Evolving to 5G: A Fast and Near-optimal Request Routing Protocol for Mobile Core Networks

Jun He and Wei Song
Faculty of Computer Science
University of New Brunswick, Fredericton, Canada
Emails: {jhe2, wsong}@unb.ca

***Abstract*—Mobile networks are undergoing fast evolution from the fourth-generation (4G)/Long Term Evolution (LTE) to the fifth generation (5G) so as to keep pace with the ever-increasing data traffic, mainly fueled by large-object delivery, such as video streams. To cope with the traffic growth, next evolution will integrate functionalities of content distribution networks (CDNs) in various manners, from in-network caching and mobile CDNs to virtualized source-service points in a software-defined mobile core. In accordance with these developments, we consider the emerging request routing problem of joint source redirection and flow routing in mobile networks with built-in content sources. We develop a fast request routing protocol, which intelligently distributes traffic demands among sourcing nodes and strategically routes flows through intermediate nodes. Theoretical analysis and computer simulations show that our protocol achieves $(1+\omega)-$optimal of traffic engineering for any $\omega > 0$. Advanced features of source virtualization and data aggregation are also supported in the protocol.**

***Index Terms*—Video request routing, software-defined mobile core network, source virtualization, data aggregation.**

I. Introduction and Related Work

A. Next Evolution: Virtualization and Central Management

Global mobile data traffic will experience an 11-fold growth in the next five years according to Cisco's forecast [1]. Large-object delivery, *i.e.,* with an object size greater than 1 megabytes, such as video streaming, is dominating the traffic growth, which usually requires high bandwidth and low latency. Such growth is, however, challenging the fundamental of current mobile systems, *e.g.,* the Long Term Evolution (LTE) systems, which are designed as "interconnecting middle-boxes" between public networks and mobile users. Mobile systems are therefore in urgent need of more efficient content delivery.

To counter this, transparent content caching and delivery techniques are extensively investigated in the literature [2]–[6] so as to move data closer to customers and improve the quality of experience (QoE). For instance, in an LTE system depicted in Fig. 1, caching capacities at serving gateways are exploited in [4], while built-in mobile content distribution networks (MCDNs) or cloud serving points are studied in [3], [7]. One essential development of LTE is the separation of control plane and data plane, where the control traffic (*e.g.,* for charging and policies) is delivered in control plane, while users' requested data are transmitted in data plane. Central management is incorporated into certain key components of the system, *e.g.,* the packet data network gateway (PDN-GW).

In order to facilitate scalability, flexibility and cost-efficiency, mobile networks are now evolving to the fifth generation (5G) [8] with virtualized network functions [9]. 5G mobile networks tend to employ software defined networking (SDN) [10] techniques in core networks, where network equipment is regarded as "computing equivalent", gathering programmable resources based on virtualization technologies. Fig. 2 displays the basic architecture of a 5G system, where data service points can be network components with storage and sourcing capacities, possibly integrated in routers [11] or gateways. To enable a more manageable system, a central controller is setup for collecting network-wide information as well as managing resources via the control plane.

In all the evolving network models above, central controllers can easily gather network information, shape network traffic and accordingly optimize network performance. On the other hand, existing routing schemes fail to incorporate content sources within core networks. Therefore, it is practical and of vital importance to develop a fast request routing protocol, which adapts to the new network architecture, jointly considering source redirection and flow routing.

B. Request Routing: Source Redirection and Flow Routing

Traffic engineering (TE), *i.e.,* minimizing maximum link utilization, is one of the most popular routing design objectives, and has a great impact on the network capacity and stability.

Conventional routing protocols for TE optimization operate at lower layers, such as open shortest path first (OSPF) at link layer and OSPF-enhanced schemes at network layer [12], [13], providing optimized flow routing for a given set of source-destination traffic demands. For fixed source-destination requests, state-of-art lower-layer routing algorithms include link-state protocols [14], multicommodity-flow model protocols [13] and the latest TE study in hybrid SDN [15]. All these works only consider flow routing, where source selection (redirection) for requests is assumed pre-determined. In this work, we move attention to higher layers, *i.e.,* source redirection at application layer and flow routing at transport layer.

