Algorithms for Server Placement in Multiple-Description-Based Media Streaming

Satyajeet Ahuja and Marwan Krunz
Department of ECE, The University of Arizona, Tucson, AZ 85721
{ahuja,krunz}@ece.arizona.edu

Abstract— **Multiple description coding (MDC) has emerged as a powerful technique for reliable real-time communications over lossy packet networks. In its basic form, it involves encoding media into m substreams that are routed independently towards a given destination. Each substream can be decoded independently and with every successful reception of a substream, the overall quality of the decoded signal is improved. In this paper, we consider the problem of placing a set of servers in the network such that a desired QoS can be provided to a community of clients that request MDC coded traffic. Specifically, we consider the *server placement* (SP) problem where the goal is to identify the "optimal" server positions and associated set of client-server paths such that if MDC content is placed at these servers a cost function that is a linear combination of average delay and path disjointness is minimized. We propose an MILP formulation and a highly efficient heuristic to solve the SP problem. Simulations are conducted to evaluate the performance of the proposed algorithm and compare it with the optimal solution obtained by using the MILP solution.**

Index Terms— **Content delivery networks, multiple description coding, path selection, path diversity, media streaming.**

I. Introduction

Content delivery networks (CDNs) have recently been the focus of intensive research (e.g., [8], [7]). This interest in CDNs stems from the ability of these networks to cope with various performance problems experienced in the Internet, including congestion and server overload. For example, CDNs can significantly improve the performance of a Web session by caching popular content on edge servers that are located close to end-users. CDNs were originally designed for the delivery of "offline" content, but recent efforts have also considered their use for streaming media [3]. In this paper, we consider the streaming of encoded media (e.g., video clips) using a CDN.

Many approaches for media streaming have been proposed in the literature (e.g., [6], [5]). One recently popularized streaming approach relies on multiple description coding (MDC) combined with multi-path diversity routing [2], [3], [8]. MDC is essentially a coding technique in which an input signal (video/audio) is coded into r bitstreams, referred to as descriptions. Each of these descriptions can be independently decoded at the receiver and can by itself provide a certain level of video quality. Furthermore, by embedding complementary information in each description, any subset of the r descriptions can be combined to achieve an improved level of quality. Hence, the quality of the video/audio stream improves with the number of successfully received descriptions. Because each description is independent of the other descriptions and descriptions are approximately equally important, the MDC technique is significantly different from the well-known layered approach. In the layered approach, the substreams (or layers) form a hierarchy. Data packets are differentiated in terms of their importance (most important packets or base-layer is given the highest priority and least important packets or enhancement-layer is given the lowest priority). In case of excessive packet loses, a minimum quality video is maintained by discarding the low-priority packets. High-priority packets are critical for the construction of the video. If they are lost, then the other low-priority packets are useless. MDC overcomes this shortcoming by providing a minimum quality video with the successful reception of any of the descriptions and with an improved quality video when the number of successfully received descriptions is increased. In its simplest form, the MDC encoder generates two descriptions, consisting of even- and odd-numbered frames, respectively.

Path diversity is a technique used in packet networks to send data simultaneously over multiple paths. The multiple paths may or may not originate from the same server. The use of multiple paths for streaming media has been shown to reduce packet loss [2]. MDC along with path diversity uses different paths to route different descriptions to a client. Blackouts due to a link or node failure along the path of one of the descriptions are avoided. The reduction in the packet loss through path diversity uses the fact that a link failure over two different routes are likely to be independent of each other. Architecturally, MDC content delivery is supported by a "logical" server, which receives requests for media and redirects the requests to multiple servers associated with the different descriptions. These servers may or may not be geographically co-located. Each physical server may have one or more of the descriptions. Placing the servers of different descriptions on different nodes (e.g., routers) helps in finding diverse paths for multiple clients. An intelligent placement of these servers will enable a given client to efficiently choose a subset of servers with maximally disjoint paths to the client.

