content variant would result. There has been work [17] to construct a transcoding chain through the pool of available transcoders. The transcoders advertise their capabilities using a simple BNF (Backus Naur Form) grammar, called *capability advertisement* (CA). The system allows type-preserving and type-changing transformations to be combined within the scope of a single intermediary. The approach, however, does not take into account the performance of the transcoders when constructing the chains. An improvement would be to consider performance when there is a choice between two or more equivalent chains.

3. PRE-ADAPTATION REQUIREMENTS

In order to select a subset of significant content variants, *i.e.*, V_{pre}, to be pre-adapted in the web server with consideration of cost factors such as transcoding overhead and spatial consumption, we need to have a set of requirements for the pre-adaptation strategy. The following four requirements are of primary importance.

1. The Sufficiency Requirement
   The selected content variant subset pre-adapted in the web server should be sufficient to synthesize any deliverable content variant that will ever be requested by clients.

2. The Mandatory Pre-adaptation Requirement
   As transcoding involved in dynamic adaptation is lossy in nature, there will be situations where adaptation by the system will be inadequate or even counter-productive [26]. Therefore, some content versions may

Figure 1: The Content Adaptation Framework.

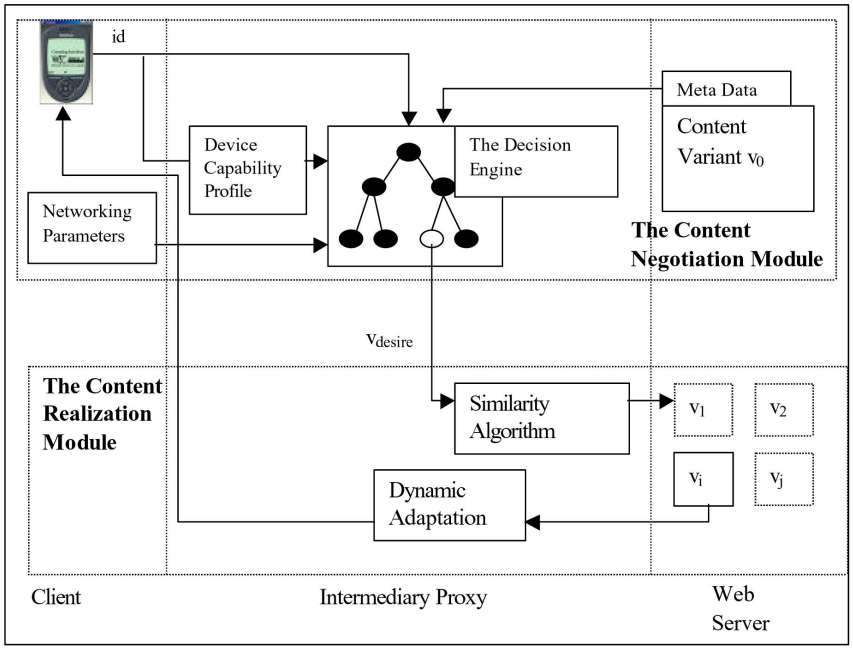
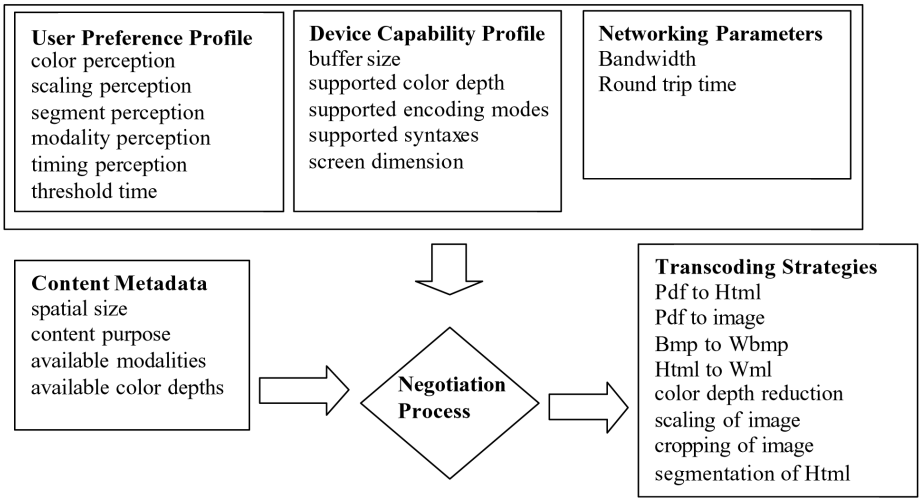


Figure 2: The Negotiation Module.



need to be created manually by the author at content creation time so that the important semantics will not be lost during real-time adaptation. This set of manually pre-adapted versions is indicated as V_{Man} in this paper.

3. The Spatial Boundary Requirement
   The number of pre-adapted content versions should be constrained by the specific capability of the web server. In particular, the space consumption by such contents should not exceed a threshold imposed by the server.

4. The Minimal Cost Requirement
   The time cost needed to generate any deliverable content variant should be minimized with the pre-adapted content variants as the transcoding starting point.

4. TRANSCODING RELATION GRAPH

4.1 Mapping to a graph representation

We use a transcoding relation graph (TRG) to represent the different versions of the same object and the allowed transcoding operations that link them from one to another. The set of vertices $V = \{v_0, v_1, \ldots, v_n\}$ represent the different content variants. Each content variant is different along the axes of fidelity and modality. For example, v_i can be with scaling factor of 75%, color depth of 4 bits and encoded in bitmap while v_j can be with scaling factor of 50%, color depth of 1 bit and encoded in the Wireless Bitmap format. The vertex set corresponds to the set of all the possible content versions that can be recommended by the decision engine. The TRG is a directed graph which consists of the edge set $E = \{E_1, E_2, \ldots, E_m\}$. The edge (v_i, v_j) indicates the feasibility of transcoding from v_i to v_j. Each edge will be associated with a cost to indicate the *time cost* required for going through this particular transcoding process.