Evolving from 4G to 5G, mobile operators are integrating content sources within their core networks, *e.g.,* virtualized data-service points in SDN-based cores, information-centric networks with improved routers [11]. In evolving mobile core networks, the central controller gathers request patterns from

This research was supported by Mariner Partners Inc. and Atlantic Canada Opportunities Agency, Canada.

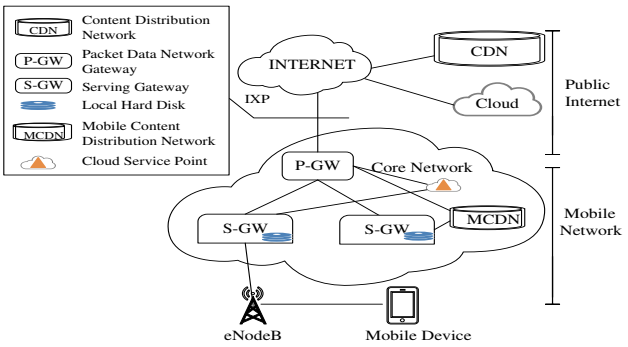


Figure 1. LTE network architecture.

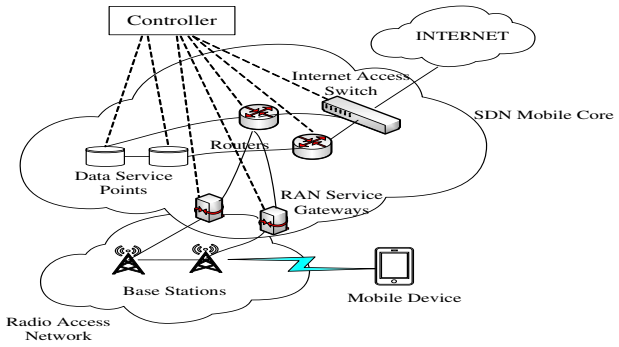


Figure 2. Architecture of future mobile network.

edge nodes, *e.g.,* serving gateways in Fig. 1 or radio access network (RAN) service gateways in Fig. 2, and redirects requests to corresponding sources as well as determines flow routing decisions. Meanwhile, the dynamics of requests with continuous traffic demands (*e.g.,* video streams), are usually in several seconds [16], allowing such centralized routing optimization. However, given the fact that source selection remains an intractable problem in generic networks [17], request routing in mobile networks is therefore more challenging.

In our previous work [18], we have considered similar request routing problem for video delivery. This work complements [18] by studying more design details of the routing protocol, considering a protocol-oriented routing problem, and extending the former work to emerging mobile networks and generic object delivery.

C. Our Contributions

Our contributions are three-fold. First, to our best knowledge, this is the first work to define and study the request routing problem in the evolving mobile network, where joint source redirection and flow routing are investigated. This problem is essential to high-performance data delivery in various mobile network models evolving to 5G.

Second, we propose a complete request routing protocol with close scrutiny into new networking paradigms. Our protocol handles source redirection at application layer integrating source virtualization while optimizing generated flows at transport layer with improved data aggregation.

Third, we formulate a protocol-oriented routing problem aiming to minimizing maximum link utilization (traffic engineering), where requested data are split among sources as well as network edges. We propose a fully polynomial-time approximation algorithm for the formulated problem, which achieves an approximation guarantee of $1 + \omega$ with time complexity proportional to ω^{-2}, for any $\omega > 0$. The algorithm outputs independent decisions of source redirection and flow splitting to be implemented in the routing protocol.

The rest of this paper is organized as follows. Section II presents our request routing protocol. Section III formulates the optimization problem followed by the the proposed algorithm in Section IV. Section V presents numerical results. Finally, Section VI concludes this paper.

II. Target Routing Protocol

In this section, we introduce the detailed design and features of our protocol, while leaving routing decisions to be modeled and determined in Sections III and IV.

A. Protocol Model: What to Be Routed

We consider large-object delivery in an evolving mobile network with a central controller. Large objects are transmitted in a progressive downloading mode associated with periodical requests at fixed data rates per several seconds. For instance, a video clip of 30 megabytes encoded in Flash (FLV) supports traffic engagement of $3 \sim 4$ minutes (in the same order of playback time) with a resolution of 640×360. We assume that the controller has the knowledge of the network topology as well as content distribution of data sources. The content distribution is typically updated on a weekly basis. We also assume that requested flow can be arbitrarily split among source nodes and intermediate nodes, which is a common assumption in existing works, *e.g.,* [17], [19], [20].

