THE UTILITY MODEL FOR ADAPTIVE MULTIMEDIA SYSTEMS

SHAHADAT KHAN, KIN F. LI
Department of Electrical and Computer Engineering

ERIC G. MANNING
Departments of ECE and Computer Science
University of Victoria
Victoria, BC, Canada V8W 3P6
E-mail: {skhan,kinli,emanning}@ece.uvic.ca

We present the *Utility Model* — a mathematical model to incorporate the dynamics of adaptive multimedia systems (AMSs). It is based on the concepts of quality profile, quality-resource mapping, session and system utility, and of system resource constraints. The main problem in a multi-session AMS is to find an operating quality for each session such that the overall system utility is maximized under the system resource constraints. This is called the adaptive multimedia problem (AMP). The Utility model formulates the AMP as a dynamic optimization problem, and maps it to a multiple-choice multiconstraint 0-1 knapsack problem. This work provides a unified approach to both admission control of new multimedia sessions and the dynamic quality adaptation of existing sessions. It also provides a computationally feasible real-time method to optimize the utility (e.g., revenue) of real multimedia systems.

1 Introduction

Multimedia applications require system support with quality of service (*QoS*) guarantees. For instance, in order to support a video stream at a refresh rate of 10 frame/sec, the stream handler has to be scheduled onto the CPU once every 0.1 second. In a multimedia system, QoS may arise at many levels: *perceived QoS* of a session user such as good, okay or choppy; *media QoS* such as video refresh rate and resolution; *system QoS* such as latency and throughput; and *network and device QoS* such as connection bandwidth and loss rate, and audio device sampling rate [1].

In a dynamic system environment, multimedia applications are potentially adaptive. For example, one can have real-time video transmitted at 30 frame/sec refresh rate when there is enough bandwidth available; or one may reduce the bandwidth requirement to half by reducing the rate to 15 frame/sec. How do we design a multimedia system to exploit the potential for adaptation of multimedia applications?

In this article, we present a mathematical model, called the Utility Model, to encompass the dynamic factors and forces within a multi-session *adaptive* *multimedia system* (AMS), where the quality of the multimedia sessions is dynamically adapted according to the system state at run-time.

The rest of the article is organized as follows. Section 2 introduces adaptive multimedia systems. Section 3 explains the concepts of the Utility Model and formulates the adaptive multimedia problem (AMP) as a dynamic optimization problem. Section 4 maps the AMP to a multiple-choice multiconstraint knapsack problem (MMKP), and presents our experience with a heuristic solution of this problem. Section 5 presents three system resource allocation policies. Finally section 6 concludes the article with suggestions for further research.

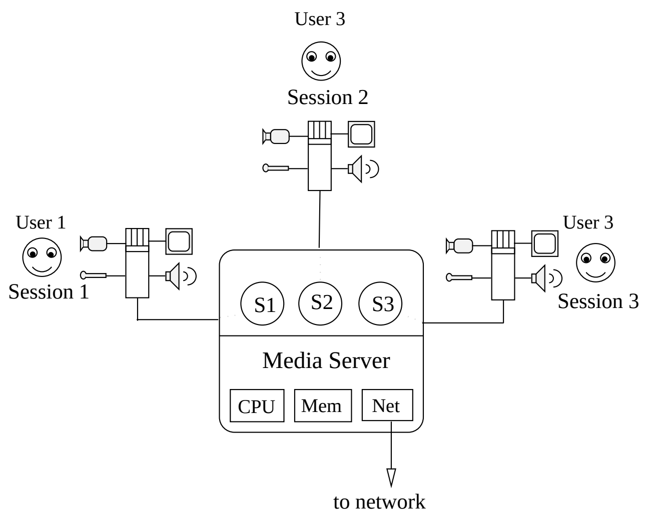


Figure 1: A multi-user system with a multi-session media server.

2 Adaptive Multimedia Systems

Let us consider a multi-user multimedia system configuration as shown in Figure 1. We assume a small group of users, such as a family or a research group. Each user uses her own multimedia terminal for media recording and playout, and contacts the media server for the processing, storage and transmission of media. At any time, the media server may have to support multiple concurrent sessions from multiple users.

