QoS-Aware and Reliable Traffic Steering for Service Function Chaining in Mobile Networks

Ruozhou Yu, *Student Member, IEEE*, Guoliang Xue, *Fellow, IEEE*, and Xiang Zhang, *Student Member, IEEE*

***Abstract*— The ever-increasing mobile traffic has inspired deployment of capacity and performance enhancing network services within mobile networks. Owing to recent advances in network function virtualization, such network services can be flexibly and cost-efficiently deployed in the mobile network as software components, avoiding the need for costly hardware deployment. Nevertheless, this complicates network planning by bringing the need for service function chaining. In this paper, we study mobile network planning through a software-defined approach, considering both quality-of-service and reliability of different classes of traffic. We define and formulate the traffic steering problem for service function chaining in mobile networks, which turns out to be $\mathcal{NP}$-hard. We then develop a fast approximation scheme for the problem, and evaluate its performance via extensive simulation experiments. The results show that our algorithm is near-optimal, and achieves much better performance compared with baseline algorithms.**

***Index Terms*— Software-defined networking, mobile networks, service function chaining, quality-of-service, reliability.**

I. Introduction

THE recent years have witnessed a drastic growth on global mobile traffic, due to the prevalent use of personal mobile devices and the emergence of the internet-of-things (IoT). Billions of devices are connected via mobile networks, posing a severe challenge to current mobile infrastructures. Moreover, the greatly abundant mobile and IoT applications have very heterogeneous requirements, including quality-of-service (QoS), security, availability, etc. Satisfying these requirements is difficult for current mobile networks, largely due to their hierarchical nature: most QoS and security features are implemented at the gateway or in the cloud, in a centralized manner. The gateway, with the need to both serve the huge amount of traffic and provide fine-grained network control, becomes a severe performance bottleneck of the mobile network.

There have been many efforts in addressing this performance bottleneck. The key idea is to resolve as much traffic as possible within the mobile network, alleviating the load on the gateway. One promising method is to deploy capacity and performance enhancing network services, also called *middleboxes*, to provide in-network traffic processing before traffic reaches the gateway. These include security components such as firewall, intrusion detection/prevention system (IDS/IPS) and deep packet inspection (DPI), network optimization tools such as load balancer (LB) and TCP optimizer, network address translator (NAT), etc. Deploying network services can bring a lot of benefits, such as early resolution of useless or malicious traffic, load balancing, security enhancement, etc.

Traditionally, each network service is implemented via dedicated hardware pieces, hence can only be deployed at specific locations (most likely at the gateway) due to cost issue. Thanks to the recent advances in network function virtualization (NFV), many network services can now be implemented as software components hosted on general-purpose computation platforms at the network edge, such as fog computing nodes within the mobile network. Edge deployment of network services has several advantages. First, this alleviates the excessive traffic load at the gateway. Second, in-network processing can effectively reduce traffic size in many scenarios, *e.g.*, data preprocessing for big data analytics, or preventing distributed deny-of-service (DDoS) attacks. Third, this reduces the delay experienced by mobile traffic, especially those transmissions whose both end-points reside in the mobile network (*e.g.*, machine-to-machine communications). With the emergence of fog computing [2], network services can be flexibly distributed in the network, which further helps in network optimization to balance and resolve mobile traffic load.

Nevertheless, benefits often do not come without a cost. Along with the enhanced performance and enriched flexibility, comes the increased complexity for *service function chaining* (SFC). In SFC, each traffic class is assigned a *service function chain*, which is a sequence of network services (also called *service functions*) that the traffic needs to pass through before exiting the network. Different traffic classes may have different service function chains, due to their various QoS, security and reliability requirements. An important problem is to steer each class of traffic through its required network services in the given order, wherein both routing and bandwidth allocation need to be determined based on the traffic class's requirements.

In this paper, we study the traffic steering problem in mobile networks. We take a software-defined approach, where a centralized controller collects global network information, and makes joint routing and allocation decisions for all traffic classes together. A software-defined approach commonly achieves better routing and resource optimization, compared to distributed or local optimization approaches. We formulate the QoS-aware and Reliable Traffic Steering (QRTS) problem in mobile networks, considering heterogeneous requirements

Manuscript received April 1, 2017; revised September 12, 2017; accepted September 25, 2017. Date of publication October 5, 2017; date of current version December 1, 2017. This work was supported by NSF under Grant 1461886 and Grant 1704092. *(Corresponding author: Guoliang Xue.)*

The authors are all with the School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ 85287 USA (e-mail: ruozhouy@asu.edu; xue@asu.edu; xzhan229@asu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSAC.2017.2760158

of traffic classes, including QoS (throughput and delay), reliability, security and type-of-transmission constraints, etc. Both QRTS and its optimization version (OQRTS) are proved to be $\mathcal{NP}$-hard. We then propose a Fully-Polynomial Time Approximation Scheme (FPTAS) for the optimization problem. Through extensive simulation experiments, we validate that our proposed algorithm produces near-optimal solutions, and greatly outperforms two baseline heuristic algorithms.

Our main contributions are summarized as follows:

- To the best of our knowledge, we are the first to formulate the traffic steering problem in mobile networks with QoS and reliability requirements, and prove its $\mathcal{NP}$-hardness.
- We develop a Fully-Polynomial Time Approximation Scheme for the optimization problem.
- We evaluate the performance of our proposed algorithm via extensive simulation experiments, which validates the near-optimal performance of our algorithm.

The rest of this paper is organized as follows. In Section II, we introduce existing work related to this paper. In Section III, we present our network and service model. In Section IV, we formally define and formulate the QRTS problem and its optimization version, and prove that both problems are $\mathcal{NP}$-hard. In Section V, we then propose our algorithm for the problem, and analyze its performance guarantee and time complexity. In Section VI, we present our performance evaluation results. In Section VII, we conclude this paper.

II. Background and Related Work

A. NFV and SFC

NFV has been recognized as one of the enabling technologies to next-generation mobile networks [13], [21]. Various network components can be implemented via virtualization [13], including gateway, mobility support, charging, etc. In addition, traditional services like firewall, IDS/IPS, network optimizer and NAT, can also be implemented in mobile networks based on operator and user demands. Recent advances in NFV enable flexible and cost-efficient deployment of various network services in mobile networks [9], [17], [20].

SFC is a problem arisen in network management in the presence of network services. Gember *et al.* [8] proposed a network orchestration layer, in which issues such as elastic scaling, flexible placement and flow distribution are addressed. Zhang *et al.* [31] studied SFC in the view of network protocols, and proposed a heuristic solution for SFC-aware network service placement. Bari *et al.* [1] studied service chain embedding (joint traffic steering and network service placement), and proposed another heuristic solution. Rost *et al.* [26] proposed the first approximation algorithm for service chain embedding, though the approximation is not constant-ratio. For solely the traffic steering problem, Cao *et al.* [3] proposed an FPTAS, which is similar to the result reported in this paper. However, their problem does not consider traffic QoS, hence is not $\mathcal{NP}$-hard. Our problem considers the heterogeneous QoS requirements of applications, and is $\mathcal{NP}$-hard, hence an FPTAS is the best possible algorithm that we can expect unless $\mathcal{P} = \mathcal{NP}$. Guo *et al.* [11] studied traffic steering with pre-defined path sets, which are not assumed in our paper. The applications of SFC in mobile networks have been summarized in [12].

Failure of network services can be frequent and can have a large impact on the performance of applications and services [24], hence is one of the major considerations in this paper. Rajagopalan *et al.* [25] proposed a Software-Defined Networking (SDN) based replication framework for network services. Sherry *et al.* [27] proposed a log-based recovery model for network services or middleboxes, which can be used to fast recover failed network services. Fan *et al.* [5] and Ye *et al.* [29] studied the problem of reliable service chain embedding, and proposed different heuristic algorithms based on both dedicated backup and shared backup provisioning. Kanizo *et al.* [18] proposed a network-agnostic solution for network service backups based on bipartite matching. The above efforts all focus on providing full recovery of the failed network services. On the contrary, we argue that this "all-or-nothing" protection is an overkill for many applications, as shown in existing work [30]. Therefore we propose a "soft" reliability mechanism, such that only a bounded portion of throughput is affected during an arbitrary service failure.

B. Software-Defined Mobile Networks

SDN has recently been applied to facilitate network configuration and management in various network environments, including the mobile networks. Different usages of SDN in mobile networks have been studied, including resource allocation in the Radio Access Networks (RANs) [10], traffic control in the Mobile Core Networks (MCN) [15], topology reconfiguration [23], etc. Our approach utilizes the centralized control as in the above, but considers its application in QoS-aware and reliable traffic steering for SFC in mobile networks.

