MinDelay: Low-Latency Joint Caching and Forwarding for Multi-hop Networks

Milad Mahdian, Edmund Yeh
Department of Electrical and Computer Engineering
Northeastern University
Email: {mmahdian, eyeh}@ece.neu.edu

***Abstract*—We present a new unified framework for minimizing congestion-dependent network cost in caching networks by jointly optimizing forwarding and caching strategies. As caching variables are integer-constrained, the resulting optimization problem is NP-hard. To make progress, we focus on a relaxed version of the optimization problem, where caching variables are allowed to be real-valued. We develop necessary optimality conditions for the relaxed problem, and leverage this result to design MinDelay, an adaptive and distributed joint forwarding and caching algorithm, based on the conditional gradient algorithm. The MinDelay algorithm elegantly yields feasible routing variables and integer caching variables at each iteration, and can be implemented in a distributed manner with low complexity and overhead. Over a wide range of network topologies, simulation results show that MinDelay typically has significantly better delay performance in the low to moderate request rate regions. Moreover, the MinDelay and VIP algorithms complement each other in delivering superior delay performance across the entire range of request arrival rates.**

I. Introduction

Caching networks have many applications including information-centric networking (ICN), content delivery networks (CDNs) and peer-to-peer networks. In particular, ICN architectures have gained a significant amount of attention over the past few years. One prominent issue in these networks is how to jointly design traffic engineering and caching strategies to maximally exploit the bandwidth and storage resources of the network for optimal performance. While traffic engineering and caching have been investigated separately for many years, their joint optimization is still an under-explored area.

There have been many recent papers on caching strategies in ICN networks [1], [2], [3], [4], [5], [6], [7]. When designing and evaluating the effectiveness of a cache management scheme for such networks, the primary performance metrics have been cache hit probability [3], [5], the reduction of the number of hops to retrieve the requested content [4], or content download delay [6].

Similarly, there have been a number of attempts to enhance traffic engineering in ICN [8], [9], [10], [11], [12]. In [8], Carofiglio et al., formulate the problem of joint multipath congestion control and request forwarding in ICN as an optimization problem. By decomposing the problem into two subproblems of maximizing user throughput and minimizing overall network cost, they develop a receiver-driven window-based congestion control algorithm and a hop-by-hop dynamic request forwarding algorithm which aim to balance the number of pending Interest Packets of each content object (flow) across the outgoing interfaces at each node. Unfortunately, the work in [8] does not consider caching policies.

In [10], Detti et al. study different multipath forwarding strategies. In particular, they compare strategies based on balancing Round-Trip Time (RTT) averages of each flow across the outgoing interfaces with strategies based on balancing pending interest packets across outgoing interfaces.

In [13], Yeh et al., present one of the first unified frameworks for joint forwarding and caching for ICN networks with general topology, in which a virtual control plane operates on the user demand rate for data objects in the network, and an actual plane handles Interest Packets and Data Packets. They develop VIP, a set of distributed and dynamic forwarding and caching algorithms which adaptively maximizes the user demand rate the ICN can satisfy.

In this work, we present a new unified framework for minimizing congestion-dependent network cost by jointly choosing node-based forwarding and caching variables, within a quasi-static network scenarios where user request statistics vary slowly. We consider the network cost to be the sum of link costs, expressed as increasing and convex functions of the traffic rate over the links. When link cost functions are chosen according to an M/M/1 approximation, minimizing the network cost corresponds to minimizing the average request fulfillment delay in the network. As caching variables are integer-constrained, the resulting joint forwarding and caching (JFC) optimization problem is NP-hard. To make progress toward an approximate solution, we focus on a relaxed version of the JFC problem (RJFC), where caching variables are allowed to be real-valued. Using techniques first introduced in [14], we develop necessary optimality conditions for the RJFC problem. We then leverage this result to design MinDelay, an adaptive and distributed joint forwarding and caching algorithm for the original JFC problem, based on a version of the conditional gradient, or Frank-Wolfe algorithm. The MinDelay algorithm elegantly yields feasible routing variables and integer caching variables at each iteration, and can be implemented in a distributed manner with low complexity and overhead.

Finally, we implement the MinDelay algorithm using our Java-based network simulator, and present extensive experi-

This work was supported by National Science Foundation grant CNS-1423250, CNS-1718355 and a Cisco Systems research grant.

978-1-5386-3180-5/18/$31.00 ©2018 IEEE

mental results. We consider three competing schemes, including the VIP algorithm [13], which directly competes against MinDelay as a jointly optimized distributed and adaptive forwarding and caching scheme. Over a wide range of network topologies, simulation results show that while the VIP algorithm performs well in high request arrival rate regions, MinDelay typically has significantly better delay performance in the low to moderate request rate regions. Thus, the MinDelay and VIP algorithms complement each other in delivering superior delay performance across the entire range of request arrival rates.

II. Network Model

Consider a general multi-hop network modeled by a directed and (strongly) connected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ and $\mathcal{E}$ are the node and link sets, respectively. A link $(i, j) \in \mathcal{E}$ corresponds to a unidirectional link, with capacity $C_{ij} > 0$ (bits/sec). We assume a content-centric setting, e.g. [15], where each node can request any data object from a set of objects $\mathcal{K}$. A data object is requested by a requesting node generating an Interest Packet retrieving a corresponding Data Packet which contains the data object. We assume the Forwarding Interest Base (FIB) tables has already been populated at each node, using a loop-free content-based routing algorithm. Further, we assume *symmetric routing*, where Data Packets containing the requested data object take the same path as their corresponding Interest Packets, in the reverse direction. Our framework can be extended to the case where each data object consists of a sequence of data chunks, each contained in a Data Packet which is retrieved by a corresponding Interest Packet.