In this paper, we study the server placement (SP) problem

This work was supported by NSF under grants ANI-0095626, ANI-0313234, and ANI-0325979. Any opinions, findings, conclusions, and recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

for supporting MDC-media streaming over CDNs. Our goal is to simultaneously maximize path diversity and minimize the average server-client delay by intelligently placing the servers over the network. In Section II, we define the SP problem. In Section III-A, we formulate it as a mixed integer linear program (MILP), and then in Section III-B we provide a highly efficient iterative algorithm to solve it. By solving the SP problem, we also determine the associated set of paths between a client and its corresponding servers. Simulations are conducted in Section IV to show the effectiveness of the proposed algorithm and compare it with the optimal solution obtained by using the MILP formulation. The paper is concluded in Section V.

II. Server Placement Problem

In this section, we study the basic problem of placing a set of multiple description (MD) servers within a CDN that supports a community of C clients. As shown in Figure 1, each server is assumed to have all the descriptions but a client can get only one description from each server. Later on, we relax this assumption and allow multiple descriptions to be streamed from the same server. Let the network consists of N nodes. We assume that the MD servers can be placed only at $S \subseteq N$ locations. We refer to these locations as *potential server locations*. The servers cannot be placed in other locations due to geographical and security reasons. Let $D(p)$ be the delay associated with a path p. Our objective is to find the best r locations in S such that the *average* delay associated with delivering

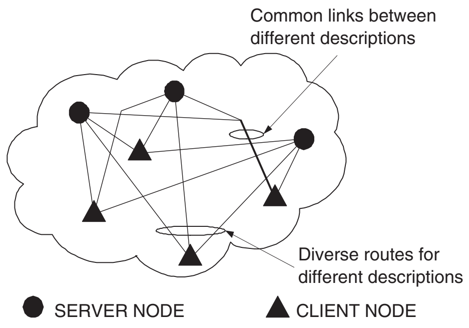

Common links between different descriptions
Diverse routes for different descriptions
● SERVER NODE
▲ CLIENT NODE

Fig. 1. Content delivery network with MDC media traffic.

the MDs from the servers to the set of C clients is minimized and the paths traversed by the MD traffic towards a particular client are as disjoint as possible. Notice that if the goal is to minimize the average delay only, then this problem can be easily solved by finding the shortest path between all pairs of nodes (i, j), $i \in C$ and $j \in S$ (e.g., Floyd-Warshall's algorithm [1]), and then choosing the r server locations that have the r-minimum average delays to all client nodes. Such a solution, however, ignores path disjointness, i.e., the traffic from different servers to a given client may be routed using many common links. A failure of a common link will result in missing multiple descriptions per client, significantly degrading the performance of the media playout. On the other hand, if we try to find server positions based on diverse paths only, then the problem can be solved by running a modified version of the max-flow algorithm that returns maximally disjoint paths for all possible combinations of server locations and client nodes. For each combination of servers, we would run the max-flow algorithm and find the maximally disjoint paths from the servers to the client. Among all possible combinations, we would choose the combination with maximum path disjointness or minimum number of common links between different descriptions to a client. Different descriptions will tend to take longer paths, resulting in a larger average path length. Although the failure of any link will not hamper the download rate, each description will traverse more hops and will consume excessive network resources. The worst-case complexity of such a solution is $\mathcal{O}(C(N,r)N^2 \log N)$.

Instead, our goal is to *jointly* optimize the average delay and path diversity. For that, we consider a cost function that is a linear combination of the two metrics. A network designer can choose an appropriate scaling factor (α) for this linear combination based on the design guidelines and service agreements. First, we study the problem by formulating it as an MILP. We then propose an efficient iterative algorithm to solve it. Before proceeding further, we define the diversity associated with a set of paths.

