A Comparative Analysis of Server Selection in Content Replication Networks

Tao Wu and David Starobinski

***Abstract*— Server selection plays an essential role in content replication networks, such as peer-to-peer (P2P) and content delivery networks (CDNs). In this paper, we perform an analytical investigation of the strengths and weaknesses of existing server selection policies, based initially on an $M/G/1$ Processor Sharing (PS) queueing-theoretic model. We develop a theoretical benchmark to evaluate the performance of two general server selection policies, referred to as EQ_DELAY and EQ_LOAD, which characterize a wide range of existing server selection algorithms. We find that EQ_LOAD achieves an average delay always higher than or equal to that of EQ_DELAY. A key theoretical result of this paper is that in an N-server system, the worst-case ratio between the average delay of EQ_DELAY or EQ_LOAD and the minimal average delay (obtained from the benchmark) is precisely N. We constructively show how this worst-case scenario can arise in highly heterogeneous systems. This result, when interpreted in the context of selfish routing, means that the price of anarchy in unbounded delay networks depends on the topology, and can potentially be very large. Our analytical findings are extended in asymptotic regimes to the $G/G/1$ First-Come First-Serve and multi-class $M/G/1$-PS models and supported by simulations run for various arrival and service processes, scheduling disciplines, and workload exhibiting temporal locality. These results indicate that our analysis is applicable to realistic scenarios.**

***Index Terms*— Content delivery networks, peer-to-peer networks, load balancing, distributed systems, price of anarchy, game theory.**

I. Introduction

Content replication has emerged as one of the most useful paradigms for the provision of scalable and reliable Internet services [1]. With content replication, the same data (e.g., Web pages, multimedia files, etc.) is stored at multiple geographically distant servers. Requests by clients are then forwarded to one of these servers. Because of its inherent scalability and fault-tolerance, content replication has become a cornerstone of most modern networking architectures, including content delivery networks (CDNs) and peer-to-peer (P2P) networks [2, 3].

One of the key issues arising with content replication is that of server selection. In most content replication networks, a number of *server-selection nodes* (e.g., enhanced DNS servers in CDNs [2, 4] or supernodes in P2P networks [3]) are responsible for aggregating incoming client requests and forwarding them to one of the servers. Given the geographical span and the scale of content replication networks, server selection is fundamentally different from traditional load balancing problems, which usually assume that servers are co-located [5].

Because server response time is an increasing function of the load, the best server selection solution is usually not the one that directs all the requests to a single server (e.g., the fastest), which would slow down or even crash the server [6]. Thus, a number of server selection policies have been proposed in the literature or implemented in commercial products. Several of these policies fall within one of the following two categories: (i) Equal load (EQ_LOAD), where the access probabilities to the servers are set so that all servers have the same utilization. Examples of policies following this approach include round-robin, or weighted-round-robin (WRR) for heterogeneous servers [5], and certain adaptive algorithms such as Least Loaded [4] and WebSeAl [7]; (ii) Equal delay (EQ_DELAY), where the access probabilities are set so that the average delay at all the selected servers is equal or at least on the same order. SPAND [8] and the application layer anycast architecture of [6] implement variations of this approach.

Although server selection is key to content replication networks' performance, existing evaluations of the above server selection policies have mostly been obtained from parameterized simulations (e.g., [9, 10]) due to the inherent complexity of modeling such networks. However, in the absence of theoretical guidelines, it is unknown to what extent experimental results obtained from a specific network configuration can be applied to other settings. Such guidelines can provide essential analytical understanding of the following important questions: 1) How do the above server selection policies compare to each other in terms of system-level performance (e.g., average delay)? 2) How do factors such as network size, server capacities and network utilization affect their performance? 3) What are the theoretical performance limitations and bounds of these policies?

In this paper, we develop an analytical benchmark to address these questions. We first model the behavior of the servers using an $M/G/1$-PS (Processor Sharing) queueing-theoretic model [11], and derive *closed-form* solutions for a theoretical benchmark policy called OPT that minimizes the average delay. We analytically compare EQ_DELAY and EQ_LOAD's average delays against OPT, and asymptotically extend the results (at high load) to more general arrival processes and service disciplines, including $G/G/1$-FCFS and multi-class $M/G/1$-PS. Additionally, we simulate the policies performance under various service discipline and workload conditions, e.g., based on synthetic traces generated by the ProWGen Web workload generator [12].

The main results of this paper are as follows: 1) The average

Tao Wu is with Nokia Research Center, Cambridge, MA. David Starobinski is with the ECE Department at Boston University.

This work was supported in part by NSF CAREER grant ANI-0132802. Preliminary findings of this work were presented in the Networking 2005 Conference and GameNets 2006 Workshop.

delay of EQ_LOAD is always higher than or equal to that of EQ_DELAY. In a system consisting of N servers, EQ_LOAD's average delay can be as much as N times that of EQ_DELAY. However, the difference between the performance of these two policies vanishes when the load increases. 2) Both EQ_DELAY and EQ_LOAD perform sub-optimally compared to the benchmark policy, OPT. A key analytical result developed in this paper is that, for an N-server system, the worst-case ratio between the average delay of EQ_DELAY or EQ_LOAD and the minimal average delay achieved by the OPT policy is exactly N. Thus, the potential inefficiency of EQ_DELAY and EQ_LOAD increases with the number of servers and can be very large. 3) The highest inefficiency of EQ_DELAY and EQ_LOAD occurs in heterogeneous systems. Thus, networks whose nodes tend to have very different processing power and connectivity may benefit from more efficient server selection policies. 4) The above results appear to be quite insensitive to inter-arrival and service time distributions, service disciplines and multi-class network settings. They are validated by asymptotic analysis as well as simulations with correlated arrivals, various service disciplines and multi-class networks.