For simplicity, we do not consider interest suppression, whereby multiple Interest Packets requesting the same data object are collapsed into one forwarded Interest Packet. The algorithm we develop can be extended to include Interest suppression, by introducing a virtual plane in the manner of [13].

For $k \in \mathcal{K}$, let $src(k)$ be the source node of content object k. Each node in the network has a local cache of capacity c_i (object units), and can optionally cache Data Packets passing through on the reverse path. Interest Packets requesting a given data object can enter the network at any node, and exit the network upon being satisfied by a matching Data Packet at the content source for the object, or at the nodes which decide to cache the object. For simplicity, we assume all data objects have the same size L (bits). The results of the paper can be extended to the more general case where object sizes differ.

We focus on quasi-static network scenarios where user request statistics vary slowly. Let $r_i(k) \geq 0$ be the average exogenous rate (in requests/sec) at which requests for data object k arrive (from outside the network) to node i. Let $t_i(k)$ be the total average arrival rate of object k requests to node i. Thus, $t_i(k)$ includes both the exogenous arrival rate $r_i(k)$ and the endogenous arrival traffic which is forwarded from other nodes to node i.

Let $x_i(k) \in \{0, 1\}$ be the (integer) caching decision variable for object k at node i, where $x_i(k) = 1$ if object k is cached at node i and $x_i(k) = 0$ otherwise. Thus, $t_i(k)x_i(k)$ is the portion of the total incoming request rate for object k which is satisfied from the local cache at node i and $t_i(k)(1-x_i(k))$ is the portion forwarded to neighboring nodes based on the forwarding strategy. Furthermore, let $\phi_{ij}(k) \in [0, 1]$ be the (real-valued) fraction of the traffic $t_i(k)(1-x_i(k))$ forwarded over link (i, j) by node $i \neq src(k)$. Thus, $\sum_{j\in\mathcal{O}(i,k)} \phi_{ij}(k) = 1$, where $\mathcal{O}(i, k)$ is the set of neighboring nodes for which node i has a FIB entry for object k. Therefore, the total average incoming request rate for object k to node i is

$$t_i(k) = r_i(k) + \sum_{l\in\mathcal{I}(i,k)} t_l(k)(1 - x_l(k))\phi_{li}(k), \qquad (1)$$

where $\mathcal{I}(i, k)$ is the set of neighboring nodes of i which have FIB entries for node i for object k.

Next, let F_{ij} be the Data Packet traffic rate (in bits/sec) corresponding to the total request rate (summed over all data objects) forwarded on link $(i, j) \in \mathcal{E}$:

$$F_{ij} = \sum_{k\in\mathcal{K}} L \cdot t_i(k)(1 - x_i(k))\phi_{ij}(k). \qquad (2)$$

Note that by routing symmetry, the Data Packet traffic of rate F_{ij} actually travels on the reverse link (j, i).

As in [14], we assume the total network cost is the sum of traffic-dependent link costs. The cost on link $(j, i) \in \mathcal{E}$ is due to the Data Packet traffic of rate F_{ij} generated by the total request rate forwarded on link (i, j), as in (2). We therefore denote the cost on link (j, i) as $D_{ij}(F_{ij})$ to reflect this relationship.[1] We assume $D_{ij}(F_{ij})$ is increasing and convex in F_{ij}. To implicitly impose the link capacity constraint, we assume $D_{ij}(F_{ij}) \to \infty$ as $F_{ij} \to C_{ji}^-$ and $D_{ij}(F_{ij}) = \infty$ for $F_{ij} \geq C_{ji}$. As an example,

$$D_{ij}(F_{ij}) = \frac{F_{ij}}{C_{ji} - F_{ij}}, \qquad \text{for } 0 \leq F_{ij} < C_{ji}. \qquad (3)$$

gives the average number of packets waiting for or under transmission at link (j, i) under an $M/M/1$ queuing model [16]. Summing over all links, the network cost $\sum_{(i,j)} D_{ij}(F_{ij})$ gives the average total number of packets in the network, which, by Little's Law, is proportional to the average system delay of packets in the network.

III. Optimization Problem

We now pose the Joint Forwarding and Caching (JFC) optimization problem in terms of the forwarding variables $(\phi_{ij}(k))_{(i,j)\in\mathcal{E},k\in\mathcal{K}}$ and the caching variables $(x_i(k))_{i\in\mathcal{N},k\in\mathcal{K}}$ as follows:

$$\begin{cases} \min \sum_{(i,j)\in\mathcal{E}} D_{ij}(F_{ij}) & \\ \text{subject to:} & \\ \sum_{j\in\mathcal{O}(i,k)} \phi_{ij}(k) = 1, & \text{for all } i \in \mathcal{N}, k \in \mathcal{K} \\ \phi_{ij}(k) \geq 0, & \text{for all } (i, j) \in \mathcal{E}, k \in \mathcal{K} \\ \sum_{k\in\mathcal{K}} x_i(k) \leq c_i, & \text{for all } i \in \mathcal{N} \\ x_i(k) \in \{0, 1\}, & \text{for all } i \in \mathcal{N}, k \in \mathcal{K}. \end{cases} \qquad (4)$$

[1] Since Interest Packets are small compared to Data Packets, we do not account for costs associated with the Interest Packet traffic on link (j, i).

