Ultra-Reliable Distributed Cloud Network Control with End-to-End Latency Constraints

Yang Cai, *Student Member, IEEE*, Jaime Llorca, *Member, IEEE*, Antonia M. Tulino, *Fellow, IEEE*, and Andreas F. Molisch, *Fellow, IEEE*

***Abstract*—We are entering a rapidly unfolding future driven by the delivery of real-time computation services, such as industrial automation and augmented reality, collectively referred to as augmented information (AgI) services, over highly distributed cloud/edge computing networks. The interaction intensive nature of AgI services is accelerating the need for networking solutions that provide strict latency guarantees. In contrast to most existing studies that can only characterize average delay performance, we focus on the critical goal of delivering AgI services ahead of corresponding deadlines on a per-packet basis, while minimizing overall cloud network operational cost. To this end, we design a novel queuing system able to track data packets' lifetime and formalize the *delay-constrained least-cost dynamic network control problem*. To address this challenging problem, we first study the setting with average capacity (or resource budget) constraints, for which we characterize the delay-constrained stability region and design a throughput-optimal control policy leveraging Lyapunov optimization theory on an equivalent virtual network. Guided by the same principle, we tackle the peak capacity constrained scenario by developing the *reliable cloud network control* (RCNC) algorithm, which employs a two-way optimization method to make actual and virtual network flow solutions converge in an iterative manner. Extensive numerical results show the superior performance of the proposed control policy compared with the state-of-the-art cloud network control algorithm, and the value of guaranteeing strict end-to-end deadlines for the delivery of next-generation AgI services.**

***Index Terms*—Distributed cloud network control, edge computing, delay-constrained stability region, strict latency, reliability**

I. Introduction

THE so-called automation era or fourth industrial revolution will be driven by the proliferation of compute- and interaction-intensive applications, such as real-time computer vision, autonomous transportation, machine control in Industry 4.0, telepresence, and augmented/virtual reality (AR/VR), which we collectively refer to as *augmented information (AgI) services* [2]–[4]. In addition to the communication resources needed for the delivery of data streams to corresponding destinations, AgI services also require a significant amount of computation resources for the real-time processing, via possibly multiple functions, of source and intermediate data streams.

The evolution of user equipments (UEs) towards increasingly small, lightweight, seamless devices, and their associated limitations in power and computing capabilities, has been pushing the need to offload many computation-intensive tasks to the cloud. However, increased access delays associated with distant centralized cloud data centers are fueling advanced network architectures such as fog and mobile edge computing (MEC) that push computation resources closer to the end users in order to strike a better balance between resource efficiency and end-to-end delay [4]–[6]. In this work, we refer to the overall wide-area distributed computation network that results from the convergence of telco networks and cloud/edge/UE resources as a *distributed cloud network*.

Delay and cost are thus two essential metrics when evaluating the performance of AgI service delivery over a distributed cloud network. From the consumers' perspective, excessive end-to-end delays can significantly impact quality of experience (QoE), especially for delay-sensitive AgI applications (e.g., industrial automation, augmented reality) where packets must be delivered by a strict deadline in order to be effective (i.e., packets delivered after their deadline become irrelevant and/or break application interactivity). In this context, *timely throughput*, which measures the rate of effective packet delivery (i.e., within-deadline packet delivery rate), becomes the appropriate performance metric [7]–[9]. On the other hand, network operators care about the overall resource (e.g., computation, communication) consumption needed to support the dynamic service requests raised by end users.

Both delay and cost will ultimately be dictated by the choice of cloud/edge locations where to execute the various AgI service functions, the network paths over which to route the service data streams, and the corresponding allocation of computation and communication resources. Therefore, to maximize the benefit of distributed cloud networks for the delivery of AgI services, two fundamental problems need to be jointly addressed:

- *where to execute the requested AgI service functions, and how much computation resource to allocate*
- *how to route and schedule data streams through the appropriate sequence of service functions, and how much communication resource to allocate*

In addition, due to the dynamic and unpredictable nature of AgI service requests, the above placement, processing, routing, and resource allocation problems must be addressed in an

This work was supported by the National Science Foundation (NSF) under CNS-1816699. Part of this paper was presented at IEEE ICC 2021 [1].

Y. Cai and A. F. Molisch are with the Department of Electrical and Computer Engineering, University of Southern California, Los Angeles, CA 90089, USA (e-mail: yangcai@usc.edu; molisch@usc.edu).

J. Llorca is with the Electrical and Computer Engineering Department, New York University, Brooklyn, NY 11201 USA (e-mail: jllorca@nyu.edu).

A. M. Tulino is with the Electrical and Computer Engineering Department, New York University, Brooklyn, NY 11201 USA, and also with the Department of Electrical Engineering, Universityà degli Studi di Napoli Federico II, Naples 80138, Italy (e-mail: atulino@nyu.edu; antoniamaria.tulino@unina.it).

online manner, in response to stochastic network conditions and service demands.

A. Related Work

With the advent of software defined networking (SDN) and network function virtualization (NFV), network (and, by extension, AgI) services can be deployed as a sequence of software functions or service function chains (SFCs) instantiated over distributed cloud locations. A number of studies have investigated the problem of joint SFC placement and routing over multi-hop networks with the objective of either minimizing overall operational cost [10]–[14], or maximizing accepted service requests [15]–[18]. Nonetheless, these solutions exhibit two main drawbacks. First, the problem is formulated as a *static* optimization problem without considering the dynamic nature of service requests. In addition, when it comes to delay performance, these studies mainly focus on propagation delay [12], [13], [15], [16], while neglecting queuing delay, or using simplified models (e.g., M/M/1) to approximate it [17]. Second, due to the combinatorial nature of the problem, the corresponding formulations typically take the form of (NP-hard) mixed integer linear programs and either heuristic or loose approximation algorithms are developed, compromising the quality of the resulting solution.

More recently, a number of studies have addressed the SFC optimization problem in *dynamic* scenarios, where one needs to make joint packet processing and routing decisions in an online manner [19]–[21]. The works in [19], [20] employ a generalized cloud network flow model that allows joint control of processing and transmission flows. The work in [21] shows that the traffic control problem in distributed cloud networks (involving joint packet processing and routing decisions) can be reduced to a packet routing problem on a properly constructed *layered graph* that includes extra edges to characterize the processing operations (i.e., packets pushed through these edges are interpreted as being processed by a service function). By this transformation, many control policies designed for packet routing can be extended to address cloud network control problems (i.e., packet processing and routing), especially those aiming at maximizing network throughput with bounded *average* delay performance.

In particular, back-pressure (BP) [22] is a well-known algorithm for throughput-optimal routing that leverages Lyapunov drift control theory to steer data packets based on the *pressure* difference (differential backlog) between neighbor nodes. In addition, the Lyapunov drift-plus-penalty (LDP) control approach [23] extends the BP algorithm to also minimize network operational cost (e.g., energy expenditure), while preserving throughput optimality. Despite the remarkable advantages of achieving optimal throughput performance via simple local policies without requiring any knowledge of network topology and traffic demands, both BP and LDP approaches can suffer from poor average delay performance, especially in low congestion scenarios, where packets can take unnecessarily long, and sometimes even cyclic, paths [24]. Average delay reductions were then shown to be obtained in [25] by combining BP and hop-distance based shortest-path routing, using a more complex Markov decision process (MDP) formulation in [26], or via the use of source routing to dynamically select acyclic routes for incoming packets, albeit requiring global network information, in [27].

Going beyond average delay and analyzing per-packet delay performance is a more challenging problem with much fewer known results, even in the context of packet routing and under *static arrivals*. In particular, the restricted shortest path (RSP) problem, which aims to find the min-cost path for a given source-destination pair subject to an end-to-end delay (or path length) constraint, is known to be NP-hard [28]. Considering *dynamic arrivals* becomes a further obstacle that requires additional attention. An opportunistic scheduling policy is proposed in [29] that trades off worst-case delay and timely throughput, which preserves the delay guarantee when applied to hop-count-limited transmissions. However, it requires a link selection procedure (to meet the hop-count requirement) that weakens its performance in general networks (e.g., mesh topologies); besides, the timely throughput is with respect to the worst-case delay, rather than the deadline imposed by the application, leading to either sub-optimal throughput under stringent deadline constraints, or looser guarantees on the worst-case delay; finally, it treats packet scheduling on different links separately, lacking an end-to-end optimization of the overall delay. In [30], the authors formulate the problem of timely throughput maximization as an exponential-size constrained MDP (CMDP), and derive an approximate solution based on solving the *optimal single-packet transportation problem* for each packet; in addition, [31] addresses the more complex set-up of wireless networks with link interference. While this approach reduces the complexity from exponential (of a general solution that makes joint packet decisions) to polynomial, it requires solving a dynamic programming problem for each packet at every time slot, which can still become computationally expensive. Furthermore, none of these works takes operational cost minimization into account, an important aspect in modern elastic cloud environments.

B. Contributions

In this paper, we investigate the problem of multi-hop cloud network control with the goal of delivering AgI services with strict per-packet deadline constraints, while minimizing overall operational cost. More concretely, we focus on **reliable service delivery**, which requires the timely throughput of each service, i.e., the rate of packets delivered by their *deadlines*, to surpass a given level in order to meet a desired QoE. We study the problem in dynamic scenarios, i.e., assuming the service requests are unknown and time-varying.

There are two main challenges that prohibit the use of existing cloud network control methods (e.g., [19]) and associated queuing systems for reliable service delivery. In particular, existing queuing systems: (i) do not take packet deadlines into account and cannot track associated *packet lifetimes*; (ii) do not allow *packet drops*, which becomes critical in delay-constrained routing, since dropping *outdated* packets can benefit cost performance without impacting *timely* throughput.

To overcome these drawbacks, we construct a novel queuing system with separate queues for different deadline-driven

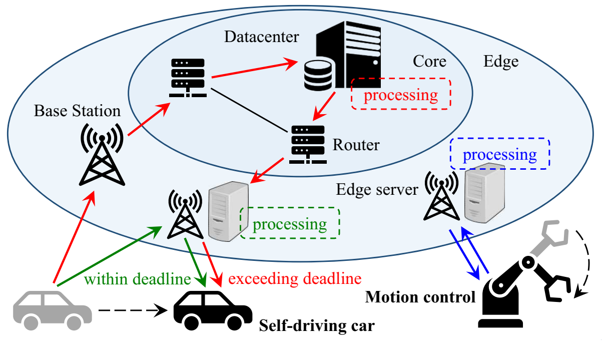


Fig. 1. Illustration of the delivery of two delay-sensitive applications over a distributed cloud network. The red route (for the *self-driving car* application) illustrates a configuration that can exceed end-to-end latency constraints.

packet lifetimes, and allow packet drops upon lifetime expiry. In contrast to standard queuing systems, where packets are transmitted to reduce network congestion and keep physical queues stable [23], the new queuing model is fundamentally different: stability of physical queues becomes irrelevant (due to packet drops), and packet transmission is driven by the requirement to deliver packets on time (**reliable service delivery**). The proposed solution is presented in two stages. First, we study a relaxed *average-constrained network control* problem and derive an exact solution via a flow matching technique where flow scheduling decisions are driven by an LDP solution to an equivalent virtual network control problem. Then, we address the original *peak-constrained* problem, with the additional challenge of non-equivalent actual and virtual network formulations, and devise an algorithm that adapts the LDP plus flow matching technique via an iterative procedure.

Our contributions can be summarized as follows:

1) We develop a novel queuing model that allows tracking data packet lifetimes and dropping outdated packets, and formalize the delay-constrained least-cost dynamic network control problem $\mathscr{P}_0$.
2) We derive a relaxed problem $\mathscr{P}_1$ targeting the same objective in an average-capacity-constrained network, and characterize its delay-constrained stability region based on a lifetime-driven flow conservation law.
3) We design a fully distributed near-optimal (see Proposition 3 for throughput and cost guarantees) control policy for $\mathscr{P}_1$ by (i) deriving an equivalent virtual network control problem $\mathscr{P}_2$ that admits an efficient LDP-based solution, (ii) proving that $\mathscr{P}_1$ and $\mathscr{P}_2$ have identical stability region, flow space, and optimal objective value, and (iii) designing a randomized policy for $\mathscr{P}_1$ guided by matching the virtual flow solution to $\mathscr{P}_2$.
4) We leverage the flow matching technique to develop an algorithm for $\mathscr{P}_0$, referred to as *reliable cloud network control* (RCNC), whose solution results from the convergence of actual (for $\mathscr{P}_0$) and virtual (for $\mathscr{P}_2$) flows via an iterative optimization procedure.

The rest of the paper is organized as follows. In Section II, we introduce network model and associated queuing system. In Section III, we define the policy space and formulate the original problem $\mathscr{P}_0$. In Section IV, we study the relaxed problem $\mathscr{P}_1$ and derive an equivalent LDP-amenable formulation $\mathscr{P}_2$. Section V presents the algorithm for solving $\mathscr{P}_1$ as well as its performance analysis, which is extended to develop an iterative algorithm for $\mathscr{P}_0$ in Section VI. Numerical results are shown in Section VII, and possible extensions are discussed in Section VIII. Finally, we summarize the main conclusions in Section IX.

TABLE I
TABLE OF NOTATIONS

Symbol	Description
$\mathcal{G}$; $\mathcal{V}$, $\mathcal{E}$; d	Network graph model; node, edge sets; destination.
δ_i^-, δ_i^+	Sets of incoming/outgoing neighbors of i.
C_{ij}, e_{ij}	Transmission capacity and cost of link (i, j).
l, L, $\mathcal{L}$	Lifetime, maximum lifetime, set of lifetimes.
$a(t)$, λ	Number of arrival packets, arrival rate.
$x(t)$; $\nu(t)$, $\mu(t)$	Flow variable; virtual, actual flows.
γ, Λ	Reliability level, network stability region.
$\mathcal{F}$, Γ	Feasible policy space, flow space.
$Q(t)$, $U(t)$, $R(t)$	Actual queue, virtual queue, request queue.

