Communication-Aware Scheduling of Serial Tasks for Dispersed Computing

Chien-Sheng Yang[†], Ramtin Pedarsani[*], and A. Salman Avestimehr[†]
[†] University of Southern California [*] University of California, Santa Barbara

***Abstract*—There is a growing interest in development of in-network *dispersed computing* paradigms that leverage the computing capabilities of heterogeneous resources dispersed across the network for processing massive amount of data is collected at the edge of the network. We consider the problem of task scheduling for such networks, in a dynamic setting in which arriving computation jobs are modeled as chains, with nodes representing tasks, and edges representing precedence constraints among tasks. In our proposed model, motivated by significant communication costs in dispersed computing environments, the communication times are taken into account. More specifically, we consider a network where servers are capable of serving all task types, and sending the results of processed tasks from one server to another server results in some communication delay that makes the design of optimal scheduling policy significantly more challenging than classical queueing networks. As the main contributions of the paper, we first characterize the capacity region of the network, then propose a novel virtual queueing network encoding the state of the network. Finally, we propose a Max-Weight type scheduling policy, and considering the virtual queueing network in the fluid limit, we use a Lyapunov argument to show that the policy is throughput-optimal.**

I. Introduction

In many large-scale data analysis application domains, such as surveillance, autonomous navigation, and cyber-security, much of the needed data is collected at the edge of the network via a collection of sensors, mobile platforms, and users' devices. In these scenarios, continuous transfer of the massive amount of collected data from edge of the network to back-end servers (e.g., cloud) for processing incurs significant communication and latency costs. As a result, there is a growing interest in development of in-network *dispersed computing* paradigms that leverage the computing capabilities of heterogeneous resources dispersed across the network (e.g., edge computing, fog computing, etc [1], [2]).

At a high level, a dispersed computing scenario consists of a group of networked computation nodes, such as wireless edge access points, network routers, and users' computers that can be utilized for offloading the computations. There is, however, a broad range of computing capabilities that may be supported by different computation nodes. Some may perform certain kinds of operations at extremely high rate, such as high throughput matrix multiplication on GPUs, while the same node may perform worse on single threaded performance. Communication bandwidth between different nodes in dispersed computing scenarios can also be very limited and heterogeneous. As a result, it is critical to design efficient algorithms for scheduling of computation tasks in such networks by carefully accounting for computation and communication heterogeneity.

In this paper, we consider the task scheduling problem in a dispersed computing network in which arriving jobs are modeled as chains, with nodes representing tasks, and edges representing precedence constraints among tasks. Each server is capable of serving all the task types and the service rate of a server depends on which task type it is serving. Our computation and network models are related to [3]. However, the model that we consider in this paper is more general, as the communication times between servers are taken into account. In our network model, sending the results of processed tasks from one server to another results in some communication constraints that makes the design of efficient scheduling policy even more challenging. More specifically, after one task is processed by a server, the server can either process the children task locally or send the result to another server in the network to continue with processing of the children task. However, each server has a bandwidth constraint that determines the delay for sending the results. Given this *communication-aware* task scheduling problem, we are interested in finding a throughput-optimal scheduling policy for the network.

A significant challenge in communication-aware scheduling is that unlike traditional queueing networks, processed tasks are not sent from one queue to another queue probabilistically. Indeed, the scheduling decisions also determine the routing of tasks in the network. Therefore, it is not clear what is the maximum throughput (or, equivalently, the capacity region) that one can achieve in such networks, and what scheduling policy is throughput-optimal.

As the main contributions of the paper, we first characterize the capacity region of this problem (i.e., the set of all arrival rate vectors of computations for which there exists a scheduling policy that makes the network rate stable). To capture the complicated computing and communication procedures in the network, we propose a novel virtual queueing network encoding the state of the network. Then, we propose a Max-Weight type scheduling policy for the virtual queueing network, and show that it is throughput-optimal. To the best of our knowledge, our work is the first throughput-optimal communication-aware scheduling policy for scheduling tasks with precedence constraints.

