

Efficient Streaming for Delay-tolerant Multimedia Applications

A synopsis submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

by

Saraswathi Krithivasan

(Roll No. 03429601)

Under the guidance of

Prof. Sridhar Iyer



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY–BOMBAY

2008

Abstract

Delay-tolerant multimedia applications, where clients specify the time when playout must start, fit the profile of many emerging applications, including distance education and corporate training. Such applications typically depend on a Content Delivery Network (CDN), where the Content Service Provider (CSP) disseminates multimedia content to geographically dispersed clients through a distribution network from servers located on the CDN. In these applications, a client C_i connects to the source \mathcal{S} at time t_0 and requests for contents with base encoding rate Γ and playout duration \mathcal{T} ; C_i specifies the time it is willing to wait for the playout to start, its *delay tolerance* δ_i ; Playout at C_i starts at $(t_0 + \delta_i)$ and ends at $((t_0 + \delta_i) + \mathcal{T})$. Note that t_0 may also be the start time of a scheduled streaming session.

This thesis deals with the issue of maximizing the quality of the multimedia contents delivered at the clients even when link bandwidths are constrained in the path from the source to the clients. To achieve this, we develop a suite of algorithms that can exploit clients' delay tolerance considering the following parameters: (i) *Service type*: whether contents are streamed according to a schedule or occur on-demand and (ii) *Bandwidth*: whether link capacity is static or variable, using appropriate *resources* at the nodes – Transcoders, Layer encoders, or Streaming servers with transcoding or layering functionality. In particular, we consider the following three cases: (i) Scheduled streaming when bandwidths are static, (ii) Scheduled streaming when bandwidths are varying and predicted, and (iii) On-demand streaming when bandwidths are static. The algorithms developed are a result of formulating the objectives in an optimization framework or by an analysis of properties of the network topology and client requirements. Furthermore, where an optimal algorithm is computationally expensive, we develop algorithms based on heuristics as practical alternatives. The approaches are validated through extensive simulations, using topologies derived from real-world scenarios.

The algorithms developed in this thesis would help a CSP serving clients in a subscription based network in: (i) improving the quality of reception at the clients by leveraging their delay tolerance values, (ii) estimating the resources required to provide the best delivered rates to clients, and (iii) determining placement of resources to maximize the delivered rates at clients. Thus, using the analysis presented and algorithms developed in this thesis, a CSP can deploy resources in order to ensure effective quality of service to clients and efficient resource utilization by leveraging clients' delay tolerance.

Introduction

With the proliferation of world-wide computer networks, several popular streaming media applications have emerged: Universities offering their courses to a set of global subscribers, service providers streaming movies requested by their clients, and multinational corporations providing training to employees across cities. Heterogeneous communication architectures comprising of satellite, terrestrial links as well as the Internet are increasingly deployed for such applications. In these applications, a source disseminates multimedia contents (that may be encoded at different rates) to a set of geographically distributed clients, through links of varying capacities and characteristics [3].

We consider applications where the clients specify two requirements: a minimum rate at which the contents are to be played out (a minimum quality requirement) and a startup delay after which they want the playout to start (a specific start time requirement). We term such applications as **delay-tolerant applications**. Most of the contents in the education and entertainment domain are in this category. For such applications it is possible to allow clients the convenience of starting the playout at a specific time while improving the quality of the playout, even in the presence of bandwidth constrained links along the path from the source to the clients. We have explored different dimensions of this problem in this thesis.

Focus area

Figure 1 depicts the area of focus for this thesis. We have considered two user-level parameters, *loss in quality* and *start-up delay*. Loss is the result of dropped data packets when there is insufficient bandwidth on a link to support the rate at which data is flowing through it. Start-up delay is the time lag due to the need to buffer sufficient data at the client so as to reduce jitter during the playout. We can classify applications along two dimensions: *sensitivity* and *tolerance*.

