# On-Demand Multicast Routing Scheme and Its Algorithms

Te-Chou Su and Jia-Shung Wang
Department of Computer Science, National Tsing Hua University
Hsinchu, Taiwan 30043
phone: 886-35-731067, fax: 886-35-723694
email: jswang@cs.nthu.edu.tw

## *Abstract*

*Multicast communication is one of the major techniques to share the transmission cost. However, in the typical multicast service, all destinations are expected to receive the same multicast stream from the source at the same time. Unfortunately, this is a severe restriction for some on-demand multimedia applications, such as video-on-demand (VOD), in which customers' requests are expected at various time. Therefore, only few customers can be served in the same multicast group; thus additional bandwidth is required. In this paper, we propose a new multicast communication scheme to allow the destinations accessing the same multicast stream at different time by buffering technique and hence tremendously reduce the communication bandwidth. We call this type of communications as the on-demand multicast communications. We also define the optimization problems of the on-demand multicast routing and give a theoretical analysis. A greedy algorithm for this problem is also presented and had been implemented on our VOD systems.*

## 1. Introduction

Multicast communications is an important technique to share the transmission cost for multimedia applications, such as distance education, teleconferencing, and video-on-demand (VOD) [1, 2]. In this communication mode, the same data stream are sent simultaneously from the source to multiple destinations by setting up a routing path to connect the members of the multicast group. Therefore, designing a cost-effective multicast routing algorithm with limited transmission delay is one of the fundamental issues [3, 4, 5, 6]. The commonly used approach for this problem is the tree-based routing which can share many links in transmitting the data stream to the destinations. In general, this style of multicast concept assumes all destinations receive the transmitted data at approximately the same time. However, this concept severely restricts the on-demand multimedia applications [7, 8] because the customers viewing the same movie at different time can not be served in the same multicast group. Therefore, individual multicast group is required for each different request time. This is not efficient. In this paper, we propose new multicast communication scheme in which the destinations requesting the same movie at different time can be served in the same multicast group. We call this type of multicast communications as the *on-demand multicast communications*. In addition, this scheme also provides the VCR-like interactive control between the source and the individual destination. The comparison between the on-demand multicast tree and the conventional one is demonstrated in Fig. 1. In Fig. 1a, all destination nodes receive the same stream at the same time (ignoring the transmission delay). In contrast, as indicated in Fig. 1b, the on-demand multicast scheme can build up a multicast tree allowing different access time.

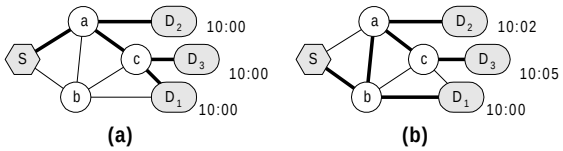


**Fig. 1.** The conparison of the conventional multicast tree and the on-demand multicast tree.

In this paper, we design a simple buffering mechanism to construct the on-demand multicast trees. Based on this mechanism, we define an optimization problem for this on-demand multicast routing and give a theoretical analysis on this problem. An optimal greedy algorithm in a fully connected topology is also presented. In addition, the transmission load is also proved balance and light in our algorithm. To verify the performance of the on-demand multicast communication, we implement it on our VOD systems, which is also described in this paper.

The rest of this paper is organized as follows. Section 2

presents the basic mechanism to construct the on-demand multicast trees. Section 3 defines an optimization problem for the on-demand multicast routing and presents an optimal greedy routing algorithm for this problem. In Section 4, we discuss and simulate the application of the on-demand multicast scheme on the VOD systems. Conclusions are finally made in Section 5.

