Server Placement in Multiple-Description-Based Media Streaming

Satyajeet Ahuja and Marwan Krunz
Dept. of ECE, University of Arizona
{ahuja,krunz}@ece.arizona.edu

Abstract

Multiple description coding (MDC) is a powerful source coding technique that involves encoding a media stream into r independently decodeable substreams. With every successful reception of a substream, decoded signal quality improves. 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. We formulate the server placement (SP) problem, whose goal is to identify 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 is NP-complete. We propose a mixed-integer linear programming (MILP) formulation and heuristic solution for the SP problem. Simulations are conducted to evaluate the performance of the proposed algorithm and to compare it with the MILP solution.

I. Introduction

Content delivery networks (CDNs) have recently been the focus of intensive research (e.g., [9], [7]) because of their ability to cope with various transport problems associated with 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. Because the content is delivered from edge servers and not from the original (remote) servers, relatively short network paths are used. This reduces response time, packet loss rate, and 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 [3]. In this paper, we focus on the streaming of encoded media (e.g., video clips) using a CDN.

Many approaches for media streaming have been proposed in the literature (see [6] and the references therein). One recently popularized approach relies on multiple description coding (MDC) combined with multi-path diversity routing [2], [3]. MDC is essentially a coding technique in which a signal is encoded into r bitstreams, referred to as descriptions. Each description can be decoded independently and can alone provide a certain level of video quality. The quality of the video/audio stream improving as number of successfully received descriptions increases. The decodability of individual descriptions and their equal importance makes the MDC technique significantly different from the well-known layered approach. In layered approach, data packets are differentiated in terms of their importance; the most important packets form the base layer, and are given the highest transport priority, while the less important packets form "enhancement layers," and are given lower priority. In case of network congestion, a minimum quality video is maintained by discarding the low-priority packets. 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 originating from possibly different locations. Media streaming over multiple paths reduces packet loss rate [2]. MDC media streaming with path diversity employs diverse paths for different descriptions resulting in reduction of blackouts due to link/node failures. An intelligent placement of the content servers will enable a client to efficiently choose a subset of servers with maximally disjoint paths between these servers and the given client (see Figure 1(a)).

In this paper, we study the server placement (SP) problem for supporting MDC-media streaming over CDNs. Our goal is to find the minimum number of server locations such that

1068-0314/08 $25.00 © 2008 IEEE
DOI 10.1109/DCC.2008.63

the delays of the paths between a pool of clients and there 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.

II. Path Diversity

Before formulating the SP problem, we start by defining 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 if all descriptions of a frame are lost or delayed. 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. For the purpose of calculating the packet loss rate, links along the two paths can be rearranged such that all common links are connected. 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 1(b), with $\mathcal{P}_1 = (e_{11}\text{-}e_{12}\text{-}e_{13}\text{-}e_{31}\text{-}e_{32})$ and $\mathcal{P}_2 = (e_{21}\text{-}e_{22}\text{-}e_{23}\text{-}e_{31}\text{-}e_{32})$. We simplify the paths by creating "super-edges" $e_1 = (e_{11}\text{-}e_{12}\text{-}e_{13})$, $e_2 = (e_{21}\text{-}e_{22}\text{-}e_{23})$, and $e_3 = (e_{31}\text{-}e_{32})$, as shown in Figure 1(c). Let p_i be the packet loss rate over super-edge e_i, i = 1, 2, 3. The packet loss rate over a super-edge is the aggregate packet loss rate observed over the sub-path represented by that super-edge. We are interested in finding the probability that at least one description is received successfully at client c. Let $L(\mathcal{P}_1, \mathcal{P}_2)$ denote such probability. Then,

$$L(\mathcal{P}_1,\mathcal{P}_2) \overset{\text{def}}{=} 1 - \Pr[\text{none of the two descriptions from } s_1 \text{ and } s_2 \text{ is received at } c]. \quad (1)$$

$$= 1 - [1-(1-p_1)(1-p_3)] \times [1-(1-p_2)(1-p_3)] \quad (2)$$

$$= 1 - [p_1p_2 + p_2p_3 + p_1p_3 + p_3^2 - 2p_1p_2p_3 - (p_1+p_2)p_3^2 + p_1p_2p_3^2]. \quad (3)$$

