Reliable Service Chain Composition in NFV Systems: A Game Perspective

Simeng Bian, Xi Huang, Ziyu Shao, Yang Yang
School of Information Science and Technology, ShanghaiTech University
Email: {biansm, huangxi, shaozy, yangyang}@shanghaitech.edu.cn

***Abstract*—Network function virtualization (NFV) initiates a revolution of network service (NS) delivery by forming each NS as a chain of virtual network functions across commodity servers. Despite various benefits of NFV, deciding the chains that induce short latency and less congestion remains a key challenge, *a.k.a.* service chain composition problem. Existing works mainly resort to centralized solutions that require full knowledge of the network state to coordinate different users' traffic and NSs, overlooking privacy issues and the non-cooperative interactions among users. Moreover, failures due to user/resource unavailability make the problem even more challenging. By modeling the service chain composition as a *non-cooperative game*, we formulate the problem as searching the Nash Equilibrium (NE) with the best system performance. By exploiting the unique problem structure, we show that the game is a *potential game*, meaning that the unilateral improvement made by any player also leads a global system performance improvement. Leveraging Markov approximation techniques, we propose DISCCA, an efficient algorithm that guides the system towards the NE with short latency and low congestion, through distributed decision making by individual users. By extending the model with user and resource unavailability, we show that the new game is still potential, while proposing a variant of DISCCA to adapt to failures. Results from extensive simulations show that DISCCA efficiently approximates the optimum system performance within mild-value of iterations, even in the presence of failures.**

I. Introduction

Propelled by the ever-growing variety and quality of network services (NSs), traditional NS provisioning based on dedicated hardwares has come to its end because of its poor flexibility and scalability in network management. Recently, network function virtualization (NFV) revolutionizes the way of NS delivery by virtualizing and deploying network functions as applications on commodity servers, facilitating better flexibility in NS provisioning, manageability, reliability, and performance guarantee.

In a NFV system, each virtualized network function (VNF) has multiple instances distributed on different servers. A typical NS is then constructed by chaining instances of different VNFs successively, *a.k.a.* service chain composition [1]. Once deployed, users who subscribe the NS generate traffic that flow through the instances and acquire treatments in a pipeline fashion. The key challenge is to decide the service chaining decisions that induce short latency and low congestion. On the one hand, if the chained instances in a NS are distributed on servers with poor network conditions, users will experience long latency, leading to SLA violation and reduction in operator revenues. On the other hand, NSs may share the same VNF instance on the same server, leading to unfavorable resource contention and service quality degradation.

TABLE I
Comparison between existing solutions and ours.

	Game	Distributed	Failures	Broadcast
[3]	×	×	×	none
[4]	×	×	×	none
[5]	×	×	×	none
[6]	✓	✓	×	entire network states
Our solution	✓	✓	✓	only two bits

The service chain composition problem is in general $\mathcal{NP}$-hard [2]. Existing works mainly resort to designing heuristics or approximation algorithms to solve the problem. Bari *et al.* [3] formulate it as an ILP problem, and develop a heuristic solution based on Viterbi algorithm, leading to solutions with 1.3 times of performance to the optimal. Beck *et al.* [4] propose a heuristic method that coordinate the composition of VNF chains and their embedding into the substrate network, scaling well with hundreds of nodes. Zhang *et al.* [5] leverage the open Jackson queue network model and propose heuristic algorithms for joint service chaining and request scheduling problems. All above solutions investigate the problem in a centralized way, coordinating NSs to reach the optimal system performance with fully knowledge of system state. These solutions can be impractical due to privacy issues and non-cooperative interaction among users in practice.

In fact, game theory provides a framework that greatly captures the strategic interaction among players. Inspired by this, D'Oro *et al.* [6] model the chain composition as a *congestion game*, while proposing a distributed algorithm with provable convergence performance to a Nash Equilibrium (NE) state of the game in polynomial time. However, the resultant equilibrium state is not necessarily the one with optimal system performance. In fact, it is in general $\mathcal{NP}$-hard to find the optimal NE [2]. On the other hand, though being distributed over users, their solution still requires a considerable amount of network state broadcast within each iteration, causing undesirable messaging overheads and communication costs. Moreover, their model does not cover user and resource failures, which are inevitable in practice and has significant affect on system performance as stated in [7]–[11]. We summarize our solution in this paper with previous ones in Table I.

In this paper, we address the service chain composition problem by modeling it as a non-cooperative game while considering the unavailability of both users and resources. By extracting the problem's unique structure, we show that the game is a *potential game* with a particular weighted potential function, which is just the overall system performance. In such a way, individual users require only local information to make unilateral efforts that leads to global system performance improvement. By leveraging Markov approximation techniques, we propose DISCCA, an low-complexity, distributed algorithm that achieves the optimal NE with only polynomial number of iterations. Our main contributions are summarized as follows:

- ◇ **Modeling and Formulation:** We develop an exquisite model that accurately captures the system dynamics and user interactions. Based on the model, we are the first to formulate the service chaining composition as a non-cooperative game while taking both user and resource failures into consideration. We show that the game is a potential game and formulate the problem as the one to find the optimal NE of the game.
- ◇ **Algorithm Design:** By exploiting the unique problem structure and leveraging Markov approximation techniques [12], we propose DISCCA, a distributed and computationally efficient algorithm that decides the reliable chain compositions, effectively shortening latency and reducing congestion. DISCCA allows independent decision making of individual users with local information and minimum interactions with the others.
- ◇ **Experimental Verification and Analysis:** Simulation results show that DISCCA approximates the optimal system performance, in terms of both short latency and low congestion, through only mild-value of iterations. Specifically, the increase in convergence time is almost linearly proportional to the number of users. In addition, simulations also reveal the rapid adaptivity of DISCCA to significant change in network dynamics, even in the presence of failures.

The rest of the paper is organized as follows. Section II introduces some basic notations and the service chain composition problem. Section III reformulates it as a non-cooperative game while proving its potential property. In this part, we propose DISCCA, an efficient distributed algorithm to chain instances with near-optimal latency and congestion. Next, Section IV extends the previous model with user and resource failures, proves that the new game is still potential, and proposes a variant of DISCCA. Section V presents our results from comprehensive simulations and the corresponding analysis, while Section VI concludes this paper.

II. NFV System Model

A. Model Notations

We consider a NFV system owned by a Telco Operator that provides network services to its users [6]. Figure 1 shows such a sample NFV system. Inside the system there is a set of servers $\mathcal{V}$, with a size of $V \triangleq |\mathcal{V}|$. Servers are able to run

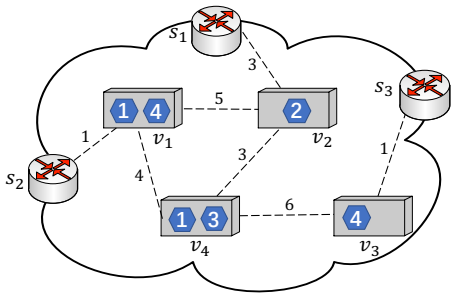


Fig. 1. A sample NFV system with four servers $\mathcal{V} = \{v_1, v_2, v_3, v_4\}$ and three gateway routers $\mathcal{S} = \{s_1, s_2, s_3\}$. The blue hexagons are VNF instances (VMs) running on servers, *e. g.* $\mathcal{F}_{v_1} = \{1, 4\}$.

virtual network function (VNF) instances on virtual machines (VMs). Each server $v \in \mathcal{V}$ is equipped with a fixed set of VNF, denoted by $\mathcal{F}_v$. Each VNF has only one instance on a server, while each instance is deployed in a standalone VM. Our model can be directly extended to the case with more than one instance of the same type on a server. To distinguish instances of the same VNF on different servers, we use (v, f) to denote a specific VNF instance, where v denotes the server where the instance resides, and f denotes the VNF that the instance belongs to. Following this notation, we use $\mathcal{R}$ to denote the set of all VNF instances (VMs)[1] in this system. For each $r = (v, f) \in \mathcal{R}$, we denote v by r_1 and f by r_2.

