Optimal Multicast Service Chain Control: Packet Processing, Routing, and Duplication

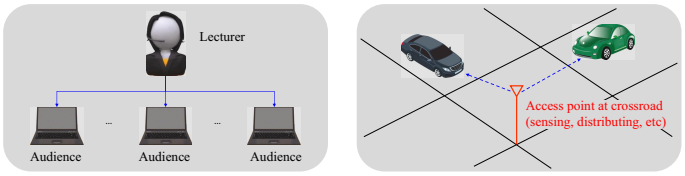
Yang Cai*, Jaime Llorca†, Antonia M. Tulino†‡, Andreas F. Molisch*
*University of Southern California, CA 90089, USA. Email: {yangcai, molisch}@usc.edu
†New York University, NY 10012, USA. Email: {jllorca, atulino}@nyu.edu
‡Universityà degli Studi di Napoli Federico II, Naples 80138, Italy. Email: antoniamaria.tulino@unina.it

***Abstract*—Distributed computing (cloud) networks, e.g., mobile edge computing (MEC), are playing an increasingly important role in the efficient hosting, running, and delivery of real-time stream-processing applications such as industrial automation, immersive video, and augmented reality. While such applications require timely processing of real-time streams that are simultaneously useful for multiple users/devices, existing technologies lack efficient mechanisms to handle their increasingly multicast nature, leading to unnecessary traffic redundancy and associated network congestion. In this paper, we address the design of distributed packet processing, routing, and duplication policies for optimal control of multicast stream-processing services. We present a characterization of the enlarged capacity region that results from efficient packet duplication, and design the first fully distributed *multicast traffic management* policy that stabilizes any input rate in the interior of the capacity region while minimizing overall operational cost. Numerical results demonstrate the effectiveness of the proposed policy to achieve throughput- and cost-optimal delivery of stream-processing services over distributed computing networks.**

I. Introduction

The proliferation of real-time stream-processing applications such as augmented reality, telepresence, and industrial automation [1], is pushing the evolution of networking and cloud technologies in order to meet their stringent low latency and compute-intensive requirements [2]. Traditional approaches treat network and cloud resources separately, with fairly centralized core clouds handling the processing of compute-intensive tasks, while the network takes care of routing data streams from sources to the cloud, and back to their destinations. However, next-generation services can be decomposed into chains of individual functions that allows a more flexible and granular processing of data streams at distributed cloud locations. Service function chaining precisely refers to the routing of traffic flows through an ordered sequence of service functions deployed at multiple cloud locations [2].

In recent years, multicast data-streams (contents with multiple destinations) have become an increasingly dominant component of network traffic,[1] especially in the coming Internet of things (IoT) era. For example, multi-user conferencing (Fig. 1a) requires to encode and deliver source information to several audiences; applications of another type, which can be summarized as joint decision making of multi-agent systems, including robot (or car) coordination in smart factory (or intelligent vehicle system, as shown in Fig. 1b), also require the access point to distribute the sensing information/decided actions to multiple end nodes.



(a) Multi-user conferencing. (b) Smart vehicle system.

Fig. 1. Two widely-used applications involving multicast network traffic: a) multiuser conferencing and b) vehicle coordination, which require the source information to be processed and delivered to multiple destination nodes.

In order to maximize the benefit of distributed computing networks to support multicast services, two fundamental problems need to be addressed:

- how to instantiate processing functions on edge/cloud servers and route the data-stream through them;
- how to schedule and allocate network (computing and transmission) resources for different requests.

The first problem, usually referred to as service function chain (SFC) optimization, involves jointly allocating tightly coupled cloud and network resources in order to decide where to run each service function and how to route service flows through the appropriate sequence of functions in order to maximize throughput and minimize overall operational cost. A number of recent works have addressed the SFC optimization problem with the goal of either maximizing accepted service requests or minimizing overall resource cost [4]–[6]. However, the problem is usually formulated under a static configuration, without taking into account increasingly prominent uncertain network conditions and time-varying service demands.

For the second problem, a closely related research field is *dynamic packet routing*, which has been extensively studied in the past, with two main celebrated mechanisms for decision making. On one hand, *source routing* schemes determine the entire route of the packet to the destination at the source node. In [7], a universal throughput-optimal source routing policy is designed for both unicast and multicast traffic. On the other hand, distributed routing schemes based on the *fluid model* determine packet routes based on local decisions

[1]To clarify, the term *multicast* in this paper refers to content delivery to multiple destinations, not related to the wireless communication technique of transmitting data to multiple nodes simultaneously, as considered in [3].

on a hop-by-hop basis. The backpressure algorithm [8] is an example of such policy that achieves throughput-optimal routing for unicast traffic. While, in general, source routing can achieve better delay performance, its centralized nature incurs additional overhead in collecting network-wide state information, making it more suitable for regimes with low congestion levels and relatively stable arrival rates. On contrary, distributed fluid-based algorithms only require local information exchange and decision making, and while they can suffer from inefficient *loopy* routes in low congestion scenarios, they are especially suitable for high congestion regimes. Besides, a recent study [9] proposes a distributed, backpressure-fashioned network control policy, which is designed to support services with stringent latency constraints.

