The Critical Network Flow Problem: Migratability and Survivability

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

***Abstract*—In this paper, we propose a new network abstraction, termed *critical network flow*, which models the bandwidth requirement of modern internet applications and services. A critical network flow defines a conventional flow in a network with explicit requirement on its aggregate bandwidth, or the flow value as commonly termed. Unlike common bandwidth-guaranteed connections whose bandwidth is only guaranteed during normal operations, a critical network flow demands strictly enforced bandwidth guarantee during various transient network states, such as network reconfiguration or network failures. Such a demand is called the *bandwidth criticality* of a critical network flow, which is characterized both by its flow value and by its capability to satisfy bandwidth guarantee in the transient states. We study algorithmic solutions to the accommodation of critical network flows with different bandwidth criticalities, including the basic case with no transient network state considered, the case with network reconfiguration, and the case with survivability against link failures. We present a polynomial-time optimal algorithm for each case. For the survivable case, we further present a faster heuristic algorithm. We have conducted extensive experiments to evaluate our model and validate our algorithms.**

***Keywords*—Critical network flow, traffic engineering, bandwidth guarantee, flow migration, survivability**

I. Introduction

Networks have become an integrated component of modern computing infrastructures, due to the rapid advances of network-based computing paradigms, such as cloud computing, mobile edge computing, etc. As an upside, networks bring scalability and robustness, advancing the computation power beyond the individual chips. Yet as a downside, networks also bring hardness in performance isolation and guarantee, due to their shared nature and the competition among entities. Network congestion can greatly degrade the performance of overlying services, affecting user experience and the profitability of the service providers.

This has motivated a large body of work on network performance guarantee in various network scenarios, including load balancing, traffic engineering, network virtualization (virtual network embedding), etc. Among these, *traffic engineering* (TE) is one of the most attended approaches, due to its model simplicity (compact representation of different traffic as network flows), problem tractability in theory (polynomial-time solvability when flows are splittable), and practicality for implementation (via existing protocols like Multi-Protocol Label Switching Traffic Engineering (MPLS-TE) [35] or systems like Software-Defined Networking (SDN) [24]). In plain words, TE jointly plans end-to-end routes and per-route bandwidth allocation for each connection, hence achieving both route flexibility and total bandwidth guarantee.

Existing TE solutions focus on guaranteeing bandwidth in the stable state. This means that while each connection has guaranteed bandwidth in the long run, it may temporarily suffer from arbitrarily bad performance during transient network states. These states include but are not limited to network reconfiguration and network failures. Existing researches reveal that transient states can have a great impact on the performance of many network-based services [14], [20].

In this paper, we investigate how to provide proper bandwidth guarantee during transient network states. We start by proposing bandwidth-guaranteed network flows as a new model for network traffic engineering. A network flow corresponds to a user traffic request (termed a *commodity*) in the network. We call a commodity with explicit bandwidth requirement a *critical commodity*, and the corresponding network flow a *critical network flow* (short as a *critical flow*). While a traditional commodity is merely defined by its ingress and egress points, a critical commodity has two additional dimensions: its *bandwidth requirement*, and its *criticality requirement*. The criticality requirement of a commodity is defined as the set of transient states (in addition to the stable state) in which the commodity's bandwidth guarantee must be enforced. Therefore, each critical commodity essentially asks for strict bandwidth guarantee, both when the network is stable and when the network is undergoing any transient state specified in its criticality requirement.

Based on these concepts, we further study how to accommodate critical commodities in an online network system, where (both critical and non-critical) commodities arrive randomly over time. Starting from a basic model where no transient state is considered, we build solutions for accommodating critical commodities with network reconfiguration requirements (migratability requirements), and with network link failure requirements (survivability requirements). We present mathematical formulations of these problems, and propose polynomial-time optimal algorithms per accommodation. For the latter problem with a high-complexity optimal algorithm (accommodation with survivability requirements), we further propose a heuristic algorithm. Important performance metrics of the proposed solutions have been evaluated through extensive simulation experiments.

To summarize, our contributions are as follows:

- We propose a novel network abstraction, the *critical network flow* (short as *critical flow*), to model the demand

Yu, Xue and Zhang ({ruozhouy, xue, xzhan229}@asu.edu) are all with Arizona State University, Tempe, AZ 85287. All correspondences should be addressed to Guoliang Xue. This research was supported in part by NSF grants 1421685, 1461886 and 1704092. The information reported here does not reflect the position or the policy of the federal government.

for bandwidth guarantee of user traffic, over different transient network states in addition to the stable state.
- We study the online accommodation of critical commodities with two kinds of criticality requirements: migratability requirement and survivability requirement. We propose polynomial-time optimal and faster heuristic solutions to accommodate each arriving critical commodity.
- We evaluate the performance of our model and algorithms via extensive simulation experiments.

The rest of this paper is organized as follows. We introduce existing work related to this paper in Section II. We present our system model, including the proposed critical flow abstraction, in Section III. We then illustrate the basic model for critical flow accommodation, with no consideration of transient states, in Section IV. We illustrate the accommodation with migratability requirement, where critical commodities can have guaranteed bandwidth during network reconfigurations, in Section V. We further illustrate the accommodation with survivability requirement, where critical commodities can survive an arbitrary single link failure in the network, without having their bandwidth guarantee affected, in Section VI. We present our simulation results for performance evaluation in Section VII. We conclude this paper in Section VIII.

II. Background, Motivation and Related Work

A. Quality-of-Service Routing and Traffic Engineering

Our proposed model is most related to research in the areas of Quality-of-Service (QoS) routing and TE. Both areas seek to find joint routing and bandwidth allocation solutions to ensure throughput of network commodities. QoS routing considers the routing and bandwidth allocation of a single commodity at a time, and has been studied in [6], [10], [23], [38], [44] and other related papers. TE extends QoS routing by considering multiple co-existing commodities, which arrive either at the same time (offline TE) or randomly over time (online TE) [37]. Two types of routing schemes have been studied in TE: single-path routing and multi-path routing. TE with single-path routing allocates bandwidth for each commodity along only one data path, and has been studied in [7], [40] and other related work. On the contrary, TE with multi-path routing can allocate bandwidth on multiple paths simultaneously, and thus enables more flexible utilization of the network bandwidth [17], [22], [32]. While early TE solutions focused on utilizing Equal-Cost Multi-Path (ECMP) for multi-path TE, recent advances in network protocols and systems, like Multipath with MPLS [35] and SDN [24], have enabled more flexible path selection and traffic splitting. This paper falls into the category of online TE with multi-path routing. The reason is that online arrival of commodities is more in line with real network environments, and multi-path routing is more suitable for utilizing the rich path diversity in modern networks [11], [22], [28].

Network flow and multi-commodity flow have been extensively used to model TE with multi-path routing, for example, in [16], [20], [22], [28]. A network flow essentially describes the per-link bandwidth allocation for a commodity, wherein a multi-commodity flow captures the bandwidth sharing of different commodities over the same link. Our method follows existing multi-commodity flow approaches for multi-path TE, but additionally considers bandwidth guarantee during transient network states, which has not been taken into account in the above papers and other related work.

B. Network Reconfiguration

One transient network state considered in this paper is network reconfiguration, when the network updates routing paths and bandwidth allocation to accommodate new commodities or demand changes of existing commodities. Many have focused on different issues arisen in network reconfiguration, including path consistency [30], policy consistency [26], [27], [36], and congestion-free flow migration [5], [14], [15], [21]. The problem we study falls into the category of congestion-free flow migration, which requires that network reconfiguration must not introduce congestion to existing critical flows, and is similarly modeled as in [5], [14], [15]. However, we consider deriving a new network state which can be migrated from the current state without congestion, while [14] and [15] only seek to find congestion-free migration phases given the new network state as input. Reference [5] also studies deriving new network states with congestion-free migration, but their solution may involve an exponential number of migration phases. In practice, this may lead to large migration overhead. Our solutions focus on one-phase migration for the sake of overhead control. In addition, we also consider flow migration during network failures, and pro-actively derive congestion-free migration for failure recovery.

C. Survivable Routing and TE

Multi-path routing naturally provides a certain extent of reliability, in the sense that a single link failure will not fully halt a flow from transmitting, but only degrades its throughput by a certain amount. However, this is not sufficient for critical flows, which demand strictly guaranteed throughput during failures. Two types of survivability mechanisms provide protection against a specific level of failure. *Pro-active* mechanisms provision extra bandwidth initially during commodity accommodation, and use the bandwidth to fast recover affected commodities when failure happens [20]. *Reactive* mechanisms do not provision extra bandwidth, but seek for backup bandwidth only after the failure happens [20]; as a result, reactive mechanisms do not guarantee recovery of the accepted commodities. We focus on pro-active mechanism design in order to satisfy the stringent bandwidth requirement of critical commodities. We also aim to minimize the impact of our survivability mechanism over other unaffected commodities. Specifically, we aim to ensure congestion-free migration of all critical flows during the failure recovery process; we also aim to achieve work conservation by enabling utilization of the pre-provisioned backup resources when no failure is present.

Traditional survivable TE solutions have focused on path-based approaches, where each primary path is accompanied by a backup path, which protects the commodity from different failure scenarios [12], [13], [18], [19], [34]. On the contrary, Acharya *et al.* [2] found out that dedicated path-based protection is commonly an overkill for achieving reliability goals,

and proposed the first flow-based reliability mechanism; their work was further extended by Zhang *et al.* [48] to incorporate differential delay constraints. However, references [2] and [48] do not ensure full bandwidth protection against single link failure. Rather, they focus on bounding the amount of bandwidth loss due to a single link failure. Instead, we aim to fully ensure the throughput of critical commodities during single link failure, yet still avoid the overconsumption of bandwidth for provisioning dedicated backup paths. Liu *et al.* [20] also considered full bandwidth guarantee with pro-active backup provisioning, yet their solution involves pre-computed tunnels for each commodity as input; our solution does not require pre-computation of pairwise tunnels. In addition, we consider congestion-free migration of critical flows during the failure recovery process that further enforces critical commodities' bandwidth requirements, which is different from all the above and other related work.

Other than TE, survivable routing has been studied in the context of optical networks [25], [29], [31], [41]–[43]. Similar to the above, these studies also focus on providing "all-or-nothing" protection using disjoint routing paths. Recently, the concept of *tunable survivability* has been proposed, which provides a quantitative measure of routing survivability based on the failure probability of links [3], [45], [46]. However, the intrinsic difference between our study and the above is that we consider bandwidth protection in addition to route protection. In our model, critical commodities ask for bandwidth guarantee during different network states. Hence these existing results do not apply in our model.

