Optimal Cloud Network Control with Strict Latency Constraints

Yang Cai*, Jaime Llorca†, Antonia M. Tulino†‡, Andreas F. Molisch*
*University of Southern California, CA 90089, USA. Email: {yangcai, molisch}@usc.edu
†New York University, NY 10012, USA. Email: {jllorca, atulino}@nyu.edu
‡Universityà degli Studi di Napoli Federico II, Naples 80138, Italy. Email: antoniamaria.tulino@unina.it

***Abstract*—The timely delivery of resource-intensive and latency-sensitive services (e.g., industrial automation, augmented reality) over distributed computing networks (e.g., mobile edge computing) is drawing increasing attention. Motivated by the insufficiency of *average* delay performance guarantees provided by existing studies, we focus on the critical goal of delivering next generation real-time services ahead of corresponding deadlines *on a per-packet basis*, while minimizing overall cloud network resource cost. We introduce a novel queuing system that is able to track *data packets' lifetime* and formalize the optimal cloud network control problem with strict deadline constraints. After illustrating the main challenges in delivering packets to their destinations before getting dropped due to lifetime expiry, we construct an equivalent formulation, where relaxed flow conservation allows leveraging Lyapunov optimization to derive a provably near-optimal fully distributed algorithm for the original problem. Numerical results validate the theoretical analysis and show the superior performance of the proposed control policy compared with state-of-the-art cloud network control.**

I. Introduction

The past decade has seen a proliferation of resource- and interaction-intensive applications, such as real-time computer vision, autonomous transportation, machine control in Industry 4.0, multiuser video conferencing, and augmented/virtual reality [1], which we collectively refer to as augmented information (AgI) services. 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 of source data streams. In contrast, user equipments (UEs) are evolving towards increasingly small, portable devices (and inevitably, with constrained power and computing capabilities), pushing the need to offload many computing tasks to the cloud, especially those running advanced architectures such as fog and mobile edge computing (MEC), which deploy computation resources closer to the end users in order to strike a better balance between access delay and resource efficiency.

Delay and cost are thus two essential metrics when evaluating the performance of AgI service delivery. From the consumers' perspective, excessive end-to-end delays can significantly impact quality of experience (QoE), especially for delay-sensitive AgI applications where packets must be delivered by a strict deadline in order to be effective. 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 [2]. In contrast, network operators care about the overall resource (e.g., computation, communication) consumption (and associated cost) needed to support the dynamic service requests raised by end users, which are dictated by the decision of route selection, function execution, and the corresponding resource allocation [3].

Previous studies have shown that the cloud network control problem, involving packet routing and processing decisions over a distributed computing network, can be connected to the *packet routing* problem in traditional communication networks via a properly constructed *cloud-augmented* or *layered-graph* formulation [4], [5]. For packet routing, many dynamic control policies have been developed aimed at maximizing network throughput, including the celebrated back-pressure (BP) algorithm [6] and its extension, the Lyapunov drift-plus-penalty (LDP) control approach [7] that, in addition, optimizes network resource cost (e.g., energy expenditure). While having the remarkable advantage of achieving throughput optimal 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 unnecessary long and sometimes even cyclic paths. In an attempt to address this problem, [8] proposed a combination BP and shortest-path routing; while [9] designed a centralized source routing approach, referred to as UMW, which reduces the average delay by dynamically selecting an acyclic route for each incoming packet, however requiring global network information.

Going beyond average delay and analyzing per-packet delay performance is a much more challenging problem with much fewer known results. In [10], a variant of the BP algorithm is developed that provides worst-case delay guarantees by allowing packet drops, leading to a tradeoff between delay and achievable throughput; however, the relationship is not tight enough for practical purposes. In [11], the authors formulate the problem of timely throughput maximization as a constrained Markov decision process (CMDP), and address it by solving the *single-packet routing* problem separately for each packet; yet, its computational complexity is prohibitive for practical implementation. A more comprehensive literature review is presented in [12].

In this paper, we investigate the problem of multi-hop

This work was supported in part by the National Science Foundation (NSF) under CNS-1816699.

distributed cloud network control with the goal of delivering multiple AgI services with strict deadline constraints on a per-packet basis, while minimizing overall resource cost. Our contributions can be summarized as follows:

1) We characterize the *delay-constrained network capacity region* leveraging a novel *lifetime-driven* generalized flow conservation law.
2) We develop a fully distributed control algorithm for the delivery of delay-sensitive services, shown to achieve *timely-throughput* optimality while minimizing overall resource cost.
3) We present numerical results illustrating the delay-constrained network capacity region and the superior delay-cost performance tradeoff of the proposed policy.