Since the proposed virtual queueing network is quite different from traditional queueing networks, it is not clear that the capacity region of the proposed virtual queueing network is equivalent to the capacity region of the original scheduling problem. Thus, to prove throughput-optimality Max-Weight policy, we first show the equivalence of two capacity regions: one for the dispersed computing problem that is characterized by an LP, and one for the virtual queueing network characterized by a mathematical optimization problem that is not an LP. Then, under the Max-Weight policy, we consider the stochastic network in the *fluid limit*, and using a Lyapunov argument, we show that the fluid model of the virtual queueing network is *weakly stable* [4] for all arrival vectors in the capacity region, and *stable* for all arrival vectors in the interior of the capacity region. This implies that the Max-Weight policy is throughput-optimal for the virtual queueing network as well as for the original scheduling problem.

Related Work: Task scheduling problem has been widely studied in the literature, which can be divided into two main categories: static scheduling and dynamic scheduling. In the static or offline scheduling problem, jobs are present at the beginning, and the goal is to allocate tasks to servers such that a performance metric such as average computation delay is minimized. In most cases, the static scheduling problem is computationally hard, and various heuristics or approximation algorithms are proposed (see e.g., [5], [6]).

In the dynamic or online scheduling problem, which is more relevant to this paper, jobs arrive to the network according to a stochastic process and get scheduled dynamically over time. In many prior works in the literature, the tasks have dedicated servers for processing, and the goal is to establish stability conditions for the network [7]. Given the stability results, the next natural goal is to compute the expected completion times of jobs or delay distributions. However, very few analytical results are available for characterizing the delay performance, except for the simplest models. When the tasks do not have dedicated servers, one aims to find a throughput-optimal scheduling policy (see e.g. [8]), i.e. a policy that stabilizes the network, whenever it can be stabilized. Max-Weight scheduling, proposed in [9], is known to be throughput-optimal for wireless networks and flexible queueing networks. However, there has been no work prior to our work that develops a Max-Weight type policy for *communication-aware* task scheduling.

II. System Model and Problem Formulation

A. Computation and Network Models

We consider the problem of dispersed computing, in which jobs arrive to a network in an online fashion, and jobs are processed across servers in the network. As shown in Fig. 1, each job is modeled as a chain including serial tasks. Each node of the chain represents one task type, and each edge of the chain represents a precedence constraint. Moreover, we consider M types of jobs, where each type is specified by one chain structure.

For this scheduling problem, we define the following terms. Let $(\mathcal{I}_m, \{c_k\}_{k\in\mathcal{I}_m})$ be the chain corresponding to the job of type m, $1 \leq m \leq M$, where $\mathcal{I}_m$ denotes the set of nodes of type-m jobs, and c_k denotes the output size (bits) of type-k task. Let the number of tasks of a type-m job be K_m, i.e. $|\mathcal{I}_m| = K_m$, and the total number of task types in the network be K, so that $\sum_{m=1}^{M} K_m = K$. Since $\mathcal{I}_1, \mathcal{I}_2, \dots, \mathcal{I}_m$ are disjoint, we can index the task types in the network by k, $1 \leq k \leq K$, starting from job type 1 to M. Therefore, task type-k belongs to job type $m(k)$ if $\sum_{m'=1}^{m(k)-1} K_{m'} < k \leq \sum_{m'=1}^{m(k)} K_{m'}$.

We call task $k^{'}$ a parent of task k if they belong to the same chain and there is a directed edge from $k^{'}$ to k. Without loss of generality, we let task k be the parent of task $k+1$, if task k and task $k+1$ belong to the same chain, i.e. $m(k) = m(k+1)$. In order to process task $k+1$, the processing of task k should be completed. Node k is said to be the root of chain type $m(k)$ if $k = 1 + \sum_{m'=1}^{m(k)-1} K_{m'}$. Also, node k is said to be the last node of chain type $m(k)$ if $k = \sum_{m'=1}^{m(k)} K_{m'}$. Then, we denote $\mathcal{H}$ as the set of the last nodes of the chains, i.e. $\mathcal{H} = \{k : k = \sum_{m=1}^{i} K_m,\ \forall\ 1 \leq i \leq M\}$.