Another related problem is to test end-to-end connectivity of a given network, which can be used for failure detection. This problem has been studied in recent papers [8], [9].

III. System Model

Network model: We consider a network denoted by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of network nodes, and $\mathcal{E}$ is the set of links. We assume that the network has $|\mathcal{V}| = n$ nodes and $|\mathcal{E}| = m$ links in total. For $v \in \mathcal{V}$, we use $in(v) \subseteq \mathcal{E}$ and $out(v) \subseteq \mathcal{E}$ to denote the sets of incoming and outgoing links of node v respectively. Each link $e \in \mathcal{E}$ is associated with a capacity, denoted by $c_e > 0$. In the following, we use the terms "vertex" and "node" interchangeably; similarly, "edge" and "link" are used interchangeably.

Request model: The network accepts data transfer requests from network users. Each user request, termed a *commodity*, is defined as a tuple $(s, t; d)$, where $s, t \in \mathcal{V}$ are the source and destination nodes respectively, and $d \geq 0$ is the request's bandwidth demand. Specifically, if a commodity has $d > 0$, the commodity is a *critical commodity*, which asks for bandwidth guarantee of d. If a commodity has $d = 0$, the commodity is a *non-critical commodity*, and can be served in best efforts.

Without loss of generality, we assume that all critical commodities in the network share the same criticality requirement, *i.e.*, the network decides which transient states to be covered for all critical commodities. However, this is only for ease of illustration, and our solutions can be easily extended to cover different transient states for different critical commodities.

In practice, bandwidth criticality can be implemented using packet prioritization and rate limiting. Prioritization is enforced such that critical commodity packets have strictly higher priority than non-critical commodity packets. By limiting the maximum rate of each commodity along each path, congestion can be avoided if the allocated bandwidth enforces link bandwidth capacity constraints. Rate limiting can be employed either at the data source (application- or operating system-level rate limiting) or at the ingress switch (e.g., using OpenFlow [24]).

Flow model: We assume that the network is capable of multi-path routing. To accommodate a commodity, the network needs to determine the routing paths and the bandwidth allocation for each path. The *network flow* abstraction captures both characteristics, and is defined as follows:

Definition 3.1: A *single-commodity flow* (or a *flow*) regarding commodity $K = (s, t; d)$ is defined by a mapping $F : \mathcal{E} \mapsto \mathbb{R}^*$ (where $\mathbb{R}^*$ is the non-negative real number set) that satisfies the following constraints:

- **Flow conservation:** For $\forall v \in \mathcal{V} \setminus \{s, t\}$, $\sum_{e \in in(v)} F(e) - \sum_{e \in out(v)} F(e) = 0$;
- **Capacity:** $F(e) \leq c_e$ for $\forall e \in \mathcal{E}$;
- **Bandwidth guarantee:** If K is critical ($d > 0$), then $\sum_{e \in in(t)} F(e) - \sum_{e \in out(t)} F(e) = d$.

The value $f(F) \triangleq \sum_{e \in in(t)} F(e) - \sum_{e \in out(t)} F(e)$ is called the *flow value* of flow F. □

Let $\mathcal{K} = \langle K_1, \ldots, K_k \rangle$ be an ordered set of commodities. Note that the order is only for ease of illustration, and does not impair the generality of our model. The corresponding flows of the commodities are defined as follows:

Definition 3.2: A *multi-commodity flow* regarding commodities $\mathcal{K} = \langle K_1, \ldots, K_k \rangle$ is defined by an ordered set $\mathcal{F} = \langle F_1, \ldots, F_k \rangle$, where for $i = 1, 2, \ldots, k$, $F_i : \mathcal{E} \mapsto \mathbb{R}^*$ is a flow for commodity K_i as defined in Definition 3.1, and jointly all flows in $\mathcal{F}$ satisfy the following constraint:

- **Joint capacity:** $\sum_{i=1}^{k} F_i(e) \leq c_e$ for $\forall e \in \mathcal{E}$.

We use $\boldsymbol{f}(\mathcal{F}) = \langle f_1, \ldots, f_k \rangle$ to denote the corresponding flow values of the multi-commodity flow, where $f_i = f(F_i)$ is the flow value of F_i, for $i = 1, 2, \ldots, k$. □

For simplicity, we use $\boldsymbol{f}$ instead of $\boldsymbol{f}(\mathcal{F})$ if no ambiguity is introduced. We also assume that if the corresponding commodity $K_i \notin \mathcal{K}$, then $F_i(e) = 0$ for $\forall e \in \mathcal{E}$, and the corresponding flow value $f_i = 0$.

Service model: We assume that the network operates in an online manner. User commodities arrive randomly throughout time. Before a commodity's arrival, no knowledge about its characteristics is known to the system. Hence the network processes commodities one-by-one in the order of their arrivals, based on the network states at their arrivals respectively. We next define the network state at an arbitrary time.

Definition 3.3: The *network state* at an arbitrary time is defined by a tuple $S = (\mathcal{G}, \mathcal{K}, \mathcal{F})$, where $\mathcal{G}$ is the network topology, $\mathcal{K}$ is the set of currently accommodated commodities, and $\mathcal{F}$ is the corresponding multi-commodity flow of $\mathcal{K}$. We use $\mathcal{N} \subseteq \mathcal{K}$ to denote the set of non-critical commodities, and $\mathcal{C} \subseteq \mathcal{K}$ to denote the set of critical commodities. □

Given the current network state S, and a new critical commodity K^+, the critical flow problem is to find a new network state S^+, which accommodates K^+ with bandwidth guarantee, and does not affect any existing critical commodity's guarantee, meanwhile achieving certain network goals and properties. Non-critical commodities can always be accommodated in best efforts with the remaining bandwidth left by the critical commodities. We use $\mathcal{K}^+ = \mathcal{K} \oplus K^+$ to denote the new set of commodities after the arrival of K^+ ($\oplus$ as the insert-at-tail operator of an ordered set), and $\mathcal{C}^+ = \mathcal{C} \oplus K^+$ to denote the new critical commodity set; $\mathcal{N}^+ = \mathcal{N}$ for the purpose of this problem, hence is not explicitly marked. Next we study different versions of the critical flow problem.

IV. The Critical Flow Problem: Basic Case

We begin with the basic critical flow problem. Our first-class goal is to secure sufficient bandwidth for each critical commodity, including the newly arrived one. In the mean time, we want to achieve work conservation by maximizing non-critical flows given the satisfaction of all critical flows. The basic critical flow problem is defined as follows:

Definition 4.1: Given the current (old) network state $S = (\mathcal{G}, \mathcal{K}, \mathcal{F})$, and a new critical commodity K^+, the basic critical flow problem **CFP**$(\mathcal{G}, \mathcal{K}, \mathcal{F}, K^+)$ seeks a new network state $S' = (\mathcal{G}, \mathcal{K}^+, \mathcal{F}^+)$, such that 1) for each critical commodity $K_i \in \mathcal{C}^+$, its flow value satisfies its bandwidth requirement, i.e., $f_i = d_i$, and 2) the sum of flow values of all non-critical commodities $\mathcal{N}$ is maximized. □

Let $\mathcal{F}^+ = \langle F_1^+, \ldots, F_k^+ \rangle$ be the (ordered) set of variables for per-commodity flow assignments, and $\boldsymbol{f} = \langle f_1^+, \ldots, f_k^+ \rangle$ be the flow value variables, where $k = |\mathcal{K}^+|$ is the number of commodities (including the new one), we formulate the basic critical flow problem as follows:

Program (1)**: CFP**$(\mathcal{G}, \mathcal{K}, \mathcal{F}, K^+)$

Variables:

$F_i^+(e)$	new flow assignment of $K_i \in \mathcal{K}^+$ on e
f_i^+	flow value of $K_i \in \mathcal{N}$

$$\max \sum_{K_i \in \mathcal{N}} f_i^+ \tag{1a}$$

s.t. $\forall K_i \in \mathcal{K}^+, \forall v \in \mathcal{V} \setminus \{s_i, t_i\}$:

$$\sum_{e \in in(v)} F_i^+(e) - \sum_{e \in out(v)} F_i^+(e) = 0; \tag{1b}$$

$\forall K_i \in \mathcal{C}^+$:

$$\sum_{e \in in(t_i)} F_i^+(e) - \sum_{e \in out(t_i)} F_i^+(e) = d_i; \tag{1c}$$

$\forall K_i \in \mathcal{N}$:

$$\sum_{e \in in(t_i)} F_i^+(e) - \sum_{e \in out(t_i)} F_i^+(e) = f_i^+; \tag{1d}$$

$$\forall e \in \mathcal{E} : \sum_{K_i \in \mathcal{K}^+} F_i^+(e) \le c_e; \tag{1e}$$

$$\forall K_i \in \mathcal{K}^+, \forall e \in \mathcal{E} : \; F_i^+(e) \ge 0, f_i^+ \ge 0. \tag{1f}$$

Explanation: The goal of this program is to guarantee bandwidth for critical flows, meanwhile maximizing non-critical flows. Objective (1a) is to maximize the sum of flow values of all non-critical commodities. Constraint (1b) defines flow conservation at each node. Constraint (1c) defines the flow value of critical commodity $K_i \in \mathcal{C}^+$ to be exactly the bandwidth demand d_i. Constraint (1d) further defines the flow value of each non-critical commodity $K_i \in \mathcal{N}$, which is to be maximized in the objective. Constraint (1e) specifies the per-link bandwidth capacity constraints. Constraint (1f) defines the range of each variable.

Remark 4.1: In the **CFP** formulation (Program (1)), critical commodities have higher priority than non-critical commodities, as we enforce strict bandwidth guarantee for critical commodities in Constraint (1c), while maximizing throughput of non-critical commodities only when all critical commodities are satisfied. □

We propose a solution to **CFP** in the following theorem:

Theorem 4.1: CFP can be optimally solved in $O((km + k)^3\mathcal{L})$ time, where m is the number of links, k is the number of commodities, and $\mathcal{L}$ is the input size. □

Proof: The **CFP** problem, as formulated in Program (1), is a linear program with $O(km+k)$ variables. Based on existing solution [47], it can be solved in $O((km + k)^3\mathcal{L})$ time. ■

If the program returns a feasible solution, the new critical commodity is then accommodated based on the flow assignments; otherwise, the commodity is rejected by the network due to lack of sufficient bandwidth.