II. System Model

Due to space limitations, in this paper we describe the proposed approach for the *packet routing* problem and refer the reader to the longer version in [12] for its cloud network control generalization.[1] Nonetheless, we do use the general cloud network control model for the numerical results in Section V.

Consider the packet routing network modeled by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with $\mathcal{V}$ and $\mathcal{E}$ denoting the node and edge set, respectively. The nodes can transmit packets via the links $(i, j) \in \mathcal{E}$ between them, and we denote by δ_i^- and δ_i^+ the incoming and outgoing sets of node i, respectively.

Time is slotted, and we quantify the available communication resource and the associated cost as [4]

- C_{ij}: the average transmission capacity,[2] i.e., the maximum average transmission rate of link (i, j);
- e_{ij}: the transmission cost, i.e., the cost of transmitting one unit size of data, on link (i, j).

The problem of interest is to deliver the packets to their destinations ahead of corresponding deadlines, to which *data packets' lifetime* is closely related, defined as the number of time slots before the information contained in the packet becomes useless. A packet is called *effective* if its remaining lifetime $l > 0$, and *outdated* otherwise. We assume that only the delivery of effective packets counts (i.e., outdatedness is as bad as packet loss); the associated metric, *timely throughput*, i.e., the rate of effective packets delivery, is employed to characterize the capability of the communications network.

A. Request Model

For ease of exposition, we formulate the problem for a single destination-based commodity described by a destination node (or user) $d \in \mathcal{V}$ that requests of a given application, with the straightforward extension to multiple commodities given in [4]. We assume that the input packets of the given commodity can originate at any node $i \in \mathcal{V} \setminus \{d\}$ (restriction of a specific source node set is straightforward) and that the source of each packet is aware of its lifetime $l \in \mathcal{L} \triangleq \{1, \cdots, L\}$ at birth, where L denotes the maximal lifetime. We denote by $a_i^{(l)}(t)$ the number of lifetime-l packets arriving at node i on time slot t, which is assumed to be i.i.d. over time (with an upper bound of $A_{\max}$); besides, the mean arrival rate is defined as $\lambda_i^{(l)} \triangleq \mathbb{E}\{a_i^{(l)}(t)\}$, and the collection $\boldsymbol{\lambda} = \{\lambda_i^{(l)} : \forall\, i \in \mathcal{V}, l \in \mathcal{L}\}$ is called the arrival vector.

B. Queuing System

We construct a queuing system $\boldsymbol{Q}(t)$ that includes distinct queues for packets of different *current lifetimes* $l \in \mathcal{L}$, and denote by $Q_i^{(l)}(t)$ the queue backlog (i.e., number of packets in the queue) of lifetime l packets at node i on time slot t.

Each time slot is divided into two phases. In the *decision* phase, the nodes make and accomplish the transmission decisions (which packets are sent out, to which neighboring node); in the *receiving* phase, the incoming packets, including those from neighboring nodes and the exogenous packets, are collected and loaded into the queuing system. Let $x_{ij}^{(l)}(t)$ be the number of lifetime l packets that are sent from node i to j in time slot t; we refer to it as *flow variable*.

In general, the queuing dynamics are given by[3]

$$Q_i^{(l)}(t+1) = Q_i^{(l+1)}(t) - x_{i\rightarrow}^{(l+1)}(t) + x_{\rightarrow i}^{(l+1)}(t) + a_i^{(l)}(t) \quad (1)$$

where $x_{\rightarrow i}^{(l)}(t) = \sum_{j\in\delta_i^-} x_{ji}^{(l)}(t)$ and $x_{i\rightarrow}^{(l)}(t) = \sum_{j\in\delta_i^+} x_{ij}^{(l)}(t)$ denote the total incoming and outgoing packets of node i.

In addition, we make the following assumptions: 1) outdated packets (not contribute to timely throughput) are dropped, i.e.,

$$Q_i^{(0)}(t) = 0, \quad \forall\, i \in \mathcal{V}, \quad (2)$$

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

$$Q_d^{(l)}(t) = 0, \quad \forall\, l \in \mathcal{L}. \quad (3)$$

C. Admissible Policy Space

The considered control policies make decisions on packet routing and scheduling in each time slot,[4] which are specified by the flow variables $\boldsymbol{x}(t) = \{x_{ij}^{(l)}(t) : \forall\,(i,j) \in \mathcal{E}, l \in \mathcal{L}\}$. We restrict to the space of *admissible* control policies with the decided flow variables satisfying the following conditions:

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

$$x_{ij}^{(l)}(t) \geq 0 \text{ for } \forall\,(i,j) \in \mathcal{E}, \text{ or } \boldsymbol{x}(t) \succeq 0; \quad (4)$$

2) average capacity constraint, i.e.,

$$\overline{\{\mathbb{E}\{x_{ij}(t)\}\}} \leq C_{ij}, \ \forall (i,j) \in \mathcal{E} \quad (5)$$

where $x_{ij}(t) \triangleq \sum_{l\in\mathcal{L}} x_{ij}^{(l)}(t)$, and $\overline{\{z(t)\}}$ denotes the expected long-term average operation, defined as $\overline{\{z(t)\}} = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} z(t)$;

[1] The generalization is based on the layered-graph technique [5].