The above mixed-integer optimization problem can be shown to be NP-hard [17]. To make progress toward an approximate solution, we relax the problem by removing the integrality constraint in (4). We formulate the Relaxed JFC (RJFC) problem by replacing the integer caching decision variables $x_i(k) \in \{0,1\}$ by the real-valued variables $\rho_i(k) \in [0,1]$:

$$\begin{cases} \min D \triangleq \sum_{(i,j)\in\mathcal{E}} D_{ij}(F_{ij}) \\ \text{subject to:} \\ \sum_{j\in\mathcal{O}(i,k)} \phi_{ij}(k) = 1, & \text{for all } i \in \mathcal{N}, k \in \mathcal{K} \\ \phi_{ij}(k) \geq 0, & \text{for all } (i,j) \in \mathcal{E}, k \in \mathcal{K} \\ \sum_{k\in\mathcal{K}} \rho_i(k) \leq c_i, & \text{for all } i \in \mathcal{N} \\ 0 \leq \rho_i(k) \leq 1, & \text{for all } i \in \mathcal{N}, k \in \mathcal{K}. \end{cases} \tag{5}$$

It can be shown that D in (5) is non-convex with respect to (w.r.t.) the forwarding and caching variables (ϕ, ρ), where $\phi \equiv (\phi_{ij}(k))_{(i,j)\in\mathcal{E},k\in\mathcal{K}}$ and the caching variables $\rho \equiv (x_i(k))_{i\in\mathcal{N},k\in\mathcal{K}}$. In this work, we use the RJFC formulation to develop an adaptive and distributed forwarding and caching algorithm for the JFC problem.

We proceed by computing the derivatives of D with respect to the forwarding and caching variables, using the technique of [14]. For the forwarding variables, the partial derivatives can be computed as

$$\frac{\partial D}{\partial \phi_{ij}(k)} = (1-\rho_i(k)) L t_i(k) \delta_{ij}(k), \tag{6}$$

where the marginal forwarding cost is

$$\delta_{ij}(k) = D'_{ij}(F_{ij}) + \frac{\partial D}{\partial r_j(k)}. \tag{7}$$

Note that $\frac{\partial D}{\partial r_j(k)}$ in (7) stands for the marginal cost due to a unit increment of object k request traffic at node j. This can be computed recursively by

$$\frac{\partial D}{\partial r_j(k)} = 0, \text{ if } j = src(k),$$

$$\frac{\partial D}{\partial r_i(k)} = (1-\rho_i(k)) L \sum_{j=\mathcal{O}(i,k)} \phi_{ij}(k)\delta_{ij}(k), \text{ if } i \neq src(k). \tag{8}$$

Finally, we can compute the partial derivatives w.r.t. the (relaxed) caching variables as follows:

$$\frac{\partial D}{\partial \rho_i(k)} = -L t_i(k) \sum_{j=\mathcal{O}(i,k)} \phi_{ij}(k)\delta_{ij}(k). \tag{9}$$

The minimization in (5) is equivalent to minimizing the Lagrangian function $L(F,\lambda,\mu) = \sum_{(i,j)\in\mathcal{E}} D_{ij}(F_{ij}) - \sum_{i,k}\lambda_{ik}\left(\sum_j \phi_{ij}(k) - 1\right) + \sum_i \mu_i \left(\sum_{k\in\mathcal{K}} \rho_i(k) - c_i\right)$, subject to the following constraints:

$$\begin{aligned} 0 \leq \rho_i(k) \leq 1, &\quad \text{for all } i \in \mathcal{N}, k \in \mathcal{K}, \\ \phi_{ij}(k) \geq 0, &\quad \text{for all } (i,j) \in \mathcal{E}, k \in \mathcal{K}, \\ \mu_i \geq 0, &\quad \text{for all } i \in \mathcal{N}. \end{aligned}$$

A set of necessary conditions for a local minimum to the RJFC problem can now be derived as

$$\frac{\partial D}{\partial \phi_{ij}(k)} \begin{cases} = \lambda_{ik}, & \text{if } \phi_{ij}(k) > 0 \\ \geq \lambda_{ik}, & \text{if } \phi_{ij}(k) = 0 \end{cases} \tag{10}$$

$$\frac{\partial D}{\partial \rho_i(k)} \begin{cases} = -\mu_i, & \text{if } 0 < \rho_i(k) < 1 \\ \geq -\mu_i, & \text{if } \rho_i(k) = 0 \\ \leq -\mu_i, & \text{if } \rho_i(k) = 1 \end{cases} \tag{11}$$

with the complementary slackness condition

$$\mu_i \left(\sum_{k\in\mathcal{K}} \rho_i(k) - c_i \right) = 0, \text{for all } i \in \mathcal{N}. \tag{12}$$

The conditions (10)-(12) are necessary for a local minimum to the RJFC problem, but upon closer examination, it can be seen that they are not sufficient for optimality. An example from [14] shows a forwarding configuration (without caching) where (10) is satisfied at every node, and yet the operating point is not optimal. In that example, $t_i(k) = 0$ at some node i, which leads to (10) being automatically satisfied for node i. This degenerate example also applies to the joint forwarding and caching setting considered here.

A further issue arises for the joint forwarding and caching setting where $\rho_i(k) = 1$ for some i and k. In this case, the condition in (10) at node i is automatically satisfied for every $j \in \mathcal{O}(i,k)$, and yet the operating point need not be optimal. A simple network example to illustrate this issue is described in [18], which is omitted here for brevity.