To begin, we need to build an initial TRG, TRG_0, by Algorithm 1, such as the one shown in Figure 3. The algorithm takes a set of vertices, representing all possible content variants, as input and connects them with edges. In the figure, the edges have labels which are costs of the corresponding transcoding processes associated with the edges. These costs are added to the graph based on the cost model to be discussed next. Note that this initial TRG is also the final TRG used in a dynamic approach, where any content version, except the original version itself, is generated dynamically on-the-fly.

Algorithm 1 Building an initial TRG.

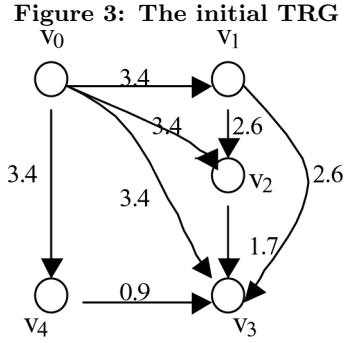
Build-$TRG_0(V, E)$:

$E \leftarrow \emptyset$
if the fidelity level associated with v is larger than or equal to that of u for any $v, u \in V$ $(v \neq u)$
and v is transcodable in modality to u
then $E \leftarrow E \bigcup (v, u)$
return $G(V, E)$

4.2 Transcoding cost model

As soon as the client device submits a request for some content, the whole content delivery process will suffer a number of delays, such as the transmission delay on the slow wireless network or the transcoding time of the adaptation process. As transcoding tends to be time-consuming, it is vital to try to guarantee that the transcoding overhead would not turn into a bottleneck for the whole content delivery process. A better performance in time can be achieved if we apply transcoding to a content variant already pre-adapted in the web server with certain fidelity and modality. Some studies have provided experimental models for the relationship of transcoding time against different factors such as number of pixels (image dimension) and input bytes [14]. They can offer a foundation for the calculation of the transcoding cost.

Figure 3: The initial TRG



A cost model is needed in order to capture the transcoding overhead during the process of dynamic adaptation. The model forms the basis for determining the cost attribute to attach to each edge in the TRG. The cost model should be datatype-specific, and under normal circumstances, a linear cost model can be assumed [4, 7]:

$$t_j = m \times |v_i| + c$$

where t_j is the transcoding time required to synthesize a content version, v_j, using source content version v_i; c is the fixed overhead of synthesizing any content version regardless of content size. m is the transcoding time per unit size of the source content version; $|\ |$ is the size operator (in bytes or dimensional size in area).

This model was applied to image transcoding [14], where experiments revealed that the transcoding time can be approximated by a linear function of the area of the input image. It was suggested that this linear dependency is due to the fact that the largest component of time in image encoding and decoding (in JPEG) is due to the DCT operation. The number of DCT's invoked per image is a direct linear function of the number of blocks, hence the area, of the image.

The cost model is cleanly separable from the TRG. One can feed any cost model to the system to capture the specific transcoding time behavior while the underlying mechanism is kept unchanged.

4.3 Building an intermediate TRG

When each time we add a content variant to the pre-adaptation set, the TRG representation will change accordingly. In keeping with the requirements presented earlier, the change of the graph must be such that

1. each un-pre-adapted content version must be synthesizable from the pre-adapted content versions (Sufficiency); and

2. the transcoding involved in synthesizing any content version will have the least cost among all possibilities (Minimal Cost).

That means that each vertex that does not belong to the pre-adaptation set V_{pre} must be pointed to by an edge from one of the pre-adapted vertices with the least cost (among all possibilities) indicated on the edge. The Build-Graph Procedure (BG) will build up the TRG according to the above guidelines and return a TRG denoted by G'.

In the TRG representation, we indicate a pre-adapted content variant with a filled vertex and a variant not in the pre-adapted set with a hollow vertex. Note that e is a subset of E, and G' a subgraph of G.

Algorithm 2 The Build-Graph (BG) Procedure

BG(V_{pre}):

1. $e \leftarrow \emptyset$
2. **for** $z \in V - V_{pre}$
3. **for** all $(v, z) \in E$ where $v \in V_{pre}$
4. find the minimum $c(v, z)$
5. $e \leftarrow e \bigcup (v, z)$
6. **return** $G'(V, e)$

5. PRE-ADAPTATION SELECTION ALGORITHM

The goal of our work is to develop techniques to satisfy the four primary requirements presented above. We need to devise a *Pre-adaptation Selection Algorithm* with which a subset of the content variants will be identified for pre-adaptation at the web server. The four requirements can be mapped to the TRG representation as follows.

1. Each hollow vertex can be reached by one filled vertex (guaranteed by the BG Algorithm). The untranscoded, original content version, v_0, should be included in V_{pre} so that the Sufficiency Requirement is always satisfied.
2. V_{pre} at the beginning should contain the set V_{Man} to satisfy the Mandatory Pre-adaptation Requirement.
3. We are constrained by the capability of the web server to pre-adapt only a subset of the content variants. The summation of the spatial sizes of the filled vertices should be bounded by $S_{threshold}$ (the Spatial Boundary Requirement).
4. If there are two or more edges that can point to a vertex, we must choose the one with the minimum cost. Obviously, there are many possible final TRG's, but we should aim at the one with the summation of all the edge costs being the minimum among all possibilities (the Minimal Cost Requirement).

This optimization problem is NP-complete—it is a straightforward reduction from the set-cover problem. We therefore look for an approximate solution. One way to solve the problem is by a "greedy" algorithm, where we select a sequence of content versions one after another, each of which is the best choice given what has been selected before cannot be changed. The problem is very much like the minimum spanning tree algorithm [27] in the sense that each pair of nodes is connected by one and only one edge, but they are different in the terminating condition and directionality of edges. Here we need to terminate at any time whenever the spatial threshold is reached, and every edge has a direction. We apply the greedy selection algorithm as suggested in [15] with modifications.