Suppose user 1 is participating in a video conference, user 2 is watching a movie, and user 3 is wandering around in a virtual entertainment park. At times, the resources at the media server may not be sufficient to provide the desired quality of media to all sessions, for example because the CPU is overloaded, or the network is congested. How should this system adapt to these dynamic situations? When sufficient resources are available, the system should provide every user the desired quality. However, when available resources cannot support the desired quality, the system may compromise the quality. For video, this may mean reduced refresh rate or reduced resolution, and for audio it may mean mono quality instead of stereo. Thus in Figure 1, the server should be able to adapt the QoS of an individual session, based on policies concerning relative importance and other constraints. For example, when the higher priority video-conference session needs better quality, compromising the quality of the cyberspace session is an option.

In a multi-session system with dynamically changing users' demands and dynamically changing amounts of unallocated resources – resource availability – the problem of determining the QoS of individual sessions, in order to maximize some objective function, subject to a set of system resource constraints, is called the *adaptive multimedia problem (AMP)*.

Adaptation in a multimedia system may be triggered by many events, such as change of system load, change of network load or congestion/failure in the network, and dynamic change of users' requirements. An AMS should provide an adaptive resource management scheme in order to make best use of the system resources. Effective solution of the AMP involves many factors such as the system objective, the relative importance of sessions, the relative importance and interrelations of media within a session, and users' requirements/constraints.

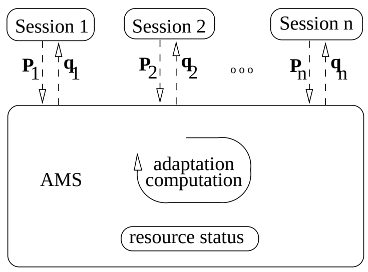


Figure 2: Adaptive multimedia system requirements.

2.1 AMS Requirements

The quality perceived by the user of a multimedia session depends on the quality of individual media components, and this in turn depends on such things as video refresh rate, number of sound channels (mono, stereo or surround), and the relation among media components such as video-audio synchronization. In an AMS, the QoS of individual sessions is dynamically adapted in order to maximize some objective function (e.g., total revenue earned), also taking into account the users' preferences and the system resource status. Some of the requirements of an AMS are as follows [Figure 2]:

- The user of a session should be able to specify an adaptive quality requirement or preference. This is done using a *quality profile* ($\mathbf{P}_i$) defined below.
- The system should dynamically adapt the *operating quality* ($\mathbf{q}_i$) of each session based on the quality profile and the system resource status. To do this, the system must know the dynamic resource status of the system.

3 The Utility Model

The Utility Model is a mathematical model for AMS based on the concepts of quality profile, quality-resource mapping, session and system utility, and system resource constraints. These concepts are explained in the following subsections.

3.1 Quality Profile

The quality profile specifies the quality preferences of the user of a particular session. For simplicity, we consider three media: audio (a), video (v) and image (i). The *operating quality* $\mathbf{q}_i$ of session i is the vector (3-tuple)[a] of individual media qualities, i.e.,

$$\mathbf{q}_i = (q_{ia}, q_{iv}, q_{ii}). \tag{1}$$

The *quality profile* of a session is a sequence of acceptable operating qualities in increasing order of preference (from minimum acceptable quality to highest desired quality). Mathematically the quality profile of session i can be expressed as a vector,

$$\mathbf{P}_i = (\mathbf{q}_{i1}, \mathbf{q}_{i2}, \mathbf{q}_{i3}, \ldots, \mathbf{q}_{il_i}), \tag{2}$$

[a] We use boldface symbols to express vector quantities and non-boldface symbols to express scaler quantities.

where l_i is the number of qualities in $\mathbf{P}_i$, $\mathbf{q}_{i1}$ is the minimum acceptable quality and $\mathbf{q}_{il_i}$ is the highest desired quality. For example, Figure 3 shows a session with quality profile $\mathbf{P}_i = (\mathbf{q}_{i1}, \mathbf{q}_{i2}, \mathbf{q}_{i3}, \mathbf{q}_{i4})$ where $\mathbf{q}_{i1}$ indicates low-resolution video and mono audio, $\mathbf{q}_{i2}$ indicates low-resolution video and stereo audio, $\mathbf{q}_{i3}$ indicates medium-resolution video and stereo audio, and $\mathbf{q}_{i4}$ indicates high-resolution video and surround sound audio.