V. The Critical Flow Problem: Migratability

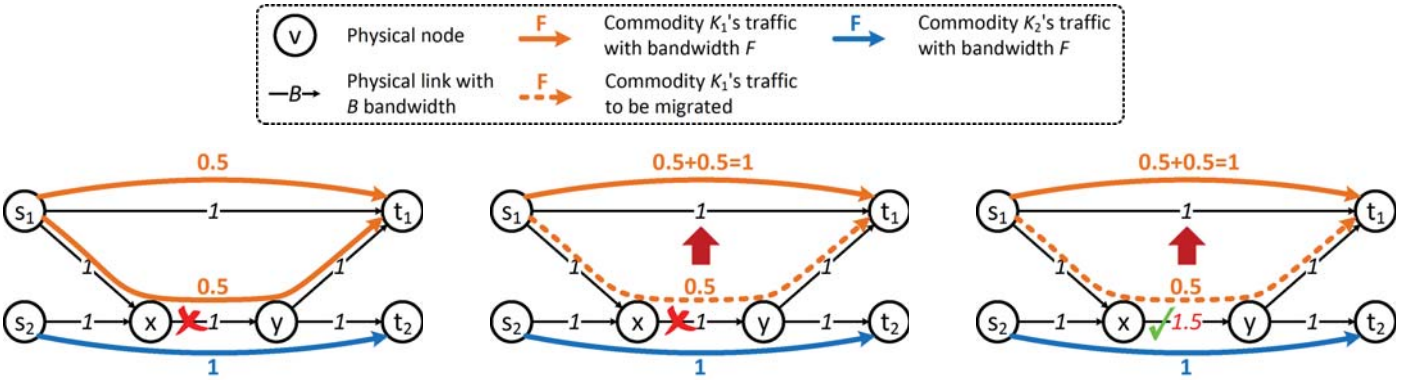
In the last section, we present a basic formulation and solution for the critical flow problem, where bandwidth criticality is only guaranteed in both the old and the new network states, but not in between. During the migration process, congestion may happen due to the asynchronous update of flows [5], [14], which leads to violated bandwidth criticality.

In this section, we study the critical flow problem with congestion-free migration (also termed *migratability requirement* in this paper). A congestion-free migration ensures that during the migration process, no congestion happens on any link. We aim to find a new network state that allows congestion-free migration for critical commodities, meanwhile minimizing the migration cost and maximizing the throughput of non-critical flows. Hence both the new critical commodity and the existing critical commodities will have their bandwidth requirements satisfied during the migration.

A. Congestion-free Migration

The migration of flows incurs changing network paths for individual commodities. Since each commodity may adjust its paths asynchronously, a link may carry both the old flows and the new flows at the same time. This may cause congestion if the load on some link is temporarily higher than its capacity.

Fig. 1 shows an example of possible congestion during flow migration and a congestion-free migration. In a network of 6 nodes, a commodity $K_1 = (s_1, t_1; 1)$ already exists, with its flow consisting of two paths: $p_1 = (s_1 \to t_1)$ and $p_2 =$



(a) Congestion due to new request (b) Congestion during flow migration (c) Congestion-free flow migration

Fig. 1: Example of (a) congestion on link (x, y) due to arrival of a new commodity $K_2 = (s_2, t_2; 1)$, (b) congestion due to asynchronous migration of flows for K_1 and K_2, and (c) congestion-free migration when (x, y) has more bandwidth.

$(s_1 \to x \to y \to t_1)$, both with 0.5 bandwidth. Now, a new commodity $K_2 = (s_2, t_2; 1)$ arrives in Fig. 1(a). If simply adopted to the only path $p_3 = (s_2 \to x \to y \to t_2)$ that it can take, K_2 will cause congestion on link (x, y), downgrading the performance of K_1, as in Fig. 1(a). To avoid this issue, commodity K_1's traffic on p_2 (which shares link (x, y) with K_2's path) has to be migrated to p_1, as in Fig. 1(b). However, while this migration avoids congestion between K_1 and K_2 in the long term, it still does not guarantee congestion-freeness during the migration, due to the possible asynchrony between K_1's and K_2's migrations. Assume that K_2 starts transmission on path p_3 before K_1 can migrate its flow on path p_2. This still causes congestion on link (x, y). The only way to provide congestion-free migration is shown in Fig. 1(c). By enlarging link (x, y)'s bandwidth to 1.5, now even when K_2 transmits before the completion of K_3's migration, the total bandwidth of their flows are still no greater than 1.5, hence avoiding congestion even during the migration. Essentially, each link's capacity should be sufficient to cover the maximum bandwidth between the old and the new flows for any commodity whose either flow uses the link, in order to ensure a congestion-free migration.

In practice, enlarging network capacity during runtime is not realistic. To avoid congestion, the network then needs to carefully provision the new network state, such that the capacity constraint of each link is enforced when the network flows are updated in arbitrary order. Specifically, we define a congestion-free migration as follows:

Definition 5.1: Given two network states $S = (\mathcal{G}, \mathcal{K}, \mathcal{F})$ and $S' = (\mathcal{G}, \mathcal{K}', \mathcal{F}')$, a congestion-free migration exists from S to S' iff $\sum_{K_i \in \mathcal{C} \cup \mathcal{C}'} \max\{F_i(e), F_i'(e)\} \leq c_e$ for $\forall e \in \mathcal{E}$. □

Note that our definition follows from the consistent migration definition in [5], [14]. We only consider 1-phase migration of flows, meaning that only two network states are considered: the old state and the new state. Involving multiple migration phases may result in better chance of successful accommodation, but it also brings undesirable overheads including reconfiguration costs and prolonged flow switching time, as well as large overhead for computations. Nevertheless, our approach can be easily extended to consider up to a constant number of phases, where multiple intermediate states may be generated; the formulation size, however, increases linearly with the number of phases considered, as shown in [14].

B. Optimization Objectives

We consider a two-layer optimization objective for accommodating the new critical flow with congestion-free migration. Even when the migration process is congestion-free, network state migration can still incur overhead and potential performance loss for the critical commodities. Therefore, we aim to minimize the flow offsets in the migration. We introduce the *migration cost* of critical flows, defined as follows:

Definition 5.2: Let each link $e \in \mathcal{E}$ be associated with a non-negative unit migration cost π_e, which is the cost for changing unit flow value for any critical commodity on this link, set by the network operator. Let S and S' be the network states before and after migration respectively. For each critical commodity $K_i \in \mathcal{C} \cup \mathcal{C}'$ and each link $e \in \mathcal{E}$, the *flow offset* of K_i on e is defined by $\phi_i(e) \triangleq |F_i(e) - F_i'(e)|$. The *migration cost* of commodity K_i on e is then $\pi_e \phi_i(e)$. □

Our primary objective is to minimize the sum of migration costs over all commodities on all links.

Given the minimum migration cost of critical flows, our secondary objective is to maximize the throughput of the non-critical flows, in order to achieve work conservation and utilize the remaining network resources as much as possible.

C. Problem Formulation

Given the above, the critical flow problem with migratability requirement is defined as follows:

Definition 5.3: Given the current (old) network state $S = (\mathcal{G}, \mathcal{K}, \mathcal{F})$, and a new critical flow K^+, the critical flow problem with migratability requirement, **MCFP**$(\mathcal{G}, \mathcal{K}, \mathcal{F}, K^+)$, seeks a new network state $S' = (\mathcal{G}, \mathcal{K}^+, \mathcal{F}^+)$, such that 1) for each critical commodity $K_i \in \mathcal{C}^+$, its flow value satisfies its bandwidth requirement, i.e., $f_i = d_i$, 2) each critical commodity $K_i \in \mathcal{C}$ can be migrated from S to S' without congestion as defined in Definition 5.1, 3) the solution minimizes the total migration cost as primary objective, and 4)

the solution maximizes the total throughput of all non-critical commodities as secondary objective. □

Since this is a multi-objective optimization problem, we propose the following hierarchical formulation for the problem.

Program (2): MCFP$(\mathcal{G}, \mathcal{K}, \mathcal{F}, K^+)$

Variables:

$F_i^+(e)$	new flow assignment of $K_i \in \mathcal{K}^+$ on e
$\phi_i(e)$	flow change of $K_i \in \mathcal{C}^+$ on link e
f_i^+	flow value of $K_i \in \mathcal{N}$

$$\min\ \Phi^* = \sum_{K_i \in \mathcal{C}^+} \sum_{e \in \mathcal{E}} \pi_e \phi_i(e) \quad \text{(primary)} \tag{2a}$$

$$\max \sum_{K_i \in \mathcal{N}} f_i^+ \quad \text{(secondary)} \tag{1a}$$

s.t. (1b), (1c), (1d), (1e)

$$\forall e \in \mathcal{E}: \sum_{K_i \in \mathcal{C}^+} \max\{F_i(e), F_i^+(e)\} \le c_e; \tag{2b}$$

$$\forall K_i \in \mathcal{C}^+, \forall e \in \mathcal{E}: \ \phi_i(e) = |F_i^+(e) - F_i(e)|; \tag{2c}$$

$$\forall K_i \in \mathcal{K}^+, \forall e \in \mathcal{E}: \ F_i^+(e), f_i^+, \phi_i(e) \ge 0. \tag{2d}$$

Explanation: The program is a hierarchical formulation with two layers of objectives. It inherits the variable set of Program (1), with additional variables $\phi_i(e)$ to denote the per-commodity per-link flow changes. As the primary objective, it minimizes migration cost defined by Φ^* in (2a); as the secondary objective, it inherits the maximization of non-critical throughput as in Program (1). Other than all constraints in Program (1), Program (2) has two more sets of constraints. Constraint (2b) enforces per-link bandwidth capacity constraints regarding all critical commodities, and is defined using the congestion-free definition as in Definition 5.1; note that it automatically incorporates the joint capacity constraint of $\mathcal{F}^+$ as in Definition 3.2. Note that by inheriting Constraint (1e), only the new flows of the critical commodities are considered when sharing with the non-critical commodities. This is because non-critical commodities have no impact on the migration of critical commodities due to packet prioritization. Constraint (2c) defines the per-commodity per-link flow changes, which is used to define the primary objective function. Constraint (2d) specifies the range of each variable.

