Dynamic Network Service Optimization in Distributed Cloud Networks

Hao Feng*†, Jaime Llorca†, Antonia M. Tulino†, Andreas F. Molisch*
*University of Southern California, CA. Email: {haofeng, molisch}@usc.edu
†Bell Labs, Nokia, NJ. Email: {jaime.llorca, a.tulino}@nokia.com

***Abstract*—Distributed cloud networking enables the deployment of network services in the form of interconnected virtual network functions instantiated over general purpose hardware at multiple cloud locations distributed across the network. The service distribution problem has been previously considered as a static global optimization problem of finding the placement of virtual functions and the routing of network flows that meet a given set of demands with minimum cost. In this paper, motivated by the scale, dynamics, and heterogeneity of network services in distributed cloud networks, we address the design of distributed online solutions that provide global objective guarantees via local interactions without knowledge of the statistics of service demands. We characterize the cloud network capacity region and propose a distributed joint flow scheduling and resource allocation algorithm that stabilizes the underlying queuing system within this region, while achieving arbitrarily close to minimum average network cost, with a tradeoff in network delay. Numerical results confirm our theoretical analysis and demonstrate remarkably good convergence to the optimal cloud network configuration within a number of network settings.**

I. Introduction

Distributed cloud networking is a new networking paradigm that builds on network functions virtualization (NFV) and software defined networking (SDN) to enable the deployment of network services in the form of elastic virtual network functions instantiated over commercial off the shelf (COTS) servers at multiple cloud locations and interconnected via a programmable network fabric [1], [2]. In this evolved programmable virtualized environment, network operators can host a variety of highly adaptable services over a common physical infrastructure, reducing both capital and operational expenses, while providing quality of service guarantees.

Together with the evident opportunities of this attractive scenario, there also come a number of technical challenges. A critical aspect that drives both cost and performance is the actual placement of the network services' virtual functions. The ample opportunities for running network functions at multiple locations opens an interesting and challenging space for optimization. In addition, placement decisions must be accompanied with routing decisions that steer the network flows to the appropriate network functions, and with resource allocation decisions that determine the amount of resources (*e.g.,* virtual machines) allocated to each function.

Recent works have addressed the service distribution problem from a static global optimization point of view [3], [4]. In this setting, the goal is to find the placement of virtual functions and the routing of network flows that meet a given set of demands with minimum cost. However, the complexity associated with the resulting global optimization problem and the need to have global knowledge of the service demands, limits the use of centralized algorithms, specially in large-scale distributed cloud networks and under time-varying demands. With the increasing scale, heterogeneity, and dynamics inherent to both service demands and the underlying cloud network, we argue that proactive centralized solutions, while useful for relatively small-scale services with relatively static demands, must be complemented with distributed online algorithms that enable rapid adaptation to changes in network conditions and service demands, while providing global system objective guarantees.

In this work, we leverage recent advances in dynamic network control to design efficient distributed online solutions that drive local routing, processing, and resource allocation decisions with global system guarantees. To this end, we extend the dynamic network control *Lyapunov drift-plus-penalty* [5] algorithm to account for both transmission and processing of network flows and the corresponding allocation of network and cloud resources. We first characterize the cloud network capacity region and then propose a distributed joint flow scheduling and resource allocation algorithm that stabilizes the underlying queuing system within this region, while achieving arbitrarily close to minimum average cloud network cost, with a tradeoff in network delay.

To the best of our knowledge, this is the first work to: *i)* provide a queueing model for cloud networks that captures, not only routing, but also processing of network flows by the required network services; *ii)* characterize the capacity region of a cloud network, in terms of the set of input flow rates that can be stably processed by the required network services and delivered to the required destinations; *iii)* design a distributed online algorithm for the network service distribution problem, that drives local routing, processing, and resource scaling decisions, while minimizing the average cloud network cost.

II. Related Work

The problem of placing virtual network functions in distributed cloud networks was recently addressed in [3]. The VNF placement problem is shown to be a generalization of Generalized Assignment (GA) and Facility Location (FA), and hence NP-Hard. The authors in [3] provide a $(O(1), O(1))$ bi-criteria approximation with respect to both overall cost and capacity constraints. The work in [4] introduced the cloud

service distribution problem (CSDP), where the goal is to find the placement of cloud service functions (*e.g.,* VNFs) and the routing of network flows that minimize the overall infrastructure cost. The authors in [4] formulate the CSDP as a minimum cost network flow problem on a properly augmented graph, in which flows consume both network and cloud resources as they go through the required virtual functions. Interestingly, while NP-Hard in general, the CSDP is shown to admit a polynomial-time linear programming solution under linear costs and fractional flows. In terms of dynamic network control, the Lyapunov drift-plus-penalty algorithm was shown to stabilize any set of input flow rates within the capacity region of *traditional* (transmission) networks while minimizing the average network cost [5]-[7]. To the best of our knowledge, our work is the first to address the service distribution problem in a *dynamic* cloud network setting, in which demands are assumed to be unknown and time-varying, extending the Lyapunov drift-plus-penalty control algorithm to account for both transmission and processing flows consuming network and cloud resources.