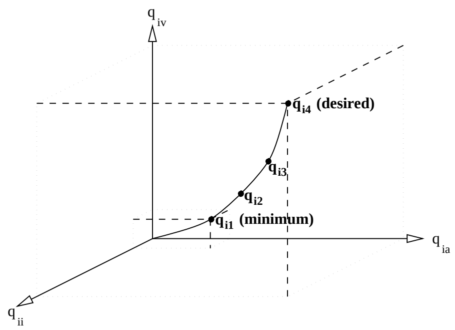


Figure 3: Quality profile of a session.

3.2 Quality-resource Mapping

We assume the existence of a mapping from an operating quality to the resources required to provide that quality. This is called the *quality-resource mapping.* In this model, we assume that a media QoS may be mapped uniquely to the required resources[b]. We note that, *unique* does not necessarily imply static. Let us assume only three resources within a system: processor cycles (p), main memory (m), and network bandwidth (b). Then the required resources $\mathbf{r}_i$ of session i can be expressed by a three-tuple,

$$\mathbf{r}_i = (p_i, m_i, b_i) \tag{3}$$

[b]However, in practice, quality-resource mappings may not be unique. For example, if we use a simple compression algorithm a certain perceived QoS may require 10% of the total CPU cycles and 3Mbps of network bandwidth. On the other hand, if we use a more sophisticated compression algorithm, the same perceived QoS may require 40% of CPU cycles and 1Mbps of network bandwidth. We note the processor-bandwidth trade-off depending on the choice of compression algorithm. In such cases, we may achieve uniqueness by fixing the choice of compression algorithm, or by arbitrarily choosing one of several resource values.

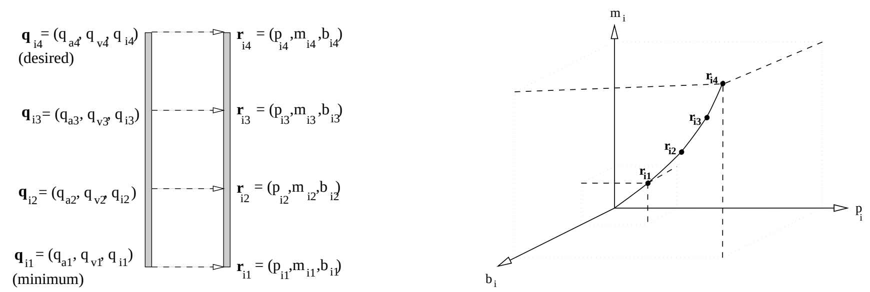
where

$$
\begin{aligned}
p_i &= p(\mathbf{q}_i) = p(q_{ia}) + p(q_{iv}) + p(q_{ii}), \\
m_i &= m(\mathbf{q}_i) = m(q_{ia}) + m(q_{iv}) + m(q_{ii}), \\
b_i &= b(\mathbf{q}_i) = b(q_{ia}) + b(q_{iv}) + b(q_{ii}).
\end{aligned}
$$

Here $p(\cdot)$, $m(\cdot)$ and $b(\cdot)$ are quality-to-resource mapping operators for processor, memory and bandwidth resources respectively. In vector notation, the quality-resource mapping can be expressed as

$$\mathbf{r}_i = \mathbf{r}(\mathbf{q}_i). \tag{4}$$

The quality-resource mapping is illustrated in Figure 4. Figure 4 (a) shows how each operating quality of the quality profile is mapped to the required resource vector. For example, operating quality $\mathbf{q}_{i1}$ would require resource vector $\mathbf{r}_{i1}$, i.e. $\mathbf{r}_{i1} = \mathbf{r}(\mathbf{q}_{i1})$. The quality-resource mapping transforms the quality profile to the session resource profile, which is a list of required resources for different choices of operating qualities. The resulting *resource profile* is illustrated in Figure 4 (b).