## 2. The construction of the on-demand multicast trees

Before describing how to construct an on-demand multicast tree, we first introduce a mechanism to allow the destinations accessing the same multicast stream at different time, using the buffering technique. In a tree-based multicast routing approach, transmission delay may not be avoided when messages are passed through or replicated on the routing nodes in the multicast tree. Thus, reducing the transmission delay is an important design issue of a routing algorithm. However, this is not our concern, in contrast, we deliberately intent to delaying the incoming stream a specific time to satisfy the specific conditions. In our approach, the incoming stream was stored in the routing node and then sent out at various time to produce outgoing streams at different time. Fig. 2 shows an example of this buffering technique. In Fig. 2a, the incoming stream is flowed out immediately, no transmission delay. In Fig. 2b, to simultaneously provide another stream with 2 seconds transmission delay, the incoming stream was stored in a pipe buffer and then sent out after 2 seconds. Likewise, as shown in Fig. 2c, if the size of the pipe buffer can store 5 seconds incoming stream, another stream with 5 seconds transmission delay can be provided in this routing node.

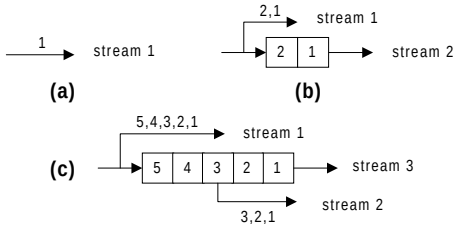


**Fig. 2**. The buffering technique.

Based on this technique, the construction process of the on-demand multicast trees can be explained by an illustrative example of what happens when a destination node issues a request for receiving the same stream. For explanation, we partition the requested stream into consecutive segments, $s_1, s_2, \ldots, s_n$, where $s_i$ represents the $i^{th}$ minutes segment. Therefore, a two hours video stream is represented by segments $s_1, s_2, \ldots, s_{120}$.

Consider the example shown in Fig. 3. In Fig. 3a, the destination $D_1$ issues the first request at 10:00 and a direct path was established along the bold line from the source node $S$ to node $D_1$. Fig. 3b shows the second request from node $D_2$ at 10:02. At this moment, the segment $s_3$ is ready to flow into the path from node $b$ to node $D_1$ but the segment $s_1$ and $s_2$ had gone (if the other transmission delay factor can be ignored). Therefore, the node $b$ and $D_1$ can provide the data segments started from $s_3$ from now on. Thus, node $D_2$ can immediately make a route to get the segments $s_1$ and $s_2$ from the source node $S$ and then get the rest of segments from node $b$ or $D_1$, but two minutes later. In this example, node b was chose. Notice that the first route is temporary and will be released two minutes later when the segment $s_1$ and $s_2$ are received, therefore, making this route efficient is not as critical as the other routes. On the other hand, to get the rest of segments from node $b$ two minutes later, a path must be established from node $b$ to node $D_2$ with delay of two minutes, which means two minutes buffer is required along this path. In this example, they are provided by node $a$. In fact, there is a small trick may possibly reduce the duration of the temporary route, even do not require this route. The trick is to pre-buffer the data as much as possible in the available buffer on each node along the routing path. Although this pre-buffered data maybe useless in the future, it is not a problem since this buffer is still available for other routings. For example, if node $b$ pre-buffer 4 minutes stream at the first route, the temporary route of destination $D_2$ is not necessary since the segment $s_1$ and $s_2$ are still retained in node $b$.

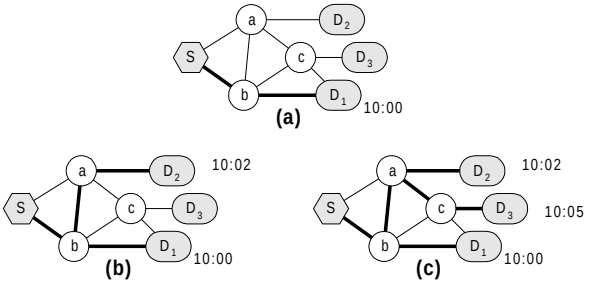


**Fig. 3.** An example of constructing on-demand multicast tree.

When the next request comes at 10:05, as shown in Fig. 3c, we need only 3 minute buffer at node $c$ to establish a path from node $a$ to node $D_3$ with delay of 3 minutes. In this case, the temporary path from node $D_3$ to the source node $S$ will be released 3 minutes later. Therefore, following these procedures, an on-demand multicast tree can be established.