II. System Model

A. Cloud Layered Graph

The ultimate goal of this work is to design control policies for distributed cloud networks to reliably support multiple delay-sensitive AgI services, where the network is equipped with computation resources (cloud servers, edge/fog computing nodes, etc.) able to host service functions and execute corresponding computation tasks.

While in traditional packet routing problems, each node treats its *neighbor nodes* as outgoing interfaces over which packets can be scheduled for transmission, a key step to address the AgI service control problem is to treat the co-located *computing resources* as an additional outgoing interface over which packets can be scheduled for processing [19]. Indeed, as illustrated in [21], the AgI service control problem, involving both packet routing and processing, can be reduced to a packet routing problem on a *layered graph* where cross-layer edges represent computation resources.

Motivated by such a connection and for ease of exposition, in this paper, w.l.o.g., we illustrate the developed approach focusing on the *single-commodity delay-constrained min-cost packet routing* problem. We remark that (i) it is still an open problem even in traditional communication networks, and (ii) the extension to distributed cloud networks hosting AgI services is presented in Appendix H.

B. Network Model

The considered packet routing network is modeled via a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where edge $(i, j) \in \mathcal{E}$ represents a network link supporting data transmission from node $i \in \mathcal{V}$ to $j \in \mathcal{V}$, and where δ_i^- and δ_i^+ denote the incoming and outgoing neighbor sets of node i, respectively.

Time is divided into equal-sized slots, and the available transmission resources and associated costs at each network link are quantified as:

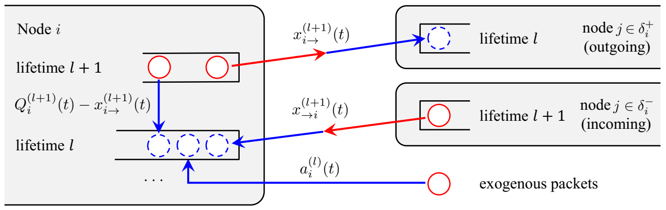


Fig. 2. Interaction between lifetime queues. Red and blue colors denote packet states and actions during *transmitting* and *receiving* phases, respectively.

- C_{ij}: the transmission capacity, i.e., the maximum number of data units (e.g., packets) that can be transmitted in one time slot, on link (i, j);
- e_{ij}: the unit transmission cost, i.e., the cost of transmitting one unit of data in one time slot, on link (i, j).

We emphasize that in the layered graph, cross-layer edges represent data processing, i.e., data streams pushed through these edges are interpreted as being processed by corresponding service functions, and the capacity and cost of these edges represent the *processing capacity* and *processing cost* of the associated computation resources (e.g., cloud/edge servers).

C. Arrival Model

In this work, we focus on a delay-sensitive application, assuming that each packet has a strict deadline by which it must be delivered to the destination $d \in \mathcal{V}$. In other words, each packet must be delivered within its *lifetime*, defined as the number of time slots between the current time and its deadline. A packet is called *effective* if its remaining lifetime l is positive, and *outdated* otherwise. In addition, we define *timely throughput* as the rate of effective packet delivery.

We assume that input packets can originate at any source node of the application, and in general, we assume that the set of source nodes can be any network node except the destination, $\mathcal{V} \setminus \{d\}$. The packet's initial lifetime $l \in \mathcal{L} \triangleq \{1, \cdots, L\}$ is determined by the application (based on the sensitivity of the contained information to delay), which can vary from packet to packet, with L denoting the maximum possible lifetime. Denote by $a_i^{(l)}(t)$ the number of exogenous packets (i.e., packets generated externally) of lifetime l arriving at node i. We assume that the arrival process is i.i.d. over time, with mean arrival rate $\lambda_i^{(l)} \triangleq \mathbb{E}\{a_i^{(l)}(t)\}$ and an upper bound of $A_{\max}$; besides, we define the corresponding vectors $\boldsymbol{a}(t) = \{a_i^{(l)}(t) : \forall\, i \in \mathcal{V},\, l \in \mathcal{L}\}$ and $\boldsymbol{\lambda} = \mathbb{E}\{\boldsymbol{a}(t)\}$.

D. Queuing System

Since each packet has its own delivery deadline, keeping track of data packets' lifetimes is essential. A key step is to construct a queuing system with distinct queues for packets of different *current lifetimes* $l \in \mathcal{L}$. In particular, we denote by $Q_i^{(l)}(t)$ the queue backlog of lifetime l packets at node i at time slot t, and define $\boldsymbol{Q}(t) = \{Q_i^{(l)}(t) : \forall\, i \in \mathcal{V},\, l \in \mathcal{L}\}$. Let $x_{ij}^{(l)}(t)$ be the *actual* number of lifetime l packets transmitted from node i to j at time t (which is different from a widely used assigned flow model, as explained in Remark 2).

Each time slot is divided into two phases, as illustrated in Fig. 2. In the *transmitting* phase, each node makes and executes transmission decisions based on observed queuing states. The number of lifetime $l+1$ packets at the end of this phase is given by

$$\breve{Q}_i^{(l+1)}(t) = Q_i^{(l+1)}(t) - x_{i\to}^{(l+1)}(t) \tag{1}$$

where $x_{i\to}^{(l+1)}(t) \triangleq \sum_{j\in\delta_i^+} x_{ij}^{(l+1)}(t)$ denotes the number of outgoing packets. In the *receiving* phase, the incoming packets, including those from neighbor nodes $x_{\to i}^{(l+1)}(t) \triangleq \sum_{j\in\delta_i^-} x_{ji}^{(l+1)}(t)$ as well as exogenously arriving packets $a_i^{(l)}(t)$, are loaded into the queuing system, and the queuing states are updated as:

$$Q_i^{(l)}(t+1) = \big[\breve{Q}_i^{(l+1)}(t) + x_{\to i}^{(l+1)}(t)\big] + a_i^{(l)}(t) \tag{2}$$

where lifetime $l+1$ packets, including those still in the queue as well as those arriving from incoming neighbors during the *transmitting* phase of time slot t (i.e., terms in the square bracket) turn into lifetime l packets during the *receiving* phase of time slot t. In addition, lifetime l exogenous packets, $a_i^{(l)}(t)$, also enter the lifetime l queue during the *receiving* phase of slot t. All such arriving packets become ready for transmission at the *transmitting* phase of slot $t+1$.

To sum up, the queuing dynamics are given by

$$Q_i^{(l)}(t+1) = Q_i^{(l+1)}(t) - x_{i\to}^{(l+1)}(t) + x_{\to i}^{(l+1)}(t) + a_i^{(l)}(t) \tag{3}$$

for $\forall\, i \in \mathcal{V}, l \in \mathcal{L}$.

In addition, we assume: 1) as the information contained in outdated packets is useless, i.e., outdated packets do not contribute to timely throughput, they are immediately dropped to avoid inefficient use of network resources:

$$Q_i^{(0)}(t) = 0, \quad \forall\, i \in \mathcal{V}, \tag{4}$$

and 2) for the destination node d, every effective packet is consumed as soon as it arrives, and therefore

$$Q_d^{(l)}(t) = 0, \quad \forall\, l \in \mathcal{L}. \tag{5}$$

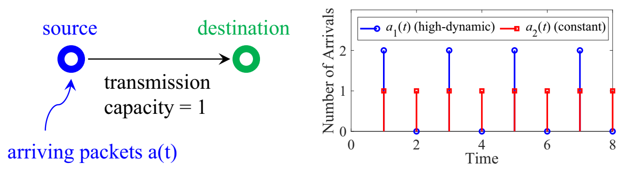
Considering the lifetime reduction over time slots, in general, we do not send packets of lifetime 1, i.e., $x_{ij}^{(1)}(t) = 0$, since the packets turn outdated at node j at the next time slot. The only exception occurs when $j = d$: we assume that the packets of lifetime $l = 1$ are consumed as soon as the destination node receives them, while they are still effective.

III. Problem Formulation

In this section, we introduce the admissible policy space, the reliability constraint, and the formalized *delay-constrained least-cost dynamic network control* problem.

A. Admissible Policy Space

The control policies of interest make packet routing and scheduling decisions at each time slot, which are dictated by the flow variables $\boldsymbol{x}(t) = \{x_{ij}^{(l)}(t) : \forall\, (i,j) \in \mathcal{E}, l \in \mathcal{L}\}$. In particular, we focus on the space of *admissible* control policies with decision flow variables satisfying:



(a) Single-hop network. (b) Arrival processes.

Fig. 3. A one-hop example network. Packets of lifetime $L = 1$ arrive at the source according to two arrival processes of equal mean arrival rate $\lambda = 1$.

1) non-negativity constraint, i.e.,

$$x_{ij}^{(l)}(t) \geq 0 \text{ for } \forall\,(i,j) \in \mathcal{E}, \text{ or } \boldsymbol{x}(t) \succeq 0; \tag{6}$$

2) peak link capacity constraint, i.e.,

$$x_{ij}(t) \triangleq \sum_{l \in \mathcal{L}} x_{ij}^{(l)}(t) \leq C_{ij}, \ \forall\,(i,j) \in \mathcal{E}; \tag{7}$$

3) availability constraint, i.e.,

$$x_{i\to}^{(l)}(t) \leq Q_i^{(l)}(t), \ \forall\, i \in \mathcal{V}, \, l \in \mathcal{L}. \tag{8}$$

The availability constraint (8) requires the total number of (scheduled) outgoing packets to not exceed those in the current queuing system, since we define $\boldsymbol{x}(t)$ as the actual flow (see Remark 2 for a detailed explanation). As will be shown throughout the paper, it plays an equivalent role to *flow conservation* in traditional packet routing formulations.

B. General Network Stability Region

In addition to the above admissibility constraints, we require the timely throughput achieved by the designed control policy to surpass a given level specified by the application, i.e.,

$$\overline{\{\mathbb{E}\,\{x_{\to d}(t)\}\}} \geq \gamma \|\boldsymbol{\lambda}\|_1 \tag{9}$$

where γ denotes the *reliability level*, $\|\boldsymbol{\lambda}\|_1$ is the total arrival rate, and $\overline{\{z(t)\}} \triangleq \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} z(t)$ denotes the long-term average of random process $\{z(t) : t \geq 0\}$.

The reliability constraint (9) imposes the requirement on the routing policy to provide reliable (delay-constrained) packet delivery. It forces packets to be routed *efficiently* and avoid excessive in-network packet drops due to lifetime expiry. The reliability level γ characterizes the robustness of the considered service to missing information, i.e., a percentage of up to $(1-\gamma)$ of the packets can be *dropped* without causing a significant performance loss. The reliability constraint plays an equivalent role to network stability in traditional packet routing formulations.

Definition 1: For a given capacitated network $\mathcal{G}$, we define the *delay-constrained stability region* as the set of $(f_{\boldsymbol{a}}, \gamma)$ pairs that can be supported by an admissible policy, i.e., the pairs $(f_{\boldsymbol{a}}, \gamma)$ such that there exists an admissible policy that satisfies (9) under an arrival process with probability density function (pdf) $f_{\boldsymbol{a}}$.

Note that via the complete information of the pdf $f_{\boldsymbol{a}}$, the mean arrival vector $\boldsymbol{\lambda}$ in (9) can be derived, which is employed to characterize the stability region in many existing works (e.g., [19], [23]). However, such first order characterization is not sufficient for the studied problem, as illustrated in Remark 1, showing the necessity to include the entire pdf information.

Remark 1: Consider the **Example** shown in Fig. 3, where the initial lifetime of every packet is equal to 1. The achievable reliability level is $\gamma_1 = 50\%$ under a high-dynamic arrival $a_1(t)$, and $\gamma_2 = 100\%$ under the constant arrival $a_2(t)$; while the two arrival processes have the same rate of 1. This example shows that: in addition to arrival rate, arrival dynamics can also impact the performance in the studied problem.

Remark 2: In the existing literature of stochastic network optimization (e.g., [19], [22], [23], [29]), a key element that has gained widespread adoption to improve tractability is the use of the *assigned* flow, which is different from the *actual* flow in that it does not need to satisfy the availability constraint (8). Dummy packets are created when there are not sufficient packets in the queue to support the scheduling decision, making the decision variables not constrained by the queuing process. Such formulation, however, is not suitable for delay-constrained routing, where reliable packet delivery is imposed on the *actual* packets received by the destination (via constraint (9)).

C. Problem Formulation

The goal is to develop an admissible control policy that guarantees reliable packet delivery, while minimizing overall network operational cost. Formally, we aim to find the policy with decisions $\{\boldsymbol{x}(t) : t \geq 0\}$ satisfying

$$\begin{aligned}
\mathscr{P}_0 : \ \min_{\boldsymbol{x}(t)\succeq 0} \quad & \overline{\{\mathbb{E}\,\{h(\boldsymbol{x}(t))\}\}} && \text{(10a)}\\
\text{s. t.} \quad & \overline{\{\mathbb{E}\,\{x_{\to d}(t)\}\}} \geq \gamma\|\boldsymbol{\lambda}\|_1 && \text{(10b)}\\
& x_{ij}(t) \leq C_{ij}, \ \forall\,(i,j) \in \mathcal{E} && \text{(10c)}\\
& x_{i\to}^{(l)}(t) \leq Q_i^{(l)}(t), \ \forall\, i \in \mathcal{V}, l \in \mathcal{L} && \text{(10d)}\\
& \boldsymbol{Q}(t) \text{ evolves by (3) – (5)} && \text{(10e)}
\end{aligned}$$

where the instantaneous cost of the decision $\boldsymbol{x}(t)$ is given by

$$h(t) = h(\boldsymbol{x}(t)) = \sum\nolimits_{(i,j)\in\mathcal{E}} e_{ij} x_{ij}(t) = \langle \boldsymbol{e}, \boldsymbol{x}(t) \rangle \tag{11}$$

with $\langle \cdot, \cdot \rangle$ denoting the inner product of the two vectors.