Besides servers, the system also contains a set of gateway routers $\mathcal{S}$ that manage the communication with outer networks. For any two nodes $u_1, u_2 \in \mathcal{S} \cup \mathcal{R}$, l_{u_1,u_2} denotes the latency between them.[2] Outside the system, there is a set of users $\mathcal{N}$ that subscribe different NSs, with a size of $N \triangleq |\mathcal{N}|$. Each user $i \in \mathcal{N}$ generates a traffic flow of rate λ_i. The traffic flows into the system through some gateway router $s_i \in \mathcal{S}$, and leaves through another router $t_i \in \mathcal{S}$. The user traffic requests a specific network service (NS) $\mathcal{F}_i$, denoted by an ordered chain of VNFs $\{f_{i,1}, f_{i,2}, \cdots, f_{i,|\mathcal{F}_i|}\}$. All required VNFs are assumed available in the system, *i.e.*, $\bigcup_{i\in\mathcal{N}} \mathcal{F}_i \subseteq \bigcup_{v\in\mathcal{V}} \mathcal{F}_v$.

For each user's request $\mathcal{F}_i$, we use $\mathbf{w}_i$ to denote the service chaining decision. $w_{i,j} = r$ means that user i's jth VNF $f_{i,j}$ will be assigned onto VM r to execute. Note that a VNF f can only be assigned to those VMs that run VNF f, *i.e.*, $w_{i,j} \in \{r \in \mathcal{R} \mid r_2 = f_{i,j}\}$. For simplicity, we use $\mathbf{w}_{-i}$ to denote the service chaining decisions of all other users excluding i, *i.e.*, $\mathbf{w}_{-i} \triangleq \{\mathbf{w}_k \mid k \in \mathcal{N}/\{i\}\}$, and denote $\mathbf{w} \triangleq (\mathbf{w}_i, \mathbf{w}_{-i})$. Table II summarizes the main notations of this paper.

B. Cost Functions

In this part, we define the cost function of user i incurred by different service chaining decisions.

1) Latency: Response time often makes one of the most concerned requirements in user SLAs. Under different service chaining decisions, users' traffic may experience different

[1]In this paper, we use the terms VNF instance and VM interchangeably.
[2]The latency can be set as the number of hops, round-trip time, *etc.*

TABLE II
NOTATIONS USED IN THIS PAPER.

Notation	Description
$\mathcal{V}$	Set of commodity servers
V	Number of servers, $V \triangleq \lvert\mathcal{V}\rvert$
$\mathcal{F}_v$	Set of VNFs provided by server v
$\mathcal{R}$	Set of VMs (VNF instances) in the system
$\mathcal{S}$	Set of gateway routers
l_{u_1,u_2}	Latency between node u_1 and u_2
$\mathcal{N}$	Set of users
N	Number of users, $N \triangleq \lvert\mathcal{N}\rvert$
λ_i	Traffic rate of user i
s_i, t_i	Ingress and egress gateway routers of user i
$\mathcal{F}_i$	Network service request of user i
$f_{i,j}$	The jth VNF in user i's NS request
$\mathbf{w}_i$	Service chaining decision of user i
$w_{i,j}$	The VM chosen for the jth VNF of user i's request
$\mathbf{w}_{-i}$	Service chaining decisions of other users excluding i
$\mathbf{w}$	Combination of all users' decisions

levels of latency. Given decision $\mathbf{w}_i$, we define the total latency experienced by user i as

$$c_i^{(\mathrm{L})}(\mathbf{w}_i) \triangleq l_{s_i,w_{i,1}} + \sum_{j=2}^{|\mathcal{F}_i|} l_{w_{i,j-1},w_{i,j}} + l_{w_{i,|\mathcal{F}_i|},t_i}. \quad (1)$$

2) Congestion: Since traffic flows of different users may adopt the same instance on the same server, the congestion due to their resource contention is inevitable. Given the set of users that select instance r, denoted by

$$\mathcal{M}_r(\mathbf{w}) \triangleq \{k \in \mathcal{N} | r \in \mathbf{w}_k\}, \quad (2)$$

we define the workload on instance r as $\delta_r(\mathbf{w})$

$$\delta_r(\mathbf{w}) \triangleq \sum_{i \in \mathcal{M}_r(\mathbf{w})} \lambda_i. \quad (3)$$

Then the congestion cost of user i is defined as

$$c_i^{(\mathrm{C})}(\mathbf{w}) \triangleq \sum_{r \in \mathbf{w}_i} \delta_r(\mathbf{w}). \quad (4)$$

3) Cost: Combining the cost of latency and congestion together, we define total cost for user i as

$$c_i(\mathbf{w}) \triangleq \alpha c_i^{(\mathrm{L})}(\mathbf{w}_i) + c_i^{(\mathrm{C})}(\mathbf{w}). \quad (5)$$

where α is a positive weighting parameter.

III. Problem Formulation and Algorithm Design

Given the model in the previous section, one can formulate the service chain composition problem as an optimization problem that aims to coordinate users' behavior so that the system can reach a global optimum. However, that may require the full information of network state and users' cooperation, which may be impractical due to privacy concerns and resource contention among users. In fact, game theory provides a general framework to characterize the strategic interaction among users. Following this entry, we switch to viewing the chain composition process as a game among various users.

A. Preliminary to Game Theory

Typically, a game consists of a set of players. Each player has her own action space and a function that maps all players' actions to a resultant cost or payoff. In a non-cooperative game, each rational player strategically chooses action in response to others' actions so as to minimize her cost (maximize the payoff). A *Nash Equilibrium* (NE) refers to such a state that no player has the incentive to change her current action unilaterally. Among different types of game, *potential games* are often favorable due to its nice property that the existence of NE is always guaranteed. A game is said to be *potential* if any player's incentive to change her own action can be equivalently expressed by a global potential function [13], [14].