Notice that, in an on-demand service, the requested stream will be received in a few seconds after the request was issued. Thus, in the rest of the paper, we assume that there is no transmission delay between the source node and the destination nodes if our buffering technique does not be employed.

In addition, the interaction between the source node and the destination nodes can be simply achieved by the on-demand multicast routing scheme. For example, consider the case that node *D* requests a stream at 10:00, pauses at 10:04 and resumes at 10:08. When node *D* resumes, it is equivalent to issue a new request at 10:08 but the requested stream is started from the segment $s_5$. Therefore, we need a node to provide the required stream for node *D*. Let us consider the following 3 routing cases.

The simplest case is to find a node providing the segment started from $s_5$ immediately. In this case, we just make a route from this node to node *D* with no transmission delay. The second case is to find a node in which the segment $s_5$ had been sent out and not stored in its buffer, for example, only the segments started from $s_7$ are provided. Note that this case is similar to the previous construction process. Therefore, we simultaneously make a temporary route to get the segments $s_5$ and $s_6$, and then make another route from the selected node with delay of two minutes to get the rest of segments started from $s_7$. In the last case, if no node is satisfied the previous two cases, we route this new request directly to the source node.

Similarly, the fast-forward operation can be made by the same idea. At first, because of the data rate is different, a temporary route is needed to provide the fast-forward stream and the previous routing idea can be applied when this operation is finished. The other operations (ex, stop, jump-forward, jump-backward, slow-down, …) have the same property as the previous discussed one, we skip it here.

## 3. On-demand multicast routing problem and algorithm design

In this section, we formally define the problem of constructing the on-demand multicast routing trees. In addition, a greedy algorithm for this problem on a fully connected topology is presented also.

### 3.1. On-demand multicast routing problem

The on-demand multicast routing problems may have a variety of contexts, particularly in the domain of resource allocation. We concern the following two versions which are focus on the number of required multicast streams since the less streams required, the more transmission bandwidth shared.

*I. Off-line on-demand multicast routing problem*: Given a network topology *G*, a source node *S*, and a set of destination nodes labeled with a time for issuing the request to receive the same stream, each node *i* contains a buffer of size $B_i$, and each link *j* has a bandwidth capacity $C_j$, what is the minimum number of required on-demand multicast trees to serve all the given requests?

*II. On-line on-demand multicast routing problem*: The definition is similar to the off-line version except that the time for issuing the request is not pre-defined in advance.

Clearly, the process of on-line constructing the on-demand multicast trees is very complicated since several constraints should be considered together. For example, there may exist many routing paths satisfying the constraints, and even in a path, there still are many different buffer allocation combinations available. Furthermore, each routing path will affect the following routing situations. Therefore, how to select one of the paths and to determine the buffer allocation plan is quite complicated. We think such routing problem is NP-complete since the typical multicast problem (without buffer allocation) on the general network is NP-complete [11]. In this paper, we only discuss this problem on the complete graph. For the off-line problem, the minimum number of required on-demand multicast trees over the complete graph could be derived in the following analysis.

Let *G* denote a complete graph with *n* nodes. Let *B* denote the total buffer size in all routing nodes and *r* is the number of destination nodes. Let $D_i$ represent the destination node issuing the $i^{th}$ request at time $t_i$, without loss of generality, we assume $t_1 < t_2 < \ldots < t_r$. Clearly, if the on-demand multicast tree is constructed by making a route connecting $D_i$ to $D_{i-1}$, least buffers used than all other routing paths. Let $d_i$ be the minimum buffer required to connecting $D_i$ to $D_{i-1}$, which is equal to $t_i - t_{i-1}$. Note that the first request must route to the source, thus, $d_1=0$. The minimum number of the required on-demand multicast trees can be calculated in two cases.