5.1 A greedy selection algorithm

To suit our working scenario, we design a similar greedy-selection algorithm which is a modification of the original algorithm for data cubes [15] with our own selection function having graph building intelligence.

Algorithm 3 The Pre-Adaptation Selection Algorithm

Greedy-Selection(V_{pre}):

1. $V_{pre} \leftarrow V_{Man}$
2. $S_{total} \leftarrow |V_{pre}|$
3. **while** $S_{total} < S_{threshold}$
4. select v not in V_{pre} such that C($v \bigcup V_{pre}$) is minimized
5. $V_{pre} \leftarrow V_{pre} \bigcup v$
6. $S_{total} \leftarrow |V_{pre}|$
7. **return** V_{pre}

The Greedy-Selection Algorithm will select the content versions one by one based on a parameter called the *aggregated transcoding cost*, while keeping to the four requirements during the process. The aggregated transcoding cost, $C()$, is the sum of the transcoding cost associated with each edge in the TRG, as defined by Algorithm 4. $|V|$ stands for the total spatial size of any set V of content variants. The mandatory pre-adapted set V_{man} is first assigned to the set V_{pre}. In each round of the loop (lines 3–6), the unselected v yielding the smallest aggregated cost will be picked and then added to the set V_{pre} until the threshold on spatial size, $S_{threshold}$, is reached. It is important to note that in each round of the selection, the V_{pre} set will be different because of the newly added content variant.

Algorithm 4 The evaluation of the aggregated transcoding cost.

C(V_{pre}):

G' $\leftarrow$ BG(V_{pre})
return $\sum_{(i,z) \in E[G']} c(i, z)$

Take Figure 4 as an example, for which the value of $S_{threshold}$ is set to 850 KBytes. At the beginning, only v_0 is in the pre-adapted set V_{pre}, which is the situation in the figure. The

Figure 4: $V_{pre} = \{v_0\}$.

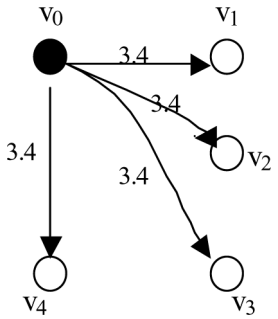
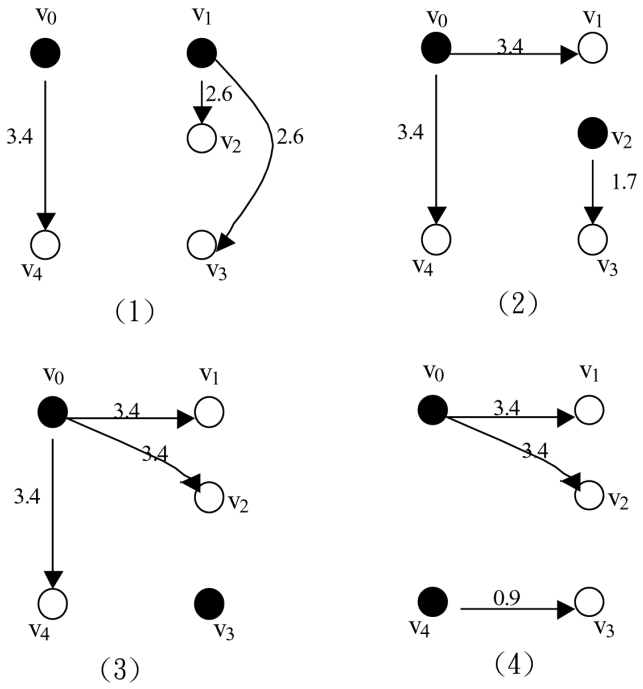


Figure 5: Greedy-Selection at $V_{pre} = \{v_0\}$.



Greedy-Selection Algorithm will select the next vertex that would give the minimum aggregated cost to perform pre-adaptation. The transcoding time is used as the cost. Referring to Figure 5(4) and Table 1, the minimum occurs when $V_{pre} = \{v_0, v_4\}$. So v_4 is picked and added to V_{pre}. S_{total} becomes 600 KBytes which is below the server's threshold. The next round picks v_1 with S_{total} being increased to 850 KBytes. The algorithm stops here, with $V_{pre} = \{v_0, v_4, v_1\}$, since there is not enough space within the threshold to accommodate any more vertices. Figure 6(1) shows the final TRG established.

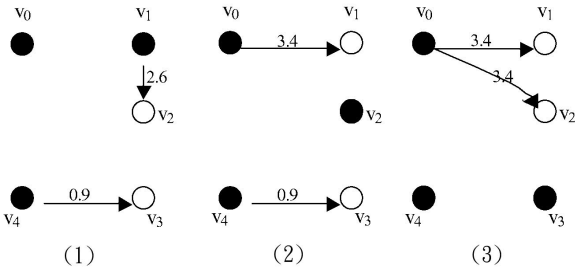
5.2 Taking space into account

The Greedy-Selection Algorithm concerns mainly with the transcoding (time) costs so long as the total space required is within the threshold. We are interested also in a solution that can optimize on both space and time. If two solutions have similar time costs, we will then pick the one that uses less space. To achieve this purpose, we need to extend the basic greedy selection function to take into account the parameter of spatial consumption.

Table 1: The greedy selection order.

V_{pre}	Transcoding cost (sec)	Total Space (KBytes)
v_0	13.6	500
v_0, v_1	8.6	750
v_0, v_2	8.5	700
v_0, v_3	10.2	580
v_0, v_4	7.7	600
v_0, v_4, v_1	3.5	850
v_0, v_4, v_2	4.3	800
v_0, v_4, v_3	6.8	680

Figure 6: Greedy-Selection at $V_{pre} = \{v_0, v_4\}$.



