Enhancing Mobile Networks With Software Defined Networking and Cloud Computing

Zizhong Cao, *Student Member, IEEE*, Shivendra S. Panwar, *Fellow, IEEE*, Murali Kodialam, *Member, IEEE*, and T. V. Lakshman, *Fellow, IEEE, ACM*

***Abstract*—In the past decade, mobile devices and applications have experienced an explosive growth, and users are expecting higher data rates and better quality services every year. In this paper, we propose several ideas to increase the functionality and capacity of wireless networks using software-defined networking (SDN) and cloud computing technologies. Connections between users and services in mobile networks typically have to pass through a required set of middleboxes. The complex routing is one of the major impetus for the SDN paradigm, which enables flexible policy-aware routing in the next generation mobile networks. In addition, the high costs of middleboxes and limited capabilities of mobile devices call for revolutionary virtualization technologies enabled by cloud computing. Based on these, we consider an online routing problem for mobile networks with SDN and cloud computing. In this problem, connection requests are given one at a time (as in a real mobile system), and the objective is to steer traffic flows to maximize the total amount of traffic accepted over time, subject to capacity, budget, policy, and quality of service constraints. A fast log-competitive approximation algorithm is developed based on time-dependent duals.**

***Index Terms*—Mobile network, routing, software-defined network, cloud computing, network function virtualization.**

I. Introduction

MOBILE data is continuing to grow exponentially, and is taking up a larger portion of total Internet traffic in recent years. As various portable mobile devices like smartphones, tablets, and netbooks become more prevalent, people are using mobile Internet more frequently and have shifted a large portion of their online activities, including web surfing, online banking, video streaming, social networking, and online gaming, from traditional cable Internet services to their mobile counterparts.

Such an exponential growth of mobile data, as well as the vast variety of online activities, imposes a heavy burden on mobile network service providers. Not only do these carriers need to upgrade their network capacities frequently, but they also need to provide differentiated services and meet the varied QoS requirements imposed by different mobile Internet applications. Mobile networks significantly differ from fixed

Manuscript received January 24, 2016; revised August 16, 2016; accepted November 14, 2016; approved by IEEE/ACM Transactions on Networking Editor S. Puthenpura. Date of publication January 16, 2017; date of current version June 14, 2017.

Z. Cao and S. S. Panwar are with the Tandon School of Engineering, New York University, Brooklyn, NY 11201 USA (e-mail: caozizhong@gmail.com; panwar@catt.poly.edu).

M. Kodialam and T. V. Lakshman are with Bell Laboratories, Nokia, Crawford Hill, NJ 07733 USA (e-mail: muralik@alcatel-lucent.com; t.v.lakshman@alcatel-lucent.com).

Digital Object Identifier 10.1109/TNET.2016.2638463

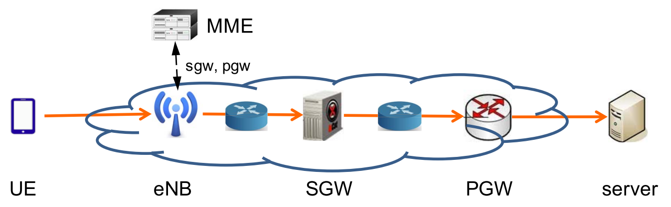


Fig. 1. Routing in 4G LTE mobile network.

Internet service provider (ISP) networks in that they adopt a different set of protocols for addressing/routing, and special network elements (NE) are used for various control and data plane tasks.

The latest 4G LTE network uses an all-IP evolved packet core (EPC) [1], but the network structure is still hidden from the public Internet, and mobile traffic from/to a mobile user equipment (UE) must pass through special NE including eNodeB, serving gateway (SGW), packet data network gateway (PGW), and certain middleboxes in a specific manner. Depending on the specific applications, typical middleboxes [2], [3] may include intrusion detection and prevention for enterprise customers, link monitoring and rate adaptation for video streaming, echo cancellation for voice-over-IP calls, data volume measurement for billing and charging, etc.

At present most NE are statically deployed in the network, and routing paths among these nodes are often pre-determined. According to 3GPP specifications [4], mobile data traffic in current 4G LTE networks is routed in a hop-by-hop manner, either according to fixed routing tables or with limited load balancing capabilities, as shown in Fig. 1. More specifically, a UE selects an eNodeB according to channel measurements, the eNodeB selects a Mobility Management Entity (MME) according to tracking area identification (TAI), and the MME selects a PGW according to access point name (APN) and an SGW according to TAI. The PGW is solely responsible for QoS. It maps traffic flows into different QoS classes, which are then translated into DiffServ priorities enforced in a per-hop manner. This strategy cannot fully utilize the networks to meet customers' needs due to its poor flexibility when facing asymmetric traffic fluctuations, and lacks fine-grained optimization of network resources and end-to-end traffic steering.

The presence of various NE and multiple instances is one of the significant reasons for the emergence of the software defined networking (SDN) paradigm. SDN [7] decouples the data and control planes, and routing decisions are made in a

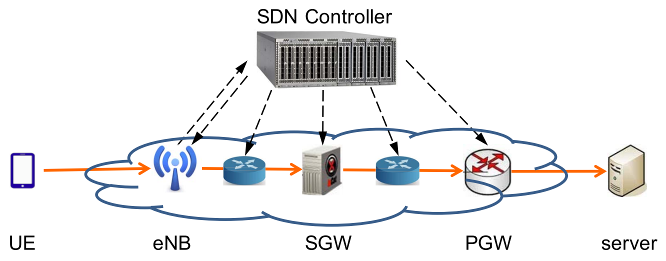


Fig. 2. Routing in a software-defined mobile network.

centralized manner rather than hop-by-hop. For each traffic flow, the controller not only selects the best NE, but can also specify the best route to take, as shown in Fig. 2. This gives SDN the ability to steer traffic in fine granularity to enforce policy-based routing, meet QoS requirements of various applications, and perform network-wide resource optimizations as well. Therefore, SDN becomes a promising candidate to enhance mobile networks [8].

Another emerging technology to enhance the performance of mobile networks is cloud computing. Instead of the scale-up approach adopted by traditional IT industry in building dedicated, stand-alone, high performance software and hardware systems, cloud computing follows a scale-out approach that utilizes a large pool of commodity servers as general-purpose computing resources and deliver configurable IT services on demand to various users. Cloud computing can be applied to several different aspects of mobile networking, such as network function virtualization (NFV) [9]–[11] and mobile cloud computing [12]–[14], and it is a promising complement to SDN.

In this paper, we address the unique challenges of traffic steering and service deployment in mobile networks, and adopt SDN and cloud computing as key enabling technologies to design and optimize next generation mobile networks. Part of this work is based on our previous workshop paper [15]. Our main contributions are:

1) We utilize an SDN framework to enable flexible policy-aware routing and centralized network optimization, and incorporate cloud computing to drive next generation mobile networks.
2) We formulate a mixed integer programming to maximize the total traffic accepted over time under capacity/budget constraints, and propose a competitive online routing algorithm to achieve policy-awareness and QoS guarantee.
3) We perform extensive numerical experiments to investigate the performance of our proposed routing algorithm under various network settings, and demonstrate its advantages over other prevalent heuristic algorithms.

II. Related Work

As key supporting components toward next generation mobile networks, routing and provisioning for mobile data traffic have attracted much attention in recent years. In [3] and [8], the authors call for rethinking of cellular core networks, and propose to use an SDN framework. It is envisioned that with SDN, mobile networks can allow for more flexible traffic steering, fine-grained monitoring, seamless mobility, distributed QoS and access control, etc. A SoftCell prototype has also been built to support these arguments. In [10], NFV and SDN are applied to the 4G LTE core gateways, and two redesigns of the these gateways are proposed: a virtualized one provisioned through NFV, and a decomposed one controlled by SDN. The authors also perform extensive measurements and solve function placement problems in the new framework. These works have greatly influenced this paper, but note that they mostly focus on the system architecture aspect but not the algorithmic aspect and thus have not considered flow scheduling and routing algorithms that work specifically with SDN and NFV.

In the meantime, middlebox implementation issues have also received increasing attention [2], [16]–[18]. It has been recognized [2], [16] that network planning has become more complicated with policy enforcing, and can be a potential bottleneck. Therefore, various load balancing strategies have been developed to distribute middlebox traffic across the networks. In [19], routing policies are enforced in an SDN framework using a non-deterministic finite automation with graph encoding. However, the authors do not provide a fast approximation scheme to tackle the NP-completeness, and traffic steering is performed offline. In [20], the authors study the problem of policy-aware application cloud embedding. The paper shows that several embedding problems are NP-complete, and deals with online embedding of flow security graphs (FSG) for each request. One caveat with this work is that the number of candidate paths must be polynomial, and it is assumed that once a request enters the system it is there permanently.

In contrast to these papers, we propose to incorporate SDN and cloud computing into mobile networks, and tackle the problem with both system architecture and algorithm designs. In addition, we consider more realistic mobile networks in which each flow can have an exponential number of candidate paths, stays for a finite duration, and is not available to the scheduler ahead of time. As will be shown later, we solve this more challenging problem by developing a fast online routing algorithm that is aware of the latency/congestion/budget and guarantees log-competitiveness with respect to the optimal offline solution.

III. Design Motivations

A. Expectations of the Next Generation 5G Network

Over the past few years, the next generation 5G mobile network has been gaining interest in both academia and industry. While the standardization of 5G and commercial deployments are still a few years away [5], there have been clearer expectations on what 5G should bring to users [6] with several proposals on air link, fronthaul, and backhaul technologies. Among various expectations on different aspects, we pick the two that are most perceived by users:

1) *Data Rate:* $1Gbps$ to tens of $Gbps$ peak data rate;
2) *Latency:* $1ms$ for extremely low latency use cases or at most $10ms$ for others.

Compared with the latest 4G LTE network, the next generation 5G network needs to sustain orders of magnitude higher data rate at orders of magnitude lower latency, which is very demanding and requires rethinking of the network architecture.

On the latency issue, we should first realize that a low latency in the order of $1ms$ is not always achievable due to physical constraints. For example, if two users are connecting directly between New York City and San Francisco, then the 2700 miles distance would cause no less than $14ms$ network latency at the speed of light. Fortunately, such an extreme case is not common, and in many cases a higher replication factor can be traded for a lower latency. For static content like images and videos, Content Delivery Networks (CDN) can replicate content to the edge of the network, which then are able to serve users in close network proximity at much lower latencies. For other dynamic web services, we can also create multiple web server instances and database replicas, and redirect users to the closest one. In the following, we shall focus on the throughput-latency tradeoff in resource replication, while in network planning there is also a latency-expense tradeoff since replication often results in higher migration and storage costs.

