Optimizing Video Request Routing in Mobile Networks with Built-in Content Caching

Jun He and Wei Song, *Member, IEEE*

Abstract—Built-in content caching in mobile networks can help improve quality of service, reduce operation expenses, simplify inter-network cooperation, and thus is a promising approach for more efficient network architectures. In addition to the complexity of content placement, routing video requests remains a challenging issue. Two problems need to be addressed: (i) how to select servers to fulfill a request (*i.e., server selection*); (ii) how to route the requested video data (*i.e., traffic engineering*). In this work, we jointly formulate these two problems with two objectives considered, namely, minimizing maximum link utilization and minimizing total link cost. We propose fast algorithms to solve the problems with provable approximation guarantees. We then develop a hop-by-hop routing protocol, which implements the optimization solutions by generating a set of flow-splitting and routing decisions for each router/caching node. Simulation results show that our algorithms significantly outperform the Shortest-Path-based algorithm under various system settings, reducing up to 68% of maximum link utilization and more than 50% of link cost, and supporting over 60% more of traffic load.

Index Terms—Collaborative video caching, optimal request routing, source selection, traffic engineering, hop-by-hop forwarding, data aggregation, 3GPP LTE networks.

1 Introduction

1.1 Obliviousness to Mobile Video Delivery

Mobile video traffic is expected to increase by 16 folds in the next five years, accounting for over 66 percent of global data traffic in mobile networks by 2017 [1]. Ubiquitous access to video streaming services is becoming a reality for mobile devices. Despite the business opportunity it brings up, the exponential demand growth is challenging the foundation of current mobile networks, *e.g.*, the Long Term Evolution (LTE) systems of the Third Generation Partnership Project (3GPP), which are designed as extensions to the networks of Internet service providers (ISPs) and are completely oblivious to traffic details. Such obliviousness may involve the risk of a slowdown or even crash of mobile services, causing extra delay and eventually frustrating end users.

A use case of video delivery in an LTE system is depicted in Fig. 1. The core network of an LTE system, namely, Evolved Packet Core (EPC), is an all-IP core network, consisting of packet data network (PDN) gateways (P-GWs) and serving gateways (S-GWs). Content providers (CPs), such as NetFlix [2], often host a large amount of online videos in their content distribution networks (CDNs) or service clouds. The dashed line in Fig. 1 illustrates the flow path for a video request initiated at a mobile device to a CP and that of the response from the CP. As seen, the mobile network first encapsulates all received packets from the base station (namely eNodeB in LTE) into frames of GPRS (general packet radio service) tunneling protocol (GTP). Then, the GTP frames are relayed all the way through S-GWs in the core network till they reach P-GW at the edge of the mobile network. At this point, the GTP frames are decapsulated into original packets, which are further delivered to the ISP's network via the Internet exchange point (IXP). Likewise, the response passes the inverse path through the entire hierarchical infrastructure with encapsulation and decapsulation before reaching the target device. Apparently, such a universal approach of data delivery is inefficient, especially for video services, which require extremely high data rate, stable connection and low experienced latency. The long-distance relay path also increases the complexity of the cooperation between ISPs and CPs, raising network instability.

• *J. He and W. Song with the Faculty of Computer Engineering, University of New Brunswick, Fredericton, NB, Canada. E-mail: jhe2@unb.ca, wsong@unb.ca*

1.2 Built-in Caching and Request Routing

Transparent caching in EPC moves video clips close to mobile devices, so that it can effectively improve the quality of experience (QoE) of customers. From this conviction, we developed a collaborative caching framework in mobile core networks by exploiting disk capacity at S-GWs in [3], [4], where selected popular videos are stored therein. The hollow arrow line in Fig. 1 depicts the flow path of video services with built-in cache at S-GWs. Further enhancement is considered in [5] by enabling direct connections between S-GWs and remote source servers, *e.g.*, CDNs and cloud services, which are termed as *source service points*. The rationale behind is that mobile access to video services will contribute such a great fraction

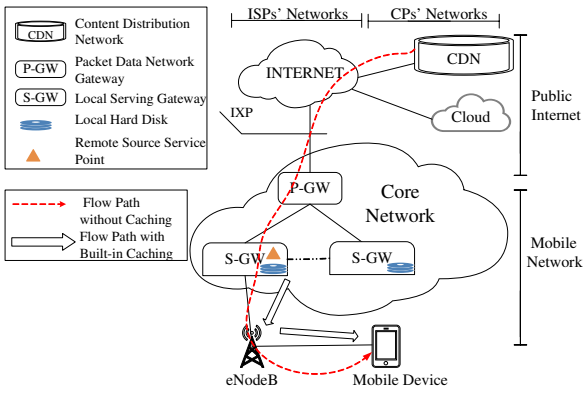


Figure 1: Use cases of video requests in an LTE system.

of requests to CPs that setting up managed source connections or even dedicated built-in mobile CDNs (MCDNs) will bring more benefits than costs. In [3], [4], [5], built-in storage is optimized and updated at a time scale of weeks. Therefore, the key problem becomes how to route "dynamic" video requests in core networks with built-in content caching.

Flow-level dynamics have a great impact on users' QoE in video streaming services [6], [7]. In order to respond to highly time-varying requests, the system needs to periodically optimize request routing, hence in-network flow, on a comparable time scale of request dynamics. Since S-GWs are interconnected in a flat topology, now with built-in caching, a video request received by an S-GW will be parsed and checked whether the requested video is available at its local cache. If so, the corresponding S-GW serves the request directly; otherwise, it will try to fetch the file from other S-GWs or through source service points. In order to route video requests that are not locally available (termed as *collaborative requests* henceforth), the system first needs to solve the *server selection* problem, *i.e.,* determining what servers (either serving nodes S-GWs or source service points) should be used to fetch the data. Then the system needs to solve the *traffic engineering* problem, in which routing decision is to be made for video streams so as to optimize network flow.

We note that in generic networks, source selection or CDN selection remains a difficult problem, due to unstructured network infrastructures and large-scale network components, such that existing systems, *e.g.,* NetFlix, employ only static configuration or simple switching algorithms [2]. In contrast, LTE systems are born with central management, *e.g.,* P-GWs. Such centralization is further enhanced in future software defined mobile networks (Mobile SDN) [8]. Moreover, a typical video file (chunk) supports playback of 10-300 seconds, allowing flow optimization in a fast scale.

1.3 Our Objectives and Contributions

In this work, we attempt to strategically route video requests in mobile core networks so as to optimize the distribution of network traffic. Specifically, two objectives are considered: (i) minimizing maximum link utilization and (ii) minimizing total link cost. The maximum link utilization is a critical factor to the stability, reliability and congestion of networks [9]. The total link cost usually reflects the operating expense (OPEX) of the network operator [10]. We formulate and solve two linear programming (LP) problems targeting the two objectives, respectively. In each LP problem, we jointly consider the server selection problem and the traffic engineering problem. In order to deploy our solutions in a real system, we then develop a complete routing protocol using the output from the solutions.

In summary, we have made three important contributions in this paper. Firstly, to the best of our knowledge, we are the first to address the video request routing problem with simultaneous server selection and traffic engineering in cache-enabled LTE networks, in which multiple sources and multiple paths for each request are considered. Two crucial goals in network flow are formulated and extensively studied, *i.e.,* minimizing maximum link utilization and minimizing total like cost.

Secondly, we propose fully polynomial-time approximation algorithms for the two formulated problems. For both problems, we can achieve an approximation guarantee of $1+\omega$ with time complexity proportional to ω^{-2}. Extensive simulations are carried out to study our algorithms. Numerical results validate the efficacy of the proposed solutions, and show that our algorithms outperform a Shortest-Path-based algorithm by up to 68% for the min-max link utilization and supports up to 60% more traffic load.

Thirdly, we further convert the proposed solutions to hop-by-hop forwarding routing decisions, which are implemented as a complete routing protocol at routers (possibly including some caching nodes) for collaborative requests. Different from conventional routing schemes, our protocol defines a set of flow-splitting rules. For every flow passing through a router, it specifies the fraction of the flow that will be forwarded to each outgoing link. The routing protocol is further improved with video stream aggregation. Such a routing protocol is compatible with most existing routers. More importantly, it is also an advanced design for the future openflow-based network [8], [11], [12], where a central controller optimizes overall network flows via the control plane.

The rest of this paper is organized as follows. Section 2 describes the system model and formulates the problems. Sections 3 and 4 introduce our solutions. Simulation studies are presented in Section 5. Section 6 discusses the hop-by-hop routing protocol.

Section 7 reviews related works. Finally, we make concluding remarks in Section 8.