Our results have an interesting game-theoretic interpretation. In the context of selfish routing, EQ_DELAY is the equilibrium outcome (i.e., the Wardrop equilibrium) of a large number of selfish clients competing for server resources with the goal of minimizing their own respective delays [13–16]. The worst-case ratio between the average delays of the EQ_DELAY policy and OPT is often referred to as *the price of anarchy* as a key indicator of system inefficiency induced by selfish behavior. Most existing bounds of the price of anarchy are based on bounded or constant delay functions, and indicate modest performance degradation that is *topology independent* [14, 17]. For unbounded delay functions that characterize most computer networks, the price of anarchy is much more difficult to determine [18]. In this paper, we show that the price of anarchy with $M/G/1$-PS networks depends on the topology in the sense that it is exactly the number of servers (or links) N. Compared to known results [18], our result is both *tight* (i.e., the lower and upper bounds match each other) and *general* (the result holds for any feasible load condition).

The remainder of the paper is organized as follows. In Section II, we introduce our model and notations. In Section III, we derive the benchmark server selection policy and obtain a closed-form expression for the minimal average delay. We also analyze the performance of EQ_DELAY and EQ_LOAD policies. In Section IV, we analytically compare the three policies, and derive matching upper and lower bounds of the ratio of EQ_DELAY and EQ_LOAD's average delays to that of the benchmark policy. In Section V, we extend our analytical results to $G/G/1$-FCFS and multi-class $M/G/1$-PS models. We present our simulation results in Section VI, and conclude the paper in Section VII.

II. Model and Problem

A. Notation

We consider a network consisting of a number of clients, M server selection nodes (SSNs) and N servers. Each server

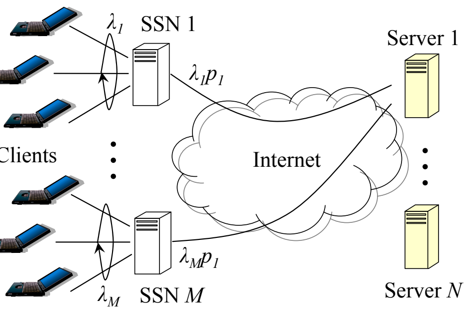


Fig. 1. System Model.

selection node j forwards requests generated by its clients following a Poisson process with rate λ_j. Further, we assume that the requests generated by different server selection nodes are independent. Therefore, the aggregate request process is also Poisson with rate $\lambda = \sum_{j=1}^{M} \lambda_j$. Each server selection node j assigns request to server i with probability p_i, independently of other requests. Therefore, the arrival process to each server i, $i = 1, 2, \ldots, N$, is an independent Poisson process with rate $p_i\lambda$. This scenario is illustrated in Fig. 1.

We assume that the service time distribution at each server i is arbitrary with mean $\bar{x}_i$. The mean service rate of server i is $\mu_i = 1/\bar{x}_i$. In Sections III and IV, we assume that each server implements a *Processor Sharing* (PS) scheduling policy, where all the requests share the server's capacity equally and continuously [19].

Under the above assumptions, each server behaves as an $M/G/1$-PS queue. In such a network, the average delay of a request forwarded to server i is therefore given by the following expression [19]

$$\overline{T}_i(p_i) = \frac{1}{\mu_i - p_i\lambda}. \tag{1}$$

We note that the average delay depends on the service time distribution only through its mean.

Let $\mathbf{p} = (p_1, p_2, \ldots, p_N)'$ denote the server access probability vector. Then, for a given vector $\mathbf{p}$, the average delay of a request in the system is

$$\overline{T}(\mathbf{p}) = \sum_{i=1}^{N} p_i \overline{T}_i(p_i) = \sum_{i=1}^{N} \frac{p_i}{\mu_i - p_i\lambda}. \tag{2}$$

Any feasible vector $\mathbf{p}$ must satisfy several conditions. First, to be a legitimate probability vector, the coordinates of $\mathbf{p}$ must be non-negative and sum to one, that is $p_i \geq 0$ for all i and $\sum_{i=1}^{N} p_i = 1$. In addition, the coordinates of $\mathbf{p}$ must satisfy the *individual stability conditions*, i.e., $p_i < \mu_i/\lambda$, to guarantee that the arrival rate of requests is smaller than the service rate at each server i. We denote by $\mathcal{P}$ the set of vectors $\mathbf{p}$ that satisfy all the above constraints. It is straightforward to show that the set $\mathcal{P}$ is non-empty if and only if the *aggregate stability condition* $\lambda < \sum_{i=1}^{N} \mu_i$ is satisfied.

B. Model Justification

The Poisson arrival assumption is theoretically justified by the fact that it represents the aggregation of requests made by a large population of clients [20]. Measurements of request arrivals to Web servers have been shown to match well a Poisson process, at least over small to moderate timescales [11].