Algorithm 5 The evaluation of the aggregated transcoding cost saving per unit space.

C'(V_{pre}):

G' ← BG(V_{pre})
return $\frac{Cost_{previous} - \sum_{(i,z) \in E[G']} c(i,z)}{|V_{pre}|}$

We define a new greedy selection function, denoted by $C'()$. This time, the corresponding algorithm will select the content variant with maximum aggregated transcoding cost saving per unit spatial consumption instead of the minimized aggregated transcoding cost as used in the previous model. $Cost_{previous}$ is the aggregated trancoding cost in the previous round of the greedy selection. We need to replace the $C()$ function in line 4 of the Greedy-Selection Algorithm by $C'()$ if we are to adopt this space-and-time approach.

We refer to this later approach based on $C'()$ as the *space-time approach*, and the previous one based on $C()$ the *time-only approach*.

5.3 The Similarity Algorithm

From the execution of the pre-adaptation selection algorithm, we obtain a final TRG, G_{final}, which will be used by the web server to carry out the actual pre-adaptation.

During real-time processing, the negotiation module will determine an optimal content version based on the client's context. This decision is then fed to the Similarity Algorithm in the realization module whereby the most transcod-

ing cost-effective content pre-adapted in the web server will be identified and from which on-the-fly dynamic adaptation is applied to synthesize the desired content. The Similarity Algorithm is a simple graph exploration algorithm that finds the source of the edge with the desired content version in question as the destination in the TRG. The cost-effectiveness is already guaranteed by the property of the TRG.

Algorithm 6 The Similarity Algorithm

Similarity(v_{desire}):

find $(v, v_{desire}) \in E[G_{final}]$
return v

Take for example Figure 6(1) where there is a final TRG as discussed in the previous section. Given a request for content variant v_3 which is a WBMP image for WAP devices with a small display, the Similarity Algorithm will find v_4 which is a two-color bitmap with small dimensional size. Hence, v_4 will act as the starting point for the generation of content variant v_3.

6. PERFORMANCE GUARANTEE

To see how well the greedy algorithm as an approximation to the optimal solution performs, we measure the improvement on transcoding time it can achieve through pre-adaptation. "Improvement" is defined as the reduction in transcoding time over all content versions. We can show that the improvement with using the greedy algorithm is at least 63% of the improvement by the optimal algorithm in most cases. The precise fraction is $\frac{e-1}{e}$, where e is the base of the natural logarithm. The proof for the greedy-selection algorithm in [15] operates with a fixed number of "views", meaning that both the optimal and greedy algorithm will produce k views regardless of the space used. But in our case, the space consumption due to the set of content versions to be pre-adapted is under a fixed constraint (the space threshold), and so the number of content versions could be different from solution to solution. We borrow the idea of the original proof of the ratio bound, but need to make modifications to reflect the difference.

We define B to be the improvement in transcoding time with using the optimal solution, and A that with the greedy solution. The ratio is found to be as follows.

$$\frac{\mathrm{A}}{\mathrm{B}} \geq 1 - (\frac{k'-1}{k'})^k$$

where k is the number of content versions selected by the greedy approach and k' is that by the optimal solution. For example, if $k = 9$ and $k' = 8$, we can say then that the greedy algorithm is at least 70% of the optimal solution in performance. The detailed proof can be found in [21].

7. EXPERIMENTATION

In this section, we report evaluation of the performance of our proposed algorithm through simulation experiments. The main objective of the simulation is to study the impact on the transcoding overhead of the strategy of mixing pre-adaptation with dynamic adaptation in content realization. Given a pre-adaptation constraint imposed by the content provider, we study the dynamic transcoding time saved by our proposed approach. The experimental results shed light on how to trade space used for a meaningful saving on content delivery time.

7.1 The simulation model

In the simulation model, we use a real data set with content variants from a PDF document adaptation system which we will discuss in more detail in the next section. The input data for the simulation model include the file size, the modality, and fidelity information of the content objects handled by the PDF system. The transcoding rate is defaulted to 20KB/s, as is used in [4, 7], and we assume a linear cost model.

7.2 Some results

Based on the output data from the simulation, we study the following aspects using various measures.

1. We use *aggregated transcoding cost saving* as the main performance measure. It is derived from the difference between the data obtained for the initial TRG (*i.e.*, pure dynamic adaptation approach) and those for the intermediate or final TRG with pre-adapted versions.

2. We use the spatial consumption of the pre-adapted contents and the *pre-adaptation capacity* as the performance measures along the dimension of storage cost. The pre-adaptation capacity is the ratio between the desired spatial consumption applicable to the content server and the summation of the sizes of all the deliverable objects to the clients—*i.e.*,

$$\text{pre-adaptation capacity} = \frac{\text{Desired spatial consumption}}{\sum \text{spatial consumption of all deliverable objects}}$$

   Experiments with variation in this capacity ratio reveal how the cost saving in transcoding overhead is affected by the ratio.

3. Another performance metric is the *coverage degree* which is defined as the ratio of the total number of the selected pre-adapted content variants to the total number of the deliverable content variants. This indicates how well a given spatial consumption in the content server can cover the number of content versions. The higher the degree, the greater would be the chance of finding a pre-adapted version that can serve as the base for transcoding or even without transcoding. A value of 1 for the coverage degree means that no extra dynamic real-time transcoding is required at all to synthesize the content variants as the space used can cover all the deliverable content variants.

Figure 7 shows the aggregated transcoding cost saving against each round of the greedy selection for the first 120 rounds. During each round, a pre-adapted content variant will be added, which invariably increases the saving on the transcoding cost. At the initial stages of the selection, the cost saving grows significantly, meaning that each newly inserted content variant has a major positive impact on the transcoding overhead. After a while the saving becomes more steady and the increase is less dramatic. The most cost-effective rounds of selection therefore are those at the

Figure 7: Aggregated transcoding cost saving in sec at each round of selection.