The above problem belongs to the category of CMDP, by defining the queuing vector $\boldsymbol{Q}(t)$ as the *state* and the flow variable $\boldsymbol{x}(t)$ as the *action*. However, note that the dimension of state-action space grows *exponentially* with the network dimension, which prohibits the application of the standard solution to this problem [32]. Even if we leave out the operational cost minimization aspect, it is still challenging to find an exact efficient solution to the remaining problem of timely throughput maximization, as studied in [30].

On the other hand, note that (10) deals with a queuing process, together with long-term average objective and constraints, which is within the scope of Lyapunov drift control [23]. However, it cannot be directly applied to solve (10) because: (i) the related queuing process (10e) is not of standard form;[1] (ii) the decision variables are actual flows and depend on the queuing states (10d), which is different from a widely used assigned flow model (see Remark 2).

IV. The Average Capacity Constrained Problem

The goal of this work is to derive an efficient approximate solution to $\mathscr{P}_0$. To this end, we start out with a less restricted setup in which only the average flow is constrained to be below capacity, leading to the following *relaxed* control problem:

$$\begin{aligned}
\mathscr{P}_1: \ \min_{\boldsymbol{x}(t)\succeq 0} \ & \overline{\{\mathbb{E}\{h(\boldsymbol{x}(t))\}\}} && \text{(12a)}\\
\text{s.t.} \ & \overline{\{\mathbb{E}\{x_{\to d}(t)\}\}} \geq \gamma\|\boldsymbol{\lambda}\|_1 && \text{(12b)}\\
& \overline{\{\mathbb{E}\{x_{ij}(t)\}\}} \leq C_{ij} && \text{(12c)}\\
& x_{i\to}^{(l)}(t) \leq Q_i^{(l)}(t) && \text{(12d)}\\
& \boldsymbol{Q}(t) \text{ evolves by (3) – (5)} && \text{(12e)}
\end{aligned}$$

which relaxes the peak capacity constraint (10c) by the corresponding average capacity constraint (12c).[2]

Mathematically, $\mathscr{P}_1$ is still a CMDP problem, making it challenging to solve. Instead of tackling it directly, in the following, we derive a tractable problem $\mathscr{P}_2$ corresponding to a *virtual network*, and establish the connection between them by showing that they have identical flow spaces, which allows to address $\mathscr{P}_1$ using the solution to $\mathscr{P}_2$ as a stepping-stone.

A. The Virtual Network

The virtual network control problem is cast as

$$\begin{aligned}
\mathscr{P}_2: \ \min_{\boldsymbol{x}(t)\succeq 0} \ & \overline{\{\mathbb{E}\{h(\boldsymbol{x}(t))\}\}} && \text{(13a)}\\
\text{s.t.} \ & \overline{\{\mathbb{E}\{x_{\to d}(t)\}\}} \geq \gamma\|\boldsymbol{\lambda}\|_1 && \text{(13b)}\\
& x_{ij}(t) \leq C_{ij} && \text{(13c)}\\
& \bar{x}_{i\to}^{(\geq l)} \leq \bar{x}_{\to i}^{(\geq l+1)} + \lambda_i^{(\geq l)} && \text{(13d)}
\end{aligned}$$

where $\bar{x}_{i\to}^{(\geq l)} = \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\big\{x_{i\to}^{(\geq l)}(t)\big\}$ denotes the average transmission rate of packets with lifetime $\geq l$, with $x_{i\to}^{(\geq l)}(t) = \sum_{\ell=l}^{L} x_{i\to}^{(\ell)}(t)$ (similarly for $\bar{x}_{\to i}^{(\geq l+1)}$).

A crucial difference in the derivation of $\mathscr{P}_2$ is to replace the availability constraint (12d) by (13d), which states the fact that the lifetime of the packets must decrease (by at least 1) as they traverse any node i, and thus is called the *causality* constraint. As a consequence, we eliminate the unconventional queuing process (12e) and the dependency of $\boldsymbol{x}(t)$ on $\boldsymbol{Q}(t)$, i.e., the two factors resulting in the failure of employing the LDP approach to address $\mathscr{P}_1$. Especially, we will use $\boldsymbol{\nu}(t)$ (instead of $\boldsymbol{x}(t)$) to represent the decisions determined in $\mathscr{P}_2$, referred to as the virtual flow.

[1] In the designed lifetime-based queuing system, a packet can traverse queues of reducing lifetimes and eventually get dropped when entering the lifetime 0 queue. On the other hand, in traditional queuing systems [19], [23], a packet stays in the same queue until selected for operation; in addition, since there are no packet drops, queue build up contributes to network congestion and creates *pressure* driving packet transmission [23].

[2] We note that such an average-constrained setting may find interesting applications of its own in next-generation virtual networks that allow elastic scaling of network resources [4].

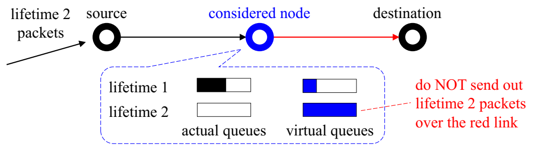


Fig. 4. Illustration of the devised virtual system. The source node supplies packets of lifetime 2, which arrive as lifetime 1 packets to the actual queue of the considered node. In the virtual system, the considered node is allowed to supply packets of any lifetime to the destination by *borrowing* them from the reservoir and building up in the corresponding virtual queue. The virtual queue of lifetime 1 is stable, since the received lifetime 1 packets from the source node can compensate the borrowed packets; while the virtual queue of lifetime 2 builds up, pushing the node to stop sending out more lifetime 2 packets. The decision derived from the stability of the virtual system is aligned with the desired operation of the actual network, as only lifetime 1 packets are available for transmission at the considered node.

1) Virtual Queue: Although there is no explicit queuing system in $\mathscr{P}_2$, it consists of long-term average objective and constraints, which can be addressed via the LDP control of a virtual queuing system [23]. More concretely, the *virtual queuing system* $\boldsymbol{U}(t) = \{U_d(t)\} \cup \{U_i^{(l)}(t) : i \in \mathcal{V}\setminus\{d\},\, l \in \mathcal{L}\}$ must be stabilized to ensure (13b) and (13d), defined as

$$\begin{aligned}
U_d(t+1) &= \max\big\{U_d(t) + \gamma A(t) - \nu_{\to d}(t),\, 0\big\}, && \text{(14a)}\\
U_i^{(l)}(t+1) &= \max\big\{U_i^{(l)}(t) + \nu_{i\to}^{(\geq l)}(t) - \nu_{\to i}^{(\geq l+1)}(t)\\
&\qquad - a_i^{(\geq l)}(t),\, 0\big\}. && \text{(14b)}
\end{aligned}$$

where $A(t) = \sum_{i\in\mathcal{V},\, l\in\mathcal{L}} a_i^{(l)}(t)$ is the total amount of packets arriving in the network at time slot t.[3] We refer to (14a) and (14b) as the virtual queues associated with node d and i.

To sum up, it is equivalent to cast $\mathscr{P}_2$ as

$$\begin{aligned}
\mathscr{P}_2^{\mathrm{e}}: \ \min_{\boldsymbol{\nu}(t)} \ & \overline{\{\mathbb{E}\{h(\boldsymbol{\nu}(t))\}\}} && \text{(15a)}\\
\text{s.t.} \ & \text{stabilize } \boldsymbol{U}(t) \text{ evolving by (14)} && \text{(15b)}\\
& 0 \leq \nu_{ij}(t) \leq C_{ij} \quad \forall (i,j) \in \mathcal{E}. && \text{(15c)}
\end{aligned}$$

2) Physical Interpretation: When deriving the virtual network control problem $\mathscr{P}_2$, we relax the precedence constraint that imposes that a packet cannot be transmitted from a node before it arrives at the given node. Instead, we assume that each node in the virtual network is a *data-reservoir* and has access to abundant (virtual) packets of any lifetime. At every time slot, each node checks packet requests from its outgoing neighbors and supplies such needs using the virtual packets borrowed from the reservoir, which are compensated when it receives packets of the same lifetime (either from incoming neighbors or exogenous arrivals). The virtual queues can be interpreted as the *data deficits* (difference between outgoing and incoming packets) of the corresponding data-reservoirs. Specially, in (14a), the destination reservoir *sends out* $\gamma A(t)$ packets to the end user (to meet the reliability requirement), while *receiving* $\nu_{\to d}(t)$ in return. When (13b) and (13d) are satisfied, or the virtual queues are stabilized, the network nodes do not need to embezzle virtual packets from the reservoirs;

[3] Here we use $A(t)$ instead of $\|\boldsymbol{\lambda}\|_1$ as the latter information is usually not available in practice; furthermore, if the arrival information cannot be obtained immediately, delayed information, i.e., $A(t-\tau)$ with $\tau > 0$, can be used as an alternative, which does not impact the result of time average.

since the achieved network flow can be attained by the actual packets, it can serve as guidance for packet routing in the actual network (see Fig. 4 for illustration).

B. Connections Between $\mathscr{P}_1$ and $\mathscr{P}_2$

We now describe key connections between the actual and virtual network control problems.

Definition 2 (Feasible Policy): For problem $\mathscr{P}_\iota$ ($\iota = 1, 2$), a policy p is called *feasible* if it makes decisions satisfying (12b) – (12e) ($\iota = 1$) or (13b) – (13d) ($\iota = 2$). The set of feasible policies is called *feasible policy space* $\mathcal{F}_\iota$.

Definition 3 (Flow Assignment): Given a feasible policy p for problem $\mathscr{P}_\iota$ ($\iota = 1, 2$) (with decisions $\{\boldsymbol{x}_p(t) : t \geq 0\}$), the achieved *flow assignment* is defined as $\boldsymbol{x}_p = \overline{\{\mathbb{E}\{\boldsymbol{x}_p(t)\}\}}$, i.e., the vector of transmission rates for packets with different lifetimes on all network links. Furthermore, the *flow space* is defined as the set of all achievable flow assignments, i.e., $\Gamma_\iota = \{\boldsymbol{x}_p : p \in \mathcal{F}_\iota\}$.

Definition 4 (Stability Region): For problem $\mathscr{P}_\iota$ ($\iota = 1, 2$), the *stability region* Λ_ι is defined as the set of $(\boldsymbol{\lambda}, \gamma)$ pairs, under which the feasible policy space $\mathcal{F}_\iota$ is non-empty.

We make the following clarifications about the above definitions: (i) we will prove (in Theorem 1) that the stability region of $\mathscr{P}_1$ only depends on the mean arrival rate $\boldsymbol{\lambda}$, in contrast to the general *Definition 1* which involves the arrival pdf $f_{\boldsymbol{a}}$; and so is that of $\mathscr{P}_2$, which is clear from its definition (13); (ii) the feasible policy space and the flow space are associated with a certain point $(\boldsymbol{\lambda}, \gamma)$ in the stability region; (iii) since the networks considered in $\mathscr{P}_1$ and $\mathscr{P}_2$ are of the same topology, the flow assignment vectors are of the same dimension.

We then reveal the intimate relationship between the two problems by the following three results.

Proposition 1: The availability constraint (12d) implies the causality constraint (13d).

Proof: See Appendix A in supplementary material. □

Theorem 1: For a given network, the stability regions of $\mathscr{P}_1$ and $\mathscr{P}_2$ are identical, i.e., $\Lambda_1 = \Lambda_2$. In addition, a pair $(\boldsymbol{\lambda}, \gamma)$ is interior to the stability region Λ_ι ($\iota = 1, 2$) if and only if there exist flow variables $\boldsymbol{x} = \{x_{ij}^{(l)} \geq 0 : \forall\, (i,j) \in \mathcal{E}, l \in \mathcal{L}\}$, such that for $\forall\, i \in \mathcal{V}$, $(i,j) \in \mathcal{E}$, $l \in \mathcal{L}$,

$$x_{\to d} \geq \gamma \|\boldsymbol{\lambda}\|_1 \tag{16a}$$

$$x_{ij} \leq C_{ij},\ \forall\, (i,j) \in \mathcal{E} \tag{16b}$$

$$x_{\to i}^{(\geq l+1)} + \lambda_i^{(\geq l)} \geq x_{i\to}^{(\geq l)},\ \forall\, i \in \mathcal{V}, l \in \mathcal{L} \tag{16c}$$

$$x_{ij}^{(0)} = x_{dk}^{(l)} = 0,\ \forall\, k \in \delta_d^+, (i,j) \in \mathcal{E}, l \in \mathcal{L}. \tag{16d}$$

Furthermore, $\forall\, (\boldsymbol{\lambda}, \gamma) \in \Lambda_\iota$, there exists a feasible randomized policy that achieves the optimal cost.

Proof: See Appendix B, C in supplementary material. □

Proposition 2: For $\forall\, (\boldsymbol{\lambda}, \gamma) \in \Lambda_1 = \Lambda_2$, the two problems have identical flow spaces, i.e., $\Gamma_1 = \Gamma_2$.

Proof: By Theorem 1, $\mathscr{P}_1$ and $\mathscr{P}_2$ have the same stability region, i.e, $\Lambda_1 = \Lambda_2$. Consider a point in the stability region $\Lambda_1 = \Lambda_2$. For any flow assignment $\boldsymbol{x} \in \Gamma_1$, there exists a feasible policy $p_1 \in \mathcal{F}_1$, with decision variables $\{\boldsymbol{x}_1(t) : t \geq 0\}$, that attains flow assignment $\boldsymbol{x}$, i.e., $\overline{\{\mathbb{E}\{\boldsymbol{x}_1(t)\}\}} = \boldsymbol{x}$. Therefore, $\{\boldsymbol{x}_1(t) : t \geq 0\}$ satisfies (12b) – (12e), which implies that $\boldsymbol{x}$ satisfies all the conditions in (16) (by Proposition 1). Using the method provided in Appendix C-B1, we can construct a feasible randomized policy $p_2 \in \mathcal{F}_2$, with decision variables $\{\boldsymbol{x}_2(t) : t \geq 0\}$, that achieves the same flow assignment, i.e., $\overline{\{\mathbb{E}\{\boldsymbol{x}_2(t)\}\}} = \boldsymbol{x}$, for $\mathscr{P}_2$. Therefore, $\boldsymbol{x} \in \Gamma_2$, and thus $\Gamma_1 \subset \Gamma_2$. The reverse direction $\Gamma_2 \subset \Gamma_1$ can be shown via the same argument. Hence, $\Gamma_1 = \Gamma_2$. □

The above propositions are explained in the following: by Proposition 1 and the fact that (13c) implies (12c), the feasible policy spaces satisfy $\mathcal{F}_1 \nsubseteq \mathcal{F}_2$ and $\mathcal{F}_2 \nsubseteq \mathcal{F}_1$; while Theorem 1 shows that they lead to the same stability region, by presenting an explicit, identical characterization (16) (where (16c) is the generalized lifetime-driven *flow conservation* law), which is in the form of a linear programming (LP) problem with $L|\mathcal{E}|$ variables (and thus of pseudo polynomial complexity); Proposition 2 further shows that $\mathscr{P}_1$ and $\mathscr{P}_2$ share the same flow space (for any point in the stability region), which is a crucial property since the two metrics of interest, i.e., timely throughput (9) and operational cost (11), are both *linear* functions of the flow assignment.