2 System Model and Formulation

2.1 System Model

We target at the scenario depicted in Fig. 1, where a cache-enabled LTE system is swamped with video requests. User engagement with one request is usually in the order of 10 seconds [13]. We consider request routing problems at a similar time scale, while within each short time period, the request pattern is assumed to be fixed and visible to a central optimizer. A source service point is represented as a special serving node in the caching framework, which stores the entire video library and provides a promised link bandwidth to any serving node it is attached to.

Given the above system model, when a request for video k arrives at a serving node, the serving node first checks whether it has a copy of the video. If so, it will serve the request using its local copy. Otherwise, it will try to find other serving nodes or source service points located at some serving nodes, which contain video k (possibly through a directory service). Thus, the serving node receiving the request can establish connections to the selected servers and pass requested data to the video client.

In this work, we assume that the video content placement is given, *e.g.*, using the approaches in [14], [15], and source service points are already setup, using the approaches in [5]. Then, we need to optimize request routing in the considered time period. There are two problems to be solved: (i) how to select servers to fulfill each request (*i.e., server selection*); and (ii) how to route the requested video streams (*i.e., traffic engineering*). In the following, we develop joint formulations to consider these two problems simultaneously.

2.2 Formulation

We formulate the LTE network as a directed graph $G = (N, V, E)$, where N is the set of video clips ($n = |N|$), V is the set of caching nodes ($v = |V|$), E is the set of links ($m = |E|$), and each link e has a capacity $c(e)$. Here caching nodes include both serving nodes and source service points. For every pair of nodes $i, j \in V$, let $\mathcal{P}_{ij}$ be the set of paths from i to j, and $\mathcal{P} = \cup_{(i,j)}\mathcal{P}_{ij}$ be the union of all path sets. Moreover, let $\mathcal{P}_e$ be the set of all paths in $\mathcal{P}$ that use edge e for any $e \in E$. Let S_k be the set of serving nodes that contain video clip $k \in N$. A request (i, k) is called *collaborative* if video k is not stored in the local cache of node i and has to be retrieved from other nodes, which is further associated to a positive demand value d_i^k. Denote the set of collaborative requests by R with $r = |R|$. Further, $\mathcal{P}_i^k$ denotes the set of all available paths[1] to serve request $(i, k) \in R$, *i.e.*, $\mathcal{P}_i^k = \cup_{j \in S_k}\mathcal{P}_{ji}$. Table 1 summarizes the notations used in this paper.

Table 1: Basic notations.

Notation	Definition
$c(e)$	Capacity of edge e
d_i^k	Demand of any request for video k at node i
E	Set of in-network links
N	Set of video clips cached in system
$\mathcal{P}$	Set of in-network paths
$\mathcal{P}_e$	Set of paths using edge e
$\mathcal{P}_i^k$	Set of paths available to request (i, k)
$\mathcal{P}_{ij}$	Set of paths from node i to node j
R	Set of collaborative requests
R_i	Set of requests at node i
S_k	Set of nodes that store video k
V	Set of serving nodes
$x(P)$	Amount of flow routed on $P \in \mathcal{P}_i^k$ w.r.t. (i, k)

Now our problems is to determine the amount of flow routed on all available paths P, $x(P)$, in order to serve all collaborative requests. Apparently, this requires to simultaneously solve both the server selection problem and the traffic routing problem. We consider two objectives: (i) minimizing the maximum link utilization; and (ii) minimizing the total link cost. Here, we assume that a client's requested flow can be arbitrarily split among source nodes, *e.g.*, using the functionalities of domain name system (DNS) servers or hypertext transfer protocol (HTTP) proxies [16].

2.3 Goal 1: min-max link utilization

Our first goal is to minimize the maximum link utilization, which is formulated as an LP problem:

$$\begin{aligned} \min_{x(P)} \quad & \lambda \\ \text{s. t.} \quad & \sum_{P: e \in P} x(P) \le \lambda c(e), \ \forall e \in E \\ & \sum_{P \in \mathcal{P}_i^k} x(P) \ge d_i^k, \ \forall (i,k) \in R \\ & x(P) \ge 0, \ \forall P \in \mathcal{P} \end{aligned} \tag{1}$$

where the first constraint implies the link capacity limitation (scaled by the link utilization factor λ), and the second constraint satisfies the demand for video k at node i.

We comment that this formulation involves a great number of variables $x(P)$, which increase exponentially with the problem size. However, it can be converted into a polynomial-size edge-flow formulation (*e.g.*, see [17]), which is an LP problem. Therefore, polynomial-time optimal algorithms exist for problem

1. Available paths are defined as logical paths associated with the specific request. If two or more requests share one common physical path, it is considered repeatedly for each request, *i.e.,* there are no sharing paths.

(1). Nevertheless, the problem size can be extremely large because the number of video clips can be very large. Hence, existing LP techniques cannot be directly employed to solve the problem. Our goal here is to develop a fast approximation algorithm with bounded performance guarantee.

2.4 Goal 2: minimizing total link cost

Attaching flows to links will incur network cost, *e.g.,* latency and bandwidth expense. With a certain flow assignment, network cost is the total cost of all links. Hence, our second objective is to minimize the aggregate weighted flow on all edges, where the weight on an edge e, denoted by $b(e)$, can be viewed as the per-unit-flow cost on that edge. The problem is then formulated as

$$\begin{aligned} \min_{x(P)} \quad & \sum_{e\in E}\sum_{P:e\in P} b(e)x(P) \\ \text{s. t.} \quad & \sum_{P:e\in P} x(P) \le c(e),\ \forall e\in E \\ & \sum_{P\in\mathcal{P}_i^k} x(P) \ge d_i^k,\ \forall (i,k)\in R \\ & x(P)\ge 0,\ \forall P\in\mathcal{P}. \end{aligned} \tag{2}$$

It is worth mentioning that this formulation is quite general. In reality, many network vendors use the 95-percentile charging scheme [10], where the cost is a linear function of traffic volume. Since the above formation can minimize the network flow, optimizing the linear cost repeatedly will eventually reduce the monthly operating expense for network operators. Besides, the problem of minimizing the total end-to-end delay for all video clips can also be converted to this problem (*e.g.,* see [18]).

This problem can also be converted to an edge-flow formulation and solved optimally via LP techniques for small or middle problem size. Since the size of the problem we are dealing with is typically too large to be efficiently solved using LP techniques, a fast approximation solution is more desirable.

3 Algorithm for Min-max Link Utilization

To solve problem (1), we first look at the following problem:

$$\begin{aligned} \max_{y(P)} \quad & \pi \\ \text{s. t.} \quad & \sum_{P:e\in P} y(P) \le c(e),\quad \forall e\in E \\ & \sum_{P\in\mathcal{P}_i^k} y(P) \ge \pi d_i^k,\quad \forall (i,k)\in R \\ & y(P)\ge 0,\quad \forall P\in\mathcal{P}. \end{aligned} \tag{3}$$

It is easy to see that if we let $x(P)=y(P)/\pi, \lambda=1/\pi$, problems (3) and (1) are equivalent to each other. Therefore, we will solve problem (3) in the following. Note that problem (3) is similar to the maximum concurrent flow problem which was studied in [19], [20], [21]. There is one key difference though. In the maximum concurrent flow problem, every commodity has one source and one destination, while in problem (3), every request for video k at node i has one destination but possibly multiple sources (*i.e.,* the set S_k). Essentially, our problem can be viewed as a variant of the maximum concurrent flow problem with multiple sources and one destination for every commodity. If we reverse the source and destination, it can be viewed as a multicast version of the maximum concurrent flow problem, which, to the best of our knowledge, has not been studied in the literature.

To leverage the result in [19], [20], [21], one naive solution is to introduce a super source node for each commodity k and connect it to every node in S_k with a directional link of infinite capacity. Then the problem becomes an exact maximum concurrent flow problem. With this naive approach, the underlying network is significantly enlarged with n (n as the number of video clips) more nodes and at least $O(n)$ more edges. The algorithm complexity also increases correspondingly. Karakostas made an important contribution in [20] to reduce the complexity of the algorithms in [19], [21]. In particular, the algorithm complexity in [20] does not depend on the number of commodities (*i.e.,* the number of video clips n), but only on the number of edges in the network (*i.e.,* m). In the following, we further improve the algorithm in [20] to solve problem (3) while maintaining very low complexity.

3.1 Proposed Approximation Algorithm

In this section, we propose a $(1+\omega)$-approximation algorithm for problem (3) for any $\omega>0$. First, the dual problem of (3) is written as

$$\begin{aligned} \min_{l(e)} \quad & D(l) \triangleq \sum_{e\in E} c(e)l(e) \\ \text{s.t.} \quad & \sum_{e\in P} l(e) \ge z_i^k,\quad \forall (i,k)\in R, \forall P\in\mathcal{P}_i^k \\ & \sum_{(i,k)\in R} z_i^k d_i^k \ge 1 \\ & l(e), z_i^k \ge 0,\quad \forall (i,k)\in R, \forall e\in E. \end{aligned} \tag{4}$$

The dual problem introduces a length function $l(e)$ for each edge e and a variable z_i^k for each collaborative request $(i,k)\in R$. For a given length function l, z_i^k can be considered as the distance of the shortest path from the nearest node in S_k to node i under the length function l, denoted by

$$\psi_i^k(l) \triangleq \min_{j\in S_k} \psi_{ji}(l),\quad \forall (i,k)\in R, \tag{5}$$

where $\psi_{ji}(l)$ denotes the minimum distance from node j to node i under the length function l.