In both cases, when $\rho_i(k) = 1$ or $t_i(k) = 0$, node i still needs to assign forwarding variables for object k in the optimal way. By removing the term $t_i(k)(1-\rho_i(k))$ from (10), non-optimal forwarding choices in these degenerate cases are prevented. Furthermore, since the term $t_i(k)(1-\rho_i(k))$ is not a function of $j \in \mathcal{O}(i,k)$, it can also be removed from condition (10) when $t_i(k)(1-\rho_i(k)) > 0$. We therefore focus on the following modified conditions:

$$\delta_{ij}(k) \begin{cases} = \delta_i(k), & \text{if } \phi_{ij}(k) > 0 \\ \geq \delta_i(k), & \text{if } \phi_{ij}(k) = 0. \end{cases} \tag{13}$$

$$t_i(k)\delta_i(k) \begin{cases} = \mu'_i, & \text{if } 0 < \rho_i(k) < 1 \\ \leq \mu'_i, & \text{if } \rho_i(k) = 0 \\ \geq \mu'_i, & \text{if } \rho_i(k) = 1. \end{cases} \tag{14}$$

where $\delta_i(k) = \min_{m\in\mathcal{O}(i,k)} \delta_{im}(k)$, and $\mu'_i = \mu_i / L$.

In (14), we used the fact that $\sum_{j=\mathcal{O}(i,k)} \phi_{ij}(k)\delta_{ij}(k) = \delta_i(k)$ if condition (13) is satisfied. Condition (13) suggests that requests for object k arriving at node i, should be forwarded on an outgoing link (i,j) whose "marginal cost" $\delta_{ij}(k)$ is minimal. Similarly, condition (14) suggests a structured caching policy, where one sorts the data objects in decreasing order with respect to the "cache scores" $\{t_i(k)\delta_i(k)\}$, and cache the top c_i objects, i.e. set $\rho_i(k) = 1$ for the top c_i objects. This is indeed the main idea underlying our proposed caching algorithm described in the next section.

IV. Distributed Algorithm: MinDelay

The conditions in (13)-(14) give the general structure for a joint forwarding and caching algorithm for solving the RJFC problem. To describe the joint forwarding and caching algorithm, we first describe a protocol for calculating the marginal costs, and then describe an algorithm for updating the forwarding and caching variables.

Note that each node i can estimate, as a time average, the link traffic rate F_{ij} for each outgoing link (i,j). This can be done by either measuring the rate of received Data Packets on each of the corresponding incoming links (j,i), or by measuring the request rate of Interest Packets forwarded on the outgoing links (i,j). Thus, given a functional form for $D_{ij}(.)$, node i can compute $D'_{ij}(F_{ij})$.

To update the marginal forwarding costs, the nodes use the following protocol. Each node i waits until it has received the value $\partial D/\partial r_j(k)$ from each of its upstream neighbors with respect to object k (with the convention $\partial D/\partial r_{src(k)}(k)=0$). Node i then calculates $\partial D/\partial r_i(k)$ according to (8) and broadcasts this to all of its downstream neighbors with respect to k. The information propagation can be done by either piggybacking on the Data Packets of the corresponding object, or by broadcasting a single message regularly to update the marginal forwarding costs of all the content objects at once.

Having described the protocol for calculating marginal costs, we now specify the algorithm for updating the forwarding and caching variables. Our algorithm is based on the conditional gradient or Frank-Wolfe algorithm [19]. Let

$$\Phi^n = \begin{bmatrix} \left(\phi^n_{ij}(k)\right)_{i\in\mathcal{N},k\in\mathcal{K},j\in\mathcal{O}(i,k)} \\ \left(\rho^n_i(k)\right)_{i\in\mathcal{N},k\in\mathcal{K}} \end{bmatrix}$$

be the vector of forwarding and caching variables at iteration n. Then, the conditional gradient method is given by

$$\Phi^{n+1} = \Phi^n + a^n(\bar{\Phi}^n - \Phi^n), \tag{15}$$

where $a^n \in (0,1]$ is a positive stepsize, and $\bar{\Phi}^n$ is the solution of the direction finding subproblem

$$\bar{\Phi}^n \in \arg\min_{\Phi\in F} \triangledown D(\Phi^n)'(\Phi - \Phi^n). \tag{16}$$

Here, $\triangledown D(\Phi^n)$ is the gradient of the objective function with respect to the forwarding and caching variables, evaluated at Φ^n. The set F is the set of forwarding and caching variables Φ satisfying the constraints in (5), seen to be a bounded polyhedron.

In applying the conditional gradient algorithm, we encounter the same problem regarding degenerate cases as seen in Section III with respect to optimality conditions. Note that when $t_i(k)(1-\rho_i(k))=0$, the $\frac{\partial D}{\partial \phi_{ij}(k)}$ component of $\triangledown D(\Phi^n)$ is zero, and thus provides no useful information for the optimization in (16) regarding the choice of $\bar{\Phi}^n$. On the other hand, when $t_i(k)(1-\rho_i(k))>0$, eliminating this term from $\frac{\partial D}{\partial \phi_{ij}(k)}$ in (16) does not change the choice of $\bar{\Phi}^n$, since $t_i(k)(1-\rho_i(k)) > 0$ is not a function of $j \in \mathcal{O}(i,k)$. Motivated by this observation, we define

$$G(\Phi^n) \triangleq \begin{bmatrix} \left(\delta^n_{ij}(k)\right)_{i\in\mathcal{N},k\in\mathcal{K},j\in\mathcal{O}(i,k)} \\ \left(-t^n_i(k)\sum_{j=\mathcal{O}(i,k)}\phi^n_{ij}(k)\delta^n_{ij}(k)\right)_{i\in\mathcal{N},k\in\mathcal{K}} \end{bmatrix},$$

where $\delta^n_{ij}(k)$ and $t^n_i(k)$ are the marginal forwarding costs and total request arrival rates, respectively, evaluated at Φ^n.