In the dispersed computing network, as shown in Fig. 1, there are J servers that are connected to each other. Each server can serve all types of tasks. We consider the network in discrete time. We assume that the arrival process of jobs of type m is a Bernoulli process with rate λ_m, $0 < \lambda_m < 1$; that is, in each time slot a job of type m arrives to the network with probability λ_m independently over time. We assume that the service times for the nodes are geometrically distributed, independent across time slots and across different nodes, and also independent from the arrival processes. When server j processes type-k tasks, the service completion time has mean $\mu_{(k,j)}^{-1}$.[1] Thus, $\mu_{(k,j)}$ can be interpreted as the service rate of type-k task when processed by server j. Similarly, we model the communication times between two servers as geometric distribution, which are independent across time slots and across different nodes, and also independent from the arrival processes. When server j communicates data of size 1 bit to another server, the communication time has mean b_j^{-1}. Therefore, b_j can be interpreted as the average bandwidth (bits/time slot) of server j for communicating data of processed tasks.

B. Problem Formulation

Given the above system model, we formulate the dispersed computing problem based on the following terms.

Definition 1. A network is *rate stable* if the number of jobs in the network does not grow linearly with time.

Definition 2. We define the *capacity region* of the network to be the set of all arrival rate vectors where there exists a scheduling policy that makes the network rate stable.

Note that we later model the network as a network of virtual queues. Since the arrival and service processes are memoryless, given a scheduling policy, the queue-length vector in this virtual queueing network is a Markov process.

Definition 3. A network is *strongly stable* if its underlying Markov process is positive recurrent for all the arrival rate vectors in the interior of the capacity region.

[1]The exponential distribution of servers' processing times is commonly observed in many computing scenarios (see e.g. [10]), and the considered geometric distribution in this paper is the equivalent of exponential distribution for discrete-time systems.

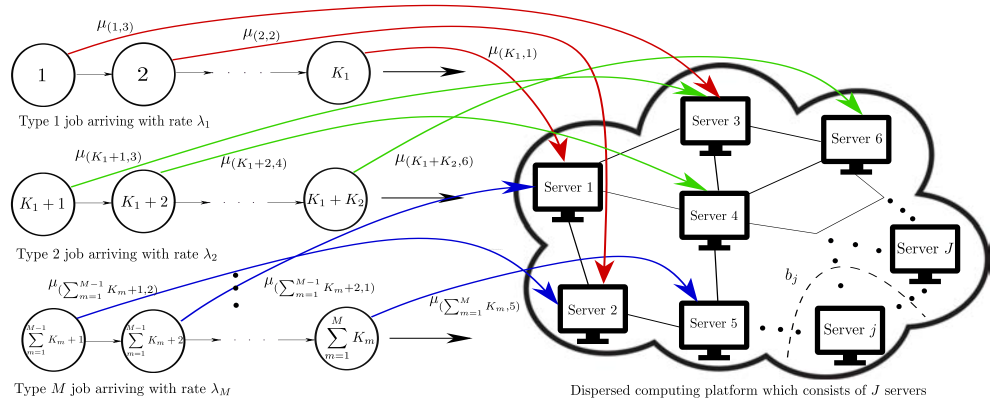
$\mu_{(1,3)}$
$\mu_{(2,2)}$
$\mu_{(K_1,1)}$
1
2
K_1
Type 1 job arriving with rate λ_1
$\mu_{(K_1+1,3)}$
$\mu_{(K_1+2,4)}$
$\mu_{(K_1+K_2,6)}$
K_1+1
K_1+2
K_1+K_2
Type 2 job arriving with rate λ_2
$\mu_{(\sum_{m=1}^{M-1} K_m+1,2)}$
$\mu_{(\sum_{m=1}^{M-1} K_m+2,1)}$
$\mu_{(\sum_{m=1}^{M} K_m,5)}$
$\sum_{m=1}^{M-1} K_m+1$
$\sum_{m=1}^{M-1} K_m+2$
$\sum_{m=1}^{M} K_m$
Type M job arriving with rate λ_M
Server 1
Server 2
Server 3
Server 4
Server 5
Server 6
Server J
Server j
b_j
Dispersed computing platform which consists of J servers

Fig. 1: Overview of task scheduling for dispersed computing. M classes of jobs come to a network which consists of J servers, where each job is specified by serial tasks. Type m job arrives to the network with rate λ_m. Type k tasks with output size c_k (bits) are processed by server j with service rate $\mu_{(k,j)}$. For communicating the data of processed tasks, server j has bandwidth b_j (bits/time slot).

