# Resource Deployment Strategies for *Delay-tolerant* Multimedia Applications

Saraswathi Krithivasan, Sridhar Iyer
IIT Bombay
saras,sri @it.iitb.ac.in

## Abstract

Consider multimedia applications in which each client $C_i$ requests for a specific multimedia stream at time $t_0$ and specifies the following requirements: (i) $\alpha_i$, denoting the minimum acceptable quality, i.e., encoding rate for the delivered content (ii) $\delta_i$, denoting *delay tolerance*, a time up to which it is willing to wait for the transmission to start. Thus, $\delta_i$ specifies the deadline till start of transmission. We look at ways to leverage $C_i$'s delay tolerance to provide improved quality through judicious deployment of resources such as buffers, transcoders, caches, and streaming servers. Specifically, while honoring clients' deadlines, we take advantage of their delay tolerance, to overcome the bottleneck links in the path from the source to the client.

## 1. Introduction

With communication networks reaching even remote parts of the world, applications such as distance education, corporate training, multicasting live events, are becoming a reality. While the basic infrastructure exists to reach remote places, intermittent connectivity and bandwidth constraints are prevalent. The type of applications which we term as *delay-tolerant* are those where multimedia information needs to be delivered at a time specified by the client ($t+\delta_i$), where t is the time when the client connects and requests for the content and $\delta_i$ is the client's delay tolerance [2]. At the defined deadline, transmission must start and proceed without any interruption and loss.

Consider a simple network having source S synchronously serving clients $C_1$, $C_2$, and $C_3$ connected through relay nodes $R_1$ and $R_2$ and links 1,2,..,5 with bandwidths in kbps as shown in Figure 1. Consider a stream of 1 hr. duration encoded at 512 kbps. Our aim is to provide loss free transmission to all three clients at appropriate rates.

Let the delay tolerance values $\delta_1$ for $C_1$, $\delta_2$ for $C_2$, and $\delta_3$ for $C_3$ are all zero. Now, the weakest link dictates the delivered rate; all clients get 128 kbps. When $\delta_1 = \delta_2 = \delta_3 = ½$ hr, $C_1$ and $C_3$ get 384 kbps, and $C_2$ gets 192 kbps. Stream encoded at 384 kbps flows through links 1,2, and 3. At $R_2$ a transcoder is needed to provide 192 kbps to $C_2$. We note that while delay tolerance of clients can be used to enhance the delivered rates, we need transcoding capability at relay nodes to provide appropriate rates to clients. The challenge is to meet clients' quality and timeliness requirements by placing transcoders at strategically chosen relay nodes.

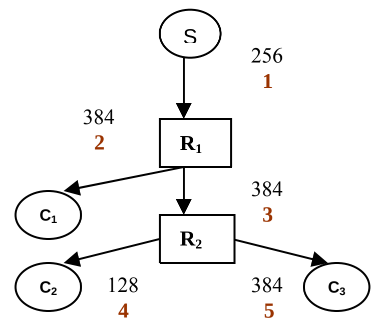


**Figure 1: Motivating example**

In Section 2, we present some basic work on determining rates deliverable to clients by leveraging their delay tolerance. Note that our strategies enhance the quality of the delivered stream by making use of the client specified startup delays; In contrast, existing work treats multimedia delivery as a soft real-time application having *immediate* play back requirement and hence propose strategies to *minimize* the startup delay [1]. We present our analysis to reduce the complexity of transcoder placement algorithms in Section 3. In Section 4 we present a content delivery network model to illustrate how our results can be used in practice. We conclude by presenting our other work in progress in Section 5.

## 2. Leveraging delay tolerance

Given a Content Service Provider's (CSP) goal of maximizing utilization of the available resources to best serve its clients, we first need to quantify the effect of delay tolerance specified by a client on its deliverable rate. Assuming static characteristics and a synchronous stream, Theorem 1 gives the expression for the deliverable rate at a client considering the path from the source to the client in isolation.

Consider a path S-$R_1$-$R_2$….-$R_m$-C, where S is the source and C is the client, $R_1$, $R_2$,…., $R_m$ are the relay nodes connected by links $l_1$, l2,…, $l_{m+1}$. Let $\delta$ be the delay tolerance of C and let the link bandwidths of $l_1$, l2,…., $l_{m+1}$ be $b_1$, $b_2$,…, $b_{m+1}$. Henceforth i is used to index the links that make up the path between S and C. Let α be the base encoded rate and T be the duration of the stream.