Consider a simplified scenario in which many users and r resources are uniformly placed on a planar area, then the average distance between each user and its closest resource is proportional to $1/\sqrt{r}$. Following this idea, in order to serve all users across the coasts of the United States within the $1ms$ latency bound, we would need at least 8 resource replicas along the east-west span. This number needs to be further multiplied by at least 5 if we take the south-north span into account as well, resulting in a total of 40 replicas. Even more would be needed if we consider other factors like network topologies and processing latencies. The same argument also applies to the gateways and middleboxes that need to be traversed by user traffic flows, because otherwise the scarcity of these NE may result in triangular routing and void the efforts we have made in replicating the resources.

In addition to the strict latency requirement, the high data rate requirement is also an important concern. The data rate of mobile traffic has experienced exponential growth over the past few years, and this trend shall continue to add more burdens to the infrastructure of mobile networks. Great efforts have been made and new technologies have been proposed to increase the network capacity commensurately. However, because network infrastructures are often statically deployed and cannot easily adapt to varying demands, traffic engineering and network scheduling are still key to high performance. On the other hand, when we replicate resources and deploy more NE to achieve a lower latency, the throughput (the amount of traffic delivered from source to destination in unit time) of the network can be boosted at the same time. As the user requests get served in closer proximity, the traffic flows will traverse through a smaller portion of routers and links, which effectively reduces the burden on the network load and increases the amount of traffic that can be sustained. For a given network with fixed topology and capacity, if we assume that all link utilizations are always kept unchanged, then the average throughput should be inversely proportional to the average latency, and thus is proportional to $\sqrt{r}$.

B. A Software Defined Mobile Network With Cloud Computing

As discussed above, the stringent latency constraints on 5G mobile networks call for extensive replications of resources and NE, which in turn add more burdens to network capacity and make traffic routing more complicated. Complex routing is one of the major impetus for the SDN paradigm, and the high costs of flexible resource/NE replications greatly benefit from cloud computing. Therefore, we propose an SDN framework with cloud computing to address the challenges of next generation mobile networking.

1) Fine-Grained Traffic Steering in a Software Defined Network: In an SDN, the control plane and the data plane are decoupled, which allows for centralized routing decisions and distributed packet forwarding. The control plane of the network is aggregated in an SDN controller which communicates with the routers using a standard interface like OpenFlow. Typically, a special request or the first packet is sent to the SDN controller for each new flow, and the following operations are done by the SDN controller:

1) *Policy Lookup:* The policy table at the controller determines the logical sequence of NE that the packets in the flow have to pass to satisfy the policy requirement.
2) *Flow Steering:* The SDN controller maps this logical sequence of NE into the physical network according to certain QoS requirements and optimization criteria.
3) *Route Installation:* Once this logical path is mapped into the physical path, the SDN controller makes changes in the forwarding table on all the routers in the path so that all packets in the flow are routed along the desired path.

A doubt may arise here that the scheduling time at the SDN controller and round-trip time to set up the routing path can become large as the network grows. This is true but we do not need to be overly pessimistic about this. On one hand, the control-plane latency does not necessarily add to the data-plane latency perceived by the end user. For those latency-sensitive applications like virtual reality, automated driving, and remote surgery, routing is performed at most once during connection initialization but does not affect the connection quality later on. For other less demanding applications like web browsing, a long-lived connection can be set up beforehand rather than dynamically allocated on demand. This idea is similar to the default vs. dedicated bearers in the current 4G LTE networks. On the other hand, the routing and scheduling tasks can be distributed to multiple SDN controllers placed at different locations, and thus come closer to the end users. This may significantly reduce the control-plane latency at the cost of looser consistency in network states and/or more sophisticated synchronization.

SDN brings about many opportunities to optimize current mobile networks [3], [21], [22]. More importantly, it gives us more freedom to rethink what future network architectures should look like and how routing should be performed to meet the increasing demands.

2) More Flexible and On-Demand Service Deployment With Cloud Computing: Network function virtualization (NFV) is a network architecture concept that proposes to virtualize

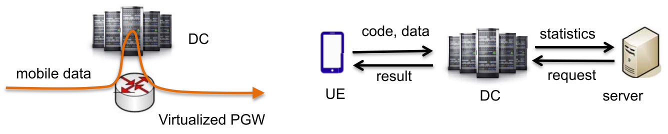


Fig. 3. Network function virtualization and mobile cloud computing.

application-specific NE functions on top of commodity servers and can leverage the cloud computing infrastructure, as shown in Fig. 3. It can boost the carrier capabilities and reduce the high capital and operational costs incurred by special-purpose NE hardware. Instead of spending large capital to purchase powerful proprietary hardware in prior, mobile network service providers can now shift to NFV with cloud computing and pay operational expenses over time. With cloud-based NFV, these special NE can be deployed on demand and spread throughout the network, which grants more flexibilities to mobile traffic scheduling. All these benefits make NFV a scalable and profitable solution for mobile networks.

Besides NFV, cloud computing has been applied to mobile UE as well, also known as mobile cloud computing (MCC). MCC can host proxies or clones of UE in large data centers or regional cloudlets, and allow mobile users to offload computation-intensive applications to the cloud. When carefully done, MCC could significantly enhance the user experience by boosting device capabilities and extending battery life at the same time without affecting users with network latencies. Full-featured MCC can also serve as a proxy for any UE to terminate connections, serve requests, or participate in data collections in the capacity of the UE.

Furthermore, cloud computing can also be used to scale the central controller in the SDN framework, which is computation-intensive and could be a bottleneck as well. Instead of using a single super-scheduler to perform all the resource scheduling and traffic engineering, we can distribute these tasks to many different servers if parallelism could be exploited from the routing algorithm.

However, note that resource virtualization through data centers should be taken into account when making allocation decisions, as virtualized resources have different costs and limitations from the dedicated ones we are familiar with.

3) Design Goals: Viewing the limitations of the latest 4G LTE network versus the requirements of the 5G mobile network, we propose an SDN architecture with cloud computing to enhance the network capabilities, as shown in Fig. 4. The main design goals are as follows:

1) Integrated resource selection and content routing. As content/computing resources become more scattered and dynamic, the traditional way of performing selection separately from routing will no longer suffice.
2) Flexible policy enforcement through SDN. Fine-grained control should be exercised to steer data flows through NE deployed throughout the network.
3) Providing truly end-to-end QoS guarantees. QoS optimization should be pushed to both ends of data flows rather than on a hop-by-hop basis.

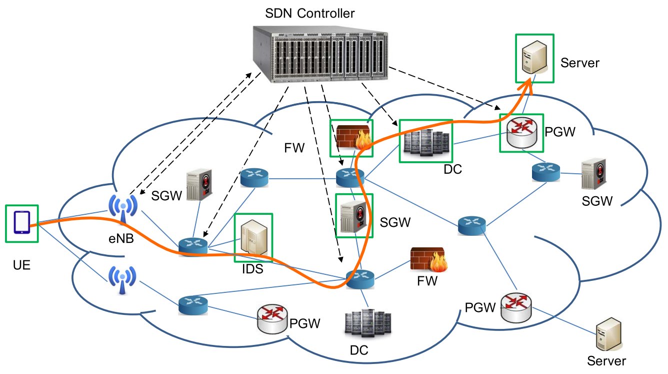


Fig. 4. Policy-aware routing in a software-defined mobile network.

4) Scattering of network functions. Light-weight NE should be widely deployed to avoid triangular routing, probably on-demand through NFV.
5) Incorporate MCC to enhance the UE capabilities using cloud infrastructures. Special considerations need to be made for allocation and routing.
6) Competitive network resource optimization. This has often been carried out in a distributed way and settled at sub-optimal heuristic solutions, yet it can be further explored in a centralized manner.
7) Coordination across mobile networks, either for data roaming across different carriers or when multiple radio access technologies (RAT) co-exist for the same carrier.

IV. Problem Formulation

We consider a generic mobile network represented by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes a set of n nodes making up the network, and $\mathcal{E}$ denotes a set of m directed links connecting the nodes, as shown in Fig. 4. The graph $\mathcal{G}$ is assumed to be mesh-connected, having multiple paths connecting each pair of nodes. Every link e is associated with a fixed capacity c_e and a unit operational cost b_{et} that may change over slotted time t. The node set $\mathcal{V}$ contains standard routers and special NE attached to routers, including gateways, middleboxes, data centers, etc. Note that we only care about those NE that are hosted inside the mobile network or within the reach of the SDN controller, and there may be multiple instances of each type of NE in the network. These NE can be either dedicated equipment with a fixed processing capacity, or dynamically provisioned through NFV from data centers at certain operational costs. Such node capacities c_v and unit costs b_{vt} are assumed to be pre-processed and transformed into link parameters mentioned above. This can be done by splitting each node v into two shadow nodes v and v', and connecting them with a single directed link $e = \overrightarrow{vv'}$ that binds with capacity $c_e = c_v$ and unit cost $b_{et} = b_{vt}$. Also assume that we have a budget B_t that restricts the total costs we could spend during any time slot t. This budget may change over time in response to varying traffic volume, electricity price, company strategy, and many other factors.

Each mobile traffic flow enters the network at its ingress node and has to visit a set of NE in a specified order before leaving the network at its egress node. Let s_d denote the ingress node for flow d, t_d denote the egress node, and k_d represent the number of NE that flow d needs to traverse. The *policy list* for flow d is defined as an ordered list $\{M_d[1], M_d[2], \ldots, M_d[k_d]\}$, with each $M_d[i]$ representing the i-th type of NE that flow d needs to visit in a sequential order. When MCC is enabled for a certain flow, its traffic should always route through the hosting data center, so this should also be included in the policy list. For each flow d, policy-aware routing aims at finding paths p that conform to its policy list, and we denote the set of all such paths by $\mathcal{P}_d$. A policy-aware path p is made up of $k_d + 1$ route segments $p^{(i)}$, where $p^{(1)}$ routes from s_d to $M_d[1]$, $p^{(i)}$ for $2 \leq i \leq k_d$ routes from $M_d[i-1]$ to $M_d[i]$, and $p^{(k_d+1)}$ routes from $M_d[k_d]$ to t_d.

Also, each traffic flow d is associated with a certain traffic demand. For mobile networks, we assume that each traffic demand can be specified using a fixed (or bounded) bandwidth h_d and a certain time duration $\tau_d \triangleq \tau_d^f - \tau_d^s + 1$, where τ_d^s represents the starting time slot, and τ_d^f represents the finishing time slot. For MCC-enabled flows, a single h_d may not accurately describe the traffic demand, because the bandwidth before and after passing the host can be different. This can happen if the MCC acts as a file server for external request, performs adaptive transcoding on incoming videos, or simply caches uplink/downlink data when the connection is unstable. Things can become even more complicated during mobile-to-mobile communications. Therefore, we add another set of weight parameters $\gamma_d^{(i)}$ for $1 \leq i \leq k_d + 1$ such that $\gamma_d^{(i)} h_d$ denotes the bandwidth on route segment $p^{(i)}$. Note that h_d is always chosen such that $\gamma_d^{(1)} = 1$.