Corollary 1: $\mathscr{P}_1$ and $\mathscr{P}_2$ have the same optimal value.

Proof: Consider a feasible policy $p_1 \in \mathcal{F}_1$, whose decisions $\boldsymbol{x}_1(t)$ attain flow assignment $\boldsymbol{x}$. According to the Proposition 2, there exists a feasible policy $p_2 \in \mathcal{F}_2$ attaining the same flow assignment $\boldsymbol{x}$ by making decisions $\boldsymbol{x}_2(t)$. The operational cost satisfies

$$\begin{aligned}\overline{\{\mathbb{E}\{h(\boldsymbol{x}_1(t))\}\}} &= \overline{\{\mathbb{E}\{\langle \boldsymbol{e}, \boldsymbol{x}_1(t)\rangle\}\}} = \langle \boldsymbol{e}, \overline{\{\mathbb{E}\{\boldsymbol{x}_1(t)\}\}}\rangle \\ &= \langle \boldsymbol{e}, \boldsymbol{x}\rangle = \langle \boldsymbol{e}, \overline{\{\mathbb{E}\{\boldsymbol{x}_2(t)\}\}}\rangle \\ &= \overline{\{\mathbb{E}\{\langle \boldsymbol{e}, \boldsymbol{x}_2(t)\rangle\}\}} = \overline{\{\mathbb{E}\{h(\boldsymbol{x}_2(t))\}\}}.\end{aligned} \tag{17}$$

The reverse direction can be shown by the same argument. As a result, they have the same range (when treating the cost as a function of the policy), and thus optimal value. □

Corollary 2: Given a feasible policy to $\mathscr{P}_2$, we can construct a feasible randomized policy for $\mathscr{P}_1$ to achieve the same flow assignment.

Proof: Suppose $\{\boldsymbol{\nu}(t) : t \geq 0\} \in \mathcal{F}_2$. The associated flow assignment $\boldsymbol{\nu} = \overline{\{\mathbb{E}\{\boldsymbol{\nu}(t)\}\}}$ satisfies (16), and we can construct a feasible randomized policy for $\mathscr{P}_1$ as follows (see Appendix B-B1 for details): at each time slot, for every packet of lifetime $l \in \mathcal{L}$ in the queuing system, node $i \in \mathcal{V}$ selects the outgoing neighbor $j \in \delta_i^+$ for it according to the pdf

$$\alpha_i^{(l)}(j) = \nu_{ij}^{(l)} \big/ \big(\nu_{\to i}^{(\geq l+1)} + \lambda_i^{(\geq l)} - \nu_{i\to}^{(\geq l+1)}\big) \tag{18}$$

otherwise the packet stays in node i. It is shown in Appendix B that this policy achieves flow assignment $\boldsymbol{\nu}$. □

V. Solution to Average-Constrained Network

In this section, we take advantage of the LDP approach to address $\mathscr{P}_2^{\text{e}}$ and guide the design of a fully distributed, near-optimal randomized algorithm for $\mathscr{P}_1$ (by Corollary 2).

A. Optimal Virtual Flow

We first present the LDP-based algorithm to solve $\mathscr{P}_2^{\text{e}}$ given by (15). Define the Lyapunov function as $L(t) = \|\boldsymbol{U}(t)\|_2^2/2$, and the Lyapunov drift $\Delta(\boldsymbol{U}(t)) = L(t+1) - L(t)$. The

LDP approach aims to minimize a linear combination of an upper bound of the Lyapunov drift (which can be derived by some standard manipulation [23]) and the objective function weighted by a tunable parameter V, or

$$\Delta(\boldsymbol{U}(t)) + Vh(\boldsymbol{\nu}(t)) \leq B - \langle \tilde{\boldsymbol{a}}, \boldsymbol{U}(t) \rangle - \langle \boldsymbol{w}(t), \boldsymbol{\nu}(t) \rangle \quad (19)$$

where B is a constant, $\tilde{\boldsymbol{a}} = \{-\gamma A(t)\} \cup \{a_i^{(\geq l)}(t) : \forall i \in \mathcal{V} \setminus \{d\}, l \in \mathcal{L}\}$, and the weights $\boldsymbol{w}(t)$ are given by

$$w_{ij}^{(l)}(t) = -Ve_{ij} - U_i^{(\leq l)}(t) + \begin{cases} U_d(t) & j = d \\ U_j^{(\leq l-1)}(t) & j \neq d \end{cases} \quad (20)$$

where the superscript $^{(\leq l)}$ refers to the operation of $\sum_{\ell=1}^{l}$.

To sum up, at every time slot, the algorithm decides the virtual flow $\boldsymbol{\nu}(t)$ by addressing the following problem

$$\max_{\boldsymbol{\nu}(t)} \ \langle \boldsymbol{w}(t), \boldsymbol{\nu}(t) \rangle, \ \text{s.t.} \ \ 0 \leq \nu_{ij}(t) \leq C_{ij}, \ \forall\,(i,j) \in \mathcal{E} \quad (21)$$

and the solution to it is in the *max-weight* fashion. More concretely, for each link (i,j), we first find the *best* lifetime $l^\star$ with the *largest weight*, and devote all the transmission resource to serve packets of this lifetime if the weight is positive. Therefore, the optimal virtual flow assignment is

$$\nu_{ij}^{(l)}(t) = C_{ij}\, \mathbb{I}\{l = l^\star, w_{ij}^{(l^\star)}(t) > 0\} \quad (22)$$

where $l^\star = \arg\max_{l \in \mathcal{L}} \ w_{ij}^{(l)}(t)$, $\mathbb{I}\{\cdot\}$ is the indicator function.

To implement the above algorithm, at each time slot, a considered node exchanges the virtual queue information with its neighbor nodes (to calculate the weight of each lifetime by (20)), and decides the virtual flow according to (22), which can be completed in a fully distributed manner; the computational complexity at node i is given by $\mathcal{O}(L|\delta_i^+|)$.

B. Flow Matching

The algorithm developed above can provide a near-optimal (will be proved in next subsection) solution $\{\boldsymbol{\nu}(t) : t \geq 0\}$ to $\mathscr{P}_2$, from which we will design a feasible, near-optimal policy for $\mathscr{P}_1$ in this section. The decided (actual) flow is denoted by $\boldsymbol{\mu}(t)$, to distinguish it from the virtual flow $\boldsymbol{\nu}(t)$.

We will design an *admissible* policy for $\mathscr{P}_1$ (i.e., satisfying (12c) – (12d)) to pursue the goal of *flow matching*, i.e.,

$$\overline{\{\boldsymbol{\mu}(t)\}} = \overline{\{\boldsymbol{\nu}(t)\}}. \quad (23)$$

The reason to set the above goal is two-fold: (i) it ensures that the designed policy can attain the same throughput and cost performance (recall that both metrics are linear functions of the flow assignment) as the virtual flow, which is feasible (satisfying the reliability constraint) and achieves near-optimal cost performance, (ii) the existence of the policy is guaranteed (as a result of identical flow spaces); actually, given the feasible solution $\{\boldsymbol{\nu}(t)\}$, Corollary 2 presents a construction procedure of a feasible policy for $\mathscr{P}_1$ to realize the goal.

Corollary 2 requires the exact values of $\overline{\{\boldsymbol{\nu}(t)\}}$ and $\boldsymbol{\lambda}$ as input, which are not available in practice. As an alternative, we employ the corresponding empirical values, i.e., the finite-horizon average of the virtual flow and the arrival rate

$$\bar{\boldsymbol{\nu}}(t) = \frac{1}{t}\sum_{\tau=0}^{t-1} \boldsymbol{\nu}(\tau), \ \ \hat{\boldsymbol{\lambda}}(t) = \frac{1}{t}\sum_{\tau=0}^{t-1} \boldsymbol{a}(\tau) \quad (24)$$

Algorithm 1 Randomized Flow-Matching Algorithm

```
1: for $t \geq 0$ and $i \in \mathcal{V}$ do
2:     Solve the virtual flow $\boldsymbol{\nu}(t)$ from (21);
3:     Update the empirical averages $\bar{\boldsymbol{\nu}}(t)$ and $\hat{\boldsymbol{\lambda}}(t)$ by (24);
4:     Update probability values $\{\hat{\alpha}_i^{(l)}(j) : j \in \delta_i^+\}_{l \in \mathcal{L}}$ by (18) (using the above empirical averages);
5:     for $l \in L$ do
6:         For each packet in $Q_i^{(l)}(t)$, decide its outgoing link according to pdf $\{\hat{\alpha}_i^{(l)}(j) : j \in \delta_i^+\}$;
7:     end for
8: end for
```

to calculate the probability values in (18), by which we decide the outgoing flow at time slot t. Since the above empirical values are updated at every time slot, it leads to a time-varying randomized policy; as $\bar{\boldsymbol{\nu}}(t) \to \overline{\{\boldsymbol{\nu}(t)\}}$ and $\hat{\boldsymbol{\lambda}}(t) \to \boldsymbol{\lambda}$ asymptotically, the policy gradually converges.[4]

The proposed control policy is summarized in Algorithm 1, and we emphasize that (i) at a given time slot, the policy in Corollary 2 makes i.i.d. decisions for packets with the same lifetime (i.e., fix the lifetime l, the pdf $\{\hat{\alpha}_i^{(l)}(j) : j \in \delta_i^+\}$ to determine the routing decision for each packet is the same). It is equivalent to make flow-level decisions based on packets' lifetime, by generating multinomial random variables with parameter $Q_i^{(l)}(t)$ and the common pdf; (ii) in addition to deciding the virtual flow, the developed randomized policy requires each node to update the empirical averages (24), calculate the pdf $\hat{\boldsymbol{\alpha}}$, and make the decisions at a complexity of $\mathcal{O}(L|\delta_i^+|)$.

Remark 3: For the studied packet routing problem (where flow scaling is not relevant), under the widely used assumption of Poisson arrivals, we can show (see Appendix D) that the instantaneous flow size $x_{ij}(t), \forall\,(i,j),\, t$, follows a Poisson distribution, which enjoys good concentration bounds.

Remark 4: In [30], a subproblem of $\mathscr{P}_1$ is studied, which involves constraints on average capacity and timely throughput, while leaving out the aspect of operational cost. The formulated CMDP problem is solved by a dynamic programming algorithm, which can also be addressed following the same procedure presented in this section, at a lower complexity.

C. Performance Analysis

In this section, we first prove that the LDP-based algorithm (for $\mathscr{P}_2$) stabilizes the virtual queues (and consequently, the timely throughput satisfies the reliability constraint (13b)) and attains near-optimal cost performance; then we show that the flow matching-based randomized policy (for $\mathscr{P}_1$) achieves the same throughput and cost performance as the previous algorithm. The effects of parameter V are also analyzed.

1) Virtual Network: In addition to proving that the algorithm stabilizes the virtual queues, we analyze the effect of V on the ε-convergence time defined as follows.

[4] It is possible that $\bar{\boldsymbol{\nu}}(t)$ can violate (16c) at some time slot, which is not qualified to construct a valid randomized policy. However, as $t \to \infty$, $\bar{\boldsymbol{\nu}}(t)$ converges to $\overline{\{\boldsymbol{\nu}(t)\}}$, which satisfies the constraints. With this asymptotic guarantee, when such violation occurs, we can choose not to update the control policy at that time slot.

Definition 5 (ε-Convergence Time): The ε-convergence time t_ε is the running time for the average solution to achieve a reliability within a margin of ε from the desired value, i.e.,

$$t_\varepsilon \triangleq \min_{\tau} \Big\{ \sup_{s \geq \tau} \Big[\gamma \|\boldsymbol{\lambda}\|_1 - \sum_{t=0}^{s-1} \frac{\mathbb{E}\{\nu_{\to d}(t)\}}{s} \Big] \leq \varepsilon \Big\}. \quad (25)$$

The existence of t_ε (under the proposed algorithm) is shown in Appendix E-B for any $\varepsilon > 0$.

Proposition 3: For any point in the interior of the stability region, the virtual queues are mean rate stable under the proposed algorithm with a convergence time $t_\varepsilon \sim \mathcal{O}(V)$ for any $\varepsilon > 0$, and the achieved cost performance satisfies

$$\overline{\{\mathbb{E}\{h(\boldsymbol{\nu}(t))\}\}} \leq h_2^\star(\boldsymbol{\lambda}, \gamma) + \frac{B}{V} \quad (26)$$

where $h_2^\star(\boldsymbol{\lambda}, \gamma)$ denotes the optimal cost performance that can be achieved under $(\boldsymbol{\lambda}, \gamma)$ in $\mathscr{P}_2$.

Proof: See Appendix E in supplementary material. □

We make the following clarifications about the above proposition, (i) for a finite horizon, the reliability (13b) and causality (13d) constraints might not be satisfied; (ii) the virtual queues are stabilized, implying that the two constraints hold asymptotically; (iii) by pushing the parameter $V \to \infty$, the achieved cost performance approaches the optimal cost (since the gap B/V vanishes), with a tradeoff in convergence time.