B. Reformulation as A Game

We consider a game among a set of players $\mathcal{N}$. Player i adopts a weight which is her own traffic rate λ_i. The VNF instances $\mathcal{R}$ are actually the resources competed by players. Each player i picks an action, *i.e.* a chaining decision $\mathbf{w}_i$, from a finite action space $\mathcal{W}_i$, including all possible service chaining decisions of player i. A special case is when all player requests have a length of F and each server is equipped with all F VNF instances, then the size of each player's action space is V^F. In general, we use $\mathcal{W} \triangleq \bigcup_{i \in \mathcal{N}} \mathcal{W}_i$ to denote the combination of all players' action spaces, also referred as state space. The cost function of player i is defined as in (5). Accordingly, we define the game $\mathcal{G}$ by the following quadruple:

$$\mathcal{G} \triangleq (\mathcal{N}, (\lambda_i)_{i \in \mathcal{N}}, \mathcal{W}, (c_i)_{i \in \mathcal{N}}). \quad (6)$$

Theorem 1: The game $\mathcal{G}$ is a weighted potential game with potential function

$$\Phi(\mathbf{w}) = 2\alpha \sum_{k \in \mathcal{N}} \lambda_k c_k^{(\mathrm{L})}(\mathbf{w}_k) + \sum_{r \in \mathcal{R}} \left[\delta_r(\mathbf{w})\right]^2, \quad (7)$$

such that for any state pair $\mathbf{w} = (\mathbf{w}_i, \mathbf{w}_{-i})$ and $\mathbf{w}' = (\mathbf{w}'_i, \mathbf{w}_{-i})$ that only player i changes her action,

$$\Phi(\mathbf{w}) - \Phi(\mathbf{w}') = 2\lambda_i \left[c_i(\mathbf{w}) - c_i(\mathbf{w}')\right]. \quad (8)$$

Proof 1: $\Phi(\mathbf{w})$ is a potential function if and only if we can deduce (8) from (7).

For any two states $\mathbf{w} = (\mathbf{w}_i, \mathbf{w}_{-i})$ and $\mathbf{w}' = (\mathbf{w}'_i, \mathbf{w}_{-i})$, from (7) we have

$$\begin{aligned} &\Phi(\mathbf{w}) - \Phi(\mathbf{w}') \\ &= 2\lambda_i \left[\alpha c_i^{(\mathrm{L})}(\mathbf{w}_i) - \alpha c_i^{(\mathrm{L})}(\mathbf{w}'_i)\right] + \sum_{r \in \mathcal{R}} \left[\delta_r^2(\mathbf{w}) - \delta_r^2(\mathbf{w}')\right]. \end{aligned} \quad (9)$$

Next, we focus on the second summation term on the right-hand side of (9). Let's divide the VMs in $\mathcal{R}$ into three subsets.

- $\mathcal{R}_1$: VMs adopted by player i in $\mathbf{w}_i$ but not in $\mathbf{w}'_i$;
- $\mathcal{R}_2$: VMs adopted in decision $\mathbf{w}'_i$ but not in $\mathbf{w}_i$;
- $\mathcal{R}_3$: VMs that are both adopted in $\mathbf{w}_i$ and $\mathbf{w}'_i$ or neither.

Note that $|\mathcal{R}_1| = |\mathcal{R}_2|$ because for any VM r such that $r \in \mathbf{w}_i, r \notin \mathbf{w}'_i$, there must exist another VM r' such that $r' \in \mathbf{w}'_i, r' \notin \mathbf{w}_i$. Then,

$$\begin{aligned}
&\sum_{r\in\mathcal{R}} \left[\delta_r^2(\mathbf{w}) - \delta_r^2(\mathbf{w}')\right] \\
&= \sum_{r\in\mathcal{R}_1} \left\{\delta_r^2(\mathbf{w}) - \left[\delta_r(\mathbf{w}) - \lambda_i\right]^2\right\} \\
&\quad + \sum_{r\in\mathcal{R}_2} \left\{\left[\delta_r(\mathbf{w}') - \lambda_i\right]^2 - \delta_r^2(\mathbf{w}')\right\} \\
&= \sum_{r\in\mathcal{R}_1} \left[2\lambda_i\delta_r(\mathbf{w}) - \lambda_i^2\right] + \sum_{r\in\mathcal{R}_2} \left[-2\lambda_i\delta_r(\mathbf{w}') + \lambda_i^2\right] \\
&= 2\lambda_i \left\{\sum_{r\in\mathcal{R}_1} \delta_r(\mathbf{w}) - \sum_{r\in\mathcal{R}_2} \delta_r(\mathbf{w}')\right\} - |\mathcal{R}_1|\lambda_i^2 + |\mathcal{R}_2|\lambda_i^2 \\
&= 2\lambda_i \left[c_i^{(\mathrm{C})}(\mathbf{w}) - c_i^{(\mathrm{C})}(\mathbf{w}')\right].
\end{aligned} \tag{10}$$

Substituting (10) into (9), we obtain

$$\begin{aligned}
&\Phi(\mathbf{w}) - \Phi(\mathbf{w}') \\
&= 2\lambda_i \left[\alpha c_i^{(\mathrm{L})}(\mathbf{w}_i) - \alpha c_i^{(\mathrm{L})}(\mathbf{w}'_i)\right] + 2\lambda_i \left[c_i^{(\mathrm{C})}(\mathbf{w}) - c_i^{(\mathrm{C})}(\mathbf{w}')\right] \\
&= 2\lambda_i \left[c_i(\mathbf{w}) - c_i(\mathbf{w}')\right].
\end{aligned} \tag{11}$$

So far we have deduced (8) from (7). Proof of *Theorem 1* is done.■

Let's look more deeply at the potential function in (7). We observe that the first term $\sum_{k\in\mathcal{N}} \lambda_k c_k^{(\mathrm{L})}(\mathbf{w}_k)$ is the summation of the product of each player's traffic rate and the latency she experiences, which actually represents the overall traffic load of the network. The second term $\sum_{r\in\mathcal{R}} \left[\delta_r(\mathbf{w})\right]^2$ is the summation of the square of workloads of all VMs, which represents the congestion level of the entire network. We interpret the weighted sum of overall traffic load and congestion load, *i.e.* the potential function, as the system performance, also referred as *social cost* in the following.

We summarize that each player aims to search action that reduces her own cost, while the Telco operator's goal is to minimize the overall social cost, *i.e.* search the *social optimum*.

C. Problem Formulation

From section III.B, to find the social optimum we should solve the following problem

$$\begin{aligned}
\underset{\mathbf{w}}{\text{Minimize}} \quad & \Phi(\mathbf{w}) \\
\text{Subject to} \quad & \mathbf{w} \in \mathcal{W}.
\end{aligned} \tag{12}$$

We define the optimal solution to problem (12) as $\mathbf{w}^*$, then we have the following corollary.

Corollary 1: The social optimum $\mathbf{w}^*$ is a Nash Equilibrium (NE) of game $\mathcal{G}$.

Proof 2: Assume $\mathbf{w}^*$ is current state and a player i prepares to change her action from $\mathbf{w}_i^*$ to $\mathbf{w}_i \in \mathcal{W}_i$. We know

$$\Phi(\mathbf{w}_i^*, \mathbf{w}_{-i}^*) - \Phi(\mathbf{w}_i, \mathbf{w}_{-i}^*) \le 0 \tag{13}$$

because $\mathbf{w}^*$ is the solution of problem (12). Then according to Theorem 1, we have

$$c_i(\mathbf{w}_i^*, \mathbf{w}_{-i}^*) - c_i(\mathbf{w}_i, \mathbf{w}_{-i}^*) \le 0, \tag{14}$$

which indicates that any action change from $\mathbf{w}_i^*$ won't brings a lower cost to player i. Therefore, $\mathbf{w}^*$ is a NE because no player has the incentive to change her action unilaterally. ■

From corollary 1, in the following part we refers finding the social optimum as searching the optimal NE of game $\mathcal{G}$.