Processor Sharing is an idealization of the round-robin policy implemented by most existing Web servers, whereby each request is given an equal share of the service capacity [21]. Indeed, one can derive Processor Sharing from round-robin by decreasing the quantum of time allocated to each request at each round to an infinitesimal value [11, 19].

Perhaps the most notable assumption we make is that the end-to-end delay that a request experiences is dominated by server delays. This assumption is made to maintain the model's analytical tractability, but it also offers a reasonable abstraction for many practical scenarios in content replication networks. For example, recent studies have shown that, with backbone over-provisioning, end-to-end bottlenecks increasingly occur in servers rather than in the backbone [10, 22–25].

It is important to point out that several of the above assumptions are going to be relaxed in Sections V, where we will consider the case of general inter-arrival time distributions, another service discipline, namely, First-Come First-Serve, and multi-class settings that capture the fact that different SSNs may favor different servers. We will show that, in certain asymptotic load regimes, the results derived for the $M/G/1$-PS apply to these extended models as well.

III. Analyzing the Benchmark, EQ_DELAY and EQ_LOAD Policies

In this section, we derive closed-form expressions for the average delay and probability vector for the benchmark policy, as well as those for EQ_DELAY and EQ_LOAD, under the $M/G/1$-PS model.

A. Derivation of The Benchmark Policy

Our benchmark policy, denoted as OPT, achieves the minimal average delay in the network introduced in Section II. The problem of deriving OPT can be formally stated as follows:

Problem 1 (OPT): Find the optimal server access probability vector

$$\mathbf{p}^* = \arg\min_{\mathbf{p}\in\mathcal{P}} \overline{T}(\mathbf{p}),$$

where $\overline{T}(\mathbf{p})$ is as defined in Eq. (2).

Problem 1 is an optimization problem, and has already been the subject of studies in the literature under the general context of load sharing in queueing networks [26–28] as well as flow assignment [29]. Our contribution here is to provide *closed-form* expressions for the optimal server access probabilities for the specific case of $M/G/1$-PS servers. These expressions will be used to prove our main results in Sections IV and V that compare between the performance of EQ_DELAY, EQ_LOAD and OPT.

As a first step to obtain the server access probabilities for Problem 1, we note that there exists a unique solution since $\overline{T}(\mathbf{p})$ is strictly convex over $\mathcal{P}$. Next, in order to solve the constrained optimization problem, we make use of Lagrange multiplier techniques [30]. We start by defining the Lagrangian function

$$\begin{aligned} L(\mathbf{p}, l, \mathbf{m}) &= \overline{T}(\mathbf{p}) + l(\sum_{i=1}^{N} p_i - 1) - \sum_{i=1}^{N} m_i p_i \\ &= \sum_{i=1}^{N} \frac{p_i}{\mu_i - p_i\lambda} + l(\sum_{i=1}^{N} p_i - 1) - \\ & \sum_{i=1}^{N} m_i p_i, \end{aligned} \quad (3)$$

where l and $\mathbf{m} = (m_1, m_2, \cdots, m_N)$ are the so-called Lagrange multipliers. The Lagrange multiplier l enforces the equality constraint $\sum_{i=1}^{N} p_i = 1$, while the multipliers m_i enforce the inequality constraints $p_i \geq 0$. In the sequel, we denote by $I(\mathbf{p})$ the set of *inactive* servers to which requests are never forwarded, that is, $I(\mathbf{p}) = \{i \mid p_i = 0\}$.

Since $\overline{T}(\mathbf{p})$ is strictly convex and the set of constraints is convex as well, the well-known Karush-Kuhn-Tucker (KKT) conditions are both necessary and sufficient for the existence of a global minimum $\mathbf{p}^*$, assuming that the individual stability conditions are satisfied [30].

Without loss of generality, we assume that the service rate vector $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_N)'$ is descending, i.e., $\mu_1 \geq \mu_2 \geq \ldots \geq \mu_N$. We now observe that in OPT, a faster server should always serve a larger fraction of requests than a slower server. Therefore, the access probabilities must satisfy the following ordering $p_1^* \geq p_2^* \geq \cdots \geq p_N^*$. As a result, the set of inactive servers $I(\mathbf{p}^*)$ must be one of the following: $\emptyset, \{N\}, \{N-1, N\}, \ldots, \{2, 3, \ldots, N\}$.

Let us denote the slowest active server in OPT as N^*, that is, $I(\mathbf{p}^*) = \{N^*+1, N^*+2, \ldots, N\}$. The following theorem establishes the value of N^* and provides a closed-form expression for the optimal solution $\mathbf{p}^*$. This theorem is proven by showing that the optimal solution satisfies all the KKT conditions as well as the individual stability conditions. Due to space limitation, the complete proof is omitted here but can be found in [31].