Define $\alpha(l)$ as the aggregate shipping cost for all requests, *i.e.*,

$$\alpha(l) \triangleq \sum_{(i,k)\in R} d_i^k \psi_i^k(l). \quad (6)$$

As observed by Garg and Könemann in [19], minimizing $D(l)$ under the dual constraints is equivalent to finding an assignment of lengths l to edges, such that $D(l)/\alpha(l)$ is minimized. Henceforth, we define the optimal dual objective (which is equal to the optimal primal objective value) as

$$\beta \triangleq \min_{l} D(l)/\alpha(l). \quad (7)$$

Algorithm 1: Algorithm for problem (3).

Input: Network graph $G = (N, V, E)$, edge capacities $c(e)$, video sources set S_k, collaborative requests set R, request demands $\{d_j^k\}$, accuracy ϵ
Output: Primal solution $\mathbf{y}$ and π

Initialize $l(e) \leftarrow \delta/c(e), \forall e, y(P) \leftarrow 0, \forall P$
while $D(l) < 1$ **do**
  for $j = 1$ to v **do**
    For all requests $(j,k) \in R_j$, initialize $\tilde{d}_j^k = d_j^k$.
    while $D(l) < 1$ and $\tilde{d}_j^k > 0$ for some k **do**
      $K \leftarrow \{k | (j,k) \in R_j, \tilde{d}_j^k > 0\}$
      $P_j^k \leftarrow$ shortest path in $\mathcal{P}_j^k$ using length l
      $\rho \leftarrow \max\left\{1, \max_{e\in\cup_{k\in K} P_j^k} \frac{\sum_{k:e\in P_j^k} \tilde{d}_j^k}{c(e)}\right\}$
      for $k \in K$ **do**
        $f_j^k \leftarrow \tilde{d}_j^k/\rho$
        $\tilde{d}_j^k \leftarrow \tilde{d}_j^k - f_j^k$
        $y(P_j^k) \leftarrow y(P_j^k) + f_j^k$
      end for
      $l(e) \leftarrow l(e)\left(1 + \epsilon\frac{\sum_{k:e\in P_j^k} f_j^k}{c(e)}\right), \forall e \in \cup_{k\in K} P_j^k$
    end while
  end for
end while
$y(P) \leftarrow y(P)/\log_{1+\epsilon}\frac{1+\epsilon}{\delta}, \forall P$
$\pi \leftarrow \min_{(j,k)\in R} \frac{\sum_{P\in\mathcal{P}_j^k} y(P)}{d_j^k}$

We first assume that $\beta \geq 1$ in algorithm presentation and analysis, and then remove the assumption in Section 3.3. The main procedure of our algorithm is outlined in Alg. 1. The algorithm proceeds in phases and each phase consists of v iterations. Initially, we set the length of an edge e to be $l(e) = \delta/c(e)$, where δ is a parameter to be defined later. In each phase, we consider all the requests at node j in iteration j and satisfy all the demands of requested videos through a series of steps. For presentation clarity, we add subscripts to indicate phase i, iteration j and step s. Let $\tilde{d}_{i,j,s}^k$ be the demand for video k that has not been fulfilled at step s and it is set to d_j^k at step 0. At step s, we only consider the requests that have not been completely fulfilled in the current iteration (*i.e.*, with $\tilde{d}_{i,j,s}^k > 0$). The set of unsatisfied commodities (video requests) of node j, denoted by K, is updated at the beginning of each step. We first compute the shortest path tree T_j rooted at node j using the length function l. Note that it is independent of the requested video k, and therefore we only need to compute it once for all video requests. Then for each requested video k, we find the shortest path in T_j and route $f_{i,j,s}^k = \tilde{d}_{i,j,s-1}^k/\rho$ unit of flow along the shortest path for video k, where ρ, determined in step 8 of Alg. 1, is a scaling factor such that the aggregate flow on any edge does not exceed its capacity $c(e)$. Then we update $\tilde{d}_{i,j,s}^k = \tilde{d}_{i,j,s-1}^k - f_{i,j,s}^k$. Let $l_{i,j,s}$ denote the length function at step s, iteration j, of the ith phase. Then, the length l on every edge e is updated as follows:

$$\begin{aligned} l_{i,j,s}(e) &= l_{i,j,s-1}(e)(1 + \epsilon \cdot \frac{\text{sum of new flows on } e}{c(e)}) \\ &= l_{i,j,s-1}(e)(1 + \epsilon \cdot \frac{\sum_{k:e\in P_j^k} f_{i,j,s}^k}{c(e)}), \end{aligned} \quad (8)$$

where $0 < \epsilon < 1$ is to be defined later.

Note that each iteration j (possibly except the last one) does not complete until all the demands/requests at node j are fully routed. At the end of each iteration, we can compute the dual objective value $D(l)$. As soon as $D(l) \geq 1$, the algorithm terminates and produces a flow routing solution $\{\mathbf{y}(P)\}$, which is infeasible for the primal problem in (3). However, applying similar analysis to that in [20], we can easily prove that a feasible solution to the primal problem can be obtained by scaling down $\mathbf{y}$ by a factor of $\log_{1+\epsilon}\frac{1+\epsilon}{\delta}$.

3.2 Correctness and Approximation

The correctness of Alg. 1 can be proved by comparing it to the naive approach of adding a super source node for each video and applying the solutions to the maximum concurrent multi-commodity flow problem (*e.g.*, in [19], [20]). Assume that we introduce super source nodes for all requested videos and connect the super nodes with other nodes with links of infinite capacity. Then problem (3) reduces to the maximum concurrent flow problem discussed in [19]. For each request (i,k), we need to find the minimum distance from the super source node of commodity k to node i under the length function l. Since the outgoing edges of any super source have infinite capacity, the length of all these edges is always 0, according to the length update equations in Alg. 1. Accordingly, finding the shortest path from the super source for video k to node i is equivalent to computing the path with the minimum distance in the shortest path tree T_k rooted at node k. Therefore, Alg. 1 finds the same solution

as the naive algorithm that adds a super source node for every video k.

We next analyze the approximation factor of Alg. 1. At step s of iteration j of the ith phase, the dual objective value $D(l)$ is increased by

$$\begin{aligned}\triangle D(l_{i,j,s}) &\overset{(8)}{=} \epsilon \sum_{e\in\cup_{k\in K}P_j^k} l_{i,j,s-1}(e) \sum_{k:e\in P_j^k} f_{i,j,s}^k \\ &= \epsilon \sum_{k\in K} f_{i,j,s}^k \sum_{e\in P_j^k} l_{i,j,s-1}(e) \\ &\overset{(5)}{=} \epsilon \sum_{k\in K} f_{i,j,s}^k \psi_j^k(l_{i,j,s-1}) \qquad (9)\end{aligned}$$

where P_j^k is the shortest path to reach video k at node j and $\psi_j^k(l_{i,j,s-1})$ is the corresponding shortest distance, which is updated using the length function in the previous step. Since $\psi_j^k(l)$ is monotonically increasing through the steps, if κ_j denotes the number of steps in iteration j of the ith phase, we have $\psi_j^k(l_{i,j,s}) \le \psi_j^k(l_{i,j,\kappa_j}), \forall s \le \kappa_j$. Hence, for the entire iteration, the dual objective increase is bounded by

$$\begin{aligned}\sum_{s=1}^{\kappa_j} \triangle D(l_{i,j,s}) &\overset{(9)}{=} \epsilon \sum_{s=1}^{\kappa_j} \sum_{k\in K} f_{i,j,s}^k \psi_j^k(l_{i,j,s-1}) \\ &\le \epsilon \sum_{s=1}^{\kappa_j} \sum_{k\in K} f_{i,j,s}^k \psi_j^k(l_{i,j,\kappa_j}) \\ &= \epsilon \sum_{k\in K} d_j^k \psi_j^k(l_{i,j,\kappa_j}).\end{aligned}$$

For notational convenience, let $l_{i,1,0}$ denote the length function after the last step of the last iteration in phase $i-1$, $D(i)$ denote the corresponding dual objective value $D(l_{i,1,0})$, and $\alpha(i)$ denote $\alpha(l_{i,1,0})$. We further obtain the increase bound of the dual objective during the ith phase as

$$\begin{aligned}\sum_{j,s} \triangle D(l_{i,j,s}) &\le \epsilon \sum_{j} \sum_{k\in K} d_j^k \psi_j^k(l_{i,j,\kappa_j}) \\ &\le \epsilon \sum_{j} \sum_{k\in K} d_j^k \psi_j^k(l_{i+1,1,0}) \\ &= \epsilon\alpha(l_{i+1,1,0}).\end{aligned}$$