Problem (12) is essentially a combinatorial problem which is in general $\mathcal{NP}$-hard to solve. By applying *log-sum-exponential* approximation [12], we transform (12) into the following form:

$$\begin{aligned}
\underset{\pi\succeq 0}{\text{Minimize}} \quad & \sum_{\mathbf{w}\in\mathcal{W}} \pi_{\mathbf{w}}\Phi(\mathbf{w}) + \frac{1}{\beta}\sum_{\mathbf{w}\in\mathcal{W}} \pi_{\mathbf{w}}\log\pi_{\mathbf{w}} \\
\text{Subject to} \quad & \sum_{\mathbf{w}\in\mathcal{W}} \pi_{\mathbf{w}} = 1.
\end{aligned} \tag{15}$$

with an approximation gap bounded by $\frac{1}{\beta}\log|\mathcal{W}|$, where β is a tunable positive parameter. Problem (15) can also be viewed as the problem of deciding the time proportion $\pi_{\mathbf{w}}$ that the system spends on state $\mathbf{w}$, in a bid to minimize the time-average overall cost Φ over a period of time with an entropy term. By solving the KKT conditions of problem (15), the optimal probability distribution turns out to be

$$\pi_{\mathbf{w}}^* = \frac{\exp[-\beta\Phi(\mathbf{w})]}{\sum_{\mathbf{w}'\in\mathcal{W}}\exp[-\beta\Phi(\mathbf{w}')]},\ \forall\ \mathbf{w}\in\mathcal{W}. \tag{16}$$

D. Constructing a Markov Chain

It is impractical to solve the optimal solution (16) directly since that requires the full knowledge of states in the enormous search space $\mathcal{W}$. Instead, we resort to designing a discrete-time Markov chain while adopting random sampling techniques to compute $\pi_{\mathbf{w}}^*$ in an asymptotic manner. In general, we have two degrees of freedom in designing the Markov chain.

1) Topology design: We construct a Markov chain with its state space as $\mathcal{W}$. Thus each state in the chain corresponds to one possible combination of all players' actions, *i.e.*, $\mathbf{w} \in \mathcal{W}$. For the sake of description, we define $H_{\mathbf{w},\mathbf{w}'}$ as the Hamming distance between the two states $\mathbf{w}$ and $\mathbf{w}'$.

$$H_{\mathbf{w},\mathbf{w}'} \triangleq \sum_{i\in\mathcal{N}} 1_{\mathbf{w}_i\neq\mathbf{w}'_i}, \tag{17}$$

in which 1_A is an indicator of event A that equals 1 when A occurs and 0 otherwise. In the topology, two states are connected if their Hamming distance $H_{\mathbf{w},\mathbf{w}'} \le 1$, *i.e.*, at most one player changes her action.

To design a time-reversible Markov chain, we should first prove the connectivity of the topology.

Proof 3: Actually, the Hamming distance $H_{\mathbf{w},\mathbf{w}'}$ represents the number of steps needed to reach state $\mathbf{w}'$ from state $\mathbf{w}$. From the definition of Hamming distance in (17), we can conclude that

$$H_{\mathbf{w},\mathbf{w}'} \le N, \tag{18}$$

because there are at most N differences. Therefore any state pairs in the topology are reachable within at most N steps, the topology is connected. ■

2) Transition probability design: We define the self transition probability of state $\mathbf{w}$ as follows:

$$p_{\mathbf{w},\mathbf{w}} \triangleq \frac{1}{N}\sum_{i\in\mathcal{N}}\frac{\exp[-\beta\Phi(\mathbf{w})]}{\sum_{\bar{\mathbf{w}}\in\mathcal{A}_i}\exp[-\beta\Phi(\bar{\mathbf{w}})]}, \tag{19}$$

where $\mathcal{A}_i$ is a subset of the state space that only player i's action is changed while others' remain unchanged, *i.e.* $\mathcal{A}_i = \{(\bar{\mathbf{w}}_i, \mathbf{w}_{-i}) \mid \bar{\mathbf{w}}_i \in \mathcal{W}_i\}$.

Next, we consider two states $\mathbf{w}$ and $\mathbf{w}'$ with $H_{\mathbf{w},\mathbf{w}'} = 1$. For simplicity, we use i to denote the only player that changes her action. Then, the transition probability from $\mathbf{w}$ to $\mathbf{w}'$ is defined as

$$p_{\mathbf{w},\mathbf{w}'} \triangleq \frac{1}{N}\frac{\exp[-\beta\Phi(\mathbf{w}')]}{\sum_{\bar{\mathbf{w}}\in\mathcal{A}_i}\exp[-\beta\Phi(\bar{\mathbf{w}})]}. \tag{20}$$

The transition probability for all other state pairs are all zeros.

We should verify the following two things to assure the designed Markov chain correct and time-reversible.

- ◇ Transition probabilities out of one state should sum to 1. For any state $\mathbf{w} \in \mathcal{W}$,

$$\begin{aligned}
&\sum_{\mathbf{w}'\in\mathcal{W}} p_{\mathbf{w},\mathbf{w}'} = \sum_{\substack{\mathbf{w}'\in\mathcal{W}\\ H_{\mathbf{w},\mathbf{w}'}=0}} p_{\mathbf{w},\mathbf{w}'} + \sum_{\substack{\mathbf{w}'\in\mathcal{W}\\ H_{\mathbf{w},\mathbf{w}'}=1}} p_{\mathbf{w},\mathbf{w}'}\\
&= p_{\mathbf{w},\mathbf{w}} + \sum_{i\in\mathcal{N}}\sum_{\substack{\hat{\mathbf{w}}\in\mathcal{A}_i\\ \hat{\mathbf{w}}\neq\mathbf{w}}} p_{\mathbf{w},\hat{\mathbf{w}}}\\
&= \frac{1}{N}\sum_{i\in\mathcal{N}}\frac{\exp[-\beta\Phi(\mathbf{w})]}{\sum_{\bar{\mathbf{w}}\in\mathcal{A}_i}\exp[-\beta\Phi(\bar{\mathbf{w}})]}\\
&\quad + \sum_{i\in\mathcal{N}}\sum_{\substack{\hat{\mathbf{w}}\in\mathcal{A}_i\\ \hat{\mathbf{w}}\neq\mathbf{w}}}\frac{1}{N}\frac{\exp[-\beta\Phi(\hat{\mathbf{w}})]}{\sum_{\bar{\mathbf{w}}\in\mathcal{A}_i}\exp[-\beta\Phi(\bar{\mathbf{w}})]}\\
&= \frac{1}{N}\sum_{i\in\mathcal{N}}\frac{\sum_{\hat{\mathbf{w}}\in\mathcal{A}_i}\exp[-\beta\Phi(\hat{\mathbf{w}})]}{\sum_{\bar{\mathbf{w}}\in\mathcal{A}_i}\exp[-\beta\Phi(\bar{\mathbf{w}})]} = 1.
\end{aligned} \tag{21}$$