We consider a modified conditional gradient algorithm where the direction finding subproblem is obtained by

$$\bar{\Phi}^n \in \arg\min_{\Phi\in F} G(\Phi^n)'(\Phi - \Phi^n),$$

which is easily seen to be separable into two subproblems.

The subproblem for $(\phi_{ij}(k))$ is given by

$$\begin{cases} \min \sum_{(i,k)}\sum_{j\in\mathcal{O}(i,k)} \delta^n_{ij}(k)(\phi_{ij}(k) - \phi^n_{ij}(k)) \\ \text{subject to:} \\ \sum_{j\in\mathcal{O}(i,k)}\phi_{ij}(k) = 1, \qquad \text{for all } i\in\mathcal{N}, k\in\mathcal{K} \\ \phi_{ij}(k)\geq 0, \qquad \text{for all } i\in\mathcal{N}, k\in\mathcal{K}, j\in\mathcal{O}(i,k). \end{cases} \tag{17}$$

where

$$\delta^n_{ij}(k) = D'_{ij}(F^n_{ij}) + \frac{\partial D}{\partial r^n_j(k)}. \tag{18}$$

It is straightforward to verify that a solution $\bar{\phi}^n_i(k) = (\bar{\phi}^n_{ij}(k))_{j\in\mathcal{O}(i,k)}$ to (17) has all coordinates equal to zero except for one coordinate, say $\bar{\phi}^n_{im}(k)$, which is equal to 1, where $m \in \arg\min_{j\in\mathcal{O}(i,k)}\delta^n_{ij}(k)$, corresponds to an outgoing interface with the minimal marginal forwarding cost.

The subproblem for $(\rho_i(k))$ is equivalent to

$$\begin{cases} \max \sum_{(i,k)} \omega^n_i(k)(\rho_i(k) - \rho^n_i(k)) \\ \text{subject to:} \\ \sum_{k\in\mathcal{K}}\rho_i(k) \leq c_i, \qquad \text{for all } i\in\mathcal{N} \\ 0\leq \rho_i(k)\leq 1, \qquad \text{for all } i\in\mathcal{N}, k\in\mathcal{K}. \end{cases} \tag{19}$$

where $\omega^n_i(k) = t^n_i(k)\left(\sum_{j\in\mathcal{O}(i,k)}\phi^n_{ij}(k)\delta^n_{ij}(k)\right)$. The subproblem (19) is a max-weighted matching problem which has an integer solution. For node i, let $\omega^n_i(k_1)\geq\omega^n_i(k_2)\geq\ldots\geq\omega^n_i(k_{|\mathcal{K}|})$ be a re-ordering of the $\omega^n_i(k)$'s in decreasing order. A solution $\bar{\rho}^n_i$ to (19) has $\bar{\rho}^n_i(k)=1$ for $k\in\{k_1,k_2,\ldots,k_{c_i}\}$, and $\bar{\rho}^n_i(k)=0$ otherwise. That is, $\bar{\rho}^n_i(k)=1$ for the c_i objects with the largest $\omega^n_i(k)$ values, and $\bar{\rho}^n_i(k)=0$ otherwise.

As mentioned above, the solutions $\bar{\rho}^n_i$ to (19) are integer-valued at each iteration. However, for a general stepsize $a^n\in(0,1]$, the (relaxed) caching variables corresponding to the update in (15) may not be integer-valued at each iteration. In particular, this would be true if the stepsize follows a diminishing stepsize rule. Although one can explore rounding techniques and probabilistic caching techniques to obtain feasible integer-valued caching variables $x^n_i(k)$ from continuous-valued relaxed caching variables $\rho^n_i(k)$ [7], this would entail additional computational and communication complexity.

Since we are focused on distributed, low-complexity forwarding and caching algorithms, we require $\rho^n_i(k)$ to be either 0 or 1 at each iteration n. This is realized by choosing the

stepsize $a^n = 1$ for all n. In this case, the update equation (15) is reduced to:

$$\Phi^{n+1} = \bar{\Phi}^n,$$

where $\bar{\Phi}^n$ is the solution to (17) and (19). That is, the solutions to the direction finding subproblems provide us with forwarding and caching decisions at each iteration. We now summarize the elegant MinDelay forwarding and caching algorithm.

MinDelay Forwarding Algorithm: At each iteration n, each node i and for each object k, the forwarding algorithm chooses the outgoing link (i, m) to forward requests for object k, where m is chosen according to

$$m \in \arg \min_{j \in \mathcal{O}(i,k)} \delta_{ij}^n(k). \tag{20}$$

That is, requests for object k are forwarded on an outgoing link with the minimum marginal forwarding cost.

MinDelay Caching Algorithm: At each iteration n, each node i calculates a cache score $CS^n(i, k)$ for each object k according to

$$CS^n(i, k) \triangleq t_i^n(k)\delta_i^n(k). \tag{21}$$

where $\delta_i^n(k) \equiv \min_{j \in \mathcal{O}(i,k)} \delta_{ij}^n(k)$. Upon reception of data object k_{new} not currently in the cache of node i, if the cache is not full, then k_{new} is cached. If the cache is full, then $CS^n(i, k_{new})$ is computed, and compared to the lowest cache score among the currently cached objects, denoted by $CS^n(i, k_{min})$. If $CS^n(i, k_{new}) > CS^n(i, k_{min})$, then replace k_{min} with k_{new}. Otherwise, the cache contents stay the same.

The cache score given in (21) for a given content k at node i is the minimum marginal forwarding cost for object k at i, multiplied by the total request rate for k at i. By caching the data objects with the highest cache scores, each node maximally reduces the total cost of forwarding request traffic.