*Case I* - $\sum_{i=1}^{r} d_i \le B$: The minimum required buffer size is less than the total amount of buffers, clearly, only one on-demand multicast tree is required.

*Case II* - $\sum_{i=1}^{r} d_i > B$: The total amount of buffers is not sufficient, thus, more than one on-demand multicast tree is required. Let $d_2', d_3', \ldots, d_n'$ be the sorted sequence of $d_2, d_3, \ldots, d_n$ in nondecreasing order, that is, $d_2' \le d_3' \le \ldots \le d_n'$. Let $k = \max\{r \mid \sum_{i=2}^{r} d_i' \le B\}$ and we define *K* be the set of the destination nodes corresponding to $d_2', d_3', \ldots, d_k'$. In the following, we prove the required number of the on-demand multicast trees is $n-k+1$ in two steps.

At first, the $n-k+1$ is the least possible number of the required on-demand multicast trees satisfying the buffer constraint. If this is not true, there is a routing result in

which using less on-demand multicast trees and still satisfying the buffer constraint. This is impossible by the definition of $k$.

Second, we present an off-line routing algorithm to construct only $n$-$k$+1 on-demand multicast trees under the buffer constraint. This off-line routing algorithm is simple and explained as follows. For each destination node in the set $K$, make it join the existing multicast tree and thus consumes the buffer of size $\sum_{i=2}^{k} d_i^{'}$ which is less than or equal to $B$. On the other hand, all other $n$-$k$ destination nodes initiate multicast trees. Thus, there are $n$-$k$+1 multicast trees in this routing method.

*Theorem 1: Let G be a fully connected topology and B be the total amount of buffers. The number of minimum required on-demand multicast trees is r – k + 1, where r is the number of destination nodes and k is the largest value such that* $\sum_{i=2}^{k} d_i^{'} \le B$, *and* $d_i^{'}$ *is the sorted sequence stated above.*

### 3.2. The greedy algorithm for on-line problem

The key idea of our greedy routing algorithm for the complete graph is to route each request to the last (just completed) request node and dynamically reroute some requests which not in the set $K$ of the off-line scheme stated in section 3.1. Our on-line algorithm is depicted in Fig. 4. This algorithm consists of two procedures: the procedure *routeRequest*() route the current request to the last request node, and the procedure *rerouteDestination*() dynamically reroute some requests to satisfy the required results. Let routing $D_j$ require $r_j$ buffer size, then $r_j = 0$ if $D_j$ directly connects to the source, otherwise, $r_j = d_j$. Let $B$ is the current total buffer size, the detail of this algorithm is explained below.

```
1.  procedure routeRequest(Dj)
2.  {
3.     if (B > rj)
4.        arbitrarily make a route from Dj to Dj-1.
5.     else
6.        if (rerouteDestination() == true)
7.           arbitrarily make a route from Dj to Dj-1.
8.        Else
9.           make a route directly connecting to the source
10. }
11. procedure rerouteDestination()
12. {
13.    if (rm = max {r1, .. , rj-1} ≤ rj)
14.       return false;
15.    else {
16.       reroute Dm directly connecting to the source
17.       return true;
18.    }
19. }
```

**Fig. 4.** The on-demand multicast routing algorithm.

At first, line 3 tests whether the total buffer is sufficient. If it is true, we make an arbitrarily route from $D_j$ *to* $D_{j-1}$ by connecting a series of nodes to sastify the buffer requirement. This routing path can be found since the topology is fully connected. In addition, we exhaust each node buffer as possible. This concept is illustrated in Fig. 5. Node $D_j$ issues a request at 10:25 and node $D_{j-1}$ is the last request at 10:00. The remaining buffer of node $D_{j-1}$ is 10 minutes and no buffer is available in node $D_j$. In this situation, we make a route connecting node $a$ (with buffer 5 minutes), node $b$ (with buffer 8 minutes) and node $c$ (with buffer 2 minutes) to satisfy the buffer requirement. The buffer in nodes $D_{j-1}$, $a$ and $b$ are exhausted, but there are still 5 minutes buffer in node $c$.