In addition to routing policies and traffic demands, mobile traffic is also subject to certain QoS constraints. Typical QoS constraints [23] include bounded packet latency, limited packet loss, etc. In this paper, we mainly focus on the end-to-end packet latency constraints, while the packet loss limit can be enforced in a similar way. The end-to-end latency constraint W_d for each mobile traffic flow d can be different. In this paper, we assume that the network latencies are mainly caused by processing at NE nodes and transmission through physical links, and each node/link contributes a fixed amount of latency w_e regardless of the traffic load. Such an assumption is reasonable if we set aside a cushion bandwidth and balance/restrict the traffic load on each link to be strictly lower than a certain threshold, which is part of the goal of fine-granularity scheduling enabled by SDN. The more challenging problem with congestion delays will be studied in future work. Therefore, the sum of latencies on any QoS-guaranteed path p serving flow d cannot exceed W_d. In addition to the end-to-end latency constraint, sometimes we may need to enforce some other constraints as well. One example is the latency between a UE and its host when MCC is supported because the user experience of MCC is often latency-sensitive. In this case, the sum of w_e on the UE-to-MCC segment p_{mcc} of any QoS-guaranteed path p serving flow d cannot exceed W_d^{mcc}. Another example of this type is bounding the UE-to-SGW latency for more precise locality or less down time during handover, which can be modeled in a similar way. See Table I for a complete list of notations.

TABLE I

LIST OF NOTATIONS

Variables	Definitions
$\mathcal{G}$	directed graph composed of node set $\mathcal{V}$ and link set $\mathcal{E}$
n	size of node set $\mathcal{V}$
m	size of directed link set $\mathcal{E}$
c_e	capacity of link e
b_{et}	unit operational cost of link e at time t
s_d	ingress node for flow d
t_d	egress node for flow d
k_d	number of NE that flow d needs to visit in a sequential order
$M_d[i]$	the i-th type of NE that flow d needs to visit
$\mathcal{P}_d$	set of paths that conform to the policy list of flow d
h_d	bandwidth requirement of flow d
τ_d^s	starting time of flow d
τ_d^f	finishing time of flow d
$p^{(i)}$	i-th route segment of policy-conforming path p
$\gamma_d^{(i)}$	scaling ratio applied to the bandwidth requirement of $p^{(i)}$
W_d	end-to-end latency constraint of flow d
w_e	latency contributed by link e

V. Online Traffic Steering for Mobile Networks

A. Maximizing Total Accepted Traffic Over Time

A major objective of mobile traffic steering is to maximize the total amount of traffic accepted over time in a policy-aware manner without violating any node/link capacity, operational budget, or QoS constraints. In this section, we formulate a variant of the well-known maximum multi-commodity flow problem [24], and incorporate other techniques to meet policy awareness and QoS constraints.

We now formulate the problem as a mixed integer programming problem. We use a binary variable x_{dp} to indicate whether path p is chosen for flow d. When a request enters the system, it is either accepted for the entire duration and carried on a single path or is rejected.

$$\text{Maximize } \alpha \triangleq \sum_d \sum_{p \in \mathcal{P}_d} h_d \tau_d x_{dp} \tag{1a}$$

$$\text{Subject to } \sum_{p \in \mathcal{P}_d} x_{dp} \leq 1, \quad \forall d \tag{1b}$$

$$\sum_{d | t \in [\tau_d^s, \tau_d^f]} \sum_{p \in \mathcal{P}_d | e \in p} \sum_{i | e \in p^{(i)}} \gamma_d^{(i)} h_d x_{dp} \leq c_e, \quad \forall e, \forall t \tag{1c}$$

$$\sum_{d | t \in [\tau_d^s, \tau_d^f]} \sum_{p \in \mathcal{P}_d} \sum_i \sum_{e \in p^{(i)}} b_{et} \gamma_d^{(i)} h_d x_{dp} \leq B_t, \quad \forall t \tag{1d}$$

$$x_{dp} \sum_{e \in p} w_e \leq x_{dp} W_d, \quad \forall d, \forall p \in \mathcal{P}_d \tag{1e}$$

$$x_{dp} \in \{0, 1\}, \quad \forall d, \forall p \tag{1f}$$

Objective (1a) seeks to maximize the total accepted traffic over time. Note that the bandwidth of a single flow d may change across different route segments i due to MCC or other NE, while we are only interested in the component h_d that is directly perceivable from the UE, because users are often charged based on this portion of mobile data. Constraint (1b) ensures that each flow d is allocated at most once. Constraint (1c) enforces the link capacity constraints at all times. Since each link can be traversed by multiple route segments of multiple flows with different bandwidth requirements, we must aggregate all of them together for the overall link utilizations. Constraint (1d) restricts the operational costs to be lower than the budget we have at any time. Constraint (1e) enforces the end-to-end latency constraints on all selected paths. For paths that are not selected, these constraints are satisfied trivially due to $x_{dp} = 0$. Other QoS constraints can be added in a similar way on demand. Constraint (1f) defines the allocation to be unsplittable (all-or-nothing), which makes the formulation a mixed integer programming (MIP).

Solving the above MIP formulation directly would incur exponential complexity due to the large number of policy-aware candidate paths in $\mathcal{P}_d$, which is undesirable. Thus we follow a primal-dual approach to reduce the complexity.

In order to write the dual, we define three sets of dual variables: z_d corresponds with every demand fulfillment constraint (1b), l_{et} corresponds with every link capacity constraint (1c); and ϕ_t corresponds with every budget constraint (1d). Enforcing QoS constraints (1e) and finding policy-aware paths $\mathcal{P}_d$ will be handled separately.

$$\text{Minimize } \beta \triangleq \sum_d z_d + \sum_t \sum_e c_e l_{et} + \sum_t B_t \phi_t \tag{2a}$$

$$\text{Subject to } \sum_{t\in[\tau_d^s,\tau_d^f]} \sum_i \sum_{e\in p^{(i)}} \frac{\gamma_d^{(i)}(l_{et}+b_{et}\phi_t)}{\tau_d} + \frac{z_d}{h_d \tau_d} \geq 1,$$

$$\forall d, \forall p \in \mathcal{P}_d \tag{2b}$$

$$l_{et} \geq 0, \quad \forall e, \forall t \tag{2c}$$

$$z_d \geq 0, \quad \forall d \tag{2d}$$

$$\phi_t \geq 0, \quad \forall t \tag{2e}$$

B. A Generic Competitive Online Routing Algorithm

As both primal and dual formulations of the problem have been set up, we can now move forward to solve them. There is abundant work in the literature giving optimal or heuristic offline solutions [25] to primal-dual problems of this kind. One major assumption of these offline solutions is that all traffic flows are known ahead of time. However, due to the continuous and first-come-first-serve (FCFS) nature of mobile data traffic, it is usually infeasible to predict the flows in advance and solve the routing problem offline or through batch processing. Therefore we shall focus on online algorithms that handle flow allocations in real-time. We shall assume for now that the algorithm is executed by a standalone server, and then shed light on practical constraints and distributed execution later.

Our objective is to find an online algorithm with a guaranteed competitive ratio. The competitive ratio of an online algorithm is the worst case ratio of the objective function achieved by the online algorithm to that achieved by the optimal offline algorithm. We first describe a generic competitive online routing algorithm [25], and later extend it to address the unique challenges in mobile traffic steering. As we shall see later, the primal-dual algorithm uses the a combination of the dual variables $l_{et} + b_{et}\phi_t$ as the link lengths, and updates them according to the primal allocations.

1) Initially, all link lengths are set to 0;
2) Whenever a new traffic flow is requested:
   a) if the shortest path to route this flow is no longer than a certain threshold, then the flow is accepted and allocated onto its shortest path, and the length of all links on this path are incremented according to their utilizations;
   b) if the shortest path to route this flow is longer than the threshold, the flow is rejected.

The intuition behind this generic algorithm is that highly congested links will accumulate larger lengths and thus are less likely to be further utilized when performing shortest path routing for future traffic flows. In this way, mobile traffic can be steered throughout the network rather than limited to random hot-spot areas. Meanwhile, each individual flow is still allocated onto a short path, consuming as few resources as possible while avoiding congestion in the network. Furthermore, when the network is already congested, those flows that require too many hops or must go through certain hot-spot areas will be dropped in order to leave space for less-demanding future requests. With all these heuristics working together, a log-competitive online algorithm can be designed with a careful selection of parameters and formulas.

Using this method, the network-wide multi-commodity flow problem is essentially transformed into a series of less complicated shortest path routing problems. Compared with other alternatives that approaches the MIP formulation directly, this algorithm retains the independence and integrity of each flow and is especially extensible for more complicated routing problems. As we shall see later, special routing policies and various QoS constraints for each flow can be easily incorporated into this algorithm.

C. Enforcing Routing Policies With Graph Layering

A key step in the generic online routing algorithm involves computation of the shortest path from s_d to the t_d for each flow d. When applying this generic algorithm to policy-aware routing, the simple shortest path will need to be replaced by a policy-aware shortest path visiting multiple NE along the way, which is much more difficult to compute. Since there are multiple instances of each NE, part of the optimization is to determine which NE to route through.

Our approach to enforce routing policies is to construct a *layered graph* $\dot{\mathcal{G}}_d$, which is constructed as follows. For any flow d, we make $k_d + 1$ copies of the original graph $\mathcal{G}$ and turn each copy $\dot{\mathcal{G}}_d^{(i)} = (\mathcal{V}^{(i)}, \mathcal{E}^{(i)})$ into one layer of $\dot{\mathcal{G}}_d$. As we shall see later, layer 1 at the bottom is used for routing from the source $s_d^{(1)}$ to $M_d^{(1)}[1]$, layer $2 \leq i \leq k_d$ is used for routing from $M_d^{(i)}[i-1]$ to $M_d^{(i)}[i]$, and the top layer k_d+1 is used for routing from $M_d^{(k_d+1)}[k_d]$ to the destination $t_d^{(k_d+1)}$. The length of link $e^{(i)}$ in layer i is set to $\gamma_d^{(i)}$ times the its mean length over the duration of flow d, i.e., $\dot{l}_{e^{(i)}} \triangleq \sum_{\tau_d^s \leq t \leq \tau_d^f} \gamma_d^{(i)}(l_{et} + b_{et}\phi_t)/\tau_d, \ \forall i, \forall e^{(i)}$. Because there are no negative links in the graph, the shortest path in each layer must be a simple path visiting each node/link at most once.