One drawback of using stepsize $a^n = 1$ in the MinDelay algorithm is that it makes studying the asymptotic behavior of the algorithm difficult. Nevertheless, in extensive simulations shown in the next section, we observe that the algorithm behaves in a stable manner asymptotically. Moreover, the MinDelay significantly outperforms several state-of-the-art caching and forwarding algorithms in important operating regimes.

V. Simulation Experiments

In this section, we present the results of extensive simulations performed using our Java-based ICN Simulator. We have considered three competing schemes for comparison with MinDelay. First, we consider the VIP joint caching and forwarding algorithm introduced in [13]. This algorithm uses a backpressure (BP)-based scheme for forwarding and a stable caching algorithm, both based on VIP (Virtual Interest Packet) queue states [13]. In the VIP algorithm discussed in [13], multiple Interest Packets requesting the same Data Packet are aggregated. Since we do not consider Interest Packet aggregation in this paper, we compare MinDelay with a version of VIP without Interest aggregation, labeled BP. We consider the VIP algorithm (or BP) to be the direct competitor with MinDelay, since to the best of our knowledge, it is the only other scheme that explicitly jointly optimizes forwarding and caching to reduce congestion-dependent costs in general multi-hop networks.

The other two approaches implemented here are based on the LFU cache eviction policy. We note that for stationary input request processes, the performance of LFU is typically much better than those of LRU and FIFO. In the first approach, denoted by LFUM-PI, multipath request forwarding is based on the scheme proposed in [8]. Here, the forwarding decision is made as follows: an Interest Packet requesting a given object is forwarded on an outgoing interface with a probability inversely proportional to the number of Pending Interest (PI) Packets for that object on that outgoing interface. In the second LFU-based approach implemented here, denoted by LFUM-RTT and described in [10], an Interest Packet requesting an object is forwarded on an outgoing interface with a probability inversely proportional to the exponentially weighted moving average of the RTT recorded for that object on that outgoing interface.

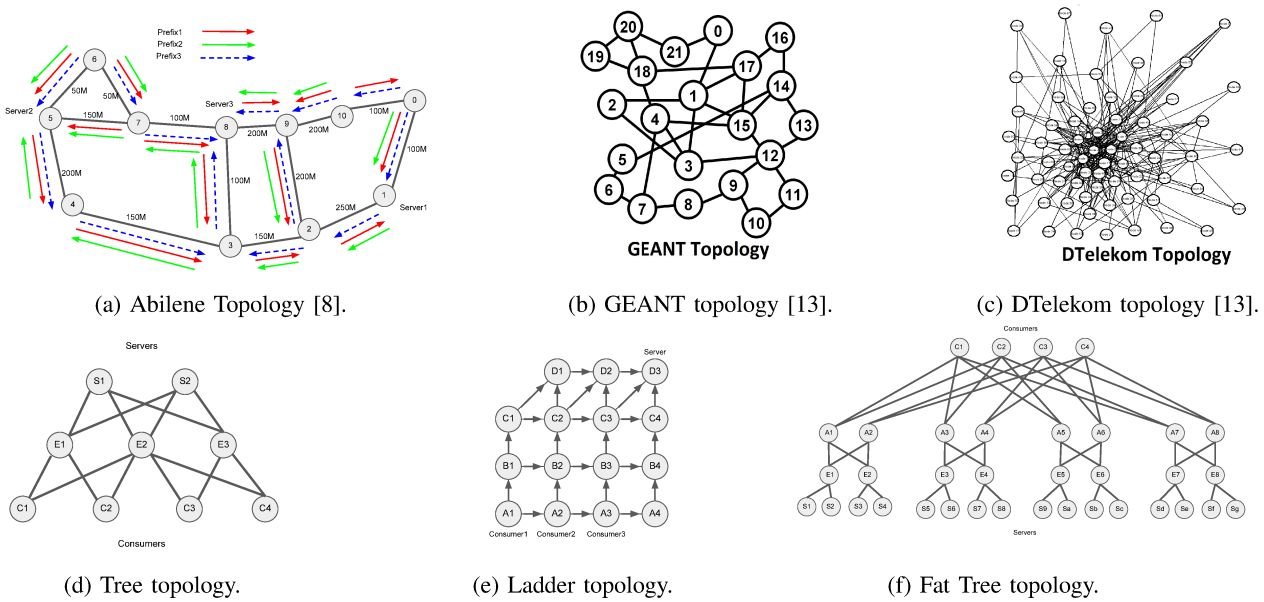
We tested the MinDelay forwarding and caching algorithm against the described approaches on several well-known topologies depicted in Fig. 1. In the following, we explain the simulation scenarios and results in detail.

The simulator emulates all the ICN functionalities, including content-based loop-free routing and the corresponding population of the FIBs. Each simulation generates requests for 1000 seconds and terminates when all the requested packets are fulfilled. For the MinDelay implementation, we update the marginal forwarding costs given in (18) at the beginning of each update interval (with a length between 2-5 seconds), and cache the results in a sorted array for future use. Hence, the forwarding decision given in (20) takes $O(1)$ operations.

For the caching strategy in MinDelay, the forwarder regularly updates the cache score of the currently-cached contents using (21) at the beginning of the update intervals and keeps a sorted list of the cached content objects using a hash table and a priority queue. When a new Data Packet arrives, the forwarder computes its cache score, and compares the score with the lowest cache score among the currently-cached content objects. If the score of the incoming Data Packet is higher than the current lowest cache score, the forwarder replaces the corresponding cached object with the incoming one. Otherwise, the cached contents remain the same.