III. System Model

A. Cloud network model

We consider a cloud network modeled as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $|\mathcal{V}| = N$ vertices and $|\mathcal{E}| = E$ edges representing the set of network nodes and links, respectively. In the context of a cloud network, a node represents a distributed cloud location, in which virtual network functions or VNFs can be instantiated in the the form of virtual machines (VMs) over COTS servers, while an edge represents a logical link (*e.g.,* IP link) between two cloud locations. We denote by $\delta(i) \in \mathcal{V}$ the set of neighbor nodes of $i \in \mathcal{V}$ in $\mathcal{G}$.

B. Network service model

A network service $\phi \in \Phi$ is described by a chain of VNFs. We denote by $\mathcal{M}_\phi = \{1, 2, \cdots, M_\phi\}$ the ordered set of VNFs of service ϕ. Hence, the tuple (ϕ, m), with $\phi \in \Phi$ and $m \in \mathcal{M}_\phi$, identifies the m-th function of service ϕ. We refer to a client as a source-destination pair (s, d), with $s, d \in \mathcal{V}$. A client requesting network service ϕ implies the request for the network flows originating at source node s to go through the sequence of VNFs specified by $\mathcal{M}_\phi$ before exiting the network at destination node d. As in [4], we adopt a *multi-commodity-chain* flow model, in which a commodity represents a network flow at a given stage of a service chain. We use the triplet (d, ϕ, m) to identify a commodity flow that is output of the m-th function of service ϕ for the destination node d, with $(d, \phi, 0)$ denoting the source commodity of service ϕ for destination d, as illustrated in Fig. 1.

Each VNF has (possibly) different processing requirements, which may also vary between cloud locations. We denote by $r^{(\phi,m)}$ the processing-transmission flow ratio of VNF (ϕ, m). That is, when a transmission flow unit goes through VNF (ϕ, m), it occupies $r^{(\phi,m)}$ processing flow units. As will be shown in Section III-C, the capacity of a cloud resource unit (e.g., virtual machine) is specified in terms of processing flow units. In the following, unless specified, a flow unit refers to a transmission flow unit.

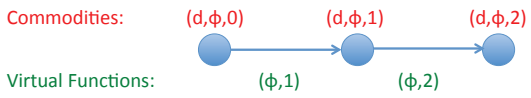

Fig. 1. A network service chain $\phi \in \Phi$ composed of $M_\phi = 2$ functions. Service ϕ takes source commodity $(d, \phi, 0)$ and delivers final commodity $(d, \phi, 2)$ after going through the sequence of functions $\{(\phi, 1), (\phi, 2)\}$. VNF (ϕ, m) takes commodity $(d, \phi, m-1)$ and generates commodity (d, ϕ, m).

In addition, our service model also captures the possibility of flow scaling. We denote by $\xi^{(\phi,m)} > 0$ the scaling factor of VNF $m \in \mathcal{M}_\phi$. That is, the size of the output flow of VNF (ϕ, m) is $\xi^{(\phi,m)}$ times as large as its input flow. We refer to a VNF with $\xi^{(\phi,m)} > 1$ as an expansion function, and to a VNF with $\xi^{(\phi,m)} < 1$ as a compression function.

Moreover, a processing delay $D_i^{(\phi,m)}$ (in time units) is incurred in executing VNF (ϕ, m) at node i, as long as the processing flow satisfies the node capacity constraint.

We remark that our service model applies to a wide range of cloud services that go beyond NFV services, and that include, for example, IoT services, expected to largely benefit from the proximity and elasticity of distributed cloud networks [8].

C. Cost model

We adopt a flexible and general cost model that includes setup and usage related costs for both cloud and network resources. Accordingly, we define the following parameters:

- $\mathcal{K}_i = \{0, 1, \cdots, K_i\}$: the set of possible processing resource units at cloud node i
- $\mathcal{K}_{ij} = \{0, 1, \cdots, K_{ij}\}$: the set of possible transmission resource units at network link (i, j)
- $C_{i,k}$: the capacity, in processing flow units, resulting from the allocation of k resource units (*e.g.,* VMs) at node i
- $C_{ij,k}$: the capacity, in transmission flow units, resulting from the allocation of k resource units (*e.g.,* 1G links) at link (i, j)
- $w_{i,k}$: the cost of setting up k resource units at node i
- $w_{ij,k}$: the cost of setting up k resource units at link (i, j)
- e_i: the cost per processing flow unit at node i
- e_{ij}: the cost per transmission flow unit at link (i, j)

IV. Dynamic Service Distribution Problem

In this section, we formulate the dynamic service distribution problem (DSDP) under a cloud network queuing model in which demands may be unknown and time-varying.

