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

Satyajeet Ahuja and Marwan Krunz
Dept. of ECE, The University of Arizona
{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 a media stream into r substreams that are sent independently from a source to a destination. Each substream (or description) can be decoded independent of the other $r-1$ substreams. With every successful reception of a substream, the quality of the decoded signal improves. In this paper, we consider the problem of placing a set of servers in the network such that a desired quality of service can be provided to a community of clients that request MDC-coded traffic. We formulate the server placement (SP) problem, with the goal of identifying the minimum number of server locations that can provide r descriptions to a set of clients such that the delay associated with each path from a chosen server location to a given client is bounded by a given delay constraint and the total "unreliability" associated with the group of paths to a given client is also upper bounded. We show that the SP problem belongs to the class of NP-complete problems. We propose a mixed-integer linear programming (MILP) formulation and an efficient heuristic solution for the SP problem. Simulations are conducted to evaluate the performance of the proposed algorithm and compare it with the optimal solution provided by the MILP solution.**

***Index Terms*—Multiple description coding, path diversity.**

I. Introduction

Content delivery networks (CDNs) have recently been the focus of intensive research (e.g., [18], [13], [19]). The interest in these networks stems from their ability to cope with various transport problems associated with Internet delivery, including congestion and server overload. CDNs significantly improve the performance of a Web session by caching popular content on servers located close to end-users, resulting in relatively shorter network paths. This reduces response time, packet loss probability, and the overall network resource usage. CDNs were originally designed for the delivery of "offline" content, but recent efforts have also considered their application in media streaming [5].

Many approaches for media streaming have been proposed in the literature. One recently popularized approach relies on multiple description coding (MDC) combined with multi-path diversity routing [4], [5], [18]. MDC is essentially a coding technique in which an input signal (video/audio) is encoded into r bitstreams, referred to as descriptions. Each description can be decoded independent of the other descriptions and can alone provide a certain level of video quality. Furthermore, by embedding complimentary information in each description, any subset of the r descriptions can be combined at the receiver, with the quality of the video/audio stream improving as the number of successfully received descriptions increases. The decodability of the individual descriptions and their (approximately) equal importance makes MDC significantly different from the well-known layered approach. In the layered approach, the substreams (or layers) form a hierarchy. Packets are differentiated according to their importance; the most important packets ("base layer") are given the highest transport priority, while less important packets ("enhancement layers") are given lower priority. High-priority packets are critically needed for the reconstruction of the video signal.

Path diversity is a technique used in packet networks to deliver data over multiple paths, which may or may not originate from the same server. The use of multiple paths for streaming media has been shown to reduce packet losses [4]. When applied to MDC streaming, path diversity uses different paths to route different descriptions to a client. Blackouts due to link or node failures along the path of one of the descriptions are avoided. Architecturally, MDC content delivery is supported by a "front-end" server, which receives requests for media and redirects them to several "back-end" servers associated with different descriptions. These back-end servers may or may not be geographically co-located. Each back-end server serves one or more descriptions. Placing the servers of different descriptions at different nodes (e.g., routers) helps in finding diverse paths for multiple clients (see Figure 1).

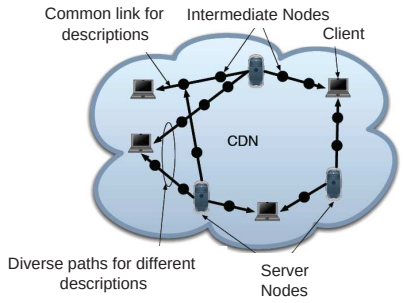


Fig. 1. CDN with MDC traffic and multi-path diversity.

The server placement (SP) problem has be extensively studied in the literature (see [10], [6], [15] and the references therein), mainly in the context of replicated data servers. The key goal in such studies is to minimize the overall cost of assigning servers and transferring data from them to various clients. In [15], a polynomial-time approximation algorithm was presented to solve the SP problem,

This research was supported in part by NSF (under grants CNS-0721935, CNS-0627118, CNS-0325979, and CNS-0313234), Raytheon, and Connection One (an I/UCRC NSF/industry/university consortium). Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the author(s) and do not necessarily reflect the views of NSF. An abridged version of this paper was presented at the IEEE Globecom 2006 Conference.

considering the round-trip delays for data requests. In all previous formulations of the SP problem, content is replicated at all locations, and clients choose the nearest server locations based on delay, hop length, or some other criteria.

In this paper, we study the SP problem for supporting MDC-media streaming over CDNs. Our goal is to find the minimum number of server locations and corresponding client-server paths such that the delays of the paths between a client and its associated servers are individually upper-bounded by a constant (delay bound) and the total "unreliability" (to be defined later) associated with these paths is also upper-bounded by a constant (diversity bound). We first prove that the SP problem is NP-complete. Accordingly, we formulate it as a mixed-integer linear program (MILP) and provide an efficient algorithmic solution to it. We also provide a pseudo-polynomial-time approximation algorithm for a special case of the SP problem (with one client and two descriptions). Simulations are conducted to show the effectiveness of the proposed algorithm and compare it with the MILP solution.

II. Path Diversity

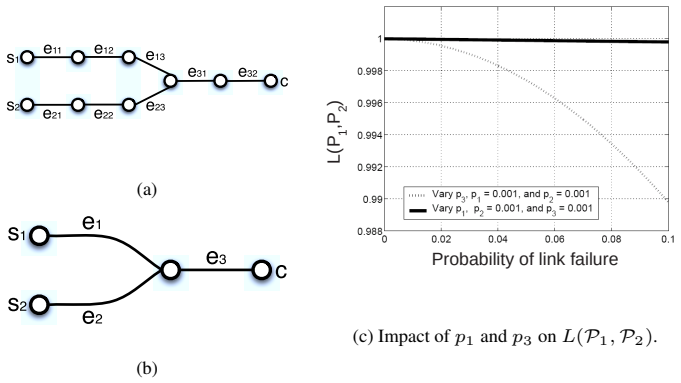
Before formulating the SP problem, we first define a metric for measuring path diversity in the context of MDC-media streaming. A client that receives multiple descriptions from different servers observes a blackout only if all corresponding descriptions of a frame are lost or all of them arrive late. Hence, to maintain continuous video playback, one should minimize the blackout probability. Intuitively, the blackout probability for a client is mostly affected by the packet loss rate over links that transport multiple descriptions. To illustrate, consider two servers, s_1 and s_2, that provide two distinct descriptions to a client c along two arbitrary paths $\mathcal{P}_1$ and $\mathcal{P}_2$ that possibly share some links. Without loss of generality, for the purpose of calculating the packet loss rate, links along the two paths can be rearranged such that all common links are contiguous. For a given path, such rearrangement does not change the overall packet loss rate observed over that path. An example of two such paths is shown in Figure 2(a), with $\mathcal{P}_1 = (e_{11} \rightarrow e_{12} \rightarrow e_{13} \rightarrow e_{31} \rightarrow e_{32})$ and $\mathcal{P}_2 = (e_{21} \rightarrow e_{22} \rightarrow e_{23} \rightarrow e_{31} \rightarrow e_{32})$. We simplify the paths by creating "super-edges" $e_1 = (e_{11} \rightarrow e_{12} \rightarrow e_{13})$, $e_2 = (e_{21} \rightarrow e_{22} \rightarrow e_{23})$, and $e_3 = (e_{31} \rightarrow e_{32})$, as shown in Figure 2(b). Let p_i be the packet loss rate over super-edge e_i, i = 1, 2, 3, i.e., the aggregate packet loss rate observed over the subpath represented by e_i. We are interested in finding the probability that at least one description is successfully received at client c. Let $L(\mathcal{P}_1, \mathcal{P}_2)$ denote such probability. Then,

$$L(\mathcal{P}_1, \mathcal{P}_2) = 1 - [1 - (1 - p_1)(1 - p_3)][1 - (1 - p_2)(1 - p_3)]$$

$$= 1 - [p_1p_2 + p_2p_3 + p_1p_3 + p_3^2 - 2p_1p_2p_3 - (p_1 + p_2)p_3^2 + p_1p_2p_3^2].$$

We assume that all links in the network have reasonably small and comparable packet loss rates. Such an assumption is valid for the current Internet, and has been corroborated by several experimental studies (e.g., [1], [21]). Links with high packet loss rates can be removed (by defining a *policy constraint*). Accordingly, it can easily be shown that $(p_1 + p_2)p_3^2 \ll p_3^2$, $p_1p_2p_3^2 \ll p_1p_2$, and $2p_1p_2p_3 \ll p_1p_2$. The probability that at least one description is received successfully at c can then be approximated by:

$$L(\mathcal{P}_1, \mathcal{P}_2) \simeq 1 - [p_1p_2 + (p_1 + p_2 + p_3)p_3]. \quad (1)$$



(c) Impact of p_1 and p_3 on $L(\mathcal{P}_1, \mathcal{P}_2)$.

Fig. 2. Impact of link loss rate on the blackout probability.