In order to enforce the routing policies, we then introduce $(0, 0, 0, \infty)$ links to stitch the layers together,

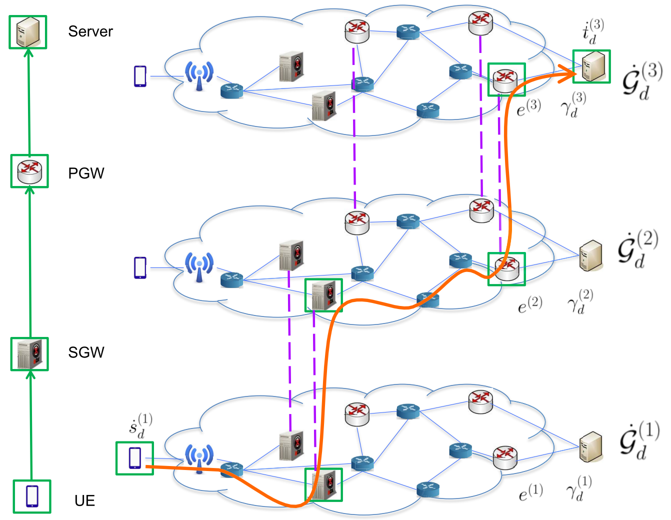


Fig. 5. Enforcing routing policies in a layered graph $\dot{\mathcal{G}}_d$.

where $(l_{et}, b_{et}, w_e, c_e)$ denotes the link parameters. Each instance of $M_d^{(i)}[i]$ in layer $1 \le i \le k_d$ is connected to its correspondent $M_d^{(i+1)}[i]$ in layer $i+1$. Also, we add a unique ingress node $\dot{s}_d$ to connect with $s_d^{(1)}$ in the bottom layer through a $(\delta, 0, 0, \infty)$ link of length $0 < \delta \ll 1$, and a unique egress node $\dot{t}_d$ to connect with $t_d^{(k_d+1)}$ in the top layer through a $(0, 0, 0, \infty)$ link. The reason why we need this small δ here will be explained later. In this way, flow d must start from the bottom layer, climb upwards through the $(0, 0, 0, \infty)$ links by visiting the NE in a specific order, and finally reach the destination in the top layer. The policy-aware shortest path $\dot{p}_d$ can be found by running a simple shortest path routing algorithm in the layered graph $\dot{\mathcal{G}}$ from ingress node $\dot{s}_d$ to egress node $\dot{t}_d$, as shown in Fig. 5. It is important to note that the parameters of all auxiliary links introduced in this paragraph are fixed and will never change throughout the optimization process regardless of the path updates.

Finally, the shortest path $\dot{p}_d$ we have just found in the layered graph $\dot{\mathcal{G}}$ can be mapped back to p_d in the original graph $\mathcal{G}$, by suppressing all copies into one single layer and eliminating auxiliary links connecting the layers.

D. Meeting QoS Requirements

Recall that when writing the dual formulation in section V-A, we have intentionally set aside the QoS constraints $x_{dp} \sum_{e \in p} w_e \le x_{dp} W_d$. These constraints are difficult to transform into the dual formulation, hence they are enforced as additional constraints to the candidate paths.

Similar to enforcing routing policies, we could substitute the simple shortest path in the generic online routing algorithm with a constrained shortest path to guarantee the QoS. There have been much work in literature addressing the constrained shortest path problem since 1980's, and many different algorithms have been proposed. Therefore, we could reuse any existing constrained shortest path algorithm as is, and plug it into the generic framework as a subroutine of our online routing algorithm.

It is also important to note that QoS-guarantee can be achieved simultaneously with policy-awareness. This is because the key step of policy enforcement in section V-C is still a simple shortest path routing, though in a much larger graph with multiple layers. If we set $w_{e^{(i)}} = w_e$ for all links $e^{(i)}$ in all layers i and replace the simple shortest path in the layered graph $\dot{\mathcal{G}}$ with a constrained shortest path, then QoS enforcement can be integrated with policy-aware routing.

Unlike the simple shortest path problem that can be solved efficiently within $O(n \log n + m)$ time in a generic graph with n nodes and m links, the constrained shortest path problem is generally NP-hard, so we have to settle for a certain level of approximation. Many proposed algorithms only work for the single constraint scenario, while only a few can be extended to support multiple constraints as well. For example, in addition to meeting end-to-end latency requirements, sometimes we may want to ensure that the selected SGW and MCC hosts are close enough to the users as well, so as to improve the user experience during cellular handover or computation handoff.

In the following we show how different $(1+\epsilon)$ approximations can be achieved in various scenarios.

1) Single-Constraint Scenario: Fast fully polynomial time approximation schemes (FPTAS), like Hassin's algorithm [26] achieving $(1+\epsilon)$ path length approximation within $O\big(\log\log\frac{UB}{LB}(\epsilon^{-1}mn + \log\log\frac{UB}{LB})\big)$ time, can be used to compute the constrained shortest path efficiently. When computing the policy-aware shortest path in $\dot{\mathcal{G}}_d$ with end-to-end latency constraint, the upper bound of the shortest path length could be set as $UB = k_{\max} n \dot{l}_{\max}$, the lower bound could be set as $LB = \delta$, and n, m should increase by $k_{\max}$ times. We shall prove later that $\dot{l}_{\max} = O(1)$, while δ in the lower bound comes from the short link connecting $\dot{s}_d$ to $s_d^{(1)}$ and is intentionally added to avoid the infinite loop to achieve $(1+\epsilon)$ path-length approximation around $L(\dot{p}_d) = 0$. Here we use a "max" subscript to indicate the maximum value of the parameter, and a "min" subscript to indicate the minimum positive value of the parameter. Therefore, the final complexity is $T_{\dot{p}_d} = O\Big(\log\log(k_{\max}n/\delta)\big(\epsilon^{-1}k_{\max}^2 mn + \log\log(k_{\max}n/\delta)\big)\Big)$.

2) Relaxed Multi-Constraint Scenario: Computing the shortest path under multiple side constraints is much more difficult. Existing literature often gives solutions to the problem after certain relaxations. In [27, Sec. 3.3], the authors relax the original problem by dropping the path integrality constraint, so that traffic flow can be split into multiple paths, and the constraints are only satisfied by the average values of these paths. Then they use a hull approach to obtain the solution to the relaxation, which is a lower bound of the multi-constrained shortest path but without a constant approximation guarantee. As a result, this method is only suitable for splittable traffic and QoS guarantee in an average sense. When enforcing several different latency constraints (end-to-end, MCC, UE-to-SGW) in a generic graph made up of n nodes and m links with integral lengths and latencies, this algorithm can derive an exact solution to the relaxation with a conjectured time complexity of $O\big(\log(nl_{\max}w_{\max})(n\log n+m)\big)$. But for a layered graph $\dot{\mathcal{G}}_d$ with real-valued parameters, if we terminate the hull approach as soon as the gap between the upper bound and lower bound of $L(\dot{p}_d)$ becomes smaller than $\epsilon\delta \le \epsilon L(\dot{p}_d)$, then the final time complexity is $T_{\dot{p}_d} = O(\log(nk_{\max}\dot{l}_{\max}w_{\max}/\epsilon\delta w_{\min})(nk_{\max}\log(nk_{\max})+mk_{\max}))$.

3) Strict Multi-Constraint Scenario: Designing an online algorithm for the strict multi-constraint shortest path is even more challenging, and fast solutions with a constant path-length approximation guarantee might not be available. Therefore, we could only fall back to classical dynamic programming approaches [27].

E. A Log-Competitive Online Algorithm for Maximizing Total Accepted Mobile Traffic Over Time

Summing up all the aforementioned techniques, a log-competitive online solution is designed as follows.

Denote the largest values of k_d and τ_d for all flows as $k_{\max}$ and $\tau_{\max}$. Also, because the network is connected, we have $n = O(m)$. Hence the time complexity for constructing $\dot{\mathcal{G}}_d$ is $T_{\mathcal{G}} = O(mk_{\max})$, the time complexity for defining the length system $\dot{\mathcal{L}}$ is $T_{\dot{\mathcal{L}}} = O(mk_{\max}\tau_{\max})$, the time complexity for finding the policy-aware and QoS-guaranteed shortest path $\dot{p}_d$ in $\dot{\mathcal{G}}_d$ is $T_{\dot{p}_d}$ as in section V-D, and the time complexity for updating the path lengths along p_d in $\mathcal{G}$ is $T_{p_d} = O(nk_{\max}\tau_{\max})$. Therefore, the overall time complexity for each flow allocation is $T = O(mk_{\max}\tau_{\max} + T_{\dot{p}_d})$.

Such a complexity can become high as the network grows larger and more traffic flows emerge, and even worse if we want to find close approximations of QoS-constrained shortest paths. In order to scale the system, we need to exploit the parallelism of the algorithm and distribute the computation-intensive tasks to multiple schedulers. Fortunately, the algorithm above is indeed parallelizable under reasonable assumptions. To be more specific, if the cost of each single traffic flow is much smaller than the link capacities and budget constraints, then the allocation of a set of traffic flows can be delegated to multiple different schedulers and performed simultaneously using their local versions of the network states (which may be slightly outdated). Allocations performed by each single scheduler need to be eventually propagated to all other schedulers to update their local versions of network states, but strong consistency among different schedulers is not a must, because the allocation of each single traffic flow will not significantly affect the network and budget utilizations. While this breaks the serializable consistency, the loss of accuracy is minor and the whole system can tolerate out-of-sync network states to some reasonable extent.