Extensions of the above policies for SFCs have also been studied in recent works, either by introducing the computation flow [10] or constructing the layered graph [11]. More concretely, [11] investigates throughput-optimal service chain source routing for both unicast and multicast traffic; [10], [12] study throughput and cost optimal service chain distributed routing and resource allocation for unicast traffic (in particular, [12] addresses the related problems under a MEC network scenario). However, no throughput-optimal fully *distributed* policies have been designed for the multicast service chain control problem.

Motivated by the increasing multicast nature of next-generation real-time stream-processing services and scalating network congestion levels, in this paper, we focus on the design of throughput and cost optimal multicast service chain control policies. Multicast routing policies are of paramount importance to avoid excessive network congestion from unnecessary traffic redundancies. However, the main challenge in the design of distributed multicast routing policies is the difficulty to capture in-network packet duplication mechanisms that break flow conservation laws.

In this work, we provide the first formal analysis of fully distributed multicast routing policies (for arbitrary communication and computation services) that include joint packet processing, routing, and duplication. Our contributions can be summarized as follows:

- We characterize the enlarged multicast computing network capacity region that results when allowing in-network packet duplication.
- We develop the first throughput- and cost-optimal fully distributed packet processing, routing, and duplication policy for multicast service chain control.
- We present numerical results demonstrating the enlarged multicast capacity region, and the tunable $[\mathcal{O}(V), \mathcal{O}(1/V)]$ cost-delay tradeoff associated with the proposed control policy.

II. System Model

A. Cloud network

We consider a wide-area distributed computing network, simply referred to as *cloud network*, modeled by graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each node $i \in \mathcal{V}$ represents a network node with computing capabilities (e.g., core cloud, edge cloud, compute-enabled base station). Data can be transmitted from node i to j via network link $(i, j) \in \mathcal{E}$. We denote by δ_i^- and δ_i^+ the incoming and outgoing neighbors of node i, respectively.

Assuming a time-slotted system, the available processing/ transmission resources, and associated costs, are defined as

- C_i: the processing capacity, e.g., the number of computing cycles per time slot, at node i;
- e_i: the processing cost, i.e., the cost of running one unit of processing resource, at node i;
- C_{ij}: the transmission capacity, i.e., the data-stream size that can be transmitted in one time slot, on link (i, j);
- e_{ij}: the transmission cost, i.e., the cost of transmitting one unit of data, on link (i, j).

B. Service Chain

The cloud network offers a set of services Φ. Each service $\phi \in \Phi$ is modeled as an ordered chain of $(M_\phi - 1)$ functions, through which incoming packets must be processed. Functions can be executed at different network locations. While, for ease of exposition, we assume every cloud node can host any service function, it is straightforward to extend our model to limit the set of functions available at each cloud node. There are two parameters associated with each function: for the m-th function of service ϕ, we define

- $\xi_\phi^{(m)}$: the scaling factor, i.e., the output data-stream size per unit of input data-stream;
- $r_\phi^{(m)}$: the workload, i.e., the amount of computing resource required to process one unit of input data-stream;

We refer to the input and output data-streams of service ϕ as the stage m and stage $m + 1$ data-streams of service ϕ, respectively. Data-streams are divided into packets of uniform length, and we assume that each packet can be processed separately.

In order to characterize the multicast nature of offered services, we assume each service $\phi \in \Phi$ is consumed by a set of destinations denoted by $\mathcal{D} = \{d_1, \cdots, d_D\}$ with $\mathcal{D} \subset \mathcal{V}$ and $|\mathcal{D}| = D$.

C. Data Management

In the unicast service control problem [10], there are two relevant packet operations, i.e., processing and transmission. For multicast service control, we add the *packet duplication* operation to allow any network node to make two copies of any incoming packet.

Originally, prior to any duplication operation, each packet of a given service is associated with the entire destination set $\mathcal{D}$. After a duplication operation, each resulting copy is associated with a new destination set. The key requirement for any duplication operation is the **coverage** of the original destination set, i.e., each destination node of the original packet must be present in the destination set of least one of the resulting copies. If the destination sets of the resulting copies do not overlap, the duplication operation is termed *efficient* (and *inefficient* otherwise).

To keep track of the changes in the destination sets after packet duplication operations, we introduce the concept of packet *duplication status*.

Definition 1 (Duplication Status): The duplication status of a packet, denoted by $q = [q_1, \cdots, q_D] \in 2^D$, is a binary vector with $q_k = 1$ $(k = 1, \cdots, D)$ indicating that $d_k \in \mathcal{D}$ is one of its current destinations.

In the above definition, 2^D is the set of indicator vectors corresponding to the power set of $\mathcal{D}$. In addition, we define a subset of it as $2^q \triangleq \{s : s_k = q_k u_k \text{ with } u \in 2^D\}$, which collects s whose entry must be 0 if the entry is 0 in q. Specially, $q = b_k$ (the binary vector with only the k-th entry equal to 1) indicates a packet with only one destination d_k (behaves as a unicast packet); and $q = \mathbf{0}$ indicates a packet with no destination, which is not of interest and all the related quantities should be ignored.