Theorem 1: The optimal solution $\mathbf{p}^*$ to Problem 1 can be obtained as follows. Define

$$\alpha_i = \frac{\mu_i}{\lambda} - \frac{\left(\sum_{j=1}^{i} \mu_j - \lambda\right)\sqrt{\mu_i}}{\lambda \sum_{j=1}^{i} \sqrt{\mu_j}}, \quad 0 \leq i \leq N. \quad (4)$$

Then,

1) N^* is the maximum index i for which $\alpha_i > 0$,
2) $I(\mathbf{p}^*) = \{N^*+1, N^*+2, \ldots, N\}$,
3) $p_i^* = \frac{\mu_i}{\lambda} - \frac{\left[\sum_{j=1}^{N^*} \mu_j - \lambda\right]\sqrt{\mu_i}}{\lambda \sum_{j=1}^{N^*} \sqrt{\mu_j}}, \quad \forall i \notin I(\mathbf{p}^*)$,
4) $p_i^* = 0, \quad \forall i \in I(\mathbf{p}^*)$.

We can now determine the average delay achieved by the optimal policy, by substituting the derived expression of $\mathbf{p}^*$ into Eq. (2):

$$\overline{T}(\mathbf{p}^*) = \sum_{i=1}^{N^*} \frac{p_i^*}{\mu_i - p_i^*\lambda} = \frac{\left(\sum_{i=1}^{N^*} \sqrt{\mu_i}\right)^2}{\lambda\left(\sum_{i=1}^{N^*} \mu_i - \lambda\right)} - \frac{N^*}{\lambda}. \quad (5)$$

B. Performance Analysis of the EQ_DELAY Policy

The goal of the EQ_DELAY policy is to set the probability access vector such that the average delay at all active servers is the same and minimal. To formalize the problem, define the set

$$\mathcal{S} = \{\mathbf{p} \mid \overline{T}_i(p_i) = \overline{T}_j(p_j), \quad \forall i, j \notin I(\mathbf{p})\}. \tag{6}$$

We can then formulate the optimization problem for the EQ_DELAY policy as follows:

Problem 2 (EQ_DELAY): Find the server access probability vector

$$\hat{\mathbf{p}} = \arg \min_{\mathbf{p} \in (\mathcal{P} \cap \mathcal{S})} \left(\overline{T}(\mathbf{p})\right).$$

As with Problem 1, it is fairly easy to verify that the set of inactive servers $I(\hat{\mathbf{p}})$ must be one of the following: $\emptyset, \{N\}, \{N-1, N\}, \ldots, \{2, 3, \ldots, N\}$. Denote the slowest active server in the minimal solution of Problem 2 as $\hat{N}$. The following theorem, which is the analog of Theorem 1, provides expressions for $\hat{N}$ and $\hat{\mathbf{p}}$:

Theorem 2: The solution $\hat{\mathbf{p}}$ to Problem 2 can be obtained as follows. Define

$$\beta_i = \frac{\lambda + i\mu_i - \sum_{j=1}^{i} \mu_j}{i\lambda}, \quad 0 \leq i \leq N. \tag{7}$$

Then,

1) $\hat{N}$ is the maximum index i for which $\beta_i > 0$,
2) $I(\hat{\mathbf{p}}) = \{\hat{N}+1, \hat{N}+2, \ldots, N\}$,
3) $\hat{p}_i = \frac{\lambda + \hat{N}\mu_i - \sum_{j=1}^{\hat{N}} \mu_j}{\hat{N}\lambda}, \quad \forall i \notin I(\hat{\mathbf{p}})$,
4) $\hat{p}_i = 0, \quad \forall i \in I(\hat{\mathbf{p}})$.

The average delay of requests for the EQ_DELAY policy is obtained by inserting the derived expression of $\hat{\mathbf{p}}$ into Eq. (2):

$$\overline{T}(\hat{\mathbf{p}}) = \sum_{i=1}^{\hat{N}} \frac{\hat{p}_i}{\mu_i - \hat{p}_i\lambda} = \frac{\hat{N}}{\sum_{i=1}^{\hat{N}} \mu_i - \lambda}. \tag{8}$$

C. Performance Analysis of the EQ_LOAD Policy

The EQ_LOAD policy aims at achieving the same utilization at each of the servers in the system. The problem can be formally stated as follows:

Problem 3 (EQ_LOAD): Find a server access probability vector $\tilde{\mathbf{p}} \in \mathcal{P}$ such that

$$\frac{\tilde{p}_i}{\mu_i} = \frac{\tilde{p}_j}{\mu_j}, \quad \forall i, j = 1, 2, \ldots, N.$$

The solution to this problem is straightforward and is given by

$$\tilde{p}_i = \frac{\mu_i}{\sum_{j=1}^{N} \mu_j}, \quad \forall i = 1, 2, \ldots, N. \tag{9}$$

The average delay for EQ_LOAD satisfies the following expression

$$\overline{T}(\tilde{\mathbf{p}}) = \frac{N}{\sum_{i=1}^{N} \mu_i - \lambda}. \tag{10}$$

IV. Benchmarking the EQ_DELAY and EQ_LOAD Policies

In this section, we compare the performance of the EQ_DELAY, EQ_LOAD and OPT policies. We first show that the average delay of EQ_LOAD is always larger than or equal to that of EQ_DELAY. At low load, the ratio between the average delays of the two policies can be as high as N, for an N-server system. At high load, however, when all servers are active, EQ_DELAY and EQ_LOAD have the same average delay. More importantly, we show that, in the worst case, the average delay of EQ_DELAY is exactly N times larger than the average delay of the OPT benchmark. We show constructively how this worst-case ratio can be achieved in a highly loaded, heterogenous system. From a game-theoretic standpoint, this result indicates that, in the context of selfish routing, the price of anarchy is topology-dependent and can potentially be very large for unbounded delay networks. Finally, we show that, in the worst case, the average delay of EQ_LOAD is also exactly N times larger than the average delay of the OPT benchmark.