(a) Quality-resource mapping. (b) Resource profile.

Figure 4: Mapping operating quality to required resources, and resource profile.

3.3 Session and System Utility

Any adaptive system must have an objective, which decides the direction of adaptation in any given situation. We assume that the objective of an AMS can be expressed as a *system utility objective*, and session i's utility can be obtained from its operating quality $\mathbf{q}_i$ by using the *session utility function* $u_i(\cdot)$.

For simplicity, we assume that system utility objective is to maximize the system utility function U given by

$$U = \sum_{i=1}^{n} u_i(\mathbf{q}_i), \tag{5}$$

a linear function of the session utilities. We note that the Utility Model is not restricted to the above system utility function. More generally, the system utility function defines the objective of the adaptive system, and may be derived from a set of resource usage policies based on issues such as revenue, fairness and priority. For example, for a multimedia service provider, a natural choice for the system utility objective would be to maximize revenue, where each session pays for the service it receives; for a friendly research lab, the goal might be to provide fair shares of resources to the sessions; whereas for a military system, the goal may be to provide a strict priority-based scheme where the generals' requests receive the highest priority and the private soldiers' requests receive the lowest priority.

3.4 System Resource Constraints

At any moment, the system state has to obey the following *system resource constraints*: for each resource, the sum of the quantities of the resource allocated to all the sessions cannot exceed the total available quantity of the resource. Suppose the available system resources are expressed as a vector $\mathbf{R} = (P, M, B)$. The resource constraints are expressed as

$$\sum_{i=1}^{n} \mathbf{r}(\mathbf{q}_i) \leq \mathbf{R}. \tag{6}$$

3.5 The Adaptive Multimedia Problem

The main concepts of the Utility Model are illustrated in Figure 5:

1. Each user specifies a quality profile which is the set of acceptable operating qualities for the session.
2. A session's operating qualities are assumed to be mapped uniquely to required resources.
3. The system is subject to the system resource constraints.
4. A session's operating qualities are assumed to be mapped uniquely to session utility.

5. The system utility is the sum of all session utilities.

Now the AMP is to *find* the operating quality $\mathbf{q}_i$ of each session i which *maximizes* the system utility U under the system *resource constraints*.

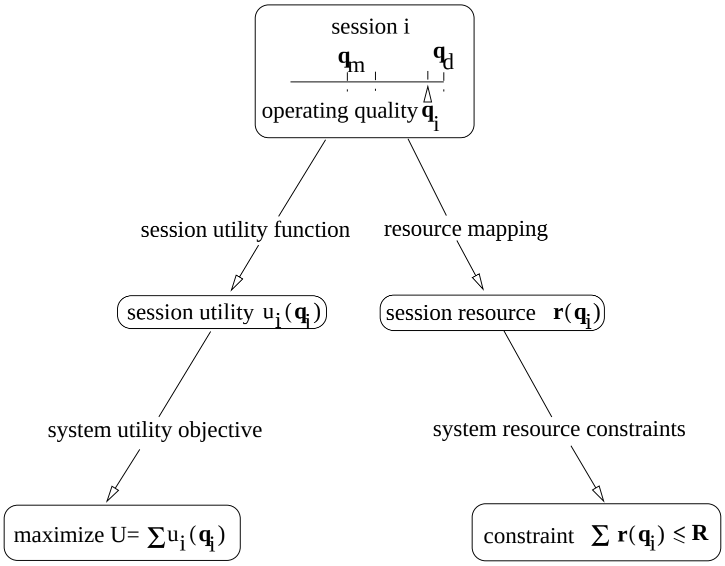


Figure 5: The main concepts of the Utility Model.

3.6 Admission of New Sessions

When a user wants to start a new multimedia session, the system should admit the session if and only if it has the available resources to support the user's minimum quality requirement. This is called *admission control*. Admission control is necessary for service with quality guarantees because the system has to ensure that enough resources are available at run-time to meet the minimum quality guarantee.