- ◇ Transition probabilities between any two states should satisfy the detailed balance equation.
  The detailed balance equation is naturally satisfied for state self transition. For any two different connected states $\mathbf{w}$ and $\mathbf{w}'$, recalling (16) and (20), we have

$$\begin{aligned}
&\pi^*_{\mathbf{w}}p_{\mathbf{w},\mathbf{w}'}\\
=&\frac{\exp[-\beta\Phi(\mathbf{w})]}{\sum_{\bar{\mathbf{w}}\in\mathcal{W}}\exp[-\beta\Phi(\bar{\mathbf{w}})]}\cdot\frac{1}{N}\frac{\exp[-\beta\Phi(\mathbf{w}')]}{\sum_{\bar{\mathbf{w}}\in\mathcal{A}_i}\exp[-\beta\Phi(\bar{\mathbf{w}})]}\\
=&\frac{\exp[-\beta\Phi(\mathbf{w}')]}{\sum_{\bar{\mathbf{w}}\in\mathcal{W}}\exp[-\beta\Phi(\bar{\mathbf{w}})]}\cdot\frac{1}{N}\frac{\exp[-\beta\Phi(\mathbf{w})]}{\sum_{\bar{\mathbf{w}}\in\mathcal{A}_i}\exp[-\beta\Phi(\bar{\mathbf{w}})]}\\
=&\pi^*_{\mathbf{w}'}p_{\mathbf{w}',\mathbf{w}}.
\end{aligned} \tag{22}$$

From all above, we prove that the Markov chain we design is time-reversible and adopts (16) as its stationary distribution.

E. Algorithm Design

To implement the designed discrete-time Markov chain, we give the following algorithm description. During each iteration, one player is chosen randomly and uniformly from N players, and gets the chance update her action. When player i is chosen, we define the in-step transition probability from state $\mathbf{w} = (\mathbf{w}_i, \mathbf{w}_{-i})$ to $\mathbf{w}' = (\mathbf{w}'_i, \mathbf{w}_{-i})$ as $p^{(i)}_{\mathbf{w},\mathbf{w}'}$. Note that $\mathbf{w}'_i$ may equal $\mathbf{w}_i$, indicating a self transition.

$$p^{(i)}_{\mathbf{w},\mathbf{w}'} \triangleq \frac{\exp[-\beta\Phi(\mathbf{w}')]}{\sum_{\bar{\mathbf{w}}\in\mathcal{A}_i}\exp[-\beta\Phi(\bar{\mathbf{w}})]}. \tag{23}$$

The transition probabilities for other state pairs are all zeros.

There are two challenges in such implementation. The first challenge is how to select one from N players randomly and uniformly per iteration in a distributed way. One possible solution is that each player maintains a Poisson countdown clock with rate μ independently. The player is chosen once her clock expires the first, and all other players are notified to suspend at current iteration. We have the following remarks:

- ◇ Any one of the N players is chosen with probability $1/N$.
- ◇ The probability that more than one players' clocks expire at the same time is 0.
- ◇ By setting $\mu \gg 1$, only a small fraction of time in each iteration is used to decide the selected player.

The second challenge is that, it is impractical to calculate the transition probabilities in (23), because often times the potential values are hard to obtain in real NFV networks. To address the challenge, we bridge the potential and cost function by defining a function $B_i : \mathcal{W}_{-i} \to \mathbb{R}$ as

$$\begin{aligned}
B_i(\mathbf{w}_{-i}) \triangleq\ & 2\alpha\sum_{k\in\mathcal{N}/\{i\}}\lambda_k c^{(\mathrm{L})}_k(\mathbf{w}_k) - |\mathcal{F}_i|\lambda_i^2\\
&+\sum_{r\in\mathcal{R}}\left(\sum_{k\in\mathcal{M}_r(\mathbf{w})/\{i\}}\lambda_k\right)^2.
\end{aligned} \tag{24}$$

Then we can rewrite the potential function in (7) as

$$\Phi(\mathbf{w}) = 2\lambda_i c_i(\mathbf{w}) + B_i(\mathbf{w}_{-i}), \tag{25}$$

where $B_i(\mathbf{w}_{-i})$ is unrelated to $\mathbf{w}_i$, the action of player i. Substituting (25) into (23) and applying the definition of $\mathcal{A}_i$, the transition probability can be reformulated as

$$p^{(i)}_{(\mathbf{w}_i,\mathbf{w}_{-i}),(\mathbf{w}'_i,\mathbf{w}_{-i})} = \frac{\exp[-2\beta\lambda_i c_i(\mathbf{w}'_i,\mathbf{w}_{-i})]}{\sum_{\bar{\mathbf{w}}_i\in\mathcal{W}_i}\exp[-2\beta\lambda_i c_i(\bar{\mathbf{w}}_i,\mathbf{w}_{-i})]}. \tag{26}$$

Hence, the transition probability only depends on the cost, which is often attainable in practical systems.

Based on previous designs, we propose an algorithm named DIstribute Service Chain Composition Algorithm (DISCCA). DISCCA solves the problem (15) and hence solve the initial problem (12) in an approximate way. We present the pseudo-code of DISCCA in **Algorithm 1**.

Additionally, for DISCCA, we have the following remarks:

- ◇ DISCCA requires only local information for the decision making by individual players; therefore, it can run in a distributed manner.

Algorithm 1 DIstributed Service Chain Composition Algorithm (DISCCA) of player i

Input: At the beginning of each iteration, a *RESET* signal is broadcast to all players.

Repeat do
if receive a signal *SUSPEND* **then**
  Suspend the clock
end if
if receive a signal *RESET* **then**
  Reset the clock with rate μ
  Begin count down again
end if
if the clock expires **then**
  Broadcast a signal *SUSPEND*
  Perform action $\mathbf{w}_i'$ chosen from action space $\mathcal{W}_i$ with probability $p^{(i)}_{(\mathbf{w}_i,\mathbf{w}_{-i}),(\mathbf{w}_i',\mathbf{w}_{-i})}$ in (26)
end if

- ◇ If we use a bit 0 to represent the *SUSPEND* signal and 1 to represent the *RESET* signal, then only two bits are broadcast within each iteration.
- ◇ The transition probability expression (26) is actually a weighted normalization. With a larger β, the system would spend a greater proportion of time on actions that incur lower costs. Thus intuitively, by increasing β, we can reduce the approximation gap of the converged cost; but that also leads to a longer time to converge to the optimal distribution [12]. In practice, the choice is often left to the system designers to make.

IV. Player and VM Failures

Sometimes, players may be offline due to network failures; likewise, VMs may also break down due to various reasons. The availability of any player or VM may greatly influence the system performance and make it off the equilibrium state [9]–[11]. Therefore, we extend our previous model with considering both player and VM failures. We assume independent failure occurrences in our model. Nonetheless, our model can be a basis for system designers to consider more complex scenarios that involve correlated failures. All players (VMs) are assumed to have identical survival probability, *i.e.*, $\Pr\{\text{player } i \text{ survives}\} = \theta,\ \forall i \in \mathcal{N}$ and $\Pr\{\text{VM } r \text{ survives}\} = \eta,\ \forall r \in \mathcal{R}$. Further, all service chains are assumed to have the same length of F. Our model can be extended to more general cases with various survival probability and chain lengths.

Next, we define the cost for player i given state $(\mathbf{w}_i, \mathbf{w}_{-i})$.