We define a commodity as the collection of packets with the same 4-tuple $(\phi, m, \mathcal{D}, q)$ description, i.e., service ϕ, stage m, destination set $\mathcal{D}$, and duplication status q. To simplify the notation, we define $c \triangleq (\phi, m, \mathcal{D})$, and label a commodity as (c, q).

Finally, we define the process of exogenous arrival of packets of commodity (c, q) at node i as $\{a_i^{(c,q)}(t) : t \geq 0\}$. All the arriving processes are assumed to be i.i.d. over time slots and independent with each other, with mean rate $\mathbb{E}\{a_i^{(c,q)}(t)\} = \lambda_i^{(c,q)}$, and finite second moment.

III. Policy Space

In this section, we first present a general policy space for multicast service control, as well as the conditions for a policy to be admissible. We then describe an efficient policy space, by restricting the duplication process to be efficient, which does not reduce the performance (capacity region and the achievable optimal cost).

A. General Policy Space

We consider a general policy space for multicast service control, encompassing all packet processing, routing, and duplication policies. The decisions made by a policy in this space can be described by the following variables

$$f(t) = \left\{ f_{i,\text{pr}}^{(c,q)}(t), f_{ij}^{(c,q)}(t) : \forall\, (c,q), i \in \mathcal{V}, (i,j) \in \mathcal{E} \right\} \quad (1)$$

which are the amount of packets of each commodity that are operated (processed or transmitted) on each interface.

A control policy is called *admissible*, if it makes decisions satisfying the following constraints: 1) non-negativity

$$f(t) \succeq 0 \quad \text{(element-wise)} \quad (2)$$

2) capacity constraints (recall that $c = (\phi, \mathcal{D}, m)$)

$$\tilde{f}_i(t) = \sum\nolimits_{(c,q)} r_\phi^{(m)} f_{i,\text{pr}}^{(c,q)}(t) \leq C_i, \quad \forall\, i \in \mathcal{V} \quad (3a)$$

$$f_{ij}(t) = \sum\nolimits_{(c,q)} f_{ij}^{(c,q)}(t) \leq C_{ij}, \quad \forall\, (i,j) \in \mathcal{E} \quad (3b)$$

3) the generalized flow conservation and duplication law, for any commodity c and $\forall\, k \in \{1, \cdots, D\}$:

$$\sum\nolimits_{\{q:q_k=1\}} \left[f_{\to i}^{(c,q)} + \lambda_i^{(c,q)} \right] \leq \sum\nolimits_{\{q:q_k=1\}} f_{i\to}^{(c,q)} \quad (4)$$

where $\{q : q_k = 1\}$ is the set of all the duplication status which indicates that $d_k \in \mathcal{D}$ is one of the current destinations; the incoming and outgoing flows are

$$f_{\to i}^{(c,q)} = \overline{\left\{ f_{\text{pr},i}^{(c,q)}(t) + \sum\nolimits_{j \in \delta_i^-} f_{ji}^{(c,q)}(t) \right\}} \quad (5a)$$

$$f_{i\to}^{(c,q)} = \overline{\left\{ f_{i,\text{pr}}^{(c,q)}(t) + \sum\nolimits_{j \in \delta_i^+} f_{ij}^{(c,q)}(t) \right\}} \quad (5b)$$

with the processed flow $f_{\text{pr},i}^{(c,q)}(t) = f_{\text{pr},i}^{(\phi,m,\mathcal{D},q)}(t)$ defined as

$$f_{\text{pr},i}^{(\phi,m,\mathcal{D},q)}(t) = \begin{cases} 0 & m = 1 \\ \xi_\phi^{(m-1)} f_{i,\text{pr}}^{(\phi,m-1,\mathcal{D},q)}(t) & m > 1 \end{cases} \quad (6)$$

and $\overline{\{\cdot\}}$ denotes the long-term average operator

$$\overline{\{z(t)\}} \triangleq \lim_{T\to\infty} \frac{1}{T} \sum\nolimits_{t=1}^{T} z(t). \quad (7)$$

The generalized flow conservation and packet duplication law (4) holds because of the **coverage** requirement (see previous section). For any destination d_k of an incoming packet, there is *at least* one outgoing packet (one of its copies if duplicated, or itself otherwise) with d_k in its destination set.

The instantaneous overall resource cost incurred by the above policy is defined as

$$h(t) = \sum\nolimits_{i \in \mathcal{V}} e_i \tilde{f}_i(t) + \sum\nolimits_{(i,j) \in \mathcal{E}} e_{ij} f_{ij}(t) \quad (8)$$

and its long-term average $\overline{\{h(t)\}}$ is employed to characterize the cost performance of the policy. Furthermore, we denote by $h^\star(\boldsymbol{\lambda})$ the optimal cost that can be achieved by the general policy space, under the arrival rate $\boldsymbol{\lambda}$.

Finally, we define the *capacity region* Λ of the cloud network as the set of all arrival vectors $\boldsymbol{\lambda} = \{\lambda_i^{(c,q)}\}$, such that there exists a control policy satisfying (2) – (6).

B. Efficient Policy Space

We now define an efficient policy space as a subset of the general space, by requiring all the duplication operations to be efficient. More concretely, if two copies are created from a packet by a duplication operation, then

$$q = s + r \quad (9)$$

with $q, s, r \in 2^D$ denoting the duplication status of the original packet and the two copies, respectively.