From (1), we make the following observations: (1) $L(\mathcal{P}_1, \mathcal{P}_2)$ is symmetric in p_1 and p_2; (2) $L(\mathcal{P}_1, \mathcal{P}_2)$ is monotonically decreasing in p_1, p_2, and p_3. We now show that $L(\mathcal{P}_1, \mathcal{P}_2)$ is more sensitive to a change in p_3 than to a change in p_1 or p_2 by computing the partial derivative of $L(\mathcal{P}_1, \mathcal{P}_2)$ w.r.t. p_1 and p_3:

$$\frac{\partial L(\mathcal{P}_1, \mathcal{P}_2)}{\partial p_1} = -(p_2 + p_3) \quad (2)$$

$$\frac{\partial L(\mathcal{P}_1, \mathcal{P}_2)}{\partial p_3} = -(p_1 + p_2 + 2p_3) \quad (3)$$

Since p_1, p_2, and p_3 are positive numbers, we have

$$\left|\frac{\partial L(\mathcal{P}_1, \mathcal{P}_2)}{\partial p_3}\right| > \left|\frac{\partial L(\mathcal{P}_1, \mathcal{P}_2)}{\partial p_1}\right|. \quad (4)$$

Hence, the rate of change in $L(\mathcal{P}_1, \mathcal{P}_2)$ w.r.t. p_3 is greater than the rate of change in $L(\mathcal{P}_1, \mathcal{P}_2)$ w.r.t. p_1. Figure 2(c) depicts $L(\mathcal{P}_1, \mathcal{P}_2)$ as a function of p_1 and p_3. From (4) and Figure 2(c), we conclude that the blackout probability is more sensitive to the loss rate over a shared link between the multiple paths than to the loss rate over an exclusive link. Our conclusion can be easily generalized to more than two paths. Hence, paths should be chosen such that the loss rate over the common links is minimized.

The above analysis was conducted assuming that loss rates over different links are i.i.d. A common link carries more MDC traffic than a non-common link, so a change in its loss rate (e.g., an increase in p_3) has a more drastic effect on the blackout probability than a non-common link. For this reason, common links with high loss rates should be avoided as much as possible. The impact of the common link on the blackout probability is even more significant if packet losses are correlated. Based on the argument presented above, we define the following measure of unreliability for a set of paths:

Definition 1: For two paths $\mathcal{P}_i$ and $\mathcal{P}_j$, their unreliability is defined as: $U(\mathcal{P}_i, \mathcal{P}_j) \stackrel{\text{def}}{=} \sum_{\ell \in \mathcal{L}} p_\ell 1_\ell(\mathcal{P}_i, \mathcal{P}_j)$, where $\mathcal{L}$ is the set of links in the network, p_ℓ is the packet loss rate over link ℓ, and $1_\ell(\mathcal{P}_i, \mathcal{P}_j)$ is an indicator function that is equal to 1 if $\mathcal{P}_i$ and $\mathcal{P}_j$ share link ℓ[1]. For a set of paths $\mathcal{R}$, their unreliability measure $U(\mathcal{R})$ is defined as

$$U(\mathcal{R}) \stackrel{\text{def}}{=} \sum_{\mathcal{P}_i \in \mathcal{R}} \sum_{\mathcal{P}_j \in \mathcal{R},\ \mathcal{P}_j \neq \mathcal{P}_i} U(\mathcal{P}_i, \mathcal{P}_j). \quad (5)$$

[1]If the network provider wishes to use a multiplicative metric to represent the unreliability measure, then this metric can be logarithmically transformed into an additive metric.

The rationale behind using the metric $U(\mathcal{R})$ as an indication of the unreliability of the set $\mathcal{R}$ is to give equal importance to various descriptions. If different MDC descriptions have different levels of importance, then an appropriately weighted sum can be used in defining the unreliability measure.

Other definitions of unreliability can also be used, for example to handle pathological cases (e.g., high loss rates over a few links). For instance, for a set of paths $\mathcal{R}$, their unreliability can be defined as a linear combination of $U(\mathcal{R})$ and the sum of loss rates of all links in $\mathcal{R}$. Such a definition discourages the use of links with high loss rates.

III. Server Placement Problem

In this section, we study the problem of placing a set of multiple description (MD) servers in a CDN that supports a set of clients $\mathcal{C}$. Each server is assumed to have all the descriptions. However, we assume for now that a client can get at most 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 consist of a set of nodes $\mathcal{N}$ and a set of links $\mathcal{L}$. We assume that the MD servers are to be chosen from a subset $\mathcal{S} \subseteq \mathcal{N}$. We refer to the nodes in $\mathcal{S}$ as *potential server locations*. Servers cannot be placed in other locations because of geographical and/or security reasons. Let r be the number of distinct descriptions and let $D(\mathcal{P})$ be the delay associated with a path $\mathcal{P}$. We formally define the SP problem as follows:

Problem 1: [Server Placement (SP)] Let $\mathcal{G}(\mathcal{N}, \mathcal{L})$ be a network graph. Suppose that each link $(u, v) \in \mathcal{L}$, where u and v are two nodes in $\mathcal{N}$, is associated with a delay value $d(u, v)$ and a packet loss rate $p(u, v)$. The goal is to find the minimum set of server locations $\mathcal{S}_C \subseteq \mathcal{S}$ and an associated sets of paths $\{\mathcal{R}_c : c \in \mathcal{C}\}$, where $\mathcal{R}_c$ is the set of paths from r servers in $\mathcal{S}_C$ to a client $c \in \mathcal{C}$, such that:

$$D(\mathcal{P}) \stackrel{\text{def}}{=} \sum_{(u,v)\in\mathcal{P}} d(u, v) \leq B_{req},\ \forall \mathcal{P} \in \mathcal{R}_c \text{ and } \forall c \in \mathcal{C} \quad (6)$$

$$U(\mathcal{R}_c) \leq D_{req},\ \forall c \in \mathcal{C} \quad (7)$$

where B_{req} and D_{req} are positive constants.
The above formulation assumes that packet loss rates are known. This can be done using various estimation techniques (e.g., [1], [21]). Estimated loss rates can be advertised via link-state dissemination approach (e.g., [9]).

The formulation in Problem 1 does not include a capacity metric. This is because we assumed that the load incurred due to MDC traffic is not significant relative to the overall traffic over a link. This assumption is made to simplify the formulation, but is also in line with realistic traffic profiles in the Internet. For cases where this assumption is not valid, capacity constraints $C_{req}(u, v)$ can be used for each link $(u, v) \in \mathcal{L}$ to limit the total traffic over the link.

When $B_{req} = \infty$ and $D_{req} = 0$, the SP problem reduces to finding the minimum set of servers that can provide r disjoint paths to each client $c \in \mathcal{C}$. In this case, some descriptions may be excessively delayed, increasing the starvation rate at the client's buffer. The other extreme is when $D_{req} = \infty$ and B_{req} is set to an arbitrarily small value. In this case, the optimal solution to the SP problem will place the server locations such that the total delay associated with the multiple paths to a client is minimized. Such a solution, however, ignores path diversity, and the traffic from different servers to a given client may be routed over many common links. A failure of any of these links will result in missing several descriptions, significantly degrading the playout performance.

While it is easy to obtain maximally disjoint paths by using a variant of the maximum-flow algorithm [2], the inclusion of delay constraint in (6) makes the problem significantly harder.

Theorem 1: The SP problem is NP-complete.

Proof: Consider the corresponding decision problem, where the goal is to determine if there exists a set of k servers such that the delay and diversity constraints are met for a given client set $\mathcal{C}$. First, we show that the SP problem belongs to the class of NP problems. The certificate for the verification algorithm is chosen as the set of servers in $\mathcal{S}_C$, the associated sets of paths $\{\mathcal{R}_c : c \in \mathcal{C}\}$, and a set of clients $\mathcal{C}$. The verification algorithm affirms that $D(\mathcal{P}) \leq B_{req},\ \forall \mathcal{P} \in \mathcal{R}_c$ and $\forall c \in \mathcal{C}$. It also verifies the diversity constraint for each client. This verification can be easily performed in polynomial time because $|\mathcal{C}|$ is $\mathcal{O}(|\mathcal{N}|)$.

Next, we prove that the SP problem is NP-hard by showing that the NP-complete Min-Max Multicenter problem (also known as the p-center problem) [8] can be reduced in polynomial time to the SP problem. The decision problem for a variant of the p-center problem is given as follows: Is there a set $\mathcal{S}_c$ of x (a known positive integer) nodes in $\mathcal{N}$ such that the maximum distance of the shortest path from any node in $\mathcal{N}$ to the closest node in $\mathcal{S}_c$ is less than B_{req}?