**Theorem 1**: Given a path S-$R_1$-$R_2$….-$R_m$-C, the deliverable rate $r_c$ at C is given by:

$$r_c = \alpha \quad \text{if } \min(b_i) >= \alpha \; ; \qquad \textbf{(1a)}$$
$$= \min(b_i)+((\min(b_i)*\delta)/T) \quad \text{otherwise}; \qquad \textbf{(1b)}$$

**Proof:** Considering (1a), the bandwidth on the weakest link in a given path is greater than or equal to α. Since the best rate that any client can get is the base encoding rate α, it can be trivially proven that the client receives the stream at α, irrespective of its delay tolerance value.

We use induction to prove (1b).

Consider the base case, having a path S-$R_1$-C as shown in Figure 2.

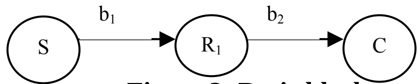


**Figure 2: Basic block**

Let Min $(b_1,b_2)= b_1$;

To provide loss free play out, the stream has to be encoded at $b_1$. Given the additional time δ, S can send $(\delta*b_1)$ bits to C over time T. (Note that we assume the worst case bit rate) Thus, the additional amount of data that S can send to C per second is $(\delta*b_1)/T$.

Hence, $r_c = b_1+ ((b_1 *\delta)/T)$.

Now consider a path S-$R_1$-$R_2$..-$R_k$-C, having bandwidths $b_1,b_2,\ldots,b_{k+1}$.

If (1b) holds for this path, we need to prove that it also holds for the path: S-$R_1$-$R_2$-…-$R_k$-$R_{k+1}$-C.

Let $\min(b_1,b_2..,b_{k+1}) = b_w$.

When an additional node is inserted into the path, it introduces one additional link. Let $b_{k+2}$ be the bandwidth of this link. If $b_{k+2}$ is greater than or equal to $b_w$, equation (1b) does not change. However, when $b_{k+2}$ is less than $b_w$, in order for C to receive loss-free transmission, the maximum encoded rate is bounded by $b_{k+2}$, which is min $(b_i)$. Hence we have,

$$r_c=\min(b_i)+ ((\min(b_i) *\delta)/T). \quad \square$$

Theorem 1 provides the *upper bound* for the deliverable rate at a client as the client's path is considered in isolation. When we consider a multicast tree as in Figure 3, limited bandwidths of shared links will further constrain the rates delivered to the clients.

Finding the deliverable rates at the clients can be posed as an optimization problem with the **Objective function**: minimize $\Sigma i\ (\alpha - x_i)^2$, where $x_i$ s are the stream rates flowing through links $l_i$s subject to:

(i) **Rate constraints**: $\forall_i$, $x_i <= \alpha$,

(ii) **Delay tolerance constraints**: $\forall_i$ $L_i <= d_i$, where $L_i$ is the latency incurred in the path from S to $C_i$, due to buffering $L_b$ and transcoding $L_t$. As per the property of streaming, since the bits are streamlined, $L_b$ depends on $b_w$, the bandwidth of the weakest link, in the path from S to $C_i$. Let $r_i$ be the delivered stream rate and T be the play out duration of the file. $L_b$ is given by (see [3]):

$$L_b = ((r_i- b_w)/ b_w)*T \qquad \textbf{(2)}$$

Assuming a constant transcoding delay,

$$L_i= L_b+ L_t \qquad \textbf{(3)}$$

(iii) **Transcoding constraints**: these depend on the transcoder placement strategy, considered in the following sections. In terms of optimization constraints, it suffices to say that when a transcoder cannot be placed at a node, the incoming rate must be equal to the outgoing rate. If this constraint is not specified, it is tantamount to the node having the capability to transcode.

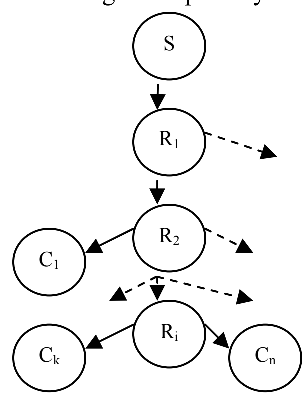


**Figure 3: Multicast tree**

Given the complexity of this optimization problem, we have come up with heuristic based algorithms that use the client specified deadlines proactively in making all the transcoder deployment decisions. See [3] for details.

## 3. Analysis of resource placement

In this section, we present two theorems that provide insight for reducing the complexity of the heuristic based algorithms. Specifically they help constrain the search space for the placement of transcoders.

Consider a series of nodes $n_1$, …, $n_k$ such that each node has a single outgoing link connecting it to the next node. We term such a series of nodes a *string*. $n_1$ can be the source or a relay node. $n_k$ can be a client, or it can be a relay node having multiple outgoing links.

**Theorem 2:** For any string $n_1$, n2 ,…,$n_m$, let $r_{in}$ be the rate flowing into $n_1$. Let $r_{out}$ be the rate delivered to $n_m$.