III. System Model

A. Network Topology

The SDN controller aggregates global network information. The network is modeled as a directed graph $G = (V, E)$, where V is the set of nodes, and E is the set of links. A mobile network consists of many heterogeneous nodes, including radio access points (RAPs), core switches, standalone fog nodes, the central cloud, and the gateway towards the Internet. The link set also consists of heterogeneous links, including high-speed fronthaul/backhaul fibre, digital subscriber lines, wireless and satellite channels, etc. Different links have different attributes including QoS, security, etc., and thus can carry different types of traffic. We consider two QoS parameters for each link: bandwidth and delay. For each link $e \in E$, we denote $b_e > 0$ as its capacity, and $d_e > 0$ as its transmission delay.

Note that we do not require a physically centralized controller architecture. A logically centralized controller (with a shared global view) is sufficient. A hierarchical control plane is also helpful in distributing the computational load in the mobile network. Distributed implementation of our proposed algorithm is out of the scope of this paper, and hence is omitted due to page limit.

B. Service Functions

The mobile network is deployed with heterogeneous network services, also called *service functions*, for in-network processing of user traffic. For example, signal processing may be virtualized and flexibly deployed on local fog nodes or in the cloud, due to recent advances in Software-Defined Radio (SDR) [16] and NFV. Different signal processing steps can be virtualized into independent functions, which can be deployed on different nodes. As modern RANs employ heterogeneous radio access technologies (RATs), different RATs may require different types and sequences of service functions for signal processing. On the other hand, the network can offer other network services for enhanced security and performance on the network edge [12], including firewall, intrusion detection, load balancer, etc. Utilizing NFV, these network services can also be implemented as virtualized software components, and flexibly deployed on fog nodes. Deployment of service functions at the network edge benefits from its low latency and high location flexibility, compared to the traditional cloud-based deployment.

Formally, we use $M = \{m_1, \cdots, m_{|M|}\}$ to denote the set of service functions provided by all nodes in the RAN. Each service function may have multiple instances, deployed at different locations. We use $V_m \subseteq V$ to denote the set of nodes that are deployed with service function $m \in M$, and $M_v \subseteq M$ as the set of service functions deployed on node $v \in V$. For each service function $m \in M_v$ deployed on node v, we are concerned with two attributes: $b_{v,m} > 0$ as its processing capacity, and $d_{v,m} > 0$ as its processing delay.

We assume available service functions have already been deployed in the network. Service function placement is out of the scope of this paper, and will be studied in future work.

C. Traffic Model

Traffic is aggregated and classified based on its access/exit points, QoS requirements, service function chain (*service chain* for short), types of traffic, and reliability requirement. In mobile networks, two most important QoS attributes are bandwidth and transmission delay. Denote all traffic classes (TCs) in the network as $\mathcal{C} = \{C_1, \cdots, C_{|\mathcal{C}|}\}$. Each TC is denoted as a 7-tuple $C_j = (s_j, t_j, B_j, D_j, \Pi_j, \mathcal{T}_j, r_j)$, where $s_j, t_j \in V$ denote the access and exit nodes respectively, $B_j > 0$ denotes the bandwidth demand, $D_j > 0$ denotes the maximum delay bound, Π_j denotes its service chain, $\mathcal{T}_j$ denotes the per-stage traffic type for each traffic stage defined in the service chain, and $r_j > 0$ denotes the reliability requirement. Explanation of the reliability requirement is deferred to Section III-E.

Each TC's service chain is defined as a sequence of service functions, $\Pi_j = (\pi_1^j, \cdots, \pi_{\kappa_j}^j)$, where each $\pi_k^j \in M$ denotes a service function required by the TC. The *service function chaining* requirement specifies that each packet of the TC originates from s_j, passes through all required service functions in the order given in Π_j, and exits at t_j. We assume each service chain contains only distinct service functions.

The chaining requirement splits transmission of the TC into $(\kappa_j + 1)$ stages: $s_j \to \pi_1^j$, $\pi_k^j \to \pi_{k+1}^j$ for $k = 1, \ldots, \kappa_j - 1$, and $\pi_{\kappa_j}^j \to t_j$. Each stage of traffic may belong to a different traffic type, and can be carried on only a subset of links. For example, in Cloud-RANs, the uplink traffic enters the network as wireless radio signals; before signal processing, such traffic can only be transmitted along high-speed fronthaul fibre which supports the Common Public Radio Interface (CPRI). On the other hand, traffic already through some security functions can no longer be transmitted via potentially insecure links. These are expressed in $\mathcal{T}_j = \{T_1^j, \cdots, T_{\kappa_j+1}^j\}$, where $T_k^j \subseteq E$ is the subset of links that can carry the stage-k traffic of C_j.

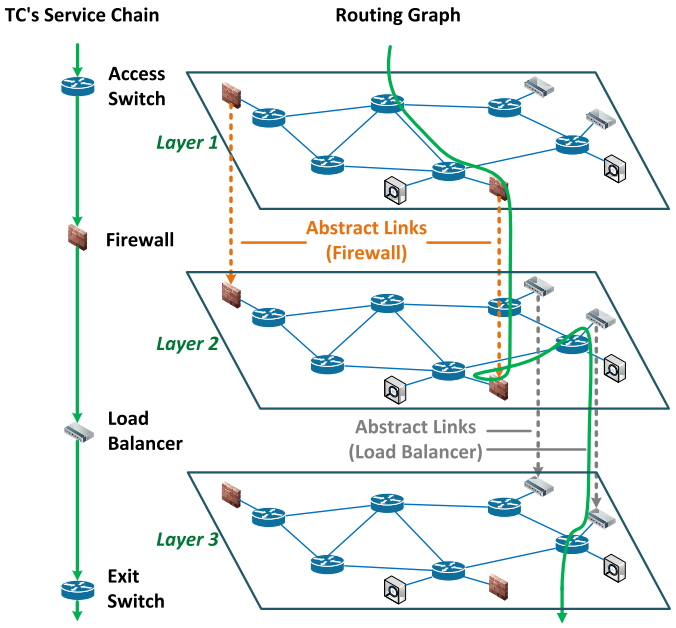


Fig. 1. A TC's service function chain and routing graph.

D. Feasible Routing Graph

We first define the feasible routing paths for TCs. Given network G and a TC C_j, a path p in G is *feasible* for C_j iff

1) p originates from s_j and ends at t_j;
2) p visits all service functions $\Pi_j = (\pi_1^j, \cdots, \pi_{\kappa_j}^j)$ in the given order; and
3) p has total transmission and processing delay within D_j.

To better establish the feasibility constraints of a routing path, we construct a per-TC routing graph $G_j^{ext} = (V_j^{ext}, E_j^{ext})$ from the original graph G, as shown in Fig. 1. G_j^{ext} has $(\kappa_j + 1)$ layers, each layer k corresponding to one copy of the subgraph of G that contains all nodes in V and all links in T_k^j; this enforces the traffic type constraints. We denote $v_k^j \in V_j^{ext}$ as the copy of node $v \in V$ in layer-k of G_j^{ext}, and $e_k^j \in E_j^{ext}$ as the copy of link $e \in E$ in layer-k of G_j^{ext}. We call link e the *prototype* of e_k^j, denoted by $proto(e_k^j)$; we also call e_k^j an *extended link* of e. Link e_k^j has the same transmission delay d_e as its prototype. Further, we establish abstract links between consecutive layers. For each service function $\pi_k^j \in \Pi_j$, we establish an abstract link from the copy of each node $v \in V_{\pi_k^j}$ in the k-th layer, to its copy

in the $(k+1)$-th layer. We denote this abstract link as $e_v^{j,k} = (v_k^j, v_{k+1}^j)$, and let it have delay d_{v,π_k^j} (processing delay of the instance). We use the pair (v, π_k^j) to denote the *prototype* of link $e_v^{j,k}$, also denoted by $proto(e_v^{j,k})$; $e_v^{j,k}$ is thus an *extended link* of prototype (v, π_k^j).

Let s_0^j be node s_j at layer 0 of G_j^{ext}, and $t_{\kappa_j}^j$ be t_j at layer κ_j of G_j^{ext}. We assume that each routing graph G_j^{ext} is $(s_0^j, t_{\kappa_j}^j)$-*connected*, meaning that there is a routing path from s_0^j to every node $v \in V_j^{ext}$, and there is a routing path from every node $v \in V_j^{ext}$ to $t_{\kappa_j}^j$. Nodes not satisfying this condition can be safely removed from the routing graph, as it does not contribute to the connectivity between s_0^j and $t_{\kappa_j}^j$.