The reduction approach takes as input an instance of the p-center problem $\{\mathcal{G}(\mathcal{N}, \mathcal{L}), B_{req}, x\}$. For the graph $\mathcal{G}(\mathcal{N}, \mathcal{L})$, set $D_{req} = 0$ (the diversity bound), $r = 1$ (the number of MDs), and $k = x$ (a total of x server locations are required). The output of the reduction algorithm is an instance of the SP problem. We now show that this output is yes (positive) if and only if the output of the p-center problem on $\{\mathcal{G}(\mathcal{N}, \mathcal{L}), B_{req}, x\}$ is also yes (positive). The output of the SP problem in this case is a set of servers $\mathcal{S}_C$ and a path from each client to one element in $\mathcal{S}_C$ (because $r = 1$). For each client, the delay associated with the path from this client to the server is less than B_{req}. If this is not the shortest path from the client to its closest server in $\mathcal{S}_C$, then Dijkstra's algorithm can be used to find the shortest path (the delay associated with the shortest path will also be less than B_{req}). Hence, a solution to the SP problem is also a solution to the p-center problem. The solution to the p-center problem will satisfy the delay and diversity bounds with $r = 1$. The reduction only requires fixing the values of B_{req}, D_{req}, and r, as well as the calculation of the shortest path from each client to the elements in $\mathcal{S}_C$, which can be done in polynomial time. ■

A. MILP Formulation

Figure 3 depicts an MILP formulation of the SP problem. The formulation assumes that each server provides at most one description to one or more clients. As we show later in this section, a slight modification can be made to allow for streaming multiple descriptions from one server location. The objective function of the MILP is to minimize the total number of selected servers. Let $s_i \in \mathcal{S}$ be one of the selected server locations and let $\mathcal{P}(s_i, c_j)$ be the path chosen to deliver the description from server s_i to client $c_j \in \mathcal{C}$. The binary variable $x(s_i, c_j, u, v)$ is set to one if $\mathcal{P}(s_i, c_j)$ contains the edge (u, v); otherwise, it is set to 0. For each node $s_i \in \mathcal{S}$, we set $z(s_i) = 1$ if an MD server is placed at s_i, and we 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

Objective function minimize $\sum_{s_i \in \mathcal{S}} z(s_i)$

Subject to the following constraints:

$$\mathrm{C}_1: \quad \sum_{v:(u,v)\in\mathcal{L}} x(s_i,c_j,u,v) - \sum_{v:(v,u)\in\mathcal{L}} x(s_i,c_j,v,u) \begin{cases} \leq 1, & u = s_i \\ \geq -1, & u = c_j \\ = 0, & \text{otherwise} \end{cases} \quad (8)$$
$$\forall s_i \in \mathcal{S}, \quad u \in \mathcal{N}, \text{ and } c_j \in \mathcal{C}.$$

$$\mathrm{C}_2: \quad \sum_{s_i\in\mathcal{S}} \sum_{u:(u,c_j)\in\mathcal{L}} x(s_i,c_j,u,c_j) = r, \ \forall c_j \in \mathcal{C}. \quad (9)$$

$$\mathrm{C}_3: \quad \sum_{(s_i,u)\in\mathcal{L}} x(s_i,c_j,s_i,u) \leq 1, \ \forall c_j \in \mathcal{C}, \ \forall s_i \in \mathcal{S}. \quad (10)$$

$$\mathrm{C}_4: \quad \sum_{(u,v)\in\mathcal{L}} \{x(s_i,c_j,u,v)d(u,v)\} \leq B_{req}, \ \forall c_j \in \mathcal{C}, \forall s_i \in \mathcal{S}. \quad (11)$$

$$\mathrm{C}_5: \quad 2y(s_i,s_j,c_k,u,v) - x(s_i,c_k,u,v) - x(s_j,c_k,u,v) \leq 0 \quad (12)$$
$$\forall (u,v) \in \mathcal{L}, \forall c_k \in \mathcal{C}, \text{ and } \forall (s_i,s_j) \in \mathcal{S}\times\mathcal{S} \text{ with } s_i \neq s_j.$$

$$\mathrm{C}_6: \quad y(s_i,s_j,c_k,u,v) - x(s_i,c_k,u,v) - x(s_j,c_k,u,v) + 1 \geq 0 \quad (13)$$
$$\forall (u,v) \in \mathcal{L}, \forall c_k \in \mathcal{C}, \text{ and } \forall (s_i,s_j) \in \mathcal{S}\times\mathcal{S} \text{ with } s_i \neq s_j.$$

$$\mathrm{C}_7: \quad \sum_{(s_i,s_j)\in\mathcal{S}\times\mathcal{S}} \sum_{(u,v)\in\mathcal{L}} \{y(s_i,s_j,c_k,u,v)p(u,v)\} \leq D_{req}, \ \forall c_k \in \mathcal{C}. \quad (14)$$

$$\mathrm{C}_8: \quad z(s_k) \geq x(s_k,c_j,s_k,u), \ \forall s_k \in \mathcal{S}, \quad \forall (s_k,u) \in \mathcal{L}, \ c_j \in \mathcal{C}. \quad (15)$$

Fig. 3. MILP formulation for the SP problem.

servers s_i and s_j to a client c_k have a common link $(u,v) \in \mathcal{L}$. Essentially,

$$y(s_i,s_j,c_k,u,v) = x(s_i,c_k,u,v)x(s_j,c_k,u,v). \quad (16)$$

Constraint C_1 in Figure 3 is the flow conservation constraint for client c_j and server s_i. It limits the number of descriptions per server-client pair to one. As shown in constraints C_2 and C_3, every client needs to get r descriptions, one from each chosen server. C_4 ensures that the delay associated with each description is less than B_{req}. C_5 and C_6 transform (16) into an MILP. This transformation is needed because for an MILP formulation the constraints should be linear in the variables. C_7 ensures that the set of paths associated with a client satisfies the diversity constraint D_{req}. C_8 ensures that $z(s_i) = 1$ if s_i provides at least one description to at least one client.

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 4). These auxiliary nodes are then treated as potential server locations, with each location providing at most a single description. The delay and packet loss rate associated with the edge between the potential server location and each auxiliary node are set to zero.

Incorporating capacity constraints: A capacity constraint on a link (u,v) can be added by bounding the total MDC traffic passing over (u,v). Let $\mathcal{J}$ be the bandwidth consumed by a description. Then, a capacity constraint on a link is given by: $\sum_{s_i\in\mathcal{S},\, c_j\in\mathcal{C}} x(s_i,c_j,u,v)\mathcal{J} \leq C_{req}(u,v), \ \forall (u,v) \in \mathcal{L}$.

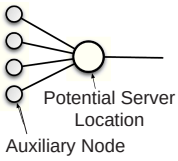


Fig. 4. Supporting multiple descriptions at a single server location.

It should be noted that the MILP approach is essentially a brute-force method. Its complexity grows *exponentially* with the number of variables in the problem. Despite this prohibitive complexity, the MILP solution provides a reference point for assessing the goodness of heuristic/approximation algorithms.

B. SP Algorithm

The exponential complexity of the MILP approach makes it impractical for large networks. Moreover, the set of clients who require content delivery may change frequently, necessitating frequent recomputation of server locations and paths. Accordingly, to ensure manageable computational complexity, we develop a heuristic algorithmic solution to the SP problem which we simply refer to as the SP algorithm. In Appendix I, we also provide a pseudo-polynomial-time approximation algorithm for a special case of SP problem (with two descriptions and one client).

In the SP algorithm, we start with an initial set of server locations and sequentially add more server locations to satisfy the delay and diversity constraints for each client. The input to the algorithm consists of $\mathcal{G}(\mathcal{N},\mathcal{L})$, $d(u,v)$ $\forall (u,v) \in \mathcal{L}$, $p(u,v)$ $\forall (u,v) \in \mathcal{L}$, $\mathcal{S}$, $\mathcal{C}$, r, B_{req}, and D_{req}. Let $\delta_d(s,c)$ be the delay of the shortest path between a server s and a client c, and let W be a $|\mathcal{N}| \times |\mathcal{N}|$ matrix with elements $W(i,j) = \delta_d(i,j)$. For a given client $c \in \mathcal{C}$, let $\triangle(c,\mathcal{S},B_{req})$ be the number of nodes s in $\mathcal{S}$ for which $\delta_d(s,c) \leq B_{req}$. A *delay cover* D_{cov} is a set of servers in $\mathcal{S}$ such that *for each client* $c \in \mathcal{C}$, $\triangle(c,D_{cov},B_{req}) \geq r$, i.e., for every client $c \in \mathcal{C}$, there is at least r servers in D_{cov} whose shortest path delays to c is less than or equal to B_{req}. The procedure for finding a delay cover is presented in Section III-D. Note that for the set $\mathcal{S}$, the delay cover is not unique. The placement algorithm starts by computing D_{cov}. At any stage a client $c \in \mathcal{C}$ maintains the following information:

1. f_c: Number of descriptions destined to client c for which server locations and feasible paths have already been found. Initially f_c is set to 0 for all $c \in \mathcal{C}$.
2. The residual graph $\mathcal{G}_c$ for client c (described later), which determines the aggregate flow along various links in the network. Initially $\mathcal{G}_c$ is set to $\mathcal{G}$.
3. The set of paths $\mathcal{R}_c$ from the chosen server locations to client c. Initially $\mathcal{R}_c$ is empty.

For a given D_{cov}, the algorithm randomly picks a client that still requires at least one more description, i.e., $f_c < r$. For a client, the algorithm picks server locations from D_{cov} in a random order such that for a selected server s, $\delta_d(s,c) < B_{req}$. The algorithm checks if server s can provide a description without violating the delay and diversity constraints by using the procedure `CheckFlow`, described in Section III-E. If there exists a feasible flow, i.e., if the output of `CheckFlow` is 0, the algorithm uses the procedure `RouteFlow`, described in Section III-E, to update the residual graph $\mathcal{G}_c$. Then, f_c and $\mathcal{R}_c$ are updated accordingly.