When a duplication is performed in an efficient way, for any destination node of a particular incoming packet, there will be *exactly* one outgoing packet steering to it. In this case, the flow conservation and duplication law can be cast as

$$\sum\nolimits_{\{q:q_k=1\}} \left[f_{\to i}^{(c,q)} + \lambda_i^{(c,q)} \right] = \sum\nolimits_{\{q:q_k=1\}} f_{i\to}^{(c,q)}. \quad (10)$$

By restricting to the efficient space, we eliminate repeated delivery of identical content to the same destination node,

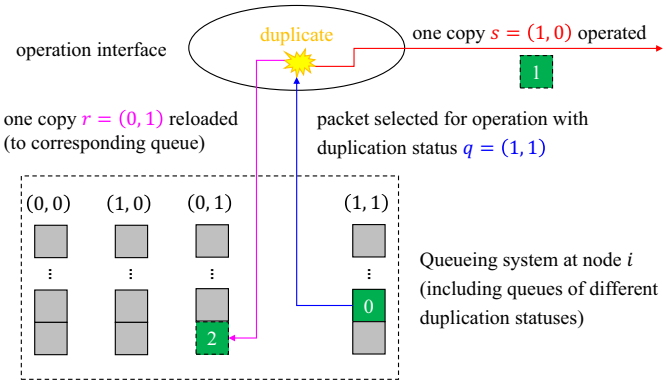


Fig. 2. Structure of the established queueing system at any network node i (for an application with $D = 2$ destination nodes). In order to distinguish packets with different current destination set, we create $2^D = 4$ queues corresponding to the 4 duplication statuses, i.e., $\{0,1\}^2$. In each time slot, in addition to the scheduling decision, i.e., which packets will be operated at which interface (the blue link), we also need to make a duplication decision, i.e., whether to split the network flow or not, and how (mathematically, to determine the duplication status of the operated copy s and the reloaded copy r). When $r = (0,0)$ (and thus $s = q$), the packet is operated without changing the assigned destination set, and no copy is created in this case (in fact, node i does not need to manage packets in the $(0,0)$ queue; we present it in the figure just for completeness).

which is beneficial for 1) alleviating the network traffic, as well as 2) reducing the resource cost. Specially, this is true when comparing with the optimal policy of the general space. As a consequence, the efficient policy space can achieve the same *capacity region* as the general space, and the achievable *optimal cost* by the efficient policy space equals to $h^\star(\boldsymbol{\lambda})$.

IV. Queueing System

We construct the queueing system by creating a queue $Q_i^{(c,q)}(t)$ for each commodity (c,q) at each node i.

The efficient policy space is considered, and we describe a typical operation procedure for a packet in one time slot in the following. Suppose a packet of duplication status q is selected for operation (processing or transmission) on a certain interface, we need to decide whether it will be duplicated or not.[2] If a packet is duplicated, only one copy is *operated* on the interface, while the other copy is *reloaded* to the queueing system at the end of the time slot (i.e., it is not involved in any other decisions in the current time slot).

The above description motivates us to involve the *posterior* duplication status $s \in 2^q$ in the formulation, which is the status of the operated copy (and by (9), the status of the reloaded copy is $q - s$). Specially, the case $q = s$ indicates that the packet is not duplicated. To sum up, the (q,s)-pair specifies a duplication decision.

[2]We consider the scheme where each packet is duplicated at most once in a time slot. Compared to a more general scheme without this restriction, the considered scheme just splits duplications into multiple steps, and that does not increase traffic, while only increasing delay by a finite amount of slots, which does not affect the capacity region or the cost performance.

A. Queueing Dynamics

Let $x_{i,\mathrm{pr}}^{(c,q,s)}(t)$ and $x_{ij}^{(c,q,s)}(t)$ $(j \in \delta_i^+)$ be the amount of packets of commodity (c,q) desired by the output interfaces, on which the duplication decision (q,s) will be performed. In general, the queueing dynamics is given by

$$Q_i^{(c,q)}(t+1) \le \left[Q_i^{(c,q)}(t) - \sum\nolimits_{s\in 2^q} \mu_{i\to}^{(c,q,s)}(t)\right]^+ + \mu_{\to i}^{(c,q)}(t) + a_i^{(c,q)}(t) \tag{11}$$

where the outgoing flow is

$$\mu_{i\to}^{(c,q,s)}(t) = x_{i,\mathrm{pr}}^{(c,q,s)}(t) + \sum\nolimits_{j\in\delta_i^+} x_{ij}^{(c,q,s)}(t) \tag{12}$$

and the (controllable) incoming flow is

$$\mu_{\to i}^{(c,q)}(t) = \sum\nolimits_{s\in 2^{\bar{q}}} \left[x_{\mathrm{pr},i}^{(c,q+s,q)}(t) + \sum\nolimits_{j\in\delta_i^-} x_{ji}^{(c,q+s,q)}(t)\right] + \sum\nolimits_{s\in 2^{\bar{q}}} \mu_{i\to}^{(c,q+s,s)}(t) \tag{13}$$

with $\bar{q} = \mathbf{1} - q$; and $[z]^+ \triangleq \max\{z, 0\}$. The two lines in (13) represent the operated and the reloaded packets, respectively. The reloaded part is explained as follows: a packet of status $q + s$ is duplicated, with the copy of status s operated; thus the other copy of status $(q + s) - s = q$ will be reloaded.