We consider a time slotted system with slots normalized to integral units $t \in \{0, 1, 2, \cdots\}$. We denote by $a_i^{(d,\phi,m)}(t)$ the exogenous arrival rate of commodity (d, ϕ, m) at node i during timeslot t, and by $\lambda_i^{(d,\phi,m)}$ the expected value (or time average value) of $a_i^{(d,\phi,m)}(t)$, referred to as average input (or demand) rate. We assume $a_i^{(d,\phi,m)}(t)$ is independently and identically distributed (i.i.d.) across timeslots.

At each timeslot t, every node makes a (transmission or processing) flow scheduling decision on each of its output interfaces. We use $\mu_{ij}^{(d,\phi,m)}(t)$ to denote the assigned flow rate at link (i,j) for commodity (d,ϕ,m) at time t, $\mu_{i,pr}^{(d,\phi,m)}(t)$ to denote the assigned flow rate from node i to its processing unit for commodity (d,ϕ,m) at time t, and $\mu_{pr,i}^{(d,\phi,m)}(t)$ to denote the assigned flow rate from node i's processing unit to node i for commodity (d,ϕ,m) at time t.

During network evolution, internal network queues buffer packets according to their commodities. We define the *queue backlog* of commodity (d,ϕ,m) at node i, $Q_i^{(d,\phi,m)}(t)$, as the amount of commodity (d,ϕ,m) in the queue of node i at the beginning of timeslot t, which evolves over time as follows:[1]

$$Q_i^{(d,\phi,m)}(t+1) \leq \left[Q_i^{(d,\phi,m)}(t) - \sum_{j\in\delta(i)} \mu_{ij}^{(d,\phi,m)}(t) - \mu_{i,pr}^{(d,\phi,m)}(t) \right]^+ + \sum_{j\in\delta(i)} \mu_{ji}^{(d,\phi,m)}(t) + \mu_{pr,i}^{(d,\phi,m)}(t) + a_i^{(d,\phi,m)}(t). \quad (1)$$

In addition, at each timeslot t, cloud network nodes can also make resource allocation decisions. We denote by $y_{ij,k}(t)$ the binary variable indicating the allocation of k transmission resource units at link (i,j) at time t, and by $y_{i,k}(t)$ the binary variable indicating the allocation of k processing resource units at node i at time t.

The goal is to design a control algorithm that, given exogenous arrival rates with average input rate matrix $\boldsymbol{\lambda} = (\lambda_i^{(d,\phi,m)})$, supports all service demands while minimizing the average cloud network cost. Specifically, we require that the cloud network be rate stable, *i.e.,*

$$\lim_{t\to\infty} \frac{Q_i^{(d,\phi,m)}(t)}{t} = 0 \quad \text{with prob. 1} \quad \forall i,(d,\phi,m). \quad (2)$$

The dynamic service distribution problem (DSDP) can then be formulated as follows:

$$\min \quad \limsup_{t\to\infty} \quad \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\{h(\tau)\} \quad (3a)$$

s.t. The cloud network is rate stable with input rate $\boldsymbol{\lambda}$ (3b)

$$\mu_{pr,i}^{(d,\phi,m)}(\tau) = \xi^{(\phi,m)} \mu_{i,pr}^{(d,\phi,m-1)}(\tau - D_i^{(\phi,m)}) \quad \forall i,d,\phi,m>0,\tau \quad (3c)$$

$$\sum_{(d,\phi,m>0)} \mu_{i,pr}^{(d,\phi,m-1)}(\tau) r^{(\phi,m)} \leq \mu_i(\tau) \leq \sum_{k\in\mathcal{K}_i} C_{i,k}\, y_{i,k}(\tau) \quad \forall i,\tau \quad (3d)$$

$$\sum_{(d,\phi,m)} \mu_{ij}^{(d,\phi,m)}(\tau) \leq \mu_{ij}(\tau) \leq \sum_{k\in\mathcal{K}_{ij}} C_{ij,k}\, y_{ij,k}(\tau) \quad \forall (i,j),\tau \quad (3e)$$

$$\mu_{i,pr}^{(d,\phi,m)}(\tau), \mu_{pr,i}^{(d,\phi,m)}(\tau), \mu_{ij}^{(d,\phi,m)}(\tau) \in \mathbb{R}^+ \quad \forall i,(i,j),d,\phi,m,\tau \quad (3f)$$

$$y_{i,k}(\tau), y_{ij,k}(\tau) \in \{0,1\} \quad \forall i,(i,j),d,\phi,m,\tau \quad (3g)$$

[1]Throughout the paper, we use $[x]^+$ to denote $\max\{x,0\}$.