2) Performance of Algorithm 1:

Proposition 4: For any point in the interior of the stability region, Algorithm 1 is feasible for $\mathscr{P}_1$, while achieving the near-optimal cost performance of $h(\overline{\{\boldsymbol{\nu}(t)\}})$.

Proof: Algorithm 1 makes decisions for the packets in the queuing system, and thus satisfying the constraints (12d). Besides, as $\lim_{t\to\infty} \bar{\boldsymbol{\nu}}(t) = \overline{\{\boldsymbol{\nu}(t)\}}$ and $\lim_{t\to\infty} \hat{\boldsymbol{\lambda}}(t) = \boldsymbol{\lambda}$, the instantaneous policy converges to a fixed policy constructed from $\overline{\{\boldsymbol{\nu}(t)\}}$ and $\boldsymbol{\lambda}$, which achieves the same flow assignment $\overline{\{\boldsymbol{\mu}(t)\}} = \overline{\{\boldsymbol{\nu}(t)\}}$ as is proved in Corollary 2, leading to identical throughput and cost performance. □

Remark 5: We note that Algorithm 1 relies on the knowledge of the arrival rate (via (18) in step 4), and the empirical estimate (24) we use for implementation may be subject to estimation errors that can impact the attained cost performance. As shown in Appendix F, in some extreme cases, the estimation error can lead to a considerable performance loss, driven by the Lagrangian multiplier (or shadow price) associated with the constraints involving λ. However, under i.i.d. arrivals, the estimated rate converges to the true value, and Algorithm 1 is guaranteed to achieve near-optimal *asymptotic* performance.

VI. Solution to Peak-Constrained Network

In this section, we aim to address the original problem $\mathscr{P}_0$ (with peak-capacity constraint), leveraging the flow matching technique we develop in the previous section.

There are two problems we need to address:

(i) the actual flow (decided by the randomized policy) can violate the peak capacity constraint (10c);
(ii) the actual and virtual flow spaces are not identical, i.e., $\Gamma_0 \subset \Gamma_2$ (while in the average-constrained case, $\Gamma_1 = \Gamma_2$).

To address problem (i), we propose a request queue stability approach in order to constrain instantaneous transmission rates. For problem (ii), we introduce an auxiliary variable $\epsilon_{ij}, \forall (i,j)$ to represent the gap in flow spaces, leading to the following optimization problem over $\{\boldsymbol{\nu}(t), \boldsymbol{\mu}(t), \boldsymbol{\epsilon}\}$:

$$\mathscr{P}_3: \ \min \ \overline{\{\mathbb{E}\{h(\boldsymbol{\nu}(t))\}\}} \quad (27a)$$
$$\text{s.t.} \ \ \nu_{ij}(t) \leq C_{ij} - \epsilon_{ij}, \text{(11b), (11d), (11e)}, \quad (27b)$$
$$\overline{\{\mathbb{E}\{\mu_{ij}(t)\}\}} = \overline{\{\mathbb{E}\{\nu_{ij}(t)\}\}}, \text{(8c) -- (8f)}, \quad (27c)$$
$$0 \preceq \boldsymbol{\epsilon} \triangleq \{\epsilon_{ij}\} \preceq \{C_{ij}\}. \quad (27d)$$

While solving $\mathscr{P}_3$ in a joint manner is difficult, we propose an iterative optimization approach:

i) fix $\boldsymbol{\epsilon}$ and $\boldsymbol{\mu}(t)$: find $\boldsymbol{\nu}(t)$ by LDP control (19), and derive the virtual flow assignment with optimal operational cost;
ii) fix $\boldsymbol{\epsilon}$ and $\boldsymbol{\nu}(t)$: find $\boldsymbol{\mu}(t)$ with the goal of flow matching (i.e., by stabilizing the request queues);
iii) fix $\boldsymbol{\nu}(t)$ and $\boldsymbol{\mu}(t)$: update $\boldsymbol{\epsilon}$ based on the gap between optimal and achievable rates, i.e., $\overline{\{\mathbb{E}\{\boldsymbol{\nu}(t)\}\}} - \overline{\{\mathbb{E}\{\boldsymbol{\mu}(t)\}\}}$, which is non-zero if (25c) is violated.

Due to the randomness of network states, we introduce a *time frame* structure: step i) and ii) are executed on a per-slot basis, while step iii) on a per-frame basis (to obtain better rate estimates). The developed algorithm, referred to as *reliable cloud network control* (RCNC), is described in Algorithm 2.

A. Request Queue

We propose to achieve (23) by making admissible flow decisions (i.e., satisfying (10c) – (10e)) to stabilize the *request queues* $\boldsymbol{R}(t) = \{R_{ij}^{(l)}(t) : \forall (i,j) \in \mathcal{E}, l \in \mathcal{L}\}$, defined as

$$R_{ij}^{(l)}(t+1) = R_{ij}^{(l)}(t) + \bar{\nu}_{ij}^{(l)}(t) - \mu_{ij}^{(l)}(t) \quad (28)$$

where $\bar{\boldsymbol{\nu}}(t)$ is given by (24), and we still adopt the notation $\boldsymbol{\mu}(t)$ to refer to the actual flow decided in $\mathscr{P}_0$, without causing ambiguity ($\mathscr{P}_1$ is not relevant in this section).

We consider the *n-slot look-ahead* scheme, under which the current decision is made together with $n-1$ (anticipated) future decisions. Such a scheme is employed since it creates flexibility for a packet to change its lifetime by delaying transmission, in favor of relieving the burden of the request queue with the heaviest backlog, as well as balancing the transmission load. From a formal point of view, we make decisions for n time slots (starting from the current time slot) to optimize the *multi-slot Lyapunov drift*, defined as

$$\Delta_n(\boldsymbol{R}(t)) \triangleq \frac{\|\boldsymbol{R}(t+n-1)\|_2^2 - \|\boldsymbol{R}(t)\|_2^2}{2}. \quad (29)$$

An upper bound for the multi-slot drift is derived in the following. We apply telescope sum on the queuing dynamics (28) for the period $t, \cdots, t+(n-1)$, which leads to

$$R_{ij}^{(l)}(t+n-1) = R_{ij}^{(l)}(t) + \sum_{\tau=t}^{t+n-1} \big[\bar{\nu}_{ij}^{(l)}(\tau) - \mu_{ij}^{(l)}(\tau)\big]. \quad (30)$$

Following the same procedure as in Section IV-A, we obtain

$$\Delta_n(\boldsymbol{R}(t)) \leq B'(t) - \textstyle\sum_{(i,j)\in\mathcal{E}} \langle \boldsymbol{R}_{ij}(t), \boldsymbol{M}_{ij}\boldsymbol{1} \rangle \quad (31)$$

where $\boldsymbol{M}_{ij}$ is a $L \times n$ matrix associated with link (i,j), with column τ $(0 \le \tau < n)$ representing the vector $\boldsymbol{\mu}_{ij}(t+\tau)$, and $B'(t)$ gathers all the uncontrollable terms.

The goal is to minimize the bound for multi-slot drift, by making admissible flow decisions for the n time slots. By applying the queuing dynamics (3) recursively, the availability constraint (10d) at the $(t+\tau)$-th time slot can be cast as

$$\sum_{j\in\delta_i^+} g_{\tau+1}(\boldsymbol{M}_{ij}) - \boldsymbol{D}\sum_{j\in\delta_i^-} g_\tau(\boldsymbol{M}_{ji}) \le g_{\tau+1}(\boldsymbol{A}_i),\ \forall i \quad (32)$$

where $\boldsymbol{A}_i = [\boldsymbol{Q}_i(t), \boldsymbol{a}_i(t), \cdots, \boldsymbol{a}_i(t+n-1)]$ is the arrival matrix, with the columns $\boldsymbol{a}_i(t+\tau) = \{a_i^{(l)}(t+\tau) : l \in \mathcal{L}\}$ representing the exogenous arrivals in the future; g denotes the delay function, given by

$$g_{\tau+1}(\boldsymbol{X}) = \sum\nolimits_{s=0}^{\tau} \boldsymbol{D}^{\tau-s}\boldsymbol{X}[\,:\,,s] \quad (33)$$

in which $\boldsymbol{D}$ is the delay matrix of order n, and $\boldsymbol{X}[\,:\,,\tau]$ is the τ-th column of matrix $\boldsymbol{X}$. We clarify that $\boldsymbol{a}_i(t+\tau)$ are *random* vectors, leading to a complex stochastic optimization problem. We simplify the problem by replacing the random vectors with their estimated averages, i.e., empirical arrival rates $\hat{\boldsymbol{\lambda}}_i$. This reduces $\boldsymbol{A}_i$ to a deterministic matrix

$$\boldsymbol{A}_i^{\text{emp}} = [\boldsymbol{Q}_i(t), \hat{\boldsymbol{\lambda}}_i, \cdots, \hat{\boldsymbol{\lambda}}_i], \quad (34)$$

and the problem to a common LP that can be addressed by standard solvers.

To sum up, at each slot, the proposed RCNC algorithm solves the following LP to determine the transmission flow

$$\mathscr{H}:\quad \max_{\mathcal{M}} \sum_{(i,j)\in\mathcal{E}} \langle \boldsymbol{R}_{ij}(t), \boldsymbol{M}_{ij}\boldsymbol{1}\rangle \quad (35a)$$

$$\text{s.t.}\quad \boldsymbol{1}^{\mathrm{T}}\boldsymbol{M}_{ij} \preceq C_{ij},\ \forall (i,j)\in\mathcal{E} \quad (35b)$$

$$(32)\ \text{with}\ \boldsymbol{A}_i = \boldsymbol{A}_i^{\text{emp}},\ \forall i,\ 0\le\tau<n \quad (35c)$$

$$\boldsymbol{M}_{ij} \succeq 0,\ \forall (i,j)\in\mathcal{E} \quad (35d)$$

where $\mathcal{M} = \cup_{i\in\mathcal{V}}\mathcal{M}_i \triangleq \{\boldsymbol{M}_{ij} : j \in \delta_i^+\}$, and (35b) is the peak capacity constraint. Note that the above problem involves all the flow variables of the entire network ($nL|\mathcal{E}|$ in total); and due to (35c), the decisions of the nodes are dependent on each other, which are determined in a centralized manner.

After the optimal solution $\boldsymbol{M}_{ij}^\star$ is obtained, its first column $\boldsymbol{\mu}_{ij}^\star(t) = \boldsymbol{M}_{ij}^\star[\,:\,,0]$ will be used as the decided flow for the current time slot. The rest of its columns are discarded, and the procedure repeats at the next time slot based on the updated information to make the corresponding decision.

Remark 6 (Choice of n): An intuitive choice is $n = L$, since the packets of the largest lifetime L will be outdated after L time slots, and we ignore the effects of the current decision at the distant time slots in the future. Another choice is $n = 1$, which simplifies the formulation by considering only the current time slot, and optimizes the LDP greedily; the solution does not involve any (estimated) future information, which can be implemented in a *distributed* manner.

Remark 7 (Distributed RCNC): To develop a distributed algorithm, we assume that future arrivals from neighbor nodes $\mu_{ji}^{(l)}(t+\tau)$ are estimated by their empirical average $\hat{u}_{ji}^{(l)}$. This leads to an LP formulation that is the same as (35), only to replace (35b) by $\sum_{j\in\delta_i^+} g_{\tau+1}(\boldsymbol{M}_{ij}) \le g_{\tau+1}(\tilde{\boldsymbol{A}}_i^{\text{emp}})$, with $\tilde{\boldsymbol{A}}_i^{\text{emp}} = [\boldsymbol{Q}_i(t), \hat{\boldsymbol{\lambda}}_i + \hat{\boldsymbol{u}}_{\to i}, \cdots, \hat{\boldsymbol{\lambda}}_i + \hat{\boldsymbol{u}}_{\to i}]$. However, numerical results suggest that this formulation does not outperform the simple algorithm using $n = 1$ (see Section VII-B3).

Algorithm 2 RCNC

```
1: for each frame $k \ge 0$ do
2:     for $t = 0 : K-1$ do
3:         Solve the virtual flow $\boldsymbol{\nu}(t)$ from (21);
4:         Solve the actual flow $\boldsymbol{\mu}(t)$ from (35);
5:         Update the request queue $\boldsymbol{R}(t)$ by (28) (using the
           virtual and actual flows derived above);
6:     end for
7:     Update the transmission capacities of the links in the
       virtual network by (36) – (38);
8: end for
```

Remark 8 (Complexity): At every time slot, the centralized algorithm requires solving an LP problem with $nL|\mathcal{V}| + n|\mathcal{E}|$ constraints and $nL|\mathcal{E}|$ variables, and the time complexity is $\mathcal{O}(n^2L^2|\mathcal{E}|^2)$ (at the centralized controller). For the distributed algorithm, the complexity reduces to $\mathcal{O}(n^2L^2|\delta_i^+|^2)$ at node i. Intuitively, the centralized algorithm with $n = L$ can achieve a better performance; however, its complexity is $\mathcal{O}(L^4|\mathcal{E}|^2)$, which can become prohibitive in practice. By selecting $n = 1$, we can obtain the most efficient algorithm (with some performance loss), at a complexity of $\mathcal{O}(L^2|\delta_i^+|^2)$.

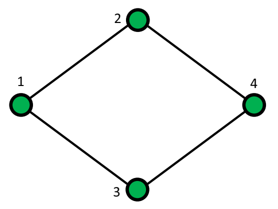
Remark 9: In practice, we can select L as tens of time slots (e.g., $L = 10$), based on following considerations. On one hand, it is on the order of network diameter and sufficient to support packet transmission within the network (note that the network diameter – representing the hop-distance of the longest path – of a hierarchical edge computing network is $\sim \mathcal{O}(\log(|\mathcal{V}|))$. On the other hand, it falls into the regime in which RCNC can run efficiently.