The algorithm recomputes the delay cover if at least one of the clients still requires at least one description, i.e., if $f_c < r$. When the delay cover is recomputed, additional servers are added to the existing delay cover such that for each client c there are $r - f_c$ additional servers with $\delta_d(s,c) \leq B_{req}$. The algorithm terminates if $\forall c \in \mathcal{C}$, $f_c \geq r$ or if $D_{cov} = \mathcal{S}$. If the algorithm terminates with $D_{cov} = \mathcal{S}$ and there is still a client c with $f_c < r$, then the algorithm cannot find a set of servers and associated paths that can simultaneously satisfy the delay and diversity constraints for all clients. The algorithm removes any server s from D_{cov} if this server does not provide a description to any client. A pseudocode for the SP algorithm is presented in Figure 5.

Procedure: SP
Input: $\mathcal{G}(\mathcal{N},\mathcal{L}), d(.,.), p(.,.), \mathcal{S}, \mathcal{C}, r, B_{req}, D_{req}$
Output: $\mathcal{S}_C,\ \mathcal{R}_c,\ c \in \mathcal{C}$

1) Initialize:
   a) $\mathcal{S}_C = \phi$, $D_{cov} = \phi$, $\mathcal{G}_c = \mathcal{G}, \forall c \in \mathcal{C}$.
   b) $\forall s \in \mathcal{S},\ W_s = Dijkstra(\mathcal{G}(\mathcal{N},\mathcal{L}), d(.,.), s)$.
   c) $Itr = 0,\ f_c = 0$, and $\mathcal{R}_c = \phi,\ \forall c \in \mathcal{C}$.
2) While $Itr == 0$,
   a) D_{cov} = `DelayCover`$(\mathcal{G}, W(.), \mathcal{S}, \mathcal{C}, f_c, D_{cov})$
   b) For each $c \in \mathcal{C}$ s.t. $f_c < r$,
      For each $s \in D_{cov}$,
      If `CheckFlow`$(s, c, \mathcal{G}_c) == 0$,
      `RouteFlow`$(s, c, \mathcal{G}_c)$
      $f_c = f_c + 1$
   c) If $\forall c \in \mathcal{C}, f_c \geq r$
      $Itr = 1$.
   d) If $D_{cov} == \mathcal{S}$ and $\exists\ c : f_c < r$
      Return: Infeasible solution
3) $\mathcal{S}_C = D_{cov}$.

Fig. 5. Pseudocode for the SP algorithm.

Note that the SP algorithm is a heuristic. It is not guaranteed to find a solution even if one such solution exists. If the SP algorithm fails to return a feasible solution, the network operator can re-execute it with slightly modified B_{req} and D_{req} values, and with a modified set of potential server locations.

C. Residual Graph of a Client

The SP algorithm requires determining the residual graph $\mathcal{G}_c$ for every client $c \in \mathcal{C}$. In this section, we define $\mathcal{G}_c$ and explain how it is determined. For a network $\mathcal{G}(\mathcal{N},\mathcal{L})$, the residual graph $\mathcal{G}_c(\mathcal{N},\mathcal{L})$ for a client c is a graph of $\mathcal{N}$ nodes and $\mathcal{L}$ links. Each link (u,v) in $\mathcal{G}_c$ is associated with two weights: $f(u,v)$, which represents the total number of descriptions (or total flow) passing through link (u,v), and $l(u,v)$, which represents a link metric used for choosing the paths over the residual network. Initially, $f(u,v)$ is set to 0 and $l(u,v)$ is set to $d(u,v)\ \forall (u,v) \in \mathcal{L}$. At any stage in the SP algorithm, if a path $\mathcal{P}$ is chosen from a server s to a client c, then $f(u,v)$ and $l(u,v)$ are updated as follows:

- For every link $(u,v) \in \mathcal{P}$, $f(u,v)$ is set to $f(u,v)+1$ and $f(v,u)$ is set to $f(v,u)-1$.
  - If $f(u,v) = 0$, then $l(u,v) = d(u,v)$.
  - If $f(u,v) > 0$, then $l(u,v) = f(u,v)(\Omega + p(u,v))$, where Ω is some large number.
  - If $f(u,v) < 0$, then $l(u,v) = -f(u,v)d(u,v)$.

A flow is then routed along the shortest path w.r.t. $l(.,.)$ between a server s and a client c on the residual graph $\mathcal{G}_c$ of c. The above setting of $l(u,v)$ ensures the following:

• If the shortest path w.r.t. $l(.,.)$ from s to c has a link (u,v) with $f(u,v) > 0$, then there can be no path $\mathcal{P}$ from s to c for which $f(u,v) = 0\ \forall (u,v) \in \mathcal{P}$.

• Among all possible paths between any server $s \in D_{cov}$ and client c, path $\mathcal{P}$ has the fewest number of links with positive flow, i.e., $f(u,v) > 0$.

• If the chosen path $\mathcal{P}$ passes through links with positive $f(.,.)$ values, then such links have the maximum total reliability, i.e., they should have the minimum $\sum_{(u,v):f(u,v)>0,(u,v)\in\mathcal{P}} f(u,v)p(u,v)$ value among all paths between any server $s \in D_{cov}$ and client c.

An example of the residual graph along path $\mathcal{P}$ is given in Figure 6. In this example, $f(u,v)$ is initially set to 0.

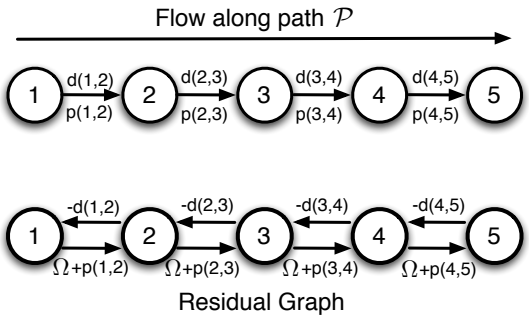


Fig. 6. Construction of the residual graph.

D. Computation of the Delay Cover

We use the following greedy approach to determine an initial value for D_{cov}. For each potential server locations, we determine N_s, which denotes the number of clients $c \in \mathcal{C}$ with $r - f_c > 0$ and with $\delta(s,c) \leq B_{req}$. We sort the list of potential server locations based on their N_s values. We pick the server s_x from the top of the sorted list. Then, we increment f_c for all clients c with $\delta(s_x,c) \leq B_{req}$ and re-sort the list based on the updated N_s values. This process continues until each client $c \in \mathcal{C}$ satisfies $f_c \geq r$ or until there are no more server locations to consider. A pseudocode for the `DelayCover` procedure is presented in Figure 7. In the pseudocode, $1_{[.]}$ is the indicator function.

Notice that a delay cover is based purely on the delay of the shortest path. It does not take into account path diversity. At any point during the execution of the SP algorithm, if the current delay cover cannot satisfy the delay and diversity constraints for all clients, then the SP algorithm has to update D_{cov} for clients $c \in \mathcal{C}$ for which $f_c < r$ and whose delay and diversity constraints cannot be satisfied under the existing delay cover.

E. Check Flow and Route Flow Procedures

Check Flow: We now describe the `CheckFlow` procedure used by the SP algorithm. The procedure is presented in Figure 7. It takes as input a client c, a server location s, and a residual graph $\mathcal{G}_c$. It then finds the shortest path $\mathcal{P}$ on $\mathcal{G}_c$ from s to c w.r.t. $l(.,.)$.

Procedure: DelayCover ($\mathcal{G}(\mathcal{N},\mathcal{L}), W(.), \mathcal{S}, \mathcal{C}, f_c, D_{cov}$)	**Procedure** CheckFlow (s, c, $\mathcal{G}_c$)
1. $T_{f_c} = f_c,\ \forall c \in \mathcal{C}$	Finds an augmenting path from s to c in the residual graph $\mathcal{G}_c$ and checks if each resultant path satisfies the delay constraint and the reliability constraint associated with paths.
2. $Itr = 0$	
3. While ($Itr == 0$)	
3a. Obtain $\mathcal{C}_r \subset \mathcal{C}$ s.t. $T_{f_c} < r\ \forall\ c \in \mathcal{C}_r$.	
3b. $\forall s \in \mathcal{S} - D_{cov}$, $R_s = \sum_{c \in \mathcal{C}_r} 1_{[\delta(s,c) < B_{req}]}$	**Procedure** RouteFlow
3c. Select $s_x : R_{s_x} = \max_{s \in \mathcal{S} - D_{cov}} R_s$	**Input:** s, c, $\mathcal{G}_c$
3d. $D_{cov} = D_{cov} \cup s_x$	**Output:** Returns 1 if $s \rightarrow c$ is a feasible flow, and 0 otherwise. Routes unit flow from s to c on residual graph $\mathcal{G}_c$ and updates link weights in $\mathcal{G}_c$.
3e. $\forall\ c \in \mathcal{C}_r$ s.t. $\delta(s_x, c) < B_{req}$, $T_{f_c} = T_{f_c} + 1$	
3f. If $\forall c \in \mathcal{C}, T_{f_c} \geq r$, $Itr = 1$.	
4. Return D_{cov}.	