Suppose the system currently has n sessions, and the current system utility is U_n. Now when a user requests a new session with a quality profile, the admission decision requires two steps:

1. The system checks whether any feasible solution of the AMP exists, that is whether there exists any solution where the $n+1$ sessions can share the currently available resources. If such a solution does not exist the new session must be rejected.

2. If there exist one or more feasible solution(s) with $n+1$ sessions, suppose the maximum system utility any feasible solution may achieve is U_{n+1}. If $U_{n+1} \leq U_n$, the new session should be rejected. Otherwise the session should be accepted.

In the above discussion, we used a simple system utility function for both quality adaptation and session admission. However it is not a restriction of the model. For example, if we preferred to admit more sessions than to give better quality to the existing sessions, we could use a system utility function such as

$$U' = f(n) + \sum_{i=1}^{n} u_i(\mathbf{q}_i) \tag{7}$$

where $f(n)$ is a non-decreasing function of n. For instance, if a user pays a revenue of c for each live session, then $f(n) = cn$ is the revenue the system earns from all the live sessions.

When a user is admitted the system must reserve the resources required for guaranteeing at least the minimum operating quality. We note that, if the user of a session wants to change the minimum acceptable quality on the fly, another admission control process will be required to change the reserved resources. However if the user changes her quality profile without changing the minimum quality, the reservation does not have to be affected.

4 AMP as a Knapsack Problem

In this subsection, we express the AMP as a multiple-choice multiconstraint 0-1 knapsack problem (MMKP) which is a generalization of the classical 0-1 knapsack problem. The MMKP may be posed as the following resource allocation problem. Suppose there are n groups of items, and m resources. Group i has l_i items. When item j of group i is picked, it yields a utility value of u_{ij}, and consumes r_{kij} amount of resource k. For resource k, the total resource consumption by the picked items is constrained by the amount of resource available, denoted by R_k. The MMKP is to pick one item from each group in order to maximize the total value of the pick, subject to the resource constraints.

Formally, the MMKP is expressed as follows:

$$U = \text{maximize} \sum_{i=1}^{n} \sum_{j=1}^{l_i} x_{ij} u_{ij}, \tag{8}$$

such that

$$\sum_{i=1}^{n}\sum_{j=1}^{l_i} x_{ij} r_{kij} \le R_k, \quad k = 1, \ldots, m,$$

$$\sum_{j=1}^{l_i} x_{ij} = 1, \quad i = 1, \ldots, n,$$

$$x_{ij} \in \{0, 1\}, \quad i = 1, \ldots, n; j = 1, \ldots, l_i.$$

Using the above definition, mapping AMP to MMKP is straight-forward:

- Quality profile $\mathbf{P}_i = (\mathbf{q}_{i1}, \mathbf{q}_{i2}, \mathbf{q}_{i3}, \ldots, \mathbf{q}_{il_i})$ can be viewed as a stack of l_i items. Thus n quality profiles map to n stacks of items, and item j of stack i denotes operating quality q_{ij} of session i.
- When session i operates at quality q_{ij}, session utility is denoted as u_{ij} (session utility function), and the amount of resource k consumed is denoted as r_{kij} (quality-resource mapping).
- The system utility of AMP is mapped to the value of the pick.
- The resource constraints of AMP are equivalent to the resource constraints of MMKP.
- Finding an operating quality for each session can be viewed as picking one item from each stack.

Figure 6 illustrates the mapping with two system resources: CPU time p and main memory m.

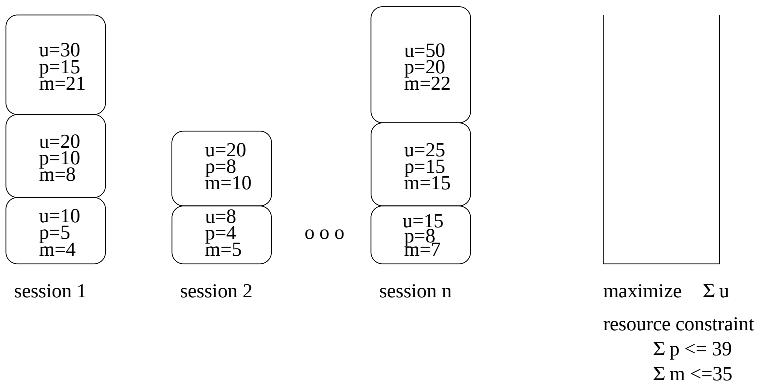


Figure 6: AMP as a multiple-choice multiconstraint 0-1 knapsack problem (MMKP).