For simplicity, we aggregate all TCs' routing graphs into a giant one, denoted as $G^{ext} = (V^{ext}, E^{ext})$, where $V^{ext} = \bigcup_{C_j \in C} V_j^{ext}$, and $E^{ext} = \bigcup_{C_j \in C} E_j^{ext}$. Each TC's subgraph G_j^{ext} is *maximally* $(s_0^j, t_{\kappa_j}^j)$-*connected* in G^{ext}, meaning that adding any node $v \notin V_j^{ext}$ makes it not $(s_0^j, t_{\kappa_j}^j)$-connected.

Given G^{ext}, the *feasible path set* of C_j is defined as all paths from s_0^j to $t_{\kappa_j}^j$, each with the sum of link delays no greater than D_j. Note that the processing delays of service function instances have already been accounted for in their extended links' delays. We use $\mathcal{P}_j$ to denote the feasible path set for C_j, and let $\mathcal{P} = \bigcup_{C_j \in C} \mathcal{P}_j$. Without loss of generality, we assume that each TC has a disjoint feasible path set $\mathcal{P}_j$.

The following notations are defined for simplicity. We denote $E_v = \{(v, m) \mid m \in M, v \in V_m\}$ as the set of all service function instances. $\mathcal{E} = E \cup E_v$, also called the *prototype* set, denotes the set of all original physical links and service function instances. We then use $E^{ext}(e)$ to denote all extended links of the same prototype $e \in \mathcal{E}$, *i.e.*, all links that share the same capacity bound b_e. We also use $\mathcal{E}(p)$ and $E_v(p) = \mathcal{E}(p) \cap E_v$ to denote the sets of all prototypes and only service function prototypes, respectively, used by path p. $\eta_p(e)$ denotes the number of times for which prototype e's extended links appear in path p. Note that $\eta_p(e) \leq 1$ if $e \in E_v$, as each service function chain contains only distinct service functions.

E. Reliability

Network service failures can downgrade or even halt the transmission of user traffic, which must be tackled to assure service continuity [24]. On the other hand, it has been revealed that the traditional "all-or-nothing" protection is actually an overkill for many data applications [30], due to the excessive resource consumption to provide such protection.

In this paper, we follow existing work and seek a "milder" way for improving service availability [30]. Instead of providing full recovery, we seek to bound the amount of throughput loss due to an arbitrary *single service function instance failure* (*single service failure* for short). Specifically, each TC $C_j \in C$ has a reliability parameter $r_j \in (0, B_j]$, denoting the *maximum tolerable throughput loss* that C_j may suffer from any single service failure; $r_j = B_j$ means no protection for C_j.

IV. Problem Statement

A. Problem Description and Formulation

In this paper, we study the traffic steering problem in mobile networks. Specifically, given the network G and the set of TCs C, the network operator's goal is to find a subset of feasible routing paths, as well as allocate bandwidth for each path, to fulfill the bandwidth demand of each TC, meanwhile satisfying both capacity bounds and reliability requirements.

Definition 1 (Bandwidth Allocation): Let $P \in \mathcal{P}$ be a subset of feasible routing paths. A *bandwidth allocation* of P is defined by a mapping $\mathcal{L} : P \mapsto \mathbb{R}^+$, where $\mathbb{R}^+$ is the positive real number set. We say that $\mathcal{L}$ is a *feasible bandwidth allocation* of P iff for each prototype $e \in \mathcal{E}$, $\sum_{p \in P: e \in \mathcal{E}(p)} \eta_e(p) \mathcal{L}(p) \leq b_e$. The *aggregate bandwidth* of P, denoted by $b(P)$, is the sum of bandwidth allocated on all paths in P:

$$b(P) = \sum_{p \in P} \mathcal{L}(p).$$

□

Definition 2 (QRTS): Given the network $G = (V, E)$, and the TC set C, the **QoS-aware and Reliable Traffic Steering** (QRTS) problem in mobile networks is to find a tuple $\Gamma = (P, \mathcal{L})$, where $P \subseteq \mathcal{P}$ is a subset of feasible routing paths, and $\mathcal{L}$ is a feasible bandwidth allocation of P, such that

1) let $P_j \subseteq P$ be the set of feasible routing paths of TC C_j in P, then $b(P_j) \geq B_j$ for each TC $C_j \in C$; and
2) during an arbitrary single service failure, at most r_j bandwidth is lost for each $C_j \in C$. □

B. Computational Complexity

Theorem 1: QRTS is $\mathcal{NP}$-complete. □

Proof: First, QRTS is in $\mathcal{NP}$, as checking all constraints takes polynomial time. Consider the special case of QRTS where there is only one TC with an empty service chain, no link excluded from E in $\mathcal{T}_j$, and no reliability requirement ($r_j = B_j$). Therefore, there is only one layer in its routing graph, which is the same as the original topology. In this case, we obtain the Multi-Path routing with Bandwidth and Delay constraints (MPBD) problem on a general graph, which has been proven $\mathcal{NP}$-complete in [22]. As a known $\mathcal{NP}$-complete problem is a special case of QRTS, the theorem follows. ■

C. Optimization Formulation

The QRTS problem is an $\mathcal{NP}$-complete decision problem. We further define the following optimization version of QRTS:

Definition 3 (OQRTS): Given the network $G = (V, E)$, and the TC set C, the **Optimal QoS-aware and Reliable Traffic Steering** (OQRTS) problem in mobile networks is to find a tuple $\Gamma = (P, \mathcal{L})$, where $P \subseteq \mathcal{P}$ is a subset of feasible routing paths, and $\mathcal{L}$ is a feasible bandwidth allocation of P, such that

1) let $P_j \subseteq P$ be the set of feasible routing paths of TC C_j in P, then $b(P_j) \geq \xi \cdot B_j$ for each TC $C_j \in C$;

2) during an arbitrary single service failure, at most r_j bandwidth is lost for each $C_j \in \mathcal{C}$; and
3) ξ is maximized. □

In the OQRTS problem, the network operator aims to maximize the *traffic scaling ratio* ξ, defined as the minimum ratio between the aggregate bandwidth and the demand of any TC, subject to the feasible path set, feasibility of bandwidth allocation and reliability constraints. If an OQRTS instance has an optimal solution of $\xi^* \geq 1$, the corresponding QRTS instance is feasible, and vice versa.

With $\mathcal{L}(p)$ defined as the per-path variable of bandwidth allocation, and ξ as the minimum scaling ratio, we formulate the OQRTS problem as the following linear program (LP):

$$\max \ \xi \tag{1a}$$

$$\text{s.t.} \sum_{p\in\mathcal{P}_j} \mathcal{L}(p) \geq \xi B_j, \quad \forall C_j \in \mathcal{C} \tag{1b}$$

$$\sum_{p\in\mathcal{P}:e\in\mathcal{E}(p)} \eta_p(e)\mathcal{L}(p) \leq b_e, \quad \forall e \in \mathcal{E} \tag{1c}$$

$$\sum_{p\in\mathcal{P}_j:e\in E_v(p)} \mathcal{L}(p) \leq r_j, \quad \forall C_j \in \mathcal{C}, e \in E_v \tag{1d}$$

$$\mathcal{L}(p), \xi \geq 0. \ \ \forall p \in \mathcal{P}$$

Explanation: Objective (1a) is to maximize the traffic scaling ratio ξ. Constraint (1b) defines the scaling ratio for each TC: $(\sum_{p\in\mathcal{P}_j} \mathcal{L}(p))/B_j \geq \xi$. Constraint (1c) enforces per-prototype capacities. Constraint (1d) enforces the reliability for each TC C_j. Specifically, for each service instance $e \in E_v$, the amount of traffic of C_j through e must not exceed the maximum tolerable throughput loss, denoted by r_j. By this constraint, any single function instance failure will affect at most r_j bandwidth, hence satisfying the reliability requirements.

While the above formulation is a linear program (LP), it can have an exponential number of variables due to the potentially exponential number of feasible routing paths in a given graph. This prevents solving the problem using standard LP techniques. Note that since the decision problem QRTS is $\mathcal{NP}$-hard, so is the optimization problem OQRTS. In the next section, we propose our approximation algorithm for OQRTS.

V. Fully Polynomial-Time Approximation Scheme

In this section, we design an FPTAS for the OQRTS problem. Since the problem is $\mathcal{NP}$-hard, an FPTAS is the best algorithm one can hope for, unless $\mathcal{P} = \mathcal{NP}$.