Fig. 7. Pseudocodes for DelayCover, CheckFlow, and RouteFlow.

The procedure then routes a unit flow along $\mathcal{P}$ and updates the link weights on $\mathcal{G}_c$, as discussed in Section III-C. Finally, CheckFlow checks if all resultant paths satisfy the delay and diversity bounds. Note that even if all previously routed flows satisfy the delay and diversity bounds, a new flow may alter these paths and the procedure needs to check if the delay and diversity constraints are still satisfied. If one path does not satisfy the delay bound or if the diversity bound is violated, the procedure returns a failure, indicating the infeasibility of this flow. Finally, CheckFlow removes the unit flow that is routed along path $\mathcal{P}$. Note that CheckFlow requires executing Dijkstra's shortest path algorithm. Hence its worst-case complexity is $\mathcal{O}(|\mathcal{N}|^2)$.

If a capacity constraint $C_{req}(u,v)$ for a link $(u,v) \in \mathcal{L}$ is considered, then CheckFlow should verify that addition of a description (along any arbitrary path $\mathcal{P}$) for any client does not violate $C_{req}(u,v)$ for any link $(u,v) \in \mathcal{P}$. This can be done by maintaining a count on the number of descriptions already routed along each link $(u,v) \in \mathcal{L}$.

Route Flow: The RouteFlow procedure (see Figure 7) is used to update the residual graph of a client c by routing a unit flow from s to c along the shortest path w.r.t. $l(.,.)$ on $\mathcal{G}_c$. The complexity associated with RouteFlow is $\mathcal{O}(|\mathcal{N}|^2)$.

F. Complexity

Step 1-b in Figure 5 requires $|\mathcal{S}|$ executions of Dijkstra's algorithm, which can be performed in $\mathcal{O}(|\mathcal{S}||\mathcal{N}|^2)$ time. The DelayCover procedure used in Step 2-a adds at least one server to D_{cov}. Hence, the while loop in Step 2 is executed for a maximum of $|\mathcal{S}|$ times. Note that DelayCover requires $\mathcal{O}(|\mathcal{C}|)$ computations to add a server to D_{cov}. For each iteration inside the while loop, Step 2-b calculates the feasible flow for each client. The CheckFlow procedure is called $|\mathcal{S}|$ times for each client $c \in \mathcal{C}$. Step 2-b inside the while loop can be performed in $\mathcal{O}(|\mathcal{S}|^2|\mathcal{C}||\mathcal{N}|^2)$ time. Hence, Step 2 can be performed in $\mathcal{O}(|\mathcal{S}|^2|\mathcal{C}||\mathcal{N}|^2)$ time. Altogether, the overall worst-case complexity of the SP algorithm is $\mathcal{O}(|\mathcal{N}|^2|\mathcal{S}| + \mathcal{O}(|\mathcal{S}|^2|\mathcal{C}||\mathcal{N}|^2) + |\mathcal{S}||\mathcal{C}|) = \mathcal{O}(|\mathcal{S}|^2|\mathcal{C}||\mathcal{N}|^2)$.

IV. Performance Evaluation

We conduct extensive simulations to evaluate the performance of our algorithmic solutions presented in Section III. Our interest here is not only to assess the goodness of these solutions, but to also demonstrate the effectiveness of the MDC-based media streaming approach in general.

A. Assessing the Goodness of the SP Algorithm

To evaluate the performance of the SP algorithm, we run simulations on random topologies generated using the Waxman's model [20]. For each topology, the client set $\mathcal{C}$ and the set of potential server locations $\mathcal{S}$ are randomly selected. In addition to Waxman's graphs, we also use the 21-node star topology shown in Figure 8, which allows us to consider extreme scenarios (e.g., when $\mathcal{S}$ is relatively large) without running into computational problems with the MILP solution. For a given topology, each link (u,v) is assigned a

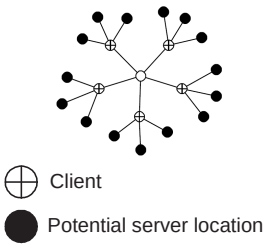


Fig. 8. Star topology with 21 nodes and 20 links.

delay value $d(u,v)$ and a packet loss rate $p(u,v)$ that are randomly sampled from the interval $[0, 50$ msec$]$ and $[0, 0.05]$, respectively. The media stream is split into 3 descriptions. For different values of D_{req} and B_{req}, we obtain the number of server locations that satisfy the delay and diversity constraints for a set of client nodes. Recall that both the MILP and SP algorithms aim at minimizing the number of server locations while meeting the given delay and diversity bounds. For each client $c \in \mathcal{C}$, we also determine the minimum and maximum delays among the r generated paths. The difference between the two reflects the amount of video buffer that is required at the client. We report the averages of both quantities over all clients. Figure 9 shows the performance for Waxman graphs of 50 and 25 nodes, and for the star topology. From this table, we observe that the number of required server locations under the SP algorithm is very close to that of the optimal (MILP) solution for all the cases considered. The average delays of the shortest and longest paths to a client are also comparable with those of the MILP solution.

B. MDC with Multiple Server Locations

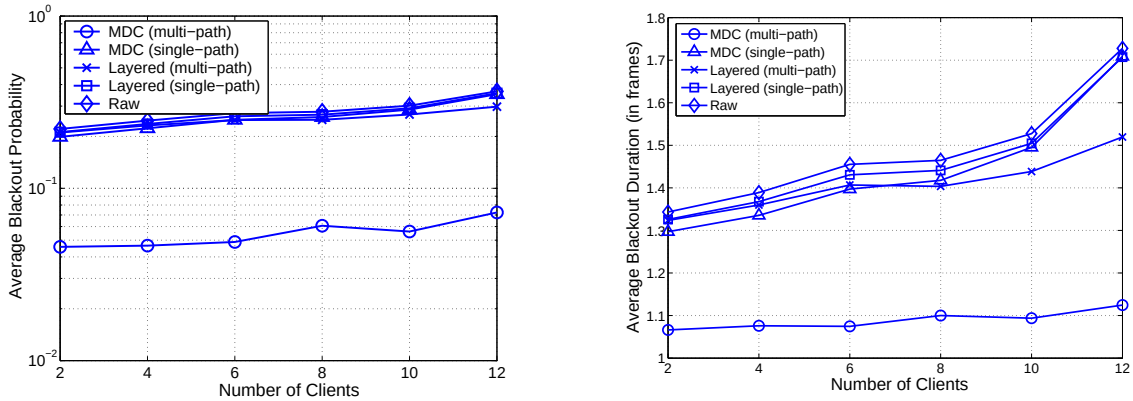
Several previous studies (e.g., [16], [11]) compared MDC and layered media streaming but without intelligent placement of content. We now use a packet-level simulator to evaluate the performance of MDC streaming with multiple, intelligently placed servers. For this set of simulations, we use a randomly generated network of 50 nodes. The sets $\mathcal{C}$ and $\mathcal{S}$ are randomly generated. The delay $d(i,j)$ of a link (i,j) is randomly sampled from a uniform distribution in the range [10, 100]msec. The transmission capacity of each link is set to 10 Mbps. Each link is associated with a two-state (good-bad) Markov model. The sojourn time of the Markov chain in the good (bad) state is exponentially distributed with mean $\mathcal{T}_g$ ($\mathcal{T}_b$). A packet is successfully transmitted over the link when the state is good, and is dropped otherwise. Each node is modeled as an M/M/1 queue with a finite buffer of size $\mathcal{B}$. A node is associated with cross-traffic, which enters and exits at the same node. Cross-traffic is used to control the average load ρ at a node. Incoming traffic at a node (which includes cross-traffic and traffic from adjacent links) enters the queuing system of a node if

Waxman's graph of 50 nodes and 65 links, $\|\mathcal{C}\|$= 10, $\|\mathcal{S}\|$= 10							
B_{req} (msec)	D_{req}	No. of required server locations		Avg path delay – short (msec)		Avg path delay – long (msec)	
		MILP	SP	MILP	SP	MILP	SP
200	1	3	3	53.39	53.98	147.9	126.74
140	1	4	4	53.83	52.04	124.45	116.1
130	1	Infeasible					
150	0.75	4	4	53.39	56.14	126.15	117.92
150	0.5	4	4	53.39	56.14	126.15	117.92
150	0.1	4	7	53.84	53.30	123.23	116.07
150	0.05	Infeasible					

Waxman's graph of 25 nodes and 38 links, $\|\mathcal{C}\|$= 10, $\|\mathcal{S}\|$= 10							
B_{req} (msec)	D_{req}	No. of required server locations		Avg path delay – short (msec)		Avg path delay – long (msec)	
		MILP	SP	MILP	SP	MILP	SP
200	1	3	4	36.09	48.32	103.27	118.33
120	1	4	6	36.09	38.99	92.82	93.14
110	1	5	Inf.	32.78	Inf.	77.42	Inf.
150	0.75	3	4	36.09	43.18	103.27	117.78
150	0.25	3	4	36.09	43.18	103.27	117.78
150	0.15	3	Inf.	36.09	Inf.	103.27	Inf.