In (3), $h(\tau)$ is the total cloud network cost at timeslot τ:

$$h(\tau) = \sum_{i\in\mathcal{V}} \left(e_i\, \mu_i(\tau) + \sum_{k\in\mathcal{K}_i} w_{i,k}\, y_{i,k}(\tau) \right) + \sum_{(i,j)\in\mathcal{E}} \left(e_{ij}\, \mu_{ij}(\tau) + \sum_{k\in\mathcal{K}_{ij}} w_{ij,k}\, y_{ij,k}(\tau) \right), \quad (4)$$

which includes processing and transmission usage and setup costs; (3c) are the multi-commodity-chain constraints; and (3d)–(3e) the cloud network capacity constraints.

V. Dynamic flow scheduling and resource allocation algorithm

We propose a dynamic cloud network control strategy that solves the DSDP given by (3) in a fully distributed manner by extending the Lyapunov drift-plus-penalty algorithm to account for both transmission and processing flow scheduling and resource allocation decisions.

At each timeslot t, given the current cloud network *backlog state* $\left\{Q_i^{(d,\phi,m)}(t)\right\}$, the goal of the algorithm is:

$$\max \quad \sum_{i\in\mathcal{V}} \left(\sum_{j\in\delta(i)} Z_{ij}^{tr}(t) + Z_i^{pr}(t) - V h_i(t) \right) \quad (5a)$$

$$\text{s.t.} \quad (3d) - (3g) \quad (5b)$$

where,

$$Z_{ij}^{tr}(t) = \sum_{(d,\phi,m)} \mu_{ij}^{(d,\phi,m)}(t) \left[Q_i^{(d,\phi,m)}(t) - Q_j^{(d,\phi,m)}(t) \right],$$

$$Z_i^{pr}(t) = \sum_{(d,\phi,m)} \mu_{i,pr}^{(d,\phi,m)}(t) \left[Q_i^{(d,\phi,m)}(t) - \xi^{(\phi,m+1)} Q_i^{(d,\phi,m+1)}(t) \right],$$

$$h_i(t) = \sum_{j\in\delta(i)} \left(e_i \sum_{(d,\phi,m)} \mu_{i,pr}^{(d,\phi,m)}(t) + \sum_{k\in\mathcal{K}_{ij}} w_{i,k} y_{i,k}(t) \right) + \left(e_{ij} \sum_{(d,\phi,m)} \mu_{ij}^{(d,\phi,m)}(t) + \sum_{k\in\mathcal{K}_i} w_{ij,k} y_{ij,k}(t) \right),$$

and where V is a non-negative control parameter that determines the degree to which cost minimization is emphasized.

Observe that maximizing (5a) is equivalent to pushing the queue backlogs towards a low congestion state while greedily minimizing the cloud network cost via a weighted sum regulated by the control parameter V. It is immediate to see that (5) can be implemented in a distributed fashion, leading to the following algorithm:

Dynamic Cloud Network Control (DCNC):

Local transmission decisions: At the beginning of each timeslot t, each node i observes the queue backlogs of all its neighbors and performs the following operations for each of its outgoing links (i,j), $j\in\delta(i)$:

1) For each commodity (d,ϕ,m), compute the *transmission utility* weights

$$W_{ij}^{(d,\phi,m)} = \left[Q_i^{(d,\phi,m)}(t) - Q_j^{(d,\phi,m)}(t) - V e_{ij} \right]^+$$

2) Compute the optimal commodity $(d,\phi,m)^*$ as:

$$(d,\phi,m)^* = \arg\max_{(d,\phi,m)} \left\{ W_{ij}^{(d,\phi,m)} \right\}$$

3) If $W_{ij}^{(d,\phi,m)^*} = 0$, then, $k^* = 0$. Otherwise,

$$k^* = \arg\max_{k} \left\{ C_{ij,k} W_{ij}^{(d,\phi,m)^*} - V w_{ij,k} \right\}$$

4) Take the following resource allocation and flow rate assignment decisions:

$$\begin{aligned} &y_{ij,k^*}(t) = 1 \\ &y_{ij,k}(t) = 0 \quad \forall k \neq k^* \\ &\mu_{ij}^{(d,\phi,m)^*}(t) = C_{ij,k^*} \\ &\mu_{ij}^{(d,\phi,m)}(t) = 0 \quad \forall (d,\phi,m) \neq (d,\phi,m)^* \end{aligned}$$

Local processing decisions: At the beginning of each timeslot t, each node i observes its local queue backlogs and performs the following operations:

1) For each commodity (d,ϕ,m), compute the *processing utility* weights

$$W_i^{(d,\phi,m)}(t) = \frac{1}{r^{(\phi,m+1)}} \left[Q_i^{(d,\phi,m)}(t) - \xi^{(\phi,m+1)} Q_i^{(d,\phi,m+1)}(t) - V e_i \right]^+$$