Definition 4 (FPTAS): Given a maximization problem Ω, an algorithm $\mathcal{A}$ is said to be a Fully-Polynomial Time Approximation Scheme (FPTAS) for Ω, iff for any instance of Ω with optimal objective value ζ^*, given any $\omega \in (0, 1)$, $\mathcal{A}$ can produce a feasible solution with objective value $\zeta \geq (1 - \omega) \cdot \zeta^*$, within time polynomial to both the input size and $1/\omega$. □

A. Dual Analysis

We first write the dual program of (1). Define dual variable $z(j)$ for Constraint (1b) with each $C_j \in \mathcal{C}$, $l(e)$ for Constraint (1c) with each $e \in \mathcal{E}$, and $\varphi(j, e)$ for Constraint (1d) with each $C_j \in \mathcal{C}$ and $e \in E_v$, the dual program is as follow:

$$\min \sum_{e\in\mathcal{E}} b_e l(e) + \sum_{C_j\in\mathcal{C}} \sum_{e\in E_v} r_j \varphi(j, e) \tag{2a}$$

$$\text{s.t.} \sum_{e\in\mathcal{E}(p)} \eta_p(e) l(e) + \sum_{e\in E_v(p)} \varphi(j, e) \geq z(j), \quad \forall C_j \in \mathcal{C}, \ p \in \mathcal{P}_j \tag{2b}$$

$$\sum_{C_j\in\mathcal{C}} B_j z(j) \geq 1, \tag{2c}$$

$$z(j), l(e), \varphi(j, e) \geq 0. \ \forall C_j \in \mathcal{C}, \ e \in \mathcal{E}$$

Explanation: Objective (2a) accounts for the constants in Constraints (1c) and (1d). Constraint (2b) is the dual constraint for primal variable $\mathcal{L}(p)$. Constraint (2c) is the dual constraint for primal variable ξ. For simplicity of notations, although $\varphi(j, e)$ is only defined for each $C_j \in \mathcal{C}$ and $e \in E_v$, we extend its definition to include $C_j \in \mathcal{C}$ and any $e \in \mathcal{E}$, and explicitly let $\varphi(j, e) = 0$ for $e \in \mathcal{E} \setminus E_v$.

Based on an observation similar to the one in [7], we have the following two lemmas:

Lemma 1: At any optimal solution of Program (2), Constraint (2c) is *binding*, *i.e.*, equality (rather than strict inequality) holds. □

Lemma 2: At any optimal solution of Program (2), there exists at least one path $p \in \mathcal{P}_j$ for any $C_j \in \mathcal{C}$, such that Constraint (2b) with C_j and p is *binding*. □

Proof: Assume that at an optimal solution, Constraint (2b) for any TC $C_j \in \mathcal{C}$ and path $p \in \mathcal{P}_j$ is not binding. It is obvious that we can reduce the value of any $l(e)$ or $\varphi(j, e)$ that has a positive value, by an arbitrarily small amount. The resulted solution is still feasible, but has a strictly smaller objective value than the optimal solution, leading to a contradiction. To prove Lemma 1, observe that if Constraint (2c) is not binding, then we can reduce the value of $z(j)$ for all $C_j \in \mathcal{C}$ by an arbitrarily small amount, which will make every Constraint (2b) to be unbinding, leading to the same contradiction. To prove Lemma 2, assume that there exists $C_j \in \mathcal{C}$ such that Constraint (2b) is not binding for any $p \in \mathcal{P}_j$, then we can increase the value of $z(j)$ by an arbitrarily small amount, which will make Constraint (2c) unbinding. This leads to the same contradiction as above. Hence both lemmas follow. ■

Based on Lemma 2, it is now clear that at any optimal solution, $z(j) = \min_{p\in\mathcal{P}_j}\{\sum_{e\in\mathcal{E}(p)} \eta_p(e) l(e) + \sum_{e\in E_v(p)} \varphi(j, e)\}$. In other words, $z(j)$ is equal to the shortest path length in $\mathcal{P}_j$ regarding the per-link length function $\varsigma(\varepsilon) = l(\varepsilon) + \varphi(j, \varepsilon)$ for $\varepsilon \in E^{ext}$, where $l(\varepsilon) = l(proto(\varepsilon))$, and $\varphi(j, \varepsilon) = \varphi(j, proto(\varepsilon))$, respectively.

Lemmas 1 and 2 help us refine the dual program into a more concise form, removing variables $z(j)$. Define $D(l, \varphi) = \sum_{e\in\mathcal{E}} b_e l(e) + \sum_{C_j\in\mathcal{C}} \sum_{e\in E_v} r_j \varphi(j, e)$ (the dual objective function), and $\alpha(l, \varphi) = \sum_{C_j\in\mathcal{C}} B_j \delta_j(l, \varphi)$, where $\delta_j(l, \varphi) = \min_{p\in\mathcal{P}_j}\{\sum_{\varepsilon\in p} \varsigma(\varepsilon)\}$ is the shortest path length in $\mathcal{P}_j$ under length function ς. The dual problem is equivalent to finding length functions l and φ that minimize $D(l, \varphi)/\alpha(l, \varphi)$:

$$\min_{l,\varphi\geq 0} \ \frac{D(l, \varphi)}{\alpha(l, \varphi)}. \tag{3}$$

B. Primal-Dual Algorithm

Our approximation scheme is based on a similar design as in [6], [7]. The intuitive is to greedily push flow along the dual-shortest feasible path for each TC, meanwhile updating the lengths such that the length of each prototype increases exponentially in the amount of its constraint violation. After a number of rounds, the flow is distributed approximately evenly in the network. By scaling the final flow with the bounded link lengths, we obtain a feasible solution that approximates the optimal. The algorithm is shown in Algorithm 1.

Algorithm 1 Primal-Dual Algorithm for OQRTS

Input: Topology G, TCs $\mathcal{C}$, tolerance ω
Output: Scaling ratio ξ, path sets $\{P_j\}$, bandwidth $\mathcal{L}$

1 Initialize $l(e) \leftarrow \gamma / b_e$ for $\forall e \in \mathcal{E}$;
2 Initialize $\varphi(j,e) \leftarrow \gamma / r_j$ for $\forall C_j \in \mathcal{C}, e \in E_v$, and $\varphi(j,e) \leftarrow 0$ for $\forall C_j \in \mathcal{C}, e \in \mathcal{E} \setminus E_v$;
3 Initialize $P_j \leftarrow \emptyset$, $\mathcal{L} \leftarrow \emptyset$;
4 Construct the routing graph G^{ext}, and let $l(\varepsilon) \leftarrow l(e)$, $\varphi(j,\varepsilon) \leftarrow \varphi(j,e)$ and $c_\varepsilon = c_e$ for $\forall \varepsilon \in E^{ext}, e = proto(\varepsilon)$;
5 $\rho \leftarrow 0$;
6 **while** $D(l,\varphi) < 1$ **do**
7 $\rho \leftarrow \rho + 1$;
8 **for** *each TC* $C_j \in \mathcal{C}$ **do**
9 $B'_j \leftarrow B_j$;
10 **while** $B'_j > 0$ **do**
11 $\tilde{p} \leftarrow \arg\min_{p \in \mathcal{P}_j} \{\sum_{\varepsilon \in p} \varsigma(\varepsilon)\}$;
12 $\phi \leftarrow \min\{B'_j, \min_{e \in \mathcal{E}(\tilde{p})} \{\frac{b_e}{\eta_{\tilde{p}}(e)}\}, r_j\}$;
13 $P_j \leftarrow P_j \cup \{\tilde{p}\}$;
14 $\mathcal{L}(\tilde{p}) \leftarrow \mathcal{L}(\tilde{p}) + \phi$, $B'_j \leftarrow B'_j - \phi$;
15 $l(e) \leftarrow l(e)(1 + \epsilon \cdot \frac{\phi \eta_{\tilde{p}}(e)}{b_e})$ $\forall e \in \mathcal{E}(\tilde{p})$, and $l(\varepsilon) \leftarrow l(e)$ for $\forall \varepsilon \in E^{ext}(e)$;
16 $\varphi(j,e) \leftarrow \varphi(j,e)(1 + \epsilon \frac{\phi}{r_j})$ $\forall e \in E_v(\tilde{p})$, and $\varphi(j,\varepsilon) \leftarrow \varphi(j,e)$ for $\forall \varepsilon \in E^{ext}(e)$;
17 **end**
18 **end**
19 **end**
20 $\xi \leftarrow (\rho - 1)/\log_{1+\epsilon} 1/\gamma$;
21 $\mathcal{L}(p) \leftarrow \mathcal{L}(p)/\log_{1+\epsilon} 1/\gamma$ for $\forall C_j \in \mathcal{C}, p \in P_j$;
22 **return** $(\xi, \{P_j\}, \mathcal{L})$.