n	d	m	Soln. qual. (%)	Comp. time (msec)
3	3	5	92	0.07
3	5	5	84	0.10
3	7	5	97	0.14
3	10	5	79	0.20
5	3	5	87	0.11
5	5	5	90	0.20
5	7	5	88	0.28
5	10	5	85	0.43
7	3	5	91	0.16
7	5	5	88	0.33
7	7	5	89	0.50
7	10	5	89	0.79
10	3	5	94	0.30
10	5	5	88	0.59
10	7	5	88	0.94
10	10	5	85	1.49
20	3	5	92	0.92
20	5	5	89	2.18
20	7	5	86	3.15
20	10	5	89	5.27

Table 1: Performance of the greedy heuristic for MMKP.

4.1 Solving the MMKP

In order to use the Utility Model for dynamic QoS adaptation in a real-time multimedia system, we have to solve the corresponding MMKP instance. The MMKP is one of the harder variants of the knapsack problem [2]. Since this is an NP-hard problem, we do not expect to have an algorithm which will find an optimal solution in less than exponential time.

For this reason, we have developed a greedy heuristic to get a fast and approximate solution of the MMKP. Due to space limitations, we cannot present the details of this approach here; please see Khan [3] for a full description. We tested the heuristic using randomly generated MMKP instances. We used a Pentium 120 MHz personal computer with 32 MB of RAM running Linux OS (RedHat Package Version 4.2, Kernel 2.0.30) for our experiments. Table 1 shows the performance numbers. Here solution quality denotes the value of the objective function achieved by the heuristic expressed as a percentage of the optimal value, and computation time is expressed in msec. We computed the optimal value using a rather slow branch-and-bound search algorithm. We observe that the heuristic achieved a solution quality which was within 21% of the optimal value, within reasonable computation time. For example, for $n = 10, l = 5$ and $m = 5$, the heuristic produced an objective value which is 15% less than the optimal value in 0.59 msec. We believe that for time-critical applications, such as QoS adaptation in a multi-session multimedia system, this heuristic will be very suitable.

5 Resource Allocation Policy

The quality profile contract between the sessions and the system requires a reservation based resource management scheme. The system reserves resources to ensure at least the minimum quality to each session. At any time, part of the system resources are reserved, and it is the remaining unreserved resources which the system can use in order to provide adaptive quality to the sessions. Thus the resources allocated to a session may have both reserved and unreserved components. In the following we present three resource allocation policies: revenue maximization, fair share, and priority, and investigate their implications for the system utility objective.

- *Revenue Maximization Policy:*

  For commercial systems, the usual objective is to maximize revenue. If session i's utility $u_i(\mathbf{q}_i)$ is interpreted as the revenue paid by the session in return for the resources of resource vector $\mathbf{r}(\mathbf{q}_i)$, then maximizing the utility function $U = \sum_{i=1}^{n} u_i(\mathbf{q}_i)$ under the resource constraint $\sum_{i=1}^{n} \mathbf{r}(\mathbf{q}_i) \leq \mathbf{R}$ implies maximization of total revenue which the system could earn by allocating its resources. Under this policy, the system tends to provide more resources to sessions which are prepared to pay more.

- *Fair Share Policy:*

  What does a fair share policy mean in a multi-session multi-resource environment? We note that the system utility objective with revenue maximization interpretation is fair in the sense that it allocates resources using a fair competition among the sessions' utilities (or offered value in the sense of revenue). If session A offers more money than session B for the same resource vector, session B will not get this resource vector unless session A also gets that or better.