1. Multimedia applications such as e-surgery which are demonstrated in real time fall under the *loss-sensitive* and *start-up delay sensitive* category, i.e., such applications need loss-free transmission in real time. These applications require dedicated bandwidth from the source to the client so that the quality of reception is guaranteed.
2. Applications such as live sports streaming fall under the *loss-tolerant* and *start-up delay*

		Loss	
		Sensitive	Tolerant
Start-up delay	Sensitive	Time-critical Applications e.g.: e-surgery	Soft real-time Applications e.g.: live streaming
	Tolerant	Delay-tolerant applications: e.g.: distance education	Loss and delay tolerant applications e.g.: static content streaming

Figure 1: The problem space

sensitive category. Such applications have been researched extensively. A review of the existing mechanisms for effective and efficient delivery of multimedia in [2] [6] indicates that existing work treats applications involving multimedia dissemination as soft real-time applications that can tolerate some transmission errors and *explores ways to minimize the startup delay*.

3. Applications such as streaming of personal video clips fall under the *loss-tolerant* and *start-up delay tolerant* category. We believe that such applications can be handled by the current best effort networks without any additional features. By allowing for *longer* buffering at the clients, quality can be improved but, without guarantees for loss-free playout. Typically, such applications do not have stringent quality requirements.
4. We focus on the fourth category of multimedia applications — those that are *loss-sensitive* and *start-up delay-tolerant*. We have developed techniques for exploiting client specified delay tolerance in order to maximize delivered multimedia quality in heterogeneous network environments. Most multimedia applications in education and entertainment where contents are valid for a period of time, fall in this category. However, there is hardly any research that deals with improving delivered quality for applications in this category. We show that by providing flexibility of viewing time to the clients and allowing them to specify the start of playout, delivered quality can be improved even in the presence of links with bandwidth constraints. The basic idea is as follows: given that weak links with limited bandwidths can occur in the path from source to a client; when a client requests

for a particular content, the source starts transmission immediately; however, since the client wants the playout to start only after a specified time, the extra time is leveraged to deliver contents at a higher encoded rate, enhancing the quality of reception at the client.

The problem: An overview

In this section, we define the context and provide a high level definition of the problem. We also provide a discussion of the parameters that affect the problem to outline the scope of the solutions presented in this thesis.

The context

Traditionally, Content Delivery Networks (CDNs) [1] serve multimedia contents to clients through proxy servers deployed at strategic locations at the edge of its core network, as shown in Figure 2. Typically, a central server which is a content repository and a set of replica servers which contain replicated contents, form the *core network* of the Content Service Provider (CSP). With the tremendous growth in multimedia applications and the number of clients accessing such applications, a replica server itself may serve clients connected to it through a multi-hop distribution network of heterogeneous links. We focus on such a distribution network comprising of a replica server serving as a source for clients over a multicast tree.

Consider a multimedia stream originating at source \mathcal{S} , the root of the multicast tree and delivered to clients C_1, C_2, \dots, C_n connected through relay nodes R_1, R_2, \dots, R_i . Without loss of generality, we define the multicast tree to be made up of the following components:

- The **Distribution network**: is a multicast tree with the source \mathcal{S} as root and the *region nodes* G_i s as leaves. We assume that this network is controlled by the CSP through Service Level Agreement (SLA) with the Internet Service Provider (ISP). Typically, a specified amount of bandwidth is allocated for a given application on every link in the distribution network; we refer to this situation as *provisioned links* or simply refer to the link being *provisioned*.
- The **Access network**: is a forest with each region node G_i as root and the clients C_i s as leaves. Typically bottlenecks occur here. CSP may not have control over the access network nodes.