Therefore, we have the following recursive relationship:

$$\begin{aligned} D(l_{i+1,1,0}) &\le D(l_{i,1,0}) + \epsilon\alpha(l_{i+1,1,0}) \\ \Longleftrightarrow D(i+1) &\le D(i) + \epsilon\alpha(i+1). \qquad (10)\end{aligned}$$

According to (7), the fact that $\beta \le D(i)/\alpha(i)$ implies that

$$\frac{D(i+1)}{D(i)} \le \frac{1}{1-\epsilon/\beta}, \quad \forall i \ge 0, \epsilon < 1.$$

Together with the initial setting $D(0) = m\delta$ (see line 1 of Alg. 1) and the assumption $\beta \ge 1$, we finally have the non-recursive inequality for $D(i)$, given by

$$\begin{aligned} D(i) &\le \frac{m\delta}{(1-\epsilon/\beta)^i} \\ &= \frac{m\delta\beta}{\beta-\epsilon}(1+\frac{\epsilon}{\beta-\epsilon})^{(i-1)} \\ &\le \frac{m\delta\beta}{\beta-\epsilon} e^{\epsilon(i-1)/(\beta-\epsilon)} \\ &\le \frac{m\delta}{1-\epsilon} e^{\epsilon(i-1)/(\beta-\epsilon)}.\end{aligned}$$

If the algorithm terminates after q phases, which implies that $D(q) \ge 1$, we have

$$\beta \le \frac{\epsilon(q-1)}{(1-\epsilon)\ln\frac{1-\epsilon}{m\delta}}.$$

Therefore, similar to [19], [20], we can claim the following conclusions. We omit the proofs here because all the remaining details are the same as those in [19], [20].

Theorem 1. *If $\beta \ge 1$, the number of phases in Alg. 1 is bounded by $\lceil \beta \log_{1+\epsilon} \frac{1+\epsilon}{\delta} \rceil$.*

Theorem 2. *Provided $\beta \ge 1$, if π denotes the objective value of problem (3) found by Alg. 1, we have $\pi > \frac{q-1}{\log_{1+\epsilon}\frac{1+\epsilon}{\delta}}$, where*

$$\delta = \frac{1}{(1+\epsilon)^{\frac{1-\epsilon}{\epsilon}}} \cdot \left(\frac{1-\epsilon}{m}\right)^{1/\epsilon}.$$

The obtained objective value is related to the optimal value β according to the inequality: $\frac{\beta}{\pi} < (1-\epsilon)^{-3}$.

Therefore, if we choose ϵ such that

$$(1-\epsilon)^{-3} = 1+\omega \qquad (11)$$

the approximation ratio $\frac{\beta}{\pi}$ is less than $(1+\omega)$ for any $\omega > 0$. Here we note that by this setting, ω and ϵ are in the same order.

3.3 The $\beta < 1$ case

In this section, we will show that in any case with $\beta < 1$, we can always scale down the demands of video requests to produce a corresponding instance with $\beta \ge 1$. To see this, we look at the dual problem (4) and its optimal objective in Eq. (7). An instant observation suggests that, in an instance of problem (4), if we scale down all demands by a factor γ, its optimal value β will increase by a factor $1/\gamma$. Then, the remaining procedure is to derive a lower bound and an upper bound of β for a given instance.

We first consider a special case where we only route the request for video k at node i, and denote the optimal fraction of demands (*i.e.*, maximum flow divided by d_i^k) by γ_i^{k*}. Using a modified version of Dijkstra's algorithm introduced in [21], we can obtain an m-approximation γ_i^k of γ_i^{k*}. In other words, $\gamma_i^k \ge \frac{1}{m}\gamma_i^{k*}$. Let $\bar{\gamma} = \min_{i\in V, k\in R_i} \gamma_i^k$. It is not hard to see that the optimal objective $\beta \le \min_{i\in V,k\in R_i} \gamma_i^{k*} \le m\bar{\gamma}$. On the other hand, it is feasible to route γ_i^k/r fraction of demand simultaneously for every video k at every node i. Hence, $\bar{\gamma}/r \le \beta \le m\bar{\gamma}$. Therefore, we can

develop an instance that ensures $\beta \geq 1$ by multiplying the initial demands by $\bar{\gamma}/r$.

However, the procedure above makes β as large as mr. Recalling Theorem 1, we observe that a large β results in more phases. We apply Garg's techniques in [19] and define $Q \triangleq 2\lceil \frac{1}{\epsilon}\log_{1+\epsilon}\frac{m}{1-\epsilon}\rceil$. If the procedure does not stop within Q phases, we know that $\beta \geq 2$. Thus, we proceed to a reduction scheme by doubling the demands of all commodities so that β is halved. After at most $\log(mr)$ times of reduction, we will reach $1 \leq \beta \leq 2$. Accordingly, the total number of phases required for the reduction is at most $\log(mr) \cdot Q$.

To further reduce the possible phases, we adopt the technique in [22], and first compute a 2-approximation to β which requires at most $O(\log(mr)\log m)$ phases[2] and returns $\bar{\beta}$, $\beta \leq \bar{\beta} \leq 2\beta$. Now we create a new instance by multiplying all the demands by $\bar{\beta}/2$. As a result, the new instance has $1 \leq \beta \leq 2$, and our procedure terminates within Q phases for it.

For a given small value $\epsilon > 0$, we can further rewrite $Q = O(\epsilon^{-2}\log m)$. Thus, the total number of phases for the original problem is $O((\epsilon^{-2} + \log(rm))\log m)$.

3.4 Running Time

The number of iterations in each phase is bounded by the number of serving nodes (v). In an iteration, at every step except the last one, the length of at least one edge is increased by a fraction of at least $(1+\epsilon)$. The length of each edge $e \in E$ is initialized to be $\delta/c(e)$, and it grows up to $(1+\epsilon)/c(e)$ when the procedure terminates. Thus, the number of steps for the $(1+\epsilon)$-length increase is at most $m\log_{1+\epsilon}\frac{1+\epsilon}{\delta} = O(\epsilon^{-2}m\log m)$. Together with the total number of iterations (incurred by the last step of each iteration), the total number of required steps is at most $O(v(\epsilon^{-2}+\log(rm))\log m) + O(\epsilon^{-2}m\log m) = O(\epsilon^{-2}m\log m + v\log m\log r)$ for a $(1+\omega)$-approximation solution to problem (1), where ω and ϵ are chosen as in Eq. (11).

Inside each step, if we consider a reversed graph G' of G, where the directions of all edges of G are reversed by using Fibonacci heaps, the computation of a shortest path tree takes $O(v\log v + m)$ time (one run of Dijkstra's algorithm on G'). Other computations take at most $O(r_{\max}+m)$ time, where $r_{\max} = \max_{i\in V}|R_i|$. We further denote the time cost for each step by $T_s = O(v\log v + m + r_{max})$ Therefore, our scheme computes a $(1+\omega)$-approximation solution to problem (1) in $\tilde{O}((\omega^{-2}+\log r)m\cdot T_s)$ time, for any $\omega > 0$, where $\tilde{O}(f) = O(f\cdot\log^{O(1)} m)$ (note that ω and ϵ are in the same order).

2. Here, we run Alg. 1 together with the above scaling procedure for the approximation ratio of $\omega = 1$. Thus, $Q = O(\log m)$.

4 Algorithm for Minimum Total Link Cost

To solve problem (2), we introduce a cost bound B and consider the following problem:

$$\begin{aligned}
\max_{x(P)} \quad & \pi \\
\text{s. t.} \quad & \sum_{P\in\mathcal{P}_i^k} x(P) \geq \pi d_i^k, \quad \forall (i,k)\in R \\
& \sum_{P:e\in P} x(P) \leq c(e), \quad \forall e \in E \\
& \sum_{e\in E}\sum_{P:e\in P} b(e)x(P) \leq B \\
& x(P) \geq 0, \quad \forall P \in \mathcal{P}.
\end{aligned} \tag{12}$$

Let the optimal solution to problem (12) be π^*. If $\pi^* \geq 1$, it is apparent that the minimum link cost of problem (2) is no more than B. Otherwise, the optimal solution to problem (2) is larger than B. Therefore, we can use binary search to find the optimal solution to problem (2) given that we can solve problem (12). As a result, the key to solving problem (2) is problem (12), which will be solved next.