Remark 5.1: Solving a two-layer optimization problem involves two steps. In the first step, the program with the primary objective and all constraints are optimized. Given the optimal primary objective, in the second step, the program with the secondary objective, all listed constraints, and an additional optimality constraint (enforcing that the primary objective function is equal to the obtained optimal primary objective in the first step), is solved, producing the final solution. □

D. Solution

Note that in the above formulation, both the link capacity constraint (2b) and the flow change constraint (2c) are non-linear non-differentiable convex constraints. Non-differentiable convex optimization can be solved using existing algorithms [4], but such process can be computation-intensive and time-consuming. Instead, we propose linearization of the above constraints to reduce complexity.

Specifically, Constraint (2b) can be linearized by introducing additional variables $\Psi_i(e) \ge 0$ for $K_i \in \mathcal{C}^+$ and $e \in \mathcal{E}$, and replacing Constraint (2b) by the following constraints:

$$\Psi_i(e) \ge F_i(e), \quad \forall K_i \in \mathcal{C}^+, \forall e \in \mathcal{E} \tag{3a}$$

$$\Psi_i(e) \ge F_i^+(e), \quad \forall K_i \in \mathcal{C}^+, \forall e \in \mathcal{E} \tag{3b}$$

$$\sum\nolimits_{K_i \in \mathcal{C}^+} \Psi_i(e) \le c_e, \quad \forall e \in \mathcal{E} \tag{3c}$$

Constraint (2c) can be linearized by replacing them with the following constraints:

$$\phi_i(e) \ge F_i^+(e) - F_i(e), \quad \forall K_i \in \mathcal{C}^+, \forall e \in \mathcal{E} \tag{4a}$$

$$\phi_i(e) \ge F_i(e) - F_i^+(e), \quad \forall K_i \in \mathcal{C}^+, \forall e \in \mathcal{E} \tag{4b}$$

We then propose a solution to **MCFP** in Theorem 5.1:

Theorem 5.1: MCFP can be optimally solved in $O((3km+k)^3\mathcal{L})$ time, where m is the number of links, k is the number of commodities, and $\mathcal{L}$ is the input size. □

Proof: After linearization, the **MCFP** formulation (Program (2)) is a two-layer linear program with $O(3km+k)$ variables. As mentioned in Remark 5.1, the problem can be solved in two steps, both solving a linear program with the same number of variables. Based on existing solution [47], both programs can be solved in $O((3km+k)^3\mathcal{L})$ time. ■

If both programs return feasible solutions, the new critical commodity is accommodated based on the flow assignments output by the second program; otherwise, it is rejected by the network due to lack of sufficient bandwidth.

VI. The Critical Flow Problem: Survivability

In this section, we further advance critical bandwidth guarantees to let them persist during an arbitrary single link failure. We propose to proactively determine the backup paths and bandwidth against each potential link failure for the critical commodities. In the mean time, we also minimize the impact of bandwidth over-provisioning on the non-critical commodities, hence achieving work conservation as well.

A. Survivable Critical Flows

To ensure survivability, each critical commodity must receive sufficient bandwidth even when any link failure happens. We use a pro-active mechanism to achieve this goal. Specifically, during the accommodation of each critical commodity, in addition to deciding its primary data paths and bandwidth allocation, the network also decides the backup paths and allocation against each possible link failure. During a link failure, the affected traffic along the link will be detoured to the paths with pre-reserved backup bandwidth, hence the bandwidth guarantee of the commodity is secured.

Formally, we extend the network flow abstraction to the following *survivable network flow* abstraction:

Definition 6.1: A *survivable flow* regarding commodity $K = (s, t; d)$ is defined by a vector of $m+1$ mappings $P = (W, R^{e_1}, \cdots, R^{e_m})$ (m is the number of links in $\mathcal{E}$), where $W : \mathcal{E} \mapsto \mathbb{R}^*$ is a single-commodity flow for K during

normal operations (with no failure), while $R^\eta : \mathcal{E} \mapsto \mathbb{R}^*$ is a single-commodity flow for K during link η's failure for $\forall \eta \in \mathcal{E}$, such that, 1) $R^\eta(\eta) = 0$ for $\forall \eta \in \mathcal{E}$, and 2) each flow has flow value $f(W) = d$ and $f(R^\eta) = d$. □

For ease of explanation, we term the flow W as the *working flow*, and R^η as the *recovery flow against failure* η for $\eta \in \mathcal{E}$.

Definition 6.1 ensures that, during an arbitrary link failure in the network, the critical commodity K is always guaranteed with a single-commodity flow with throughput d as demanded.

B. Network Sharing and Work Conservation

We are interested in finding a survivable flow for each critical commodity. We choose not to offer pro-active survivability to non-critical commodities, instead providing them best-effort survivability by reactively rerouting their traffic when failure happens. Hence non-critical commodities have only the working flows, and no recovery flow, during accommodation. Fast rerouting of non-critical traffic is out of the scope of this paper.

The reason that we distinguish between the working flows and the recovery flows in Definition 6.1 is to enable network sharing both between critical and non-critical commodities and among different critical commodities. On one hand, the recovery flow ought to be reserved dedicatedly to its critical commodity, in order to enable immediate and congestion-free flow recovery during a link failure; if the recovery bandwidth is used by other critical traffic, congestion may be introduced which affects the performance guarantee of the critical flow. On the other hand, since network failure is a relatively rare event, keeping the backup bandwidth idle will lead to under-utilization of the network when no failure happens.

As a trade-off, we allow non-critical commodities to utilize the backup bandwidth of the critical commodities during normal operations. Due to strict packet prioritization between critical and non-critical commodities, when traffic of a critical commodity is detoured (due to failure) to the recovery flow utilized by some non-critical commodity, the reserved bandwidth will be preferentially dedicated to the critical traffic, fulfilling its bandwidth guarantee. This achieves both guaranteed survivability of the critical flows and work conservation.

Based on the above, we extend the multi-commodity flow defined in Section III to incorporate both the working flows and the per-failure recovery flows:

Definition 6.2: A *survivable multi-commodity flow*, regarding an ordered set of commodities $\mathcal{K} = \langle K_1, \ldots, K_k \rangle$ with critical commodity set $\mathcal{C}$ and non-critical commodity set $\mathcal{N}$, is defined by another ordered set $\mathcal{P} = \langle P_1, \ldots, P_k \rangle$, where for critical commodity $K_i \in \mathcal{C}$, P_i is a survivable flow as defined in Definition 6.1, and for non-critical commodity $K_i \in \mathcal{N}$, P_i (or equivalently W_i) is a single-commodity flow as defined in Definition 3.1, such that, all elements in $\mathcal{P}$ jointly satisfy the following requirements:

1) **Working capacity:** $\sum_{K_i \in \mathcal{K}} W_i(e) \leq c_e$ for $\forall e \in \mathcal{E}$;
2) **Backup capacity:** $\sum_{K_i \in \mathcal{C}} R_i^\eta(e) \leq c_e$ for $\forall e, \eta \in \mathcal{E}, e \neq \eta$. □

Essentially, the capacity constraint of a conventional multi-commodity flow (Definition 3.2) is now split into m constraints for each link, where the first constraint concerns the working flows of all commodities, and the other constraints concern the recovery flows of critical commodities only.

We then extend the network state definition in Section III:

Definition 6.3: The *network state* at an arbitrary time is defined by a tuple $S = (\mathcal{G}, \mathcal{K}, \mathcal{P})$, where $\mathcal{G}$ is the network topology, $\mathcal{K}$ is the set of currently accommodated commodities, and $\mathcal{P}$ is the survivable multi-commodity flow for $\mathcal{K}$. □

C. Migration Cost

Similar to Section V, we wish to minimize the migration cost of critical commodities as our primary objective, and maximize the non-critical flows as our secondary objective. While the secondary objective takes the same form as before, the primary objective is slightly different due to the over-provisioning of critical flows. Due to the rarity of network failures [33], we assume that *no failure happens during the migration of network states due to arrival of a new critical commodity*. Therefore, while each existing critical commodity K_i is allocated with a survivable flow P_i, only the old (resp. new) working flow W_i (resp. W_i^+) carries actual traffic before (resp. after) the migration. In this case, only traffic along the working flow needs to be migrated and incurs migration costs, while the recovery flows only need to be updated without actually influencing any traffic.

Note that other than the reconfiguration caused by accommodation of a new critical commodity, migration also happens when primary critical flows need to be migrated to recovery flows due to a network failure. The migration of the affected critical flows must not degrade other non-affected critical flows. Congestion-free migration from primary flows to recovery flows ensures this property. However, since network failure is in general rare, we do not consider such migration in the computation of the migration cost.

D. Problem Formulation

Definition 6.4: Given the current (old) network state $S = (\mathcal{G}, \mathcal{K}, \mathcal{P})$, and a new critical flow K^+, the critical flow problem with survivability requirement, **SCFP**$(\mathcal{G}, \mathcal{K}, \mathcal{P}, K^+)$, seeks a new network state $S' = (\mathcal{G}, \mathcal{K}^+, \mathcal{P}^+)$, such that 1) $\mathcal{P}$ satisfies the survivability guarantee of all critical commodities as defined in Definition 6.2, 2) the working flow W_i of each critical commodity $K_i \in \mathcal{C}$ can be migrated from S to S' without congestion as defined in Definition 5.1, 3) at any failure $\eta \in \mathcal{E}$, each critical flow can be migrated to its recovery flow without congestion, 4) the solution minimizes the total migration cost as primary objective, and 5) the solution maximizes the total throughput of non-critical commodities during normal operations as secondary objective. □

A two-layer formulation is proposed for this problem:

Program (5)**: SCFP**$(\mathcal{G}, \mathcal{K}, \mathcal{P}, K^+)$

Variables:

$W_i^+(e)$	new working flow of $K_i \in \mathcal{K}^+$ on e
$R_i^{+,\eta}(e)$	recovery flow of $K_i \in \mathcal{C}^+$ on e when η fails ($e \neq \eta$)
$\phi_i(e)$	flow change of $K_i \in \mathcal{C}^+$ on link e
f_i^+	flow value of $K_i \in \mathcal{N}$

$$\min\ \Phi^* = \sum_{K_i \in \mathcal{C}^+} \sum_{e \in \mathcal{E}} \pi_e \phi_i(e) \quad \text{(primary)} \tag{2a}$$

$$\max \sum_{K_i \in \mathcal{N}} f_i^+ \quad \text{(secondary)} \tag{1a}$$

s.t.
$$\forall K_i \in \mathcal{K}^+, \forall v \in \mathcal{V} \setminus \{s_i, t_i\}:$$
$$\sum_{e \in in(v)} W_i^+(e) - \sum_{e \in out(v)} W_i^+(e) = 0; \tag{5a}$$
$$\forall K_i \in \mathcal{C}^+:$$
$$\sum_{e \in in(t_i)} W_i^+(e) - \sum_{e \in out(t_i)} W_i^+(e) = d_i; \tag{5b}$$
$$\forall K_i \in \mathcal{N}:$$
$$\sum_{e \in in(t_i)} W_i^+(e) - \sum_{e \in out(t_i)} W_i^+(e) = f_i^+; \tag{5c}$$
$$\forall K_i \in \mathcal{C}^+, \forall \eta \in \mathcal{E}, \forall v \in \mathcal{V} \setminus \{s_i, t_i\}:$$
$$\sum_{\substack{e \in in(v) \\ e \neq \eta}} R_i^{+,\eta}(e) - \sum_{\substack{e \in out(v) \\ e \neq \eta}} R_i^{+,\eta}(e) = 0; \tag{5d}$$
$$\forall K_i \in \mathcal{C}^+, \forall \eta \in \mathcal{E}:$$
$$\sum_{\substack{e \in in(t_i) \\ e \neq \eta}} R_i^{+,\eta}(e) - \sum_{\substack{e \in out(t_i) \\ e \neq \eta}} R_i^{+,\eta}(e) = d_i; \tag{5e}$$
$$\forall e \in \mathcal{E}: \sum_{K_i \in \mathcal{C}^+} \max\{W_i(e), W_i^+(e)\} \leq c_e; \tag{5f}$$
$$\forall e \in \mathcal{E}, \forall \eta \in \mathcal{E}, e \neq \eta:$$
$$\sum_{K_i \in \mathcal{C}^+} \max\{W_i^+(e), R_i^{+,\eta}(e)\} \leq c_e; \tag{5g}$$
$$\forall e \in \mathcal{E}: \sum_{K_i \in \mathcal{K}^+} W_i^+(e) \leq c_e; \tag{5h}$$
$$\forall K_i \in \mathcal{C}^+, \forall e \in \mathcal{E}: \ \phi_i(e) = |W_i^+(e) - W_i(e)|; \tag{5i}$$
$$\forall K_i \in \mathcal{K}^+, \forall e, \eta \in \mathcal{E}, e \neq \eta:$$
$$W_i^+(e), R_i^{+,\eta}(e), \phi_i(e), f_i^+ \geq 0. \tag{5j}$$

Explanation: The two objectives (2a) and (1a) are essentially the same as in the **MCFP** formulation (Program (2)). Constraints (5a) and (5d) define the flow conservation for both working flows and recovery flows during each link failure η per-critical commodity. Similarly, Constraints (5b) and (5e) define the corresponding flow values to strictly satisfy the critical commodities' bandwidth demands. Constraint (5c) further defines the flow values of non-critical flows. For congestion-free migration, two scenarios are considered in the formulation. Constraint (5f) defines the congestion-free migration from the old working flows to the new working flows, for each critical commodity. Constraint (5g) further defines the congestion-free migration from the working flows to the recovery flows during an arbitrary link failure η. This ensures that, although different critical commodities may share capacity on the same link for backup in different failure scenarios, they cannot share during the same failure; further, enough capacity needs to be reserved for congestion-free migration during failure. Constraint (5h) further defines the capacity constraints over both critical and non-critical commodities. As non-critical flows do not affect the migration of critical flows in any scenario, only the new working flows are considered. Constraint (5i) defines flow changes using working flows only. Finally, Constraint (5j) are the non-negativity constraints.

A solution to **SCFP** is proposed in the following theorem:

Theorem 6.1: **SCFP** can be optimally solved in $O((km^2 + 2km + k)^3\mathcal{L})$ time, where m is the number of links, k is the number of commodities, and $\mathcal{L}$ is the input size. □

Proof: To solve the **SCFP** formulation (Program (5)), we still apply the linearization technique proposed in Section V. After linearization, Program (5) is a two-layer linear program with $O(km^2 + 3km + k)$ variables. Using the two-layer optimization technique mentioned in Section V, we can solve it in $O((km^2+3km+k)^3\mathcal{L})$ time based on existing solution [47]. ■

*E. Faster Heuristic for **SCFP***

Although the optimal formulation of **SCFP** can be solved in polynomial time, it has $\Omega(m)$ more variables than the formulation of **CFP** and **MCFP** in the worst case. Consequently, it has $\Omega(m^3)$ times higher worst-case time complexity than **CFP** and **MCFP**. This will largely hinder its application in most practical settings. In this subsection, we propose a faster heuristic for **SCFP** that has the same asymptotic time complexity as the solutions to **CFP** and **MCFP**.

Observe that in the **SCFP** formulation (Program (5)), the increase in asymptotic formulation size is due to the per-failure provisioning of backup network flows for each critical commodity. While this enables the finest-grained allocation of backup bandwidth, in practice each critical flow may only be affected by a relatively small portion of link failures in the network, hence per-link provisioning may be an overkill in many cases. To this end, we propose to aggregate the backup bandwidth for all possible link failures for a single round of allocation, with each link failure being resolved using a portion of the backup allocation. We propose the following heuristic formulation for **SCFP**:

Program (6): SCFP-Heuristic$(\mathcal{G}, \mathcal{K}, \mathcal{P}, K^+)$

Variables:

$W_i^+(e)$	new working flow of $K_i \in \mathcal{C}^+$ on e
$Q_i^+(e)$	reserved flow of $K_i \in \mathcal{C}^+$ on e
$\phi_i(e)$	flow change of $K_i \in \mathcal{C}^+$ on link e
Δ_i	reserved backup flow value of $K_i \in \mathcal{C}^+$
f_i^+	flow value of $K_i \in \mathcal{N}$

$$\min\ \Phi^* = \sum_{K_i \in \mathcal{C}^+} \sum_{e \in \mathcal{E}} \pi_e \phi_i(e) \quad \text{(primary)} \tag{2a}$$

$$\max \sum_{K_i \in \mathcal{N}} f_i^+ \quad \text{(secondary)} \tag{1a}$$

s.t. (5a), (5b), (5c), (5f), (5h), (5i)

$$\forall K_i \in \mathcal{C}^+, \forall v \in \mathcal{V} \setminus \{s_i, t_i\}: \sum_{e \in in(v)} Q_i^+(e) - \sum_{e \in out(v)} Q_i^+(e) = 0; \tag{6a}$$

$$\forall K_i \in \mathcal{C}^+: \sum_{e \in in(t_i)} Q_i^+(e) - \sum_{e \in out(t_i)} Q_i^+(e) = d_i + \Delta_i; \tag{6b}$$

$$\forall K_i \in \mathcal{C}^+, \forall e \in \mathcal{E}:\ Q_i^+(e) \le \Delta_i; \tag{6c}$$

$$\forall e \in \mathcal{E}: \sum_{K_i \in \mathcal{C}^+} Q_i^+(e) \le c_e; \tag{6d}$$

$$\forall K_i \in \mathcal{C}^+, \forall e \in \mathcal{E}:\ W_i^+(e) \le Q_i^+(e); \tag{6e}$$

$$\forall K_i \in \mathcal{K}^+, \forall e \in \mathcal{E}: W_i^+(e), Q_i^+(e), \phi_i(e), \Delta_i, f_i^+ \ge 0. \tag{6f}$$

Explanation: Program (6) generally has a similar form as Programs (5). The difference is that the the per-link failure recovery flows (defined by $R_i^{+,\eta}(e)$) are aggregated into a single reserved single-commodity flow (defined by $Q_i^+(e)$) for each critical commodity K_i. In addition, a new variable Δ_i is introduced for each critical commodity K_i, which represents the *additional flow value reserved for* K_i beyond the bandwidth requirement d_i, as defined in Constraint (6b). To ensure survivability, we then constrain that the *reserved flow assignment* $Q_i^+(e)$ of K_i on each link e cannot exceed the additional flow value Δ_i, as in Constraint (6c). Further, we bound the working flow $W_i^+(e)$ using each critical commodity's reserved flow $Q_i^+(e)$, as in Constraint (6e). The central idea of Programs (6) is that, during arbitrary link failure, each critical commodity K_i will only lose at most Δ_i in its reserved flow by Constraint (6c), hence the remaining reserved flow is at least d_i. The recovery flow $R_i^{+,\eta}$ against each possible link failure η can be computed for each critical commodity K_i with its reserved flow Q_i^+, using simple network flow algorithms.

We use the same method as in previous sections to solve the **SCFP-Heuristic** formulation (Program (6)):

Theorem 6.2: The **SCFP-Heuristic** formulation (Program (6)) can be solved in $O((4km + 2k)^3\mathcal{L})$ time, where m is the number of links, k is the number of commodities, and $\mathcal{L}$ is the input size. □

Proof: Proof follows from the proof of Theorem 6.1. ■

Based on Theorem 6.2, the worst-case time complexity for solving the **SCFP-Heuristic** formulation (Program (6)) is asymptotically the same as that of the solutions for **CFP** and **MCFP**, and is must faster than the worst-case complexity for solving the optimal the optimal **SCFP** formulation (Program (5)). It is notable that Program (6) is actually a relaxation of Program (5). Therefore, the heuristic is not guaranteed to find an optimal solution when one exists. Example of such instances can be constructed, but is not shown due to space limit. On the other hand, it is more practical than the proposed optimal **SCFP** solution due to its low time complexity. As will be shown in the next section, the actual performance of this heuristic is comparable to the proposed optimal **SCFP** solution, with several-orders lower running time in practice.

F. Survivability Discussions

Partial survivability: Providing full survivability against all possible link failures may result in high resource consumption and algorithm time complexity, as shown in the next section. One practical approach is to provide *partial survivability* against a designated set of key links, or links with higher failure probabilities than others. Both **SCFP** and **SCFP-Heuristic** can incorporate this capability with minor modifications. Specifically, for **SCFP**, the only change to support partial survivability is to only define the recovery flows $R_i^{+,\eta}(e)$ for those designated links η to be protected. This could drastically reduce the size of the formulation, and hence the time complexity for solving the formulation, if only a small set of links are considered. Similarly, for **SCFP-Heuristic**, Constraint (6c) can be modified to be defined on only the designated links.