[2] As an extension to this work, the problem formulated under a more realistic scenario with *peak* link capacity constraint is also studied in [12]. The solution closely follows the principle of the methodology presented in this paper.

[3] The lifetime of the exogenous packets $a_i^{(l)}(t)$ are counted starting from the beginning of next time slot, i.e., the point when they are available for use; the lifetime l of the transmitted packets $x_{ij}^{(l)}$ is the current lifetime, which are available at node j in next time slot with a lifetime of $l - 1$.

[4] Note that a packet can be discarded in network due to lifetime expiry. In this sense, the designed policy also deals with admission control aspect.

3) availability constraint, which requires the number of scheduled outgoing packets not to exceed those in the current queuing system, i.e.,

$$x_{i\rightarrow}^{(l)}(t) \leq Q_i^{(l)}(t),\ \forall\, i \in \mathcal{V},\, l \in \mathcal{L}; \tag{6}$$

4) reliability constraint, i.e.,

$$\overline{\{\mathbb{E}\{x_{\rightarrow d}(t)\}\}} \triangleq \sum\nolimits_{l\in\mathcal{L}} \overline{\left\{\mathbb{E}\left\{x_{\rightarrow d}^{(l)}(t)\right\}\right\}} \geq \gamma\|\boldsymbol{\lambda}\|_1 \tag{7}$$

where γ is named the *reliability level*, and $\|\boldsymbol{\lambda}\|_1$ is the total arrival rate of the application.

The reliability constraint quantifies the extent to which the considered application can tolerate packet loss. It implies that a percentage of up to $(1-\gamma)$ of the incoming packets can be dropped without causing a significant performance loss.

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

$$h(t) = h(\boldsymbol{x}(t)) = \sum\nolimits_{(i,j)\in\mathcal{E}} e_{ij}x_{ij}(t) = \langle \boldsymbol{e}, \boldsymbol{x}(t)\rangle \tag{8}$$

where $\langle\cdot,\cdot\rangle$ denotes the inner product of the two vectors.

Remark 1: In the existing literature of stochastic network optimization, the *assigned* flow has gained widespread use (e.g., [4], [6], [7], [10]), which is different from the actual flow in that it does not take the availability constraint (6) into account, and thus the decision space in each time slot does not depend on the current queuing status. *Dummy* packets are created when there are not sufficient packets in the queue to support the decision. The formulation is not suitable for the considered problem, because the reliability constraint (7) is imposed on *actual* packets received by the destination node d; while in the previous formulation, the flow variables $x_{jd}^{(l)}(t)$ $(j \in \delta_d^-)$ can include dummy packets.

D. Problem Formulation

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

$$\mathscr{P}_1:\ \min_{\boldsymbol{x}(t)\succeq 0}\ \overline{\{\mathbb{E}\{h(\boldsymbol{x}(t))\}\}} \tag{9a}$$

$$\text{s.t.}\quad \overline{\{\mathbb{E}\{x_{\rightarrow d}(t)\}\}} \geq \gamma\|\boldsymbol{\lambda}\|_1 \tag{9b}$$

$$\overline{\{\mathbb{E}\{x_{ij}(t)\}\}} \leq C_{ij},\ \forall (i,j)\in\mathcal{E} \tag{9c}$$

$$x_{i\rightarrow}^{(l)}(t) \leq Q_i^{(l)}(t),\ \forall\, i\in\mathcal{V}, l\in\mathcal{L} \tag{9d}$$

$$\boldsymbol{Q}(t) \text{ evolves by (1) – (3).} \tag{9e}$$

We emphasize that the above problem cannot be addressed by the *LDP* approach, because (i) the queuing system (9e) allows packet drops by (2), and (ii) we employ the actual flow as the decision variable, i.e., constraint (9d), which are different from the standard formulation.

In general, (9) can be interpreted as a CMDP problem [11]. However, we emphasize that the related state vector $\boldsymbol{Q}(t)$ and action vector $\boldsymbol{x}(t)$ are *network-wide*, which leads to a dramatically increasing state-action space, and the complexity of the standard solution [13] is prohibitive for practical implementation.