A. Comparing the Average Delay and the Number of Active Servers

We first establish some basic comparative relations among EQ_DELAY, EQ_LOAD, and the OPT benchmark. We find that OPT always has the lowest average delay while EQ_LOAD always has the highest. In fact, EQ_LOAD's average delay is the same as EQ_DELAY at high load but can be N times as high as the latter at low load. We also find that EQ_DELAY utilizes the same number or fewer active servers than OPT.

Theorem 3: 1) For any given service rate vector $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_N)'$ and aggregate arrival rate λ, the average delays of OPT, EQ_DELAY and EQ_LOAD satisfy the following relation:

$$\overline{T}(\mathbf{p}^*) \leq \overline{T}(\hat{\mathbf{p}}) \leq \overline{T}(\tilde{\mathbf{p}}).$$

2) If $\hat{N} = N$, then EQ_DELAY and EQ_LOAD have the same average delay:

$$\overline{T}(\hat{\mathbf{p}}) = \overline{T}(\tilde{\mathbf{p}}).$$

This situation happens when the load is high enough, so that EQ_DELAY uses all the servers.

3) EQ_LOAD's average delay can be N times higher than that of EQ_DELAY:

$$\sup_{\boldsymbol{\mu}, \lambda} \frac{\overline{T}(\tilde{\mathbf{p}})}{\overline{T}(\hat{\mathbf{p}})} = N. \tag{11}$$

This situation may be realized with arbitrarily small error at low load, i.e., when $\lambda \to 0$.

Proof: We prove the above statements in order.

1) From the definition of the OPT policy, we already know that $\overline{T}(\mathbf{p}^*) \leq \overline{T}(\hat{\mathbf{p}})$.
Next, we prove that $\overline{T}(\hat{\mathbf{p}}) \leq \overline{T}(\tilde{\mathbf{p}})$. If $\hat{N} = N$, then from Eqs. (8) and (10), it is clear that $\overline{T}(\hat{\mathbf{p}}) = \overline{T}(\tilde{\mathbf{p}})$. Now,

suppose that $\hat{N} < N$. Since $\beta_i \leq 0$ for all $i \geq \hat{N}+1$, we deduce from Eq. (7) that

$$\hat{N}\mu_i \leq \sum_{i=1}^{\hat{N}} \mu_i - \lambda, \qquad \forall i \geq \hat{N}+1. \tag{12}$$

By summing both sides of Eq. (12) over all indices i from $\hat{N}+1$ to N, we obtain

$$\hat{N} \sum_{i=\hat{N}+1}^{N} \mu_i \leq \left(N - \hat{N}\right) \left(\sum_{i=1}^{\hat{N}} \mu_i - \lambda\right). \tag{13}$$

Adding $\left(\hat{N}\sum_{i=1}^{\hat{N}} \mu_i - \hat{N}\lambda\right)$ to both sides of Eq. (13), we have

$$\hat{N}\left(\sum_{i=1}^{N} \mu_i - \lambda\right) \leq N \left(\sum_{i=1}^{\hat{N}} \mu_i - \lambda\right)$$

or

$$\frac{\hat{N}}{\left(\sum_{i=1}^{\hat{N}} \mu_i - \lambda\right)} \leq \frac{N}{\left(\sum_{i=1}^{N} \mu_i - \lambda\right)}.$$

So we have proved that $\overline{T}(\hat{\mathbf{p}}) \leq \overline{T}(\tilde{\mathbf{p}})$.

2) This is a direct result of Eqs. (8) and (10).
3) We only consider the non-trivial case of $\lambda > 0$. We can first upper bound $\frac{\overline{T}(\tilde{\mathbf{p}})}{\overline{T}(\hat{\mathbf{p}})}$ as follows, noting that $\hat{N} \geq 1$:

$$\frac{\overline{T}(\tilde{\mathbf{p}})}{\overline{T}(\hat{\mathbf{p}})} = \frac{N}{\hat{N}} \frac{\sum_{i=1}^{\hat{N}} \mu_i - \lambda}{\sum_{i=1}^{N} \mu_i - \lambda} < N. \tag{14}$$

On the other hand, this bound is realized with arbitrarily small error at low load (i.e., the aggregate arrival rate λ is close to 0), by constructing a service rate vector $\boldsymbol{\mu}$ such that $\mu_1 \gg \sum_{i=2}^{N} \mu_i$. In this case, the number of active servers $\hat{N} = 1$ for EQ_DELAY, and Eq. (14) is infinitesimally close to N. Thus, Eq. (11) holds. ■

Note that it is possible that when a new server is added to the network, EQ_LOAD's average delay increases (see [31] for an example).

Next, we compare the number of active servers in EQ_DELAY and OPT. This result will be used later in this section.