Lines 1–2 of Algorithm 1 initialize the length of each prototype, where γ is a value to be determined in Section V-D. P_j and $\mathcal{L}$ denote the paths used by C_j and the bandwidth allocation over all paths respectively, both initialized to empty. After constructing the routing graph, we initialize the lengths of all extended links the same as their prototypes. The algorithm proceeds in phases (Lines 6–19), each phase consisting of $|\mathcal{C}|$ iterations (Lines 8–18). In the j-th iteration in each phase, the algorithm pushes B_j units of flow for TC T_j, which is done in steps (Lines 10–17). In each step, the algorithm finds the shortest feasible path $\tilde{p}$ for TC T_j under length function ς, and pushes ϕ units of flow through $\tilde{p}$, where ϕ is defined by the residual flow demand B'_j, the bottleneck capacity $\min_{e \in \mathcal{E}(\tilde{p})}\{b_e/\eta_{\tilde{p}}(e)\}$, and the reliability requirement r_j, whichever is the smallest. Since $\tilde{p}$ may pass through the same prototype for multiple times, the capacity b_e of each $e \in \mathcal{E}(p)$ is divided by $\eta_{\tilde{p}}(e)$, the number of times that it appears in $\tilde{p}$. After updating P_j, $\mathcal{L}(p)$ and B'_j, the algorithm updates the per-prototype lengths $l(e)$ and the per-TC per-function instance lengths $\varphi(j,e)$, in Line 15–16. The value of ϵ is also to be determined in Section V-D. The lengths of all extended links in the routing graph are then updated to reflect the change of lengths of their prototypes. Note that the resulting flow may exceed the capacity of each prototype. However, as we will show in Section V-D, scaling the flow on each link by $\log_{1+\epsilon} 1/\gamma$ yields a feasible solution.

C. Approximating Shortest Feasible Paths

Algorithm 1 relies on finding the shortest feasible path for each TC under length function ς. However, since the feasible path set of each TC only contains delay-bounded paths, this task is non-trivial. In fact, finding the shortest delay-bounded path is known as the Delay Constrained Least Cost path (DCLC) problem, which is also $\mathcal{NP}$-hard [28]. Nevertheless, there exist FPTASs for the DCLC problem, which output a path within $(1+\omega')$ of the shortest delay-bounded path [19], [28]. In the next subsection, we will show that for the purpose of our algorithm, it is sufficient to find a $(1+\omega')$-approximate ς-shortest path with strictly bounded delay.

D. Algorithm Analysis

Theorem 2: Given G, $\mathcal{C}$ and ω, let $\omega' = \epsilon = \frac{\omega}{4}$, and $\gamma = \left(\frac{1-(1+\omega')\epsilon}{|\mathcal{E}|+|E_v||\mathcal{C}|}\right)^{\frac{1+\epsilon(1+\omega')}{\epsilon\ (1+\omega')}}$, then Algorithm 1 outputs a feasible solution that is within $(1-\omega)$ times of the optimal solution, if the dual optimal objective value $\Delta \geq 1$. □

Proof: We prove by bounding the primal-dual ratio for the solutions derived in the algorithm. The basic idea is that the primal value increases linearly with the flow pushed in each phase, but each link's dual lengths increase exponentially with the flow through it. After a polynomial number of phases, the primal-dual ratio is then within the desired bound.

We first define some notations. For any symbol υ (including $l, \varphi, \phi, \tilde{p}$), we use $\upsilon^s_{\rho,\tau}$ to denote its value after phase-ρ, iteration-τ, and step-s of the algorithm, $\upsilon_{\rho,\tau}$ to denote its value after phase-ρ and iteration-τ, and υ_ρ to denote its value after phase-ρ. For symbols in the form of $\upsilon(l,\varphi)$ (including D, α, δ_j), we use $\upsilon^s_{\rho,\tau}$ to denote $\upsilon(l^s_{\rho,\tau}, \varphi^s_{\rho,\tau})$, and similarly $\upsilon_{\rho,\tau}$ and υ_ρ. We then have

$$\begin{aligned} D^s_{\rho,\tau} &\leq \sum_{e \in \mathcal{E}} b_e l^{s-1}_{\rho,\tau}(e) + \sum_{C_j \in C} \sum_{e \in E_v} r_j \varphi^{s-1}_{\rho,\tau}(j,e) \\ &\quad + \epsilon \phi^s_{\rho,\tau}(1+\omega') \cdot \delta^{s-1}_{j,\rho,\tau} \\ &\leq D^{s-1}_{\rho,\tau} + \epsilon \phi^s_{\rho,\tau}(1+\omega') \cdot \delta^s_{j,\rho,\tau}, \end{aligned}$$

where $j = \tau$ due to that in iteration-τ of each phase we only consider TC $C_j = C_\tau$; the first inequality is because path $\tilde{p}$ found in each step is a $(1+\omega')$-approximation of the shortest

feasible routing path; the second inequality is due to that shortest feasible path length is monotonically non-decreasing. Summing up the flow pushed in all steps of iteration-τ where we push B_j flow in total, we have

$$D_{\rho,\tau} \leq D_{\rho,\tau-1} + \epsilon B_j \ (1+\omega') \cdot \delta_{j,\rho,\tau},$$

and hence

$$\begin{aligned} D_\rho &\leq D_{\rho-1} + \epsilon \ (1+\omega') \sum\nolimits_{j=1}^{|\mathcal{C}|} B_j \cdot \delta_{j,\rho} \\ &\leq D_{\rho-1} + \epsilon \ (1+\omega')\alpha_\rho. \end{aligned}$$

Let the optimal dual solution be $\Delta = \min\limits_{l,\varphi \geq 0} \{\frac{D(l,\varphi)}{\alpha(l,\varphi)}\}$, we know that $\frac{D_\rho}{\alpha_\rho} \geq \Delta$. Since we assume that $\Delta \geq 1$, we have

$$\begin{aligned} D_\rho &\leq \frac{D_{\rho-1}}{1 - \frac{\epsilon \ (1+\omega')}{\Delta}} \leq \frac{D_0}{\left(1 - \frac{\epsilon \ (1+\omega')}{\Delta}\right)^\rho} \\ &\leq \frac{D_0}{1 - \epsilon \ (1+\omega')} e^{\frac{(\rho-1)\epsilon \ (1+\omega')}{\Delta(1-\epsilon(1+\omega'))}}, \end{aligned}$$

where the initial objective $D_0 = (|\mathcal{E}| + |E_v||\mathcal{C}|)\gamma$ due to the initial value of $l(e)$ and $\phi(j,e)$. The last inequality is due to $(1+x) \leq e^x$, where $x = \frac{\epsilon(1+\omega')}{\Delta - \epsilon(1+\omega')}$ in the inequality.

Now, assume the algorithm stops at phase ρ^*, hence $D_{\rho^*} \geq 1$ yet $D_{\rho^*-1} < 1$. Taking it into the above inequality, we have

$$\frac{\Delta}{(\rho^* - 1)} \leq \frac{\epsilon \ (1+\omega')}{(1 - \epsilon \ (1+\omega')) \ln \frac{1 - \epsilon \ (1+\omega')}{(|\mathcal{E}| + |E_v||\mathcal{C}|)\gamma}}.$$

On the other hand, by the way we update lengths $l(e)$ and $\varphi(j,e)$ at Lines 15–16, each dual variable has its value increased by at least $(1+\epsilon)$ times when the corresponding primal constraint is *filled* for once, *i.e.*, when the flow through prototype $e \in \mathcal{E}$ increases by b_e, or when the flow for TC C_j through a function instance $e \in E_v$ increases by r_j, respectively. Since $D_{\rho^*-1} < 1$, we have $l_{\rho^*-1}(e) < 1/b_e$ and $\varphi_{\rho^*-1}(j,e) < 1/r_j$. Therefore, the flow after phase $(\rho^* - 1)$ scaled by $1/\log_{1+\epsilon} 1/\gamma$ is strictly feasible, which means the final scaling ratio $\xi = (\rho^* - 1)/\log_{1+\epsilon} 1/\gamma$ is feasible. The primal-dual ratio is then bounded by

$$\frac{\xi}{\Delta} \geq \frac{(1 - \epsilon \ (1+\omega')) \cdot \ln \frac{1 - \epsilon \ (1+\omega')}{(|\mathcal{E}| + |E_v||\mathcal{C}|)\gamma}}{\epsilon \ (1+\omega') \cdot \log_{1+\epsilon} \frac{1}{\gamma}}.$$

Given the selection of ϵ, ω' and γ, we have $\frac{\xi}{\Delta} \geq 1 - \omega$. ■

For time complexity, we define $O^*(f) = O(f \log^{O(1)}(\mathbb{L}))$, where f is a function of the input size, and $\mathbb{L}$ is the number of values in the input (independent of each value's magnitude).