The *processing utility* weight $W_i^{(d,\phi,m)}(t)$ is a novel and key aspect of DCNC. Specifically, $W_i^{(d,\phi,m)}(t)$ indicates the benefit of executing function $(\phi, m+1)$ to process commodity (d,ϕ,m) into commodity $(d,\phi,m+1)$ at time t, in terms of the local backlog reduction per processing unit cost. Observe that a large value of $\xi^{(\phi,m+1)}$ reduces the benefit of processing commodity (d,ϕ,m) due to the impact of the $\xi^{(\phi,m+1)}$ flow units of commodity $(d,\phi,m+1)$ resulting from the processing of one unit of commodity (d,ϕ,m), on the overall local queue backlog. A large value of $r^{(\phi,m+1)}$ also reduces the utility weight, since a higher processing requirement leads to higher processing cost and/or slower backlog reduction of commodity (d,ϕ,m).

2) Compute the optimal commodity $(d,\phi,m)^*$ as:

$$(d,\phi,m)^* = \arg\max_{(d,\phi,m)} \left\{ W_i^{(d,\phi,m)} \right\}$$

3) If $W_i^{(d,\phi,m)^*} = 0$, then, $k^* = 0$. Otherwise,

$$k^* = \arg\max_{k} \left\{ C_{i,k} W_i^{(d,\phi,m)^*} - V w_{i,k} \right\}$$

4) Take the following resource allocation and flow rate assignment decisions:

$$\begin{aligned} &y_{i,k^*}(t) = 1 \\ &y_{i,k}(t) = 0 \quad \forall k \neq k^* \\ &\mu_{i,pr}^{(d,\phi,m)^*}(t) = C_{i,k^*} \Big/ r^{(\phi,m+1)^*} \\ &\mu_{i,pr}^{(d,\phi,m)}(t) = 0 \quad \forall (d,\phi,m) \neq (d,\phi,m)^* \end{aligned}$$

Note from the above algorithm description that the finite processing delay $D_i^{(\phi,m)}$ is not involved in the scheduling decisions of DCNC. In fact, as shown in the proof of Theorem 2 in Section VI-B, finite processing delays do not affect average cost convergence or throughput optimality.

VI. Performance Analysis

In this section, we analyze the performance of DCNC by characterizing the cloud network capacity region and proving the ability of DCNC to stablize any set of input rates within this region with arbitrarily close to minimum cost. In the following, we assume that $\xi^{(\phi,m)}$, $r^{(\phi,m)}$, $D_i^{(\phi,m)}$, and $\sum_{(d,\phi,m)} \mathbb{E}[(a_i^{(d,\phi,m)}(t))^2]$ are bounded.

A. Cloud network capacity region

The cloud network capacity region Λ is defined as the closure of all input rate matrices $(\lambda_i^{(d,\phi,m)})$ that can be stabilized by some control algorithm conforming to the given cloud network structure $\{\mathcal{G}, \Phi\}$.

Theorem 1. *The cloud network capacity region Λ consists of all average exogenous input rates $(\lambda_i^{(d,\phi,m)})$ for which, for all i,j,k,d,ϕ,m, there exist multi-commodity flow variables $f_{ij}^{(d,\phi,m)}$, $f_{pr,i}^{(d,\phi,m)}$, $f_{i,pr}^{(d,\phi,m)}$, together with probability values $\alpha_{ij,k}$, $\alpha_{i,k}$, $\beta_{ij,k}^{(d,\phi,m)}$, $\beta_{i,k}^{(d,\phi,m)}$, such that*

$$\sum_{j\in\delta(i)} f_{ji}^{(d,\phi,m)} + f_{pr,i}^{(d,\phi,m)} + \lambda_i^{(d,\phi,m)} \leq \sum_{j\in\delta(i)} f_{ij}^{(d,\phi,m)} + f_{i,pr}^{(d,\phi,m)} \tag{6}$$

$$f_{pr,i}^{(d,\phi,m)} = \xi_i^{(\phi,m)} f_{i,pr}^{(d,\phi,m-1)} \qquad (m>0) \tag{7}$$

$$f_{i,pr}^{(d,\phi,m-1)} \leq \frac{1}{r^{(\phi,m)}} \sum_{k\in\mathcal{K}_i} \alpha_{i,k} \beta_{i,k}^{(d,\phi,m-1)} C_{i,k} \quad (m>0) \tag{8}$$

$$f_{ij}^{(d,\phi,m)} \leq \sum_{k\in\mathcal{K}_{ij}} \alpha_{ij,k} \beta_{ij,k}^{(d,\phi,m)} C_{ij,k} \tag{9}$$

$$f_{i,pr}^{(d,\phi,M_\phi)} = 0,\ f_{pr,i}^{(d,\phi,0)} = 0,\ f_{dj}^{(d,\phi,M_\phi)} = 0 \tag{10}$$

$$f_{i,pr}^{(d,\phi,m)} \geq 0,\ f_{ij}^{(d,\phi,m)} \geq 0 \tag{11}$$

$$\sum\nolimits_{k\in\mathcal{K}_{ij}} \alpha_{ij,k} \leq 1, \sum\nolimits_{k\in\mathcal{K}_i} \alpha_{i,k} \leq 1 \tag{12}$$

$$\sum\nolimits_{(d,\phi,m)} \beta_{ij,k}^{(d,\phi,m)} \leq 1, \sum\nolimits_{(d,\phi,m)} \beta_{i,k}^{(d,\phi,m)} \leq 1 \tag{13}$$