F. Competitiveness Analysis of the Online Algorithm

Denote by $\Gamma_{\max}$ the maximum value of Γ_{p_d}, $\Psi_{\max}$ the maximum value of Ψ_{p_d}, $\gamma_{\min}$ the minimum positive value of $\gamma_d^{(i)}$, $b_{\min}$ the minimum positive value of b_{et}, and $\tau_{\max}$ the maximum value of τ_d. For those $\gamma_d^{(i)} = 0$ (or $b_{et} = 0$), l_{et} (or ϕ_t) is meaningless and therefore ignored. The competitiveness of Algorithm 1 can be proved as follows:

Algorithm 1 A Log-Competitive Online Algorithm for Maximizing Total Accepted Mobile Traffic Over Time

```
Data: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$; link capacities $c_e$ and unit costs
      $b_{et}$; a sequence of flows $d$, each with source $s_d$,
      sink $t_d$, bandwidth requirement $h_d$, policy list
      $\{M_d[i]\}$ together with bandwidth variation $\{\gamma_d^{(i)}\}$
      for $1 \le i \le k_d$, starting time $\tau_d^s$, and finishing time
      $\tau_d^f$; operational budgets $B_t$; a small $\delta$;
Result: Sequential decision of flow allocation $x_{dp}$;
Initialize $x_{dp} := 0,\ \forall d, \forall p$; $l_{et} := 0,\ \forall e, \forall t$; $z_d := 0,\ \forall d$;
 $\phi_t := 0,\ \forall t$;
foreach flow $d$ do
    Construct a layered graph $\dot{\mathcal{G}}_d$ with $k_d + 1$ copies of
     the original graph $\mathcal{G}$;
    Stitch adjacent layers $\dot{\mathcal{G}}_d^{(i)}$ in the layered graph $\dot{\mathcal{G}}_d$
     together by connecting each instance of $M_d^{(i)}[i]$ in
     layer $1 \le i \le k_d$ to its correspondent $M_d^{(i+1)}[i]$ in
     layer $i+1$ through $(0,0,0,\infty)$ links;
    Define a length system $\dot{\mathcal{L}}$ on each link $e^{(i)}$ such that
     $\dot{l}_{e^{(i)}} \triangleq \sum_{t\in[\tau_d^s,\tau_d^f]} \gamma_d^{(i)}(l_{et} + b_{et}\phi_t)/\tau_d,\ \forall i, \forall e^{(i)}$;
    Connect $\dot{s}_d$ to $s_d^{(1)}$ through a $(\delta,0,0,\infty)$ link, and $\dot{t}_d$
     to $t_d^{(k_d+1)}$ through a $(0,0,0,\infty)$ link;
    Find the policy-aware and QoS-guaranteed shortest
     path $\dot{p}_d$ from $\dot{s}_d$ to $\dot{t}_d$ in the layered graph $\dot{\mathcal{G}}_d$ under
     the length system $\dot{\mathcal{L}}$;
    Map $\dot{p}_d$ from the layered graph $\dot{\mathcal{G}}_d$ to its realization
     $p_d$ in the original graph $\mathcal{G}$;
    if $L(p_d) \triangleq L(\dot{p}_d) \triangleq \sum_i \sum_{e^{(i)}\in\dot{p}_d^{(i)}} \dot{l}_{e^{(i)}} < 1$ then
        $x_{dp_d} := 1$;
        $z_d := h_d\tau_d$;
        $\Gamma_{p_d} \triangleq \sum_i \sum_e \gamma_d^{(i)} \mathbb{1}_{e\in p_d^{(i)}}$;
        $l_{et} := l_{et}\left(1 + \sum_{i|e\in p_d^{(i)}} \frac{\gamma_d^{(i)} h_d}{c_e}\right) +$
         $\sum_{i|e\in p_d^{(i)}} \frac{\gamma_d^{(i)} h_d}{\Gamma_{p_d} c_e},\ \forall e \in p_d, \forall t \in [\tau_d^s, \tau_d^f]$;
        $\Psi_{p_d} \triangleq \sum_{t\in[\tau_d^s,\tau_d^f]} \sum_i \sum_{e\in p_d^{(i)}} b_{et}\gamma_d^{(i)}/\tau_d$;
        $\phi_t := \phi_t\left(1 + \sum_i \sum_{e\in p_d^{(i)}} \frac{b_{et}\gamma_d^{(i)} h_d}{B_t}\right) +$
         $\sum_i \sum_{e\in p_d^{(i)}} \frac{b_{et}\gamma_d^{(i)} h_d}{\Psi_{p_d} B_t},\ \forall t \in [\tau_d^s, \tau_d^f]$;
```

1) The algorithm produces a feasible dual solution.

Whenever a new flow d is requested, either it is accepted and we assign $z_d = h_d\tau_d$, or it is rejected because its shortest path length $L(p_d) = L(\dot{p}_d) = \sum_i \sum_{e^{(i)}\in\dot{p}_d^{(i)}} \dot{l}_{e^{(i)}} = \sum_{t\in[\tau_d^s,\tau_d^f]} \sum_i \sum_{e\in p^{(i)}} \gamma_d^{(i)}(l_{et} + b_{et}\phi_t)/\tau_d \ge 1$, so inequality (2b) always holds for any flow d. Obviously, inequalities (2c) and (2d) are also valid, hence the dual solution is feasible.

2) In each iteration, the change of the dual objective is bounded by that of the primal objective $\Delta\beta \le 4\Delta\alpha$.

If $L(p_d) \ge 1$, nothing changes, and $\Delta\alpha = \Delta\beta = 0$.

If $L(p_d) < 1$, we set $x_{dp_d} := 1$, thus $\Delta\alpha = h_d\tau_d$, whereas

$$\begin{aligned}
\Delta\beta &= \Delta z_d + \sum_t \sum_e c_e \Delta l_{et} + \sum_t B_t \Delta\phi_t \qquad (3)\\
&= h_d\tau_d + \sum_{t\in[\tau_d^s,\tau_d^f]} \sum_{e\in p_d} c_e \sum_{i|e\in p_d^{(i)}} \left(\frac{\gamma_d^{(i)} h_d l_{et}}{c_e} + \frac{\gamma_d^{(i)} h_d}{\Gamma_{p_d} c_e} \right)\\
&\quad + \sum_{t\in[\tau_d^s,\tau_d^f]} B_t \sum_i \sum_{e\in p_d^{(i)}} \left(\frac{b_{et}\gamma_d^{(i)} h_d \phi_t}{B_t} + \frac{b_{et}\gamma_d^{(i)} h_d}{\Psi_{p_d} B_t} \right)\\
&= 3h_d\tau_d + \sum_{t\in[\tau_d^s,\tau_d^f]} \sum_i \sum_{e\in p_d^{(i)}} \gamma_d^{(i)} h_d (l_{et} + b_{et}\phi_t)\\
&= 3h_d\tau_d + L(p_d) h_d \tau_d\\
&\le 4h_d\tau_d = 4\Delta\alpha \qquad (4)
\end{aligned}$$

Therefore, $\Delta\beta \le 4\Delta\alpha$ always holds.

3) The link capacity constraints in the dual problem is violated by at most $O\big(\log(\Gamma_{\max}\tau_{\max}/\gamma_{\min})\big)$, if $c_e \gg 1$ and $c_e \gg \sum_{i|e\in p^{(i)}} \gamma_d^{(i)} h_d,\ \forall d, \forall p \in \mathcal{P}_d, \forall e$.

We shall prove through mathematical deduction that the value of l_{et} just before the arrival of flow d can be bounded by the following expression:

$$l_{et}(d) \ge \frac{(1+\frac{1}{c_e})^{\sum_{d'<d|t\in[\tau_{d'}^s,\tau_{d'}^f]} \sum_{p\in\mathcal{P}_{d'}|e\in p} \sum_{i|e\in p^{(i)}} \gamma_{d'}^{(i)} x_{d'p} h_{d'}} - 1}{\Gamma_{\max}}. \qquad (5)$$

Initially $l_{et}(1) = 0$, and all $x_{dp} = 0$, so the inequality holds. Assume that the inequality holds before flow d arrives. Then after this iteration, either flow d is rejected and $x_{dp_d} = 0$ does not affect the validity of $l_{et}(d+1)$, or flow d is accepted and

$$\begin{aligned}
&l_{et}(d+1) \qquad (6)\\
&= l_{et}(d)\left(1 + \sum_{i|e\in p_d^{(i)}} \frac{\gamma_d^{(i)} h_d}{c_e}\right) + \sum_{i|e\in p_d^{(i)}} \frac{\gamma_d^{(i)} h_d}{\Gamma_{p_d} c_e}\\
&\ge l_{et}(d)\left(1 + \sum_{i|e\in p_d^{(i)}} \frac{\gamma_d^{(i)} h_d}{c_e}\right) + \sum_{i|e\in p_d^{(i)}} \frac{\gamma_d^{(i)} h_d}{\Gamma_{\max} c_e}\\
&= \frac{(1+\frac{1}{c_e})^{\sum_{d'<d|t\in[\tau_{d'}^s,\tau_{d'}^f]} \sum_{p\in\mathcal{P}_{d'}|e\in p} \sum_{i|e\in p^{(i)}} \gamma_{d'}^{(i)} x_{d'p} h_{d'}}}{\Gamma_{\max}}\\
&\quad \cdot \left(1 + \sum_{i|e\in p_d^{(i)}} \frac{\gamma_d^{(i)} h_d}{c_e}\right) - \frac{1}{\Gamma_{\max}}\\
&\approx \frac{(1+\frac{1}{c_e})^{\sum_{d'<d+1|t\in[\tau_{d'}^s,\tau_{d'}^f]} \sum_{p\in\mathcal{P}_{d'}|e\in p} \sum_{i|e\in p^{(i)}} \gamma_{d'}^{(i)} x_{d'p} h_{d'}} - 1}{\Gamma_{\max}}. \qquad (7)
\end{aligned}$$

The last approximation holds if $c_e \gg 1$ and $c_e \gg \sum_{i|e\in p^{(i)}} \gamma_d^{(i)} h_d,\ \forall d, \forall p \in \mathcal{P}_d, \forall e$. Therefore, the inequality still holds after serving flow d, and the mathematical deduction is valid.

On the other hand, due to the acceptance condition $L(p_d) < 1,\ \forall d$ and the assumption $c_e \gg \sum_{i|e\in p_d^{(i)}} \gamma_d^{(i)} h_d,\ \forall d$, $\forall p \in \mathcal{P}_d, \forall e$, we have $l_{et}(\infty) < (\tau_{\max}/\gamma_{\min}) \cdot (1+1) + 1 = O(\tau_{\max}/\gamma_{\min})$, and $\dot{l}_{\max} < 1 \cdot (1+1) + 1 = 3$. Combining this with inequality (5), the link utilization can be bounded by

$$\begin{aligned}
u_{et} &\triangleq \frac{\sum_{d|t\in[\tau_d^s,\tau_d^f]} \sum_{p\in\mathcal{P}_d|e\in p} \sum_{i|e\in p^{(i)}} \gamma_d^{(i)} h_d x_{dp}}{c_e}\\
&\le \frac{O\big(\log(\Gamma_{\max}\tau_{\max}/\gamma_{\min})\big)}{c_e \log(1+\frac{1}{c_e})}\\
&\approx O\big(\log(\Gamma_{\max}\tau_{\max}/\gamma_{\min})\big) \qquad (8)
\end{aligned}$$

The last approximation holds if $c_e \gg 1,\ \forall e$.

4) The operational budget constraints in the dual problem is violated by at most $O\big(\log(\Psi_{\max}\tau_{\max}/\gamma_{\min})\big)$, if $B_t \gg 1$ and $B_t \gg \sum_i \sum_{e\in p^{(i)}} b_{et}\gamma_d^{(i)} h_d,\ \forall d, \forall p \in \mathcal{P}_d, \forall t$.

Similar to the proof in step 3), we show through mathematical deduction that the value of ϕ_t just before the arrival of flow d is bounded by the following expression (9), as shown at the bottom of this page.