Star topology of 21 nodes and 20 links, $\|\mathcal{C}\|$ = 5, $\|\mathcal{S}\|$ = 15							
B_{req} (msec)	D_{req}	No. of required server locations		Avg path delay – short (msec)		Avg path delay – long (msec)	
		MILP	SP	MILP	SP	MILP	SP
100	1	3	3	25.32	34.52	68.05	62.58
50	1	3	4	21.95	16.21	36.03	35.32
35	1	7	7	8.38	8.38	29.97	31.66
100	0.1	3	7	19.65	31.10	51.94	59.61
100	0.01	9	10	8.38	13.35	53.59	55.84
100	0	10	11	13.01	13.70	44.06	59.63

Fig. 9. Performance comparison of the SP algorithm and the MILP solution



(a) Average blackout rate vs. $|\mathcal{C}|$. (b) Average blackout duration vs. $|\mathcal{C}|$.

Fig. 10. Blackout performance vs. $|\mathcal{C}|$ for a Waxman's network of 50 nodes ($r = 3$, $|\mathcal{S}| = 10$, $\mathcal{B} = 20000$ bytes, $\mathcal{T}_g = 100$ msec, $\mathcal{T}_b = 10$ msec, $B_{req} = 2$ sec, $D_{req} = 1.0$, and $\rho = 0.9$.)

enough buffer is available, and is discarded otherwise. In other words a packet loss over a link occurs either when the Markov chain associated with that link is in a "bad" state or if buffer overflow occurs at the node. Each media stream consists of a constant bit rate (CBR) stream of 1000-byte packets streamed at the rate of 25 packets per second. We consider five different cases for media distribution:

• *MDC-coded streaming over multiple paths*: Multiple descriptions of MDC-coded traffic are routed from $|\mathcal{S}_C|$ chosen server locations over multiple paths as determined by the SP algorithm.

• *MDC-coded streaming over a single path*: Similar to the first case, but with all MDs that are destined to a client being routed over the shortest path w.r.t. delay from the closest server in $\mathcal{S}_C$.

• *Layered video streaming over multiple paths*: Instead of MDC, we use layered video with r layers. Multiple paths (determined using the SP algorithm) are used to route the different layers, with the shortest path used for the base layer and the remaining paths for the enhancement layers based on their delay, i.e., shorter paths are used for the more important enhancement layers.

• *Layered video streaming over a single path*: Same as case 3, but with one path used for routing all the r layers to a given client c. This path is selected as in case 2.

• *Raw media streaming*: A set of $\mathcal{S}_C$ server locations are chosen from $\mathcal{S}$ so as to minimize the delay of the shortest path for all the clients. For a given client c, the closest server location in terms of delay is chosen from $\mathcal{S}_C$. This location is used to route CBR traffic to the client.

In generating MDC and layered video, appropriate relative overheads are added as indicated in [7] and [17]. Specifically, for layered video we add 5% overhead and for MDC with $r = 3$ we add 30% overhead. For MDC video, we define a "blackout" as a playback instance at which no portion of the required frame is available for playout at the receiving node. For layered video, "blackout" refers to the unavailability of the base layer of a frame at its playout instance.

Figure 10 depicts the performance of the five considered approaches as a function of the number of clients. Part(a) of the figure shows the average blackout probability, defined as the fraction of the playback instances at which blackout was encountered (averaged over all clients). Part(b) depicts the average blackout duration. Based on the two figures, we conclude the following:

• The average blackout probability increases with the number of clients. This is due to the increase in the average network load.

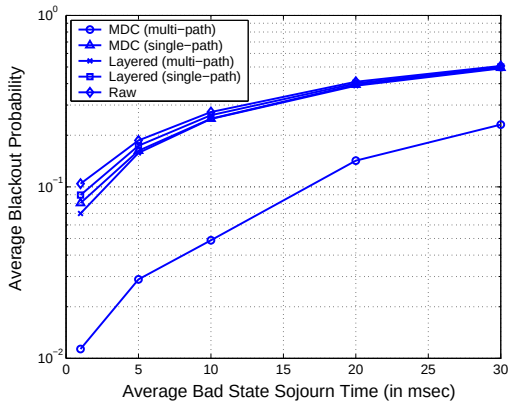
• A significant reduction in the average blackout probability is observed when MDC-coded streaming with multiple paths is used.

• The improvement observed in the average blackout duration under MDC-coded streaming with multiple paths is due to the independence of the packet loss rates over diverse paths.

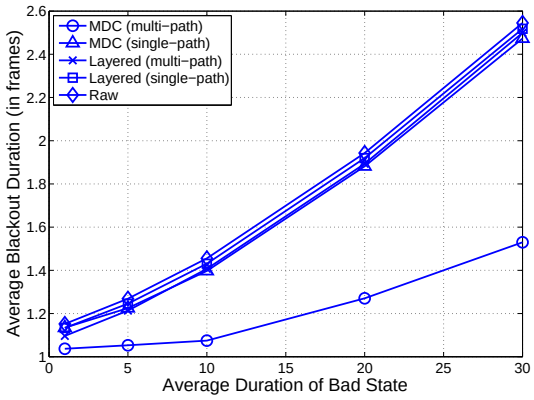
• The performance of the layered scheme with multiple paths depends on the loss rate of the path over which the base layer is transmitted. The advantage of using diverse paths for different layers is lost, because the enhancement layers alone cannot reconstruct a frame.

Figure 11 depicts the performance of the five approaches as a function of $\mathcal{T}_b$. As expected, the blackout probability and the average blackout duration increase monotonically with $\mathcal{T}_b$ for all the considered schemes. When MDC streaming with multiple paths is employed, a significant gain can be achieved in the blackout performance.

We now investigate the performance of the various schemes in



(a) Average blackout probability vs. $\mathcal{T}_b$.



(b) Average blackout duration vs. $\mathcal{T}_b$.

Fig. 11. Blackout performance vs. $|\mathcal{T}_b|$ for a Waxman's network of 50 nodes ($r = 3$, $|\mathcal{S}| = 10$, $|\mathcal{C}| = 6$, $\mathcal{B} = 20000$ bytes, $\mathcal{T}_g = 100$ msec, $B_{req} = 2$ sec, $D_{req} = 1.0$, and $\rho = 0.9$).

terms of the received PSNR values by using actual compressed media (motion JPEG2000). We consider 640×480 size frames from the Batman movie, generated at 30 frames per second. MDC descriptions are obtained by splitting the traffic such that if the jth frame is part of the first description, then the $(j+1)$th frame is part of the second description, the $(j+2)$th frame is part of the third description, and so on. Lost descriptions are compensated for using linear interpolation of the two most adjacent and successfully received frames within the playout deadline of the missing packet. Each MDC frame is compressed such that the average bits/pixel is 0.026, which results in 1000 bytes per frame. For a meaningful comparison with MDC video, the splitting of the layered video is done such that the total PSNR values (averaged over the whole Batman sequence) for MDC video with i descriptions (where $i = 1, 2, 3$) and layered video with i layers are approximately the same[2]. Such a splitting results in 181-byte packets (on average) for the base layer, 299 (552)-byte packets (on average) for the first (second) enhancement layer.

Figure 12(a) shows the PSNR performance of the various streaming approaches as a function of $\mathcal{T}_b$ with $|\mathcal{C}| = 6$. When MDC streaming with multiple paths is employed, a significant improvement in the PSNR values is achieved at high values of $\mathcal{T}_b$. Figure 12(b) shows the performance of different streaming approaches as a function of $|\mathcal{C}|$ with $\mathcal{T}_b = 10$ msec. A significant gain in the PSNR is observed when MDC streaming using multiple paths is employed. This is a direct consequence of using independently decodeable descriptions.

Figure 13 depicts the percentage of descriptions that are received before their playback deadlines. Although schemes involving single-path routing provide a higher percentage of frames when all the descriptions/layers are received, they suffer from excessively high percentage of frames when all the descriptions/layers are lost. In the single-path case, all descriptions/layers are either received or lost because of the correlated losses. MDC with multiple paths provide a significantly less blackout probability; with high probability, at least two out of three descriptions are available for reconstructing the video.

In Figure 14, we compare the SP algorithm with a random placement scheme in which a specific number of server nodes are randomly chosen without knowledge of clients' locations. For fair comparison, the number of servers in the random placement scheme is equal to the number of servers choosen by the SP algorithm. For the random placement scheme, we use the shortest path from the r closest servers to stream the MDC traffic. It is noted that the SP algorithm performs significantly better than the random placement scheme.

[2]For layered video, the JPEG2000 encoder allows the creation of user-defined base and enhancement layers by specifying the average bits per pixel per layer [17].

V. Conclusions

In this paper, we considered the problem of finding optimal locations for MDC servers that deliver to the client community using diverse path routing. We showed that for MDC-coded media streams, the probability of blackout is significantly dependent on the reliability of the common link. We considered the server placement problem for a given client community, with the goal of to minimizing the number of server locations such that MDC-coded streams are routed along diverse paths with given delay and diversity constraints. We proposed an MILP and a highly efficient placement algorithm to solve the server placement problem. Simulation results were used to compare the performance of the placement algorithm with the optimal results obtained using MILP. The effectiveness of the MDC-coded media streaming with path diversity was demonstrated by using packet-level simulations.

Acknowledgement: The authors would like to thank Dr. Ali Bilgin for his help in simulations.