We assume that all the links in the network are reasonably reliable, i.e., packet loss rates are relatively small. 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 $2p_1p_2p_3 \ll p_1p_2$. Hence, $L(\mathcal{P}_1, \mathcal{P}_2)$ can then be approximated by:

$$L(\mathcal{P}_1,\mathcal{P}_2) \simeq 1 - [p_1p_2 + p_2p_3 + p_1p_3 + p_3^2] \; = \; 1 - [p_1p_2 + (p_1+p_2+p_3)p_3]. \quad (4)$$

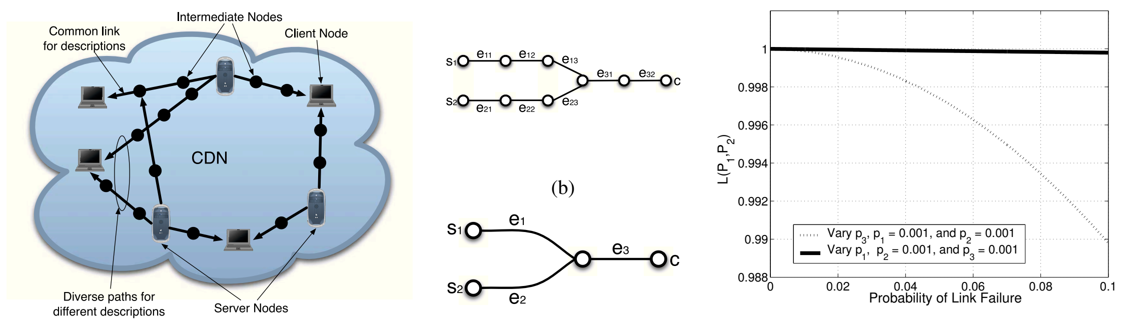
From (4), we observe that $L(\mathcal{P}_1, \mathcal{P}_2)$ is symmetric in p_1 and p_2 and is monotonically decreasing in p_1, p_2, and p_3. We now show that $L(\mathcal{P}_1, \mathcal{P}_2)$ is more sensitive to a change in p_3 than to p_1 or p_2. We first compute the partial derivative of $L(\mathcal{P}_1, \mathcal{P}_2)$ w.r.t. p_1 and p_3:

$$\frac{\partial L(\mathcal{P}_1,\mathcal{P}_2)}{\partial p_1} = -(p_2+p_3) \Rightarrow \left|\frac{\partial L(\mathcal{P}_1,\mathcal{P}_2)}{\partial p_1}\right| = p_2+p_3, \text{ and } \frac{\partial L(\mathcal{P}_1,\mathcal{P}_2)}{\partial p_3} = -(p_1+p_2+2p_3) \Rightarrow \left|\frac{\partial L(\mathcal{P}_1,\mathcal{P}_2)}{\partial p_3}\right| = p_1+p_2+2p_3. \quad (5)$$

$$\left|\frac{\partial L(\mathcal{P}_1,\mathcal{P}_2)}{\partial p_3}\right| > \left|\frac{\partial L(\mathcal{P}_1,\mathcal{P}_2)}{\partial p_1}\right|, \text{because } p_1,\ p_2, \text{ and } p_3 \text{ are positive numbers} \quad (6)$$

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 1(d) depicts $L(\mathcal{P}_1, \mathcal{P}_2)$ as a function of p_1 and p_3. From (6) and Figure 1(d), 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, multiple paths should be chosen such that the loss rate over the common links is minimized. Based on the above argument, we define the measure of unreliability for a set of paths as follows:

Definition 1: For two paths $\mathcal{P}_i$ and $\mathcal{P}_j$, their unreliability $U(\mathcal{P}_i, \mathcal{P}_j)$ is defined as the total unreliability of the number of common links between $\mathcal{P}_i$ and $\mathcal{P}_j$. Specifically, $U(\mathcal{P}_i, \mathcal{P}_j) \overset{\text{def}}{=}$


Intermediate Nodes
Common link for descriptions
Client Node
CDN
Diverse paths for different descriptions
Server Nodes
s1 e11 e12 e13 e31 e32 c
s2 e21 e22 e23
s1 e1 e3 c
s2 e2
L(P1,P2)
Probability of Link Failure
Vary p3, p1 = 0.001, and p2 = 0.001
Vary p1, p2 = 0.001, and p3 = 0.001

(a) Content delivery networks with MDC traffic and multi-path diversity. (b) (c) (d) Impact of p_1 and p_3 on $L(\mathcal{P}_1, \mathcal{P}_2)$.

Fig. 1. Example and figure to illustrates the impact of path diversity on the blackout probability.

$\sum_{\ell \in \mathcal{L}} \{p_\ell 1_\ell(\mathcal{P}_i, \mathcal{P}_j)\}$ 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 when both paths $\mathcal{P}_i$ and $\mathcal{P}_j$ pass through link ℓ. Finally, for a set of paths $\mathcal{R}$, their unreliability measure $U(\mathcal{R})$ is defined as

$$U(\mathcal{R}) \overset{\text{def}}{=} \sum_{\mathcal{P}_i \in \mathcal{R}} \sum_{\mathcal{P}_j \in \mathcal{R},\, \mathcal{P}_j \neq \mathcal{P}_i} U(\mathcal{P}_i, \mathcal{P}_j). \tag{7}$$

III. Server Placement Problem

In this section, we study the problem of placing a set of multiple description (MD) servers for CDN that support $\mathcal{C}$ clients. 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. For a network of $\mathcal{N}$ nodes and $\mathcal{L}$ links, we assume that MD servers can be chosen only 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 $\mathcal{P}$. Our objective is to find the smallest possible set of server locations $\mathcal{S}_C \subseteq \mathcal{S}$ and the 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 the delay incurred in delivering each description to each client is less than an upper bound B_{req} and the unreliability measure associated with delivering the r descriptions to each client is upper-bounded by D_{req}.

Problem 1— Server Placement (SP): Let $\mathcal{G}(\mathcal{N}, \mathcal{L})$ be a network graph. Suppose that each link $(u, v) \in \mathcal{L}$ 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 $\{\mathcal{R}_c : c \in \mathcal{C}\}$ from r servers in $\mathcal{S}_C$ to each client $c \in \mathcal{C}$ such that:

$$D(\mathcal{P}) \overset{\text{def}}{=} \sum_{(u,v) \in \mathcal{P}} d(u, v) \leq B_{req},\ \forall \mathcal{P} \in \mathcal{R}_c \text{ and } \forall c \in \mathcal{C} \tag{8}$$

$$U(\mathcal{R}_c) \leq D_{req},\ \forall c \in \mathcal{C} \tag{9}$$

where B_{req} and D_{req} are positive constants.

For the case when $B_{req} = \infty$ and $D_{req} = 0$, the problem reduces to finding the minimum set of servers that can provide r disjoint paths to each client $c \in \mathcal{C}$. Due to the path disjointness requirement ($D_{req} = 0$), different descriptions will tend to take longer paths than when path diversity is not accounted for, resulting in larger average path delays. Although the failure of any link will not hamper the download rate, each description will traverse more hops and will consume excessive network resources. In this case, some of the descriptions for a client may be excessively delayed, hence increasing the starvation rate of the receiver's playback buffer. The other extreme is when $D_{req} = \infty$ and B_{req} is set to an arbitrarily small value that ensures the existence of a feasible solution. 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 common links will result in missing several descriptions, significantly degrading the performance of the media playout.