Definition 1— Path diversity: : The diversity for a set of paths is a measure of their disjointness. More specifically, let $\mathbb{P}(c_k)$ be the set of paths from the r server locations to a client c_k and let $l(p_1, p_2)$ be the number of common links between any two paths $p_1 \in \mathbb{P}(c_k)$ and $p_2 \in \mathbb{P}(c_k)$. We define the diversity of the set $\mathbb{P}(c_k)$ as $L(\mathbb{P}(c_k)) = \frac{1}{2}\left\{\sum_{(p_1,p_2)\in\mathbb{P}(c_k)\times\mathbb{P}(c_k),p_1\neq p_2} l(p_1,p_2)\right\}$.

A factor of $\frac{1}{2}$ is introduced because the summation includes twice the number of common links between each path pair. Intuitively, the lower the value of $L(\mathbb{P}(c_k))$, the more diverse is the set of paths in $\mathbb{P}(c_k)$. Let $L(C) = \sum_{c_k \in C} L(\mathbb{P}(c_k))$. We now formally state the server placement problem.

Problem 1— SP Problem: Let $G(V, E)$ be a network graph, where each link $(u, v) \in E$ is associated with a delay parameter $d(u, v)$. Let C be the set of client nodes, S the set of potential server locations, and α a scaling factor. For a path p, let $D(p) = \sum_{(u,v)\in p} d(u, v)$ be the delay associated with path p. Our goal is to find r nodes from the set S and an associated set of r paths $\mathbb{P}(c_k)$ for each client $c_k \in C$ from these chosen nodes such that, if MD servers are placed at these r nodes (one server per node) and each one of these servers delivers one description to the clients $c_k \in C$ using a path in $\mathbb{P}(c_k)$, then

$$\frac{(1-\alpha)}{r|C|}\left(\sum_{c_k\in C}\sum_{p\in\mathbb{P}(c_k)} D(p)\right) + \alpha\left(\sum_{c_k\in C} L(\mathbb{P}(c_k))\right)$$

is minimized among all possible choices of the r servers from the set S.

III. Proposed Solutions

We now present two different approaches to solve the SP problem. We first formulate the SP problem as an MILP and later present an algorithm to solve it.

Objective:

$$\text{Minimize}\left\{\frac{(1-\alpha)}{r|C|}\sum_{s_i\in S}\sum_{c_j\in C}\sum_{(u,v)\in E}x(s_i,c_j,u,v).d(u,v)+\alpha\sum_{c_k\in C}\sum_{(u,v)\in E}\sum_{(s_i,s_j)\in S^2}y(s_i,s_j,c_k,u,v)\right\}$$

Constraints:

$$\mathrm{C}_1:\quad \textstyle\sum_{v:(u,v)\in E}x(s_i,c_j,u,v)-\sum_{v:(v,u)\in E}x(s_i,c_j,v,u)\quad \begin{array}{ll}\le 1, & u=s_i\\ \ge -1, & u=c_j\\ =0, & \text{otherwise.}\end{array} \tag{1}$$

$$\forall s_i\in S,\ u\in V, \text{ and } c_j\in C.$$

$$\mathrm{C}_2:\quad 2y(s_i,s_j,c_k,u,v)-x(s_i,c_k,u,v)-x(s_j,c_k,u,v)\quad\le 0 \tag{2}$$

$$\forall(u,v)\in E,\forall c_k\in C,\forall(s_i,s_j)\in S^2, \text{ and } s_i\ne s_j.$$

$$\mathrm{C}_3:\quad y(s_i,s_j,c_k,u,v)-x(s_i,c_k,u,v)-x(s_j,c_k,u,v)+1\quad\ge 0 \tag{3}$$

$$\forall(u,v)\in E,\forall c_k\in C,\forall(s_i,s_j)\in S^2, \text{ and } s_i\ne s_j.$$

$$\mathrm{C}_4:\quad \textstyle\sum_{s_i\in S}\sum_{i:(i,c_j)\in E}x(s_i,c_j,i,c_j)\quad=r. \tag{4}$$