III. The Equivalent Problem

In this section, we present a new problem $\mathscr{P}_2$, which is referred to as the *virtual network*. We prove the equivalence of the two problems in terms of flow space and capacity region, and use $\mathscr{P}_2$ as a stepping-stone to find the solution to $\mathscr{P}_1$.

A. The Equivalent Problem

The new problem is cast as

$$\mathscr{P}_2:\ \min_{\boldsymbol{x}(t)\succeq 0}\ \overline{\{\mathbb{E}\{h(\boldsymbol{x}(t))\}\}} \tag{10a}$$

$$\text{s.t.}\quad \overline{\{\mathbb{E}\{x_{\rightarrow d}(t)\}\}} \geq \gamma\|\boldsymbol{\lambda}\|_1 \tag{10b}$$

$$x_{ij}(t) \leq C_{ij} \tag{10c}$$

$$\overline{\left\{\mathbb{E}\left\{x_{i\rightarrow}^{(\geq l)}(t)\right\}\right\}} \leq \overline{\left\{\mathbb{E}\left\{x_{\rightarrow i}^{(\geq l+1)}(t)\right\}\right\}} + \lambda_i^{(\geq l)} \tag{10d}$$

where the superscript $(\geq l)$ indicates that the term includes all the lifetimes ℓ satisfying $\ell \geq l$, e.g., $x_{ij}^{(\geq l)}(t) = \sum_{\ell=l}^{L} x_{ij}^{(\ell)}(t)$.

1) Virtual Queues: The crucial difference in deriving $\mathscr{P}_2$ is to eliminate (i) the unconventional queuing system (9e) and (ii) (9d) that makes $\boldsymbol{x}(t)$ dependent on $\boldsymbol{Q}(t)$, i.e., the two factors prohibiting direct application of the LDP approach. Instead, we introduce the relaxed *causality* constraint (10d) (see Proposition 1) to state the fact that the lifetime of the packets must decrease as they traverse any node i.

Although there is no explicit queuing system involved in $\mathscr{P}_2$, it consists of long-term average objective and constraints, which can be addressed by the LDP approach via the use of virtual queues [7] (we denote the solution by $\boldsymbol{\nu}(t)$). More concretely, ensuring constraints (10b) and (10d) is equivalent to stabilize the following virtual queues:

$$U_d(t+1) = \max\left\{0,\, U_d(t) + \gamma A(t) - \nu_{\rightarrow d}(t)\right\} \tag{11a}$$

$$U_i^{(l)}(t+1) = \max\left\{0,\, U_i^{(l)}(t) + \nu_{i\rightarrow}^{(\geq l)}(t) - \nu_{\rightarrow i}^{(\geq l+1)}(t) - a_i^{(\geq l)}(t)\right\} \quad (i\in\mathcal{V}\setminus\{d\}, l\in\mathcal{L}) \tag{11b}$$

where $A(t) = \sum_{i\in\mathcal{V}}\sum_{l\in\mathcal{L}} a_i^{(l)}(t)$ is the total amount of packets arriving at the network in time slot t.[5] We refer to (11a) and (11b) as the virtual queues at node d and i, respectively.

2) Physical Interpretations: $\mathscr{P}_2$ describes a virtual network modeling each node as a *data-reservoir*, which has access to abundant (virtual) packets of any lifetime. As neighboring nodes request packets from node i, it supplies the needs by using the virtual packets from the reservoir in advance, which are compensated when the node receives incoming packets of the same lifetime. The virtual queues can be roughly explained as the *accumulated data deficits* (outgoing flow minus incoming flow) of the corresponding data-reservoirs; specially, in (11a), the destination reservoir *sends out* $\gamma A(t)$ packets to the end user (as is required by the reliability constraint), while *receiving* $\nu_{\rightarrow d}(t)$ in return. When (10d) is satisfied, node i no longer embezzles the virtual packets from

[5]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.

its reservoir; and if it is true for all nodes, the data streams in the network include only actual packets. The resulting flow assignment (defined in next subsection) can instruct the packets to find their paths in the actual network.

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

Similar to Section II-C, for a given pair of $(\boldsymbol{\lambda}, \gamma)$, we define a policy p to be *admissible* for $\mathscr{P}_2$ if it satisfies (10b) – (10d). In addition, suppose an admissible policy $p \in \mathcal{A}_n$ $(n = 1, 2)$ makes decisions $\boldsymbol{x}_p(t) = \big\{x_{ij}^{(l)}(t) : (i,j) \in \mathcal{E},\, l \in \mathcal{L},\, t \geq 0\big\}$, then the associated *flow assignment* is defined as $\boldsymbol{x}_p = \overline{\{\mathbb{E}\{\boldsymbol{x}_p(t)\}\}}$, which collects the transmission rates of all links.