While it is easy to obtain maximally disjoint paths by using a variant of the maximum-flow algorithm presented in [1], the inclusion of the delay constraint in (8) makes the problem significantly harder. In fact, we now show that SP problem is NP-complete.

Theorem 1: The SP problem is NP-complete.

Proof: Consider the corresponding decision problem for the SP 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 $|\mathcal{C}| = \mathcal{O}(|\mathcal{N}|)$.

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) [5] 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 P of x (a known positive integer) nodes in V such that the maximum distance of the shortest path from any node in V to the closest node in P is less than T?

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

A. MILP Formulation of the SP Problem

Figure 2 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 at server s_i to a 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 servers s_i and s_j to a client c_k have a common link $(u, v) \in \mathcal{L}$. Essentially,

$$y(s_i, s_j, c_k, u, v) = x(s_i, c_k, u, v)x(s_j, c_k, u, v). \tag{10}$$

Objective function minimize $\sum_{s_i \in \mathcal{S}} z(s_i)$
Subject to the following constraints:

$$\begin{array}{lrll}
\mathrm{C}_1: & \sum_{v:(u,v)\in\mathcal{L}} x(s_i, c_j, u, v) - \sum_{v:(v,u)\in\mathcal{L}} x(s_i, c_j, v, u) & \le 1, & u = s_i \\
 & & \ge -1, & u = c_j \\
 & & = 0, & \text{otherwise} \quad (11) \\
 & \forall s_i \in \mathcal{S}, & u \in \mathcal{N}, & \text{and } c_j \in \mathcal{C}. \\
\mathrm{C}_2: & \sum_{s_i\in\mathcal{S}} \sum_{u:(u,c_j)\in\mathcal{L}} x(s_i, c_j, u, c_j) & = r, & \forall c_j \in \mathcal{C}. \quad (12) \\
\mathrm{C}_3: & \sum_{(s_i,u)\in\mathcal{L}} x(s_i, c_j, s_i, u) & \le 1, & \forall c_j \in \mathcal{C},\ \forall s_i \in \mathcal{S}. \quad (13) \\
\mathrm{C}_4: & \sum_{(u,v)\in\mathcal{L}} \{x(s_i, c_j, u, v)d(u, v)\} & \le B_{req}, & \forall c_j \in \mathcal{C}, \forall s_i \in \mathcal{S}. \quad (14) \\
\mathrm{C}_5: & 2y(s_i, s_j, c_k, u, v) - x(s_i, c_k, u, v) & -x(s_j, c_k, u, v) \le 0 & \quad (15) \\
 & \forall(u, v) \in \mathcal{L}, \forall c_k \in \mathcal{C}, \text{ and } \forall(s_i, s_j) \in \mathcal{S}\times\mathcal{S} & \text{with } s_i \ne s_j. & \\
\mathrm{C}_6: & y(s_i, s_j, c_k, u, v) - x(s_i, c_k, u, v) - x(s_j, c_k, u, v) + 1 & \ge 0 & \quad (16) \\
 & \forall(u, v) \in \mathcal{L}, \forall c_k \in \mathcal{C}, \text{ and } \forall(s_i, s_j) \in \mathcal{S}\times\mathcal{S} & \text{with } s_i \ne s_j. & \\
\mathrm{C}_7: & \sum_{(s_i,s_j)\in\mathcal{S}\times\mathcal{S}} \sum_{(u,v)\in\mathcal{L}} \{y(s_i, s_j, c_k, u, v)p(u, v)\} & \le D_{req}, & \forall c_k \in \mathcal{C}. \quad (17) \\
\mathrm{C}_8: & z(s_k) \ge x(s_k, c_j, s_k, u),\ \forall s_k \in \mathcal{S}, & \forall(s_k, u) \in \mathcal{L},\ c_j \in \mathcal{C}. & \quad (18)
\end{array}$$

Fig. 2. MILP formulation for the SP problem.