Specially, (11) does not apply to queues of *destination state*, i.e., $Q_{i_0}^{(c_0,q)}(t)$ with $i_0 = d_k \in \mathcal{D}$ and $c_0 = (\phi, M_\phi, \mathcal{D})$. If a packet of commodity (c_0,q) (with $q_k = 1$) arrives at i_0, it will be consumed. But due to the multicast nature of the packet (in general), it will be duplicated into two copies of status b_k and $q' = q - b_k$ (and thus $q'_k = 0$), with the copy of b_k departing the network, and the other copy reloaded to the queue q'. Therefore, the queue q is always empty, while queue q' receives an extra packet compared to the general case. To sum up, in this case, the queueing dynamics is given by

$$Q_i^{(c,q)}(t+1) \le \begin{cases} 0 & q_k = 1 \\ R + \mu_{\to i}^{(c,q+b_k)}(t) + a_i^{(c,q+b_k)}(t) & q_k = 0 \end{cases} \tag{14}$$

where R is the right-hand-side of (11).

B. Problem Formulation

Based on the queueing system introduced in the previous section, mathematically, the multicast service chain control problem is formulated as

$$\min_{\boldsymbol{x}(t)} \quad \overline{\{\mathbb{E}\{h(t)\}\}} \tag{15a}$$

$$\text{s. t.} \quad \text{stabilizing the queueing system (11) – (14)} \tag{15b}$$

$$x_{\mathrm{pr},i}^{(\phi,m+1,\mathcal{D},q,s)}(t) = \xi_\phi^{(m)} x_{i,\mathrm{pr}}^{(\phi,m,\mathcal{D},q,s)}(t) \tag{15c}$$

$$\tilde{x}_i(t) \triangleq \sum_{(c,q,s)} r_\phi^{(m)} x_{i,\mathrm{pr}}^{(c,q,s)}(t) \le C_i \quad \forall\, i \in \mathcal{V} \tag{15d}$$

$$x_{ij}(t) \triangleq \sum_{(c,q,s)} x_{ij}^{(c,q,s)}(t) \le C_{ij} \quad \forall\, (i,j) \in \mathcal{E} \tag{15e}$$

$$\boldsymbol{x}(t) \succeq 0 \text{ (element-wise)}. \tag{15f}$$

Remark 1: In the above formulation, note that decisions $\boldsymbol{x}(t)$ are made regardless of the available packets in the queue, it can happen that the requests raised by the interfaces cannot

be satisfied. In that case, *dummy* packets will be created and sent to the interface to compensate for the lack of actual packets, as is considered in [10] for the unicast case.

V. Capacity Region

In this section, we present a characterization for the capacity region of cloud network with multicast flows, which is based on the celebrated **fact** [13] that there exists a stationary randomized policy $*$ to stabilize any point within the capacity region, while achieving the optimal objective (cost) value.

Theorem 1: An arrival vector $\boldsymbol{\lambda}$ is within Λ if and only if there exists flow variables $\boldsymbol{f} = \left\{f_{i,\text{pr}}^{(c,q,s)}, f_{ij}^{(c,q,s)}\right\} \succeq 0$ together with probability values $\{\beta_i^{(c,q,s)}\}_{(c,q,s)}$ and $\{\beta_{ij}^{(c,q,s)}\}_{(c,q,s)}$ for $\forall\, i \in \mathcal{V}$, $(i,j) \in \mathcal{E}$ such that

$$\sum_{s\in 2^{\bar{q}}}\left[f_{\text{pr},i}^{(c,q+s,q)} + \sum_{j\in\delta_i^-} f_{ji}^{(c,q+s,q)} + f_{i,\text{pr}}^{(c,q+s,s)} + \sum_{j\in\delta_i^+} f_{ij}^{(c,q+s,s)}\right] + \lambda_i^{(c,q)} \le \sum\nolimits_{s\in 2^q}\left[f_{i,\text{pr}}^{(c,q,s)} + \sum\nolimits_{j\in\delta_i^+} f_{ij}^{(c,q,s)}\right] \quad (16a)$$

$$f_{\text{pr},i}^{(\phi,m+1,\mathcal{D},q,s)} = \xi_\phi^{(m)} f_{i,\text{pr}}^{(\phi,m,\mathcal{D},q,s)} \quad (16b)$$

$$f_{i,\text{pr}}^{(c,q,s)} \le \left(C_i/r_\phi^{(m)}\right)\beta_i^{(c,q,s)} \quad (16c)$$

$$f_{ij}^{(c,q,s)} \le \beta_{ij}^{(c,q,s)} C_{ij}. \quad (16d)$$

and the stationary randomized policy $*$ specified by the probability values β makes decisions $\boldsymbol{x}^*(t)$ such that

$$\overline{\{\mathbb{E}\{h(\boldsymbol{x}^*(t))\}\}} = h^\star(\boldsymbol{\lambda}) \quad (17)$$

with $h^\star(\boldsymbol{\lambda})$ denoting the optimal cost that can be achieved when the arrival vector is $\boldsymbol{\lambda}$.