$$\mathrm{C}_5:\quad z(s_i)|C|-\textstyle\sum_{c_j\in C}\sum_{i:(s_i,k)\in E}x(s_i,c_j,s_i,k)\quad=0. \tag{5}$$

$$\text{where } |C| \text{ is the number of elements in the client set.}$$

$$\mathrm{C}_6:\quad \textstyle\sum_{c_j\in C}\sum_{(u,s_j)\in E}x(s_i,c_j,u,s_j)\quad=0,\ \forall s_i\in S. \tag{6}$$

$$\mathrm{C}_7:\quad \textstyle\sum_{s_i\in S}\sum_{(u,c_j)\in C}x(s_i,c_j,u,c_j)\quad=0,\ \forall c_j\in C. \tag{7}$$

$$\mathrm{C}_8:\quad \textstyle\sum_{s_i\in S}z(s_i)=\quad r. \tag{8}$$

Fig. 2. MILP formulation of the SP problem.

A. MILP Formulation of the SP Problem

Figure 2 depicts an MILP formulation for the SP problem. The formulation assumes that each server provides only one description to all clients. However, as we show later in this section, a slight modification can be made to incorporate multiple descriptions at a server location. The objective function of the MILP in Figure 2 is a linear combination of two terms. The first term is purely based on minimizing the average link delay while the second is purely based on maximizing path diversity. As explained before, the optimization of one term may result in a highly suboptimal solution for the other. In order to jointly minimize the two objectives, a network designer can choose an appropriate linear combination of the two objectives by fixing the scaling factor α. Let $s_i \in S$ be one of the selected server locations. Let $p(s_i, c_j)$ be the path chosen for delivering the description at server s_i to client $c_j \in C$. The binary variable $x(s_i, c_j, u, v)$ is set to one if $p(s_i, c_j)$ contains the edge (u, v); otherwise, it is set to 0. For each node $s_i \in S$, we set $z(s_i) = 1$ if an MD server is placed at s_i, and set it to zero otherwise. Finally, $y(s_i, s_j, c_k, u, v)$ is a binary variable that indicates whether the paths taken by descriptions from two different servers s_i and s_j to a client c_k have a common link $(u, v) \in E$. Essentially,

$$y(s_i,s_j,c_k,u,v)=x(s_i,c_k,u,v)\times x(s_j,c_k,u,v). \tag{9}$$

Constraint C_1 in Figure 2 is a flow conservation constraint for client c_j and server s_i. It limits the number of descriptions per server-client pair to one. Constraints C_2 and C_3 transform Equation 9 into an MILP. This transformation is needed because for an MILP formulation the constraints should be linear in the variables. As shown in Constraints C_4 and C_5, every client needs to get r descriptions, one from each chosen server. Constraint C_8 retricts the number of server locations to r. Due to the inequality in Constraint C_1, cycles may be formed at some nodes in S and some clients instead of a valid path between them. We call these cycles as self-cycles. In a conventional network flow formulation [1], flow conservation equations have equality constraints and hence these self-cycles are automatically avoided. Constraints C_6 and C_7 ensure that there are no self-cycles in the solution.

Incorporating multiple descriptions at a server location: Multiple descriptions can be supported at a server location by adding auxiliary nodes to this location (see Figure 3). These auxiliary nodes are then treated as potential server locations, with each location providing at most a single description.

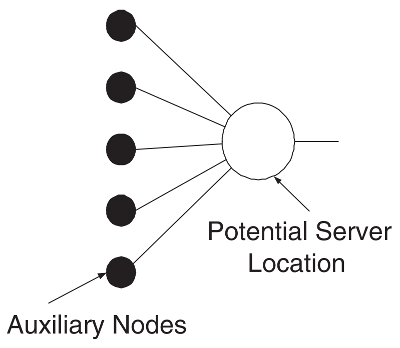


Fig. 3. Auxiliary node construction for supporting multiple descriptions at a single server location.