Multi-failure survivability: Both **SCFP** and **SCFP-Heuristic** can also be extended to cover multiple simultaneous failure. In **SCFP**, we define recovery flows $R_i^{+,\eta_1,\ldots,\eta_\sigma}$ for any combination of σ link failures $(\eta_1, \ldots, \eta_\sigma)$. Note that this automatically covers any number of link failures less than σ. In **SCFP-Heuristic**, instead of defining Constraint (6c) for each single link failure η, we define it as $\sum_{e \in (\eta_1,\ldots,\eta_\sigma)} Q_i^+ \le \Delta_i$ for any combination of σ link failures $(\eta_1, \ldots, \eta_\sigma)$. Note that covering multiple simultaneous failures inevitably incurs time complexity exponential to the number of failures. Hence this should be combined with the aforementioned partial survivability to only protect from key combinations of link failures.

VII. Performance Evaluation

A. Experiment Settings

To evaluate the performance of our proposed model and algorithms, we carried out simulations on both well-known network topologies and randomly generated topologies. Three well-known topologies were used: the NSFNet topology with 14 nodes and 21 undirected links, the ItalyNet topology with 14 nodes and 29 undirected links, and the ARPANet topology with 20 nodes and 32 undirected links [1]. We assumed the links had duplex capability, and hence divided each undirected link into two directed links in opposite directions with independent available capacities. For random topologies, we generated

graphs based on the Waxman model [39], with 15 nodes by default, and $\alpha = 0.5$ and $\beta = 0.6$ in the Waxman model. Link capacities were uniformly drawn from $[100, 1000]$ Mbps by default; migration cost is set to 1 for every link.

Both online and offline experiments were carried out. For each online experiment, we generated 500 commodity requests (with randomly chosen source and destination nodes) which arrived in a Poisson process with arrival interval mean of 25 time units and lifetime mean of 1000 time units by default. Each request had equal probability of being a critical commodity or a non-critical commodity, where the bandwidth demand of each critical commodity was uniformly drawn from $[30, 300]$ Mbps by default. The proposed programs were used for both critical and non-critical commodities, upon both their arrival and departure; however, at the arrival of a non-critical commodity or the departure of any commodity, the programs were modified to involve only the non-critical commodities, while the rest critical commodities were untouched in order to minimize critical flow migration. We used the algorithms to process each incoming commodity, reserved bandwidth if it were accepted, and released the bandwidth when it left.

For each offline experiment, we applied a load in the network based on a specified load factor $\lambda \in [0, 1]$. Specifically, for each link l in the network, we randomly made $\lambda_l \in [0, 2\lambda] \cap [2\lambda - 1, 1]$ of its capacity unavailable (the range is bounded by $[0, 1]$ and centered at λ). In offline scenarios, all generated requests were critical commodities with demand uniformly drawn from $[30, 300]$ Mbps by default. For each request, we used the same network state information to test it under all compared algorithms; hence we did not reserve resources after the request was accepted.

Four online algorithms were evaluated against each other. **CFA** is the online algorithm that solves **CFP** (Program (1)) at the arrival or departure of each commodity, where no transient network state is considered. **MCFA** is the algorithm by solving **MCFP** (Program (2)), where network reconfiguration is considered for bandwidth guarantee. **SCFA** is the optimal algorithm that solves **SCFP** (Program (5)), while **SCFH** is the heuristic algorithm that solves **SCFP-Heuristic** (Program (6)). Four variants were also involved, namely **SCFA-NS**, **SCFH-NS**, **SCFA-P**, and **SCFH-P**. **SCFA-NS** and **SCFH-NS** correspond to the non-sharing versions of **SCFA** and **SCFH**, respectively. In the non-sharing versions, non-critical commodities cannot share the backup bandwidth of critical commodities even when no failure happens. We used them to evaluate the work conserving property of our proposed algorithms. **SCFA-P** and **SCFH-P** correspond to the versions of **SCFA** and **SCFH** that consider *partial survivability*, respectively, as shown in Section VI-F.

We evaluated the performance of our algorithms using the following metrics. The **critical acceptance ratio** measures the number of accepted critical commodities over all critical commodity requests (note that non-critical commodities do not need to be accepted and can always stay in the system in order to utilize spare bandwidth left by the critical commodities). It reflects an algorithm's capability to accept as many critical commodities given their various bandwidth requirement. The **non-critical sum throughput** measures the average total throughput of non-critical flows throughout time, and is used to validate the effectiveness of our work conserving design. The **running time** measures the average computation time of each algorithm per commodity request.

To comprehensively evaluate algorithm performance, we varied different parameters in each experiment, including the network size (both number of nodes and links in the random graph case), ratio of critical commodities over all commodities, average bandwidth demand of critical commodities, and the key link ratio (the number of key links over all links in the network, where only key links are protected by the partial-survivability algorithms). Each experiment was repeated for 20 times under the same experiment setting, and the result was averaged over all runs. All experiments were conducted on a Ubuntu Linux PC with Quad-Core 3.4GHz CPU and 16GB memory.

B. Evaluation Results

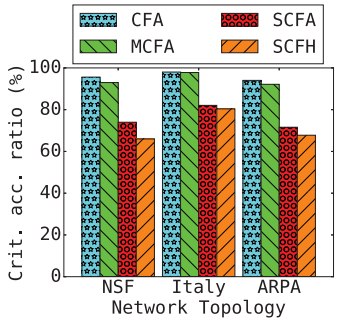
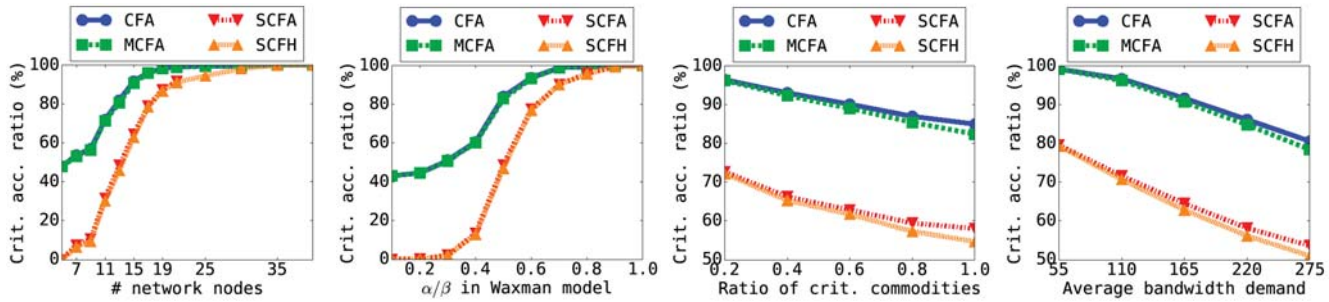


Fig. 2: Acceptance ratio on different topologies.

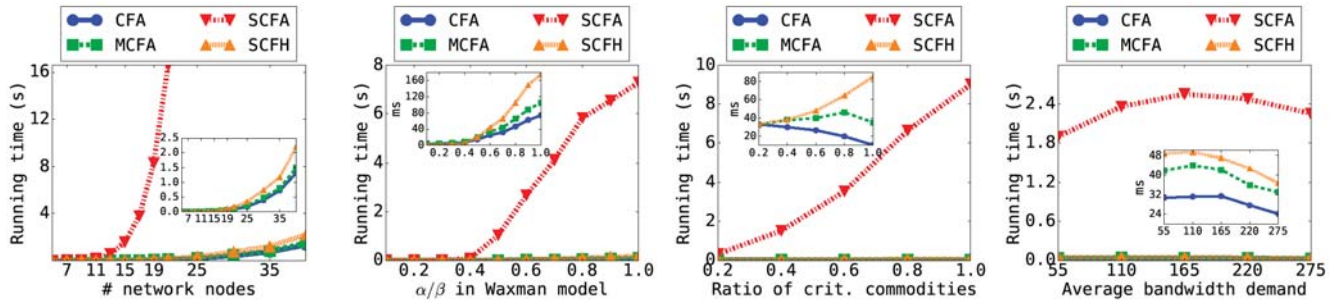
Figs. 2–4 show the results of our online experiments. Fig. 2 shows the acceptance ratio of the four algorithms with the well-known topologies. We can observe that while enforcing migratability requirement (**MCFA**) only imposes minor impact on the acceptance ratio, enforcing survivability requirement (**SCFA** and **SCFH**) has a much larger impact, with respect to the basic case where no criticality is enforced during transient network states (**CFA**). Also observe that the heuristic solution **SCFH** indeed has some performance degradation compared to the per-critical commodity optimal solution **SCFA**, especially on networks with larger sizes (ItalyNet and ARPANet), but the difference is minor.

Fig. 3 shows the acceptance ratio with random topologies, where we varied the topology size (number of nodes), connectivity (number of links, determined by the α and β parameters in the Waxman model), ratio of critical commodities, and the per-critical commodity average bandwidth demand, in Figs. 3(a)–3(d) respectively. The trends of the curves match the intuition, where acceptance ratio would increase with increasing number of nodes or connectivity, or decreasing number of critical commodities or per-critical commodity bandwidth demand. On the other hand, we can observe the similar comparison among the four algorithms, where **MCFA** has only slightly lower acceptance ratio than **CFA**, but **SCFA** and **SCFH** have much lower acceptance ratios, due to the more stringent requirement on the network resources for backup



(a) (b) (c) (d)

Fig. 3: Acceptance ratio vs. (a) # nodes, (b) connectivity, (c) critical commodity ratio, and (d) average bandwidth demand.



(a) (b) (c) (d)

Fig. 4: Average running time vs. (a) # nodes, (b) connectivity, (c) critical commodity ratio, and (d) average bandwidth demand.

reservation. This basically suggests that enforcing survivability requirement is a much harder task, and thus should reflect higher prices to the requested users in the perspective of the network operator. Also observe that the acceptance loss of our proposed heuristic **SCFH** is still minor compared to the per-commodity optimal solution **SCFA**.

Fig. 4 shows the corresponding per-commodity running time of the algorithms with random topologies. Basically the running time of every algorithm increases with network sizes (nodes and links) and the number of critical commodities. We notice that enforcing migratability requirement (**MCFA**) does not incur much computation overhead compared to the basic case (**CFA**). However, enforcing survivability requirement using the (per-commodity) optimal solution (**SCFA**) incurs a much higher computation overhead, as its running time can be several orders higher than the former two. On the other hand, our proposed heuristic solution (**SCFH**) incurs a much lower computation overhead (several orders lower) than **SCFA**, only slightly higher than the former two. For example, the **SCFA** algorithm run on a network with 21 nodes is about 8 times slower than the **SCFH** algorithm run on a network with 40 nodes on average. This difference in running time will even increase with larger network sizes due to the larger sizes of **SCFA** LPs. Combined with its near-optimal performance in terms of acceptance ratio, **SCFH** is much more suitable for practical use than **SCFA**, the per-commodity optimal solution.