Problem (12) is similar to the minimum cost multicommodity flow problem, studied by Garg and Könemann in [19]. The main difference is that, the source for video k is a set S_k here, whereas the source for each commodity is a single node in [19]. Again, although we can add a super source node for each video k and then leverage the algorithm in [19], [20], [21] to solve the modified problem, the complexity of the solution is much increased due to the inflated problem size. In the following, we adapt the algorithm in [20] to find a low-complexity approximation solution to problem (12).

The dual of problem (12) is defined as

$$\begin{aligned}
\min_{l(e),\phi} \quad & D(l,\phi) \triangleq \sum_{e\in E} c(e)l(e) + B\phi \\
\text{s.t.} \quad & \sum_{P:e\in P} l(e) + \phi b(e) \geq z_i^k, \quad \forall (i,k)\in R, \forall P \in \mathcal{P}_i^k \\
& \sum_{(i,k)\in R} z_i^k d_i^k \geq 1 \\
& \phi, l(e), z_i^k \geq 0, \quad \forall (i,k)\in R, \forall e \in E
\end{aligned} \tag{13}$$

where ϕ is the dual variable for the cost constraint.

We next modify Alg. 1 to solve this problem. In the algorithm, the initial ϕ is δ/B. Instead of using the length function l, we use a new length function $l + b\cdot\phi$ to find the available path of minimum cost. The flow sent in each step should be scaled down, so that the total cost does not exceed the link capacity or the budget B. That is, line 8 in Alg. 1 is changed to

$$\rho \leftarrow \max\left\{1, \max_{e\in\cup_{k\in K}P_j^k} \frac{\sum_{k:e\in P_j^k} f_j^k}{c(e)}, \frac{\sum_{k\in K}\sum_{e\in P_j^k} f_j^k b(e)}{B}\right\}.$$

Then after each step, the length function for each edge is updated similarly but ϕ is updated in the following

manner:

$$\phi \leftarrow \phi \left(1 + \epsilon \cdot \frac{\sum_{k \in K} \sum_{e \in P_j^k} f_j^k b(e)}{B}\right).$$

The analysis of the algorithm is similar to that for the min-max link utilization problem in Section 3. Therefore, we can find $(1+\omega)$-approximation solution to problem (12) in $\tilde{O}((\omega^{-2} + \log r)mT_s)$ time, for any $\omega > 0$. Using binary search, we can find a solution to problem (2) in $\tilde{O}((\omega^{-2} + \log r)mT_s \log M)$ time, where M is the maximum total link cost.

5 Performance Evaluation

5.1 Setup

In this section, we evaluate the proposed algorithms with computer simulations. Basic video caching environment is pre-set using the schemes in [5] for both content placement and setup of source service points. Since our algorithms are independent of specific network topologies or distributions of video clips, we use a network model as depicted in Fig. 2, where any two nodes can reach each other through multiple paths. The topology is a combination of a star network and a ring network. Here, $v-1$ nodes are connected in a ring, each of which has a bi-directional link to the central node 1. Two source service points are attached to the network serving nodes 1 and $v-1$ via bi-directional links. All links are bi-directional and each direction has a capacity of 1 Gbps.

There are $200,000$ video clips stored among the serving nodes, where the number of source nodes for each clip is uniformly generated within $1 \sim U$. Different upper bounds U are considered in the experiments. The demand of each video is uniformly generated from 128 kbps to 1 Mbps. The unit cost $b(e)$ of each link $e \in E$ is randomly generated within $[\sqrt{5}, 5^3]$ in an exponential fashion as [23], *i.e.*, it is a value with base 5 and exponent uniformly distributed between 0.5 and 3. Every caching node may serve as a router for required traffic. For each test case, we randomly generate collaborative requests at each node with certain traffic intensity. The *traffic intensity* is represented by the average number of collaborative requests initiated by each node.

As suggested in most existing works, *e.g.*, [14], [24], for request routing, we also consider an alternative scheme named "Shortest Path," for comparison, which serves each collaborative request by the nearest available source through the shortest path.

5.2 Minimization of Maximum Link Utilization

In Section 3, we have shown that Alg. 1 produces a $(1+\omega)$-approximation solution to the problem of minimization of maximum link utilization for any $\omega > 0$. Therefore, if the experimental results indeed show that the maximum link utilization λ obtained by Alg. 1 with parameter ω is the minimum, we can conclude that the lower bound of the optimal solution is $\frac{\lambda}{(1+\omega)}$. Fig. 3 shows the approximate solutions found by Alg. 1 with different approximation parameters ω. The traffic intensity is 1600 for the three scenarios with the number of caching nodes $v = 30, 50, 70$, respectively. We plot the lower bound along with the maximum link utilization found by Alg. 1. From Fig. 3, we conclude that an arbitrarily near-optimal solution can be found by Alg. 1 with a proper ω.

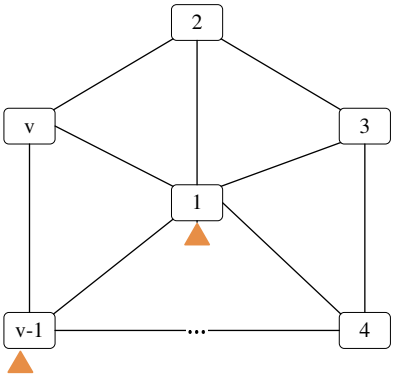


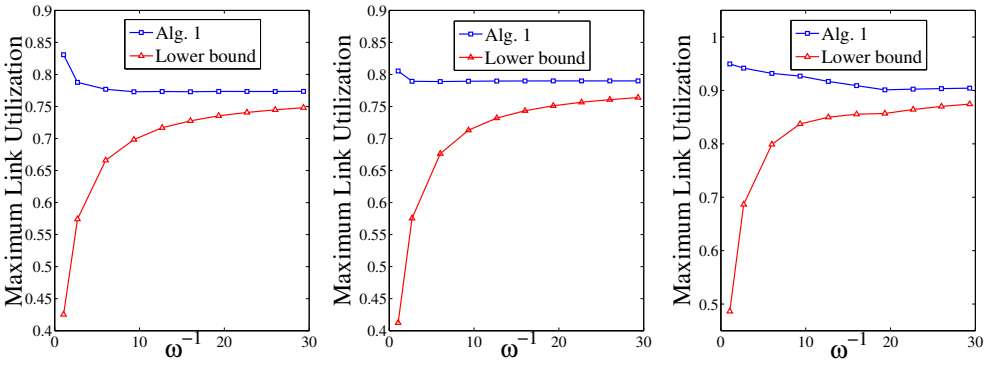
Figure 2: Network model.

However, as discussed in Section 3, a better solution requires more phases in the algorithm. We record the total number of phases for each instance, and show the relationship between the number of required phases and the approximation parameter ω in Fig. 4(a). The three quadratic curves in Fig. 4(a) also confirm the conclusion that the total number of phases in Alg. 1 is a quadratic function of ω^{-1} (see Section 3.3). As expected, Fig. 4(b) shows that the number of phases decreases as the traffic intensity grows. According to Theorem 1, as the traffic intensity increases, the min-max link utilization decreases inversely and so does the required number of phases.

In the three cases with the number of caching nodes $v = 30, 50, 70$, Figs. 5(a), 5(b), and 5(c) on the next page show the performance of our algorithm under different traffic intensities. The number of sources for each video is generated from $1 \sim 8$, ie, $U = 8$. Compared to the Shortest Path approach, our algorithm achieves much lower min-max link utilization. If we further compare the over-congestion points of the two schemes, we find that our algorithm supports up to 60% more traffic load than the Shortest Path approach.

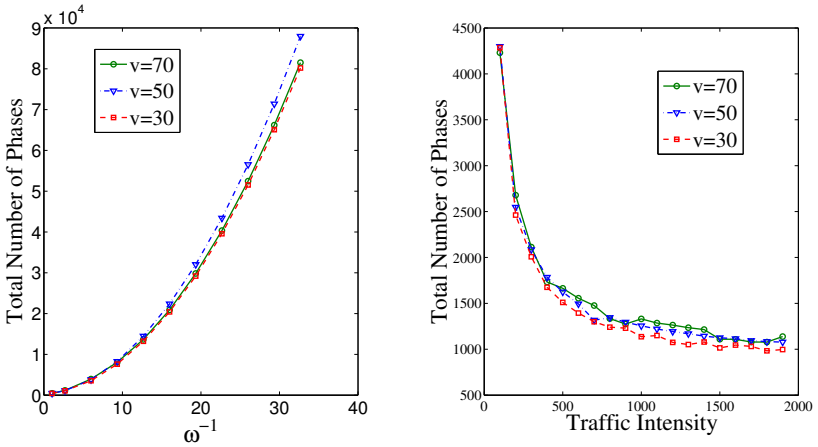
Figs. 5(d), 5(e), and 5(b) plot the min-max link utilization with $U = 3, 5, 8$, respectively. Recall that U is the parameter controlling the maximum number of sources for each video clip. We can see that as the number of video sources increases, the gap between our algorithm and the Shortest Path algorithm increases. This is because more available sources give more space for performance optimization.