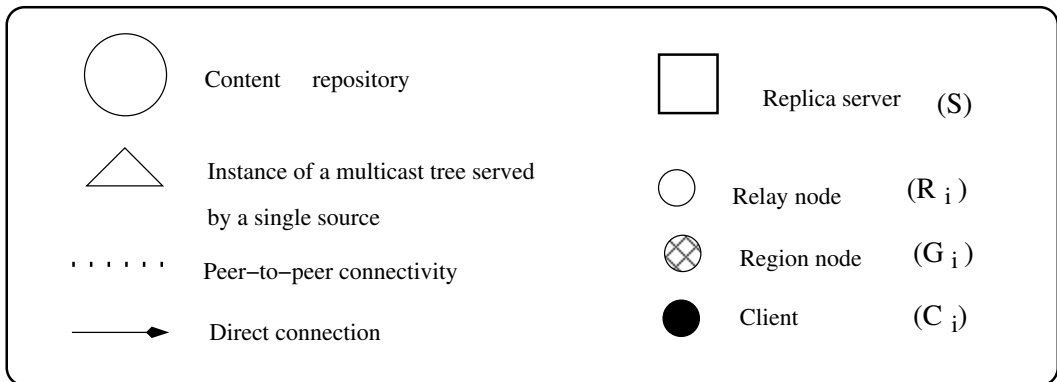
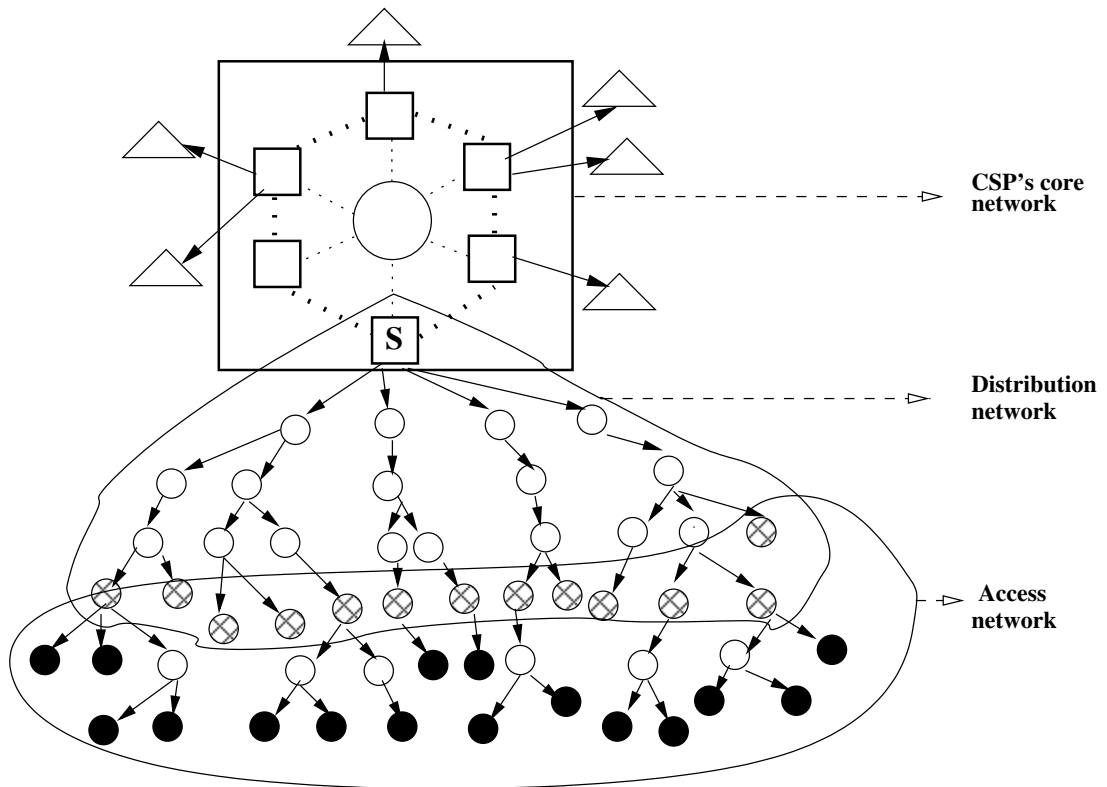


Figure 2: The context

- **Region nodes** G_i s connect the distribution network to the access network. These nodes are edge nodes in the distribution network; they are typically designated by the CSP.