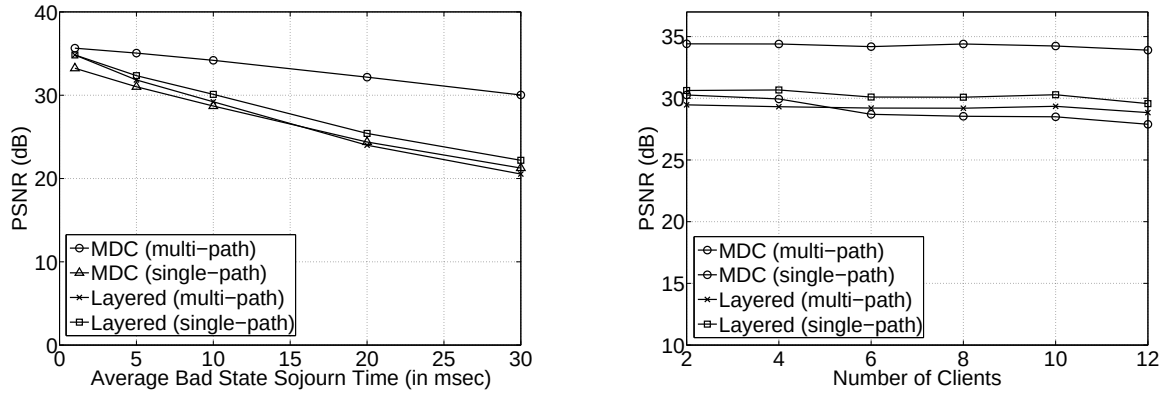
Appendix: Approximation Algorithm

We now present a pseudo-polynomial time approximation algorithm for SP problem for a specific case of single client. First, we define the α-approximate solution for $r = 2$.

Definition 2: Given an instance of the SP problem with a client c and $r = 2$, an α-approximate solution to the SP problem is a solution with two paths $\mathcal{P}_1' \equiv s_1 \rightarrow c$ and $\mathcal{P}_2' \equiv s_2 \rightarrow c$, where $s_1,\ s_2 \in \mathcal{S}$, s.t. $D(\mathcal{P}_1') + D(\mathcal{P}_2') \leq 2\alpha B_{req}$ and $U(\mathcal{P}_1', \mathcal{P}_2') \leq D_{req}$.

Before proceeding further, we state the well known restricted shortest path problem [14]:

Problem 2: Restricted Shortest Path (RSP): Given a source node s, a destination node t, and a delay constraint B_{req}, find an



(a) Average PSNR vs. $\mathcal{T}_b$

(b) Average PSNR vs. number of clients

Fig. 12. Decoded PSNR for different streaming approaches ($r = 3$, $\mathcal{T}_g = 100$ msec, $B_{req} = 2$ sec, $D_{req} = 1.0$, $\rho = 0.9$, $|\mathcal{S}| = 10$, $\mathcal{B} = 20000$ bytes).

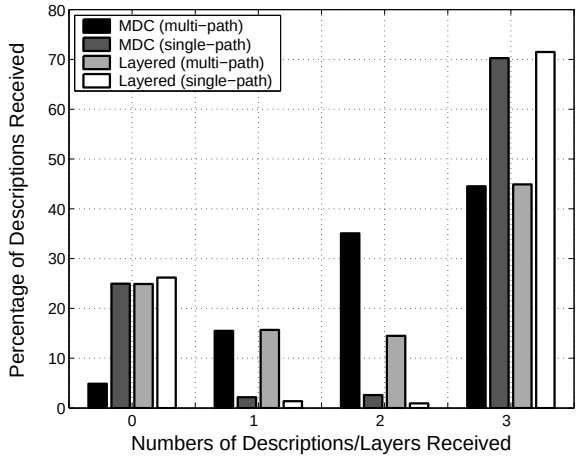


Fig. 13. Histogram of the average number of descriptions /layers received on time for a Waxman's network of 50 nodes ($r = 3$, $|\mathcal{S}| = 10$, $|\mathcal{C}| = 6$, $\mathcal{B} = 20000$ bytes, $\mathcal{T}_g = 100$ msec, $\mathcal{T}_b = 20$ msec, $B_{req} = 2$ sec, $D_{req} = 1.0$, and $\rho = 0.9$).

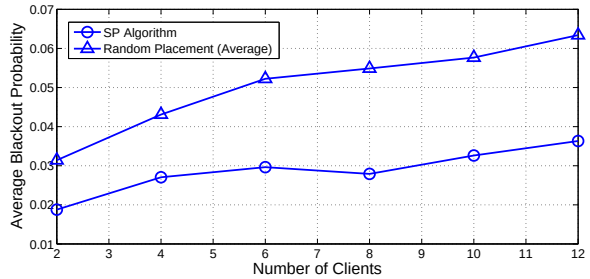


Fig. 14. Blackout probability vs. $|\mathcal{C}|$ for a Waxman's network of 50 nodes ($r = 3$, $\mathcal{B} = 20000$ bytes, $\mathcal{T}_g = 100$ msec, $\mathcal{T}_b = 10$ msec, $B_{req} = 2$ sec, $D_{req} = 1.0$, and $\rho = 0.9$).

(s, t) path $\mathcal{P}$ such that $D(\mathcal{P}) \leq B_{req}$ and $C(\mathcal{P}) \leq C(\mathcal{P}')$ for any other (s, t) path $\mathcal{P}'$ that satisfies $D(\mathcal{P}) \leq B_{req}$.

RSP is known to be NP-hard [8]. Several approximation algorithms have been proposed for it [12], [14]. An efficient scheme presented in [12] has a computational complexity of $\mathcal{O}(|\mathcal{L}||\mathcal{N}|(\frac{1}{\epsilon} + \log\log|\mathcal{N}|))$. It computes a path with delay of at most B_{req} and cost of at most $(1 + \epsilon)$ times the optimum. We will refer to this scheme as RSP algorithm. In [14], an approximation algorithm was presented to solve the disjoint path problem using the RSP algorithm. We employ a similar approach. We assume $r = 2$ and equally reliable links ($p(i, j) = p$, $\forall(i, j) \in \mathcal{L}$). In this case, the unreliability constraint can be translated into a limit on the number of common links H permitted between two paths, $H = D_{req}/p(i, j)$. We use the link loss rate as the cost metric.

The basic idea of the algorithm is to identify a suitable flow f between server locations and a client c such that $|f| = 2$ and then decompose f into two paths $\mathcal{P}_1$ and $\mathcal{P}_2$. The algorithm employs the path augmentation approach [2], which is a standard approach for network flow and disjoint path problems. We assume that the capacity of each link is two units.

The first step of the algorithm is to connect all server locations $s \in \mathcal{S}$ to the super server S (a virtual node) using links $(\mathtt{S}, s)$. Delay and packet loss rate for link $(\mathtt{S}, s)$ are set to zero ($d(\mathtt{S}, s) = 0$, $p(\mathtt{S}, s) = 0$). The algorithm then computes a path $\mathcal{P}_1$ between S and c that satisfies B_{req} and minimizes the cost (link unreliability). $\mathcal{P}_1$ is constructed by applying RSP algorithm on $(\mathcal{G}, \mathtt{S}, t, B_{req}, p(.,.), \epsilon)$. It defines a unit flow f. Let $h(\mathcal{P}_1)$ be the hop count of $\mathcal{P}_1$. In the next step, the algorithm augments this flow in order to increase its value to 2 by adding another path. It first constructs the residual network $\mathcal{G}(f)$ imposed by the flow f. Intuitively, the residual network consists of links that can admit more flow. To address the diversity constraint, it manipulates the link cost based on path $\mathcal{P}_1$.

Residual Network: Given a network $\mathcal{G}$ with each link associated with two units of capacity and a flow $f = \{\mathcal{P}_1\}$, the residual network $\mathcal{G}(f)$ is constructed as follows:

For each link $(u, v) \in \mathcal{G}$ for which $f(u, v) = 0$, we add a link (u, v) to $\mathcal{G}(f)$ of zero cost and the same delay as in $\mathcal{G}$. For each link $(u, v) \in \mathcal{G}$ for which $f(u, v) = 1$, we add to $\mathcal{G}(f)$:

1) A link (u, v) of the same delay and cost as in $\mathcal{G}$.
2) A reverse link (v, u) of zero cost and delay.

The augmenting flow f' in the residual network $\mathcal{G}(f)$ is a path from S to c. f' can now be augmented on $\mathcal{G}(f)$ as follows:

• From f, remove each link (v, u) whose reverse link (u, v) appears in f'. This will cancel the flow along the reverse direction.

• Add to f each link $(v, u) \in f'$ whose reverse link (u, v) does not appear in f.