- ◇ If player i fails, she will receive an incompletion cost ω;
- ◇ If player i survives, but at least one of the selected VMs fails, she will also receive an incompletion cost ω;
- ◇ If both player i and all the chosen VMs survive, her latency cost remains unchanged as defined in (1), but the congestion cost depends also on other players' survival state. The new congestion cost is

$$\hat{c}_i^{(\mathrm{C})}(\mathbf{w}_i, \mathbf{w}_{-i}) = \sum_{r\in\mathbf{w}_i}\left(\lambda_i + \sum_{k\in\mathcal{M}_r(\mathbf{w})/\{i\}} \theta\cdot\lambda_k\right). \tag{27}$$

Accordingly, the new cost function of player i is

$$\begin{aligned}\hat{c}_i(\mathbf{w}_i, \mathbf{w}_{-i}) = (1-\theta)\cdot\omega + \theta\cdot\left(1-\eta^F\right)\cdot\omega \\ + \theta\cdot\eta^F\cdot\left(\alpha c_i^{(\mathrm{L})}(\mathbf{w}_i) + \hat{c}_i^{(\mathrm{C})}(\mathbf{w}_i, \mathbf{w}_{-i})\right).\end{aligned} \tag{28}$$

Though extended with player and VM failures, we show that the new game is still a potential game, as explicated by the following theorem.

Theorem 2: The new game $\hat{\mathcal{G}}$ with player and VM failures is still a weighted congestion game with potential function

$$\phi(\mathbf{w}) = 2\alpha\theta\sum_{k\in\mathcal{N}}\lambda_k\cdot c_k^{(\mathrm{L})}(\mathbf{w}_k) + \sum_{r\in\mathcal{R}}\left(\sum_{k\in\mathcal{M}_r(\mathbf{w})}\theta\cdot\lambda_k\right)^2, \tag{29}$$

such that for any state $\mathbf{w}'$ that only player i changes her action,

$$\phi(\mathbf{w}) - \phi(\mathbf{w}') = \frac{2\lambda_i}{\eta^F}\left[\hat{c}_i(\mathbf{w}) - \hat{c}_i(\mathbf{w}')\right]. \tag{30}$$

Proof 4: From the definition of ϕ in (29), we derive that for any two states $\mathbf{w}$ and $\mathbf{w}'$ that only differs in player i's action:

$$\begin{aligned}\phi(\mathbf{w}) - \phi(\mathbf{w}') &= 2\theta\lambda_i\left[\alpha c_i^{(\mathrm{L})}(\mathbf{w}_i) - \alpha c_i^{(\mathrm{L})}(\mathbf{w}_i')\right] \\ &+ \sum_{r\in\mathcal{R}}\left[\left(\sum_{k\in\mathcal{M}_r(\mathbf{w})}\theta\lambda_k\right)^2 - \left(\sum_{k\in\mathcal{M}_r(\mathbf{w}')}\theta\lambda_k\right)^2\right].\end{aligned} \tag{31}$$

Let's focus on the second summation term on the right-hand side. We divide the set of VMs $\mathcal{R}$ into three subsets $\mathcal{R}_1$, $\mathcal{R}_2$ and $\mathcal{R}_3$ similar as in *proof* 1.

$$\begin{aligned}&\sum_{r\in\mathcal{R}}\left[\left(\sum_{k\in\mathcal{M}_r(\mathbf{w})}\theta\lambda_k\right)^2 - \left(\sum_{k\in\mathcal{M}_r(\mathbf{w}')}\theta\lambda_k\right)^2\right] \\ &= \sum_{r\in\mathcal{R}_1}\left[\left(\sum_{k\in\mathcal{M}_r(\mathbf{w})}\theta\lambda_k\right)^2 - \left(\sum_{k\in\mathcal{M}_r(\mathbf{w})}\theta\lambda_k - \theta\lambda_i\right)^2\right] \\ &\quad + \sum_{r\in\mathcal{R}_2}\left[\left(\sum_{k\in\mathcal{M}_r(\mathbf{w}')}\theta\lambda_k - \theta\lambda_i\right)^2 - \left(\sum_{k\in\mathcal{M}_r(\mathbf{w}')}\theta\lambda_k\right)^2\right] \\ &= 2\theta\lambda_i\left[\sum_{r\in\mathcal{R}_1}\left(\sum_{k\in\mathcal{M}_r(\mathbf{w})}\theta\lambda_k\right) - \sum_{r\in\mathcal{R}_2}\left(\sum_{k\in\mathcal{M}_r(\mathbf{w}')}\theta\lambda_k\right)\right] \\ &= 2\theta\lambda_i\left(\hat{c}_i^{(\mathrm{C})}(\mathbf{w}) - \hat{c}_i^{(\mathrm{C})}(\mathbf{w}')\right).\end{aligned} \tag{32}$$

Recalling (28), we have

$$\begin{aligned}&\hat{c}_i(\mathbf{w}) - \hat{c}_i(\mathbf{w}') \\ &= \theta\eta^F \cdot \left[\alpha c_i^{(\mathrm{L})}(\mathbf{w}_i) - \alpha c_i^{(\mathrm{L})}(\mathbf{w}_i') + \hat{c}_i^{(\mathrm{C})}(\mathbf{w}) - \hat{c}_i^{(\mathrm{C})}(\mathbf{w}')\right].\end{aligned} \tag{33}$$

Combining (31), (32) and (33), we obtain

$$\phi(\mathbf{w}) - \phi(\mathbf{w}') = \frac{2\lambda_i}{\eta^F}\left(\hat{c}_i(\mathbf{w}) - \hat{c}_i(\mathbf{w}')\right). \tag{34}$$

Theorem 2 is proved. ■

Similar as in previous non-failure case, we can reformulate the new potential function ϕ as

$$\phi(\mathbf{w}_i, \mathbf{w}_{-i}) = 2\theta\lambda_i \hat{c}_i(\mathbf{w}_i, \mathbf{w}_{-i}) + \hat{B}_i(\mathbf{w}_{-i}), \tag{35}$$

where

$$\begin{aligned}\hat{B}_i(\mathbf{w}_{-i}) = 2\alpha\theta \sum_{k\in\mathcal{N}/\{i\}} \lambda_k c_k^{(\mathrm{L})}(\mathbf{w}_k) - |\mathcal{F}_i|\theta^2\lambda_i^2 \\ + \sum_{r\in\mathcal{R}} \left(\sum_{k\in\mathcal{M}_r(\mathbf{w})/\{i\}} \theta\cdot\lambda_k\right)^2.\end{aligned} \tag{36}$$

To find the optimal NE of this game, we transform the problem into a combinatorial problem that finds the optimal state $\mathbf{w}$ to minimize the potential function ϕ. Applying Markov approximation techniques, we solve its respective approximation problem by designing another Markov chain similar as in section III.D, while the transition probability within the step that player i is chosen is

$$\hat{p}^{(i)}_{(\mathbf{w}_i,\mathbf{w}_{-i}),(\mathbf{w}_i',\mathbf{w}_{-i})} = \frac{\exp[-2\beta\theta\lambda_i\hat{c}_i(\mathbf{w}_i',\mathbf{w}_{-i})]}{\sum_{\bar{\mathbf{w}}_i\in\mathcal{W}_i}\exp[-2\beta\theta\lambda_i\hat{c}_i(\bar{\mathbf{w}}_i,\mathbf{w}_{-i})]}. \tag{37}$$

The algorithm is similar to **Algorithm 1**, while the transition probability is replaced by (37). Actually we can view previous model as a special case by setting both the surviving probabilities θ and η as 1.