Constraint C_1 in Figure 2 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. Constraint C_4 ensures that the delay associated with each description is less than B_{req}. Constraints C_5 and C_6 transform (10) into an MILP. This transformation is needed because for an MILP formulation the constraints should be linear in the variables. Constraint C_7 ensures that the set of paths associated with a client satisfies the diversity constraint D_{req}. Constraint 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(a)). These auxiliary nodes are then treated as potential server locations, with each location providing at most a single description. The delay and the packet loss rate associated with the edges between the potential server location and each auxiliary node are set to zero.

B. Server Placement (SP) Algorithm

The MILP solution presented in Section III-A provides the minimum number of server locations but the complexity associated with it grows exponentially which 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 algorithmic complexity, we now develop a heuristic algorithm for solving the SP problem. In this algorithm, we start with an initial set of server locations and then sequentially add more server locations to satisfy delay and diversity constraints for all client.

The input to the algorithm consists of the graph $\mathcal{G}(\mathcal{N}, \mathcal{L})$, the link delays $d(u, v)$, packet loss rates $p(u, v) \forall (u, v) \in \mathcal{L}$, potential server locations $\mathcal{S}$, client nodes $\mathcal{C}$, the number of descriptions r, delay constraint B_{req}, and diversity constraint D_{req}. Let $\delta_d(s, c)$ be the delay associated with the shortest path w.r.t. the delay metric between two nodes s and c, and let W be a $|\mathcal{N}| \times |\mathcal{N}|$ matrix with $W(i, j) = \delta_d(i, j)$. For an arbitrary client $c \in \mathcal{C}$, let $\triangle(c, \mathcal{S}, B_{req})$ be the number of nodes in $\mathcal{S}$ for which $\delta_d(s, c) \leq B_{req}$. A *delay cover* D_{cov} is defined as the set of servers such that for each client $c \in \mathcal{C}$, $\triangle(c, D_{cov}, B_{req}) \geq r$. The procedure for finding a delay cover is presented in Section III-D. The placement algorithm starts by computing D_{cov}. At any stage of the algorithm, for each client $c \in \mathcal{C}$ we maintain the following information:

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

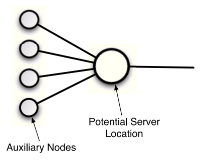
The algorithm recomputes the delay cover if at least one of the clients still requires at least one description, i.e., if $f_c < r$. When the delay cover is recomputed, additional servers are added to the existing delay cover such that for each client c there are $r - f_c$ additional servers with $\delta_d(s, c) \leq B_{req}$. 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 $f_c \leq r$, then 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 `SP` algorithm is presented in Figure 3.

C. *Residual Graph of a Client*

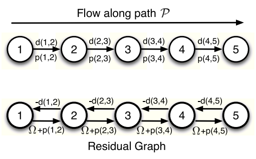
We now define the residual graph $\mathcal{G}_c$ for every client $c \in \mathcal{C}$ and explain how it is determined. For a network $\mathcal{G}(\mathcal{N}, \mathcal{L})$, the residual graph $\mathcal{G}_c(\mathcal{N}, \mathcal{L})$ for a client c is a graph with $\mathcal{N}$ nodes and $\mathcal{L}$ 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 of 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:

```
Procedure: SP ( $\mathcal{G}(\mathcal{N},\mathcal{L}), d(.,.), p(.,.), \mathcal{S}, \mathcal{C}, r, B_{req}, D_{req}$)
Output: $\mathcal{S}_C,\ \mathcal{R}_c,\ c \in \mathcal{C}$
  1) $\mathcal{S}_C = \phi$, $D_{cov} = \phi$, $Itr = 0$, $\mathcal{G}_c = \mathcal{G}$, $f_c = 0$, and $\mathcal{R}_c = \phi$, $\forall c \in \mathcal{C}$.
  2) $W = Floyd\_Warshall(\mathcal{G}(\mathcal{N},\mathcal{L}), d(.,.))$.
  3) 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($s, c, \mathcal{G}_c$) $== 1$,
                RouteFlow($s, c, \mathcal{G}_c$)
                $f_c = f_c + 1$
       c) If $\forall c \in \mathcal{C}$, $f_c \geq r$
            $Itr = 1$.
  4) Remove $s$ from $\mathcal{S}_C$ if $s$ does not provide a description to at least one client.
```

Fig. 3. Pseudocode for the SP algorithm.