Lemma 1: For any service rate vector $\boldsymbol{\mu}$, load λ and number of servers N, EQ_DELAY has the same or smaller number of active servers than OPT:

$$\hat{N} \leq N^* \leq N.$$

Proof: It suffices to show that $\beta_{N^*+1} \leq 0$ for all $N^* < N$:

$$\begin{aligned} & \lambda(N^*+1)\beta_{N^*+1} \\ = \ & \lambda - \sum_{j=1}^{N^*+1} \mu_j + (N^*+1)\mu_{N^*+1} \\ \leq \ & \lambda - \sum_{j=1}^{N^*+1} \mu_j + \sqrt{\mu_{N^*+1}} \sum_{j=1}^{N^*+1} \sqrt{\mu_j} \\ = \ & (\alpha_{N^*+1})\,\lambda \left(\sum_{j=1}^{N^*+1} \sqrt{\mu_j}\right) / \sqrt{\mu_{N^*+1}}. \end{aligned} \tag{15}$$

Since $\alpha_{N^*+1} \leq 0$, the proof of the lemma follows. ■

B. Bounding the Inefficiency of EQ_DELAY

We next show that, the worst-case ratio of EQ_DELAY's average delay to that of OPT is exactly N. This is achieved by developing matching lower- and upper-bounds of the delay ratio.We relate these results with the price of anarchy in selfish routing.

To improve readability, we first present this section's conclusion in the following theorem:

Theorem 4: The worst-case ratio between EQ_DELAY's average delay and that of OPT is exactly N:

$$\sup_{\boldsymbol{\mu},\lambda} \frac{\overline{T}(\hat{\mathbf{p}})}{\overline{T}(\mathbf{p}^*)} = N. \tag{16}$$

Proof: This theorem is the direct result of Lemma 2 (which lower-bounds the ratio) and Lemma 3 (which upper-bounds the ratio), and noting that $N^* \leq N$ from Lemma 1. ■

Remark 1: Before moving to the derivation of Lemmas 2 and 3, we note that the above result has an important game-theoretic interpretation. In the field of selfish routing, EQ_DELAY is of particular interest because it is the equilibrium outcome of selfish and uncoordinated agents trying to optimize their own performance. On the other hand, OPT represents the social optimum under centralized control. The left hand side of Eq. (16), referred to as *the price of anarchy*, represents the worst-case loss of efficiency when the centralized, system-level control is replaced by decentralized control designed for individual optimization [15, 16]. For bounded delay functions, the price of anarchy is often modest and independent of the network topology [14, 17]. Theorem 4 shows that, in networks with unbounded delay functions such as $M/G/1$-PS, the price of anarchy is *topology dependent* in the sense that it is exactly the number of the servers N, and this result holds for any service rate and load condition. It implies that the price of anarchy increases with the number of servers and can be potentially very large. Note that for $M/G/1$-PS (or $M/M/1$) type delay functions, Roughgarden derived an upper bound of the price of anarchy [14]. This bound is useful only when the aggregate traffic load is smaller than the capacity of the slowest server, otherwise it becomes infinite.

Now, as a first step of proving Theorem 4, we derive a lower bound of the delay ratio between EQ_DELAY and OPT.

Lemma 2: For any given N, we have

$$\sup_{\boldsymbol{\mu},\lambda} \frac{\overline{T}(\hat{\mathbf{p}})}{\overline{T}(\mathbf{p}^*)} \geq N. \tag{17}$$

Proof: For any N, we derive an example with a particular configuration of arrival rate and service rates parameters such that the ratio of the average delays tends to N. Assume the sum of service rates of all the servers is 1. Furthermore, assume μ_1 is close to 1 and $\mu_1 \gg \mu_2 = \mu_3 = \cdots = \mu_N$. Thus, we have

$$\mu_i = \frac{1-\mu_1}{N-1}, \quad 1 < i \leq N. \tag{18}$$

In addition, assume $\sum_{i=1}^{N-1} \mu_i < \lambda < 1$. Then $N^* = \hat{N} = N$ and $\overline{T}(\tilde{\mathbf{p}}) = \overline{T}(\hat{\mathbf{p}})$. The ratio of Eq. (8) to Eq. (5) yields

$$\begin{aligned} \frac{\overline{T}(\hat{\mathbf{p}})}{\overline{T}(\mathbf{p}^*)} &= \frac{N\lambda}{\left(\sum_{i=1}^{N}\sqrt{\mu_i}\right)^2 - N\left(\sum_{i=1}^{N}\mu_i - \lambda\right)} \\ &\geq \frac{N\lambda}{\left(\sum_{i=1}^{N}\sqrt{\mu_i}\right)^2} \\ &= \frac{N\lambda}{\left(\sqrt{\mu_1} + \sqrt{(N-1)(1-\mu_1)}\right)^2}. \end{aligned} \quad (19)$$

Note that for any given N, the expression $(N-1)(1-\mu_1)$ appearing in the denominator of Eq. (19) becomes arbitrarily small as $\mu_1 \to 1$. Since $\mu_1 < \lambda < 1$, we also have that $\lambda \to 1$ as $\mu_1 \to 1$. Therefore, Eq. (19) becomes arbitrarily close to N as $\mu_1 \to 1$ and the theorem is proven. ■

The major difficulty in proving Theorem 4 is to develop a tight upper bound. In Lemma 3, we provide an upper bound that matches the lower bound of Lemma 2, based on the construction of two auxiliary systems.