Definition 1: For given $(\boldsymbol{\lambda}, \gamma)$, the *admissible policy space* $\mathcal{A}_n$ is defined as the collection of all admissible control policies for problem $\mathscr{P}_n$ $(n = 1, 2)$.

Definition 2: The *network capacity region* Λ_n is defined as the set of $(\boldsymbol{\lambda}, \gamma)$ pairs, under which the admissible policy space $\mathcal{A}_n$ is non-empty $(n = 1, 2)$.

Definition 3: For given $(\boldsymbol{\lambda}, \gamma) \in \Lambda_n$, the *flow space* is defined as the set of all flow assignments that can be achieved by the admissible policies, i.e., $\Gamma_n = \big\{\boldsymbol{x}_p : p \in \mathcal{A}_n\big\}$ $(n = 1, 2)$.

Next, we present the relationships between the two problems, in terms of the above quantities.

Proposition 1: (9d) implies (10d), and (10c) implies (9c).

Proposition 2: For a given network, the capacity regions of the two problems are identical, i.e., $\Lambda_1 = \Lambda_2$.

A pair $(\boldsymbol{\lambda}, \gamma)$ is within the capacity region Λ_n $(n = 1, 2)$ 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 $\forall\, i \in \mathcal{V}$, $(i,j) \in \mathcal{E}$, $l \in \mathcal{L}$,

$$x_{\to d} \geq \gamma \|\boldsymbol{\lambda}\|_1 \tag{12a}$$

$$x_{ij} \leq C_{ij},\ \forall\, (i,j) \in \mathcal{E} \tag{12b}$$

$$x_{\to i}^{(\geq l+1)} + \lambda_i^{(\geq l)} \geq x_{i\to}^{(\geq l)},\ \forall\, i \in \mathcal{V}, l \in \mathcal{L} \tag{12c}$$

$$x_{ij}^{(0)} = x_{dk}^{(l)} = 0,\ \forall\, k \in \delta_d^+, (i,j) \in \mathcal{E}, l \in \mathcal{L}. \tag{12d}$$

Furthermore, for any point within the capacity region, there exists a randomized policy $*$ to support it while attaining optimal cost performance.

Proposition 3: For any point $(\boldsymbol{\lambda}, \gamma) \in \Lambda_1 = \Lambda_2$, the associated flow space $\Gamma_1 = \Gamma_2$.

Proof: All proofs can be found in [12]. In Proposition 2, given $\boldsymbol{x}$ satisfying (12), the feasible randomized policy $*$ (for $\mathscr{P}_1$) operates as follows: in each time slot, any packet of lifetime $l \in \mathcal{L}$ at node $i \in \mathcal{V}$ has a probability

$$\alpha_i^{(l)}(j) = x_{ij}^{(l)} \Big/ \Big(x_{\to i}^{(\geq l+1)} + \lambda_i^{(\geq l)} - x_{i\to}^{(\geq l+1)}\Big) \tag{13}$$

to be sent to node j, and stay in node i otherwise; and the policy $*$ achieves the flow assignment $\boldsymbol{x}$. □

The previous propositions can be explained as follows: by Proposition 1, in general, the admissible policy spaces $\mathcal{A}_1 \nsubseteq \mathcal{A}_2$ and $\mathcal{A}_2 \nsubseteq \mathcal{A}_1$; Proposition 2 suggests that they lead to the same capacity regions by presenting an explicit identical characterization (12), where (12c) is interpreted as the generalized *flow conservation* law when considering the packets' lifetime; Proposition 3 further shows that $\mathscr{P}_1$ and $\mathscr{P}_2$ share the same flow space for any $(\boldsymbol{\lambda}, \gamma)$, which is a crucial property for the considered problem, where the two metrics of interest, i.e., timely throughput (7) and resource cost (8), are both *linear* functions of the flow assignment.

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

Proof: Because they have the same flow space. □

IV. Proposed Control Policy

In this section, we provide a solution for $\mathscr{P}_2$ leveraging Lyapunov optimization theory, and take advantage of Propositions 2 and 3 to develop an algorithm for $\mathscr{P}_1$ based on it.