Our protocol works periodically. At the beginning of each period, the controller gathers the information of all data requests, and decides how these requests are fulfilled among accessible sources and so-generated traffic are routed over links. Data requests are initiated by clients, while reaching the core network at edge nodes. We study request routing in core networks, focusing on delivery of requested data from sources to corresponding edge nodes.

B. Protocol Overview: How It Works

In the protocol, the central controller periodically computes the source-redirection decisions as well as the flow-splitting decisions according to the network condition and request pattern in the current period. The source-redirection decisions indicate how the demanding load of each request is shared among available sources. And the flow-splitting decisions guide how an intermediate node splits and forwards incoming flow destined to specific edge nodes over its outgoing links.

Source-redirection decisions are associated with corresponding requests. For instance, an edge node i requests for an object k with traffic demand d_i^k. S_k is the set of source nodes that can serving object k. Then a source-redirection decision regarding the request specifies the amount of flow to be loaded on each potential source. With these parameters, the central controller notifies involved source nodes to deliver the corresponding amount of flows for each request destined to the requesting node. Specifically, for each destination D, a source node generates and combines all the requested data traffic towards D upon requests from the controller and adds a

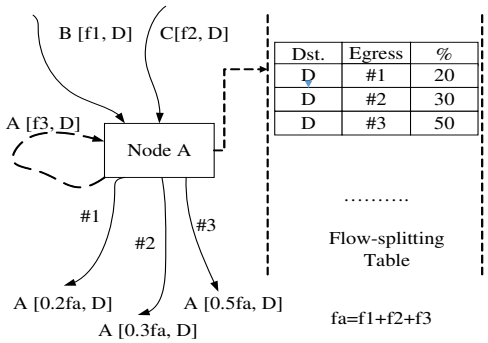


Figure 3. A routing example based on flow-splitting table. A has two physical incoming flows $f1$ from B and $f2$ from C with the same destination D, as well as one logical incoming flow $f3$ generated by itself (A is a target source node selected for some objects requested by node D). For transmission purpose, A reads the destination identifier of each incoming flow, and then groups flows with the same destination, *e.g.,* D. For each destination, A then looks up local flow-splitting table to obtain the splitting fractions for next-hop nodes. After that, it combines same-destined flows, splits and forwards aggregated flows to respective links, *i.e.,* $\#1$, $\#2$, and $\#3$, according to flow-splitting fractions.

logical incoming flow to it, possibly via local loop-back. This operation is implemented at application layer.

Meanwhile, the controller generates and disseminates a flow-splitting table to each node regarding the flow-splitting decisions. One tuple of the table consists of three elements: destination identifier, outgoing link identifier and splitting ratio. Fig. 3 shows an example of the flow-splitting decision in our protocol. This is operated at transport layer.

C. Features: Source Virtualization and Data Aggregation

Even though source redirection and flow routing is jointly studied later in this paper, the controller disseminates these two types of traffic policies independently, in two specific phases, namely a flow-generating phase and a flow-routing phase, as discussed in Section II-B. In the flow-generating phase, an on-demand source fetches network objects, generates and attaches flows to its loop-back link. In the flow-routing phase, flows are labeled only by destination identifiers. In this phase, all nodes are considered as intermediate nodes without knowledge of sources, which implements source virtualization in a sense.

Network nodes in an LTE mobile core exchange data using the general packet radio service (GPRS) tunneling protocol (GTP), which is built on the top of user datagram protocol (UDP). Data packets are encapsulated into GTP frames on entering the core network and decapsulated on exiting. Inside the core network, encapsulated GTP frames are transmitted in a connectionless fashion among nodes. Connectionless delivery provides convenience and flexibility for data aggregation, *i.e.,* UDP frames destined to same edge nodes are readily aggregated to form larger frames. Such data aggregation simplifies traffic management and reduces protocol overhead [21].

The remaining work is therefore to determine the two sets of flow-splitting parameters for our protocol, regarding flow amount generated at sources for each request and flow-splitting ratios at each node for incoming flows destined to edge nodes, respectively. We next formulate the the routing problem mathematically in Section III and propose a fast solution in Section IV to obtain the required parameters.