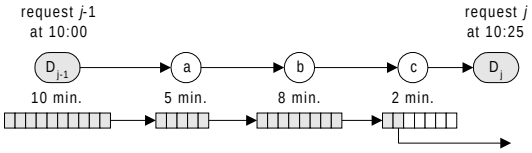


**Fig. 5.** An example of buffer allocation in a routing path.

On the other hand, if the answer in line 3 is false, that is, the remaining node buffer is insufficient to serve the current request, the procedure *rerouteDestination*() is executed. To achieve the same results as the off-line routing algorithm, this procedure only reroute the node not in the set $K$ and then release the occupied buffers for other routings.

In line 13, if the largest consumed buffer is less than $D_j$'s required buffer, line 14 just return false and no buffer is released. Thus, line 9 is executed. On the other hand, we reroute $D_m$ directly to the source and release the occupied buffer size $r_m$ in line 16. Note that this released buffer can be used in routing node $D_j$ in line 7. This rerouting process is illustrated in Fig. 6.

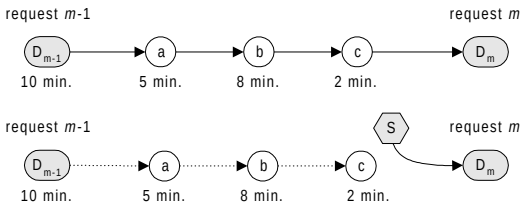


**Fig. 6.** An illustrative example of the rerouting process.

In the following, we prove that, at final, only those destination nodes in the set $K$ are connected to the existing on-demand multicast trees and the other nodes initiate new on-demand multicast trees.

For each destination node $D_j$, if the procedure *rerouteDestination*() was successfully executed, the node consuming buffer of size $p$=max{$r_1$, …, $r_{j-1}$} will be rerouted and the node $D_j$ will connect to the existing multicast tree. Note that $p$=max{ $r_1$, …, $r_{j-1}$} and $p$>$r_j$, i.e.

$p$=max{ $r_1$, …, $r_{j-1}$, $r_j$}, the subsequent released buffer must be less than or equal to $p$, that is, $p_1 \geq p_2 \geq p_3 \geq \ldots \geq p_r \geq \max\{r_i\}$, for all $i$, and $p_k$ is the $k^{th}$ released buffer size. Thus, at final, only those destination nodes in the set $K$ (the nodes corresponding to the smallest $r_i$) will connect to the existing multicast tree and the other nodes will be rerouted to the source by our procedure. This proves our on-line greedy algorithm is optimal.

In the following, we show that the required link capacity of our algorithm is at most 2 times of the stream bandwidth. To attain this result, we assume that all requests occur at different time. This assumption implies that an established path must consume some buffer. In addition, the link is assumed to be full duplex.

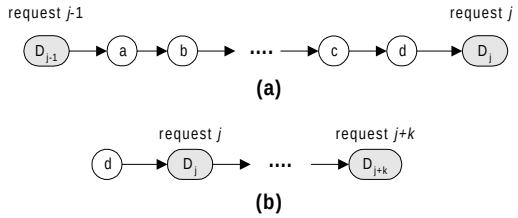


**Fig. 7.** An example for counting the required link capacity.

Consider a routing path from node $D_j$ to node $D_{j-1}$ as shown in Fig. **7**. As mentioned before, the procedure *routeRequest*() exhaust the node buffers on the routing path as possible, thus, only the last two nodes could remain buffers. In Fig. **7**a, the buffers from node $D_{j-1}$ to node $c$ are exhausted and there may have some buffer remained in node $d$ and node $D_j$. Now please consider the following two cases.