B. Placement Algorithm

The MILP formulation returns the optimal solution for the SP problem but the worst-case complexity is exponential in the number of constraints, making it practically infeasible for large networks. We now describe a highly efficient algorithm that is based on a greedy approach to solve the SP problem. The basic idea behind the algorithm is to find a subset $\lfloor \alpha r \rfloor$ of the $|S|$ potential locations whose locations are optimized for diversity. Then, among the remaining potential locations, choose $r - \lfloor \alpha r \rfloor$ servers that have the minimum average delay to client nodes. For the first $\lfloor \alpha r \rfloor$ servers, the algorithm first fixes a server location by choosing the node in S that has the minimum *average* delay to all clients. It finds such a node by running the Floyd-Warshall's algorithm [1], and then choosing a node with the minimum *average* delay per client. The algorithm then routes one unit of flow from the chosen server location to each client along their respective shortest paths. It then calculates the residual graph for each client $c_i \in C$. At any stage in the execution, $f_{c_i}(u, v)$ represents the amount of flow along an arc (u, v) in the residual graph for client c_i. During the execution of the algorithm, the residual graph for each client $c_i \in C$ is maintained separately. Let $w_{c_i}(u, v)$ be the weight of an edge in the residual graph of client c_i. Let $\Delta \overset{\text{def}}{=} N \max_{(u,v) \in E} d(u, v)$ be the maximum link delay associated with a simple path in the network. Initially, $w_{c_i}(u, v) = d(u, v)$, where $d(u, v)$ is the delay associated with the link $(u, v) \in E$. After the flow augmentation for any client c_i, $w_{c_i}(u, v)$ is updated by using the procedure presented in Figure 4. Notice that with every flow augmentation, the weight of a forward edge (u, v) carrying the positive flow increases by Δ. This ensures that if an augmenting path follows the edge (u, v), then there is no other path p between the given server-client pair such that $\max_{(i,j) \in p} f(i, j) = 0$. It also ensures that the path chosen by the flow has the minimum number of common edges with all the already established

```
1. If $w_{c_i}(u, v) > 0$,
2.     Set $w_{c_i}(u, v) \leftarrow (f_{c_i}(u, v) + 1) \times \Delta$
3.     Set $w_{c_i}(v, u) \leftarrow -(f_{c_i}(u, v) + 1) \times d(u, v)$
4. else // $w_{c_i}(v, u) < 0$
5.     Set $w_{c_i}(u, v) \leftarrow -(f_{c_i}(u, v) - 1) \times d(u, v)$
6.     Set $w_{c_i}(v, u) \leftarrow (f_{c_i}(u, v) - 1) \times \Delta$
7.     If $w_{c_i}(v, u) = 0$,
8.         Set $w_{c_i}(u, v) \leftarrow d(u, v)$
9.         Set $w_{c_i}(v, u) \leftarrow d(u, v)$
```

Fig. 4. Calculation of edge weights for the residual graph of client $c_i \in C$.

flows. Similarly, the weight of the reverse arc corresponding to the forward arc containing a positive flow is reduced linearly. This encourages the algorithm to use the reverse arc in its successive flow augmentations. An example of the construction of the residual graph is shown in Figure 5. For an edge e with link delay D_e, if there is an existing flow of $F_e > 0$ units along the