A. Solution to the Virtual Network Problem

We define the Lyapunov function as $L(t) = \|\boldsymbol{U}(t)\|_2^2/2$, and Lyapunov drift $\Delta(\boldsymbol{U}(t)) = L(t+1) - L(t)$. The LDP approach advocates to minimize a linear combination of the Lyapunov drift (see [12]) and the cost function weighted by a tunable parameter V (which controls the tradeoff between network congestion and operational cost), i.e.,

$$\Delta(\boldsymbol{U}(t)) + Vh(\boldsymbol{\nu}(t)) \leq B - \langle \tilde{\boldsymbol{a}}, \boldsymbol{U}(t)\rangle - \langle \boldsymbol{w}(t), \boldsymbol{\nu}(t)\rangle \tag{14}$$

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

$$w_{ij}^{(l)}(t) = -Ve_{ij} - U_i^{(\leq l)}(t) + \begin{cases} U_d(t) & j = d \\ U_j^{(\leq l-1)}(t) & j \neq d \end{cases} \tag{15}$$

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

To sum up, the developed algorithm aims to solve the following problem in each time slot

$$\max_{\boldsymbol{\nu}(t) \succeq 0} \ \langle \boldsymbol{w}(t), \boldsymbol{\nu}(t)\rangle,\ \text{s.t. } \nu_{ij}(t) \leq C_{ij},\ \forall\, (i,j) \in \mathcal{E}. \tag{16a}$$

The solution is in the *max-weight* fashion. More concretely, for each link (i,j), we first find the lifetime $l^\star$ with the largest weight, and spend all the transmission resource to transmit packets of this lifetime if the associated weight is positive. To sum up, the optimal flow assignment is

$$\nu_{ij}^{(l)}(t) = C_{ij}\, \mathbb{I}\big\{l = l^\star, w_{ij}^{(l^\star)}(t) > 0\big\} \tag{17}$$

where the optimal lifetime choice is $l^\star = \arg\max_{l \in \mathcal{L}} w_{ij}^{(l)}(t)$, and $\mathbb{I}\{\cdot\}$ denotes the indicator function, which equals 1 when the two events in the bracket are both true.

Due to the additive form of the objective function, which is composed of sub-problems that can be completed in each individual node, the algorithm can be implemented in a fully distributed manner.

B. Performance Analysis

In this part, we present a proposition analyzing the performance of the proposed control policy related to the timely throughput (for reliability constraint (7)) and resource cost.

Definition 4 (ε-Convergence Time): The ε-convergence time t_ε is the first time index, such that the achieved reliability level is within a gap of ε from the desired value ever after, i.e.,

$$t_\varepsilon \triangleq \min_{\tau} \Big\{ \sup_{s \geq \tau} \Big[\gamma \|\boldsymbol{\lambda}\|_1 - \sum_{t=0}^{s-1} \frac{\mathbb{E}\{\nu_{\to d}(t)\}}{s}\Big] \leq \varepsilon \Big\}. \tag{18}$$

Proposition 4: For any point in the interior of the capacity region, under the proposed algorithm, the virtual queues are mean rate stable with a convergence time $t_\varepsilon \sim \mathcal{O}(V)$ for any $\varepsilon > 0$, and the achieved cost performance satisfies

$$\overline{\{\mathbb{E}\{h_2(\boldsymbol{\nu}(t))\}\}} \leq h_2^\star(\boldsymbol{\lambda}, \gamma) + B/V \tag{19}$$

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 [12]. □

From the above proposition, we find that by pushing the parameter $V \to \infty$, the achieved cost performance approaches the optimal cost (since the gap B/V vanishes), while compromising the convergence time.

C. Flow Matching

In this section, we develop a near-optimal control policy for $\mathscr{P}_1$. According to Proposition 3 and Corollary 1, there exists a randomized policy (specified by probability values $\boldsymbol{\alpha}$) to be optimal, and we aim to find the solution to $\mathscr{P}_1$ in this categoty. Instead of find $\boldsymbol{\alpha}$ directly (not straightforward), the proposed approach will leverage (13) and calculate the parameters therein using empirical virtual flow decisions. In the following, we denote the decided flow for $\mathscr{P}_1$ on time slot t by $\boldsymbol{\mu}(t) = \{\mu_{ij}^{(l)}(t)\}$ (to distinguish it from $\boldsymbol{\nu}(t)$).

The goal of the designed policy is to conform with the constraints in $\mathscr{P}_1$ (i.e., satisfying (9b) – (9d)), while pursuing the goal of *flow matching*, i.e., $\overline{\{\boldsymbol{\mu}(t)\}} = \overline{\{\boldsymbol{\nu}(t)\}}$. The reason to set the above goal is two-fold. (i) It ensures that the two algorithms attain the same throughput and cost performance (as mentioned earlier, both metrics are linear functions of the flow assignment); therefore, $\{\boldsymbol{\mu}(t)\}$ satisfies the reliability constraint and achieves the same (and thus near-optimal by Corollary 1) cost performance as $\{\boldsymbol{\nu}(t)\}$. (ii) The existence of the policy is guaranteed (as a result of identical flow spaces). Actually, given a feasible flow assignment $\boldsymbol{x}$ satisfying (12) (specifically, $\overline{\{\boldsymbol{\nu}(t)\}}$), we are already aware of the construction procedure of the randomized policy $*$ to achieve it (see *Proof* to Proposition 2 in Section III-B).