Accordingly, we can select appropriate time slot lengths based on the delay requirement of the supported applications,[5] to achieve a value of L as marked above. For delay-sensitive applications, such as VR (7–20 ms) [33] and real-time gaming (50 ms) [34], a choice of L on tens of time slots would result in time slot length of around 1 ms. A larger slot length can be considered for applications with higher delay budgets, e.g., it can be selected as 15 ms for live streaming (150 ms) [34].

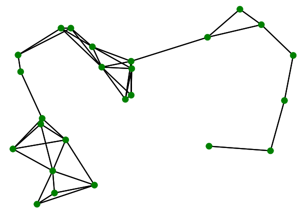
B. Capacity Iteration

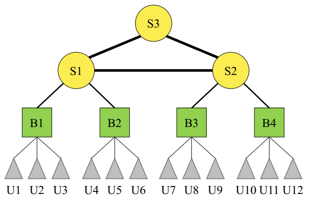
The flow matching technique proposed in the previous section assumes that the virtual flow assignment is achievable. This is not necessarily true since there is no guarantee for the equivalence of the flow spaces, as opposed to the average-constrained case. In fact, when deciding the flow assignment in the virtual network, the network controller prefers to transmit the packets along the low-cost routes, leading to a considerable amount of *bottleneck* links (i.e., for whom the assigned rate equals the transmission capacity), especially in

[5] Note that network slicing allows customizing (virtualized) networks for applications with similar delay requirements.



(a) Illustrative network.

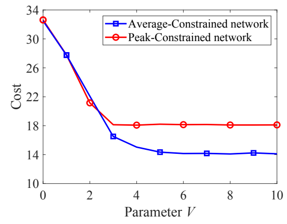

(b) Mesh cloud network.



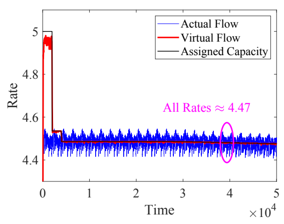
Processing	U	B	S
(Capacity C_i [GHz], cost e_i [per GHz])	(0.5, 5)	(1, 2)	(5, 1)
Transmission	(U, B)	(B, B) (B, S)	(S, S)
(Capacity C_{ij} [Gbps], cost e_{ij} [per Gbps])	(1, 1)	(2, 1)	(5, 1)

(c) Hierarchical cloud network (devices of the same type have the same configuration).

Fig. 5. The studied networks.



(a) Operational cost.



(b) Capacity iteration of link $(1, 2)$.

Fig. 6. The effect of V on the achieved operational cost, and the flow assignment for peak-constrained network under $V = 5$.

the high-congestion regime. However, the achieved rate on the bottleneck link is sensitive to the dynamic input, which is usually strictly lower than the link capacity due to truncation (recall **Example** in Section IV). This motivates us to reduce the assigned virtual flow on these links, which can be realized by decreasing the corresponding link capacity in the virtual network.

In practice, a sign of unsuccessful flow matching is the instability of the request queues (i.e., $R_{ij}^{(l)}(t)$ grows linearly). In addition to the reason of *overestimating* the transmission capacity of the bottleneck link, in a multi-hop network, the request queue of link (i, j) can also exhibit unstable behavior when its source i receives *insufficient* packets from the neighbors compared to the virtual flow assignment. Both factors will be considered when updating the parameters (i.e., link capacities) of the virtual network.

To sum up, the parameters of the virtual network are updated on a larger timescale unit, we referred to as frames. Each frame k consists of K time slots, during which the algorithm developed in the previous subsection is performed in an attempt to stabilize the request queues. At the end of the frame, the increasing rate of the request queue for each link $(i, j) \in \mathcal{E}$ is calculated by

$$r_{ij} = \sum_{l \in \mathcal{L}} r_{ij}^{(l)}, \text{ with } r_{ij}^{(l)} \triangleq \max\left\{0, \frac{1}{K} R_{ij}^{(l)}(K)\right\} \tag{36}$$

and its link capacity is updated by

$$C_{ij}(k+1) = \left[(1-\kappa)\left[C_{ij}(k) - \epsilon_{ij}^{(k)}\right] + \kappa C_{ij}\right]_0^{C_{ij}} \tag{37}$$

in which

$$\epsilon_{ij}^{(k)} = r_{ij} - r_{\to i}\left(\bar{\nu}_{ij}/\bar{\nu}_{i\to}\right) \tag{38}$$

where $\kappa \in (0, 1)$ is a constant, $[z]_0^{C_{ij}} \triangleq \min\{\max\{0, z\}, C_{ij}\}$. The update rule is explained as follows. First, the second term in (38) results from insufficient input, where $r_{\to i}$ is the total amount of insufficient input to node i, and $\bar{\nu}_{ij}/\bar{\nu}_{i\to}$ is the percentage that link (i, j) takes up among all the outgoing interfaces. Second, note that even if the request queue is stabilized, it is possible for $\epsilon_{ij}^{(k)}$ to be positive due to random arrival, leading to too conservative flow assignment; therefore, we add κC_{ij} in (37) to avoid such situation, which explores the possibility to increase the assigned flow rate in the considered link (i, j).

VII. Numerical Experiments

In this section, we carry out numerical experiments to evaluate the performance of the proposed design. We start with an illustrative example (Fig. 5a), in which we explain the related concepts, as well as showing some intermediate results. After that, a more realistic scenario of edge computing network (Fig. 5b) is studied. Both average- and peak-constrained networks are considered, and the term "link capacity" should be interpreted either way. We set $n = L$ and $K = 2 \times 10^3$ as the default setting for the proposed RCNC algorithm.

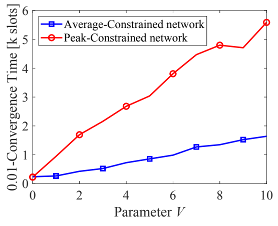
Some key observations are listed as follows: 1) the analytical results (e.g., Proposition 3) are validated; 2) there is a performance gap between average- and peak-constrained problems, which vanishes as we reduce the arrival dynamics; 3) the throughput and cost performance improve with longer admissible lifetimes; 4) the distributed algorithm with $n = 1$ can achieve a comparable performance (especially in low-congestion regimes) with much lower complexity.

A. Illustrative Example

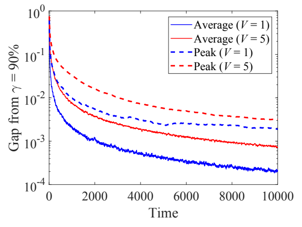
We study the packet routing problem based on the illustrative network in Fig. 5a, which consists of 4 nodes and 4 undirected links. The links exhibit homogeneous transmission capacity of $C_{ij} = 5$ for $\forall (i, j) \in \mathcal{E}$, with different costs given by: $e_{12} = e_{24} = 1$, $e_{13} = e_{34} = 5$.

A single commodity is considered, where the packets of interest emerge at node 1 (the source node), and are desired by node 4 (the destination node). Each packet is of maximum lifetime of $L = 2$ at birth, which implies that it can not be delayed for even one single time slot in order to be effective. The packet arrival process follows a Poisson distribution with parameter $\lambda = 6$, and a reliability level of $\gamma = 90\%$ is demanded by the application.

1) Effects of Parameter V: In this experiment, we study the tradeoff between the convergence time and operational cost controlled by parameter V. We implement and run the control algorithm using various parameters $V \in \{0, 1, \cdots, 10\}$. For

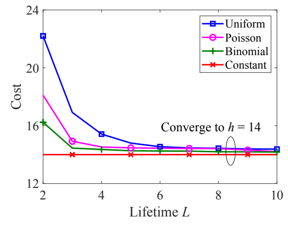


(a) .01-convergence time.

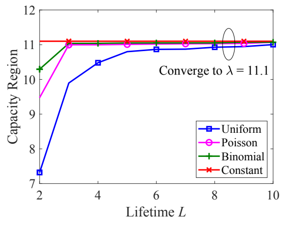


(b) The achieved reliability level.

Fig. 7. The effect of V on the ε-convergence time (with $\varepsilon = .01$), and the achieved reliability level over time under various settings.



(a) Operational cost.



(b) Stability region.

Fig. 8. The effect of maximum lifetime L and the arrival model on the achieved operational cost and stability region.

each V value, we carry out 100 experiments, and observe the system for $T = 1 \times 10^6$ time slots. The results are depicted in Fig. 6 and 7, and we make the following observations.

First, for the average-constrained network, the operational cost reduces with V (Fig. 6a), which is in accordance with the analytical results presented in Proposition 3 and 4 (it also implies that flow matching is achieved). By intuition, we find that in this two-route example, the cheap route $1 \to 2 \to 4$ (with cost $e_{12} + e_{24} = 2$) is preferable when transmitting the packets to benefit the cost performance, which should be exploited to the largest extent; while to satisfy the throughput constraint, some packets still need to be pushed through the expensive route $1 \to 3 \to 4$ (with cost $e_{13} + e_{34} = 10$). More concretely, the flow assignment of the entire network is $x_{12} = x_{24} = \min\{C_{12}, C_{24}\} = 5$ (the corresponding link capacity), $x_{13} = x_{34} = \gamma\lambda - x_{12} = 5.4 - 5 = 0.4$, leading to a cost performance of $h_1^\star = 5 \times 2 + 0.4 \times 10 = 14$. As we can observe in Fig. 6a, the blue curve converges to the value of 14 as V increases, which agrees with the above result.

On the other hand, for the peak-constrained network, the principle to prioritize the cheap route also applies when transmitting the packets. However, due to the dynamics of the arrival process $a(t)$ and the truncation, the amount of packets that can be scheduled for this route is $x_{12}(t) = \min\{a(t), C_{12}\}$ at every time slot. Under the assumption of i.i.d. Poisson arrival, it can be calculated that $\overline{\{x_{12}(t)\}} \approx 4.47$. Therefore, the optimal flow assignment for the peak-constrained network is $x_{12} = x_{24} = 4.47$, and $x_{13} = x_{34} = \gamma\lambda - x_{12} = 0.93$, leading to a cost of $h_2^\star = 4.47 \times 2 + 0.93 \times 10 = 18.24$. The proposed RCNC algorithm finds the flow assignment by trial and exploration, as shown in Fig. 6b (with $V = 5$): we use the link capacity $C_{12} = 5$ as the initial guess for the achievable flow rate, which overestimates the transmission capacity of the link; then its link capacity in the virtual network is reduced (on a frame basis), which gears the corresponding virtual flow; finally, flow matching is achieved when the link capacity $\approx$ achieved rate ≈ 4.47.

Finally, we emphasize that by increasing the value of V, it takes longer to converge to the desired reliability level. Fig. 7b shows the gap between the achieved and the desired reliability level over time, for two particular values $V = 1$ and $V = 5$. We find that (i) all the gap curves reduce over time, implying convergence to the desired value, (ii) it takes longer for the peak-constrained network to converge than the corresponding averaged case (under the same V), which is due to the additional procedure of capacity iteration to find the feasible flow assignment, and (iii) the gap grows with V at a fixed time point for both average- and peak-constrained networks, i.e., a larger V results in slower convergence. In particular, we study the ε-convergence time with $\varepsilon = .01$, and the result is plotted in Fig. 7a. For the average-constrained network, the convergence time grows linearly with V, which supports the analytical result of $\mathcal{O}(V)$ in Proposition 3; similar result is observed from the peak-constrained network.

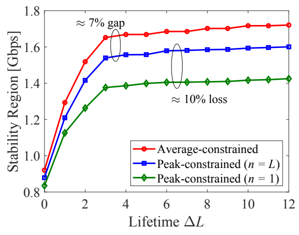
2) Effects of Lifetime and Arrival Model: Next, we study the effects of the maximum lifetime L and the statistics of the arrival process. The considered lifetime L ranges from 2 to 10, and we try different models for the arrival process, including uniform $\mathcal{U}([0, 2\lambda])$, Poisson Pois($\lambda$), binomial $\mathcal{B}(2\lambda, 1/2)$, as well as the constant arrival $a(t) = \lambda$. The four distributions are of the same mean value λ, but decreasing dynamic (the corresponding variances are $\lambda^2/3 > \lambda > \lambda/2 > 0$ if we assume $\lambda > 3$). The reliability level is set as $\gamma = 90\%$, and $V = 10$ is chosen to optimize the operational cost.

Two performance metrics are studied for each settings. One is the achieved operational cost, and the other is the stability region.[6] In the first part of the experiment, we assume $\lambda = 6$ as in the previous experiments.

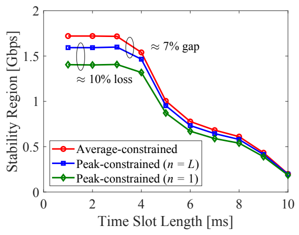
The results for the peak-constrained networks are shown in Fig. 8. As we can observe, for any arrival model, as the maximum lifetime L grows, the operational cost attained by RCNC reduces, while the stability region enlarges. The result agrees with our intuition, that as the initial lifetime grows, the packets are more likely to arrive at the destination while effective, and furthermore, through the cheap route (when possible). In this example, node 1 can withhold the packets in its queuing system at bursting time slots, leaving them for future transmission through $1 \to 2 \to 4$ to optimize the cost. As L increases, the problem reduces to the traditional packet routing problem, where packet lifetime is not relevant, and the attained operational cost and stability region *converge* to the corresponding optimal results.[7] We also find that under constant arrival, the maximum lifetime does not impact the

[6] We recall that the stability region of $\mathscr{P}_0$ is defined w.r.t. the pdf of the arrival process $f_{\boldsymbol{a}}$. In the experiment, as the model of the arrival process is fixed, we only need to specify λ to determine the pdf; in other words, the maximum arrival rate λ can represent the stability region under each model.

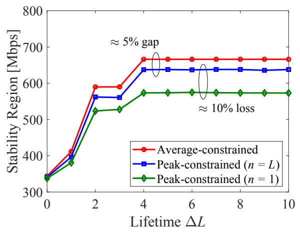
[7] However, we stress that RCNC does not guarantee convergence to the optimal performance in all cases. As we compare the uniform arrival with other models, there is gap in terms of both metrics, which is probably due to the sub-optimality of the flow matching technique in this case.