(a) Auxiliary node construction to support multiple descriptions at a single server location.



(b) Construction of the residual graph.

Fig. 4. Auxiliary node and residual graph construction.

- 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 a large number.
  - If $f(u, v) < 0$, then $l(u, v) = -f(u, v)d(u, v)$.

A flow is 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 the link weight $l(u, v)$ 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 does not exist a 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 residual graph along path $\mathcal{P}$ is given in Figure 4(b) with $f(u, v) = 0$ initially.

D. Computation of the Delay Cover

The delay cover D_{cov} for a set of clients $\mathcal{C}$ is the set of potential server locations such that $\triangle(c, D_{cov}, B_{req}) \geq r$. At the start of SP algorithm, f_c is set to zero and D_{cov} is empty. We use the following greedy approach to determine an initial value for D_{cov}. For each potential server location, we determine N_s, 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 if no server locations are left. A pseudocode for the DelayCover procedure is presented in Figure 5. 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, i.e., the number of servers in a delay cover may not be sufficient to satisfy the diversity constraint for all clients. At any state 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.

```
Procedure: DelayCover ($\mathcal{G}(\mathcal{N}, \mathcal{L}), W(.), \mathcal{S}, \mathcal{C}, f_c, D_{cov}$)
1.  $T_{f_c} = f_c$, $\forall c \in \mathcal{C}$ and $Itr = 0$
2.  While ($Itr == 0$)
2a.     Obtain $\mathcal{C}_r \subset \mathcal{C}$ s.t. $T_{f_c} < r \; \forall \, c \in \mathcal{C}_r$.
2b.     $\forall s \in \mathcal{S} - D_{cov}, R_s = \sum_{c \in \mathcal{C}_r} 1_{[\delta(s,c) < B_{req}]}$
2c.     Select $s_x$ s.t. $R_{s_x} = \max_{s \in \mathcal{S} - D_{cov}} R_s$
2d.     $D_{cov} = D_{cov} \cup s_x$
2e.     $\forall \, c \in \mathcal{C}_r$ s.t. $\delta(s_x, c) < B_{req}$, Set $T_{f_c} = T_{f_c} + 1$
2f.     If $\forall c \in \mathcal{C}, T_{f_c} \geq r$, $Itr = 1$.
3.  Return $D_{cov}$.
```

Procedure `CheckFlow` (s, c, $\mathcal{G}_c$)
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 constraints.

Procedure `RouteFlow`
Input: s, c, $\mathcal{G}_c$, **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 $\mathcal{G}_c$.

Fig. 5. Pseudocode for `DelayCover`, `CheckFlow`, and `RouteFlow` procedures.

E. *Check Flow and Route Flow Procedures*

Check Flow: `CheckFlow` (see Figure 5) finds the shortest path $\mathcal{P}$ on $\mathcal{G}_c$ from server s to client c w.r.t. $l(.,.)$. The procedure then routes a unit flow along path $\mathcal{P}$ and updates the link weights of the residual graph, as discussed in Section III-C. Finally, `CheckFlow` checks if all resultant paths satisfy delay and diversity bounds. If one path does not satisfy the delay bound or if the diversity bound is violated, the procedure returns a failure.
Route Flow: `RouteFlow` (see Figure 5) updates 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 the residual graph $\mathcal{G}_c$.