Table 2 shows the ratio of the maximum link utilization achieved by our algorithm to that of the Shortest



(a) $v = 30$. (b) $v = 50$. (c) $v = 70$.

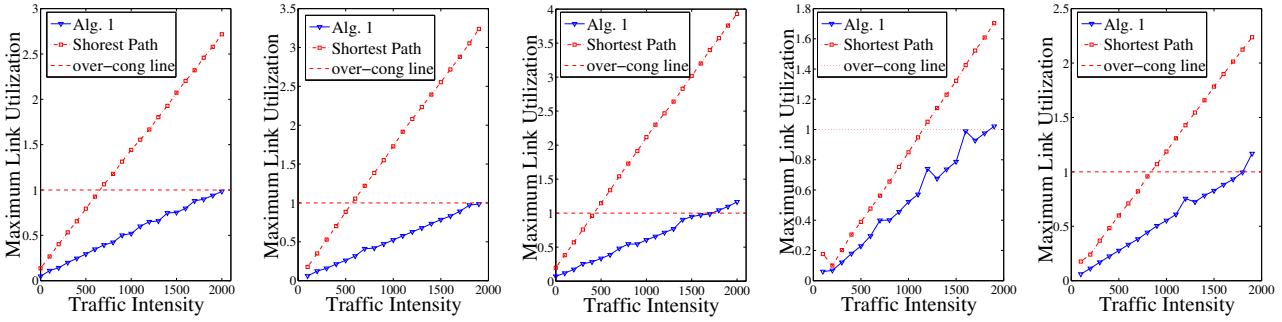
Figure 3: Maximum link utilization with traffic intensity 1600 and different approximation ratio ω.



(a) Number of required phases vs. approximation ratios.

(b) Number of required phases vs. traffic intensities.

Figure 4: Number of required phases of Alg. 1.



(a) $v = 30, U = 8$. (b) $v = 50, U = 8$. (c) $v = 70, U = 8$. (d) $v = 50, U = 3$. (e) $v = 50, U = 5$.

Figure 5: Comparison of Alg. 1 and Shortest Path scheme for minimizing maximum link utilization.

Path approach (the third column) under different system configurations for v and U. We can see that our algorithm can reduce the maximum link utilization by up to 68% compared to the Shortest Path algorithm.

Table 2: Ratio of performance achieved by our algorithms to that of Shortest Path approach.

v	U	Ratio of max link utilization	Ratio of total link cost
50	3	59%	50%
50	5	45%	45%
50	8	32%	44%
30	8	37%	48%
70	8	32%	49%

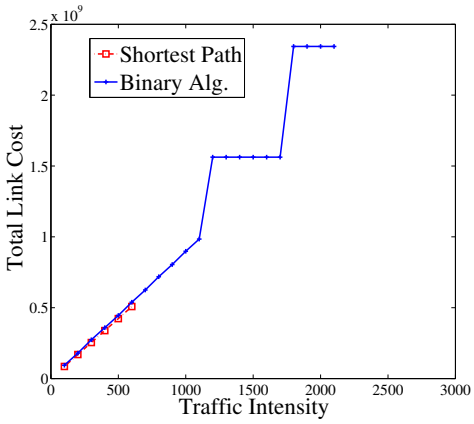


Figure 6: Minimum total link cost of Binary Algo. and Shortest Path Scheme vs. traffic intensities with $v = 50$, $U = 8$, $b(e) = 5^{1.75}, \forall e \in E$.

5.3 Minimization of Total Link Cost

For ease of presentation, we refer to our proposed algorithm for the problem of minimizing the total link cost in Section 4 as "Binary Alg.". In the following simulations, $\omega = 0.1$ is used to solve each subproblem of (12), and the binary search terminates when the gap between the lower bound and upper bound is within 1% of a feasible solution. We stop calculating the cost when no feasible routing can be developed, *i.e.*, at least one link is over-congested.

Fig. 6 compares the total link cost of Binary Alg. to that of the Shortest Path approach with the unit link cost function $b(e) = 5^{1.75}, \forall e \in E$. As seen, when the traffic intensity is below 700, the Shortest Path approach finds the optimal solution. In contrast, when the traffic intensity exceeds 700, the solution found by Shortest Path is infeasible, since it does not consider the constraints implied by the link capacity limits. These observations are intuitive since the total link cost relies on the unit cost function $b(e)$. Obviously, if the unit cost $b(e)$ is the same for all $e \in E$, the Shortest Path approach can attain the minimum total link cost except that it may violate the link capacity constraints.

To evaluate the performance of Binary Alg. in networks of different sizes, we simulate scenarios with different v for the number of caching nodes and different U for the maximum number of sources of each video clip. Figs. 7(a), 7(b) and 7(c) on the next page compare the resulting link cost of the two approaches with $v = 30, 50, 70$ and $U = 8$. Figs. 7(d), 7(e), and 7(b) evaluate the performance with $v = 50$ and $U = 3, 5, 8$, respectively. Again, Fig. 7 shows that our scheme produces better results with more available sources for each video clip. Similar to the observations in Section 5.2, our scheme produces much lower total link cost, and supports over 60% more traffic load than the Shortest Path approach.

The improvement of our algorithm is also shown in Table 2 by the ratio of the total link cost obtained using our scheme to that of the Shortest Path approach (the fourth column). Compared to the Shortest Path approach, the proposed Binary Alg. saves around 50% of link cost for routing the same set of video requests.

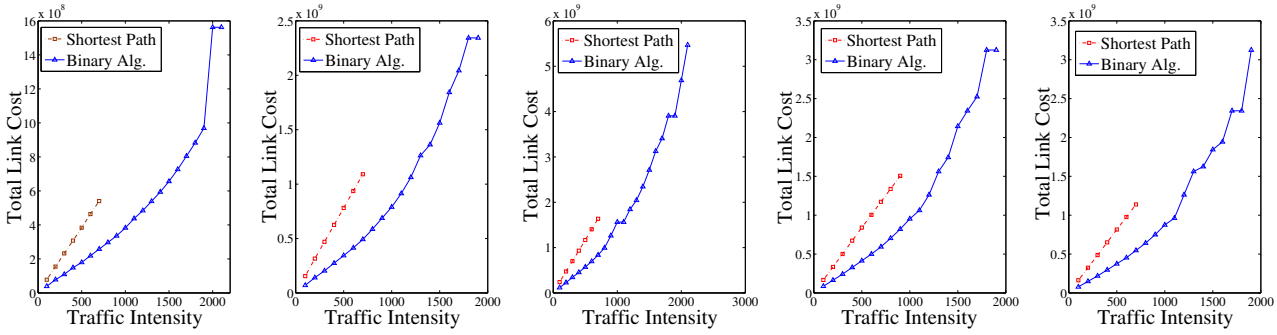
6 Routing Protocol: Implementation of Proposed Algorithms

6.1 A Hop-by-Hop Forwarding Design

We first review the two basic problems outlined at the beginning of the paper, *i.e.*, the server selection problem and the traffic engineering problem. For an instance of the request routing problem with specific optimization purposes, the algorithms proposed in Section 3 and Section 4 output a set of path-flow values $\{x(P)\}$. Accordingly, each path P w.r.t. request (i, k) is identified by a source node and a destination node. Together with the flow amount $x(P)$, the output of the proposed algorithms are sufficient to address the server selection problem. For instance, in a 3GPP LTE system, the corresponding video sources can be notified accordingly of the required flows via system signalling in the control plane.

On the other hand, the traffic engineering problem remains implicit, *i.e.*, how traffic should be routed on each router and each forwarding node. Based on similar consideration, in [20], Karakostas extended the path-flow algorithm to an *explicit* version, which in turns outputs the flow amount for every video clip (or commodity therein) on each edge with extra time $\tilde{O}(rv)$. However, it relies heavily on the basic assumption that the number of commodities is no more than the number of source-destination pairs. Otherwise, the commodities with the same source-destination pair are merged to one commodity prior to any further operations, so that all commodities are identified by corresponding source-destination pairs.

However, this assumption does not hold in our case, since here we may have a large number of video clips and each video clip may have a set of



(a) $v = 30, U = 8$. (b) $v = 50, U = 8$. (c) $v = 70, U = 8$. (d) $v = 50, U = 3$. (e) $v = 50, U = 5$.

Figure 7: Comparison of Binary Algo. and Shortest Path for minimizing total link cost.

source nodes. The number of source sets can be an exponential function of the number of source nodes. Therefore, the explicit algorithm in [20] cannot be applied. Another problem with the explicit algorithm in [20] is that the routers therein still need information of both source address and target address from any incoming flow so as to determine the routing. Nevertheless, even though both addresses are empirically accessible to routers, we note that most routers in practice technically only have chips for reading destination addresses of incoming flows [25], which further limits the application and development of similar explicit algorithms. Besides, we note that decoding data packets at intermediate nodes or to establish "tunnels" between peers for routing purposes will introduce unsustainable overhead to the system.