Problem definition

Given:

- A multicast tree enabled with a specified set of resources such as transcoders, layer encoders, and streaming servers, with source \mathcal{S} serving clients C_i s.
- Each C_i specifies two requirements: (i) a minimum acceptable rate γ_i^{min} (in kbps) and (ii) delay tolerance δ_i (in seconds), the time it is willing to wait, once connection is established, for start of playout of content.
- Content has *playout duration* \mathcal{T} and is encoded at *base encoding rate* Γ , the best rate at which the contents can be played out at any client.

Assumptions: The topology of the distribution network and access network is known with link bandwidths remaining constant over a given prediction interval \mathcal{P} .

Objective: Leverage delay tolerance δ_i specified by each C_i to:

- provide continuous playout for each C_i at a rate r_i such that each r_i is greater than or equal to the minimum acceptable rate γ_i^{min} and
- maximize r_i s across all C_i s; i.e., minimize $\forall i, i = 1, 2, \dots, m, \sum_i (\Gamma - r_i)^2$, where m is the number of clients in the multicast tree.

The parameter space

We consider the following parameters which define the two major dimensions of the problem.

1. Service types

- *Scheduled streaming:* In scheduled streaming, content from a source \mathcal{S} , is streamed synchronously to all the clients, starting at time t_0 . When \mathcal{S} is capable of transcoding or layering, each subtree in the distribution network may be served with a stream encoded at a different rate. However, the streaming session starts at t_0 for all clients.

- *On-demand streaming*: In on-demand streaming, each client may request for the contents at a different time; in this case the streaming is asynchronous. Suppose a client C_i requests for contents at time t_i specifying its requirements: a minimum acceptable rate γ_i^{min} and delay tolerance δ_i . As different clients (which may share links in their paths), request for the contents at different times, the shared links may be busy servicing a client request when another client from the same subtree requests for the contents. Thus, streaming from the source \mathcal{S} starts when the links from \mathcal{S} to C_i are free. Note that playout of the contents encoded at least at γ_i^{min} must start at C_i at time $(t_i + \delta_i)$.

2. Link Bandwidths

- *Known and static*: In this case, link bandwidths are assumed to be known and static over the period starting from when a client requests for transmission to end of playout at the client, defined as the *connection duration*, given by $(\delta_i + T)$.
- *Varying and predicted*: In this case, link bandwidths are assumed to be varying over the period starting from when a client requests transmission to the end of playout at the client. But, we assume a bandwidth profiling module which estimates bandwidths available over several prediction intervals that cover this period. We assume these predicted bandwidths to be static over each prediction interval.

Based on a detailed analysis of the appropriateness of using transcoders or layer encoders, we choose transcoders when the link bandwidths are static and layer encoders when link bandwidths are varying, to formulate our solutions. When client requests are on-demand, streaming servers are placed at appropriate relay nodes to utilize the available bandwidth on the links efficiently. Here again, streaming servers having transcoding capability are used when the bandwidths are static; When bandwidths vary over the session duration streaming servers with layer encoding capability are used. Thus, we come up with the following four combinations of parameters:

- *Scheduled streaming, Known and static link bandwidths*
- *Scheduled streaming, Varying and predicted link bandwidths*
- *On-demand streaming, Known and static link bandwidths*

- *On-demand streaming, Varying and predicted link bandwidths*

Note that when the CSP has complete control over the distribution network and the access network, it can provision all the links in the network; in such a case, the link bandwidths are *known and static*. Even when the CSP does not control the access network, in a subscription-based network, it is reasonable to assume that the CSP would have enough knowledge to predict the bandwidths on the links; in such a case, while the link bandwidths may be varying over the session duration, it is possible for the CSP to predict the bandwidths on the links.

Overview of solution approaches

In this thesis, we present the architecture of the nodes required to support the two service types (refer to Chapter 3: System model and detailed problem definition) before we proceed to discuss our solution approaches. In this section, we consider the four combinations discussed in the previous section and briefly outline our solution approach to problems identified under each case.