Theorem 3: The worst-case time complexity of Algorithm 1 is $O^*(\frac{\Delta}{\omega^3}|V||\mathcal{E}|(|\mathcal{E}| + |E_v||\mathcal{C}|)\kappa_{\max}^2)$, where $\kappa_{\max} = \max_j\{\kappa_j\}$ is the maximum service chain length of any TC. □

Proof: As above, we have $\frac{\xi^*}{\Delta} > \frac{\xi}{\Delta} \geq \frac{(\rho^*-1)}{\Delta \log_{1+\epsilon} 1/\gamma}$. By strong duality of linear programming, we have $\frac{\xi^*}{\Delta} = 1$. Therefore, the number of phases is bounded by $\rho^* \leq \lceil \Delta \log_{1+\epsilon} 1/\gamma \rceil$. The number of iterations is $|\mathcal{C}|$ times the number of phases. In each iteration, every but the last step increases the length of at least one $l(e)$ or $\phi(j,e)$ by $(1+\epsilon)$ times, hence the number of steps exceeds the number of iterations by at most $(|\mathcal{E}| + |E_v||\mathcal{C}|)\log_{1+\epsilon} \frac{1+\epsilon}{\gamma}$. Thus totally there are $O^*(\frac{\Delta}{\omega^2}(|\mathcal{C}| + |\mathcal{E}| + |E_v||\mathcal{C}|)) = O^*(\frac{\Delta}{\omega^2}(|\mathcal{E}| + |E_v||\mathcal{C}|))$ steps by the choices of ϵ, ω' and γ. Each step incurs one approximate shortest feasible path computation, which by Xue *et al.* [28] is computed in $O(|V_j^{ext}||E_j^{ext}|(\frac{1}{\omega'} + \log\log\log|V_j^{ext}|))$ time in each TC's routing graph G_j^{ext}. Both the node set and the link set are bounded by $|V_j^{ext}| = O(|V| \cdot \kappa_{\max})$ and $|E_j^{ext}| = O(|\mathcal{E}| \cdot \kappa_{\max})$ respectively. The theorem follows. ■

E. Feasibility and Demand Scaling

Theorem 2 relies on two facts: 1) the QRTS instance has a non-zero feasible solution, and 2) the optimal dual objective value $\Delta \geq 1$. On the other hand, the time complexity of Algorithm 1 is proportional to Δ, hence it should not be too large. In practice, these conditions may not be satisfied. Below we propose methods to tackle these issues.

1) Feasibility Checking: We first propose a method to check instance feasibility. Observe that as long as there is at least one feasible routing path $p \in \mathcal{P}_j$ for any TC $C_j \in \mathcal{C}$, the problem instance has a non-zero optimal objective value, as a multi-TC flow with $0 < \xi \leq \min_j \frac{r_j}{B_j}$ always exists. Therefore, as a feasibility check before running the algorithm, we first run a shortest path algorithm (regarding link delays) for each TC on the routing graph. If any TC C_j has the shortest path with delay larger than its delay bound D_j, we return that no feasible solution exists; otherwise, we proceed to the next step.

2) Demand Scaling: The next step is to ensure $\Delta \geq 1$; otherwise the algorithm may not achieve the desired bound. Note that we can scale the demands of all TCs by a common factor in order to scale ξ^*, and equivalently Δ. Hence if we can derive a lower bound on ξ^*, we can scale all demands such that $\Delta \geq 1$.

Following the method proposed by Garg and Könemann [7] and later on improved by Fleischer [6], we derive both a pair of lower and upper bounds on Δ, by finding the *feasible routing path with maximum per-prototype capacity*, denoted by p_j^*, for each TC C_j, using a binary search on the per-prototype capacity. Given a capacity threshold $b > 0$, the computation first prunes all prototypes with capacity less than b, and then find a delay-shortest path in the remaining graph; if the path delay is bounded by D_j, we increase the threshold b; otherwise we decrease b. As there are at most $|\mathcal{E}|$ distinct capacity values, the binary search takes $O(\log\ (|\mathcal{E}|))$ shortest path computations. The time complexity of finding paths for all TCs is $O(|\mathcal{C}|\log\ (|\mathcal{E}|)(|\mathcal{E}| + |V|\log\ (|V|\kappa_{max}))\kappa_{max})$ if the Dijkstra's algorithm is used for shortest path computations. When $|\mathcal{C}|$ is large, this can be further reduced by computing a single round of *all-pair shortest paths* on the pruned original graph for each of the binary search iterations, and then utilize the auxiliary graph in [3] to compute the paths for all TCs.

Let $b_j = \min_{e \in \mathcal{E}(\tilde{p}_j)}\{b_e\}$ be the bottleneck prototype capacity of path p_j^*. Since a flow can saturate all prototypes at its maximum, an upper bound on the single-TC flow value is given by $|\mathcal{E}|\min\{b_j, r_j\}$, taking into account the reliability requirement of each TC. Hence an upper bound of the optimal objective value Δ is given by $\overline{\Delta} = \min_{C_j \in \mathcal{C}}\{\bar{b}_j/B_j\}$. On the other hand, given the bottleneck prototype capacity b_j, a flow

that only contains path p_j^* and assigns $b_j/(\kappa_j+1)$ bandwidth to the path, is feasible for the TC itself, as a prototype can be used for at most κ_j+1 times. Since there are $|\mathcal{C}|$ TCs sharing the network, $\underline{b}_j = \min\{b_j/(\kappa_j+1)|\mathcal{C}|, r_j\}$ yields a lower bound on the throughput received by C_j. Hence a lower bound of Δ is given by $\underline{\Delta} = \min_{C_j \in \mathcal{C}}\{\underline{b}_j/B_j\}$.

Given these bounds, we can scale all TCs' demands by a factor of $\underline{\Delta}$ (thus Δ by a factor of $1/\underline{\Delta}$), which ensures that the scaled dual optimal objective value $\Delta \geq 1$. But now Δ can be as large as $\widetilde{\Delta} = \overline{\Delta}/\underline{\Delta}$. We then use the same technique as in [7]: if Algorithm 1 does not terminate after $\lceil 2\log_{1+\epsilon} 1/\gamma \rceil$ phases, then we know that $\Delta \geq 2$. In this case, we double all demands B_j (thus halving the optimal solution Δ), and re-run Algorithm 1. Given the upper bound on Δ, this takes $O(\log\ (\widetilde{\Delta}))$ rounds of demand scaling.

Combined with Theorem 3, we have our main theorem

Theorem 4: Algorithm 1 (combined with feasibility checking and demand scaling) produces a $(1-\omega)$-approximation in time $O^*(\frac{1}{\omega^3}|V||\mathcal{E}|(|\mathcal{E}|+|E_v||\mathcal{C}|)\kappa_{\max}^2 + |\mathcal{C}||\mathcal{E}|\kappa_{\max})$, and hence is an FPTAS for OQRTS. □

Proof: To compute the initial bounds, it takes $O(|\mathcal{C}|\log\ (|\mathcal{E}|)(|\mathcal{E}|+|V|\log|V|)\kappa_{max}\log\ (\kappa_{max}|V|))$ time. By the values of $\overline{\Delta}$ and $\underline{\Delta}$, we have $\widetilde{\Delta} \leq |\mathcal{E}||\mathcal{C}|\kappa_{max}$. Hence the number of demand scaling rounds is $O(\log\ (|\mathcal{E}||\mathcal{C}|\kappa_{max}))$. Each round consists of at most $\lceil 2\log_{1+\epsilon} 1/\gamma \rceil$ phases in Algorithm 1. By Theorem 3, each round runs in $O^*(\frac{1}{\omega^3}|V||\mathcal{E}|(|\mathcal{E}|+|E_v||\mathcal{C}|)\kappa_{\max}^2)$ time. Combining the above and omitting the logarithm terms, the final time complexity follows. ■

F. Extension to Multiple QoS Requirements

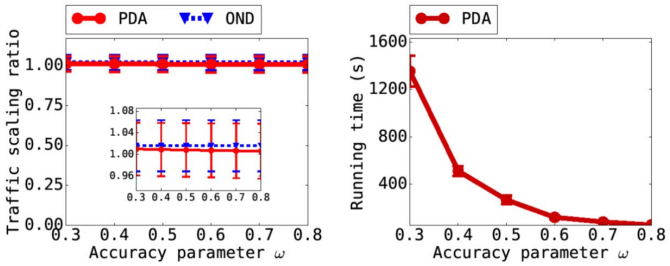
Our proposed model and algorithm can be extended to incorporate other QoS requirements than delay, such as jitter, packet drop rate, etc. In general, assume each TC considers up to Q additive QoS parameters. We can simply replace the DCLC FPTAS in Section V-C with a Multi-Constrained Path (MCP) FPTAS [28]. The resulting algorithm is able to enforce one QoS requirement strictly, while approximating the other $Q-1$ requirements within a factor of $(1+\omega')$, as shown in [28].