On the other hand, due to the acceptance condition $L(p_d) < 1,\ \forall d$ and the assumption $B_t \gg \sum_i \sum_{e\in p^{(i)}} b_{et}\gamma_d^{(i)} h_d,\ \forall d, \forall p \in \mathcal{P}_d, \forall t$, we have $\phi_t(\infty) \le (\tau_{\max}/b_{\min}\gamma_{\min}) \cdot (1+1) + 1 = O(\tau_{\max}/b_{\min}\gamma_{\min})$. Combining this with inequality (5), the budget utilization can be bounded by

$$\begin{aligned}
\eta_t &\triangleq \frac{\sum_{d|t\in[\tau_d^s,\tau_d^f]} \sum_{p\in\mathcal{P}_d} \sum_i \sum_{e\in p^{(i)}} b_{et}\gamma_d^{(i)} h_d x_{dp}}{B_t}\\
&\le \frac{O(\log\big(\Psi_{\max}\tau_{\max}/b_{\min}\gamma_{\min})\big)}{B_t \log(1+\frac{1}{B_t})}\\
&\approx O\big(\log(\Psi_{\max}\tau_{\max}/b_{\min}\gamma_{\min})\big) \qquad (10)
\end{aligned}$$

The last approximation holds if $B_t \gg 1,\ \forall t$.

Rather than viewing u_{et} in steps 3) and η_t in 4) as violations to the capacity and budget constraints, we can scale c_e and B_t instead, and absorb the logarithmic factors in the competitive ratio so as to avoid violations. It is possible to show that (within constants) this is the best possible competitive ratio.

VI. More Discussion

A. Support for Multi-RAT Scenarios

Latest UE may support multiple radio access technologies and connect to different mobile networks that differ significantly in performance, availability, and routing policies. In addition to the difference in their inherent capabilities, the network performances are also affected by fluctuations in

$$\phi_t(d) \ge \frac{(1+\frac{1}{B_t})^{\sum_{d'<d|t\in[\tau_{d'}^s,\tau_{d'}^f]} \sum_{p\in\mathcal{P}_{d'}|e\in p} \sum_i \sum_{e\in p^{(i)}} b_{et}\gamma_{d'}^{(i)} x_{d'p} h_{d'}} - 1}{\Psi_{\max}}. \qquad (9)$$

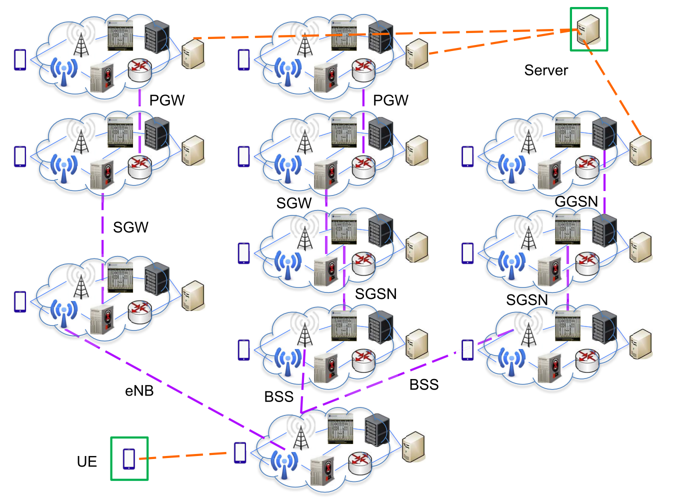


Fig. 6. Extension on graph layering to support multi-RAT.

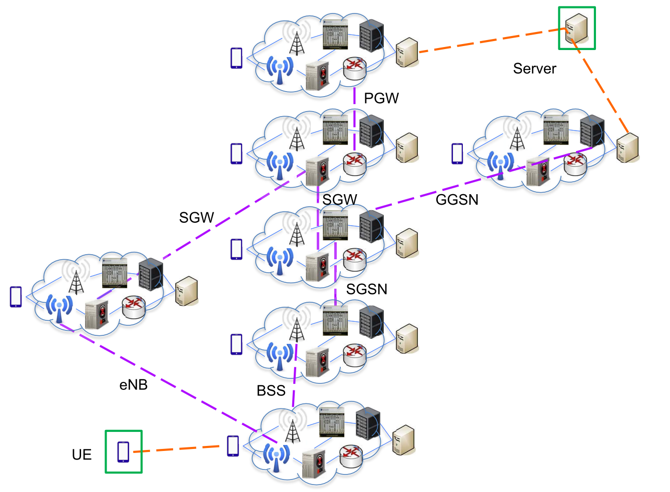


Fig. 7. Merging layers representing common route segments in multi-RAT.

traffic demands and change of user locations. Therefore, when routing mobile traffic flows, these differences must also be taken into account in order to optimize the system performance and the user experience.

In multi-RAT scenarios, throughput maximization can be done in the same way as in a single-RAT scenario, except that we should merge multiple networks into one large graph and enforce different routing policies for each possible routing option that the flow can be served with.

A straightforward solution is to reuse the graph layering technique to set up separate branches for each routing option. For example, assume that the UE can access either 2G or 4G LTE at this point, then it has at least three routing options:

1) UE $\rightarrow$ BTS $\rightarrow$ SGSN $\rightarrow$ GGSN $\rightarrow$ Server over 2G GERAN and GPRS core network.
2) UE $\rightarrow$ BTS $\rightarrow$ SGSN $\rightarrow$ SGW $\rightarrow$ PGW $\rightarrow$ Server over GERAN and GPRS interworking with EPC.
3) UE $\rightarrow$ eNodeB $\rightarrow$ SGW $\rightarrow$ PGW $\rightarrow$ Server over 4G LTE E-UTRAN and EPC.

If we merge both 2G and 4G LTE networks and set up a branch for each routing option, then the layered graph $\dot{\mathcal{G}}_d$ should look like the following Fig. 6, which requires 3 branches and a total of 11 copies of $\mathcal{G}$. Note that different copies of the destination node t_d in each branch should be aggregated at the top because we only need one feasible path from s_d to t_d across all 3 branches.

The straightforward solution meets our expectations but can be further improved. Because option 1) shares route segment UE $\rightarrow$ BTS $\rightarrow$ SGSN with option 2), and option 2) shares segment SGW $\rightarrow$ PGW $\rightarrow$ Server with option 3), it is possible to use less copies of $\mathcal{G}$ to cover all routing options in the multi-RAT scenario by merging those layers associated with common route segments. As shown in Fig. 7, only 7 copies of $\mathcal{G}$ is sufficient to get the same result.

B. Routing Across Network Boundaries

In many countries, mobile networks are operated by multiple mobile Internet service providers across different regions. In many cases, a UE may not receive adequate cellular coverage from its home network and thus requires roaming to a visited network. However, charges made by different mobile networks and QoS offered by different routing options may vary widely. In such scenarios, coordination across different mobile networks and communications between their SDN controllers are necessary.

We adopt a master-slave model for network cooperations. Since it is often the home network that determines a user's privileges and charges, we let the visited network serve as the slave to provide a list of routing options to the master, while the home network should act as the master to make the final routing decisions based on the given options.

In 4G LTE networks, home-routed roaming traffic utilizes a visited SGW (V-SGW) and a home PGW (H-PGW), and crosses the boundary of home and visited mobile networks via an IP exchange (IPX) network. We assume that the visited network determines the route segment from the UE to the IPX (denoted by node v) and makes a competitor charge depending on the resources used, while the home network takes over the control thereafter. In this process, the visited network decides how to model the link lengths/costs and how to route the roaming traffic till the IPX node v, e.g., according to the proposed primal-dual algorithm, and then informs the home network of the competitor charge g_v and the actual network latency w_v. On the other hand, the home network effectively sees a single $(0, g_v, w_v, \infty)$ link between the UE and the IPX node v, and the total cost of routing is the sum of node/link costs from node v to the server plus the competitor charge g_v, as shown in Fig. 8. In case there are multiple IPXs connecting these two networks, the visited network needs to provide routing options from the UE to all possible IPXs, while the home network needs to select the best route from all possible IPXs to the server. The rest of the problem is exactly the same as before.

The method above applies to two selfish network operators with conflict of interest and limited cooperation. Another version of the problem is for two network operators with a shared objective or two SDN controllers in the same network

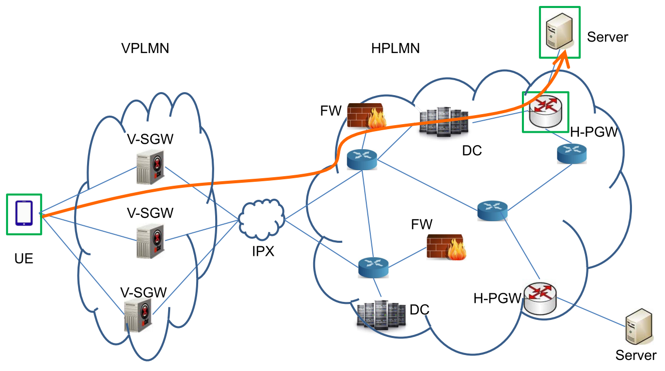


Fig. 8. Routing across two mobile network operators.

aiming for full cooperation. In this case, it is tricky to divide the end-to-end latency quota across the two networks, but with shared network state information each SDN controller is able to make the routing decision alone and the scheduling tasks can be distributed to both controllers.

VII. Simulation Results

In this section, we perform numerical simulations to compare the effectiveness of the proposed algorithm against other prevalent heuristics, and demonstrate how our algorithm outperforms these alternatives.

First, we list the routing algorithms to be compared:

1) *PhSp*: per-hop shortest-path routing. This is the baseline algorithm in which all links are given a uniform unit length, and the scheduler iteratively finds the closest (in terms of the number of links) next-hop NE in a hop-by-hop manner and connects these segments up to construct an end-to-end path.
2) *PhMl*: per-hop minimum-latency routing. This works in a similar way to *PhSp* except that the network latency is minimized instead of the length of each route segment.
3) *Sp*: end-to-end shortest-path routing. This algorithm is an improved version of *PhSp* that utilizes the graph layering technique to find the end-to-end shortest (in terms of the number of links) path instead of connecting up shortest route segments hop by hop.
4) *Ml*: end-to-end minimum-latency routing. This works in a similar way to *Sp* except that we are now trying to find an end-to-end path with the minimum network latency rather than the number of links that compose the path.
5) *Csp*: end-to-end constrained shortest-path routing. In this algorithm, both network latencies and path lengthes are considered, and our objective is to minimize the end-to-end path length under network latency constraints.
6) *PdCsp*: end-to-end primal-dual constrained shortest-path routing. This is the proposed competitive online algorithm that is aware of bandwidth variations and follows the primal-dual approach to assign link lengths and perform end-to-end latency-constrained shortest path routing.