III. System Formulation and Optimization Framework

A. System Formulation

We model the mobile core network as a directed graph $G = (O, V, E)$, where O is the set of available objects ($o = |O|$), V is the set of interconnected networking nodes ($n = |V|$), and E is the set of links ($m = |E|$). Each link e has a capacity $c(e)$. Set V may include content source nodes, access gateways, intermediate routers or any components with networking functionalities.

Let (i, k) denote a data request at node i for object k to be served by the set of nodes S_k, which contain object $k \in O$. Each request is associated with a positive flow demand d_i^k. Denote the set of all requests by R with $r = |R|$.

A routing scheme determines how the requests to be fulfilled over the network. That is, for each request (i, k), the controller computes the data amount generated by each source in S_k, denoted by $b_i^k(s), s \in S_k$. Then, the controller needs to calculate the flow-splitting ratios for each destination i over every intermediate node $u \in V$, denoted by $q_i(u, e)$, where $sn(e) = u$. For ease of presentation, we define functions $sn(\cdot)$ and $tn(\cdot)$ that return the start node and the end node of an input link e, respectively, *i.e.,* $e = (sn(e), tn(e))$.

B. Optimization Framework

Let $f_i^k(e)$ denote the flow amount associated with each request (i, k) over each link e. Then, the problem is to determine the flow set $\{f_i^k(e)\}, \forall e \in E, (i, k) \in R$ so as to achieve the TE objective, *i.e.,* minimum maximum link utilization. Apparently, this requires to simultaneously address server redirection and flow routing. Specifically, we formulate the following linear programming (LP) problem to minimize the maximum link utilization:

$$\begin{aligned} \min \quad & \lambda \\ \text{s. t.} \quad & \sum_{(i,k)\in R} f_i^k(e) \le \lambda c(e), \ \forall e \in E \\ & \sum_{u\in S_k} \Big(\sum_{e:sn(e)=u} f_i^k(e) - \sum_{e:tn(e)=u} f_i^k(e) \Big) \ge d_i^k, \ \forall (i,k) \in R \\ \text{vars.} \quad & f_i^k(e) \ge 0, \ \forall (i,k) \in R, e \in E \end{aligned} \tag{1}$$

where the first constraint implies the link capacity limitation, scaled by the link utilization factor λ, and the second constraint satisfies the demand for object k at node i. Here, $f_i^k(e)$ represents the flow amount destined to node i, which also implies flow conservation [22]. The optimization in (1) is to be computed periodically at the controller at the same time scale as the dynamics of clients' requests, *e.g.,* for video delivery, typically in seconds.

Although exact solutions to problem (1) are tractable, fast computation is infeasible with modern LP solvers due to the prohibitively large size. For instance, in a production system with 100 nodes, the number of requests in R is $O(10^5)$, and the number of edge is $O(10^3)$. The rudimentary size of problem (1) is with $O(10^8)$ variables and $O(10^5)$ constraints.

We note that the formulation (1) is similar to the multicommodity flow problem, which has been extensively studied in

the literature [12], [13]. Similar TE problem in Intermediate System-Intermediate System (IS-IS) is formulated and solved with exact optimal solution using link-state routing in [14]. There is a key difference though. Existing works and solutions rely heavily on the fact that peer-to-peer traffic requirements are pre-calculated such that flows are split only among intermediate nodes/routers, since the major traffic of conventional networks is connection-oriented, *e.g.,* via the transmission control protocol (TCP). In contrast, our model targets more scalability and flexibility where requested flows can also be split among sources, making existing solutions infeasible. Jointly considering the computing time limit (bounded by the length of a period), we conclude that a fast approximation algorithm with bounded performance guarantees is more desirable.

Recalling the requirements of our protocol in Section II-C, we comment that only the proportional flow amount of $\{f_i^k(e)\}$ needs to be calculated, while the source generated flow amount can be computed with requested demands thereafter.

IV. Proposed Fast Algorithm

A. Alternative Flow-Path Model

For each pair of nodes $i, j \in V$, $\mathcal{P}_{ij}$ denotes the set of paths from i to j, and $\mathcal{P} = \cup_{(i,j)}\mathcal{P}_{ij}$ presents the union of all path sets. Moreover, let $\mathcal{P}_i^k$ be the set of all available paths to serve request $(i,k) \in R$, formally, $\mathcal{P}_i^k = \cup_{j \in S_k}\mathcal{P}_{ji}$.