In all topologies shown in Figure 1, the number of content objects is 5000. Each requester requests a content object according to a Zipf distribution with power exponent $\alpha = 0.75$, by generating an Interest Packet each of size 1.25 KBytes. All content objects are assumed to have the same size and can be packaged into Data Packets of size 500 KBytes. The link capacity of all the topologies, except in Abilene topology illustrated in Fig. 1a, is 50 Mbps.

We first consider the Abilene topology [8] depicted in Figure 1a. There are three servers, at nodes 1, 5, and 8, each serving 1/3 of the content objects. That is, object k is served



(a) Abilene Topology [8].

(b) GEANT topology [13].

(c) DTelekom topology [13].

(d) Tree topology.

(e) Ladder topology.

(f) Fat Tree topology.

Fig. 1: Network topologies used for the simulations.

by server k mod $3+1$ for $k = 1, 2, \ldots, 5000$. The other eight nodes of the topology request objects according to Zipf distribution with $\alpha = 0.75$. Also, each requester has a content store of size 250 MBytes, or equivalently 500 content objects.

In the GEANT and DTelekom topologies, illustrated in Figures 1b and 1c, there are 22 and 68 nodes in the network, respectively. All nodes request content objects. Each content object is randomly assigned to one of the nodes in the network as its source node. Each node has a content store of size 250 MBytes, or equivalently 500 content objects.

In the Tree topology, depicted in Figure 1d, there are four requesting nodes at the leaves, C1, C2, C3 and C4. There are three edge nodes, E1, E2, and E3. Each content object is randomly assigned to one of the two source nodes, S1 and S2. Each requesting and edge node has a content store of size 125 MBytes, or equivalently 250 content objects.

In the Ladder topology [8], depicted in Figure 1e, there are three requesting nodes, A1, A2 and A3. The source of all the content objects is at node D3. Each node in the network, except node D3, has a content store of size 125 MBytes, or equivalently 250 content objects.

Finally, in the Fat Tree topology, depicted in Figure 1f, requesters are at the roots, i.e., nodes C1, C2, C3 and C4. There are 16 servers at the leaves. In this topology, each content object is randomly assigned to two servers, one chosen from the first 8 servers, and the other from the second 8 servers. All the requesting nodes as well as Aggregation and Edge nodes have a content store, each of size 125 MBytes, or equivalently 250 content objects.

In Figure 2, the results of the simulations are plotted. The figures illustrate the performance of the implemented schemes in terms of total network delay for satisfying all generated requests (in seconds) versus the arrival rate in requests/node/second, respectively. We define the delay for a request as the difference between the creation time of the Interest Packet and the arrival time of its corresponding Data Packet at the requesting node.

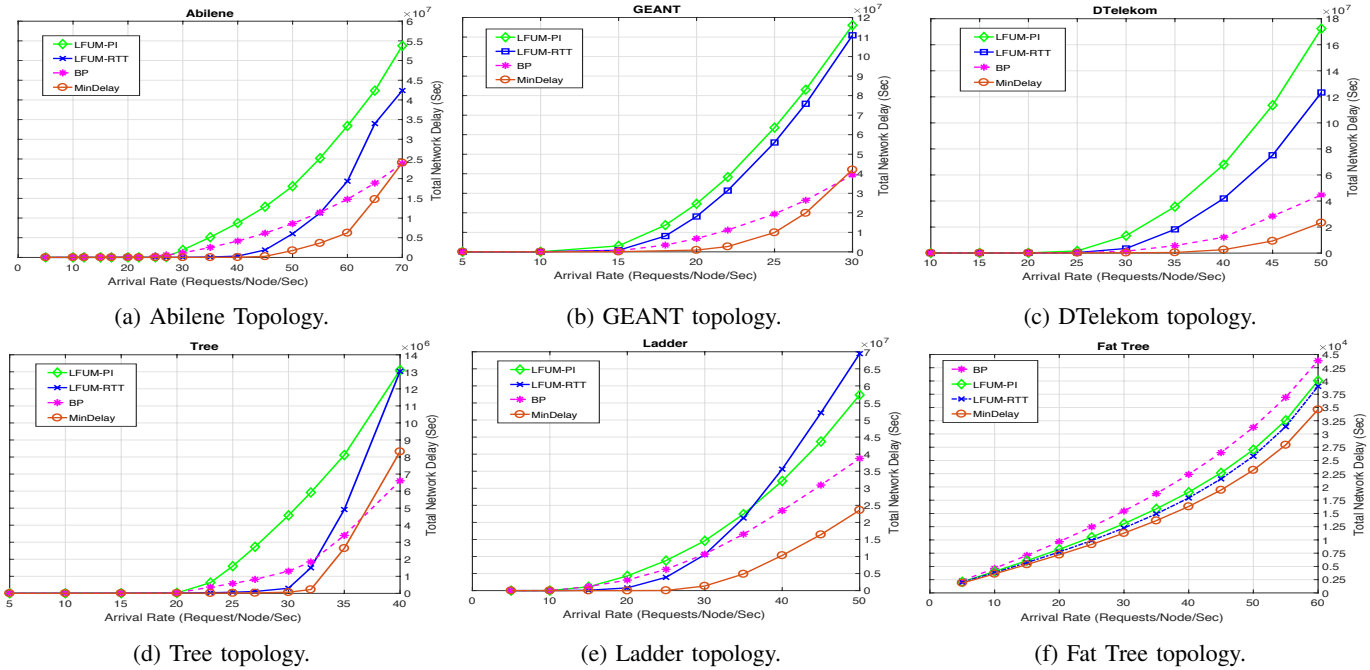
To reduce randomness in our results, we ran each simulation 10 times, each with a different seed number, and plotted the average performance of each scheme in Figure 2.