Fig. 5 shows experiments validating the effectiveness of our work conserving design. We compare both **SCFA** and **SCFH** to their non-sharing versions, namely **SCFA-NS** and **SCFH-NS** respectively. In the non-sharing versions, both algorithms

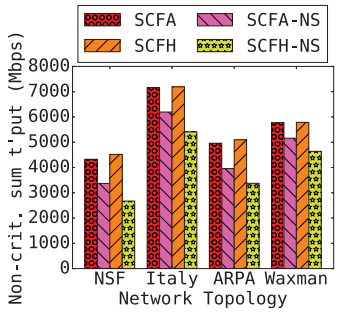
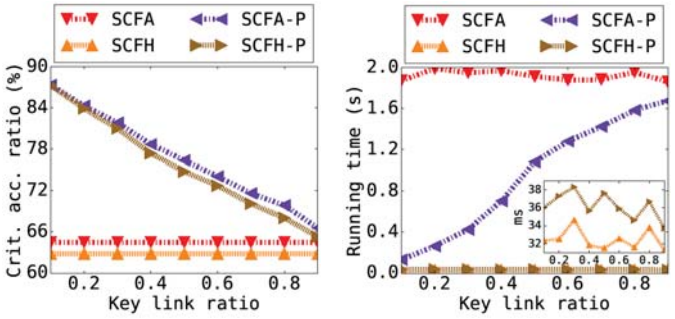


Fig. 5: Non-critical throughput with sharing vs. non-sharing.

do not allow non-critical commodities to share bandwidth allocated to the recovery flows of critical commodities even when no failure happens; the non-sharing versions thus accept the same set of critical commodities as their sharing opponents respectively, yet result in less bandwidth to be used by non-critical commodities. We can see that sharing greatly promotes non-critical throughput compared to non-sharing regarding both algorithms. Also note that (non-)sharing has more impact on the heuristic solution than on the optimal algorithm, because the heuristic solution generally results in more backup resource reserved for critical commodities. This confirms the effectiveness of our work conserving design. Note that comparison of non-critical throughput between **SCFA** (resp. **SCFA-NS**) and **SCFH** (resp. **SCFH-NS**) is meaningless, as they accepts different sets of critical commodities in general.

Fig. 6 further shows experiment results that evaluate partial survivability versus full survivability. In the experiments we



(a) Acceptance ratio (b) Running time

Fig. 6: Acceptance ratio and running time vs. key link ratio.

varied the *key link ratio*, *i.e.*, the portion of links over all links which are protected by our partial-survivability algorithms **SCFA-P** and **SCFH-P**. We randomly selected a subset of links to protect based on the key link ratio. From Fig. 6(a), we can see that by protecting only a subset of the overall links, the acceptance ratio can be decently improved compared to protecting against every possible link failure. Such improvements degrade when the number of links we need to protect increases. Furthermore, from Fig. 6(b), the time complexity for the optimal partial-survivability algorithm (**SCFA-P**) is much lower than the optimal full-survivability algorithm **SCFA**. When the number of key links increases, time complexity of **SCFA-P** also increases. This is due to the fact that the time complexity for solving LPs increases with the number of variables (program size), which grows linearly with the number of links to protect in **SCFA** and **SCFA-P**. On the other hand, for the heuristic algorithms **SCFH** and **SCFH-P**, we observe the opposite phenomenon: partial survivability even increases the time complexity. This is because **SCFH-P** accepts more critical commodities than **SCFH** as shown in Fig. 6(b). In our implementation, each accepted critical commodity involves solving *three* LPs in total, *two* for accommodation when it arrives and *one* for adjustment when it leaves; on the contrary, if a critical commodity is rejected, only *one* LP is solved: it will be rejected immediately after solving the first LP due to infeasibility of the corresponding program. Since given the same network state, both **SCFH** and **SCFH-P** take approximately the same amount of time to solve a single LP (as they have the same program sizes), the algorithm with higher acceptance ratio inevitably incurs more average time for solving each instance. This is in contrast to the comparison between **SCFA** and **SCFA-P**, which have different program sizes. Due to the high complexity of solving LPs regarding the program sizes [47], the excessive time for solving each LP with a larger size in **SCFA** dominates the time incurred for solving more LPs each with a smaller size in **SCFA-P**, which leads to the former being much slower than the latter in all experiments.

The above online experiments cannot verify the optimality of **SCFA** compared to **SCFH**, although the former outperforms the latter in all results shown. Hence we further conducted offline experiments to specifically verify the optimality of **SCFA** for solving the per-commodity **SCFP** problem, and the results are shown in Fig. 7.

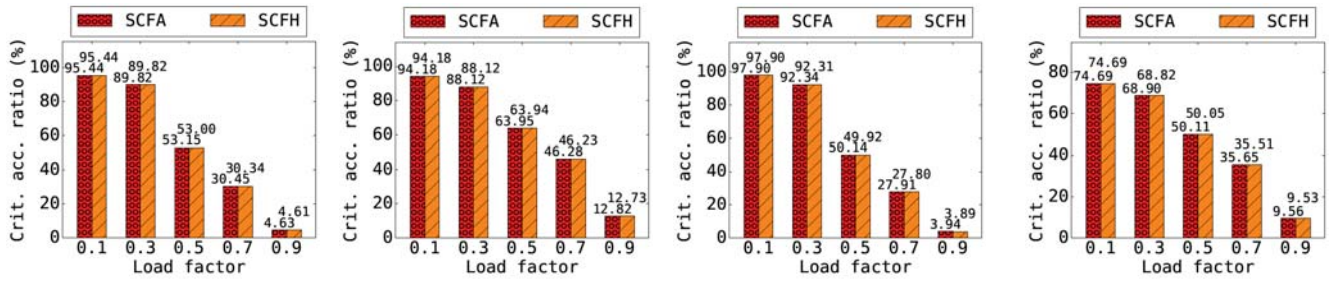
Fig. 7 shows the acceptance ratio on both well-known and random topologies, where we varied the load factor λ from 0.2 to 1.0. Since the results are akin for **SCFA** and **SCFH** in almost all cases, we specifically list all the acceptance ratios on the corresponding bars. Two things can be observed. First, **SCFH** never performs better than **SCFA** in any case, which aligns with the optimality of the latter. Second, the sub-optimality of **SCFH** is minimal; almost no performance degradation can be observed in the offline experiments. This further illustrates the effectiveness of our proposed heuristic solution. As for the running time, again **SCFA** is several orders slower than **SCFH**; since the comparison is similar to that shown in Fig. 4, the results are not displayed in this paper.

Our findings are summarized as follows:

- Enforcing bandwidth criticality during transient network states (network reconfiguration and failures) indeed incurs performance overhead: in terms of both less accepted critical commodities, and increased computation time.
- While enforcing migratability requirement incurs only minimal overhead, enforcing survivability requirement can be much more resource- and time-intensive, hence should be used with caution by the network operators.
- When enforcing survivability requirement is necessary, it is more suitable to employ our heuristic solution than the per-critical commodity optimal solution, due to both the comparable performance (in terms of acceptance ratio) and the much better time-efficiency of the heuristic.
- Another way for improving scalability is to employ partial survivability, where only a small subset of key links are to be protected instead of all the links in the network. This can both reduce resource overhead (thus increase acceptance ratio), and greatly reduce time complexity for the optimal algorithm.

VIII. Conclusions

In this paper, we studied how to provide bandwidth guarantee to network commodities during transient network states. We first proposed the concept of critical flow, for which the commodity explicitly specifies its bandwidth requirement, and the network states during which the bandwidth demand must be satisfied, namely its criticality requirement. We further studied the problem of accommodating a newly arrived critical commodity, given the current set of active flows in the network. Starting from the basic scenario where no transient network state is considered, we progressively built problem formulations and solutions for accommodating critical commodities with migratability requirement, and with survivability requirement. We further proposed a heuristic solution for accommodating critical commodities with survivability requirement, in order to reduce time complexity over the optimal solution. We conducted extensive simulation experiments on well-known topologies and random topologies with randomly generated requests. The results showed that while providing bandwidth guarantee for transient network states indeed improves performance received by each accepted commodity, it inevitably consumes more bandwidth and increases computation time, and hence tampers the network's ability in



(a) NSFNet (b) ItalyNet (c) ARPANet (d) Random topology

Fig. 7: Offline acceptance ratio on (a) NSFNet, (b) ItalyNet, (c) ARPANet, and (d) random topologies.

serving more commodities. We also observed that survivability is much more expensive than migratability, in terms of both resource usage and computation time. On the other hand, our proposed heuristic can achieve comparable performance to, but is several-orders more time-efficient than, the per-commodity optimal survivable solution. As for future work along this line, other transient network states can be considered to further improve network performance, including multiple simultaneous failures, traffic dynamics, topology dynamics, bursty traffic, etc.

REFERENCES

[1] "Reference Network Topologies." URL: http://www.av.it.pt/anp/on/refnet2.html

[2] S. Acharya, B. Gupta, P. Risbood, and A. Srivastava, "PESO: Low Overhead Protection for Ethernet over SONET Transport," in *Proc. IEEE INFOCOM*, 2004, pp. 165–175.

[3] R. Banner and A. Orda, "The Power of Tuning: A Novel Approach for the Efficient Design of Survivable Networks," *IEEE/ACM Trans. Netw.*, vol. 15, no. 4, pp. 737–749, 2007.

[4] S. Boyd, L. Xiao, and A. Mutapcic, "Subgradient Methods," 2003. URL: https://web.stanford.edu/class/ee392o/subgrad_method.pdf

[5] S. Brandt, K.-T. Forster, and R. Wattenhofer, "On Consistent Migration of Flows in SDNs," in *Proc. IEEE INFOCOM*, 2016.

[6] S. Chen and K. Nahrstedt, "An Overview of Quality of Service Routing for Next-Generation High-Speed Networks: Problems and Solutions," *IEEE Netw.*, vol. 12, no. 6, pp. 64–79, 1998.

[7] B. Fortz and M. Thorup, "Internet Traffic Engineering by Optimizing OSPF Weights," in *Proc. IEEE INFOCOM*, 2000, pp. 519–528.