While we do not wait until the exact value of $\overline{\{\boldsymbol{\nu}(t)\}}$ is obtained (actually it takes forever) to construct the randomized policy $*$; as an alternative, its empirical values are employed. In each time slot, we first calculate the probability values $\boldsymbol{\alpha}(t)$ according to (13), using the *finite-horizon average* $\bar{\boldsymbol{\nu}}(t) = \frac{1}{t}\sum_{\tau=0}^{t-1}\boldsymbol{\nu}(\tau)$ as the flow assignment $\boldsymbol{x}$, and estimating $\boldsymbol{\lambda}$ from the empirical arrivals by $\hat{\lambda}_i^{(\geq l)} = \frac{1}{t}\sum_{\tau=0}^{t-1} a_i^{(\geq l)}(\tau)$; then node i transmits packets of lifetime l to node j according to the obtained distribution. It leads to a time-varying randomized policy, but we stress that $\bar{\boldsymbol{\nu}}(t)$ converges to $\overline{\{\boldsymbol{\nu}(t)\}}$ asymptotically, which no longer changes over time.[6]

In addition to deciding the virtual flow by the algorithm in Section IV-A, the developed randomized policy requires each node to record *its own* incoming and outgoing flows to calculate the probability values by (13), which can be completed locally. Therefore, the proposed design can operate in a fully distributed manner.

Proposition 5: For any point in the interior of the capacity region, the proposed control policy is admissible, while achieving the near-optimal cost performance of $h(\overline{\{\boldsymbol{\nu}(t)\}})$.

Proof: See [12]. □

V. Numerical Experiments

In this section, we carry out several numerical experiments to evaluate the performance of the proposed design, based on the Abilene US continental network in Fig. 1.

We take a simple AgI service for example,[7] which requires the source data-stream to be processed by one function, and we assume that each node (representing a data center) in the network can host the given service function. To describe the function's computation resource requirement, we assume that a CPU (a measure of computing resource) is capable of processing the incoming data-stream at a rate of 50 Mbps, and the output data-stream has the same size as the input.

The available network resource and the associated cost is described in the following: each node in the network has a an average resource consumption budget of 2 CPUs and the associated cost is 1 /CPU at $i \in \{5, 6\}$, and 2 /CPU at other data centers; each link exhibits the same average transmission rate of 1 Gbps, with a cost of 1 /Gb.

There are two clients, i.e., (source, destination) pairs, requesting the service, i.e., $(1, 9)$ and $(3, 11)$, both at the reliability level of $\gamma = 90\%$. The packets arrive at the source nodes according to independent Poisson processes of parameter λ. The lifetime of all packets at birth equals to the maximum lifetime L (≥ 5, including 4 time slots for transmission via the shortest path, and 1 time slot for processing).

A. Network Capacity Region

We first study the network capacity regions achieved by the proposed algorithm, assuming different maximum lifetimes L. We run the proposed algorithm (with $V = 0$) on the network for 1×10^6 time slots, recording the queue backlog of the virtual network, and the 0.005-convergence time for the actual network (i.e., when achieved reliability level $\geq 89.5\%$).

The results are shown in Fig. 2, and we make the following observations. First, for fixed L, both the virtual queue (solid lines) and the convergence time (dashed lines) blow up after the arrival rate exceeds a critical point, which is interpreted as the boundary of the capacity region. The result verifies that the virtual and the actual network have the same capacity regions (as stated in Proposition 2). Second, by increasing the value of L, the capacity region enlarges, since the packets can detour to farther network locations for extra computing resources, while still arriving at the destinations within the deadline. When $L = 5$, the packets must follow the shortest paths to the destinations, and the two clients must share the constrained

[6] It is possible that the finite-horizon average $\bar{\boldsymbol{\nu}}(t)$ can violate (12c) at some time slot, and thus does not make a qualified candidate for $\boldsymbol{x}$. However, as is mentioned, $\lim_{t\to\infty}\bar{\boldsymbol{\nu}}(t) = \overline{\{\boldsymbol{\nu}(t)\}}$, which satisfies the constraints. With this asymptotic guarantee, when such violation occurs, we can choose not the update the control policy in that time slot.

[7] In the setup of the experiments, we adopt a simple example for illustrative purposes, and we refer the readers to [12] for more realistic AgI services.

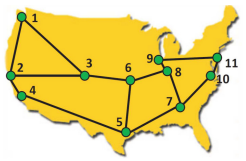

Fig. 1. The studied network.