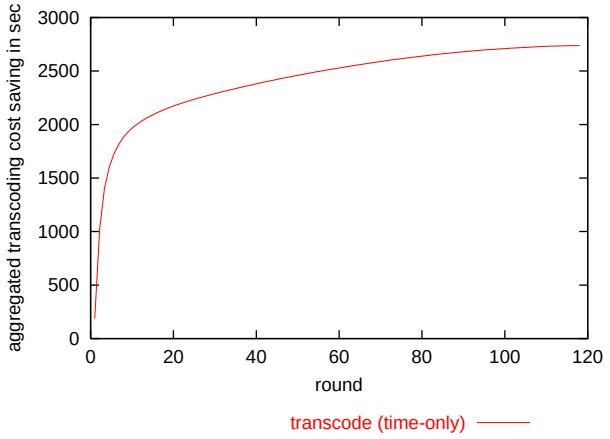


Figure 8: Spatial consumption in KB against aggregated transcoding cost saving.

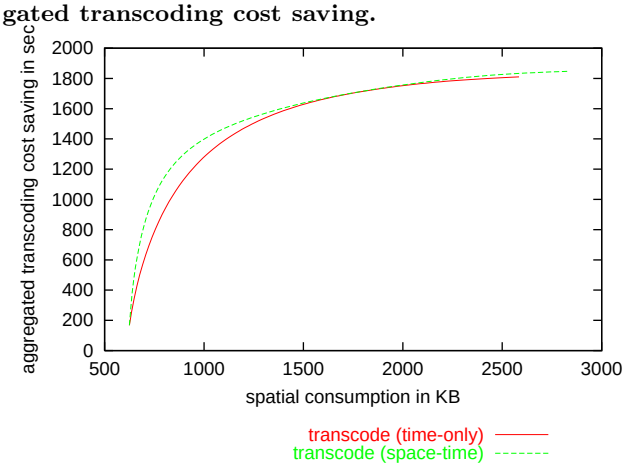


Figure 9: Pre-adaptation capacity against aggregated transcoding cost saving.

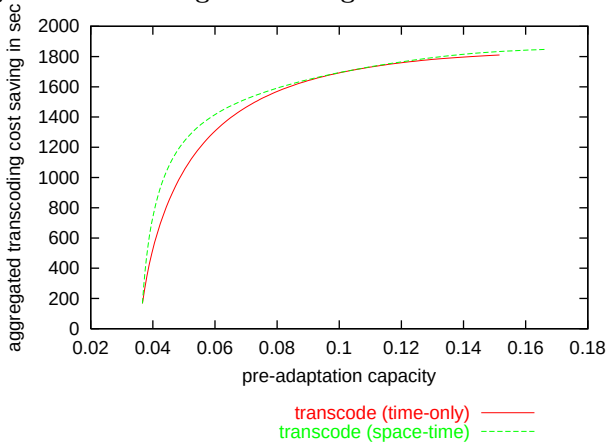
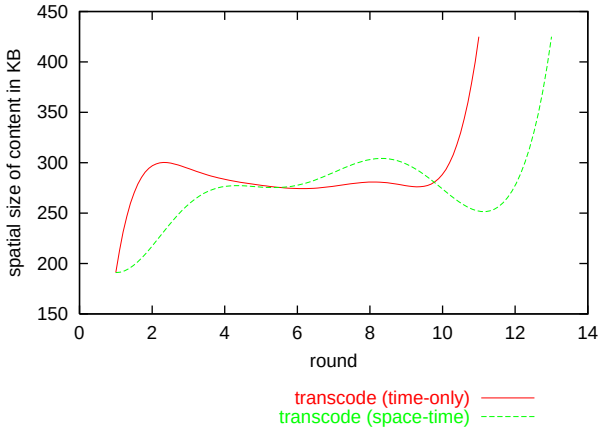


Figure 10: Spatial size of content selected at each round.



initial stages. If we stop at some point during these initial rounds, the space consumption will be minimal. Hence, it is worth focusing on this region of the graph.

Each round of the greedy selection corresponds to a certain increase in the spatial consumption. Figure 8 shows the relationship betweeen spatial consumption and transcoding time saving.

We can modify the simulation model a little by using the pre-adaptation capacity as the spatial constraint, which may be more useful for the content provider when doing resource planning. For instance, a constraint of pre-adaptation capacity of 0.1 means one-tenth in total size of the web objects are allowed to be pre-adapted. Figure 9 shows the relationship between this pre-adaptation capacity and the time saving. For this one and some of the subsequent simulations, we tested both the time-only approach (based on $C()$) and the space-time approach (based on $C'()$, as discussed in Section 5.2). The former is represented by a dashed line, and the latter by a dotted line in the graphs.

In both Figures 8 and 9, we can see that the space-time approach outperforms the time-only approach that considers only the transcoding time. This can be easily explained as the space-time selection strategy would favor those content variants that are can produce more benefits in time but use the same amount of space (or less) as some others.

We can also look at the performance the other way round—that is, given a desired spatial consumption or capacity ratio, we would like to know how well the space is being utilized—*i.e.*, the coverage degree. As shown in Figure 11, the coverage degree increases with the spatial cost. We can observe in Figure 10 that initially, contents with larger spatial consumption are more easily picked by the basic $C()$ function since these versions are probably richer in content and therefore can yield a more significant transcoding cost saving over all the content versions. A threshold on the pre-adaptation capacity is set to 0.15 for the simulation in Figure 10. For this particular capacity ratio, the $C'()$ function can pack in a few more content variants. As a result, the space-time approach performs better than the time-only approach, as shown in Figure 11.

8. A PDF DOCUMENT CONTENT ADAPTATION SYSTEM

We have implemented the proposed greedy algorithm in a PDF document content adaptation system. Currently, the system is aware of the user context in five quality domains: color, downloading time, scaling, modality, and segmenta-

Figure 11: Spatial consumption against content coverage ratio.