If $r_{out} < r_{in}$, i.e., transcoders are needed in this string,

(i) it is sufficient to place a transcoder at $n_1$
(ii) if $n_m$ is a client, $r_{out} = r_m$, the deliverable rate at $n_m$ determined using Theorem 1.

if $n_m$ is the root of a subtree, $r_{out} = \max(r_i)$, where $r_i$s are the deliverable rates at clients in the subtree rooted at $n_m$, determined using Theorem 1.

**Proof:** We consider part (i) of the theorem. Given $n_1$, $n_2$, …, $n_m$. Let $n_i$, $n_j$, $n_k$ be some intermediate nodes transcoding the stream from $\beta$ to $\beta_1$ to $\beta_2$ to $\beta_3$ where $\beta > \beta_1 > \beta_2 > \beta_3$. Let $n_k$ be the last node with transcoding capability, i.e., nodes $n_{k+1}$ to $n_m$ do not have transcoders. $n_m$ receives the stream encoded at rate $\beta_3$. The stream encoded at a higher rate $\beta_2$ flowing from $n_j$ to $n_k$ is not useful to any client. So, $n_j$ can transcode the stream at the rate $\beta_3$ and send it to $n_k$. Similarly the rate $\beta_1$ which is greater than $\beta_2$ and $\beta_3$ flowing from $n_i$ to $n_j$ is not useful. Hence $n_i$ can transcode the stream to $\beta_3$. Extending the same logic, instead of sending a stream with higher encoded rate from $n_1$ to $n_i$, $n_1$ can transcode the stream to $\beta_3$ which will flow across the string to provide the optimal encoded rate to $n_m$. Thus, only one transcoder is needed at the first node of a string to provide the stream encoded at the optimal rate at the last node of the string.

Considering part (ii) of the theorem, when $n_m$ is a client, it is trivial that $r_{out}=r_i$, as $n_m$ is the only client served by $n_i$. When $n_m$ is the root of a subtree, let $C_1$, $C_2$, and $C_3$ be clients served by $n_m$. Let $r_1$, $r_2$ and $r_3$ be the deliverable rates as determined by Theorem 1 at $C_1$, $C_2$, and $C_3$ respectively such that $r_1>r_2>r_3$. Suppose $r_2$ is chosen as $r_{out}$, deliverable rate at $C_1$ has to be compromised as $r_2$ is the best rate that can be delivered to any client in that subtree. Note that using the transcoding capability at $n_m$, $C_3$ can be serviced with $r_3$. Thus, given that a stream can only be transcoded down to a lower bit rate, in order to serve *all* the clients with their maximum deliverable rates, $r_{out} = \max(r_i)$. □

We use Theorem 2 to eliminate redundancy while deploying transcoders in the multicast tree. According to this theorem, it is sufficient to enable nodes at the start of strings with transcoding capability. In other words, transcoding capability is not required at nodes having only one link emerging out of them. This leads to the following *corollary*: In a multicast tree, placing transcoders *only* at relay nodes having multiple outgoing links yields same deliverable rates at clients as placing transcoders at *all* relay nodes.

We now consider a multicast tree rooted at M and the subtrees originating from it. Note that each subtree can be: (i) one-hop, to a client $C_i$ (ii) multiple hop, to a client $C_i$, or (iii) a tree, with multiple clients having at least one shared link in their path. By Theorem 2, for subtrees of type (i) and (ii), placing a transcoder at the root M is sufficient to provide optimal rates at the clients. We derive the deliverable rates at clients for type (iii) using Theorem 3, when only the root of the tree is capable of transcoding.

**Theorem 3:** Consider a subtree $S_p$ rooted at node P. Let $C_1$, $C_2$,.., $C_n$ be the clients in $S_p$ having delay tolerance values $\delta_1$, $\delta_2$, …., $\delta_n$. Let $r_1$, $r_2$, …, $r_n$ be the loss-free deliverable stream rates at $C_1$, $C_2$, ..., $C_n$ taking into account their $\delta$ values. If the only transcoder in $S_p$ is placed at P, deliverable stream rates at $C_1$, $C_2$, ..., $C_n$ are:

$r_1 = r_2 =, ..., = r_n = \mathrm{Min}\,(r_1, r_2, ..r_n)$, when the shared link bandwidth can support utmost one stream.

**Proof:** Let the deliverable rates calculated using $\delta$ values of $C_1$, $C_2$, .., $C_n$ be such that $r_1 > r_2 >..> r_n$. Given $C_1$, $C_2$, .., $C_n$ share at least one link. Since P is the only node capable of transcoding, assuming that bandwidth of at least one of the shared links is less than $(r_{n-1}+r_n)$, only one stream can flow through it. In order to provide loss-free transmission to *all* clients, this rate has to be $\min(r_1, .., r_n)$, i.e., $r_n$. □

According to Theorem 3, the client with the lowest deliverable rate determines the rates received by other clients in that subtree sharing at least one link, when only the root of the subtree is capable of transcoding. This leads to the following *corollary*: Given a subtree with clients having similar deliverable rates, it is sufficient to place a transcoder at the root of the subtree. This corollary helps in deciding optimal placement of a limited number of transcoders considering the heterogeneity of deliverable rates of clients in a subtree.