VI. Performance Evaluation

A. Experiment Settings

We implemented the following algorithms for comparison:

- PDA: Our primal-dual FPTAS (Algorithm 1). The accuracy parameter $\omega = 0.5$ by default. PDA has two variants, PDA-ND and PDA-NR, denoting the algorithms without delay and reliability requirements respectively.
- OND: An optimal algorithm for solving the OQRTS problem without consideration of TC delay constraints, obtained by solving an edge-flow multi-commodity flow LP. This yields an upper bound on the optimal solution.
- MPBDH: A flow-based heuristic which first computes a (delay-agnostic) maximum concurrent flow for all TCs, and then keeps finding feasible (delay-bounded) paths for each TC until no feasible path is left; extended from [22].
- SP: A baseline heuristic that decides the traffic scaling ratio based on shortest-path routing for each TC. As SP is a single-path routing algorithm, its solution can never exceed the minimum reliability ratio of any TC.



(a) Traffic scaling ratio vs. ω (b) Running time vs. ω

Fig. 2. Comparison with upper bound, with varying accuracy.

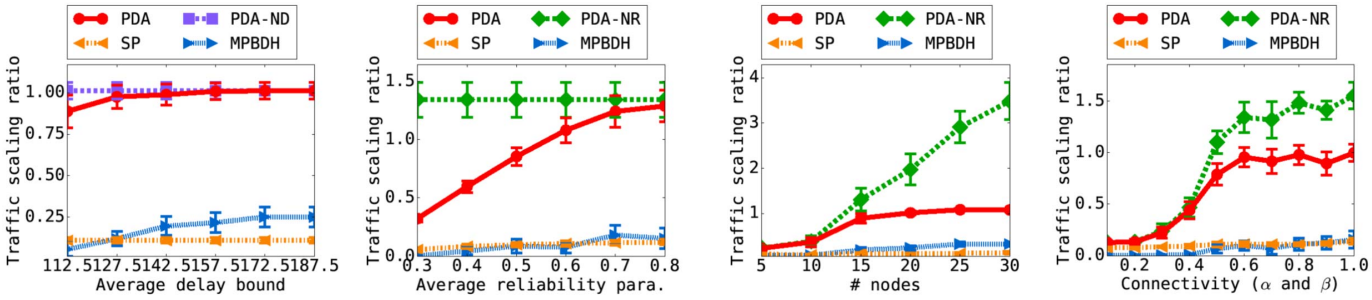
We randomly generated networks for evaluation. Topologies were generated based on the Waxman model [4]. By default, the network had 20 nodes. The network offered 10 types of service functions, each having 3 instances randomly deployed on nodes. Each instance had a random capacity within [50, 100] Mbps, and a random delay within [3, 30] ms. Connectivity parameters were set as $\alpha = \beta = 0.6$ in the Waxman model. Each link had a random capacity within [10, 100] Mbps, and a random delay within [1, 10] ms. In each experiment, we generated 20 TCs with random sources and destinations. Each TC had a random service chain with length within [1, 5], a bandwidth demand within [3, 30] Mbps, a delay bound within [125, 250] ms, and a reliability requirement within [0.35, 0.65] of its bandwidth demand. The above were the default parameters. In each set of experiments, we varied one control parameter for evaluation under different scenarios.

Two metrics were used to evaluate each algorithm. The **traffic scaling ratio** (objective function value) evaluates the algorithm's performance. The **average running time** evaluates the algorithm's overhead for producing the result.

We developed a C++-based simulator implementing all the above algorithms. For OND and MPBDH, we used the Gurobi optimizer [14] to solve the LPs. Each experiment was conducted on a Ubuntu Linux PC with Quad-Core 3.4GHz CPU and 16GB memory. Experiments were repeated for 20 times under the same settings to average out random noises. Each experiment was repeated for 20 times under the same setting, and results were averaged over all runs.

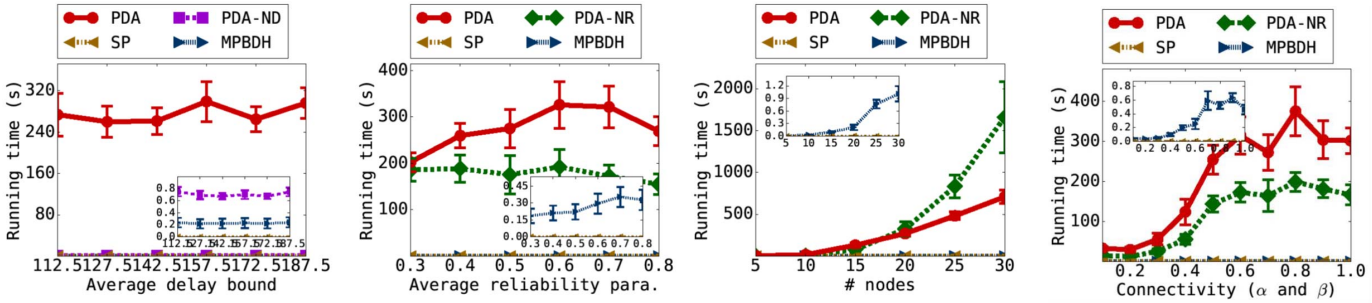
B. Evaluation Results

1) Comparison With Theoretical Upper Bound: Fig. 2 shows the comparison between PDA and OND. Note that the error bars show the 95% confidence intervals around the mean. Since OND is delay-agnostic, its optimal value yields an upper bound on the optimal value of OQRTS. As shown in Fig. 2(a), the solution produced by PDA is extremely close to the upper bound produced by OND, much higher than the theoretical guarantee $(1-\omega)$. Also, though the solution degrades with looser accuracy parameter ω, the degradation is minor. The observed optimality gap is within 1%. The running time of PDA, shown in Fig. 2(b), is decreasing polynomially to $1/\omega$. In conclusion, a loose accuracy parameter ω, such as no less than 0.5, is typically sufficient for practical use.



(a) Objective vs. delay bounds (b) Objective vs. reliability (c) Objective vs. # nodes (d) Objective vs. connectivity

Fig. 3. Traffic scaling ratio with varying delay bounds, reliability requirements, number of nodes, and connectivity parameters.



(a) Running time vs. delay bounds (b) Running time vs. reliability (c) Running time vs. # nodes (d) Running time vs. connectivity

Fig. 4. Running time with varying delay bounds, reliability requirements, number of nodes, and connectivity parameters.

2) Comparison With Baseline Heuristics: Figs. 3 and 4 show the comparison of PDA (including PDA-ND or PDA-NR) with the two baselines, MPBDH and SP, under various scenarios.

Figs. 3(a) and 4(a) show the objective values and running times respectively with varying TC delay bounds. With increasing delay bounds, both PDA and MPBDH achieve better traffic scaling ratio. SP has consistent performance with varying delay bounds as it only considers the shortest path. However, PDA outperforms both heuristics drastically, with an average improvement of 5.9× compared to MPBDH and 7.9× compared to SP. The enhanced performance indeed comes with increased time complexity, as shown in Fig. 4(a) The delay bounds have limited impact on time complexity in Fig. 4(a). Finally, comparing PDA and PDA-ND, the delay constraints lead to both degraded throughput and much larger time complexity, the latter due to the computation of a delay constrained least cost path instead of a simple shortest path.

Figs. 3(b) and 4(b) show the objective values and running times respectively with varying TC reliability parameter (the average ratio of maximum tolerable loss over bandwidth demand). Increased tolerable loss results in increased traffic scaling ratio in general, due to more bandwidth available to each TC's traffic at each service function instance. PDA again outperforms both heuristics in terms of throughput, with an average improvement of 6.1× and 8.5× compared to MPBDH and SP respectively. Comparing PDA and PDA-NR, the latter has a better throughput due to the relaxation of the reliability requirements, and a lower time complexity due to the less number of constraints (thus the number of dual variables) when reliability is not considered.

Figs. 3(c) and 4(c) show experiments with varying number of nodes in the network. Increasing number of nodes leads to increased traffic scaling ratios. However, after a certain threshold, the scaling ratio derived by PDA saturates. This is because the number of instances of each service function remains the same, and hence the the scaling ratios are constrained by the reliability requirements instead of the link capacities when the number of nodes become large. The throughput achieved by PDA surpasses MPBDH and SP significantly, with an average improvement of 3.7× and 8.2× compared to MPBDH and SP respectively. The running times increase with the number of nodes, due to the increased number of links.

Figs. 3(d) and 4(d) show experiments with varying network connectivity, which is controlled by parameters α and β in the Waxman model. Increased connectivity leads to increased throughput. Comparisons among algorithms are similar to the above. On average, PDA outperforms MPBDH and SP by 6.5× and 5.4×, respectively. MPBDH performs worse than SP, again due to the increased bandwidth on infeasible paths. The running times increase with network connectivity, due to the increase in both the problem size and the time for finding (approximate) shortest feasible paths.