Definition 4. A scheduling policy is *throughput-optimal* if, under this policy, the network is rate stable for all arrival rate vectors in the capacity region; and strongly stable for all arrival rate vectors in the interior of the capacity region.

Based on above definitions, our problem is now formulated as the following.

Problem. *Consider a dispersed computing network consisting of network and computation models as defined in Section II-A, we pose the following two questions:*

- *What is the capacity region of the network as defined in Definition 2?*
- *What is the throughput-optimal scheduling policy for the network as defined in Definition 4?*

III. Capacity Region Characterization

Before finding the throughput-optimal scheduling policy, we first characterize the capacity region of the network.

Now, we consider an arbitrary scheduling policy. Let $p_{(k,j)}$ be the long-run fraction of capacity that server j allocates for processing type-k tasks. Let $p = [p_{(k,j)}]$ be the capacity allocation vector. An allocation vector p is *feasible* if

$$\sum_{k=1}^{K} p_{(k,j)} \leq 1, \ \forall\, 1 \leq j \leq J. \tag{1}$$

Let $q_{(k,j)}$ be the long-run fraction of the bandwidth that server j allocates for communicating data of processed type-k tasks. We can define $q = [q_{(k,j)}]$ to be the bandwidth allocation vector. Therefore, an allocation vector q is feasible if

$$\sum_{k \in \{1,2,\dots,K\}\backslash\mathcal{H}} q_{(k,j)} \leq 1, \ \forall\, 1 \leq j \leq J. \tag{2}$$

Given a capacity allocation vector p, consider task k and task $k+1$ which are in the same chain on server j. As time t approaches infinity, $t \to \infty$, up to time t, the number of type-k tasks processed by server j is $\mu_{(k,j)}p_{(k,j)}t + o(t)$ and the number of type-$(k+1)$ tasks processed by server j is $\mu_{(k+1,j)}p_{(k+1,j)}t + o(t)$. Therefore, we can write

$$\mu_{(k,j)}p_{(k,j)}t - \mu_{(k+1,j)}p_{(k+1,j)}t + o(t) \tag{3}$$

as the number of type-$(k+1)$ tasks that server j is not capable of serving up to time t, as $t \to \infty$. Clearly, the type-$(k+1)$ tasks which cannot be served by server j has to be processed by other servers. Hence, up to time t and $t \to \infty$, server j has to at least communicate data of $\mu_{(k,j)}p_{(k,j)}t - \mu_{(k+1,j)}p_{(k+1,j)}t + o(t)$ processed type-k tasks to other servers.

On the other hand, given a bandwidth allocation vector q, up to time t and $t \to \infty$, the number of the type-k tasks communicated by server j is $\frac{b_j q_{(k,j)} t}{c_k} + o(t)$. Therefore, to make the network stable, we obtain the following constraints:

$$\frac{b_j q_{(k,j)}}{c_k} \geq \mu_{(k,j)}p_{(k,j)} - \mu_{(k+1,j)}p_{(k+1,j)}, \tag{4}$$

for $\forall\ 1 \leq j \leq J$ and $\forall\ k \in \{1,2,\dots,K\}\backslash\mathcal{H}$.

We introduce a linear program (LP) that characterizes the capacity region of the network. The nominal traffic rate to all the tasks of job type m in the network is λ_m. Let $\nu = [\nu_k] \in \mathbb{R}_+^K$ be the set of nominal rates of tasks in the network. Then, $\nu_m = \lambda_m$ if $m(k) = m$, i.e., if $\sum_{m'=1}^{m-1} K_{m'} \leq k \leq \sum_{m'=1}^{m} K_{m'}$. The LP that characterizes capacity region of the network makes sure that the total service capacity allocated to each node in the network is at least as large as the nominal traffic rate to that node, and the communication rate of each server is at least as large as the rate of tasks that the server is not capable of serving. Thus, the LP known as the *static planning problem (SPP)* [11] - is defined as follows:

Static Planning Problem (SPP):

$$\text{Maximize} \quad \delta \tag{5}$$

$$\text{subject to} \quad \nu_k \leq \sum_{j=1}^{J} \mu_{(k,j)}p_{(k,j)} - \delta, \ \forall k. \tag{6}$$

$$\frac{b_j q_{(k,j)}}{c_k} - \delta \geq \mu_{(k,j)}p_{(k,j)} - \mu_{(k+1,j)}p_{(k+1,j)},$$
$$\forall j, \ \forall k \in \{1,2,\dots,K\}\backslash\mathcal{H}. \tag{7}$$

$$1 \geq \sum_{k=1}^{K} p_{(k,j)}, \ p_{(k,j)} \geq 0, \ \forall j. \tag{8}$$

$$1 \geq \sum_{k \in \{1,2,\dots,K\}\backslash\mathcal{H}} q_{(k,j)}, \ q_{(k,j)} \geq 0, \ \forall j. \tag{9}$$

Based on SPP above, the capacity region of the network can be characterized by following proposition.