Next, we consider an implicit form of problem (1), formulated as follows:

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

Solutions to problem (2) output the flow amount $x(P)$ on each path P, which is readily mapped to flow amount over links by enumerating links over each path P in time $O(|\mathcal{P}|W)$, where W is the average width of the network.

Note that in problem (2), $\{x(P)\}$ is scalable to the objective γ. In other words, if we scale all $\{x(P)\}$ by $1/\gamma$, we alter problem (2) into the form of the maximum concurrent flow problem [23], which is formulated as:

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

If we let $x(P) = y(P)/\pi, \gamma = 1/\pi$, we note that problems (2) and (3) are equivalent. Problem (3) is a variant of the maximum concurrent flow problem, where each commodity has one destination and (possibly) multiple sources. This problem was first formulated and investigated in our previous work [18]. In the following, we extend the algorithm developed in [18] to solve problem (1) without extra enumeration of paths.

B. Proposed Approximation Algorithm

First, we list the dual form of problem (3) as follows:

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

where $l(e)$ is the length function of edge e, z_i^k is the dual variable associated with request (i,k).

According to the analysis in [18], z_i^k can be viewed as the minimum cost for shipping unit data from sources in S_k to node i under length l. We define the aggregate shipping cost for all requests by $\alpha(l)$, given by

$$\alpha(l) \triangleq \sum_{(i,k)\in R} d_i^k \psi_i^k(l) \tag{5}$$

where $\psi_i^k(l)$ denotes the length value of the shortest path over all available paths regarding (i,k) under function l. Then, solving problem (4) is equivalent to assigning a length l to each edge, such that $D(l)/\alpha(l)$ is minimized. The optimal value of problem (4) is expressed as:

$$\beta \triangleq \min_l D(l)/\alpha(l). \tag{6}$$

Problem (4) is readily solved using the algorithm in [18]. The main procedure of our algorithm is outlined in Alg. 1. Similarly, our algorithm proceeds in a phase-iteration-step manner. Initially, the length value of edge $e \in E$ is set to $l(e) = \delta/c(e)$, where δ is a parameter to be defined later. The algorithm then proceeds in phases, while in each phase, all demands of requests are satisfied by $|V|$ iterations. Each iteration j considers all requests initiated at node j through a series of steps. In each step, shortest available paths are considered and flows are assigned to links such that at least one edge is saturated or no more flow remains to be routed. At the end of each step, the length function along shortest paths is updated by

$$l(e) \quad = \quad l(e)(1 + \epsilon \cdot \frac{\text{sum of new flows on } e}{c(e)})$$

where ϵ is an accuracy factor.

In Alg. 1, $y_i(e)$ denotes the flow amount destined to node i on edge e, E_i records all the edges used for delivering responses to node i, $\tilde{d_j^k}$ presents the unfulfilled demand of request (j,k) and is initialized with d_j^k at each phase, and $c'(e)$ is the remaining capacity of edge e in each step. In step (i) of phase j, either all remaining demands at node j are fulfilled (see line 17) or at least one edge is saturated (see line 23). Here, f_j^k is the actual amount of flow attached to the shortest path with respect to request (j,k). For each unit of routed flow, we accumulate it in source generated flow amount parameter $\{b_j^k(s_j^k)\}$ as well as in edge load $\{y_j(e)\}$ correspondingly.

With similar analysis in [23], Alg. 1 results in an infeasible edge-flow solution after termination of phases, which can be

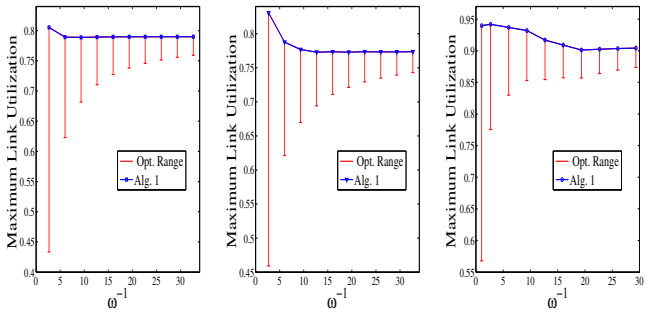
Algorithm 1: Algorithm for problem (1).