The next step is to construct a mobile network and some traffic flows for simulation. In order to reflect real-world latencies, we abstract the contiguous United States territory as a $2800mi \times 1500mi$ rectangular plane, and deploy network nodes and links on this plane. We first construct a backbone network with 20 backbone routers and 30 backbone links. These backbone nodes are randomly placed and linked but we ensure that the whole network is connected. The backbone links are assigned a uniform capacity of $40Gbps$, which corresponds with the OC-768 fiber links that were deployed by AT&T around 2008. Then we add 80 more routers and some more links to simulate the wide area networks (WANs) operated by local ISPs. These WAN nodes are deployed sequentially and they only connect to one or two nearest already deployed nodes. These WAN links are assumed to have a uniform capacity of $10Gbps$. Regardless of backbone network or WAN, the network latency of a link is approximated as the geographical distance between the two end nodes divided by the speed of light in fiber. After that, 2000 traffic flows are generated with random starting/ending time slots, UE/resource nodes, and bandwidths between $1Mbps$ and $1Gbps$. Each traffic flow has only one UE node, but may have multiple resource nodes to account for CDN or geographical distribution of resources. In Experiment I, we assume that all UE-generated traffic flows need to traverse through a SGW and a PGW before reaching the server. These UE, SGW, PGW, and server nodes are randomly attached to backbone and WAN nodes. A list of all parameters used in the simulation is summarized as in Table II, where $randInt(a, b)$ represents a random integer value uniformly chosen from range $[a, b]$.

TABLE II
Simulation Parameters for Experiment I

Parameters	Value
simulation period T	100
geographic area	$2800mi \times 1500mi$
$n\_backbone\_routers$	20
$n\_wan\_routers$	80
$n\_backbone\_links$	30
n_wan_links	109
$c\_backbone\_link$	$40Gbps$
c_wan_link	$10Gbps$
D	2000
τ_d	$randInt(1, 20)$
h_d	$randInt(1Mbps, 1Gbps)$
W_d	$2ms, 4ms, 6ms, 8ms, 10ms, infinity$
n_sgw, n_pgw	$4, 8, 12, 16, 20$
$n\_resources$	$randInt(1, 7) \times 1, 2, 3, 4, 5$
$\gamma_d^{(i)}$	1

We do not impose any operational costs, capacity limits, or bandwidth variations on NFV and MCC elements in this experiment, so that the advantages of the proposed algorithm in terms of routing under capacity and latency constraints as well as the impact of different NE densities can be clearly demonstrated. Also note that all topologies and capacity/demand values are arbitrarily chosen here for demonstration purposes only; the actual values may change significantly over time. When applying the algorithms to real networks, up-to-date link capacities, accurate user demand estimations, and realistic budget/cost assessment are key considerations.

We first compare the performance of all 6 routing algorithms under different NE densities. The metric to be used for

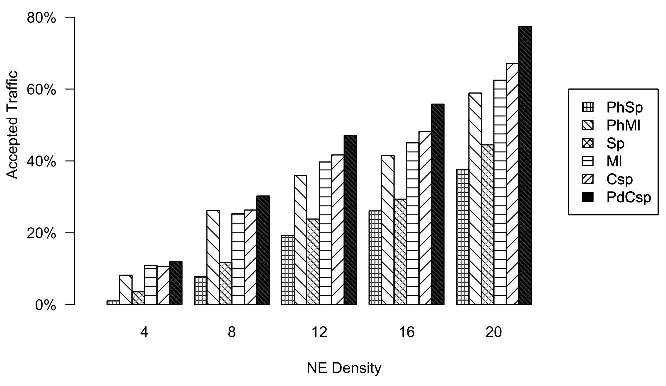


Fig. 9. Total amount of traffic accepted under different NE density levels.

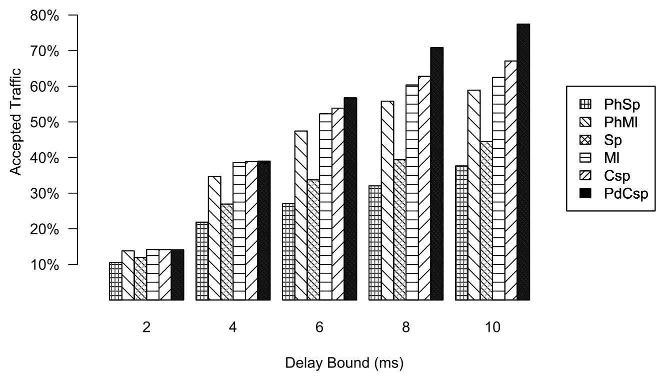


Fig. 10. Total amount of traffic accepted under different latency bounds.

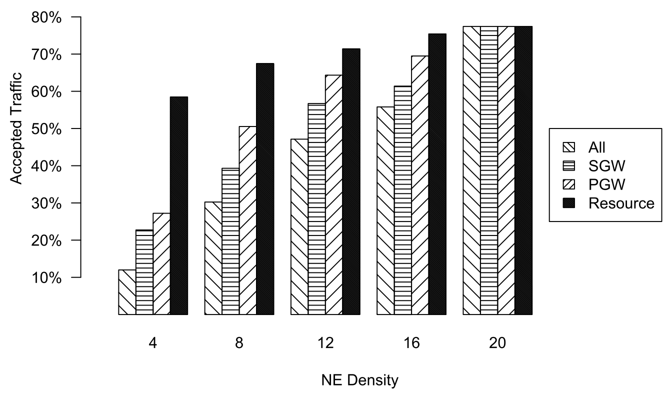


Fig. 11. Impact of SGW/PGW/resource density levels on the total amount of traffic accepted over time under $10ms$ latency constraints.

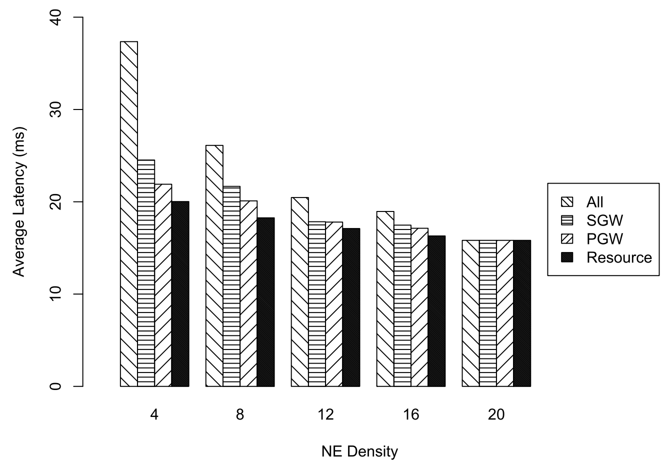


Fig. 12. Impact of SGW/PGW/resource density levels on the network latency.

evaluation is the total amount of traffic accepted over time, which is also the objective of the problem addressed in this paper. The simulation results are shown in Fig. 9. As we can see, the performance of the baseline *PhSp* algorithm is always lower than others in this simulation. *PhMl* performs better in this simulation because it seeks the minimize the latency for each flow so that more flows can meet the latency constraints. Sp and Ml are superior to their per-hop counterparts in this simulation because the routing path is optimized from end to end. Csp achieves even higher performance in this simulation by balancing the tradeoff between path lengths and network latencies. Finally, our proposed $PdCsp$ algorithm is able to accept the highest amount of traffic over time, which is more than 100% higher than the baseline algorithm in many cases. The significant improvement should be attributed to 1) end-to-end path optimization, 2) congestion-aware traffic steering, and 3) the primal-dual approximation.

Next, we gradually decrease the latency bound from $10ms$ to $2ms$, and see how different algorithms react to such changes. The results are plotted in Fig. 10, which has a similar look to Fig. 9. Again, the proposed $PdCsp$ algorithm significantly outperforms other heuristics. Also, we can see from both figures that the performances of *Ml*, *Csp*, and *PdCsp* are very close when the latency constraint is stringent given the NE density. This is due to the fact that 1) there are few alternative routes that can meet the low latency bound, and 2) the network is generally under-utilized so congestion avoidance does not take effect.

Then we set out to investigate the impact of separate SGW, PGW, and resource densities on the network performance. In addition to the "All" scenario in which the densities of all NE are changed simultaneously, we also test the network performances when only one kind of NE is affected at a time (e.g., $n\_sgw = 4, 8, 12, 16, 20$) while all others are kept the same (e.g., $n\_pgw = n\_resources = 20$), as shown in Fig. 11. Note that all bars at $n\_sgw = n\_pgw = n\_resources = 20$ are the same because they are essentially using the same settings. From the plot we can see that the impact of SGW density is more significant than that of PGW density, which in turn is more significant than that of resource density. The reason why SGW density plays a more important role is due to the fact that there is only one UE involved in each flow, whereas multiple candidate servers can participate in the same route selection process. Therefore, the triangular routing problem is more sensitive to the SGW density. On the other hand, resource density has a much weaker influence than SGW and PGW densities because flows terminate at resource nodes and the choice of resource nodes do not cause any further routing inefficiencies.

Network latency is another important metric to be considered. In all previous simulations, we have set a latency bound, which not only limits the amount of traffic that can be accepted

TABLE III
SIMULATION PARAMETERS FOR EXPERIMENT II

Parameters	Value
D	1000
W_d	$10ms$
n_sgw, n_pgw, n_mcc	20
$n\_resources$	$randInt(1,7) \times 5$
$\gamma_d^{(i)}$	$randFloat(1/5, 5)$
b_{vt}	$randFloat(0, 2)$
B_t	$10, 20, 40, 80, 160, infinity$

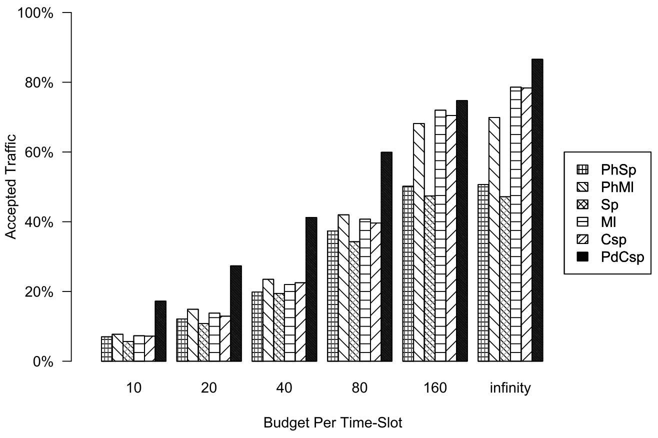


Fig. 13. Total amount of traffic accepted over time under different budgets.

but also biases the average network latency observed by all flows. In Fig. 12, we remove the latency constraints from all flows, and use the proposed *PdCsp* algorithm to show how different NE densities affect the average network latency. Note that after we remove the latency constraints, almost all traffic demands can be accepted in most cases, but the average network latency increases significantly. This is because*PdCsp* seeks to maximize the total amount of traffic accepted over time by balancing the link utilizations throughout the network, and thus the paths chosen may not have low latencies after the latency constraints are removed. Again, we could see that SGW density most significantly affects the average network latency, followed by PGW and resource densities.