Proposition 1. *The capacity region Λ of the network characterizes the set of all rate vectors $\lambda \in \mathbb{R}_+^M$ for which the corresponding optimal solution δ^* to the static planning problem (SPP) satisfies $\delta^* \geq 0$.*

The proof of Proposition 1 follows the standard fluid limit argument for stochastic networks [4], provided in [12].

IV. Queueing Network Model

Based on the computation model and network model described in Section II, we first illustrate how we model a network of virtual queues that encodes the state of the network. The virtual queueing network consists of two kinds of queues, *processing queue* and *communication queue*, which are modeled in the following manner:

1) **Processing Queue**: We maintain one virtual queue called (k, j) for type-k tasks which are processed at server j.
2) **Communication Queue**: For $k \notin \mathcal{H}$, we maintain one virtual queue called $(k, j), c$ for processed type-k tasks to be sent to other servers by server j.

Now, we describe the dynamics of the virtual queues in the network. Let's consider one type of job which consists of serial tasks. A root task k of the job is sent to processing queue (k, j) if the task k is scheduled on server j when a new job comes to the network. For any node k in this chain, the result of process in queue (k, j) is sent to queue $(k+1, j)$ if task $k+1$ is scheduled on server j. Otherwise, the result is sent to communication queue $(k, j), c$. If task $k+1$ in queue $(k, j), c$ is scheduled on server l, it is sent to queue $(k+1, l)$, where $l \in \{1, 2, \ldots, J\}\backslash\{j\}$. Clearly, if k is a root of one chain, the traffic to queue (k, j) is only the traffic of type-$m(k)$ jobs coming to the network. Otherwise, the traffic to queue (k, j) is from queue $(k-1, j)$ and queues $(k-1, l), c, \forall\, l \in \{1, 2, \ldots, J\}\backslash\{j\}$. Furthermore, the traffic to queue $(k, j), c$ is only from queue (k, j), where $k \in \{1, 2, \ldots, K\}\backslash\mathcal{H}$.

Let $Q_{(k,j)}$ denote the length of queue (k, j) and $Q_{(k,j),c}$ denote the length of queue $(k, j), c$. A task of type k can be processed by server j iff $Q_{(k,j)} > 0$ and a processed task of type k can be sent by server j to other servers iff $Q_{(k,j),c} > 0$. Let $d^n_{(k,j)} \in \{0, 1\}$ be the number of processed tasks of type k by server j at time n, $a^n_{m(k)} \in \{0, 1\}$ be the number of jobs of type m that arrives to the network at time n and $d^n_{(k,j),c} \in \{0, 1\}$ be the number of processed type-k tasks sent to other servers at time n. We denote $u^n_{m\to j} \in \{0, 1\}$ as the decision variable that root task of the type-m job is scheduled on server j at time n, $w^n_{k,j\to l} \in \{0, 1\}$ as the decision variable that processed type-k task in queue $(k, j), c$ is sent to queue $(k+1, l)$ at time n, and $s^n_{k,j\to j} \in \{0, 1\}$ as the decision variable that processed type-k task in queue (k, j) is sent to queue $(k+1, j)$ at time n. Thus, we state the dynamics of the queueing network. If k is a root node of the chain, then

$$Q^{n+1}_{(k,j)} = Q^n_{(k,j)} + a^n_{m(k)} u^n_{m(k)\to j} - d^n_{(k,j)}; \tag{10}$$

else,

$$Q^{n+1}_{(k,j)} = Q^n_{(k,j)} + \sum_{1\leq l\leq J,\ l\neq j} d^n_{(k-1,l),c} w^n_{k-1,l\to j} + d^n_{(k-1,j)} s^n_{k-1,j\to j} - d^n_{(k,j)}. \tag{11}$$

For all $k \in \{1, 2, \ldots, K\}\backslash\mathcal{H}$,

$$Q^{n+1}_{(k,j),c} = Q^n_{(k,j),c} + d^n_{(k,j)}(1 - s^n_{k,j\to j}) - d^n_{(k,j),c}. \tag{12}$$

Before we introduce an optimization problem that characterizes the capacity region of the described queueing network, we first define the following terms.