1. Problem 1A: Scheduled streaming, static link bandwidths

- *Determining optimal rates delivered at clients for a given transcoder placement option:* Our objective is to develop solutions to find the optimal rates delivered at clients given that a specific set of nodes have transcoders. We formulate the problem as an optimization function, subject to the following three sets of constraints: (i) **Rate constraints** which bound the lower and upper limits of the encoded rate, (ii) **Transcoder constraints** that enforce the property of transcoding, viz., the encoded rates can only be non-increasing, and (iii) **Delay tolerance constraints** which enforce the client specified delay tolerance, to find the maximum rate at which the stream should be played out for each client.

Given the exponential search space, as we experimentally show, the optimization based approach can incur computation overheads which make it impractical. We show that it is possible to develop a two-pass algorithm, *find_opt_rates_I* with $O(DC)$ complexity, where D is the number of levels in the multicast tree and C is the number of clients. This algorithm finds the optimal delivered rates at clients, for a given transcoder placement option.

- *Determining optimal number of transcoders and their placement for providing best delivered rates at clients:* When all relay nodes are transcoder capable, algorithm *find_opt_rates_I* determines the stream rates flowing through each link in the network and the optimal rates delivered at the clients. We call these as the *best delivered rates*, when there is no constraint on the number of transcoders. Even though all relay nodes have transcoders, all of them need not be enabled to serve the clients with the best delivered rates. Our objective is to find the minimum set of transcoders referred to as the *optimal set*, denoted by \mathcal{O} required to serve the clients with the best delivered rates.

By adding checks to detect redundant transcoders at nodes, our modified algorithm *find_opt_rates_NR* determines the stream rates through the links and optimal delivered rates at the clients for any transcoder placement option. The stream rates as determined by *find_opt_rates_NR* for the AT option are used by algorithm *find_min_transcoders* to find \mathcal{O} .

- *Determining optimal placement for a given number of transcoders:* In practical scenarios, resources available are limited. Given a number of transcoders $q < |\mathcal{O}|$, we know that the delivered rates at some clients would be less than their best delivered rates. Our objective is to place the transcoders such that the decrease in delivered rates across clients is minimal; i.e., given a limited number of transcoders, we find the placement option that delivers the best possible rates to clients.

We formulate the problem as an optimization function subject to the rate constraints and delay tolerance constraints as explained in 1A. Instead of the transcoder constraints, we specify **number of transcoder constraint** to limit the number of transcoders deployable to the specified value q . We also develop optimal algorithms that consider combinations of nodes to find the best placement. We show that by considering the nodes in the *useful set* denoted by \mathcal{U} , we can determine the optimal placement for q transcoders. \mathcal{U} is at best equal to \mathcal{O} and at worst equal to $\mathcal{O} \cup \text{super nodes}$. Here it suffices to say that super nodes are nodes not in \mathcal{O} that need to be considered for finding optimal placement of q transcoders. This algorithm reduces the time required to find the optimal placement by several orders of magnitude depending on the number of super nodes.

We experimentally show that the computation time required by the optimal algo-

gorithms render them infeasible in practice. This motivates us to develop two greedy alternatives: a *max-gain* algorithm and a *min-loss* algorithm and evaluate these algorithms. Placement chosen by the greedy algorithms perform close to the optimal algorithms. A CSP with limited number of transcoders can use our algorithms to find the best placement for these transcoders.

2. Problem 1B: Scheduled streaming, varying link bandwidths

When link bandwidths vary over the session duration, we assume a bandwidth prediction module that provides an *advance estimate* of the available link bandwidths over prediction intervals spanning the session duration. Suppose t_0 is the scheduled start of the streaming session; client requirements are given. The advance estimate provides predicted bandwidths over prediction intervals spanning $(t_0 + \max(\delta_i))$ where δ_i s are the delay tolerance values of the clients.