Proof: The result is derived by applying the **fact** to the queueing system in Section IV-A [13]. Details can be found in [14].

The policy $*$ is defined as follows. For each interface, select the commodity (c,q) and the duplication action (q,s) independently in every time slot according to the probability value β; duplicate the packets according to (q,s), and use all the available resource to operate the copies of status s. □

VI. Control Policy Design

Problem (15) can be solved by Lyapunov drift-plus-penalty (LDP) approach [13], as is shown in the following section.

A. The LDP Approach

We first define the Lyapunov function as $L(t) = \|\boldsymbol{Q}(t)\|_2^2/2$ with $\boldsymbol{Q}(t) = \left\{Q_i^{(c,q,s)}(t)\right\}$, quantifying the current network congestion, and define the drift as $\Delta(t) = L(t+1) - L(t)$.

The LDP approach advocates to minimize (the upper bound of) a linear combination of the Lyapunov drift $\Delta(t)$ and the objective function $h(t) = h(\boldsymbol{x}(t))$ weighted by a tunable parameter V, given by [13]

$$\Delta(t) + Vh(t) \le B - \sum_{i\in\mathcal{V}}\sum_{(c,q,s)} w_i^{(c,q,s)} x_{i,\text{pr}}^{(c,q,s)}(t) - \sum_{(i,j)\in\mathcal{E}}\sum_{(c,q,s)} w_{ij}^{(c,q,s)} x_{ij}^{(c,q,s)}(t) \quad (18)$$

where B is a constant, and the weights are given by

$$w_i^{(c,q,s)} = \frac{Q_i^{(c,q)}(t) - Q_i^{(c,q-s)}(t) - \xi_\phi^{(m)} Q_i^{(c',s)}(t)}{r_\phi^{(m)}} - Ve_i \quad (19a)$$

$$w_{ij}^{(c,q,s)} = Q_i^{(c,q)}(t) - Q_i^{(c,q-s)}(t) - Q_j^{(c,s)}(t) - Ve_{ij} \quad (19b)$$

where $c' = (\phi, m+1, \mathcal{D})$.

The constraints on the decision variables $\boldsymbol{x}(t)$ are given by (15d), (15e) and (15f), which leads to a solution in the form of *max-weight*, presented in the following section.

B. Control Policy

Note that minimizing (18) can be completed separately on each interface (due to the additive form). The processing (or transmission) decisions are made by the following steps: for each node $i \in \mathcal{V}$ (or each link $(i,j) \in \mathcal{E}$),

1) calculate the weight for each tuple (c,q,s) according to (19a) (or (19b)), based on the observed queue status;

2) find the tuple (q,s,c) with the largest weight, i.e.,

$$(q,s,c)^\star = \arg\max_{(q,s,c)}\ w_i^{(q,s,c)} \ \left(\text{or } w_{ij}^{(q,s,c)}\right); \quad (20)$$

3) the optimal flow assignment is given by

$$x_{i,\text{pr}}^{(q,s,c)}(t) = \frac{C_i}{r_{\phi^\star}^{(m^\star)}}\,\mathbb{I}\left\{(q,s,c) == (q,s,c)^\star, w_{ij}^{(q,s,c)^\star}(t) > 0\right\}$$
$$x_{ij}^{(q,s,c)}(t) = C_{ij}\,\mathbb{I}\left\{(q,s,c) == (q,s,c)^\star, w_{ij}^{(q,s,c)^\star}(t) > 0\right\} \quad (21)$$

where $\mathbb{I}\{\cdot\}$ denotes the indicator function, which equals to 1 only when the two conditions are both satisfied.

The developed algorithm only requires local information exchange and decision making, which can be implemented in a fully distributed manner.

C. Performance Analysis

We evaluate the performance of the proposed algorithm in the following theorem, using the achievable optimal cost as the benchmark.

Theorem 2: For any arrival vector $\boldsymbol{\lambda}$ that is in the interior of the capacity region, the queue backlog and the cost achieved by the proposed algorithm satisfy

$$\overline{\{\mathbb{E}\{\|\boldsymbol{Q}(t)\|_1\}\}} \le \frac{B}{\epsilon} + \left[\frac{h^\star(\boldsymbol{\lambda}+\epsilon\mathbf{1}) - h^\star(\boldsymbol{\lambda})}{\epsilon}\right] V \quad (22)$$

$$\overline{\{\mathbb{E}\{h(t)\}\}} \le h^\star(\boldsymbol{\lambda}) + \frac{B}{V} \quad (23)$$

for any $\epsilon > 0$ such that $\boldsymbol{\lambda} + \epsilon\mathbf{1} \in \Lambda$.

Proof: The proof closely follows the philosophy of the proof of Theorem 2 in [10]. □

The above theorem reveals the $[\mathcal{O}(V), \mathcal{O}(1/V)]$ tradeoff between the delay (which is proportional to queue backlog by Little's theorem) and cost performance achieved by the proposed algorithm. In addition, for any fixed V, the queue backlog is mean rate state (i.e., $\overline{\{\mathbb{E}\{\|\boldsymbol{Q}(t)\|_1\}\}} < \infty$), implying that the proposed algorithm is throughput-optimal.