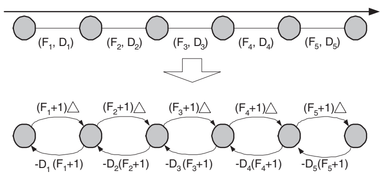
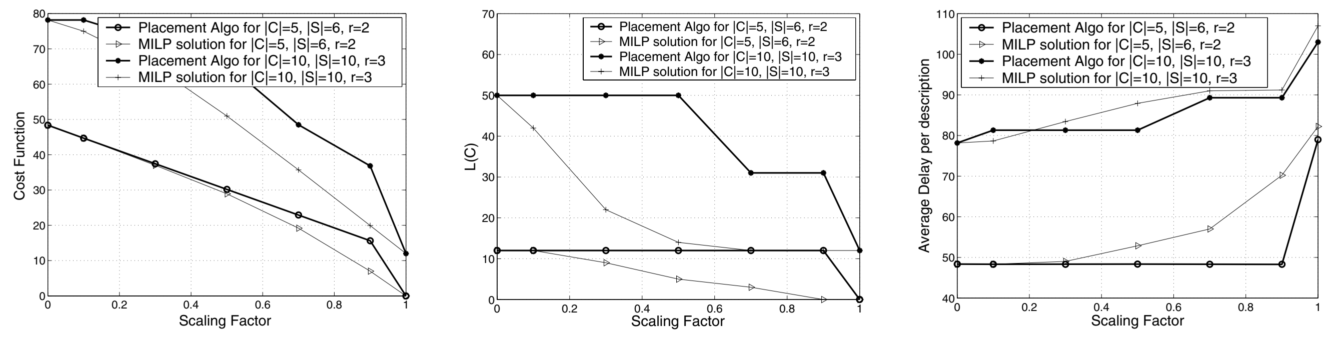


Fig. 5. Residual graph construction.

edge, then the cost of the forward arc is set to $(F_e + 1)\Delta$ and the cost of the reverse arc is set to $-D_e(F_e + 1)$. The algorithm then finds the subsequent $\lfloor \alpha r \rfloor - 1$ server locations by routing one unit of flow from the different potential server locations to each client on their respective residual graphs and then choosing the location that minimizes the average link delay among all possible server locations. After finding the $\lfloor \alpha r \rfloor$ server locations, the remaining $r - \lfloor \alpha r \rfloor$ servers are chosen from the set of remaining potential server locations ($|S| - \lfloor \alpha r \rfloor$). These servers are chosen sequentially, such that total average delay of the shortest path from the client nodes to the server is minimized ($\sum_{c \in C} \delta(s_t, c)$). $\delta(s_t, c)$ is the shortest paths w.r.t. to delay from s_t to c. Notice that these remaining servers do not take path diversity into account. The pseudocode for the algorithm is presented in Figure 6.

IV. Simulation Results

In this section, we describe via simulations the performance of the proposed placement algorithm. Our simulations are based on random topologies that obey the recently observed power laws [4]. These topologies were generated using the BRITE software. The link delay, client set C, and the set of potential server locations S are randomly generated. For a given topology, each link (i, j) is assigned a delay value $w(i, j)$, which is randomly sampled from a uniform distribution in the range $[0, 50]$. We show the effectiveness of placement algorithm by comparing it with optimum performance obtained using the MILP solution.



(a) Cost function vs α (b) $L(C)$ vs α (c) Average delay per description vs α

Fig. 7. Performance of placement algorithm and MILP solution (BRITE network topology with 50 nodes).

```
**Placement_Algorithm**$(G(V,E), S, C, d(.), \alpha)$
1.  $Count = 0$, $\mathbb{L} = \phi$, $\Delta = \max_{(u,v)\in E} d(u,v)$
2.  $D = Floyd\_Warshall(G, d)$
3.  Pick $s \in S$ s.t. $\sum_{j\in C} D[s][j] = \min_{i\in S} \sum_{j\in C} D[i][j]$
4.  $\mathbb{L} = \mathbb{L} \cup s$, $Count = Count + 1$
5.  Route a unit flow from $s$ to each client $c \in C$
6.  Update the residual graph for each client
7.  While $Count < \lfloor \alpha r \rfloor$,
8.      Set $X \leftarrow \infty$
9.      For each $s_t \in S - \mathbb{L}$
10.         Compute $X = \min(X, \sum_{c\in C} \delta(s_t, c))$
11.     $s_{itr}$ is the server corresponding to $X$
12.     Route a unit flow from $s_{itr}$ to each client $c \in C$
13.     Update the residual graph for each client
14.     $\mathbb{L} = \mathbb{L} \cup s_{itr}$
15.     $Count = Count + 1$
16. End While
17. Choose $r - \lfloor \alpha r \rfloor$ servers from the list of
    remaining servers which minimize $\sum_{j\in C} D[i][j]$
18. $X$ = Calculate_Total_Delay($C$)
19. $Y$ = Calculate_Total_Diversity($C$)
20. Return $Z = (1-\alpha) \times \frac{X}{r\times|C|} + \alpha \times Y$
```