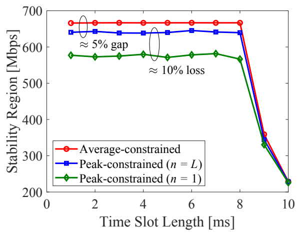
(a) Effect of lifetime (*mesh*).



(b) Effect of time slot length (*mesh*).



(c) Effect of lifetime (*hierarchical*).



(d) Effect of time slot length (*hierarchical*).

Fig. 9. Stability regions of Algorithm 1 and RCNC (for average- and peak-constrained cases, respectively), under different lifetimes and time slot lengths.

performance, which approximately equals the optimal values of the traditional problem.[8]

Finally, we explain the effect of the arrival model. By fixing the maximum lifetime (e.g., $L = 2$), we compare different arrival models, and find that a higher dynamic of the arrival process can increase the operational cost, while shrinking the stability region, both due to the truncation effect of the peak-constrained links. With a given mean rate λ, it is more likely for a high-dynamic arrival process to exceed the transmission capacity, and more packets must be delayed for transmission, which can possibly lead to packet outdatedness, and thus reducing the achievable output rate. This is true for any link in general, and in particular, the links lying in the cheap routes; as a result, a worsened performance of stability region and operational cost can be expected.[9]

B. *Practical Scenarios*

In this section, we demonstrate the performance of the proposed RCNC algorithm in two representative network scenarios:

- *mesh*: a mesh edge computing network including 25 servers that are randomly placed in an $1\,\text{km} \times 1\,\text{km}$ square area, with links established between any pair of servers within a distance of $250\,\text{m}$, as shown in Fig. 5b, which is representative of a generic unstructured scenario.
- *hierarchical*: a hierarchical edge computing network [35] composed of core, edge, access, and user layers, as shown in Fig. 5c, which represents envisioned practical MEC systems.

In the *mesh* network, each link has a transmission capacity of $C_{ij} = 1$ Gbps with a cost of $e_{ij} = 1$ /Gbps, and each server has a processing capacity of $C_i = 2$ GHz with a random cost $e_i \in \{5, 7.5, 10\}$ /GHz, which accounts for the heterogeneity of the computing devices; the parameters of the *hierarchical* network are summarized in the table in Fig. 5c. The default time slot length is 1 ms in both networks.

We adopt the AgI service model used in [19]. The AgI service ϕ is modeled by a sequence of ordered functions, through which incoming packets must be processed to produce consumable results. The service functions can be executed at different network locations, and we assume that each network location can host all the service functions. Each function (say the m-th function of service ϕ) is specified by two parameters: (i) $\xi_\phi^{(m)}$: scaling factor, i.e., the output flow units per input flow unit. (ii) $r_\phi^{(m)}$: workload, i.e., the required computation resource per input flow unit.

In this experiment, we consider two AgI services including 2 functions, with parameters given by (the workload $r_\phi^{(m)}$ is in GHz/Mbps):

$$\text{Service 1}: \ \xi_1^{(1)} = 1,\ \xi_1^{(2)} = 2;\ r_1^{(1)} = \frac{1}{300},\ r_1^{(2)} = \frac{1}{400},$$
$$\text{Service 2}: \ \xi_2^{(1)} = \frac{1}{3},\ \xi_2^{(2)} = \frac{1}{2};\ r_2^{(1)} = \frac{1}{200},\ r_2^{(2)} = \frac{1}{100}.$$

Each service has an i.i.d. Poisson arrival process (for packets with maximum lifetime), with $\lambda_1 = \lambda_2 = \lambda$ Mbps, and requires a reliability level of $\gamma_1 = \gamma_2 = 90\%$.[10] The source-destination pair of each service is selected at random (with the shortest distance between them denoted by σ). The maximum lifetime is then chosen as $L = \sigma + 2 + \Delta L$, where $\sigma + 2$ is the least lifetime for packet delivery ("2" account for two processing slots), and $\Delta L \geq 0$ denotes some allowable relaxation slots.

1) Stability Region: In this section, we present the network stability regions achieved by the proposed algorithms, under different lifetime constraints and time slot lengths. We use a slot length of 1 ms when conducting experiments for lifetime (Fig. 9a and 9c), and fix the delay constraint as $L = 50$ ms when studying the effect of slot length (Fig. 9b and 9d).

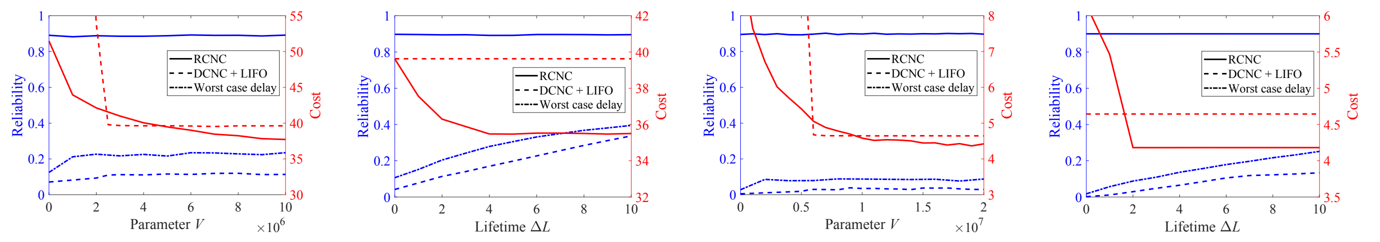
Fig. 9a and 9c depict the effect of lifetime, and we make following observations. First, the stability region enlarges with more available lifetimes, since packets can explore more network locations for additional computation resource; in particular, Fig. 9c saturates at $\Delta L = 4$ because the bottleneck links are constrained by the transmission limits. Second, the gap between the stability regions of average- and peak-constrained networks, is not significant (around 7% for *mesh* and 5% for *hierarchical*).[11] Finally, by comparing $n = 1$ and $n = L$, we find that including more look-ahead slots can benefit the

[8] In addition, for average-constrained networks, the various settings (including maximum lifetime and arrival model) do not impact the performance, either, which is the same as the plotted results for constant arrival.

[9] The results suggest that when 1) the packet lifetime is abundant for transmission, or 2) the arrival process is of low-dynamic, $\mathscr{P}_1$ makes a good approximation for the original problem $\mathscr{P}_0$. However, the gap can be large in some extreme cases, e.g., uniform arrival with $L = 2$ in the experiment.

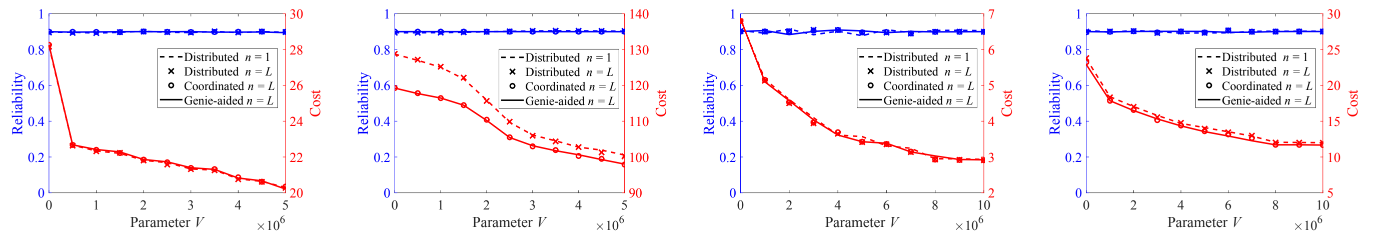
[10] The service chain can expand or compress the size of the input flow, and we calculate the throughput on the basis of the input flow size. See Appendix H for detailed explanation.

[11] We emphasize that RCNC does NOT guarantee to achieve the entire stability region in the peak-constrained case. In other words, the exact stability region in the peak-constrained case lies between the blue and red curves.



(a) V (*mesh*, with $\Delta L = 2$). (b) ΔL (*mesh*, with $V = 1 \times 10^8$). (c) V (*hierarchical*, with $\Delta L = 2$). (d) ΔL (*hierarchical*, with $V = 1 \times 10^8$).

Fig. 10. Throughput and cost achieved by DCNC (with LIFO) [19], worst-case delay [29] and RCNC.



(a) Low-congestion regime (*mesh*). (b) High-congestion regime (*mesh*). (c) Low-congestion regime (*hierarchical*). (d) High-congestion regime (*hierarchical*).

Fig. 11. Performances attained by RCNC under different configurations.

throughput performance (by around 10%), while resulting in a higher complexity of $\mathcal{O}(n^2L^2) = \mathcal{O}(L^4)$.

Next, we tune the time slot length to study its impact on the stability region, with the results shown in Fig. 9b and 9d. As we increase the slot length, the attained throughput in *mesh* starts to degrade when it exceeds 3 ms, since the resulting maximum lifetime L (≤ 12) is not admissible to support the delivery of some services; while the throughput remains unchanged until a slot length of 8 ms in *hierarchical* due to its simpler topology. On the other hand, a larger slot length can accelerate the algorithm: for $n = 1$, as we increase the slot length from 1 to 10 ms, the running time for decision making reduces: 52.8, 13.5, 6.6, 4.1, 2.7, 2.2, 1.8, 1.5, 1.2, 0.94 in *mesh*, and 27.2, 7.3, 3.7, 2.4, 1.6, 1.4, 1.2, 0.99, 0.83, 0.68 in *hierarchical* (in milliseconds).[12]

2) Throughput and Cost: In this experiment, we compare the timely throughput and operational cost attained by RCNC with two benchmark algorithms:

- DCNC [19], which is shown to achieve optimal throughput and (near-optimal) cost performances, combined with last-in-first-out (LIFO) scheduling [36];
- an opportunistic scheduling algorithm [29] that provides worst-case delay guarantees for hop-count-limited transmissions, using the following parameters (notations are in line with [29]): $g_m(x) = x$, $\beta = \nu_m = 1$, $A_m^{\max} = 1.25\lambda$, $D_n^{(m),\max} = \max\{\epsilon, 1_n^{(m)} A_m^{\max} + \mu_n^{\max,in}\}$, with ϵ found by grid search to optimize the timely throughput.

We assume $\lambda = 500$ Mbps for *mesh* and 200 Mbps for *hierarchical*, and Fig. 10 depicts the achieved throughput and cost under different lifetimes ΔL and V values.

First, we focus on the reliability (or timely throughput) attained by the algorithms. The proposed RCNC algorithm can achieve a reliability level of 90% under any ΔL and V that meets the requirement of the services (and the reliability constraint (10b) holds with equality). For DCNC, although it proves to be throughput optimal, the attained *timely* throughput is much lower, which increases with ΔL since more packets are counted as effective under a more relaxed lifetime constraint.[13] The worst-case delay algorithm [29] behaves slightly better than DCNC; however, there is still a considerable gap between the attained reliability and the imposed requirement (always guaranteed by the proposed RCNC algorithm), especially when the deadline constraint is stringent.

Next, we compare the operational cost of RCNC and DCNC.[14] As shown in Fig. 10a and 10c, the operational costs of both algorithms reduce as V increases. When V is small, the cost of DCNC is significantly higher, since it might deliver packets through cyclic routes; while RCNC can reduce the number of extra transmissions due to the deadline constraint. Second, in Fig. 10b and 10d, as we relax the deadline constraint (or increase ΔL), RCNC can achieve better cost performance, since packets can "detour" to cheaper network locations for processing; while packet lifetime is not relevant in DCNC, and its cost performance stays constant (which is near-optimal). Last but not least, we note that the operational cost of RCNC is lower than DCNC, because DCNC delivers *all* the packets to the destination; while RCNC only delivers *effective* packets to meet the reliability requirement.

[12] Results are obtained using MATLAB 2021a running on a 3.2 GHz computer, which leaves room for improvement, e.g., using a commercial solver and/or a faster processor.

[13] As $\Delta L \to \infty$, timely throughput converges to throughput, and DCNC can achieve a reliability level of 100% since it is throughput optimal.

[14] The worst-case delay algorithm [29] is excluded from the comparison, because (i) it does not optimize the operational cost (and the attained costs are around 70 for *mesh* and 30 for *hierarchical*), and (ii) in contrast to the other two algorithms, the parameter V has an essentially different interpretation.

3) Performance of RCNC: Finally, we compare the performance of RCNC with different implementations: using $n = 1$ or L look-ahead slots (see Remark 6), centralized or distributed decision making (see Remark 7). A genie-aided algorithm serves as the benchmark, which can (by assumption) use *accurate* future arrival information to calculate $\boldsymbol{A}_i$ (32) for $n = L$ look-ahead slots. Assume $\Delta L = 2$.

As we can observe from Fig. 11, in the low-congestion regime ($\lambda = 20\%$ of the stability region, Fig. 11a and 11c), the four implementations achieve comparable performance in both *mesh* and *hierarchical* scenarios. However, when the network traffic becomes heavier ($\lambda = 80\%$ of the stability region, Fig. 11b and 11d), the two distributed algorithms (using $n = 1$ or L) achieve sub-optimal cost performance (while still satisfying the reliability constraint); in contrast, the centralized algorithm remains robust, and the difference of its cost performance compared to the genie-aided algorithm is negligible. The result shows the importance of coordinated decision making among the nodes in the high-congestion regime to preserve the optimality of the solution; yet, it also motivates the use of the simplified algorithm ($n = 1$) in practical systems, especially for networks with simpler topologies (such as *hierarchical*), which can achieve sub-optimal performance with greatly reduced computational complexity.

VIII. Extensions

In this section, we briefly discuss flexible extensions to the proposed approach in order to handle scenarios of practical relevance.

A. Mixed Deadline-Constrained and Unconstrained Users

It is flexible to combine the proposed approach with existing queuing techniques [23] in order to treat hybrid scenarios that include both deadline-constrained and unconstrained users [26]. To be specific, we can establish a queuing system that is a hybrid of the proposed lifetime queues for the constrained users, and standard queues (i.e., without lifetime structure) for unconstrained users. As shown in Appendix G, the decisions for the two groups of users are *loosely* coupled, where unconstrained users follow the *max-weight* rule [23] for scheduling, and interact with constrained users via *one* additional variable for each link and look-ahead slot that represents the entire group, regardless of the number of unconstrained users.