Input: Network graph $G = (O, V, E)$, edge capacities $c(e)$, object sources set S_k, requests set R, request demands $\{d_j^k\}$, accuracy factor ϵ

Output: Source generated flow amount $\{b_i^k(s)\}$ and flow-splitting fraction $\{q_i(u,e)\}$

1: Initialize
$l(e) \leftarrow \delta/c(e), \forall e \in E; y_i(e) \leftarrow 0,\ \forall e \in E, i \in V;$
$b_i^k(s_k) \leftarrow 0,\ \forall (i,k) \in R, s_k \in S_k; E_i \leftarrow \emptyset, \forall i \in V.$
2: **while** $D(l) < 1$ **do**
3: **for** $j = 1$ to v **do**
4: For all requests $(j,k) \in R_j$, initialize $\tilde{d}_j^k \leftarrow d_j^k$
5: **while** $D(l) < 1$ and $\tilde{d}_j^k > 0$ for some k **do**
6: $K \leftarrow \{k | (j,k) \in R_j, \tilde{d}_j^k > 0\}$
7: $P_j^k \leftarrow$ shortest path in $\mathcal{P}_j^k$ using l, $\forall k \in K$
8: $c'(e) \leftarrow c(e)$, $\forall e \in \bigcup_{k \in K} P_j^k$
9: $E_j \leftarrow \bigcup_{k \in K} P_j^k \bigcup E_j$
10: **for** $k \in K$ **do**
11: $c \leftarrow \min_{e \in P_j^k} c'(e)$, s_j^k is the first node in P_j^k.
12: **if** $\tilde{d}_j^k \leq c$ **then**
13: $b_j^k(s_j^k) \leftarrow b_j^k(s_j^k) + \tilde{d}_j^k$
14: **for** $e \in P_j^k$ **do**
15: $y_j(e) \leftarrow y_j(e) + \tilde{d}_j^k$, $c'(e) \leftarrow c'(e) - \tilde{d}_j^k$
16: **end for**
17: $f_j^k \leftarrow \tilde{d}_j^k$, $\tilde{d}_j^k \leftarrow 0$
18: **else**
19: $b_j^k(s_j^k) \leftarrow b_j^k(s_j^k) + c$
20: **for** $e \in P_j^k$ **do**
21: $y_j(e) \leftarrow y_j(e) + c$
22: **end for**
23: $\tilde{d}_j^k \leftarrow \tilde{d}_j^k - c$, $f_j^k \leftarrow c$, **break**
24: **end if**
25: **end for**
26: $l(e) \leftarrow l(e)\left(1 + \epsilon \frac{\sum_{k: e \in P_j^k} f_j^k}{c(e)}\right)$, $\forall e \in \cup_{k \in K} P_j^k$
27: **end while**
28: **end for**
29: **end while**
30: **for** $i \in V$ *and* $(i,k) \in R_j$ **do**
31: $b_i^k(s_i^k) \leftarrow \frac{b_i^k(s_i^k)}{\sum_{s \in S_k} b_i^k(s)} \times d_i^k, \forall s_i^k \in S_k$
32: **end for**
33: **for** $i \in V$ **do**
34: Let U be the vertex set of E_i
35: **for** $u \in U$ **do**
36: For each e with $sn(e) = u$,
$q_i(u,e) \leftarrow \frac{y_i(e)}{\sum_{e': sn(e') = u \wedge e' \in E_i} y_i(e')}$
37: **end for**
38: **end for**

scaled down proportionally to obtain a feasible solution. In contrast, here we only need the proportional solution. Each request demand is split among available sources and is further converted to source generated flow amount (see line 31). Flow-splitting ratios are then calculated on each edge regarding specific destinations (see line 36).



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

Figure 5. Optimal gap with different approx. ratios ω.

C. Correctness, Approximation and Running Time

Compared to the algorithm in [18], we comment that the path-flow solution therein can be easily recovered from output of the modified algorithm here. Accordingly, the correctness therein also applies to Alg. 1.

The analysis in [18] also results in the following conclusion:

Theorem 1. *The algorithm in [18] obtains a* $(1+\omega)-$*approximation result for problem (2) for any* $\omega > 0$ *by setting*

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

and $\epsilon = 1 - 1/\sqrt[3]{(1+\omega)}$.