Till now we have been focusing on the performance under network capacity and latency constraints only. The capabilities of the proposed *PdCsp* algorithm in handling bandwidth variations over different route segments and meeting total operational budgets bring more advantages to the proposed algorithm. These shall be verified in Experiment II. In this experiment, we reuse the backbone and WAN network settings. As outlined in Table III, a fixed number of 20 SGW, PGW, and MCC nodes are attached to the network, each having a random operational cost between 0 and 2 with an average of 1 unit per time-slot. There are 1000 randomly generated traffic flows. Each user request has 20 candidate resource nodes on average, and must sequentially pass through one instance of SGW, MCC, and PGW before reaching the resource. The bandwidth may vary randomly between $1/5$ and 5 when passing through the MCC host.

In Fig. 13, we compare the performance of various routing algorithms under different budget constraints. It can be seen that *PdCsp* gains huge advantage over other alternatives when the budget is tight, because it gives preference to less-demanding traffic flows in this situation while all others are ignorant of the capacity and budget constraints. The advantage of budget awareness gradually goes away as the budget increases and the performance becomes network-constrained. Another interesting phenomenon in this simulation is that the order of performances here significantly differs from our previous observations from Experiment I: *Sp* performs worse than *PhSp*, *Ml* performs worse than *PhMl*, and *Csp* performs worse than *Ml*. Such a contradiction seems surprising at first glance, but this is actually reasonable because all alternative algorithms other than *PhCsp* are just heuristics. Some heuristics may happen to perform better than others in certain situations, but the lack of simultaneous congestion and budget awareness makes them incapable of securing a guaranteed performance improvement.

Finally, the advantages of the proposed *PdCsp* algorithm over other alternatives are summarized in Table IV.

TABLE IV
COMPARISON OF ALGORITHMS

Characteristics	*PhSp*	*PhMl*	*Sp*	*Ml*	*Csp*	*PdCsp*
hop awareness	✓	×	✓	×	✓	✓
latency awareness	×	✓	×	✓	✓	✓
congestion awareness	×	×	×	×	×	✓
budget awareness	×	×	×	×	×	✓
end-to-end optimization	×	×	✓	✓	✓	✓
performance guarantee	×	×	×	×	×	✓

VIII. CONCLUSION

In this paper, we reveal the infrastructure inefficiencies and routing inflexibilities in current mobile networks, and propose an SDN-based architecture with cloud computing to address the potential bottlenecks and challenges of next generation mobile networking. In order to enable policy-aware routing and provide QoS guarantee, we formulate a multi-commodity flow problem with side constraints, and follow the primal-dual approach to obtain a log-competitive solution using a fast online approximation algorithm. Extensive simulations have been performed to investigate the impact of various parameters on the network performances. These results also show that our algorithm significantly outperforms prevalent heuristics due to end-to-end optimization, congestion/budget awareness, and primal-dual approximation.

REFERENCES

[1] *Network Architecture*, document 3GPP TS 23.002 Release 13, 3GPP, 2015.
[2] V. Sekar *et al.*, "Design and implementation of a consolidated middlebox architecture," in *Proc. NSDI*, 2012, pp. 323–336.
[3] L. E. Li, Z. M. Mao, and J. Rexford, "Toward software-defined cellular networks," in *Proc. EWSDN*, Oct. 2012, pp. 7–12.
[4] *General Packet Radio Service (GPRS) Enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN)*, document 3GPP TS 23.401 Release 13, 3GPP, 2014.
[5] J. Best. *The Race to 5G: Inside the Fight for the Future of Mobile as we Know it*. [Online]. Available: http://www.techrepublic.com/article/does-the-world-really-need-5g, Dec. 2014.

[6] NGMN Alliance, *5G White Paper*. [Online]. Available: https://www.ngmn.org/uploads/media/NGMN_5G_White_Paper_V1_0.pdf, Feb. 2015.

[7] Open Networking Foundation, "Software-defined networking: The new norm for networks, " *ONF White Paper*, Apr. 2012.

[8] X. Jin, L. E. Li, L. Vanbever, and J. Rexford, "SoftCell: Taking control of cellular core networks," Computer Science Department, Princeton Univ., Princeton, NJ, USA, Tech Rep. TR-950-13, 2013.

[9] P. Demestichas *et al.*, "5G on the horizon: Key challenges for the radio-access network," *IEEE Veh. Technol. Mag.*, vol. 8, no. 3, pp. 47–53, Sep. 2013.

[10] A. Basta *et al.*, "Applying NFV and SDN to LTE mobile core gateways, the functions placement problem," in *Proc. ACM Workshop All Things Cellular*, 2014, pp. 33–38.

[11] T. Taleb, "Toward carrier cloud: Potential, challenges, and solutions," *IEEE Wireless Commun.*, vol. 21, no. 3, pp. 80–91, Jun. 2014.

[12] B. G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in *Proc. EuroSys*, Apr. 2011, pp. 301–314.

[13] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct./Dec. 2009.

[14] W. Zhang, Y. Wen, and D. O. Wu, "Energy-efficient scheduling policy for collaborative execution in mobile cloud computing," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 190–194.

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

[16] A. Gember *et al.* (2013). "Stratos: A network-aware orchestration layer for middleboxes in the cloud." [Online]. Available: https://arxiv.org/abs/1305.0209

[17] J. Sherry *et al.*, "Making middleboxes someone else's problem: Network processing as a cloud service," in *Proc. ACM SIGCOMM*, 2012, pp. 13–24.

[18] A. Gember, R. Grandl, A. Anand, T. Benson, and A. Akella, "Stratos: Virtual middleboxes as first-class entities," Univ. Wisconsin-Madison, Madison, WI, USA, Tech. Rep. TR1771, 2012.

[19] R. Soulé, S. Basu, R. Kleinberg, E. G. Sirer, and N. Foster, "Managing the network with Merlin," in *Proc. ACM HotNets*, 2013, Art. no. 24.

[20] L. E. Li *et al.*, "PACE: Policy-aware application cloud embedding," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 638–646.

[21] S. Jain *et al.*, "B4: Experience with a globally-deployed software defined WAN," in *Proc. ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 3–14, 2013.

[22] H. Wang, S. Chen, H. Xu, M. Ai, and Y. Shi, "SoftNet: A software defined decentralized mobile network architecture toward 5G," *IEEE Netw.*, vol. 29, no. 2, pp. 16–22, Mar./Apr. 2015.

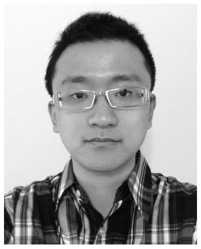
[23] *Policy and Charging Control Architecture*, document 3GPP TS 23.203 Release 13, 3GPP, 2015.

[24] N. Garg and J. Konemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems," *SIAM J. Comput.*, vol. 37, no. 2, pp. 630–652, 2007.

[25] N. Buchbinder and J. Naor, "The design of competitive online algorithms via a primal-dual approach," *Found. Trends Theoretical Comput. Sci.*, vol. 3, nos. 2–3, pp. 263–293, 2007.

[26] R. Hassin, "Approximation schemes for the restricted shortest path problem," *Math. Oper. Res.*, vol. 17, no. 1, pp. 36–42, 1992.

[27] M. Ziegelmann, "Constrained shortest paths and related problems," Ph.D. dissertation, Int. Max Planck Res. School Comput. Sci., Saarland Univ., Saarland, Germany, 2001.



Zizhong Cao (S'12) received the B.S. degree in electrical engineering from Shanghai Jiao Tong University, China, in 2010, and the Ph.D. degree in electrical engineering from the Tandon School of Engineering, New York University, USA, in 2016.

His research interests include network design and optimization in switching, data center, and mobile networks. He was a co-recipient of the Best Paper Award for the IEEE INFOCOM 2014 and ACM SIGCOMM DCC 2014.

Shivendra S. Panwar (S'82–M'85–SM'00–F'11) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology (IIT) Kanpur, Kanpur, India, in 1981, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Massachusetts, Amherst, MA, USA, in 1983 and 1986, respectively. He is currently a Professor and the Department Chair of Electrical and Computer Engineering with the Tandon School of Engineering, New York University, Brooklyn, NY, USA.

He is also the Director of the New York State Center for Advanced Technology in Telecommunications and a member of NYU WIRELESS. His research interests include the performance analysis and design of networks. Current work includes co-operative wireless networks, switch performance, and multimedia transport over networks.

Dr. Panwar was a recipient of the IEEE Communication Society's Leonard G. Abraham Prize in the Field of Communication Systems for 2004 and the Best Paper Award in Multimedia Communications 2011. He has served as the Secretary of the Technical Affairs Council of the IEEE Communications Society.

Murali Kodialam (A'99–M'14) received the Ph.D. degree from the Massachusetts Institute of Technology. He joined Bell Laboratories in 1992, where he is currently a Distinguished Member of Technical Staff.

He has authored over 80 refereed conference and journal publications, and holds over 40 patents. His general research interests are in resource allocation in communication networks. He has worked on IP routing, switch scheduling and data structures and algorithms for fast packet processing in wired and wireless networks.

Dr. Kodialam was an Associate Editor of the IEEE/ACM TRANSACTIONS ON NETWORKING. He was a co-recipient of the 2008 IEEE Leonard Abraham Prize, the 2011 IEEE Bennett Prize, and the IEEE INFOCOM 2014 best paper awards.

T. V. Lakshman (M'85–SM'98–F'05) received the master's degree in physics from the Indian Institute of Science, Bangalore, India, in 1984, and the Ph.D. degree in computer science from the University of Maryland, College Park, in 1986.

He is currently the Head of the Networks Research Group at Nokia Bell Laboratories. His research contributions span a spectrum of networking topics, including switch architectures, network design, TCP performance, traffic management, and software-defined networking.

Dr. Lakshman is a Fellow of Bell Laboratories and ACM. He was a recipient of several IEEE and ACM awards, including the IEEE Leonard Abraham Prize, the IEEE Communication Society William R. Bennett Prize, the IEEE INFOCOM Achievement award, the IEEE Fred W. Ellersick Prize Paper Award, and the ACM SIGMETRICS Best Paper Award. He also received the 2010 Thomas Alva Edison Patent Award from the R&D Council of New Jersey. He has been an Editor of the IEEE/ACM TRANSACTIONS ON NETWORKING and the IEEE TRANSACTIONS ON MOBILE COMPUTING.