- *Priority Policy:*

  The system utility objective with revenue maximization interpretation can easily be extended to a priority-based policy. This can be done by using a system utility objective which is to maximize U given by,

  $$U = \sum_{i=1}^{n} w_i u_i(\mathbf{q}_i), \tag{9}$$

  where w_i is a weight factor for session i's utility. Larger w_i means higher priority for resource allocation. In revenue terms, w_i can be interpreted as an exchange rate to convert session i's local currency to a universal currency in which system utility is measured.

6 Related Work

Moser presented an Optimally Graceful QoS Degradation (OGQD) Model where the realized QoS of a *single session* multimedia service is gracefully degraded to the change of resource availability [4]. Suppose a multimedia service consists of g elementary services such as audio input, audio output, video input and video output. Elementary service i has n_i implementations $\mathbb{I}_i = (I_i^1, I_i^2, \ldots, I_i^{n_i})$ where each implementation has m_i parameters. The QoS of elementary service i is expressed as a QoS vector, $\mathbf{q}_i = (d_{i1}, d_{i2}, \ldots, d_{im_i})$. Similarly the utility of elementary service i is expressed as a utility vector, $\mathbf{u}_i = (u_{i1}, u_{i2}, \ldots, u_{im_i})$. The user specifies the service requirements expressed as $\mathbb{Q} = ((\mathbf{q}_1, \mathbf{u}_1), (\mathbf{q}_2, \mathbf{u}_2), \ldots, (\mathbf{q}_g, \mathbf{u}_g))$ where $\mathbf{q}_i$ is the desired QoS of elementary service i and $\mathbf{u}_i$ is the corresponding utility.

For each elementary service, the system has to find a suitable implementation and the operating QoS of that implementation depending on the available resources. The goal of the OGQD system is to provide multimedia service with minimum degradation from desired QoS.

Moser formulates the OGQD problem as follows [4].

Find

$$\mathbf{q}(I_i) = (q_{i1}, q_{i2}, ..., q_{im_i}),\ \ I_i \in \mathbb{I}_i \text{ for } i = 1, 2, \ldots, g, \tag{10}$$

to maximize

$$\sum_{i=1}^{g} \left[\frac{1}{m_i} \sum_{l=1}^{m_i} u_{il} - \frac{1}{m_i} \sum_{l=1}^{m_i} u_{il} \left(\frac{d_{il} - q_{il}}{u_{il}} \right)^2 \right]^2$$

subject to

$$\sum_{i=1}^{g} \mathbf{r}(I_i) \leq \mathbf{R}.$$

The OGQD and Utility models have some high level similarities: both models view the dynamic QoS adaptation as an optimization problem, and they assume a similar quality-resource mapping. However the two models are fundamentally different in their scope, objective, problem formulation, and mapping:

- The Utility Model is a simple mathematical model for *multi-session* adaptive multimedia systems. Each session provides a quality profile with *multiple* operating qualities, and the system finds the dynamic QoS of each session in order to maximize some system utility function. This model degrades the QoS of one or more sessions when there is a scarcity of available resources.

  On the other hand, the OGQD Model is a powerful model to express the QoS degradation of a *single multimedia session*. The session provides a desired QoS, and the system finds a realized QoS in order to maximize a session utility function. The model tries to exploit resource-resource tradeoff for providing the desired QoS (e.g., processor-bandwidth tradeoff by changing compression algorithm), and when it fails, it degrades the QoS by exploiting resource-QoS tradeoff.

- The Utility Model provides a unified approach for dealing with session admission and quality adaptation. However, since OGQD deals only with a single session, session admission is out of its scope.

- The Utility Model expresses the multi-session adaptive multimedia system where the QoS of individual sessions may be dynamically adapted to the resource availability of the system and to the choice of the users as an MMKP. Moser also maps the OGQD problem to an MMKP instance, but how an optimization problem with a nonlinear objective function may be mapped to an MMKP is not clear from his article [4].

Kawachiya *et al.* of the Keio MMP project proposed a system architecture for QoS adaptation among multimedia sessions [5]. The Utility Model and the MMP architecture share the same general goal of designing an adaptive multimedia system with multiple concurrent sessions. However the foci of the two projects are fundamentally different and rather complementary: the focus of the Utility Model is to find the adaptive quality of each session at any given system state; whereas the focus of the MMP architecture is to provide a mechanism to enforce quality adaptation using a reservation-based resource management scheme.