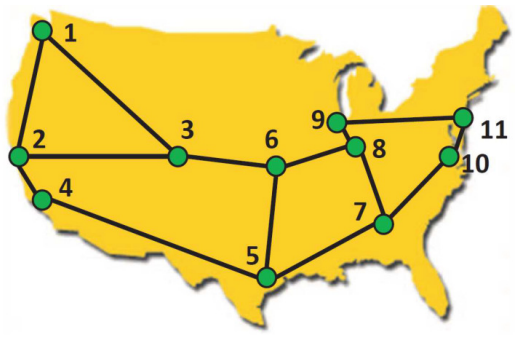


Fig. 3. The continental US Abilene network.

D. Complexity Issue

Finally, we analyze the complexity of the proposed algorithm, from both the communication and computation aspects.

1) Communication Overhead: the proposed algorithm requires local exchange of queue backlog information in every time slot. In contrast to transmitting the entire queueing status $\sim \mathcal{O}(2^D)$ in every time slot, we take advantage of the underlying max-weight structure of the proposed algorithm. More concretely, in every time slot, the proposed algorithm selects one commodity to operate on each interface; as a result, only one element of the queueing vector of node j changes. Therefore, the number of queues with varying backlogs is $\sim \mathcal{O}(\delta^+_{\max})$, where $\delta^+_{\max}$ is the largest incoming degree. By transmitting information related to only these queues, the communication overhead can be greatly reduced.

2) Computational Complexity: In every time slot, each node needs to calculate the weights of all (c, q, s) tuples in order to decide the best commodity to operate on, and make the duplication decision. It can be shown that for a fixed content c, the number of possible (q, s) pairs is $3^D - 2^D \sim \mathcal{O}(3^D)$. Although to calculate the weight for each (c, q, s) (at each interface) by (19) requires only simple algebraic operations, the number of the tuples grows exponentially with the size of the destination set, and there is no quick way to reduce the computation complexity of the algorithm to polynomial-time.[3]

To sum up, with more destination nodes, we can envision larger performance improvement compared to the **simple approach** that treats them as individual unicast flows (since the proposed method has the potential to reuse more intermediate results). However, the algorithm also becomes more computationally demanding, making it not suitable to apply to large scale networks. Developing an efficient, approximate algorithm is the topic of our ongoing research work, and a polynomial-time heuristic algorithm will be reported in [14].

VII. Numerical Results

We perform the numerical experiments based on the continental US Abilene network, as is shown in Fig. 3. The processing capability of each node is $C_i = 20$ CPUs, and the processing cost is $e_i = 0.5$ /CPU per second. The cloud network links exhibit homogeneous transmission capabilities and costs, given by $C_{ij} = 10$ Gbps, and $e_{ij} = 1$ /Gb. We set the length of each time slot as $\tau = 1$ ms, and unify the size of each packet as $F = 1$ kb.

Two services are provided by the cloud network, each consisting of 2 functions, with the following parameters

$$\phi_1: \ \xi_1^{(1)} = 1, \ \xi_1^{(2)} = 2; \ 1/r_1^{(1)} = 300, \ 1/r_1^{(2)} = 400$$
$$\phi_2: \ \xi_2^{(1)} = \frac{1}{3}, \ \xi_2^{(2)} = \frac{1}{2}; \ 1/r_2^{(1)} = 200, \ 1/r_2^{(2)} = 100$$

where $1/r_\phi^{(m)}$ [Mbps/CPU] denotes the supportable input size given 1 CPU resource.

We consider any destination set $\mathcal{D}$ consisting of two nodes selected from $\{7, 8, 9, 10, 11\}$ (e.g., $\{7, 10\}$), and hence there are 10 possible destination sets in total. Each destination set can request both services ϕ_1 and ϕ_2, which originate from any source node in $\{1, 2, 3, 4\}$. The packets of commodity $(c, q) = (\phi_i, 1, \mathcal{D}, \mathbf{1})$ $(i = 1, 2)$ arrive at each source node, and it is modeled by i.i.d. Poisson process, independent of each other, with parameter λ.

We employ the **simple approach** (see Section VI-D2) as the baseline for comparison, i.e., treating data-streams for different destination nodes as separated unicast flows. A more comprehensive comparison of the proposed approach with existing multicast techniques will be reported in [14].

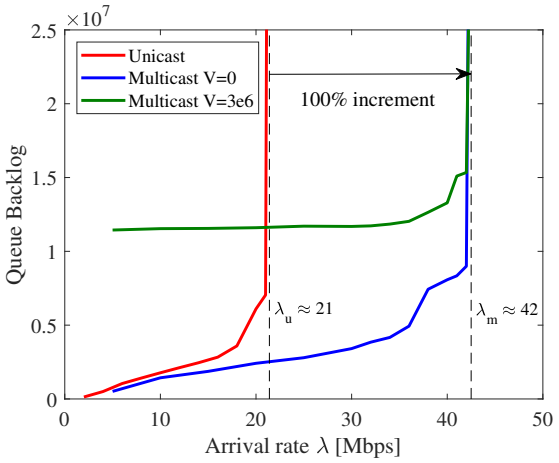


Fig. 4. The capacity region achieved by the multicast algorithms (with $V = 0$ and $V = 3 \times 10^6$), as well as the unicast-based solution.