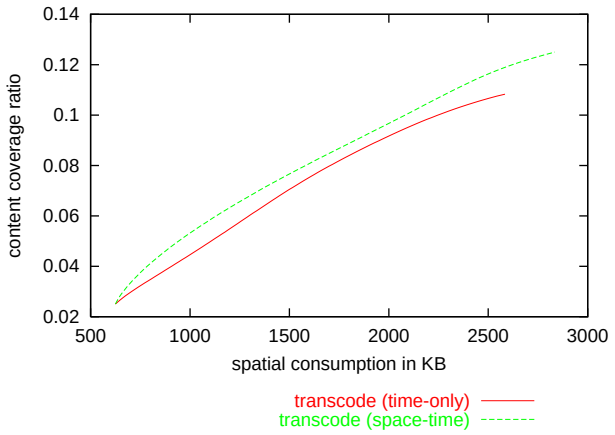
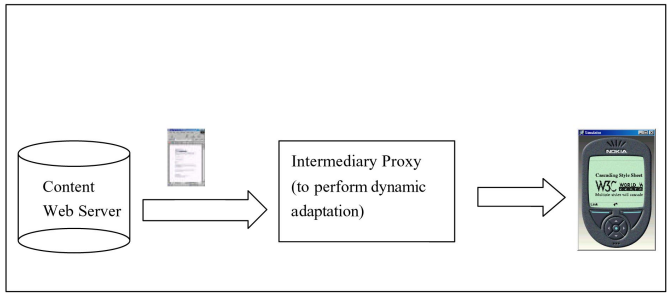


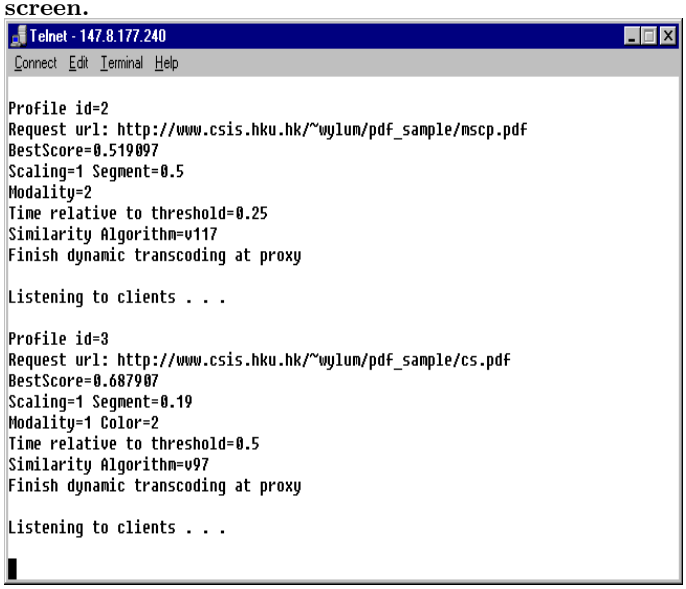
Figure 12: The content generation process.



tion. The modality domain corresponds to how the user feels about preserving the mode or accepting transcoding to a different mode. The segmentation domain corresponds to how the user feels about cropping of the content. The user can also specify the maximum downloading time he/she can tolerate for the content delivery. For the context of device capability, we use information for some commercial portable device models, and the system will take into account the screen size, the supported color depth, media type, markup language and the memory buffer size (*e.g.*, the maximum deck size acceptable for the WAP phone). For the network context, a set of parameters such as bandwidth and round-trip time of some current popular communication channels (*e.g.*, CDMA, GPRS, CDPD, etc.) can be applied to our system so that the system can adapt to the networking situation as well. Figure 2 in Section 2 shows a set of these context parameters being divided into different categories.

After receiving the decision on the optimal content version from the content negotiation module, the content realization module is set in motion where various transcoding operations will be invoked in order to synthesize the desired content. Some of these technical transcoding operations can also be found in Figure 2. The Similarity Algorithm will determine the most cost-effective content version that has been pre-adapted in the web server and make a request for it. For the current system, we set the pre-adaptation capacity to 0.1 so that only one-tenth in size of all the deliverable content variants would be pre-adapted and become available on the

Figure 13: Useful messages appearing on the proxy's screen.



web server. Having been sent the requested pre-adapted version, the intermediary proxy will carry out real-time dynamic adaptation to synthesize the desired content to be returned to the client. An example in shown in Figure 12 where the desired content version is determined by the negotiation module to be one in WBMP format with appropriate scaling and segmenting. The Similarity Algorithm in the intermediary proxy then selects a two-color BMP version to be the most transcoding cost-effective content version in the web server. This version is sent by the web server to the intermediary proxy for processing to generate the desired content version. The generation trace will be displayed at the intermediary proxy, as shown in Figure 13, to show the steps during content realization, together with the URL of the web server, the modality and fidelities of the desired content variant determined by the negotiation module, and the closest matching content variant in the web server selected by the Similarity Algorithm.

With the cooperation between the negotiation module with QoS sensitivity and the content realization module equipped selection algorithms to optimize on spatial and transcoding costs, the most appropriate content version is returned to the client device in the end. By varying the various settings of the context parameters serving as input to the negotiation module, different content variants will result from the content realization module. Figure 14 shows the sample deliverables of the same PDF document to a WAP device and a PDA using our system. The difference in the presentation is due to the difference of user preference for the two conflicting parameters, the downloading time and the modality preservation of the negotiation module. As shown in Figure 14 (a) to (e), different versions of the requested object, image in WBMP, text in WML, PDF, image in BMP, and text in HTML, emerge as a result of the dynamic adaptation of the corresponding pre-adapted content versions determined by the Similarity Algorithm in the intermediary proxy

The user can also specify the maximum downloading time

Figure 14: Change of modalities due to capability of devices and user preference.