To summarize, our findings are as follows:

- Our algorithm achieves near-optimal solutions even when the accuracy parameter is relatively loose. In general, the optimality gap is within 1%. Thus a loosely selected accuracy parameter is sufficient for most practical uses.
- Our algorithm outperforms both baseline heuristics (MPBDH and SP) significantly.
- The running time overhead of our algorithm is acceptable in practice, as network planning typically happens in much longer periods, for example, once per several hours.

VII. Conclusions

In this paper, we studied the QoS-aware and Reliable Traffic Steering problem for service function chaining in mobile

networks. We formulated the problem in a software-defined approach, considering various requirements of different classes of traffic, including service chaining, QoS, reliability, and type-of-transmission constraints. The problem, along with its optimization version, was proved to be $\mathcal{NP}$-hard. We then proposed an FPTAS for the optimization problem, which produces a $(1-\omega)$-approximate solution within time polynomial to the input size and $1/\omega$. We evaluated our algorithm through extensive simulation experiments, which validated that our algorithm has near-optimal performance, and achieves much better throughput than the baseline heuristics.

REFERENCES

[1] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions," in *Proc. IEEE CNSM*, Nov. 2015, pp. 50–56.

[2] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. ACM MCC*, 2012, pp. 13–16.

[3] Z. Cao, M. Kodialam, and T. V. Lakshman, "Traffic steering in software defined networks: Planning and online routing," in *Proc. ACM DCC*, 2014, pp. 65–70.

[4] M. Faloutsos, C. Faloutsos, and C. Faloutsos, "On power-law relationships of the Internet topology," in *Proc. ACM SIGCOMM*, 1999, pp. 251–262.

[5] J. Fan, Z. Ye, C. Guan, X. Gao, K. Ren, and C. Qiao, "GREP: Guaranteeing reliability with enhanced protection in NFV," in *Proc. ACM HotMiddlebox*, 2015, pp. 13–18.

[6] L. K. Fleischer, "Approximating fractional multicommodity flow independent of the number of commodities," *SIAM J. Discret. Math.*, vol. 13, no. 4, pp. 505–520, 2000.

[7] N. Garg and J. Konemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems," in *Proc. ACM FOCS*, 1998, pp. 300–309.

[8] A. Gember *et al.*, (May 2013). "Stratos: A network-aware orchestration layer for virtual middleboxes in clouds." [Online]. Available: https://arxiv.org/abs/1305.0209

[9] A. Gember-Jacobson *et al.*, "OpenNF: Enabling innovation in network function control," in *Proc. ACM SIGCOMM*, 2014, pp. 163–174.

[10] A. Gudipati, D. Perry, L. E. Li, S. Katti, and B. Labs, "SoftRAN: Software defined radio access network," in *Proc. ACM HotSDN*, 2013, pp. 25–30.

[11] L. Guo, J. Pang, and A. Walid, "Dynamic service function chaining in SDN-enabled networks with middleboxes," in *Proc. IEEE ICNP*, Nov. 2016, pp. 1–10.

[12] W. Haeffner, J. Napper, M. Stiemerling, D. Lopez, and J. Uttaro, *Service Function Chaining Use Cases in Mobile Networks*, document RFC7498 and RFC7665, 2017, pp. 1–26. [Online]. Available: https://tools.ietf.org/pdf/draft-ietf-sfc-use-case-mobility-07.pdf

[13] H. Hawilo, A. Shami, M. Mirahmadi, and R. Asal, "NFV: State of the art, challenges, and implementation in next generation mobile networks (vEPC)," *IEEE Netw.*, vol. 28, no. 6, pp. 18–26, Nov. 2014.

[14] *Gurobi Optimizer*. Accessed: Oct. 10, 2017. [Online]. Available: http://www.gurobi.com/products/gurobi-optimizer

[15] X. Jin, L. E. Li, L. Vanbever, and J. Rexford, "SoftCell: Scalable and flexible cellular core network architecture," in *Proc. ACM CoNEXT*, 2013, pp. 163–174.

[16] F. K. Jondral, "Software-defined radio: Basics and evolution to cognitive radio," *EURASIP J. Wireless Commun. Netw.*, vol. 2005, no. 3, pp. 275–283, 2005.

[17] M. Kablan, B. Caldwell, R. Han, H. Jamjoom, and E. Keller, "Stateless network functions," in *Proc. ACM HotMiddlebox*, 2015, pp. 49–54.

[18] Y. Kanizo, O. Rottenstreich, I. Segall, and J. Yallouz, "Optimizing virtual backup allocation for middleboxes," in *Proc. IEEE ICNP*, Jun. 2016, pp. 1–10.

[19] D. H. Lorenz and D. Raz, "A simple efficient approximation scheme for the restricted shortest path problem," *Oper. Res. Lett.*, vol. 28, no. 5, pp. 213–219, Jun. 2001.

[20] J. Martins *et al.*, "ClickOS and the art of network function virtualization," in *Proc. USENIX NSDI*, 2014, pp. 459–473.

[21] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. 4Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Survey Tuts.*, vol. 18, no. 1, pp. 236–262, 1st Quart., 2016.

[22] S. Misra, G. Xue, and D. Yang, "Polynomial time approximations for multi-path routing with bandwidth and delay constraints," in *Proc. IEEE INFOCOM*, Apr. 2009, pp. 558–566.

[23] M. Moradi, W. Wu, L. E. Li, and Z. M. Mao, "SoftMoW: Recursive and reconfigurable cellular WAN architecture," in *Proc. ACM CoNEXT*, 2014, pp. 377–390.

[24] R. Potharaju and N. Jain, "Demystifying the dark side of the middle: A field study of middlebox failures in datacenters," in *Proc. ACM IMC*, 2013, pp. 9–22.

[25] S. Rajagopalan, D. Williams, and H. Jamjoom, "Pico replication: A high availability framework for middleboxes," in *Proc. ACM SOCC*, 2013, p. 1.

[26] M. Rost and S. Schmid. (Apr. 2016). "Service chain and virtual network embeddings: Approximations using randomized rounding." [Online]. Available: https://arxiv.org/abs/1604.02180

[27] J. Sherry *et al.*, "Rollback-recovery for middleboxes," in *Proc. ACM SIGCOMM*, 2015, pp. 227–240.

[28] G. Xue, W. Zhang, J. Tang, and K. Thulasiraman, "Polynomial time approximation algorithms for multi-constrained QoS routing," *IEEE/ACM Trans. Netw.*, vol. 16, no. 3, pp. 656–669, Jun. 2008.

[29] Z. Ye, X. Cao, J. Wang, H. Yu, and C. Qiao, "Joint topology design and mapping of service function chains for efficient, scalable, and reliable network functions virtualization," *IEEE Netw.*, vol. 30, no. 3, pp. 81–87, May 2016.

[30] W. Zhang, J. Tang, C. Wang, and S. de Soysa, "Reliable adaptive multipath provisioning with bandwidth and differential delay constraints," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.

[31] Y. Zhang *et al.*, "StEERING: A software-defined networking for inline service chaining," in *Proc. IEEE ICNP*, Oct. 2013, pp. 1–10.

Ruozhou Yu (S'13) received the B.S. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2013. He is currently pursuing the Ph.D. degree with the School of Computing, Informatics, and Decision Systems Engineering, Arizona State University. His research interests include network optimization, network virtualization, software-defined networking, network function virtualization, and cloud and data center networks.

Guoliang Xue (M'96–SM'99–F'11) received the Ph.D. degree in computer science from the University of Minnesota in 1991. He is currently a Professor of computer science and engineering with Arizona State University. He has authored over 280 papers in these areas, many of which in top conferences, such as INFOCOM, MOBICOM, NDSS and top journals such as the IEEE/ACM TRANSACTIONS ON NETWORKING, the IEEE JSAC, and the IEEE TMC. His research interests span the areas of QoS provisioning, network security and privacy, crowdsourcing and network economics, RFID systems and Internet of Things, smart city, and smart grids. He was a keynote speaker at IEEE LCN'2011 and ICNC'2014. He was the TPC Co-Chair of the IEEE INFOCOM'2010 and the General Co-Chair of the IEEE CNS'2014. He has served on the TPC of many conferences, including the ACM CCS, the ACM MOBIHOC, the IEEE ICNP, and the IEEE INFOCOM. He served on the Editorial Board of the IEEE/ACM *Transactions on Networking*. He is the VP-Conferences of the IEEE Communications Society. He serves as an Area Editor of the IEEE *Transactions on Wireless Communications*, overseeing 13 editors in the Wireless Networking area.

Xiang Zhang (S'13) received the B.S. degree from the University of Science and Technology of China, Hefei, China, in 2012. He is currently pursuing the Ph.D. degree with the School of Computing, Informatics, and Decision Systems Engineering, Arizona State University. His research interests include network economics and game theory in crowdsourcing and cognitive radio networks.