[8] L. Fu, X. Wang, and P. R. Kumar, "Are We Connected? Optimal Determination of Source-Destination Connectivity in Random Networks," *IEEE/ACM Trans. Netw.*, vol. PP, no. 99, pp. 751–764, 2016.

[9] X. Fu, Z. Xu, Q. Peng, L. Fu, and X. Wang, "Complexity Vs. Optimality: Unraveling Source-Destination Connection in Uncertain Graphs," in *Proc. IEEE INFOCOM*, 2017.

[10] R. Hassin, "Approximation Schemes for the Restricted Shortest Path Problem," *Math. Oper. Res.*, vol. 17, no. 1, pp. 36–42, feb 1992.

[11] J. He, Jiayueand Rexford, "Toward Internet-Wide Multipath Routing," *IEEE Netw.*, vol. 22, no. 2, pp. 16–21, mar 2008.

[12] P.-H. Ho, J. Tapolcai, and T. Cinkler, "Segment Shared Protection in Mesh Communications Networks With Bandwidth Guaranteed Tunnels," *IEEE/ACM Trans. Netw.*, vol. 12, no. 6, pp. 1105–1118, dec 2004.

[13] P.-H. Ho, J. Tapolcai, and H. Mouftah, "On Achieving Optimal Survivable Routing for Shared Protection in Survivable Next-Generation Internet," *IEEE Trans. Reliab.*, vol. 53, no. 2, pp. 216–225, jun 2004.

[14] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving High Utilization with Software-Driven WAN," in *Proc. ACM SIGCOMM*, 2013, pp. 15–26.

[15] X. Jin, H. H. Liu, R. Gandhi, S. Kandula, R. Mahajan, M. Zhang, J. Rexford, and R. Wattenhofer, "Dynamic Scheduling of Network Updates," in *Proc. ACM SIGCOMM*, 2014, pp. 539–550.

[16] M. Johansson and A. Gunnar, "Data-driven Traffic Engineering: Techniques, Experiences and Challenges," in *Proc. IEEE ICBCNS*, 2006.

[17] S. Kandula, D. Katabi, B. Davie, and A. Charny, "Walking the Tightrope: Responsive Yet Stable Traffic Engineering," in *Proc. ACM SIGCOMM*, 2005, pp. 253–264.

[18] K. Kar, M. Kodialam, and T. Lakshman, "Routing Restorable Bandwidth Guaranteed Connections Using Maximum 2-Route Flows," *IEEE/ACM Trans. Netw.*, vol. 11, no. 5, pp. 772–781, oct 2003.

[19] M. Kodialam and T. Lakshman, "Dynamic Routing of Restorable Bandwidth-Guaranteed Tunnels Using Aggregated Network Resource Usage Information," *IEEE/ACM Trans. Netw.*, vol. 11, no. 3, pp. 399–410, jun 2003.

[20] H. H. Liu, S. Kandula, R. Mahajan, M. Zhang, and D. Gelernter, "Traffic Engineering with Forward Fault Correction," in *Proc. ACM SIGCOMM*, 2014, pp. 527–538.

[21] H. H. Liu, X. Wu, M. Zhang, L. Yuan, R. Wattenhofer, and D. Maltz, "zUpdate: Updating Data Center Networks with Zero Loss," in *Proc. ACM SIGCOMM*, 2013, pp. 411–422.

[22] X. Liu, S. Mohanraj, M. Pioro, and D. Medhi, "Multipath Routing From a Traffic Engineering Perspective: How Beneficial Is It?" in *Proc. IEEE ICNP*, 2014, pp. 143–154.

[23] 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.

[24] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, mar 2008.

[25] M. Medard, S. Finn, R. Barry, and R. Gallager, "Redundant Trees for Preplanned Recovery in Arbitrary Vertex-Redundant or Edge-Redundant Graphs," *IEEE/ACM Trans. Netw.*, vol. 7, no. 5, pp. 641–652, 1999.

[26] T. Mizrahi, O. Rottenstreich, and Y. Moses, "TimeFlip: Scheduling Network Updates with Timestamp-Based TCAM Ranges," in *Proc. IEEE INFOCOM*, 2015, pp. 2551–2559.

[27] T. Mizrahi, E. Saat, and Y. Moses, "Timed Consistent Network Updates," in *Proc. ACM SOSR*, 2015.

[28] L. Muscariello, D. Perino, and D. Rossi, "Do Next Generation Networks Need Path Diversity?" in *Proc. IEEE ICC*, 2009.

[29] S. Ramamurthy and B. Mukherjee, "Survivable WDM Mesh Networks, Part I-Protection," in *Proc. IEEE INFOCOM*, 1999, pp. 744–751 vol.2.

[30] M. Reitblatt, N. Foster, J. Rexford, C. Schlesinger, and D. Walker, "Abstractions for Network Update," in *Proc. ACM SIGCOMM*, 2012, pp. 323–334.

[31] S. Subramaniam and D. Zhou, "Survivability in Optical Networks," *IEEE Netw.*, vol. 14, no. 6, pp. 16–23, 2000.

[32] M. Suchara, D. Xu, R. Doverspike, D. Johnson, and J. Rexford, "Net-

work Architecture for Joint Failure Recovery and Traffic Engineering," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 39, no. 1, pp. 97–108, jun 2011.

[33] D. Turner, K. Levchenko, A. C. Snoeren, and S. Savage, "California Fault Lines: Understanding the Causes and Impact of Network Failures," in *Proc. ACM SIGCOMM*, 2010, p. 315.

[34] J.-P. Vasseur, M. Pickavet, and P. Demeester, *Network Recovery: Protection and Restoration of Optical, SONET-SDH, IP, and MPLS*, 2004.

[35] C. Villamizar, "Use of Multipath with MPLS and MPLS Transport Profile (MPLS-TP)," 2012. URL: https://tools.ietf.org/html/rfc7190

[36] S. Vissicchio and L. Cittadini, "FLIP the (Flow) Table: Fast Lightweight Policy-Preserving SDN Updates," in *Proc. IEEE INFOCOM*, 2016.

[37] N. Wang, K. Ho, G. Pavlou, and M. Howarth, "An Overview of Routing Optimization for Internet Traffic Engineering," *IEEE Commun. Surv. Tutorials*, vol. 10, no. 1, pp. 36–56, 2008.

[38] Z. Wang and J. Crowcroft, "Quality-of-service Routing for Supporting Multimedia Applications," *IEEE J. Sel. Areas Commun.*, vol. 14, no. 7, pp. 1228–1234, 1996.

[39] B. M. Waxman, "Routing of Multipoint Connections," *IEEE J. Sel. Areas Commun.*, vol. 6, no. 9, pp. 1617–1622, 1988.

[40] X. Xiao, A. Hannan, B. Bailey, and L. Ni, "Traffic Engineering with MPLS in the Internet," *IEEE Netw.*, vol. 14, no. 2, pp. 28–33, 2000.

[41] D. Xu, Y. Xiong, C. Qiao, and G. Li, "Failure Protection in Layered Networks with Shared Risk Link Groups," *IEEE Netw.*, vol. 18, no. 3, pp. 36–41, may 2004.

[42] G. Xue, L. Chen, and K. Thulasiraman, "Quality-of-service and Quality-Of-Protection Issues in Preplanned Recovery Schemes Using Redundant Trees," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 8, pp. 1332–1345, oct 2003.

[43] G. Xue, R. Gottapu, X. Fang, D. Yang, and K. Thulasiraman, "A Polynomial-Time Algorithm for Computing Disjoint Lightpath Pairs in Minimum Isolated-Failure-Immune WDM Optical Networks," *IEEE/ACM Trans. Netw.*, vol. 22, no. 2, pp. 470–483, apr 2014.

[44] 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.

[45] J. Yallouz and A. Orda, "Tunable QoS-Aware Network Survivability," *IEEE/ACM Trans. Netw.*, vol. 25, no. 1, pp. 139–149, feb 2017.

[46] J. Yallouz, O. Rottenstreich, and A. Orda, "Tunable Survivable Spanning Trees," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1853–1866, jun 2016.

[47] Y. Ye, "An $O(n^3L)$ Potential Reduction Algorithm for Linear Programming," *Math. Program.*, vol. 50, no. 1, pp. 239–258, mar 1991.

[48] W. Zhang, J. Tang, C. Wang, and S. de Soysa, "Reliable Adaptive Multipath Provisioning with Bandwidth and Differential Delay Constraints," in *Proc. IEEE INFOCOM*, 2010.

Ruozhou Yu (Student Member 2013) received his B.S. degree from Beijing University of Posts and Telecommunications, Beijing, China, in 2013. Currently he is a Ph.D student in the School of Computing, Informatics, and Decision Systems Engineering at Arizona State University. His research interests include network virtualization, software-defined networking, cloud and data center networks, edge computing and internet-of-things, etc.

Guoliang Xue (Member 1996, Senior Member 1999, Fellow, 2011) is a professor of Computer Science and Engineering at Arizona State University. He received the PhD degree in Computer Science from the University of Minnesota in 1991, the MS degree in Operations Research from Qufu Normal University in 1984, and the BS degree in Mathematics from Qufu Normal University in 1981. His research interests span the areas of Quality of Service provisioning, network security and privacy, crowdsourcing and network economics, RFID systems and Internet of Things, smart city and smart grids. He has published over 280 papers in these areas, many of which in top conferences such as ICNP, INFOCOM, MOBICOM, MOBIHOC, NDSS and top journals such as IEEE/ACM Transactions on Networking, IEEE JSAC, IEEE TMC. He was a keynote speaker at IEEE LCN'2011 and ICNC'2014. He was a TPC Co-Chair of IEEE INFOCOM'2010 and a General Co-Chair of IEEE CNS'2014. He has served on the TPC of many conferences, including ACM CCS, ACM MOBIHOC, IEEE ICNP, and IEEE INFOCOM. He served on the editorial board of IEEE/ACM Transactions on Networking and the Computer Networks Journal. He serves as the Area Editor of IEEE Transactions on Wireless Communications, overseeing 13 editors in the Wireless Networking area. He is an IEEE Fellow, and the VP-Conferences of the IEEE Communications Society.

Xiang Zhang (Student Member 2013) received his B.S. degree from University of Science and Technology of China, Hefei, China, in 2012. Currently he is a Ph.D student in the School of Computing, Informatics, and Decision Systems Engineering at Arizona State University. His research interests include network economics and game theory in crowdsourcing and cognitive radio networks.