In the next section, we outline our ongoing work on finding optimal transcoder placement strategies for different network scenarios using a content dissemination model.

## 4. Applying our analysis

Given a heterogeneous dissemination network, without loss of generality, we model the network as having two parts:

(i) The CSP's *core network*, controlled by the CSP where typically links are provisioned. We term the edge nodes in the core network as *region nodes*.

(ii) The access network through which the clients connect to the region nodes. Typically bottlenecks occur here. CSP may not have control over the access network nodes.

Note that we include region nodes as part of both the core and the access networks.

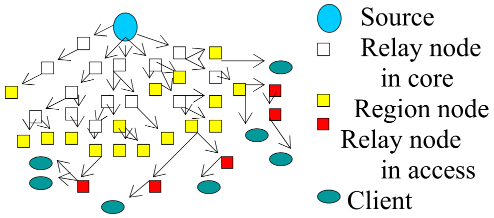


**Figure 4: A content dissemination network**

Given a network model as shown in Figure 4, the following options exist when we consider transcoder placement:

(1) Considering only the core network up to the region nodes:

- at S, and/or
- at the relay nodes within the core and/or
- at the region nodes

(2) Considering only the access network, originating at the region nodes:

- at the region nodes, and/or
- at the relay nodes within the access network.

(3) At nodes within the combined (core+access) network. This essentially is a combination of (1) and (2).

Note that even though some of the options given above overlap, we propose the notion of core and access networks to understand the possibilities in a CSP's perspective, given limited resources and restricted access to some network nodes.

The following scenarios and hypotheses are being investigated to get insight into optimal placement of transcoders:

*When the CSP does not have control over the access network nodes*, i.e., transcoders cannot be placed within the access network:

--When the core is well provisioned, and the number of region nodes is small: enabling region nodes with transcoding capability maximizes benefit.

--In other cases, transcoding at relay nodes with multiple outgoing links within the core network is most beneficial.

*When CSP has control over the access network*:

-- If the core is well provisioned, transcoding at relay nodes in the access network serving multiple clients can be beneficial as typically "last mile" problems impose bandwidth bottlenecks.

Our goal is to recommend a particular placement strategy that would maximize the benefit for the CSP, given a content dissemination network with specific link bandwidths and client arrival patterns for various pre-scheduled streams.

## 5. Other work-in-progress

For a given set of client requests with specific requirements, we consider the following parameters that impact the resources needed and placement of these resources:

(i) *Network Topology*: static or dynamic

(ii) *Bandwidths*: constant or varying

(iii) *Client Arrival***:** known or predicted

(iv) *Number of Simultaneous Streams*: one or many

(v) *Number of Content Files accessed*: one or many

In the first phase of our work we dealt with a static multicast tree with constant bandwidth links and known clients when all clients are synchronously serviced with a single stream. We extend our work considering the following cases: (i) when client requests for the same content arrive dynamically. (ii) when link bandwidths vary during synchronous transmission of a stream.

In (i) synchronous streaming is not possible. When the first client requests for the stream, S initiates the stream. Any requests made during the first client's transmission, from other clients in the same subtree have to wait for the shared links to be freed. In [4], we propose a Hybrid Streaming Mechanism (HSM) where contents are dynamically downloaded to appropriate relay nodes which are enabled with streaming capability. When the CSP's core network is highly provisioned, the links would be underutilized when S is the only node capable of streaming. By deploying additional streaming servers at chosen relay nodes, shared links can be freed earlier, thus increasing the number of serviced clients. The streaming servers also temporarily cache the contents for serving future requests. In (ii) we relax the assumption of static bandwidth over the transmission period.

## References


[1] J. Liu, B. Li, Adaptive Video Multicast over the Internet, IEEE Multimedia, January-March 2003.

[2] S.Krithivasan, S.Iyer, Enhancing QoS for Delay-tolerant Multimedia Applications: Resource utilization and Scheduling from a Service Provider's Perspective, Student workshop, IEEE INFOCOM , April 2006.

[3] S. Krihtrivasan, S. Iyer, Strategies for efficient streaming in delay-tolerant multimedia applications, accepted paper, IEEE ISM 2006, December 2006.

[4] A.Th. Rath, S.Krithivasan, S.Iyer, HSM: A Hybrid Streaming Mechanism for delay-tolerant multimedia applications, accepted paper, MoMM 2006, December 2006.