Fig. 6. Pseudocode for the placement algorithm.

We vary the scaling factor α for a given number of clients, potential server locations, and a fixed value of r. Figure 7(a) compares the placement algorithm with the optimal solution in terms of cost function. The performance of the placement algorithm is comparable to the MILP solution. It should be noted that the algorithm uses a discrete optimization by fixing the number of servers positions that optimize for diversity and average delay, hence the algorithm performs exceptionally well at the extremes ($\alpha = 0$ and $\alpha = 1$). The difference between the cost function values increases as the value of α is increased but becomes zero at the extreme. This is a direct consequence of the discrete optimization performed by fixing the number of servers that optimize for delay and then for diversity. Figure 7(b) compares the performance of placement algorithm and the MILP solution in terms of $L(C)$ for various values of α. The placement algorithm performs similar to MILP at the extremes. The difference in $L(C)$ is higher for $0 < \alpha < 1$ because the placement algorithm minimizes the cost function by fixing the number of servers that optimize for delay and diversity. Figure 7(c) compares the performance of the placement algorithm with the MILP results in terms of the average delay associated with a description received by the client node. In general, the algorithm performs significantly close to the MILP results.

V. Conclusions

In this paper, we considered the problem of finding optimal locations for the servers that provide MDC coded data to a client community using diverse path routing. We considered the basic server placement problem in which the goal is to minimize a linear combination of average delay associated with the paths over which these descriptions are routed and the number of common links between these paths. We proposed an MILP solution and a highly efficient placement algorithm to solve the problem. Simulation results show the effectiveness of the proposed algorithm.

References

[1] R. Ahuja, T. Magnanti, and J. Orlin. *Network flows: Theory, Algorithm, and Applications.* Prentice Hall Inc., 1993.

[2] J. Apostolopoulos. Reliable video communication over lossy packet networks using multiple state encoding and path diversity. *Proceedings of the International Workshop on Visual Communications and Image Processing*, Jan 2001.

[3] J. Apostolopoulos, T. Wong, W. Tan, and S. Wee. On multiple description streaming media content delivery networks. *Proceedings of the IEEE INFOCOM Conference*, 3:1736–1745, June 2002.

[4] M. Faloutsos, P. Faloutsos, and C. Faloutsos. Power-laws of the Internet topology. *Proceedings of the ACM SIGCOMM Conference*, pages 251–262, Sept 1999.

[5] A. Mahanti, D. Eager, M. Vernon, and D. Sundaram-Stukel. Scalable on-demand media streaming with packet loss recovery. *Proceedings of SIGCOMM '01*, pages 97–108, 2001.

[6] M. Rocha, M. Maia, T. Cunha, J. Almeida, and S. Campos. Scalable media streaming to interactive users. *Proceedings of the ACM Multimedia conference*, pages 966–975, 2005.

[7] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy. An analysis of Internet content delivery systems. *SIGOPS Oper. Syst. Rev.*, 36(SI):315–327, 2002.

[8] A. Vakali and G. Pallis. Content delivery networks: status and trends. *Proceedings of the IEEE INFOCOM Conference*, 7:68–74, Nov-Dec. 2003.