Note that each link in $\mathcal{G}$ has two units of capacity. Hence, the final flow f can have $f(u, v) = 2$ for some links after augmenting two units of flow. With the augmenting path approach, a unit capacity flow f' is added to flow f. In particular, our al-

gorithm identifies an augmenting path $\mathcal{P}_2$ in $\mathcal{G}(f)$ that satisfies a delay constraint $3B_{req}$. To identify path $\mathcal{P}_2$, we apply the RSP algorithm on $(\mathcal{G}(f), \mathtt{S}, c, 3B_{req}, \epsilon)$, for $C(h(\mathcal{P}_1), (h(\mathcal{P}_1) - H)^+)$ (where $C(n,r) = \frac{n!}{(n-r)!r!}$) different graphs. In each graph, we remove a combination of $h(\mathcal{P}_1) - H$ links from path $\mathcal{P}_1$ and then search for path from S to c. We show in Theorem 2 that among all possible $C(h(\mathcal{P}_1), (h(\mathcal{P}_1) - H)^+)$ combinations, there exists a combination for which a path between S to c exists.

After identifying the path, we augment the flow f along path $\mathcal{P}_2$. For each link (u, v) that belongs to $\mathcal{P}_2$, we increment the flow $f(u, v)$ along the link. The value of the resulting flow f is 2.

Finally, the flow is decomposed into two paths $\mathcal{P}'_1$ and $\mathcal{P}'_2$ such that $D(\mathcal{P}'_1) \leq D(\mathcal{P}'_2)$. For this purpose, we apply the flow decomposition algorithm. We start at the source node S and select a link $(\mathtt{S}, u)$ such that $f(\mathtt{S}, u) > 0$. If u is the client node, we stop; otherwise, there must be a link (u, v) for which $f(u, v) > 0$. This process is repeated until the destination node is encountered. For all traversed links (u, v), the flow $f(u, v)$ is decremented. The second path $\mathcal{P}'_2$ is also identified using the same approach. The pseudocode of the approximation algorithm is presented in Figure 15. We prove the correctness of the algorithm in the following

Procedure SP Approximate
Input: $\mathcal{G}(\mathcal{N}, \mathcal{L}), p(.,.), d(.,.), \mathcal{S}, c, D_{req}, B_{req}$
Output : $\mathcal{P}'_1$, $\mathcal{P}'_2$

1) Add super server S and connect it using links $(\mathtt{S}, s)$, $\forall s \in \mathcal{S}$
2) Set $d(\mathtt{S}, c) = 0$ and $p(\mathtt{S}, c) = 0$
3) $H = B_{req}/p(u, v)$
4) Find path $\mathcal{P}_1$ in $\mathcal{G}$ such that $D(\mathcal{P}_1) \leq D_{req}$ by using RSP algorithm
5) $f \leftarrow \{\mathcal{P}_1\}$
6) Construct the residual network $\mathcal{G}(f)$ of $\mathcal{G}$ imposed by f:
   a) Add to $\mathcal{G}(f)$ each link in $\mathcal{G}$ that does not belong to path $\mathcal{P}_1$
   b) For each link $(u, v) \in \mathcal{P}_1$
      i) Add (v, u) to $\mathcal{G}(f)$ with delay $d(v, u) = 0$ and reliability $p(v, u) = 0$
      ii) Add (u, v) to $\mathcal{G}(f)$ with delay $d(u, v)$ and reliability $p(u, v)$ as in $\mathcal{G}$
7) For all possible link combinations $C(h(\mathcal{P}_1), (h(\mathcal{P}_1) - H)^+)$ from path $\mathcal{P}_1$
   a) Remove the link combination from $\mathcal{G}(f)$
   b) Identify path $\mathcal{P}_2$ from S to c in the residual graph
   c) Break, if $\mathcal{P}_2$ is obtained
8) Add $\mathcal{P}_2$ to flow f
9) Obtain path $\mathcal{P}'_1$ and $\mathcal{P}'_2$ by flow decomposition

Fig. 15. Pseudocode for the SP approximate algorithm.

theorem:

Theorem 2: For residual graph $\mathcal{G}(f)$ of $\mathcal{G}$ imposed by $f = \{\mathcal{P}_1\}$, $\exists$ a path $\mathcal{P}_2 \in \mathcal{G}(f)$ such that $D(\mathcal{P}_2) \leq 3B_{req}$ and which satisfies the diversity bound D_{req}, i.e., $U(\mathcal{P}_1, \mathcal{P}_2) \leq D_{req}$.
Proof of Theorem 2 is presented in [3].

Complexity: A path on the residual graph $\mathcal{G}(f)$ is computed using Dijkstra's algorithm and can be done in $\mathcal{O}(|\mathcal{N}|^2)$. The approximation algorithm in the worst case executes $C(|\mathcal{L}|, H) + 1$ instances of Dijkstra's algorithm. Hence, the overall complexity of the approximation algorithm is $\mathcal{O}(C(|\mathcal{L}|, H)|\mathcal{N}|^2)$.

REFERENCES

[1] Felix: Independent monitoring for network survivability. ftp://ftp.bellcore.com/pub/mwg/felix/index.html.

[2] R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall Inc., 1993.

[3] S. S. Ahuja and M. Krunz. Technical report: Server placement in multiple-description-based media streaming. *University of Arizona, Department of ECE, TR-UA-ECE-2008-1*, 2008.

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

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

[6] Y. Chen, R. Katz, and J. Kubiatowicz. Dynamic replica placement for scalable content delivery. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems*, pages 306–318, Cambridge, Mar. 2002.

[7] F. H. Fitzek, B. Can, R. Prasad, and M. Katz. Overhead and quality measurements for multiple description coding for video services. *Wireless Personal Multimedia Communications (WPMC)*, 2:524–528, Sept. 2004.

[8] M. Garey and D. Johnson. *Computers and Intractability: Theory of NP-Completeness*. W. H. Freeman, 2000.

[9] T. Korkmaz and M. Krunz. Bandwidth-delay constrained path selection under inaccurate state information. *IEEE/ACM Transactions on Networking*, 11(3):384–398, June 2003.

[10] R. Krishnan, D. Raz, and Y. Shavitt. The cache location problem. *IEEE/ACM Transactions on Networking*, 8(5):568–582, Oct. 2000.

[11] Y. C. Lee, J. Kim, Y. Altunbasak, and R. M. Mersereau. Layered coded vs. multiple description coded video over error-prone networks. *Signal Processing: Image Communication*, 18(5):337–356, May 2003.

[12] D. Lorenz and D. Raz. A simple efficient approximation scheme for the restricted shortest path problem. *Operations Research Letters*, 28(5):213–219, June 2001.

[13] J. Ni, D. Tsang, I. Yeung, and X. Hei. Hierarchical content routing in large-scale multimedia content delivery network. In *Proceedings of the IEEE ICC Conference*, volume 2, pages 854–859, May 2003.

[14] A. Orda and A. Sprintson. Efficient algorithms for computing disjoint QoS paths. In *Proceedings of the IEEE INFOCOM Conference*, Hong Kong, Mar. 2004.

[15] G. Rodolakis, S. Siachalou, and L. Georgiadis. Replicated server placement with QoS constraints. In *Proceedings of the International Workshop on QoS in Multiservice IP Networks*, Catania, Italy, Feb. 2005.

[16] R. Singh, A. Ortega, L. Perret, and W. Jiang. Comparison of multiple description coding and layered coding based on the network simulations. In *SPIE Image and Video Communications and Processing*, San Jose, Jan. 2000.

[17] D. Taubman and M. Marcellin. *JPEG2000: Image Compression Fundamentals, Standards and Practice*. Springer, 2001.

[18] A. Vakali and G. Pallis. Content delivery networks: Status and trends. In *Proceedings of the IEEE INFOCOM Conference*, volume 7, pages 68–74, San Francisco, Apr. 2003.

[19] L. Wang, V. Pai, and L. Peterson. The effectiveness of request redirection on CDN robustness. *SIGOPS Oper. Syst. Rev.*, 36(SI):345–360, 2002.

[20] B. M. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 69:1617–1622, Dec. 1988.

[21] W. Zhu. A distributed approach to estimate link-level loss rates. In *Proceedings of the International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP), LNCS, Distributed and Parallel Computing*, volume 3719, pages 386–395, Oct. 2005.

Satyajeet S. Ahuja received the B.E. (Hons.) degree in Electronics and Communication Engineering from Maulana Azad College of Technology (MACT), Bhopal, India, in 1999. He received his M.E. (Hons.) degree in Electrical and Communication Engineering (Telecommunications) from Indian Institute of Science (IISc), Bangalore, India, in 2002. He also held research positions at Google Inc. Kirkland WA; Tejas Networks, Bangalore, India; and VSNL, Bangalore, India. He is currently working towards his Ph.D. degree at the University of Arizona, Tucson, USA. His research interests include QoS routing, design and analysis of algorithms, path selection, and network monitoring problems for next generation optical networks.

Marwan M. Krunz is a professor of electrical and computer engineering at the University of Arizona. He received the Ph.D. degree in electrical engineering from Michigan State University in 1995. From 1995 to 1997, he was a postdoctoral research associate with the department of computer science, University of Maryland, College Park. His recent research interests include medium access and routing protocols for mobile ad hoc networks, quality of service provisioning over wireless links, constraint-based routing, traffic modeling, and media streaming. He has published more than 120 journal articles and refereed conference papers in these areas. He received the National Science Foundation CAREER Award (1998-2002). He currently serves on the editorial board for the IEEE/ACM Transactions on Networking and the IEEE transactions on Mobile Computing, and the Computer Communications Journal. He served as the technical program co-chair for the IEEE INFOCOM 2004 Conference.