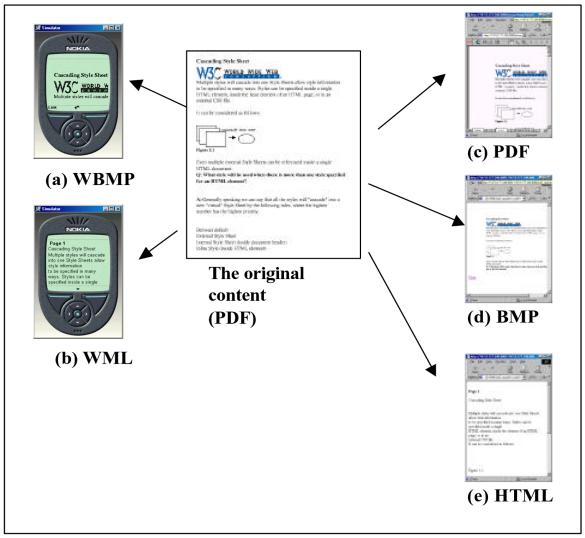


Figure 15: Sample deliverables vs. change of maximum tolerable downloading time.

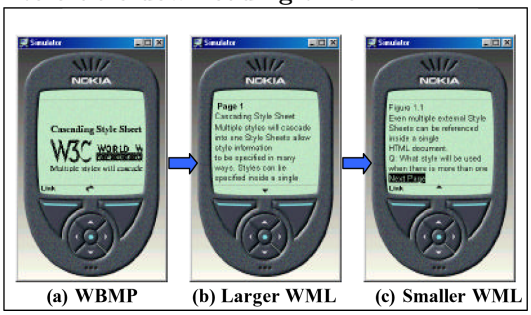
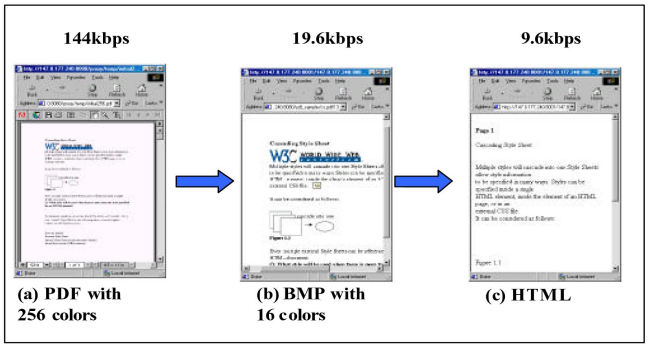


Figure 16: Adaptability of the system to the networking context.



he/she feels is tolerable and the system will generate the appropriate, cost-effective content version automatically, as shown in Figure 15. The adaptability of the system to the networking context (mainly the bandwidth of the connection) is shown in Figure 16.

9. CONCLUSION AND FUTURE WORK

This paper presents a strategy for realizing the actual content in response to the decision given by the decision engine in the negotiation module. The proposed framework can guarantee the cost-effectiveness of the approach in terms of real-time transcoding overhead and the storage space used to accommodate the various pre-adapted content variants. We presented a brief summary of the experimental results which confirm the viability of the pre-adaptation strategy. A content adaptation system for PDF documents has been implemented to show the collaborative operations between the negotiation decision engine and the content realization module, and the results of collaboration.

The pre-adaptation selection presented here has assumed that all the content variants are equally popular. We can further propose a variation of the algorithm in which different content variants may be assigned different popularities. Knowledge such as poplarity of certain mobile device and user patterns of content subscription can be used in the popularity assignment. For example, Palm PDA's, which enjoy a fairly large market share, all ask for 320×240 4-bit content, and so the corresponding content variants having this setting can be assigned a higher popularity. The idea is quite similar to the PBS-U Algorithm (Pick-By-Size) proposed in [29] whereby the selection of a view to be materialized in a database is determined by its size weighted by probability of occurrence. Further research can be developed in this direction.

10. REFERENCES

[1] T.W. Bickmore and B.N. Schilit, "Digestor: Device-independent Access to the World Wide Web", in Proceedings of the 6th World Wide Web Conference (WWW6), Apr. 1997, pp. 655–663.

[2] S. Boll, W. Klas, and J. Wanden, "A Cross-Media Adaptation Strategy for Multimedia Presentation", in Proceedings of the ACM Multimedia '99, Orlando, Florida, USA, Oct. 1999.

[3] K.H. Britton, R. Case, A. Citron, R. Floyd, Y. Li, C. Seekamp, B. Topol, and K. Tracey, "Transcoding: Extending e-business to new environments", IBM Systems Journal, vol. 40, no. 1, 2001.

[4] V. Cardellini, P.S. Yu, and Y.W. Huang, "Collaborative Proxy System for Distributed Web Content Transcoding", in Proceedings of the 2000 ACM CIKM International Conference on Information and Knowledge Management (CIKM 2000), McLean, VA, USA, Nov. 2000, pp. 520–527.

[5] S. Chandra, C.S. Ellis, and A. Vahdat, "Differentiated Multimedia Web Services Using Quality Aware Transcoding", 19th Annual Joint Conference of the IEEE Computer and Communication Societies (INFOCOM 2000), Tel Aviv, Israel, Mar. 2000, pp. 961–969.

[6] S. Chandra and C.S. Ellis, "JPEG Compression Metric as a Quality Aware Image Transcoding", In

2nd Usenix Symposium on Internet Technologies and Systems (USITS '99), Oct. 1999.

[7] C.Y. Chang and M.S. Chen, "Exploring Aggregate Effect with Weighted Transcoding Graphs for Efficient Cache Replacement in Transcoding Proxies", 18th International Conference on Data Engineering (ICDE 2002), San Jose, California, Feb. 2002.

[8] R. Comerford, "Handhelds duke it out for the Internet", IEEE Spectrum, Aug 2000, pp. 35–41.