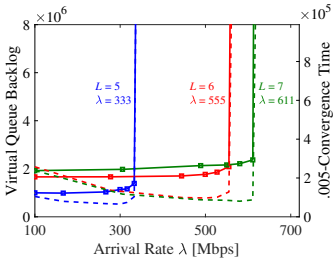

Fig. 2. Capacity regions.

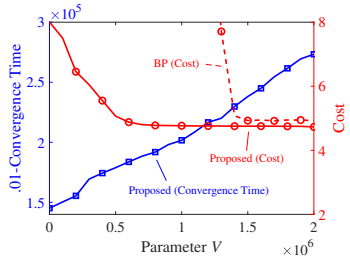

Fig. 3. Tradeoff controlled by V.

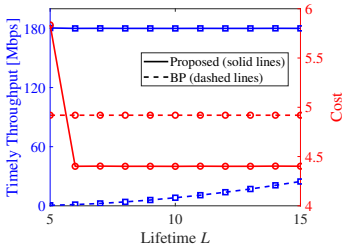

Fig. 4. Effects of packets' lifetime.

computing resource at the common nodes on the paths (data centers $3, 6, 8, 9$); when $L = 6$, client $(3, 11)$ can detour along $3 \to 6 \to 5 \to 7 \to 10 \to 11$ for extra computing resource, boosting the capacity region; when $L \geq 7$, the computing resources of the entire network are fully exploited.

B. Effects of Parameter V

Next, we study the convergence time and the resource cost achieved by the proposed algorithm under various V (using $L = 7$ and $\lambda = 100$ Mbps). In addition, we compare the results with the state-of-the-art min-cost max-throughput algorithm, referred to as DCNC [4].

The result is depicted in Fig. 3. First, we focus on the performance of the proposed algorithm (solid lines), which exhibits the $[\mathcal{O}(V), \mathcal{O}(1/V)]$ tradeoff between the convergence time and the resource cost, as is presented in Proposition 4. Second, for the DCNC algorithm, we observe from the experiment that the achieved timely throughput is around 10 Mbps (i.e., a reliability level of 10%, see next section for more results), failing the reliability constraint (resulting in a convergence time of ∞). For the cost performance, when $V \leq 1 \times 10^6$, it leads to a much higher resource cost ($\geq$ 15) than the proposed algorithm, since the packets can take cyclic paths to the destination, incurring extra cost; as V grows, the cost reduces, while it is still higher than the proposed algorithm because it delivers all the packets (even the outdated ones).

C. Effects of Lifetime L

Finally, we present the timely throughput and the resource cost achieved by the proposed and the DCNC algorithms, under various maximum lifetimes ($\lambda = 100$ Mbps, and a large $V = 5 \times 10^7$ is selected to ensure near-optimal resource cost).

As we can observe from Fig. 4, the proposed algorithm attains a (sum) timely throughput of 180 Mbps, i.e., a reliability level of 90% for each client, where the reliability constraint holds with equality (the existence of the .01-convergence time in the previous experiment also supports the result). In contrast, the DCNC algorithm achieves much lower timely throughput, e.g., 20 Mbps when $L = 15$, where the packets are provided 10 extra time slots for transmission. Finally, we point out that the resource cost of the proposed algorithm significantly improves when L turns 6, where the two clients can follow the network paths (i) client 1: $1 \to 3 \to 6\,(\text{processing}) \to 8 \to 9$, (ii) client 2: $3 \to 6 \to 5\,(\text{processing}) \to 7 \to 10 \to 11$ to benefit from cheap computation resources at node 5 and 6.

VI. Conclusion

In this paper, we investigated the problem of optimal cloud network control with strict deadline constraints. We established a new queuing system to keep track of the data packets' lifetimes, on which basis we formalized the problem $\mathscr{P}_1$. An equivalent problem $\mathscr{P}_2$ was derived, for which we provided a solution leveraging Lyapunov optimization theory. We then took advantage of their close relationship (identical capacity region, flow space, and optimal cost value), to develop a provably near-optimal, fully distributed algorithm for $\mathscr{P}_1$, from the empirical decisions made for $\mathscr{P}_2$. Numerical results validated the theoretical analysis and the performance gain of the proposed design over the state-of-the-art algorithm.

References

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

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

[3] 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.

[4] 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.

[5] J. Zhang, A. Sinha, J. Llorca, A. Tulino, and E. Modiano, "Optimal control of distributed computing networks with mixed-cast traffic flows," in *Proc. IEEE INFOCOM*, Honolulu, HI, USA, 2018, pp. 1880–1888.

[6] 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.

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

[8] 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.

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

[10] 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.

[11] 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.

[12] Y. Cai, J. Llorca, A. M. Tulino, and A. F. Molisch, "Ultra-reliable low-cost cloud network control with strict deadline constraints," *submitted to IEEE/ACM Trans. Netw.*

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