7 Conclusion

We have presented the Utility Model for adaptive multimedia systems. It is based on the concepts of quality profile, resource mapping, session and system utility, and system resource constraint. This model incorporates the dynamics of resource management and the interplay of media components within a multi-session multimedia environment.

We have defined the adaptive multimedia problem within an AMS, formulated it as a dynamic optimization problem, and mapped it to a multiple-choice multiconstraint 0-1 knapsack problem. We have shown how this model may be used in dynamic quality adaptation of existing multimedia sessions, and in admission control of new sessions. It provides a computationally feasible real-time method to optimize the utility (e.g., revenue) of real multimedia systems.

Some of the issues which deserve farther investigation are as follows:

- Resource Mapping: Further research is required for mapping operating qualities to required resources. Ongoing research at Keio University and at Washington University at St. Louis is addressing this problem [6,7].
- System Architecture Design and Prototype Implementation: We are currently working on the design of a system architecture and a prototype implementation for adaptive multimedia systems based on the Utility Model [8].
- Extending the Utility Model: Currently, the Utility Model incorporates the dynamic behavior of multiple sessions within a single end-system. It will be interesting to see whether it may be extended to express the dynamics of a distributed system where resources are scattered around a communication network. This involves complex issues such as load and resource locality, deterministic and statistical multiplexing, and communication cost.

  The Utility Model may be extended to distributed virtual worlds where the consistency level among virtual worlds is adapted, based on available resources. For example, when enough bandwidth is available avatars may send facial expressions updates as well as position updates. However when the network is congested, the avatars may only send position updates. In this context, an architecture for scaling virtual worlds is suggested in [9].

Acknowledgments

We wish to thank our friends at the UVic Panda lab for sharing their views and insights on this research. John Foxgord provided valuable comments on an earlier draft. We also thank our reviewers for suggestions that improved the presentation significantly. Shahadat Khan was supported by a Canadian Commonwealth Scholarship, and by NSERC of Canada.

References

1. K. Nahrstedt. *An Architecture for End-to-End Quality of Service Provision and its Experimental Validation.* PhD thesis, Department of Computer and Information Science, University of Pennsylvania, August 1995.
2. Martin Moser, Dušan P. Jokanović, and Norio Shiratori. An Algorithm for the Multidimensional Multiple-Choice Knapsack Problem. *IEICE Transactions on Fundamentals of Electronics*, 80(3):582–589, March 1997.
3. Shahadat Khan, Kin F. Li, and Eric G. Manning. Optimal and Fast Approximate Solutions for the Multiconstraint Multiple-choice Knapsack Problem. Technical Report ECE-97-2, Department of ECE, University of Victoria, September 1997.
4. Martin Moser. Declarative Scheduling for Optimally Graceful QoS Degradation. In *Proceedings of IEEE Multimedia Systems*, Hiroshima, Japan, 1996.
5. Kiyokuni Kawachiya, Masanobu Ogata, Nobuhiko Nishio, and Hideyuki Tokuda. QoS Control of Continuous Media: Architecture and System Support. Technical Report RT0108, IBM Tokyo Research Lab, Kanagawa, Japan, April 1995.
6. Nobushiko Nishio and Hideyuki Tokuda. QOS Translation and Session Coordination Techniques for Multimedia Systems. In *Proceedings of Sixth International Workshop on Network and Operating System Support for Digital Audio and Video*, Jushi, Japan, 1996.
7. R. Gopalakrishnan and G. Parulkar. Efficient quality of service support in multimedia computer operating systems. Technical Report WUCS-94-26, Dept. of Computer Science, Washington University, St. Louis, 1994.
8. Shahadat Khan, Kin F. Li, and Eric G. Manning. Padma: An Architecture for Adaptive Multimedia Systems. In *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, 1997.
9. Rodger Lea, Pierre G. Raverdy, Yasuhiko Honda, and Kouichi Matsuda. Scaling a Shared Virtual Environment, 1997. Available at http://www.csl.co.jp/person/rodger.