B. Time-Varying Slot Length

While the technique developed in this paper assumes a fixed slot length, it is also possible to adopt a varying slot length via "lifetime mapping". In principle, when the slot length changes, we can construct a new queuing system, assign packets to queues of corresponding lifetimes, and map the decisions produced by the original policy to suite the new slot length. For example, if the slot length changes from 1 ms to 2 ms, we can add up the decisions (i.e., transmitted flows) for lifetime $2k - 1$ and $2k$ packets to obtain the decision for lifetime k packets based on the new slot length. The design of the mapping functions and associated performance loss analysis are topics worth further investigation.

IX. Conclusions

In this paper, we investigated the delay-constrained least cost dynamic network control problem. We established a new queuing system to keep track of data packets' lifetime, based on which we formalized the problem $\mathscr{P}_0$. To find an efficient approximate solution to this challenging problem, we first derived a relaxed problem with average capacity constraints $\mathscr{P}_1$ and designed a fully distributed, near-optimal solution that matches the LDP assigned flow on an equivalent virtual network $\mathscr{P}_2$. The methodology was then extended to solve $\mathscr{P}_0$, where we proposed a two-way optimization approach in order to use the assigned flow in $\mathscr{P}_2$ to guide the flow solution to $\mathscr{P}_0$. Extensive numerical results were presented to validate the analytical results, illustrate the performance gain, and guide the system configuration.

Acknowledgment

The authors would like to thank Prof. Michael J. Neely for helpful discussions.

References

[1] Y. Cai, J. Llorca, A. M. Tulino, and A. F. Molisch, "Optimal cloud network control with strict latency constraints," in *Proc. IEEE Int. Conf. Commun.*, Montreal, Canada, Jun. 2021, pp. 1–6.

[2] Y. Cai, J. Llorca, A. M. Tulino, and A. F. Molisch, "Compute- and data-intensive networks: The key to the Metaverse," to be published in *2022 1st International Conference on 6G Networking (6GNet)*. [Online]. Available: https://arxiv.org/abs/2204.02001, Apr. 2022.

[3] H. Feng, J. Llorca, A. M. Tulino, and A. F. Molisch, "On the delivery of augmented information services over wireless computing networks," in *Proc. IEEE Int. Conf. Commun.*, Paris, France, May 2017, pp. 1–7.

[4] M. Weldon, The future X network: A Bell Labs perspective. Boca Raton, FL, USA: CRC Press, 2016.

[5] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, Mar. 2017.

[6] Y. Cai, J. Llorca, A. M. Tulino, and A. F. Molisch, "Mobile edge computing network control: Tradeoff between delay and cost," in *Proc. IEEE Global. Telecomm. Conf.*, Taipei, Taiwan, Dec. 2020, pp. 1–6.

[7] S. Lashgari and A. S. Avestimehr, "Timely throughput of heterogeneous wireless networks: Fundamental limits and algorithms," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 8414–8433, Sep. 2013.

[8] K. Chen and L. Huang, "Timely-throughput optimal scheduling with prediction," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2457–2470, Sep. 2018.

[9] J. Sun, L. Wang, Z. Jiang, S. Zhou, and Z. Niu, "Age-optimal scheduling for heterogeneous traffic with timely throughput constraints," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 5, pp. 1485–1498, May 2021.

[10] M. Barcelo, J. Llorca, A. M. Tulino, and N. Raman, "The cloud service distribution problem in distributed cloud networks," in *Proc. IEEE Int. Conf. Commun.*, London, UK, May 2015, pp. 344–350.

[11] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions in NFV," in *Int. Conf. on Netw. Service Manag. (CNSM)*, Barcelona, Spain, Nov. 2015, pp. 50–56.

[12] B. Addis, D. Belabed, M. Bouet, and S. Secci, "Virtual network functions placement and routing optimization," in *IEEE Int. Conf. Cloud Netw. (CLOUDNET)*, Niagara Falls, Canada, Oct. 2015, pp. 171–177.

[13] M. Barcelo, A. Correa, J. Llorca, A. M. Tulino, J. L. Vicario, and A. Morell, "IoT-cloud service optimization in next generation smart environments," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 4077–4090, Oct. 2016.

[14] K. Poularakis, J. Llorca, A. M. Tulino, and L. Tassiulas, "Approximation algorithms for data-intensive service chain embedding," in *Mobihoc '20*, Virtual Event, USA, Oct. 2020, pp. 131–140.

[15] M. Huang, W. Liang, Y. Ma, and S. Guo, "Maximizing throughput of delay-sensitive NFV-enabled request admissions via virtualized network function placement," *IEEE Trans. Cloud Comput.*, vol. 9, no. 4, pp. 1535–1548, Oct.-Dec. 2021.

[16] Z. Xu, Z. Zhang, W. Liang, Q. Xia, O. Rana, and G. Wu, "QoS-aware VNF placement and service chaining for IoT applications in multi-tier mobile edge networks," *ACM Trans. Sen. Netw.*, vol. 16, no. 3, pp. 1–27, Aug. 2020.

[17] Y. Yue, B. Cheng, M. Wang *et al.*, "Throughput optimization and delay guarantee VNF placement for mapping SFC requests in NFV-enabled networks," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 4, pp. 4247–4262, Dec. 2021.

[18] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, "Service placement and request routing in MEC networks with storage, computation, and communication constraints," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1047–1060, Jun. 2020.

[19] H. Feng, J. Llorca, A. M. Tulino, and A. F. Molisch, "Optimal dynamic cloud network control," *IEEE/ACM Trans. Netw.*, vol. 26, no. 5, pp. 2118–2131, Oct. 2018.

[20] Y. Cai, J. Llorca, A. M. Tulino, and A. F. Molisch, "Optimal multicast service chain control: Packet processing, routing, and duplication," in *Proc. IEEE Int. Conf. Commun.*, Montreal, Canada, Jun. 2021, pp. 1–7.

[21] J. Zhang, A. Sinha, J. Llorca, A. M. Tulino, and E. Modiano, "Optimal control of distributed computing networks with mixed-cast traffic flows," *IEEE/ACM Trans. Netw.*, vol. 29, no. 4, pp. 1760–1773, Aug. 2021.

[22] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Autom. Control*, vol. 37, no. 12, pp. 1936–1948, Dec. 1992.

[23] M. J. Neely, *Stochastic network optimization with application to communication and queueing systems*. San Rafael, CA, USA: Morgan & Claypool, 2010.

[24] L. Bui, R. Srikant, and A. Stolyar, "Novel architectures and algorithms for delay reduction in back-pressure scheduling and routing," in *Proc. IEEE INFOCOM*, Rio de Janeiro, Brazil, Apr. 2009, pp. 2936–2940.

[25] L. Ying, S. Shakkottai, A. Reddy, and S. Liu, "On combining shortest-path and back-pressure routing over multihop wireless networks," *IEEE/ACM Trans. Netw.*, vol. 19, no. 3, pp. 841–854, Jun. 2011.

[26] M. J. Neely and S. Supittayapornpong, "Dynamic Markov decision policies for delay constrained wireless scheduling," *IEEE Trans. Autom. Control*, vol. 58, no. 8, pp. 1948–1961, Aug. 2013.

[27] A. Sinha and E. Modiano, "Optimal control for generalized network flow problems," *IEEE/ACM Trans. Netw.*, vol. 26, no. 1, pp. 506–519, Feb. 2018.

[28] M. R. Garey and D. S. Johnson, *A guide to the theory of NP-completeness*. New York: WH Freemann, 1979.

[29] M. J. Neely, "Opportunistic scheduling with worst case delay guarantees in single and multi-hop networks," in *Proc. IEEE INFOCOM*, Shanghai, China, Apr. 2011, pp. 1728–1736.

[30] R. Singh and P. R. Kumar, "Throughput optimal decentralized scheduling of multihop networks with end-to-end deadline constraints: Unreliable links," *IEEE Trans. Autom. Control*, vol. 64, no. 1, pp. 127–142, Oct. 2018.

[31] R. Singh and P. R. Kumar, "Adaptive CSMA for decentralized scheduling of multi-hop networks with end-to-end deadline constraints," *IEEE/ACM Trans. Netw.*, vol. 29, no. 3, pp. 1224–1237, Jun. 2021.

[32] E. Altman, *Constrained Markov decision processes*. Boca Raton, FL, USA: CRC Press, 1999.

[33] E. Cuervo, K. Chintalapudi, and M. Kotaru, "Creating the perfect illusion: What will it take to create life-like virtual reality headsets?" in *HotMobile '18*, Tempe, AZ, USA, Feb. 2018, pp. 7–12.

[34] C. Anton-Haro and M. Dohler, Machine-to-machine (M2M) communications: architecture, performance and applications. Boca Raton, FL, USA: Elsevier, 2014.

[35] K. Kamran, E. Yeh, and Q. Ma, "DECO: Joint computation scheduling, caching, and communication in data-intensive computing networks," *IEEE/ACM Trans. Netw.*, vol. 1, no. 99, pp. 1–15, 2021.

[36] L. Huang, S. Moeller, M. J. Neely, and B. Krishnamachari, "LIFO-backpressure achieves near-optimal utility-delay tradeoff," *IEEE/ACM Trans. Netw.*, vol. 21, no. 3, pp. 831–844, Sep. 2013.

Yang Cai (Student Member, IEEE) received B.E. and M.S. degrees from the Department of Electronic Engineering, Tsinghua University, in 2015 and 2018, respectively. Now he is a Ph.D. candidate in the Department of Electrical Engineering at University of Southern California, with the Annenberg Graduate Fellowship award from 2018 to 2022. He is currently working on modeling, analysis, evaluation, and control of next-generation computing networks and services.

Jaime Llorca (Member, IEEE) is a Technology Consultant and Research Professor with the NYU Tandon School of Engineering, NY. He previously held a Senior Research Scientist position with the Network Algorithms Group at Nokia Bell Labs, NJ, a Research Scientist position with the End-to-End Networking Group at Alcatel-Lucent Bell Labs, NJ, and a post-doctoral position with the Center for Networking of Infrastructure Sensors, MD. He received M.S. and Ph.D. degrees in Electrical and Computer Engineering from the University of Maryland, College Park, MD. His research interests are in the field of network algorithms, optimization, machine learning, and distributed control, with applications to next generation communication networks, distributed/edge cloud, and content distribution. He has made fundamental contributions to the mathematics of content delivery and distributed cloud networks, including pioneering cooperative caching, network coding, and cloud network control algorithms. He has authored more than 100 peer-reviewed publications and 20 patents. He currently serves as Associate Editor for the IEEE/ACM Transactions on Networking. He is a recipient of the 2007 IEEE ISSNIP Best Paper Award, the 2016 IEEE ICC Best Paper Award, and the 2015 Jimmy H.C. Lin Award for Innovation.

Antonia M. Tulino (Fellow, IEEE) is a full professor at Università degli Studi di Napoli Federico II. She held research positions with the Center for Wireless Communications in Oulu, Princeton University, a Università degli Studi del Sannio, Italy and Bell Labs, NJ. Since 2019, she is Research Professor at Dep. of Electrical and Computer Engineering NYU Tandon School of Engineering, also director of the 5G Academy jointly organized by the Universit degli Studi di Napoli, Federico II and several leading company in digital transformation. Her research interests lay in the area of communication networks approached with the complementary tools provided by signal processing, information theory, and random matrix theory. From 2011 to 2013, she has been a member of the Editorial Board of the IEEE Transactions on Information Theory and in 2013, she was elevated to IEEE Fellow. From 2019 to 2021, she has been chair of the Information Theory society Fellows Committee.

She has received several paper awards, including the 2009 Stephen O. Rice Prize in the Field of Communications Theory. She was recipient of the UC3M-Santander Chair of Excellence from 2018 to 2019 and selected by the National Academy of Engineering for the Frontiers of Engineering program in 2013.

Andreas F. Molisch (Fellow, IEEE) received his degrees (Dipl.Ing. 1990, PhD 1994, Habilitation 1999) from the Technical University Vienna, Austria. He spent the next 10 years in industry, at FTW, AT&T (Bell) Laboratories, and Mitsubishi Electric Research Labs (where he rose to Chief Wireless Standards Architect). In 2009 he joined the University of Southern California (USC) in Los Angeles, CA, as Professor, and founded the Wireless Devices and Systems (WiDeS) group. In 2017, he was appointed to the Solomon Golomb Andrew and Erna Viterbi Chair.

His research interests revolve around wireless propagation channels, wireless systems design, and their interaction. Recently, his main interests have been wireless channel measurement and modeling for 5G and beyond 5G systems, joint communication-caching-computation, hybrid beamforming, UWB/TOA based localization, and novel modulation/multiple access methods. Overall, he has published 5 books (among them the textbook Wireless Communications, third edition in 2022), 21 book chapters, 280 journal papers, and 370 conference papers. He is also the inventor of 70 granted (and more than 10 pending) patents, and co-author of some 70 standards contributions. His work has been cited more than 59,000 times, his h-index is > 100, and he is a Clarivate Highly Cited Researcher.

Dr. Molisch has been an Editor of a number of journals and special issues, General Chair, Technical Program Committee Chair, or Symposium Chair of multiple international conferences, as well as Chairperson of various international standardization groups. He is a Fellow of the National Academy of Inventors, Fellow of the AAAS, Fellow of the IEEE, Fellow of the IET, an IEEE Distinguished Lecturer, and a member of the Austrian Academy of Sciences. He has received numerous awards, among them the IET Achievement Medal, the Technical Achievement Awards of IEEE Vehicular Technology Society (Evans Avant-Garde Award) and the IEEE Communications Society (Edwin Howard Armstrong Award), and the Technical Field Award of the IEEE for Communications, the Eric Sumner Award.