Figure 2 shows the total network delay in seconds versus the per-node arrival rate in request/seconds, for the above-mentioned topologies. As can be seen, in all the considered topologies, MinDelay has lower delay in the low to moderate arrival rate regions. In the higher arrival rate regions, BP's outperforms MinDelay in 3 of the tested topologies (Abilene, GEANT, and Tree),

As shown in [13], the BP performs well in high arrival rate regions since the VIP algorithm adaptively maximizes the throughput of Interest Packets, thereby maximizing the stability region of user demand rates satisfied by the network. When the network is operating well within the stability region, however, MinDelay typically has superior performance. Thus, the MinDelay and VIP algorithms complement each other in delivering superior delay performance across the entire range of request arrival rates.

VI. Conclusion

In this work, we established a new unified framework for minimizing congestion-dependent network cost in multi-hop caching networks by jointly choosing node-based forwarding and caching variables. Relaxing integer constraints on caching variables, we used a version of the conditional gradient algorithm to develop MinDelay, an adaptive and distributed joint forwarding and caching algorithm for the original mixed-integer optimization problem. The MinDelay algorithm ele-



(a) Abilene Topology. (b) GEANT topology. (c) DTelekom topology.

(d) Tree topology. (e) Ladder topology. (f) Fat Tree topology.

Fig. 2: Total network delay (sec) vs. request arrival rate (requests/node/sec).

gantly yields feasible routing variables and integer caching variables at each iteration, and can be implemented in a distributed manner with low complexity and overhead.

Simulation results show that while the VIP algorithm performs well in high request arrival rate regions, MinDelay has significantly better delay performance when the network is operating well within its stability region. Thus, the MinDelay and VIP algorithms complement each other in delivering superior delay performance across the entire range of request arrival rates.

REFERENCES

[1] G. Carofiglio, L. Mekinda, and L. Muscariello, "Lac: Introducing latency-aware caching in information-centric networks," in *2015 IEEE 40th Conference on Local Computer Networks (LCN)*, Oct 2015, pp. 422–425.

[2] D. Nguyen, K. Sugiyama, and A. Tagami, "Congestion price for cache management in information-centric networking," in *2015 IEEE Conference on Computer Communications Workshops*, Apr. 2015, pp. 287–292.

[3] G. Carofiglio, M. Gallo, L. Muscariello, and D. Perino, "Modeling data transfer in content-centric networking," in *Proceedings of the 23rd International Teletraffic Congress*, ser. ITC '11. International Teletraffic Congress, 2011, pp. 111–118.

[4] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache ?less for more? in information-centric networks," in *International Conference on Research in Networking*. Springer, 2012, pp. 27–40.

[5] M. Dehghan, L. Massoulie, D. Towsley, D. Menasche, and Y. C. Tay, "A utility optimization approach to network cache design," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, Apr. 2016, pp. 1–9.

[6] M. Badov, A. Seetharam, J. Kurose, V. Firoiu, and S. Nanda, "Congestion-aware caching and search in information-centric networks," in *Proceedings of the 1st international conference on Information-centric networking*. ACM, 2014, pp. 37–46.

[7] S. Ioannidis and E. Yeh, "Adaptive caching networks with optimality guarantees," in *Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science*. ACM, 2016, pp. 113–124.

[8] G. Carofiglio, M. Gallo, L. Muscariello, M. Papalini, and S. Wang, "Optimal multipath congestion control and request forwarding in information-centric networks," in *Network Protocols (ICNP), 2013 21st IEEE International Conference on*, Oct 2013, pp. 1–10.

[9] D. Posch, B. Rainer, and H. Hellwagner, "Saf: Stochastic adaptive forwarding in named data networking," *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 1089–1102, Apr. 2017.

[10] A. Detti, C. Pisa, and N. B. Melazzi, "Modeling multipath forwarding strategies in information centric networks," in *2015 IEEE Conference on Computer Communications Workshops*, Apr. 2015, pp. 324–329.

[11] M. Mahdian, S. Arianfar, J. Gibson, and D. Oran, "Mircc: Multipath-aware icn rate-based congestion control," in *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, ser. ACM-ICN '16. New York, NY, USA: ACM, 2016, pp. 1–10.

[12] C. Yi, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "Adaptive forwarding in named data networking," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 3, pp. 62–67, Jun. 2012.

[13] E. Yeh, T. Ho, Y. Cui, M. Burd, R. Liu, and D. Leong, "Vip: A framework for joint dynamic forwarding and caching in named data networks," in *Proceedings of the 1st ACM Conference on Information-Centric Networking*, ser. ACM-ICN '14. New York, NY, USA: ACM, 2014, pp. 117–126.

[14] R. Gallager, "A minimum delay routing algorithm using distributed computation," *Communications, IEEE Transactions on*, vol. 25, no. 1, pp. 73–85, Jan 1977.

[15] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of CoNEXT '09*. New York, NY, USA: ACM, 2009, pp. 1–12.

[16] D. Bertsekas and R. Gallager, *Data Networks (2Nd Ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1992.

[17] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Transactions on Information Theory*, vol. 59, no. 12, pp. 8402–8413, 2013.

[18] M. Mahdian and E. M. Yeh, "Mindelay: Low-latency forwarding and caching algorithms for information-centric networks," *CoRR*, vol. abs/1710.05130, 2017. [Online]. Available: http://arxiv.org/abs/1710.05130

[19] D. P. Bertsekas, *Nonlinear programming*. Athena scientific Belmont, 1999.