A. Capacity Region

We first study the capacity region of the cloud network with multicast flow (using the proposed algorithm), compared with the achieved capacity region by treating the problem as separate unicast problems. The initial queue backlog is set as $\boldsymbol{Q}(0) = \mathbf{0}$, and we observe the system for 10^6 time slots. The stable queue backlogs are recorded under various λ values. If the queue keeps growing at the end of the period, the stable queue length is set as ∞.

[3]This is determined by the combinatorial nature of the multicast problem. Another solution to the multicast problem provided by [11] requires to solve the minimum Steiner tree problem to determine the route for each packet, which is a NP-complete problem.

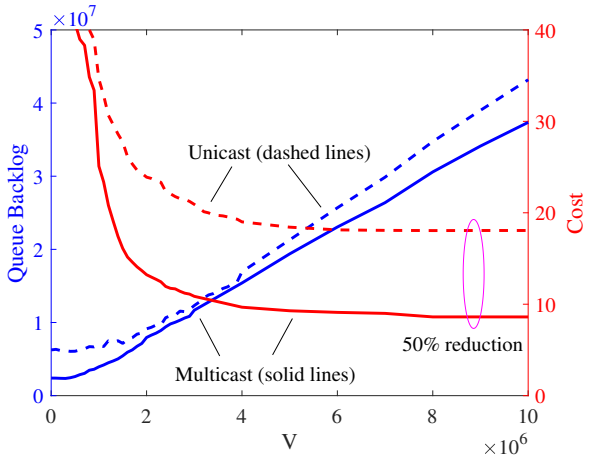


Fig. 5. The queue backlog and cost performance of the proposed algorithm under various values of V.

The results is shown in Fig. 4. It is obvious that the queue backlog grows monotonously with the arrival rate for all the three cases. Then we focus on the queue backlog performance of the proposed algorithm under various values of V. We find that a larger value of V results in a heavier queue backlog; however, the two values $V = 0$ and $V = 3 \times 10^6$ lead to an identical critical point $\lambda_{\text{m}} \approx 42$ Mbps, which can be interpreted as the boundary of the capacity region. The result validates the conclusion that the proposed algorithm, using any fixed value of V, always achieves finite queue backlog within the capacity region, and therefore is throughput-optimal. Finally, we compare the capacity regions achieved by the proposed algorithm with the unicast-based solution, which is $\lambda_{\text{u}} \approx 21$ Mbps. An increment of 100% is gained, by making smart duplication decision, which reuses some intermediate results to fully exploit the available resource.

B. Delay-Cost Tradeoff

Next, we study the queue backlog, as well as the cost performance of the proposed algorithm under various V. The arrival rate is selected as $\lambda = 20$ Mbps. The results are compared with the unicast-based solution.

The results are depicted in Fig. 5. Visually, it exhibits a $[\mathcal{O}(V), \mathcal{O}(1/V)]$ tradeoff between the queue backlog and the resource cost, as is established in (22) and (23). Considering the decreasing rate, we anticipate the optimal cost of the proposed algorithm to be 8, which reduces by 50% when comparing with the optimal cost 17 achieved by the unicast-based solution. Again, the reduction is thanks to the reuse gain as is explained in the previous experiment. A larger gain can be expected for a destination set with more nodes, but this comes at the price of increasing the algorithm complexity.

VIII. Conclusions

In this paper, we investigated the problem of cloud network control in the presence of multicast flows. We proposed a queueing system that allows flow-level (rather than packet-wise) decision making, and presented an efficient policy space that is cost-optimal. The characterization of the new capacity region was presented, and we developed a fully distributed control algorithm guided by Lyapunov optimization theory. Numerical results showed the performance gain of the proposed algorithm over the unicast-based solution, in terms of the capacity region and the achieved resource cost.

IX. Acknowledgments

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

References

[1] A. B. Craig, *Understanding augmented reality: concepts and applications*. Newnes, 2013.

[2] M. Weldon, *The future X network: a Bell Labs perspective*. CRC Press, 2016.

[3] H. Feng, J. Llorca, A. M. Tulino, and A. F. Molisch, "Optimal control of wireless computing networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 12, pp. 8283–8298, Dec. 2018.

[4] 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.

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

[6] D. Bhamare, R. Jain, M. Samaka, and A. Erbad, "A survey on service function chaining," *Journal of Network and Computer Applications*, vol. 75, no. 1, pp. 138–155, Nov. 2016.

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

[8] 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.

[9] Y. Cai, J. Llorca, A. M. Tulino, and A. F. Molisch, "Optimal cloud network control with per-packet deadline constraint," in *Proc. IEEE Int. Conf. Commun.*, Montreal, Canada, Jun. 2021, pp. 1–6.

[10] 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.

[11] 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.

[12] 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.

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

[14] Y. Cai, J. Llorca, A. M. Tulino, and A. F. Molisch, "Optimal dynamic cloud network control with generalized network flows," *to be submitted*.