Consider an arbitrary scheduling policy. Let $u_{m\to j}$ be the long-run fraction that root tasks of the type-m job are scheduled on server j. We define $u = [u_{m\to j}]$ to be the root-node allocation vector. A root-node allocation vector u is feasible if

$$\sum_{j=1}^{J} u_{m\to j} = 1, \quad \forall\ 1 \leq m \leq M. \tag{13}$$

For the type-k tasks served by server j, we denote $s_{k,j\to j}$ as the long-run fraction that their child tasks (type-$(k+1)$) are scheduled on server j. An allocation vector $s = [s_{k,j\to j}]$ is feasible if

$$0 \leq s_{k,j\to j} \leq 1,\ \forall\ j,\ \forall\ k \in \{1, 2, \ldots, K\}\backslash\mathcal{H}. \tag{14}$$

For the outputs of process in virtual queue $(k, j), c$, we denote $w_{k,j\to l}$ as the long-run fraction that they are sent to queue $(k+1, l)$. An allocation vector $w = [w_{k,j\to l}]$ is feasible if

$$\sum_{1\leq l\leq J,\ l\neq j} w_{k,j\to l} = 1,\ \forall\ j,\ \forall\ k \in \{1, 2, \ldots, K\}\backslash\mathcal{H}. \tag{15}$$

For the type-k tasks processed by server j, we define $f_{k,j\to l}$ as the long-run fraction that their child tasks are scheduled on server l. Given s and w, we can write $f_{k,j\to l}$ as follows:

$$f_{k,j\to l} = \begin{cases} s_{k,j\to j}, & \text{if } l = j \\ (1 - s_{k,j\to j}) w_{k,j\to l}, & \text{otherwise} \end{cases} \tag{16}$$

Let $r_{(k,j)}$ and $r_{(k,j),c}$ denote the nominal rates to the virtual queues (k, j) and $(k, j), c$ respectively. If k is a root of one chain, the nominal rate $r_{(k,j)}$ can be written as

$$r_{(k,j)} = \lambda_{m(k)} u_{m(k)\to j}. \tag{17}$$

If k is not a root of one chain, the rate $r_{(k,j)}$ can be obtained by summing $r_{(k-1,l)}$ with $f_{k-1,l\to j}$ over all servers, i.e.

$$r_{(k,j)} = \sum_{l=1}^{J} r_{(k-1,l)} f_{k-1,l\to j}, \tag{18}$$

because of flow conservation. Similarly,

$$r_{(k,j),c} = r_{(k,j)}(1 - s_{k,j\to j}). \tag{19}$$

Now, we introduce an optimization problem called *queueing network planning problem (QNPP)* that characterizes the capacity region of the virtual queueing network. The problem is defined as follows:

Queueing Network Planning Problem (QNPP):

$$\text{Maximize} \quad \gamma \tag{20}$$

$$\text{subject to} \quad r_{(k,j)} \leq \mu_{(k,j)} p_{(k,j)} - \gamma,\ \forall\ j,\ \forall\ k. \tag{21}$$

$$r_{(k,j),c} \leq \frac{b_j q_{(k,j)}}{c_k} - \gamma,\ \forall\ j,\ \forall\ k \in \{1, 2, \ldots, K\}\backslash\mathcal{H}. \tag{22}$$

and subject to allocation vectors being feasible, where $r_{(k,j)}$ and $r_{(k,j),c}$ are defined in (17)-(19). Note that all the allocation vectors p, q, u, s, w are feasible if (1), (2), (13)-(15) are satisfied. The capacity region Λ' of the virtual queueing network is the set of all λ for which the solution of QNPP satisfies $\gamma \geq 0$. This can be proved similar to Proposition 1.

V. Throughput-Optimal Policy

In this section, we propose Max-Weight scheduling policy for the network of virtual queues in Section IV and show that it is throughput-optimal for the network.