Case 1. Consider the links from node $D_{j-1}$ to node $c$. Because no buffer is available on these nodes, these links cannot be used again. The other possible use of this link is that the consecutive nodes issue consecutive requests before this routing. For instance, node $a$ issues the $i^{th}$ request and node $b$ issues the $(i+1)^{th}$ request. Therefore, the required capacity of these links is at most twice of the stream bandwidth.

Case 2. Consider the link connecting node $d$ and node $D_i$. Since node $d$ remains some available buffer, this link can be used again. However, if this link is used again, the node $d$ must be an intermediate node but not be the last two nodes on the routing path. This is the case 1 and node $d$'s buffer will be exhausted (see Fig. **7**b). Therefore, the link connecting node $d$ and node $D_j$ is traversed at most two times.

Therefore, the required link capacity is at most two times of the stream bandwidth.

## 4. Applying the on-demand multicast routing on the switch-based VOD systems

The on-demand multicast communication is suitable for VOD applications. It tremendously reduces the transmission cost and also provides the VCR-like interactions for individual customer. Although the current router can not support this type of communications, we can use workstations as the routing node just like the most Mbone [12] routers. In addition, the same concept can be applied on the switch-based VOD systems in which network is organized as a complete graph. Therefore, our greedy algorithm can be directly applied on this network. In fact, we had implemented this on-demand multicast routing algorithm on the NTHU VOD systems [9].

The NTHU VOD architecture consists of one *manager* PC (Microsoft™ Windows 95 + Web server), some *video servers* (Microsoft™ Windows NT + SCSI HD), the transport system (Ethernet switch and hubs) and a number of customer PCs (Microsoft™ Windows 95). The underlined network is composed of the tree-structured switches and the scale of this architecture can be large enough to provide thousands of customers by current commercial products (see Fig. 8). For example, if root switch has 120 ports (Lucent™ P550™ Cajun™ switch) and level two switches have 48 ports each, 120×48=1340 customers can be served. Furthermore, if level two switches have 120 ports each, the total served customers are 120×120=13200.

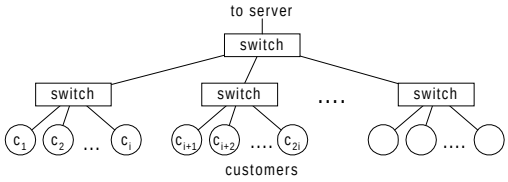


**Fig. 8.** The tree-structured VOD architecture using the switch.

In our VOD systems, every customer PC is also a video server since the video stream can be buffered and then send out later. In our experiences, if the memory is used as the cache buffer, at least 30 customers can be supported in a Pentium PC with 100 Mbps Ethernet. If hard disk is adopted, fewer customers are supported but more buffers are provided. By carefully switching the cache buffer between the memory and the hard disk, each PC can provide several minutes buffer for MPEG-1 stream. In addition, our implementation results show 100 Mbps Ethernet can transmit at least 50 MPEG-1 stream. Therefore, our tree-structured architecture is reasonable and practical since the required link capacity of our algorithm is at most two times of stream bandwidth.

In addition, we simulate our algorithm on complete

graph and the simulation parameters are assumed as follows. There are 10 video programs and the request frequency follows the Zipf's distribution [10]. There are 100 requests and the arrival rate is a Poisson distribution with mean 60 seconds. The simulation results are shown in Fig. 9. Fig. 9a shows the impact of the node buffer size on the number of the required multicast trees. Simulation shows that more buffer provided in each node, less multicast trees are needed. Notice that the required multicast trees are much less than the requests. Fig. 9b shows that most of the buffer resources are used by the popular movies, and the requests for the unpopular movies (the video program from 4 to 10) almost initiate a multicast stream. Thus, the buffer utilization is efficient.