Therefore, we conclude that Alg. 1 obtains a $(1+\omega)$ approximation solution to (1), for any $\omega > 0$, with appropriate selection of accuracy factor ϵ and δ.

In Alg. 1, the total number of steps is $O(\epsilon^{-2} m \log m + n \log m \log r)$ according to the analysis in [18]. Inside each step, using Fibonacci heaps, the computation of a shortest path tree requires $O(n \log n + m)$ for one run of Dijkstra's algorithm on the reversed graph of G. Other cost includes updating source flow variables which takes at most $O(r_{\max})$, where $r_{\max}$ denotes the maximum number of requests from one node, and updating edge flows, which takes at most $O(nW)$. Correspondingly, we denote the cost of each step by $C_s \triangleq O(r_{\max} + n \log n + nW)$. Therefore, Alg. 1 computes a $(1+\omega)$-approximation solution to problem (1) in $\tilde{O}((\omega^{-2} + \log r)m \cdot C_s)$ time, for any $\omega > 0$, where $\tilde{O}(f) = O(f \cdot \log^{O(1)} m)$.

V. Performance Evaluation

In this section, we evaluate the performance of our algorithm using computer simulations. We employ the simulator developed in [18] and implement Alg. 1 with around 500 lines of C++ code. The simulated system is a ring-star network, where $n-1$ serving nodes are interconnected in a ring and node 1 is linked to each of the other $n-1$ nodes, respectively. There is also a virtual data repository in the network, connected to node 1 and node $n-1$. All links are bi-directional with a capacity of 1 Gbps for each direction. There are $200,000$ objects in the repository, which are distributed among serving nodes so that each object has 6 replicas. We randomly generate requests at each node, controlled by the factor of *traffic density*, which represents the average number of requests initiated by each node. The demand of each request is uniformly generated from

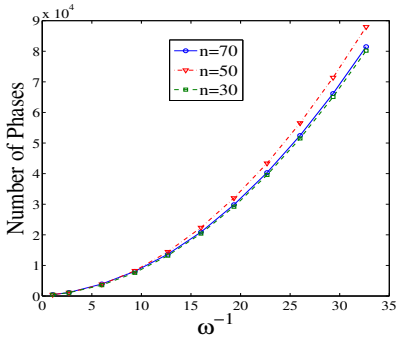


Figure 6. Number of required phases vs. approximation ratios.

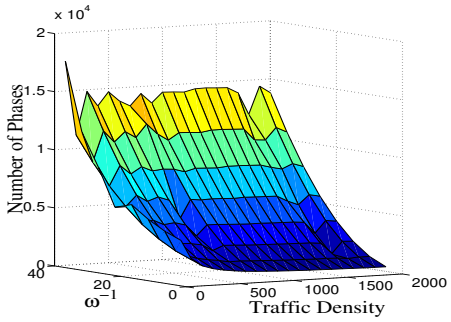


Figure 7. Number of phases vs. approximation ratios and traffic density.

128 $Kbps$ to 1 $Mbps$. Three cases with the number of nodes 30, 50, 70 are simulated under various conditions. For each instance, we run Alg. 1 to route the requests.

Alg. 1 produces an upper bound for the optimal traffic engineering of problem (1). To lay out the optimal range, we use the algorithm in [18] to obtain a lower bound. Fig. 5 shows the maximum link utilization computed by Alg. 1 with different approximation factors ω, together with the optimal range. We can conclude from the figure that Alg. 1 is arbitrarily near-optimal by selecting ω properly.

However, it is more time-consuming to obtain approximate results closer to the optimal (see Section IV-C), which requires more phases in the algorithm. To have a numerical understanding, we record the number of phases in each of the three running cases. Fig. 6 shows the relationship between the number of phases and the approximation parameter ω. The three quadratic curves therein validate the conclusion that the number of phases in Alg. 1 grows quadratically with ω^{-1} (see Section IV-C). Fig. 7 further illustrates the impact of traffic density on running time. In the network with 50 nodes, traffic density is increased from 100 to 2000 with a step size of 100. The slant surface towards higher traffic density in Fig. 7 also confirms our intuition that the link utilization increases with traffic density such that the algorithm requires fewer phases.