[9] A.K. Dey and G.D. Abowd, "Towards a Better Understanding of Context and Context-Awareness", 1st International Symposium on Handheld and Ubiquitous Computing (HUC'99), Karlsruhe, Germany, Sept. 1999.

[10] E-Bay.com, http://www.ebay.com

[11] A. Fox, I. Goldberg, S. Gribble, D.C. Lee, A. Polito, and E.A. Brewer, "Experience with TopGun Wingman: A Proxy-Based Web Browser for the 3Com PalmPilot", in Proceedings of Middleware '98, Lake District, England, Sept. 1998.

[12] A. Fox, S.D. Gribble, Y. Chawathe, and E.A. Brewer, "Adapting to network and client variation via On-Demand Distillation", in Proceedings of the 16th Symposium on Operating Systems Principles (SOSP-16), Oct. 1997.

[13] R. Han, "Factoring a Mobile Clients Effective Processing Speed Into the Image Transcoding Decision", ACM Workshop on Wireless Mobile Multimedia (WOWMOM), 1999, pp. 91–98.

[14] R. Han, P. Bhagwat, R. LaMaire, T. Mummert, V. Perret, and J. Rubas, "Dynamic Adaptation in an Image Transcoding Proxy for Mobile Web Browsing", IEEE Personal Comm., Dec. 1998, pp. 8–17.

[15] V. Harinarayan, A. Rajaraman, and J.D. Ullman, "Implementing data cubes efficiently", in Proceedings of ACM SIGMOD 1996, Montreal, June 1996, pp. 205–216.

[16] M. Hori, K. Ono, G. Kondoh, and S. Singhal, "Authoring tool for Web content transcoding", in Proceedings of Markup Technologies '99, Philadelphia, PA, 1999, pp. 77–85.

[17] S. C. Ihde, P.P. Maglio, J. Meyer, and R. Barrett, "Intermediary-based transcoding framework", in Poster Proceedings of the 9th International World Wide Web Conference (WWW9), 2000.

[18] Z. Lei and N.D. Georganas, "Context-based Media Adaptation in Pervasive Computing", in Proceedings of Can.Conf. on Electr. and Comp. Engg (CCECE 2001), Toronto, May 2001.

[19] W.Y. Lum and F.C.M. Lau, "A QoS Sensitive Content Adaptation System for Mobile Computing", COMPSAC 2002, Oxford, U.K., August 2002, to appear.

[20] W.Y. Lum and F.C.M. Lau, "A Context-Aware Decision Engine for Content Adaptation", IEEE Pervasive Computing, to appear.

[21] W.Y. Lum, F.C.M. Lau, Effective Content Adaptation: Response Time and Transcoding Overhead vs Spatial Consumption, tech. report TR-2002-02, Dept. of Computer Science and Information Systems, The University of Hong Kong, Hong Kong, 2002.

[22] A. Maheshwari, A. Sharma, K. Ramamritham, and P. Shenoy, "TranSquid: Transcoding and Caching Proxy for Heterogeneous E-Commerce Environments", in Proceedings of the 12th IEEE Workshop on Research Issues in Data Engineering (RIDE '02), San Jose, CA, Feb. 2002.

[23] J.C. Mogul, "Server-Directed Transcoding", Computer Communications 24(2):155-162, Feb. 2001.

[24] R. Mohan, J.R. Smith, and C.-S. Li, "Adapting multimedia Internet content for universal access", IEEE Trans. Multimedia, vol. 1, no. 1, Mar 1999, , pp. 104–114.

[25] K. Nagao, "Semantic transcoding: Making the World Wide Web more understandable and usable with external annotations", Proc. Int'l Conf. on Advanced in Infrastructure for Electronic Business, Science, and Education on the Internet, L'Aqila, Italy, July 2000.

[26] B.D. Noble, M. Price, and M. Satyanarayanan, "A Programming Interface for Application-aware Adaptation in Mobile Computing", Proc. 2nd USENIX Symposium on Mobile and Location-Independent Computing, Ann Arbor, MI, USA, Apr. 1995.

[27] C. Papadimitriou and K. Steiglitz, Combinatorial Optimizations, Algorithms and Complexity, Englewood Cliffs, N.J.: Prentice Hall, 1982.

[28] C. Parris, B. Dennis, "Transformation Proxy Support for Thin-Clients", http://citeseer.nj.nec.com/58739.html

[29] Deshpande, J.F. Naughton, "Materialized View Selection for Multidimensional Datasets", 24th Int'l Conf. on Very Large Database (VLDB 1998), New York City, USA, Aug. 1998, pp. 488–499.

[30] J.R. Smith, R. Mohan, and C-S. Li, "Scalable multimedia delivery for pervasive computing", Proc. ACM Multimedia '99, Orlando, Florida, Oct. 1999, pp. 131–140.

[31] Yahoo.com, http://www.yahoo.com

[32] S.S. Yau and F. Karim, "Context-Sensitive Object Request Broker for Ubiquitous Computing Environments", Proc. 8th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS 2001), Bologna, Italy, Oct. 2001.

[33] S.S. Yau and F. Karim, "Context-Sensitive Distributed Software Development for Ubiquitous Computing Environments", Proc. 25th IEEE International Computer Software and Applications Conference (COMPSAC 2001), Chicago, USA, 8-12 Oct. 2001.

[34] S.S. Yau and F. Karim, "Context-Sensitive Middleware for Real-time Software in Ubiquitous Computing Environments", Proc. 4th IEEE International Symposium on Object-Oriented Real-time Distributed Computing (ISORC 2001), Magdeburg, Germany, 2-4 May 2001, pp. 163–170.

[35] C. Yoshikawa, B. Chun, P. Eastham, A. Vahdat, T. Anderson, and D. Culler, "Using smart clients to build scalable services", Proc. Winter 1997 USENIX Technical Conf., Jan. 1997.