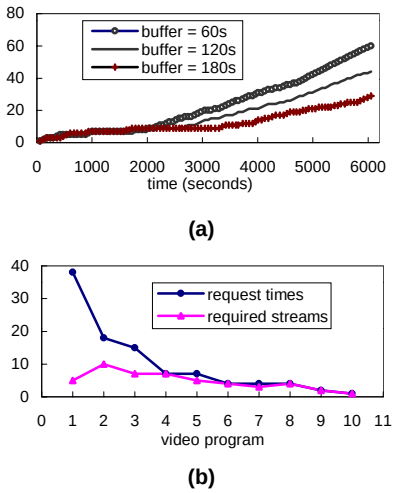


**Fig. 9.** The simulation results.

## 5. Conclusion

In this paper, we present an innovative on-demand multicast communication scheme and explore the benefit in the on-demand multimedia applications. Using the buffering technique, our scheme allows destinations accessing the same multicast stream at various time. Furthermore, applying our scheme on VOD systems, the distributed customer's buffer resource can be aggregated and shared. If the underlined network is fully connected (e.g., Ethernet switch), only one on-demand multicast tree is needed for a 2 hours video program for all customers at any time, when the total amount of distributed buffers is larger than this video length. For example, only 1 minute (about 12 M bytes for MPEG-1) buffer is required on each customer sites if there are 120 users on this system. This tremendously reduces the load on the video server and the transmission cost. In addition, our greedy algorithms dynamically adjust the buffer utilization in a quite efficient manner, that is, the buffer will be shared only by popular movies.

In the on-demand multicast routing problem, one of the major challenges is to design the routing algorithm on the general network topology. We think this is a NP-Complete problem. As future works, we plan to study the complexity of this problem and propose efficient algorithms on several particular graphs, such as tree, interval graph, chordal graph, etc.

## References


[1] Kevin C. Almeroth, Mostafa H. Ammar, "The use of multicast delivery to provide a scalable and interactive video-on-demand service." *IEEE J. Select. Areas Commun.*, vol. 14, no. 6, pp. 1110-1122, Aug. 1996.

[2] Wanjiun Liao, Victor O.K. Li, "The split and merge (SAM) protocol for interactive video-on-demand systems." *IEEE Multimedia*, vol. 4, no. 4, pp. 51-62, Oct-Dec 1997.

[3] K. Ravindran, T. J. Gong, "Cost analysis of multicast transport architectures in multiservice networks", *IEEE/ACM transactions on networking*, vol. 6, no. 1, Feb. 1998.

[4] Fred Bauer, Anujan Varma, "Distributed algorithms for multicast path setup in data networks." *IEEE J. Select. Areas Commun.*, vol. 4, no. 2, pp. 181-191, April 1996.

[5] V. P. Kompella, J. C. Pasquale, G. C. Polyzos, "Multicast routing for multimedia communication", *IEEE/ACM transactions on networking*, vol. 1, no. 3, June. 1993.

[6] Anees Shaikh, Kang Shin, "Destination-driven for low-cost multicast." *IEEE J. select. Areas Commun.*, vol. 15, no. 3, pp. 373-381, April 1997.

[7] Jean-Paul Nussbaumer, Baiju V. Patel, Frank Schaffa, P.G. Sterbenz, "Networking requirements for interactive video on demand." *IEEE J. Select. Areas Commun.*, vol. 13, no. 5, pp. 779-787, June 1995

[8] L.A. Rowe, D.A. Berger, J.E. Baldeschwieler, "The Berkeley distributed video-on-demand system." *Multimedia Computing: Proc. Sixth NEC Research Symp*.

[9] Jia-Shung Wang, et al., "The distributed hierarchical VOD system", technical report, National Tsing Hua university, 1997.

[10] Y.S. Chen, "Mathematical modeling of empirical laws in computer application: A case study." *Comput. Math. Applicat.* pp. 77-87, Oct. 1992.

[11] M.R. Garey, R.L. Graham, and D.S. Johnson, "The complexity of computing steiner minimal trees," *SIAM J. Appl. Math*, vol. 32, no. 4, pp. 835-859, June 1977.

[12] H. Eriksson, "Mbone: The multicast backbone," Commun. ACM, vol. 37, no. 8, pp. 54-60, Aug. 1994.