Furthermore, the minimum average cloud network cost required for network stability is given by

$$\bar{h}^* = \min \underline{h}\ , \tag{14}$$

$$\begin{aligned} \underline{h} = &\sum_{(i,j)} \sum_{k\in\mathcal{K}_{ij}} \alpha_{ij,k} \left(w_{ij,k} + e_{ij} C_{ij,k} \sum_{(d,\phi,m)} \beta_{ij,k}^{(d,\phi,m)} \right) \\ &+ \sum_{i} \sum_{k\in\mathcal{K}_i} \alpha_{i,k} \left(w_{i,k} + e_i C_{i,k} \sum_{(d,\phi)} \sum_{m=0}^{M_\phi - 1} \frac{\beta_{i,k}^{(d,\phi,m)}}{r^{(\phi,m+1)}} \right), \end{aligned} \tag{15}$$

where the minimization is over all $f_{ij}^{(d,\phi,m)}$, $f_{i,pr}^{(d,\phi,m)}$, $\alpha_{ij,k}$, $\alpha_{i,k}$, $\beta_{ij,k}^{(d,\phi,m)}$, $\beta_{i,k}^{(d,\phi,m)}$ satisfying Eqs. (6)-(13).

□

In the above theorem,(6) are flow conservation constraints, (8) and (9) are capacity constraints, and (10) and (11) are non-negativity and flow efficiency constraints. The probability values $\alpha_{ij,k}$, $\alpha_{i,k}$, $\beta_{ij,k}^{(d,\phi,m)}$, and $\beta_{i,k}^{(d,\phi,m)}$ define a stationary randomized policy, where:

- $\alpha_{ij,k}$: the probability that k transmission resource units are allocated to link (i,j)
- $\alpha_{i,k}$: the probability that k processing resource units are allocated at node i
- $\beta_{ij,k}^{(d,\phi,m)}$: the probability that link (i,j) transmits commodity (d,ϕ,m), conditioned to the allocation of k transmission resource units at link (i,j)
- $\beta_{i,k}^{(d,\phi,m)}$: the probability that node i processes commodity (d,ϕ,m), conditioned to the allocation of k processing resource units at node i

Proof: The proof of Theorem 1 is omitted here due to space limitations and can be found in [9]. ■

Theorem 1 demonstrates that, for any input rate matrix $(\lambda_i^{(d,\phi,m)}) \in \Lambda$, there exists a stationary randomized policy that can achieve a multi-commodity flow that supports the input rate matrix by routing all the commodities to their proper destinations via the proper network functions, and that incurs an average cost exactly given by (14). However, the difficulty in directly solving for the parameters that characterize such a stationary randomized policy and the requirement on the knowledge of $(\lambda_i^{(d,\phi,m)})$ motivate the design of DCNC, whose optimality is stated in the next theorem.

B. Optimality of DCNC

Theorem 2. *If the rate matrix* $\boldsymbol{\lambda} \triangleq (\lambda_i^{(d,\phi,m)})$ *is strictly interior to the capacity region* Λ*, then DCNC stabilizes the cloud network, while achieving arbitrarily close to minimum average cost* $\overline{h}^*(\boldsymbol{\lambda})$*; i.e.,*

$$\limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t} \mathbb{E}\left\{h(\tau)\right\} \leq \overline{h}^*(\boldsymbol{\lambda}) + \frac{NB}{V} \tag{16}$$

$$\limsup_{t\to\infty} \frac{1}{t} \sum_{\tau,i,d,\phi,m} \mathbb{E}\left\{Q_i^{(d,\phi,m)}(\tau)\right\} \leq \frac{NB+V\left[\overline{h}^*(\boldsymbol{\lambda}+\epsilon\mathbf{1})-\overline{h}^*(\boldsymbol{\lambda})\right]}{\epsilon}$$

where B *is a constant depending on the system parameters* $C_{ij,K_{ij}}$, C_{i,K_i}, $A_{\max}$, $\xi^{(\phi,m)}$, $D_i^{(\phi,m)}$ *and* $r^{(\phi,m)}$*;* ε *is a positive constant satisfying* $(\boldsymbol{\lambda}+\varepsilon) \in \Lambda$*; and, with a slight abuse of notation,* $\overline{h}^*(\boldsymbol{\lambda})$ *denotes the minimum average cost of the DSDP formulated in* (3)*.*

□

Note that the parameter V drives the average cost arbitrarily close to the minimum cost $\overline{h}^*(\boldsymbol{\lambda})$ required for network stability, with a corresponding linear increase in average network congestion (average delay performance).