V. Numerical Results

In this part, we evaluate DISCCA under various simulation scenarios. We assume there are $N = 100$ players, each with a traffic rate of 1Mbps. Besides, there is a cluster of 5 servers, each equipped with three VNFs, *e.g.*, Firewall (FW), Load Balancer (LB), and Intrusion Detection System (IDS). For simplicity, the latency between any gateway router $s \in \mathcal{S}$ and VM $r \in \mathcal{R}$, and the latency between two VMs within the same server are set as 1. Other latency values are sampled randomly and uniformly from the interval $[2, 6]$. We assume all players require the same network service, *i.e.*, FW $\rightarrow$ LB $\rightarrow$ IDS. Parameter α and β are set as 1 by default. The clock rate μ is set as 10 in our simulation. All the results in the following are obtained by averaging over 100 runs.

In the following, Part A–C evaluates DISCCA without failures ($\theta = \eta = 1$), while Part D shows the simulation results under different levels of player and VM failures.

A. Performance

We first investigate how potential value and per-player average cost changes over iterations. We define the per-player average cost (hereinafter shorten as average cost) as

$$\bar{C} = \frac{1}{N}\sum_{i\in\mathcal{N}} c_i(\mathbf{w}_i, \mathbf{w}_{-i}). \tag{38}$$

Before presenting the results of DISCCA, we calculate the optimal value of potential value and average cost. In the case with 100 player and 5 servers, the optimal state is that each server holds 20 players and each player's request is processed within a single server. In this way, all players have the same cost and the optimal average cost is

$$\bar{C}_{opt} = (1+1+1+1) + (20+20+20) = 64. \tag{39}$$

Accordingly, the optimal potential value is

$$\Phi_{opt} = 2\times 100\times 4 + 5\times 3\times 20^2 = 6800. \tag{40}$$

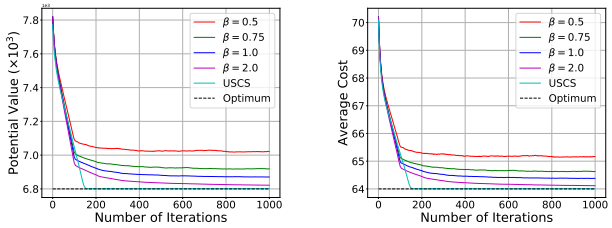
To evaluate the performance DISCCA more comprehensively, we also implement the Unilateral Service Chain Selection (USCS) algorithm designed in [6]. USCS is also a distributed algorithm but requires broadcasting large amount of data per iteration. Figure 2 present the performance of DISCCA in terms of (a) potential value and (b) average cost with different values of parameter β as well as the corresponding results of USCS. The black dotted lines present the corresponding optimal values (6800 and 64). We have the following observations:

1) USCS achieves a near-optimal cost, notably at the cost of broadcasting large amount of data with each iteration.
2) Both potential value and average cost decreases gradually and levels off as the number of iterations increases, indicating the convergence of DISCCA.
3) The values after convergence of both potential value and average cost decrease as β increases from 0.5 to 2. This is consistent with our previous analysis: larger β makes the Markov chain stays at low-potential (low-cost) states with higher probabilities, hence incurs lower potential value (average cost).
4) With $\beta = 2$, the curve approximates the optimal value closely within small number of iterations. For example, in Figure (a) the potential value at iteration 400 is about 6842, which is only 0.62% higher than the optimal value 6800. Similarly in Figure (b), the average cost at iteration 400 is 64.22, which is only 0.34% higher than the optimal value 64.

B. Scalability

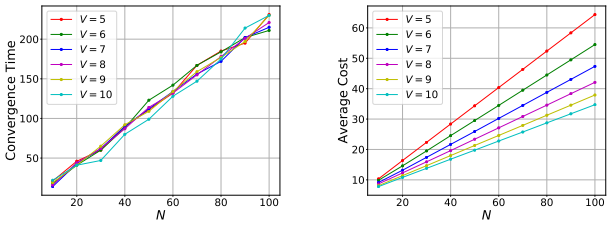
An algorithm should be scalable to be deployed in practical, large-scale systems. In this part, we investigate how the following factors affect the scalability of DISCCA:

- $\diamond$ N: number of players
- $\diamond$ V: number of servers
- $\diamond$ F: length of service chain
- $\diamond$ D: number of VNF replications



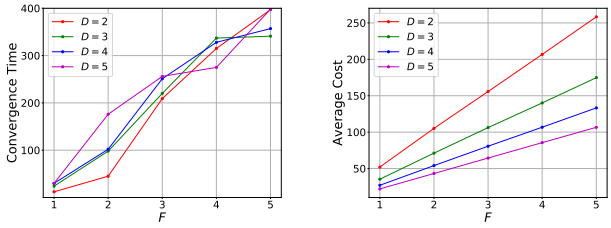
(a) Potential value. (b) Per-player average cost.

Fig. 2. Comparison of USCS and DISCCA under various values of β in terms of (a) potential value and (b) per-player average cost.



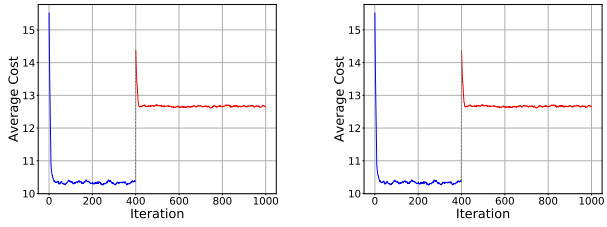
(a) Convergence time. (b) Per-player average cost.

Fig. 3. Performance of DISCCA under different settings of N and V in terms of (a) convergence time and (b) per-player average cost.



(a) Convergence time. (b) Average cost.

Fig. 4. Performance of DISCCA under different settings of F and D in terms of (a) convergence time and (b) per-player average cost.



(a) Player dynamics. (b) VM dynamics.

Fig. 5. Average cost v.s. iteration with (a) player and (b) VM dynamics.

By default, D is set as the number of servers V, *i.e.*, each VNF type has instances on all servers.

Figure 3(a) shows how parameters N and V affect the convergence time of DISCCA. We say the process converges if the sum of n successive potential differences is less than a small value ϵ. In fact, we set $n = 5$ and $\epsilon = 1$. From Figure 3(a), we make two observations:

1) The increase in convergence time is almost linearly proportional to the number of players N.
2) Increasing the number of servers nearly has no effect on the convergence of DISCCA.

Figure 3(b) presents how factors N and V affect the average cost. We have the following observations:

1) The average cost increases as N increases from 10 to 100. This is because with larger N, each individual will competes with more players, thereby suffering a higher congestion cost.
2) The average cost decreases as V increases from 5 to 10. In fact, with more servers, players are able to be distributed more separately, therefore the congestion cost is reduced.

Figure 4(a) shows how F and D affect the convergence time. We observe that the convergence time increases as F almost linearly, while parameter D has little effect. Figure 4(b) presents the average cost under different values of F and D. The average cost goes up as the service chain goes longer. The reason is that with a longer SC, the player experiences larger latency cost as well as higher congestion level. We also observe that with the same F, increasing the number of VNF replications D helps reduce the average cost. The reason is similar as increasing V, both cases bring more resources and lower the congestion cost.