We give a description of the Max-Weight policy for the proposed virtual queueing network. Given virtual queue-lengths $Q^n_{(k,j)}$ and $Q^n_{(k,j),c}$ at time n, Max-Weight policy allocates the vectors p, q, u, s and w that are

$$\arg\min_{p,q,u,s,w} \Big((Q^n)^T E[\Delta Q^n|\mathcal{F}^n] + (Q^n_c)^T E[\Delta Q^n_c|\mathcal{F}^n] \Big),$$

where $Q^n = [Q^n_{(k,j)}]$ and $Q^n_c = [Q^n_{(k,j),c}]$ are the vectors of queue-lengths at time n. The Max-Weight policy is the choice of p, q, u s and w that minimizes the drift of a Lyapunov function $V^n = \sum_{k,j}(Q^n_{(k,j)})^2 + \sum_{k,j}(Q^n_{(k,j),c})^2$.

The following theorem shows the throughput-optimality of Max-Weight policy.

Theorem 1. *Max-Weight policy is throughput-optimal for the network, i.e. Max-Weight policy is rate stable for all the arrival vectors in the capacity region Λ defined in Proposition 1, and it makes the underlying Markov process positive recurrent for all the arrival rate vectors in the interior of Λ.*

Sketch of Proof: In order to prove Theorem 1, we first state the following lemma whose proof can be found in [12].

Lemma 1. *The capacity region characterized by static planning problem is equivalent to the capacity region characterized by queueing network planning problem, i.e. $\Lambda = \Lambda'$.*

Having Lemma 1, we now show that the queueing network is rate stable for all $\lambda \in \Lambda'$, and strongly stable for all λ in the interior of Λ' under Max-Weight policy.

We consider the problem in the fluid limit. Define the amount of fluid in queue (k, j) as $X_{(k,j)}(t)$ and the amount of fluid in queue $(k, j), c$ as $X_{(k,j),c}(t)$. If k is the root of a chain, the dynamics of the fluid are as follows

$$X_{(k,j)}(t) = X_{(k,j)}(0) + \lambda_{m(k)} u_{m(k)\to j} t - \mu_{(k,j)} p_{(k,j)} t; \tag{23}$$

else,

$$X_{(k,j)}(t) = X_{(k,j)}(0) + \Big(\sum_{1\le l\le J,\ l\ne j} \frac{b_l q_{(k-1,l)}}{c_{k-1}} w_{k-1,l\to j} + \mu_{(k-1,j)} p_{(k-1,j)} s_{k-1,j\to j} - \mu_{(k,j)} p_{(k,j)} \Big) t \tag{24}$$

For any $k \in \{1, 2, \ldots, K\}\backslash\mathcal{H}$, we have

$$X_{(k,j),c}(t) = X_{(k,j),c}(0) + (1 - s_{k,j\to j}) \mu_{(k,j)} p_{(k,j)} t - \frac{b_j q_{(k,j)}}{c_k} t \tag{25}$$

Now, we define γ^* as the optimal value of QNPP. If we consider a vector λ in the interior of Λ', then $\gamma^* > 0$. It follows that there exist p^*, q^*, u^*, s^*, w^*, $\epsilon_{(k,j)}$ and $\epsilon_{(k,j),c}$ such that for $\forall\ 1 \le j \le J$ and $\forall\ 1 \le k \le K$,

$$\mu_{(k,j)} p^*_{(k,j)} = r^*_{(k,j)} + \epsilon_{(k,j)}; \tag{26}$$

and for $\forall\ 1 \le j \le J$ and $\forall\ k \in \{1, 2, \ldots, K\}\backslash\mathcal{H}$,

$$\frac{b_j q^*_{(k,j)}}{c_k} = r^*_{(k,j),c} + \epsilon_{(k,j),c} \tag{27}$$

where $\epsilon_{(k,j)} = \frac{(k - \sum_{m'=1}^{m(k)-1} K_{m'} - 1)J + 1}{(K_{m(k)} - 1)J + 1} \gamma^*$ and $\epsilon_{(k,j),c} = \frac{1}{(K_{m(k)}-1)J+1}\gamma^*$.