IV. PERFORMANCE EVALUATION

We conduct extensive simulations to evaluate the performance of our algorithmic solutions presented in Section III. We first assess the goodness of our solution and then demonstrate the effectiveness of 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 randomly generated Waxman's topology [10] with 50 nodes. The client set $\mathcal{C}$ and server locations $\mathcal{S}$ are randomly selected. Each link (u, v) is assigned a delay value $d(u, v) \sim U[0, 50 \text{ msec}]$ and a packet loss rate $p(u, v) \sim U[0, 0.05]$ that are randomly sampled. Media stream is split into $r = 3$ descriptions. For different values of D_{req} and B_{req}, we obtain the number of server locations that satisfy delay and diversity constraints for a set of client nodes. For each client $c \in \mathcal{C}$, we determine the minimum and maximum delays among the r paths and report averages of both quantities over all clients in Figure 6. We observe that the number of required server locations under SP algorithm is very close to that of optimal (MILP) solution. Average delays of the shortest and longest paths to a client are also comparable with the optimal result.

B. *Performance Evaluation of MDC with Multiple Server Locations*

We now evaluate the performance of MDC with multiple server locations using a packet-level simulator. For these simulations, we use a waxman's network of 50 nodes. The client set $\mathcal{C}$ and server locations $\mathcal{S}$ are randomly generated. Delay $d(i, j) \sim U[10, 100\text{msec}]$ is randomly sampled. Each link has a transmission capacity of 10 Mbps and is associated with

		MILP			SP Algorithm		
B_{req} (msec)	D_{req}	No. of required server locations	Avg. path delay (short) (msec)	Avg. path delay (long) (msec)	No. of required server locations	Avg. path delay (short) (msec)	Avg. path delay (long) (msec)
200	1	3	53.39	147.90	3	53.98	126.74
150	1	4	53.74	126.15	4	56.14	117.92
130	1	Infeasible			Infeasible		
150	0.5	4	53.39	126.15	4	56.14	117.92
150	0.1	4	53.84	123.23	7	53.30	116.07
150	0.05	Infeasible			Infeasible		

Fig. 6. Performance evaluation of the SP algorithm and the MILP solution ($|\mathcal{C}| = 10,\ |\mathcal{S}| = 10$).

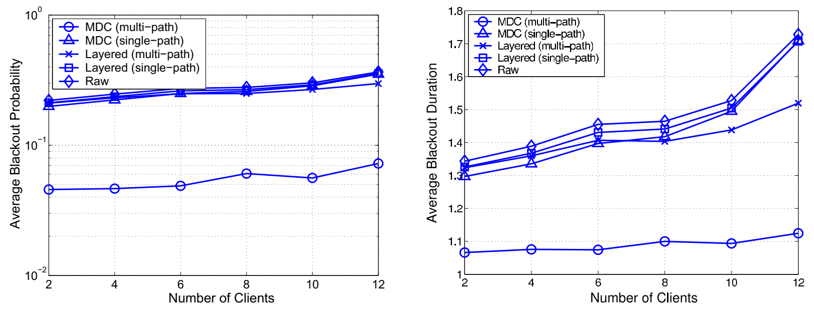


Fig. 7. Blackout performance vs. $|\mathcal{C}|$ ($r = 3$, $|\mathcal{S}| = 10$, $\mathcal{B} = 20000$ bytes, $\mathcal{T}_g = 100$ msec, $\mathcal{T}_b = 10$ msec, and $\rho = 0.9$.)

a two-state (good-bad) Markov model. Sojourn time of the Markov chain in good (bad) state is exponentially distributed with mean $\mathcal{T}_g$ ($\mathcal{T}_b$). Packet is successfully transmitted over a link when the state is good, and is dropped otherwise. Each node is modeled as an M/M/1 queue with a finite buffer $\mathcal{B}$. Each node is also associated with cross-traffic, which enters and exits at the same node and is used to simulate the average load ρ at the node. Each media stream consists of a constant bit rate (CBR) stream of 1000-byte packets (25 packets per second). We consider five different cases for distributing media to a given client community:

1) *MDC-coded stream over multiple paths*: Multiple descriptions of MDC-coded traffic are routed from $|\mathcal{S}_C|$ server locations as determined by SP algorithm.
2) *MDC-coded stream over a single path*: Similar to the first case, but all MDs destined to a client are routed over the shortest path w.r.t. delay from the closest server in $\mathcal{S}_C$.
3) *Layered video over multiple paths*: Instead of MDC, we use layered video with r layers. Multiple paths are used to route the different layers, with the shortest path used for the base layer, remaining paths are used for enhancement layers.
4) *Layered video 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.
5) *Raw media stream*: For a given client c, the closest server location from the set of chosen server $\mathcal{S}_c \in \mathcal{S}$ is used to route CBR traffic to the client.

Appropriate relative overheads are considered while generating MDC-coded (5% for $r = 3$) and layered video (5% for three layers) streams (see [4], [8]). We define "blackout" as a playback instance when no portion of the required frame is available for playout at the receiver. Figure 7 depicts the performance of the five considered approaches as a function of $|\mathcal{C}|$. Figure 7(a) shows the average blackout probability, defined as the fraction of the playback instances at which blackout was encountered (averaged over all clients). Figure 7(b) depicts the average blackout duration. Based on the two figures, we conclude the following:

- Significant reduction in average blackout probability is observed when MDC-coded streaming with multiple paths is used. This is a direct consequence of diverse path

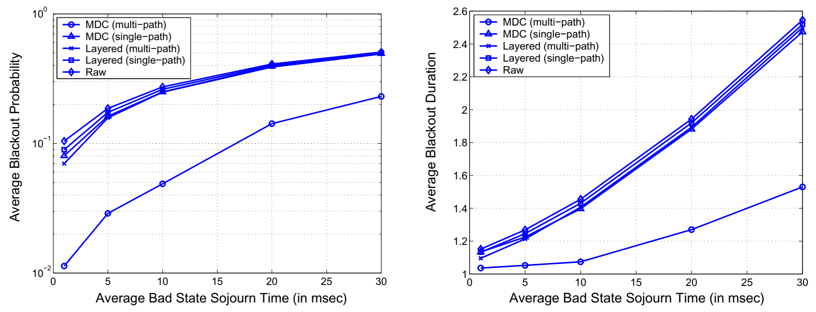


Fig. 8. Blackout performance vs. $\mathcal{T}_b$ ($r = 3$, $|\mathcal{S}| = 10$, $|\mathcal{C}| = 6$, $\mathcal{B} = 20000$ bytes, $\mathcal{T}_g = 100$ msec, and $\rho = 0.9$).

routing.

- Improvement in blackout duration under MDC-coded multiple path streaming is due to independence of the packet loss rates of the diverse paths chosen by the SP algorithm.
- Performance of layered scheme depends on the loss rate observed by the base layer.

Figure 8 depicts the performance of the five approaches as a function of the $\mathcal{T}_b$. It can be noted that the average blackout probability and the average blackout duration monotonically increase 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.

V. Conclusions

We considered the problem of finding optimal server locations which provide MDC-coded streams to a client community using diverse path routing. We show that for MDC-coded media streams, the probability of blackout is significantly dependent on the reliability of common link. We considered the server placement problem for a given client community, which minimizes 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 are used to compare the performance of placement algorithm with optimal results obtained using MILP.

References

[1] R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall Inc., 1993.
[2] 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.
[3] 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.
[4] 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.
[5] M. Garey and D. Johnson. *Computers and Intractability: Theory of NP-Completeness*. W. H. Freeman, 2000.
[6] M. Rocha, M. Maia, T. Cunha, J. Almeida, and S. Campos. Scalable media streaming to interactive users. In *Proceedings of the ACM Multimedia Conference*, pages 966–975, Singapore, Nov. 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), 2002.
[8] D. Taubman and M. Marcellin. *JPEG2000: Image Compression Fundamentals, Standards and Practice*. Springer, 2001.
[9] 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.
[10] B. M. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 69:1617–1622, Dec. 1988.