- *Determining optimal rates delivered at clients when the link bandwidths are varying:*
We formulate the problem as an optimization function, subject to the following three sets of constraints: (i) **Rate constraints** which bound the lower and upper limits of the encoded rate, (ii) **Layer constraints** that enforce the property of layering, viz., the stream rates flowing through two consecutive links must be such that the stream rate through the first link is greater than or equal to the rate flowing through the next one; note that these constraints are the same as the transcoder constraints as given in 1(a), and (iii) **Delay tolerance constraints** which enforce the client specified delay tolerance, to find the maximum rate at which the stream should be played out for each client.

Because of the complexity of the optimization approach, we explore different solutions based on the nature of bandwidth variation. We start with simple ones that convert the problem to the static bandwidth case such as (i) using the minimum value of bandwidth predicted over the session duration for each link (ii) using the average bandwidth over the session duration for each link, and (iii) treating each prediction interval as an instance of static bandwidth case, termed *interval-by-interval algorithm*. For each solution we propose, we outline the usefulness of that solution and its limitations.

Our objective is to maximize the use of available bandwidth to provide clients with

the best possible loss free delivered rates. To this end, we develop a *link-by-link algorithm* that considers each link to find the maximum rate supported by the link without loss. Considering the links in each clients' path, the algorithm determines the delivered rates at the clients. We show that the Link-by-link algorithm with linear complexity performs close to the optimal algorithm in finding the maximum loss-free delivered rates at the clients. Thus, even when link bandwidths vary over the session duration, our algorithms would aid a CSP to find the optimal delivered rates at the clients.

- *Adjusting stream rates if revised bandwidth estimates are available:* We show how to use the *revised estimate* of link bandwidths, if available at the beginning of every prediction interval, to adjust the delivered rates at the clients; note that the delivered rates are calculated by the link-by-link algorithm using the *advance estimate* across all prediction intervals. We show that this algorithm is effective in exploiting the available bandwidth on the links over each prediction interval compared to just using the advance estimate.

3. Problem 2A: On-demand streaming, static link bandwidths

In the previous two problems, we considered scheduled streaming, where streaming of a given content starts at time t_0 for all clients. Now we consider on-demand streaming where client requests are handled as and when they arrive. Unlike the scheduled streaming case, transmission from \mathcal{S} may not start immediately, if links are busy serving requests from other clients. Thus, a client C_i connects to the source \mathcal{S} at time t_i and requests for contents with base encoding rate Γ and playout duration \mathcal{T} ; C_i specifies the time it is willing to wait for the playout to start, its delay tolerance δ_i . \mathcal{S} admits C_i and services its request with an appropriate rate r_i such that playout at C_i starts at $(t_i + \delta_i)$ and ends at $((t_i + \delta_i) + \mathcal{T})$, if $r_i \geq$ the client's minimum required rate γ_i^{min} .

We develop a mechanism that combines data transfer with streaming to achieve efficient streaming for on-demand requests for same contents. Our mechanism and the algorithms that we develop for placing the streaming servers are effective in handling requests from clients for contents which is valid for a given period of time.

- *Determining optimal rates delivered at clients when the clients request for contents over a period of time:* We assume an observation period over which client request

arrivals are monitored. Network characteristics and client requirements are given. Given highly provisioned links in the core distribution network, we propose an algorithm that combines data transfer and streaming mechanisms to efficiently handle requests for the same contents over a period of time.

- *Determining appropriate placement for streaming server:* Given the arrival pattern of client requests over the observation period, streaming servers have to be placed at appropriate network nodes in order to maximize number of clients serviced. Using observations of the network characteristics related to data transfer and streaming, we develop rules for placement of the streaming servers. These rules are applied in the algorithm for finding the number of serviced clients and the rates delivered at the clients.

4. **Problem 2B: On-demand streaming, varying link bandwidths**

Determining optimal rates delivered at clients over a observation period, given the network characteristics and an arrival pattern, when the core distribution network links are highly provisioned and access network links have varying bandwidths: Using the insights gained from the analysis for solving Case 1B and Case 2A, algorithms can be designed for this case. However, this case is not considered in this thesis.