Proof: The proof of Theorem 2 is given in [9]. ■

VII. Numerical Experiments

In this section, we present numerical results obtained from simulating the proposed dynamic cloud network control (DCNC) algorithm during 10^6 timeslots.

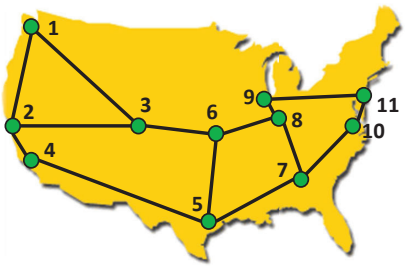

Fig. 2. Abilene US Continental Network. Nodes are indexed as: 1) Seattle, 2) Sunnyvale, 3) Denver, 4) Los Angeles, 5) Houston, 6) Kansas City, 7) Atlanta, 8) Indianapolis, 9) Chicago, 10) Washington, 11) New York.

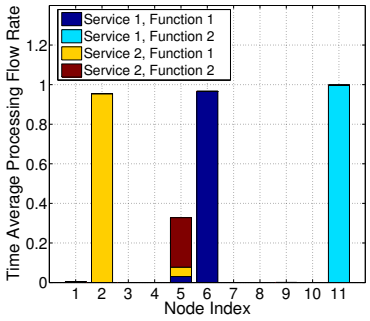

Fig. 3. Processing flow rate distribution over all cloud nodes.

We assume a cloud network based on the Abilene US continental network, shown in Fig. 2. All 14 links exhibit homogeneous capacities and costs. In particular, all links have two capacity choices $C_{ij,0} = 0$ and $C_{ij,1} = 1$, with corresponding resource allocation costs $w_{ij,0} = 0$ and $w_{ij,1} = 1$, and load-dependent costs $e_{ij} = 1$. All 11 nodes represent cloud locations with homogeneous capacity choices, $C_{i,0} = 0$ and $C_{i,1} = 1$, and resource allocation costs, $w_{i,0} = 0$ and $w_{i,1} = 0$. However, cloud nodes have different load-dependent processing costs. In particular, we assume Kansas City (node 5) and Houston (node 6) have cheaper processing costs, $e_5 = e_6 = 1$, while for any other node $e_i = 3$.

We consider two services, each composed of two virtual functions. All 4 functions have the same complexity, given by a processing-transmission flow ratio of 1 and a processing delay of 10 timeslots. In terms of flow scaling, the first and second functions of Service 1 have a scaling factor of 1 and 3, respectively. That is, the second function of Service 1 is an expansion function. For Service 2, the first and second functions have a scaling factor of 0.25 and 1, respectively. That is, the first function of Service 1 is a compression function. We assume one client or source-destination pair for Service 1, with source in Seattle (node 1) and destination in New York (node 11), and another for Service 2, with source in Sunnyvale (node 2) and destination in Atlanta (node 7). Both source nodes receive exogenous arrivals with rate satisfying i.i.d. Poisson distribution across timeslots with mean value 1. Every node in the network has the ability to implement all the functions of both services.

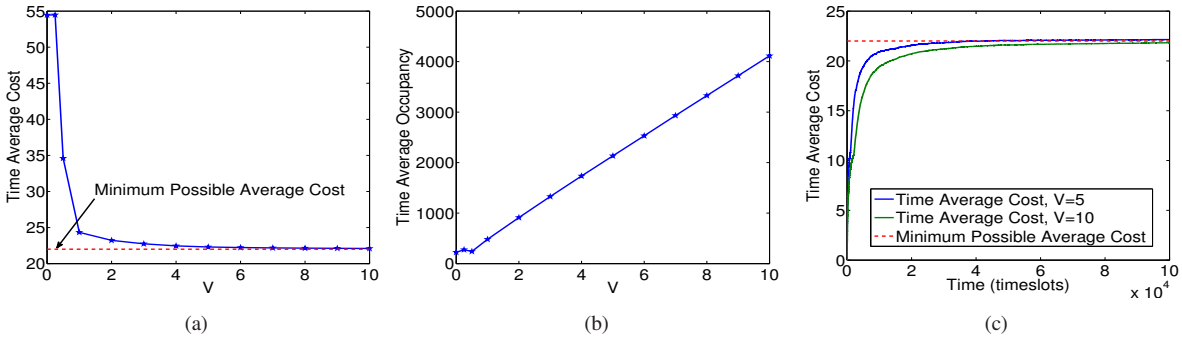

Fig. 4. Network service distribution over the Abilene US Continental Network. a) Time average cost v.s. control parameter V; b) Time average total backlog (occupancy) vs. control parameter V; c) Time average cost evolution over time.