From (17)-(19), (26) and (27), we have $\dot{X}^*_{(k,j)}(t) < 0$ and $\dot{X}^*_{(k,j),c}(t) < 0$. Then, we take $V(t) = \frac{1}{2}X^T(t)X(t) + \frac{1}{2}X^T_c(t)X_c(t)$ as the Lyapunov function where $X(t) = [X_{(k,j)}(t)]$ and $X_c(t) = [X_{(k,j),c}(t)]$. The drift of V by using Max-Weight policy is

$$\dot{V}_{\text{Max-Weight}}(t) = \min_{p,q,u,s,w} X^T(t)\dot{X}(t) + X^T_c(t)\dot{X}_c(t) \tag{28}$$

$$\le (X^*)^T(t)\dot{X}^*(t) + (X^*_c)^T(t)\dot{X}^*_c(t) < 0. \tag{29}$$

Thus, we show that $\dot{V}_{\text{Max-Weight}}(t) < 0$ if λ in the interior of Λ'. This proves that the fluid model is stable which implies the positive recurrence of the underlying Markov chain [4].

If we consider a vector $\lambda \in \Lambda'$, there exist allocation vectors p^*, q^*, u^*, s^* and w^* such that $\mu_{(k,j)} p^*_{(k,j)} = r^*_{(k,j)}$ and $\frac{b_j q^*_{(k,j)}}{c_k} = r^*_{(k,j),c}$. Similarly, one can show that $\dot{V}_{\text{Max-Weight}}(t) \le 0$ which implies $X(t) = \vec{0}$ and $X_c(t) = \vec{0}$ if $X(0) = \vec{0}$ and $X_c(0) = \vec{0}$. It proves that the fluid model is weakly stable, i.e. the queueing network process is rate stable [4]. Hence, Max-Weight policy is throughput-optimal for the queueing network. The full proof can be found in [12]

VI. Acknowledgment

This material is based upon work supported by Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001117C0053. The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. This work is also in part supported by ONR award N000141612189 and NSF Grants CCF-1703575 and NeTS-1419632 and the UC Office of President under grant No. LFR-18-548175.

References

[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the internet of things,” in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pp. 13–16, ACM, 2012.

[2] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, “Mobile edge computing—a key technology towards 5g,” *ETSI White Paper*, vol. 11, no. 11, pp. 1–16, 2015.

[3] R. Pedarsani, J. Walrand, and Y. Zhong, “Robust scheduling for flexible processing networks,” *Advances in Applied Probability*, vol. 49, 2017.

[4] J. G. Dai, “On positive harris recurrence of multiclass queueing networks: a unified approach via fluid limit models,” *The Annals of Applied Probability*, 1995.

[5] Y.-K. Kwok and I. Ahmad, “Static scheduling algorithms for allocating directed task graphs to multiprocessors,” *ACM Computing Surveys (CSUR)*, vol. 31, no. 4, pp. 406–471, 1999.

[6] Y.-H. Kao, B. Krishnamachari, M.-R. Ra, and F. Bai, “Hermes: Latency optimal task assignment for resource-constrained mobile computing,” *IEEE Transactions on Mobile Computing*, 2017.

[7] F. Baccelli, W. A. Massey, and D. Towsley, “Acyclic fork-join queuing networks,” *Journal of the ACM (JACM)*, vol. 36, no. 3, 1989.

[8] A. Eryilmaz, R. Srikant, and J. R. Perkins, “Stable scheduling policies for fading wireless channels,” *IEEE/ACM Transactions on Networking*, vol. 13, no. 2, pp. 411–424, 2005.

[9] J. G. Dai and W. Lin, “Maximum pressure policies in stochastic processing networks,” *Operations Research*, vol. 53, no. 2, 2005.

[10] A. Reisizadeh, S. Prakash, R. Pedarsani, and S. Avestimehr, “Coded computation over heterogeneous clusters,” in *Information Theory (ISIT), 2017 IEEE International Symposium on*, pp. 2408–2412, IEEE, 2017.

[11] J. M. Harrison, “Brownian models of open processing networks: Canonical representation of workload,” *Annals of Applied Probability*, 2000.

[12] C.-S. Yang, R. Pedarsani, and A. S. Avestimehr, “Communication-aware scheduling of serial tasks for dispersed computing,” *arXiv preprint arXiv:1804.06468*, 2018.