Lemma 3: For a given system load λ and associated number of active servers in OPT, N^*, the following inequality holds:

$$\frac{\overline{T}(\hat{\mathbf{p}})}{\overline{T}(\mathbf{p}^*)} < N^*. \quad (20)$$

Proof: . From Lemma 1, we know $N^* \geq \hat{N}$ for any given λ. We prove the theorem in two steps. In the first step, we show that Eq. (20) holds when $N^* = \hat{N}$. In the second step, we construct two auxiliary systems to show that Eq. (20) holds when $N^* > \hat{N}$.

Step 1: The case of $N^* = \hat{N}$. We observe that in order for server $\hat{N}$ to be active, we must have $\beta_{\hat{N}} > 0$ or

$$\lambda + \hat{N}\mu_{\hat{N}} - \sum_{i=1}^{\hat{N}} \mu_i > 0. \quad (21)$$

We now define the *threshold service rate* μ_T as

$$\mu_T = \frac{\left(\sum_{i=1}^{\hat{N}} \mu_i - \lambda\right)}{\hat{N}}. \quad (22)$$

From Eq. (21), we can show that $\mu_{\hat{N}} > \mu_T$.

Because $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_{\hat{N}}$, we have

$$\sqrt{\mu_i \mu_j} > \mu_T, \qquad \forall\ 1 \leq i, j \leq \hat{N}. \quad (23)$$

By summing both sides of Eq. (23) over all indices i between 1 and $\hat{N}$, we have

$$\hat{N}(\hat{N}-1)\mu_T < \sum_{i=1}^{\hat{N}} \sum_{\substack{j=1 \\ j \neq i}}^{\hat{N}} \sqrt{\mu_i \mu_j}. \quad (24)$$

We combine Eqs. (22) and (24) and obtain

$$\hat{N}^2 \mu_T + \lambda < \left(\sum_{i=1}^{\hat{N}} \sqrt{\mu_i}\right)^2. \quad (25)$$

Note that the derivation of Eq. (25) only assumes there are $\hat{N}$ active servers in EQ_DELAY and does not require that $N^* = \hat{N}$. Hence it can be used in Step 2 of the proof as well.

Now we prove Eq. (20) when $N^* = \hat{N}$, which is equivalent to

$$\frac{\hat{N}\lambda}{\left(\sum_{i=1}^{\hat{N}} \sqrt{\mu_i}\right)^2 - \hat{N}\left(\sum_{i=1}^{\hat{N}} \mu_i - \lambda\right)} < \hat{N}. \quad (26)$$

Taking into account that $\mu_T = \frac{\left(\sum_{i=1}^{\hat{N}} \mu_i - \lambda\right)}{\hat{N}}$, we find that Eq. (26) is equivalent to Eq. (25).

Step 2: The case of $N^* > \hat{N}$. Our original system has N servers with service rates $(\mu_1, \mu_2, \ldots, \mu_N)'$. We first remove all inactive servers in OPT from the original system. The resulting system, denoted as the $\mathbf{u}$ system, has a service rate vector

$$\mathbf{u} = (u_1, u_2, \ldots, u_{N^*})' = (\mu_1, \mu_2, \ldots, \mu_{\hat{N}}, \ldots, \mu_{N^*})'.$$

Clearly, the $\mathbf{u}$ system's average delay under the OPT policy, $\overline{T}(\mathbf{p}^*_{\mathbf{u}})$, is the same as that of the original system, $\overline{T}(\mathbf{p}^*)$. Furthermore, according to Lemma 1, all removed servers are inactive in EQ_DELAY too, so we have the same relation for EQ_DELAY: $\overline{T}(\hat{\mathbf{p}}_{\mathbf{u}}) = \overline{T}(\hat{\mathbf{p}})$. Consequently, the ratio $\frac{\overline{T}(\hat{\mathbf{p}})}{\overline{T}(\mathbf{p}^*)}$ of the original system is the same as $\frac{\overline{T}(\hat{\mathbf{p}}_{\mathbf{u}})}{\overline{T}(\mathbf{p}^*_{\mathbf{u}})}$ of the $\mathbf{u}$ system.

The key of upper-bounding the delay ratio is constructing another auxiliary system, called the $\mathbf{v}$ system, which also has N^* servers. In the $\mathbf{v}$ system, the fastest $\hat{N}$ servers have the same rates as those in the $\mathbf{u}$ system, but the remaining $N^* - \hat{N}$ servers are assigned the threshold service rate, μ_T. The service rate vector of the $\mathbf{v}$ system is thus

$$\mathbf{v} = (v_1, v_2, \ldots, v_{N^*})' = (\mu_1, \mu_2, \ldots, \mu_{\hat{N}}, \mu_T, \mu_T, \ldots, \mu_T)'.$$

Because $\mu_T < \mu_{\hat{N}}$, $\mathbf{v}$ is also descending. More importantly, the $\mathbf{u}$ and $\mathbf{v}$ systems have the same EQ_DELAY average delay, i.e., $\overline{T}(\hat{\mathbf{p}}_{\mathbf{u}}) = \overline{T}(\hat{\mathbf{p}}_{\mathbf{v}})$. This is because in the $\mathbf{v}$ system, at load λ, all servers with the threshold service rate μ_T are inactive as shown in Theorem 2. Since the two systems have the same service rates for the active $\hat{N}$ servers, they must have the same average delay when the EQ_DELAY policy is used.