C. Adaptivity

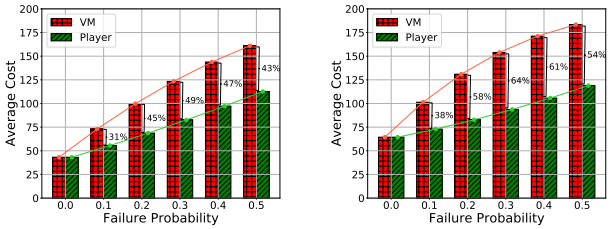
In practice, users often join and leave NFV systems dynamically [15]. Moreover, a VM may be offline due some various reasons and restart again after a while. Therefore, we should investigate the adaptivity of DISCCA in response to such network dynamics.

We run the simulation within 1000 iterations, and a new player with rate 5Mbps will join at the 400th iteration. Once the player joins, she picks an action randomly and starts an exponential count-down clock of rate μ. Figure 5(a) present the performance of DISCCA in such case with $N = 10$. We could observe that when the new player joins, the per-player average cost increases immediately and sharply. This is because the arrival of the new player increases the congestion costs of some players. But within a short period (less than ten iterations) the cost decreases and converges again. Results from Figure 5(a) verify that DISCCA can adapt to player dynamics quickly.

We also consider VM dynamics in the simulation. If one VM breaks down, all the tasks will be migrated to another randomly selected VM of the same type. During each simulation, we set the Firewall instance on the first server to terminate at 400th iteration. From Figure 5(b) we see that when the VM is closed, the cost rises up immediately because the random migration may cause imbalanced player distribution and incur high congestion cost. From the figure, we notice that after the sharp increment, the cost converges within ten iterations again. Figure 5(b) verifies the rapid adaptivity of DISCCA to VM dynamics.

TABLE III
AVERAGE COST UNDER DIFFERENT $\bar{\theta}$ AND $\bar{\eta}$.

$\bar{\theta}$ \ $\bar{\eta}$	0	0.1	0.2	0.3	0.4	0.5
0	43.2	73.1	100	123.4	143.8	161.1
0.1	55.6	83.1	107.7	129.5	148.3	164.1
0.2	68.7	93.7	116.1	135.9	153	167.4
0.3	82.6	105	125	142.7	157.9	170.8
0.4	97.2	116.8	134.4	149.8	163.2	174.4
0.5	112.6	129.3	144.2	157.3	168.6	178.2



(a) $F = 2$. (b) $F = 3$.

Fig. 6. Average cost v.s. failure probability of player (green stripped bar) and VM (red meshed bar) under different settings of SC length F.

D. DISCCA with Failure

As mentioned before, players and VMs may fail with some positive probability. In this part we investigate how the failure probability affect the performance of DISCCA. For simplicity we introduce $\bar{\theta} = 1 - \theta$ as the failure probability of players, and $\bar{\eta} = 1 - \eta$ as the failure probability of VMs. In Table III we show the average cost under different settings of $\bar{\theta}$ and $\bar{\eta}$ while the length of SC is set as 2. From this table, we observe that the average cost increases from left to right, top to bottom. In summary, larger failure probability (whether player failure or VM failure) incurs larger average cost.

We plotted Figure 6(a) to present the first row and first column of Table III. The red meshed bars present the case when $\bar{\eta}$ increases from 0 to 0.5 and $\bar{\theta}$ is default as 0 (the first row). Similarly the green stripped bars present the case when $\bar{\theta}$ ranges from 0 to 0.5 and $\bar{\eta} = 0$ (the first column). Figure 6(b) presents similar results but under the setting with $F = 3$. We have the following observations:

1) Higher failure probabilities lead to higher average costs.
2) The red meshed bars are always higher than the green stripped bars when failure probability ranges from 0.1 to 0.5. In other words, with the same probability value, VM failure has a greater impact on the average cost than player failure. The reason is that multiple VMs are employed to process a player's request, magnifying the impact of VM failures.
3) With fixed failure probability (0.2 as an example), the difference between the two bars in Figure 6(b) (58%) is larger than that in Figure 6(a) (45%). The reason is that the impact of VM failures expands further as the service chain grows.

VI. CONCLUSION

In this paper, we studied the service chain composition problem in NFV networks. By considering both user and resource failures into our model, we formulate the chain composition process as a non-corporative game. We show the game is a potential game with a given weighted potential function. By leveraging Markov approximation framework, we proposed DISCCA, a distributed and computational efficient algorithm that guides the system towards the optimal NE of the game through distributed decision making by individual users with only local information. Extensive simulation results show that DISCCA achieves the social optimum within mild-value of iterations, in terms of both short latency and low congestion, while adapting to network dynamics quickly, even in the presence of failures.

REFERENCES

[1] P. Quinn and T. Nadeau, "Problem statement for service function chaining," Tech. Rep., 2015.

[2] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *International Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 1346–1354.

[3] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions," in *International Conference on Network and Service Management (CNSM)*. IEEE, 2015, pp. 50–56.

[4] M. T. Beck and J. F. Botero, "Scalable and coordinated allocation of service function chains," *Computer Communications*, vol. 102, pp. 78–88, 2017.

[5] Q. Zhang, Y. Xiao, F. Liu, J. C. Lui, J. Guo, and T. Wang, "Joint optimization of chain placement and request scheduling for network function virtualization," in *International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 731–741.

[6] S. DOro, L. Galluccio, S. Palazzo, and G. Schembra, "Exploiting congestion games to achieve distributed service chaining in nfv networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 2, pp. 407–420, 2017.

[7] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: measurement, analysis, and implications," in *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4. ACM, 2011, pp. 350–361.

[8] D. Cotroneo, L. De Simone, A. K. Iannillo, A. Lanzaro, R. Natella, J. Fan, and W. Ping, "Network function virtualization: Challenges and directions for reliability assurance," in *International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 2014, pp. 37–42.

[9] M. Penn, M. Polukarov, and M. Tennenholtz, "Congestion games with failures," in *Proceedings of the 6th ACM Conference on Electronic Commerce*. ACM, 2005, pp. 259–268.

[10] R. Meir, M. Tennenholtz, Y. Bachrach, and P. Key, "Congestion games with agent failures." in *AAAI*, vol. 12, 2012, pp. 1401–1407.

[11] Y. Li, Y. Jia, H. Tan, R. Wang, Z. Han, and F. C. M. Lau, "Congestion game with agent and resource failures," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 3, pp. 764–778, 2017.

[12] M. Chen, S. C. Liew, Z. Shao, and C. Kai, "Markov approximation for combinatorial network optimization," *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 6301–6327, 2013.

[13] R. W. Rosenthal, "A class of games possessing pure-strategy nash equilibria," *International Journal of Game Theory*, vol. 2, no. 1, pp. 65–67, 1973.

[14] D. Monderer and L. S. Shapley, "Potential games," *Games and economic behavior*, vol. 14, no. 1, pp. 124–143, 1996.

[15] D. Shah and J. Shin, "Dynamics in congestion games," *ACM SIGMETRICS Performance Evaluation Review*, vol. 38, no. 1, pp. 107–118, 2010.