All simulations are run on a Linux machine with an Intel-i7 processor and 4G RAM. The simulator uses around $85M$ memory for each instance. In a system of 50 nodes with traffic density of 800, *i.e.,* $400K$ concurrent requests, it costs around 4 seconds to route all requests with a traffic-engineering approximation ratio of $\omega = 0.05$.

VI. Conclusion

In this work, we have studied the request routing problem in mobile core networks evolving from 4G/LTE to 5G. A complete protocol has been developed to address both source redirection and flow routing. In order to compute the routing decisions required by the protocol, we formulate the request routing problem as an LP problem. We then propose a fast algorithm which solves the formulated problem and achieves $(1 + \omega)-$optimal of traffic engineering, for any $\omega > 0$. Although our protocol targets the next generation mobile core, it also applies to a 4G system with built-in content caching and a central controller.

References

[1] Cisco, "Cisco visual networking index: Global mobile data traffic forecast update, 2013-2018," February 2014.

[2] S. Woo *et al.*, "Comparison of caching strategies in modern cellular backhaul networks," in *Proc. of ACM Annual International Conference on Mobile Systems, Applications, and Services*, 2013, pp. 319–332.

[3] F. Z. Yousaf *et al.*, "Mobile CDN enhancements for QoE-improved content delivery in mobile operator networks," *IEEE Network*, vol. 27, no. 2, pp. 14–21, 2013.

[4] J. He *et al.*, "A collaborative framework for in-network video caching in mobile networks," in *Proc. of IEEE Conference on Sensor, Mesh and Ad Hoc Comm. and Networks (SECON)*, 2013, pp. 406–414.

[5] 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 Comm. Systems (ICCS)*, 2012, pp. 378–382.

[6] J. Zhu *et al.*, "Epcache: In-network video caching for lte core networks," in *Proc. of IEEE International Conference on Wireless Communications & Signal Processing (WCSP)*, 2013, pp. 1–6.

[7] M. Liebsch *et al.*, "Runtime relocation of CDN serving points-enabler for low costs mobile content delivery," in *Proceedings of IEEE Wireless Comm. and Networking Conference (WCNC)*, 2013, pp. 1464–1469.

[8] ADVA, "Horizon 2020 advanced 5G network infrastructure for future Internet PPP, draft version 2.1," 2013.

[9] N. Chowdhury and R. Boutaba, "A survey of network virtualization," *Computer Networks*, vol. 54, no. 5, pp. 862–876, 2010.

[10] Open Networking Foundation, "Openflow-enabled SDN and network functions virtualization," *ONF Solution Brief*, 2014.

[11] B. Ahlgren *et al.*, "A survey of information-centric networking," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 26–36, 2012.

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

[13] A. Sridharan, R. Guerin, and C. Diot, "Achieving near-optimal traffic engineering solutions for current OSPF/IS-IS networks," *IEEE/ACM Transactions on Networking*, vol. 13, no. 2, pp. 234–247, 2005.

[14] 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.

[15] S. Agarwal, M. Kodialam, and T. V. Lakshman, "Traffic engineering in software defined networks," in *IEEE INFOCOM*, 2013, pp. 2211–2219.

[16] F. Dobrian *et al.*, "Understanding the impact of video quality on user engagement," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 362–373, 2011.

[17] H. Xu and B. Li, "Joint request mapping and response routing for geo-distributed cloud services," in *IEEE INFOCOM*, 2013, pp. 854–862.

[18] J. He and W. Song, "Optimizing video request routing in mobile networks with built-in content caching," University of New Brunswick, Canada, http://cs.unb.ca/~wsong/optrouting.pdf, Tech. Rep., March 2014.

[19] P. Wendell *et al.*, "DONAR: Decentralized server selection for cloud services," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4, pp. 231–242, 2010.

[20] R. Krishnan *et al.*, "Moving beyond end-to-end path information to optimize CDN performance," in *Proceedings of ACM SIGCOMM Conference on Internet Measurement Conference*, 2009, pp. 190–201.

[21] M. Castrucci *et al.*, "A cognitive future Internet architecture," in *The Future Internet*. Springer, 2011, pp. 91–102.

[22] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, "Network flows: Theory, algorithms, and applications," 1993.

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