In the following, we design a routing protocol that is independent of any individual video request. Rather, the routing decision at each router only depends on the destination of the incoming traffic. Based on the destination address, an incoming flow is (possibly) split and forwarded to next hops according to the routing table, which reflects its routing policies as in typical networks. Hence, the proposed routing protocol uses a hop-by-hop forwarding design. We comment that our protocol is compatible with most existing routers and can also be potentially implemented in openflow-based routers [11].

We create a flow-splitting database and distribute a flow-splitting table to each node. A tuple in the table contains the receiving node, destination address, next-hop neighboring node, and flow fraction to the outgoing node. Here, a node can be either a serving node or an intermediate router. Fig. 8 shows an example of the flow splitting table used in our protocol. Node $\#0$ has two incoming flows $f1$ from 10.1 and $f2$ from 10.2 with the same destination address 10.5. When the flows $f1$ and $f2$ pass into node $\#0$, the node reads and finds the common destination address 10.5. Then, node $\#0$ looks up its flow-splitting table with the key 10.5 to obtain the splitting fractions for next-hop nodes. After that, it combines the two flows, splits

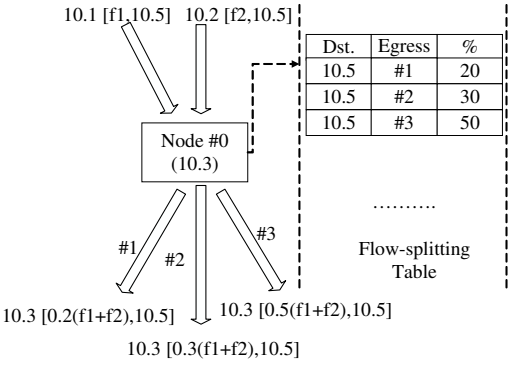


Figure 8: A routing example based on flow-splitting table.

and forwards the aggregated flow to respective links, *i.e.*, $\#1$, $\#2$, and $\#3$, according to the flow-splitting fractions.

6.2 Building Flow-Splitting Database

We now show how to build up the flow-splitting database. Each routing and splitting decision is expressed as a tuple in the database, which consists of four columns, including the receiving node, destination address, forwarding node, and splitting fraction.

We outline the main procedure in Alg. 2. The output $x(P)$ of Alg. 1 is a set of paths with positive flow amounts. Since Alg. 2 only produces relative fractions for flow splitting, the scaling procedure in Alg. 1 is not important for the routing decision.

In Alg. 2, we first group the path flows according to their destination addresses. Then in each group, all the paths output by Alg. 1 are enumerated. For each node on any of the paths, all flows passing through the node are aggregated, the fraction of the aggregated flows toward each outgoing link is calculated, and the corresponding tuple for each outgoing link is constructed and written into the database. Now with the generated database, the central controller constructs and distributes a table for each node j by selecting all the corresponding tuples whose receiving node is j from the database.

Algorithm 2: Generate flow-splitting database.

Input: Approximate solution $\mathbf{x}$
Output: Flow-splitting database
Initialize empty database
Group $\{x(P)\}$ by destination node n_{dst}, *i.e.*, $G(i_t) = \{P | x(P) \in \mathbf{x}, n_{dst} = i_t\}$
for each group $G(i_t)$ **do**
  $In_j = 0, Out_j(j_o) = 0, \forall j \in V, j_o \in N_o^j$, where N_o^j is the set of next-hop nodes of node j
  for each path $P \in G(i_t)$ **do**
    for each edge $(j, j_o) \in P$ **do**
      $In_j = In_j + x(P)$
      $Out_j(j_o) = Out_j(j_o) + x(P)$
    end for
  end for
  for each $j \in V$ and $In_j \neq 0$ **do**
    for each $j_o \in N_o^j$ **do**
      $\eta_{j_o} = Out_j(j_o)/In_j$
      Write a tuple $(j, i_t, j_o, \eta_{j_o})$ to database
    end for
  end for
end for

6.3 Running Time and Overhead

We note that all the flows in Alg. 1 are associated with paths, and it is easy to merge those flows over the same path while the solution $x(P)$ is recorded in the algorithm. We further modify the storing procedure of paths by adding destination indices. As a result, the grouping procedure in Alg. 2 takes $O(1)$ time.

The remaining procedure in Alg. 2 takes a time that is linear to the total number of hops over all the paths generated in Alg. 1. This number can be estimated by counting all potential peer-to-peer connections. There are at most v^2 source-destination pairs, each of which has at most m paths. Therefore, let W be the average width of the network, which is in the order of $O(\log^{O(1)} m)$). The time complexity of Alg. 2 is at most $O(v^2 mW) = \tilde{O}(v^2 m)$, which is independent of the number of video clips.

The splitting and forwarding operations are the same as those in conventional routing protocols, which do not bring (much) extra overhead for in-system nodes. Therefore, the only overhead is the extra traffic involved with the distribution of flow-splitting tables. These tables are disseminated in the control plane. Compared to existing function-incurred flows, *e.g.*, for charging and policies, such traffic overhead is negligible.

6.4 Correctness and Summary

Attentive readers may have already noticed that our routing protocol tends to aggregate the flows towards the same destination at intermediate nodes, which shows little evidence that the flows addressing the video requests are eventually routed in strict accordance to the $\{x(P)\}$ calculated by our algorithms. Although the consistency is not strictly guaranteed, we justify the correctness of our protocol by the following two facts:

- All target requests are fulfilled by the flows routed according to our protocol; and
- The total payload on each edge $e \in E$ aggregated according to our protocol equals the amount specified in $\{x(P)\}$.

As discussed in Section 6.1, in our protocol, the source servers first provide the corresponding flows according to the path-flow values $\{x(P)\}$. Therefore, the first fact is an instant result of flow conservation, *i.e.*, regardless of splitting and forwarding schemes, all fractions of generated flows will finally reach respective destinations.

To address the second fact, we investigate the traffic on an intermediate edge $e = (j, j_0)$. For all requests towards node i, the payload streaming into node j is $\sum_{k|(i,k)\in R} \sum_{P: j \in P \wedge P \in \mathcal{P}_i^k} x(P)$ according to the algorithms in preceding sections. Similarly, the aggregate load added to edge e is $\sum_{k|(i,k)\in R} \sum_{P: e \in P \wedge P \in \mathcal{P}_i^k} x(P)$. On the other hand, in the protocol, node j has the same amount of incoming payload and then dispenses the incoming flow to its outgoing edges using the flow-splitting table. Hence, the corresponding load attached to edge e is

$$\sum_{P: e \in P} \sum_{\substack{k|(i,k)\in R \\ P \in \mathcal{P}_i^k}} x(P) = \sum_{k|(i,k)\in R} \sum_{P: e \in P \wedge P \in \mathcal{P}_i^k} x(P).$$

Accordingly, the same conclusion can be derived for flows with any destination node i, which verifies the second fact. Given existing techniques for splitting and combining video streams, *e.g.*, using rateless coding [26], we now claim that our protocol achieves the same efficacy as the original algorithm with the path-flow values $\{x(P)\}$.

To summarize, the routing protocol in this section depends on proportional path-flow results generated by the proposed optimization algorithms, and dispenses hop-by-hop routing decisions to intermediate nodes with negligible overhead. Video streams towards the same destination are aggregated and split at forwarding nodes, while equivalent routing performance is guaranteed. Such a hop-by-hop forwarding design guided by destination address is simple and compatible with most existing routers. Therefore, it can be easily implemented as one plug-in component in 3GPP LTE systems.

Meanwhile, it is worth noting that the protocol is designed only for mobile networks and may not be generalized universally for content providing systems or ISPs. This is because the aggregation function exploited herein is not fully connection-oriented, whereas video streaming in general networks are often provisioned via connection-oriented

protocols, such as HTTP on top of transport control protocol (TCP). On the contrary, in 3GPP LTE networks, the data flows inside core networks are transported through connectionless user datagram protocol (UDP), which in turns provides the convenience and flexibility for data aggregation.

7 Related Works

There are many existing studies on content caching in video service systems. Most of them focus on the content placement problem, which is supposed to be solved in a time scale of weeks since updating content placement is highly time-costly. Applegate *et al.* [14] formulated a mixed integer programming (MIP) model to minimize the cost of aggregate data traffic. Although their model contains the link bandwidth constraints, their scheme is mainly designed for the content placement problem. Xie *et al.* [15] also considered the placement problem in an unstructured flat network model with the objective of minimizing maximum link utilization from the ISPs' perspective, in which future knowledge of request patterns are assumed to be known as input. The optimization work on video delivery in [6] exploited knowledge of estimated network traffic, instead of depending on assumptions of future network status.