Fig. 3 shows the processing flow rate distribution for the four functions across the cloud network nodes. It can be observed that VNF $(1,1)$ (first function of service 1) is mostly implemented in node 6, which is the node with lowest processing cost along the shortest path from node 1 to node 11. Note, however, that the expansion function, VNF $(1,2)$, is largely implemented in node 11, the final destination of Service 1, in order to minimize the transmission cost impact of the larger final commodity $(11,1,2)$ resulting from the execution of VNF $(1,2)$ for destination 11. For Service 2, the first function, VNF $(2,1)$, is a compression function with scaling factor 0.25. As expected, VNF $(2,1)$ is mostly implemented at the source node (node 2), in order to reduce the size of the source commodity $(7,2,0)$ before even flowing into the network. The processing of commodity $(7,2,1)$ by VNF $(2,2)$ to generate commodity $(7,2,2)$ concentrates at node 5 because this node has the lowest processing cost among the nodes on the shortest path from node 2 to node 7. Note how the average flow rate of commodity $(7,2,1)$ is approximately 0.25 due to the effect of the compression function $(2,1)$.

With the simulated average demand rate of 1 for both services, the minimum average cloud network cost can be computed by inspection. Specifically, the optimal flow paths for the two services are the two respective shortest paths. The optimal function placement are node 6 for VNF $(1,1)$, node 11 for VNF $(1,2)$, node 2 for VNF $(2,1)$, and node 5 for VNF $(2,2)$. The resulting average cost is 22.

Figs. 4a and 4b demonstrate the tradeoff between the time average cost and the time average total queue length as a function of the control parameter V. According to Fig. 4a, the time average cost, under the proposed DCNC dynamic control algorithm, converges to approximately 22. At the same time, the time average queue length increases linearly with respect to V. These results clearly demonstrate the $[O(1/V), O(V)]$ cost-delay tradeoff as suggested by the performance bounds of Theorem 2.

Fig. 4c exhibits the evolution of the average cloud network cost over time under DCNC with $V = 5$ and $V = 10$. Observe how the time average cost with both values of V converge to approximately 22. Consistently with Theorem 2, the time average cost with $V = 10$ has a smaller deviation bound from the minimum average cost than with $V = 5$, as illustrated in Fig. 4c, as well as in Fig. 4a. However, as shown in Fig. 4c, the converging speed with $V = 5$ is faster than with $V = 10$, which also indicates a tradeoff in choosing the value of V to control the evolution of the cloud network.

VIII. Conclusion

We addressed the network service distribution problem in a dynamic cloud network setting, in which demands are unknown and time varying. We extended the *Lyapunov drift-plus-penalty* network control algorithm to account for both transmission and processing of network flows and the corresponding allocation of network and cloud resources. We first characterized the cloud network capacity region and then proposed a distributed joint flow scheduling and resource allocation algorithm that stabilizes the underlying queuing system within this region, while achieving arbitrarily close to minimum average cloud network cost, with a tradeoff in network delay. Preliminary numerical results conform to our theoretical analysis and demonstrate remarkably good convergence to the optimal cloud network configuration in a sample of illustrative scenarios.

Acknowledgements

This work was supported in part by the US National Science Foundation.

References

[1] Bell Labs Strategic White Paper, "Metro Network Traffic Growth: An Architecture Impact Study," December 2013.
[2] Marcus Weldon, "The Future X Network," *CRC Press*, October 2015.
[3] L. Lewin-Eytan, J. Naor, R. Cohen, D. Raz, "Near Optimal Placement of Virtual Network Functions," *IEEE INFOCOM*, 2015.
[4] M. Barcelo, J. Llorca, A. M. Tulino, N. Raman, "The Cloud Servide Distribution Problem in Distributed Cloud Networks" *IEEE ICC*, 2015.
[5] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems", *Synthesis Lectures on Communication Networks*, Morgan & Claypool, vol. 3, pp. 1–211, 2010.
[6] M. J. Neely, "Energy optimal control for time-varying wireless networks", *IEEE Transactions on Information Theory*, vol. 52, pp. 2915–2934, July, 2006.
[7] M. J. Neely, "Optimal Backpressure Routing for Wireless Networks with Multi-Receiver Diversity", *2006 40th Annual Conference on Information Sciences and Systems*, vol. 3, pp. 18–25, March, 2006.
[8] S. Nastic, S. Sehic, D.-H. Le, H.-L. Truong, and S. Dustdar, "Provisioning software-defined IoT cloud systems," *Future Internet of Things and Cloud (FiCloud)*, pp. 288–295, 2014.
[9] H. Feng, J. Llorca, A. M. Tulino, A. F. Molisch, "Optimal Dynamic service optimization in distributed cloud networks", *to be submitted*.