Lastly, we analyze implication of buffers, considering the case when a client device is memory constrained. This discussion is relevant to clients joining the network using mobile phones or other hand-held devices which may have limited memory. Given the bandwidth of the weakest link in a client's path, we find the value of delay tolerance that delivers the stream at the base encoding rate at the client, when there is no shared link constraint imposed by other clients. Beyond this value of δ , while the delivered rate can not be improved, the stream has to be stored at the client, as playout at the client can start only after δ . This property has led to the definition of *residual delay tolerance*, the time when links in the path of a client are not utilized. We have suggested ways to leverage this residual delay tolerance to enhance revenues for the CSP through rescheduling the session.

Summary of contributions

In this thesis, we have identified multimedia applications which we have termed *delay-tolerant applications*; in these applications, delivering the content with acceptable quality is given more importance than delivering the contents in real-time. Given the nature of the contents in these applications (which are relevant for a period of time), it is prudent to use the delay tolerance of clients to improve the quality of reception at the clients.

This idea is especially useful to the current CDN scenario, where bandwidth bottlenecks occur along the path to a client; even when the CSP provisions links in its distribution network, the *last-mile* problem persists.

Thus, our first contribution in this thesis is the idea of delay-tolerant applications. Having identified applications that fit in the profile of delay-tolerant applications, we have explored various properties of such applications, given the context and parameters that affect such applications. We have also presented the architecture of nodes required to support such applications.

We identified the parameters that define the characteristics of the delay-tolerant applications and four distinct problem areas based on these parameters. We started out with scheduled streaming with the simple case when all link bandwidths are static, and formulated algorithms to find the optimal delivered rates at the clients; we also devised algorithms to find the resources required for providing best possible playout at the clients and algorithms for best placement of resources when the resources are limited. For each problem, we provided an optimal solution and when the optimal solution is expensive, we provided algorithms based on network properties and client requirements.

Then we relaxed the assumption of static bandwidths to include variability in bandwidth over prediction intervals spanning the session duration. We devised algorithms that find the loss-free delivered rates at the clients when bandwidths are varying. We also presented a formal analysis of the on-demand streaming case when bandwidths are static.

We have not studied the fourth combination, (on-demand streaming, variable bandwidth) in this thesis. However, analysis and algorithms we have developed for the other three cases can be used as building blocks to solve the problems in this area.

Given that all our algorithms assume unlimited buffers at the network nodes, we discussed size of buffers required at relay nodes and clients as the concluding part of our thesis. The size of buffer required at the client depends on delay tolerance value of the client. We derived the

expression for the delay tolerance value of a client required to deliver the stream encoded at the best possible rate to the client, given the bandwidth of the weakest link in its path. We analyzed the impact of delay tolerance on the buffer size at the client and found that beyond certain value of delay tolerance, data for the entire session is buffered at the client and the data remains stored at the client till the start of playout. If client devices are memory constrained, such scenarios can pose problems. We explored solutions to handle memory constrained client devices.

In summary, we have developed and implemented algorithms that can be used by the CSP in a tool to make decisions on service quality and resource requirement and placement. We have developed algorithms for the various sub-problems identified and demonstrated their usefulness in aiding a CSP to administer delay-tolerant multimedia applications.

Bibliography

- [1] Akamai: The Business Internet
<http://www.akamai.com/>
- [2] S. Krithivasan, Mechanisms for Effective and Efficient Dissemination of Multimedia, *Technical report*, September 2004. URL: www.it.iitb.ac.in/~saras/Papers
- [3] J. Kurose, K. Ross, Computer Networking: A Top Down Approach Featuring the Internet, *Addison Wesley*, 2003.
- [4] The MPEG Homepage
<http://www.chiariglione.org/mpeg/>
- [5] B. Shen, S-J. Lee, Transcoding-enabled Caching Proxy for Video Delivery in Heterogeneous Network Environments, *Proceedings of Internet and Multimedia Systems and Applications*, pp. 360-365, 2002.
- [6] X. Wang, and H. Schulzrinne, Comparison of Adaptive Internet Multimedia Applications, *Invited paper, Special issue on distributed processing for controlling telecommunications systems*, June 1999.