DiPalantino *et al.* in [27] and Jiang *et al.* in [18] also investigated the joint problem of server selection and traffic engineering. They both studied the conflicting objectives of content providers and ISPs. Content providers focus on the server selection problem, while ISPs consider the traffic engineering goal. The joint problem was solved via game theory by decomposition into separate server selection and traffic engineering problems. However, both works considered much simplified scenarios where all the serving nodes can serve all the commodities (*i.e.,* video requests). Under this assumption, it is possible to add one single super node connected with all caching sources, and then directly apply the algorithms in [20]. Moreover, although the algorithms developed in [18], [27] converge to optimal solutions eventually, they do not have exact complexity guarantee or approximation guarantee at a given stop time.

In our previous works [3], [4], [5], we studied video caching in mobile networks, targeting at long-term optimizations for source access and content placement. This work differs from all the works above in that it aims to optimize instant routing decisions for video requests in mobile networks at a timescale of tens of seconds. Thus, it complements existing works by considering "dynamic" routing after "long-term optimizations" are in place. Xu *et al.* proposed a link-state routing protocol which achieves optimal traffic engineering in a distributed manner [28]. Nonetheless, their solution cannot directly apply in our scenario, where server selection remains unsolved, and source-destination pairs need to be determined.

Similar request routing problems are also studied in [19], [20], [21] and are formulated as the multi-commodity flow problem [29]. For the multi-commodity flow problem, each flow has one source and one destination. This is different from the problems under study, in which each request has one destination but potentially multiple sources. A naive solution is to create a super source node, which is connected to all caching nodes containing the corresponding requested object, and then apply the approximation algorithms in [19], [20], [21]. The issue with this approach is that the problem size (in particular, the number of nodes in the network) is artificially inflated, which leads to much higher complexity. To keep pace with request dynamics, we believe that video delivery should be efficiently optimized to guarantee both time cost and efficiency. Hence, instead of taking this naive approach, we solve the joint problems with comparable approximation ratio while maintaining very low algorithm complexity.

8 Concluding Remarks

In this work, we have studied routing for video requests in mobile networks with built-in content caching. We formulate problems with joint server selection and traffic engineering, aiming at minimizing maximum link utilization and minimizing total link cost, respectively. We have proposed fast approximation algorithms for the formulated problems. Extensive simulations are carried out to evaluate the performance of our proposed solutions. Simulation results validate the analytical conclusions, and demonstrate that our solutions perform much more efficiently and support up to 60% more traffic load than the Shortest Path scheme.

Using the path-flow solutions produced by the proposed algorithms, we further develop a practical routing protocol with a hop-by-hop forwarding design and perform detailed investigation on its running time and compliance. The protocol has negligible overhead and further improves network flows by adopting data aggregation. It is shown compatible with most existing routers and can be easily implemented in 3GPP LTE mobile networks.

The upcoming evolution of mobile networks towards the fifth generation (5G) tends to deploy software defined networking (SDN) cellular cores, so as to upgrade scalability as well as flexibility. In SDN core networks, control plane is further separated from data plane (network components to be virtualized), allowing more centralized operating functionalities for central controllers. Therefore, the algorithms together with the routing protocol developed in this paper will also apply to the next generation mobile networks.

References

[1] Cisco, "Cisco visual networking index: Global mobile data traffic forecast update, 2013-2018," http://www.

cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html, 2014.

[2] V. K. Adhikari, Y. Guo, F. Hao, M. Varvello, V. Hilt, M. Steiner, and Z. Zhang, "Unreeling NetFlix: Understanding and improving multi-CDN movie delivery," in *Proceedings of IEEE INFOCOM*, 2012, pp. 1620–1628.

[3] J. He, X. Zhao, and B. Zhao, "A fast, simple and near-optimal content placement scheme for a large-scale VoD system," in *Proceedings of IEEE International Conference on Communication Systems (ICCS)*, 2012, pp. 378–382.

[4] J. He, H. Zhang, B. Zhao, and S. Rangarajan, "A collaborative framework for in-network video caching in mobile networks," in *Proceedings of IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2013.

[5] J. He and B. Zhao, "A hybrid video caching framework in mobile core networks with CDN source suppliers," Univ. of Sci. and Tech. of China, http://www.dropbox.com/s/ne6xlep4bkwm2du/hybridtr2013.pdf, Tech. Rep., May 2013.

[6] Y. Xu, S. E. Elayoubi, E. Altman, and R. El-Azouzi, "Impact of flow-level dynamics on QoE of video streaming in wireless networks," in *Proceedings of IEEE INFOCOM*, 2013.

[7] K. Lee, H. Zhang, Z. Shao, M. Chen, A. Parekh, and K. Ramchandran, "An optimized distributed video-on-demand streaming system: Theory and design," in *Proceedings of IEEE Annual Allerton Conference on Communication, Control, and Computing*, 2012, pp. 1347–1354.

[8] X. Jin, L. E. Li, L. Vanbever, and J. Rexford, "SoftCell: scalable and flexible cellular core network architecture," in *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*. ACM, 2013, pp. 163–174.

[9] R. P. Roess, E. S. Prassas, and W. R. McShane, *Traffic engineering*. Prentice Hall, 2004.

[10] Z. Zhang, M. Zhang, A. G. Greenberg, Y. C. Hu, R. Mahajan, and B. Christian, "Optimizing cost and performance in online service provider networks." in *Proceedings of USENIX NSDI*, 2010, pp. 33–48.

[11] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[12] "OpenFlow," http://www.openflow.org/.

[13] F. Dobrian, A. Awan, D. Joseph, A. Ganjam, J. Zhan, V. Sekar, I. Stoica, and H. Zhang, "Understanding the impact of video quality on user engagement," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 362–373, 2011.

[14] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, and K. K. Ramakrishnan, "Optimal content placement for a large-scale VoD system," in *Proceedings of ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, 2010.

[15] H. Xie, G. Shi, and P. Wang, "TECC: Towards collaborative in-network caching guided by traffic engineering," in *Proceedings IEEE INFOCOM*, 2012, pp. 2546–2550.

[16] R. Krishnan, H. V. Madhyastha, S. Srinivasan, S. Jain, A. Krishnamurthy, T. Anderson, and J. Gao, "Moving beyond end-to-end path information to optimize CDN performance," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*. ACM, 2009, pp. 190–201.

[17] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows*. Prentice Hall, 1993.

[18] W. Jiang, R. Zhang-Shen, J. Rexford, and M. Chiang, "Cooperative content distribution and traffic engineering in an isp network," in *Proceedings of ACM SIGMETRICS/Performance Joint International Conference on Measurement and Modeling of Computer Systems*, 2009, pp. 239–250.

[19] N. Garg and J. Könemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems," *SIAM Journal on Computing*, vol. 37, no. 2, pp. 630–652, 2007.

[20] G. Karakostas, "Faster approximation schemes for fractional multicommodity flow problems," in *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, 2002, pp. 166–173.

[21] L. K. Fleischer, "Approximating fractional multicommodity flow independent of the number of commodities," in *Annual Symposium on Foundations of Computer Science*, 1999, pp. 24–31.

[22] S. Plotkin, D. Shmoys, and É. Tardos, "Fast approximation algorithms for fractional packing and covering problems," in *Annual Symposium on Foundations of Computer Science*, 1991, pp. 495–504.

[23] Y. Shavitt, "Routing through networks with hierarchical topology aggregation," *Journal of High Speed Networks*, vol. 7, pp. 57–73, 1998.

[24] J. Kangasharju, J. Roberts, and K. W. Ross, "Object replication strategies in content distribution networks," *Computer Communications (Elsevier)*, vol. 25, no. 4, pp. 376–383, 2002.

[25] F. J. Derfler, L. Freed, P. Douglas, L. Robbins, S. Adams, and M. Illustrator-Troller, *How networks work*, 2000.

[26] J. P. Wagner, J. Chakareski, and P. Frossard, "Streaming of scalable video from multiple servers using rateless codes," in *Proceedings of IEEE International Conference on Multimedia and Expo*, 2006, pp. 1501–1504.

[27] D. DiPalantino and R. Johari, "Traffic engineering vs. content distribution: A game theoretic perspective," in *Proceedings of IEEE INFOCOM*, 2009, pp. 540–548.

[28] D. Xu, M. Chiang, and J. Rexford, "Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering," *IEEE/ACM Transactions on Networking*, vol. 19, no. 6, pp. 1717–1730, 2011.

[29] A. Ouorou, P. Mahey, and J. P. Vial, "A survey of algorithms for convex multicommodity flow problems," *Management Science (JSTOR)*, pp. 126–147, 2000.