Before we evaluate $\overline{T}(\mathbf{p}^*_{\mathbf{v}})$, we need to show that all servers in the $\mathbf{v}$ system are active under OPT. This is achieved by first observing from the derivation of N^* in Theorem 1, that in the $\mathbf{u}$ system,

$$\sqrt{u_{N^*}} \sum_{j=1}^{N^*} \sqrt{u_j} > \sum_{j=1}^{N^*} u_j - \lambda.$$

Because $u_{N^*-1} \geq u_{N^*}$, we have

$$\sqrt{u_{N^*-1}} \sum_{j=1}^{N^*-1} \sqrt{u_j} > \sum_{j=1}^{N^*-1} u_j - \lambda.$$

Repeating this procedure, we obtain

$$\sqrt{u_i} \sum_{j=1}^{i} \sqrt{u_j} > \sum_{j=1}^{i} u_j - \lambda, i = 1, 2, \ldots, N^*.$$

In particular,

$$\sqrt{u_{\hat{N}+1}} \sum_{j=1}^{\hat{N}+1} \sqrt{u_j} > \sum_{j=1}^{\hat{N}+1} u_j - \lambda$$

or

$$\sqrt{u_{\hat{N}+1}}\sum_{j=1}^{\hat{N}}\sqrt{u_j} > \sum_{j=1}^{\hat{N}} u_j - \lambda.$$

Because $v_{\hat{N}+1} = \mu_T \geq u_{\hat{N}+1}$, we have

$$\sqrt{v_{\hat{N}+1}}\sum_{j=1}^{\hat{N}}\sqrt{u_j} > \sum_{j=1}^{\hat{N}} u_j - \lambda$$

or

$$\sqrt{v_{\hat{N}+1}}\sum_{j=1}^{\hat{N}+1}\sqrt{v_j} > \sum_{j=1}^{\hat{N}+1} v_j - \lambda. \qquad (27)$$

According to Eq. (4) in Theorem 1, Eq. (27) guarantees that server $\hat{N}+1$ is active under OPT in the $\mathbf{v}$ system. Because all servers with indices larger than $\hat{N}$ in the $\mathbf{v}$ system have the same service rate, they must have the same access probability. Thus all servers in the $\mathbf{v}$ system must be active under OPT.

On the other hand, the $\mathbf{v}$ system achieves the smallest OPT delay among all N^*-server systems whose fastest $\hat{N}$ servers have the rates $\mu_1, \ldots, \mu_{\hat{N}}$ and who have $\hat{N}$ active servers under EQ_DELAY under load λ. To see this, we observe that in all systems meeting the above requirements, we have $r_i \leq \mu_T$ for $\hat{N} < i \leq N^*$, where r_i is the service rate of the i-th server in any system under consideration; otherwise the system would have more than $\hat{N}$ active servers under EQ_DELAY. Each server in the $\mathbf{v}$ system has the largest possible service rate among all systems meeting the above two requirements. As a result, it achieves the same or smaller average delay under OPT than any other system meeting the above two requirements. So we have $\overline{T}(\mathbf{p}^*_{\mathbf{v}}) \leq \overline{T}(\mathbf{p}^*_{\mathbf{u}})$.

Combining the above analysis in both EQ_DELAY and OPT, we see that

$$\frac{\overline{T}(\hat{\mathbf{p}})}{\overline{T}(\mathbf{p}^*)} = \frac{\overline{T}(\hat{\mathbf{p}}_{\mathbf{u}})}{\overline{T}(\mathbf{p}^*_{\mathbf{u}})} \leq \frac{\overline{T}(\hat{\mathbf{p}}_{\mathbf{v}})}{\overline{T}(\mathbf{p}^*_{\mathbf{v}})}. \qquad (28)$$

Now we upper bound $\frac{\overline{T}(\hat{\mathbf{p}})}{\overline{T}(\mathbf{p}^*)}$ by evaluating $\overline{T}(\hat{\mathbf{p}}_{\mathbf{v}})$ and $\overline{T}(\mathbf{p}^*_{\mathbf{v}})$. By definition and the structure of $\mathbf{v}$, we know that

$$\hat{N}\mu_T = \sum_{i=1}^{\hat{N}} \mu_i - \lambda = \sum_{i=1}^{\hat{N}} v_i - \lambda$$

and

$$N^*\mu_T = \sum_{i=1}^{N^*} v_i - \lambda.$$

Hence, the $\mathbf{v}$ system's average delay under EQ_DELAY is

$$\overline{T}(\hat{\mathbf{p}}_{\mathbf{v}}) = \overline{T}(\hat{\mathbf{p}}) = \frac{\hat{N}}{\sum_{i=1}^{\hat{N}} v_i - \lambda} = \frac{1}{\mu_T}. \qquad (29)$$

For $\overline{T}(\mathbf{p}^*_{\mathbf{v}})$, we have

$$\begin{aligned}\overline{T}(\mathbf{p}^*_{\mathbf{v}}) &= \frac{\left(\sum_{i=1}^{N^*}\sqrt{v_i}\right)^2}{\lambda\left(\sum_{i=1}^{N^*} v_i - \lambda\right)} - \frac{N^*}{\lambda} \\ &= \frac{\left[\sum_{i=1}^{\hat{N}}\sqrt{\mu_i} + (N^* - \hat{N})\sqrt{\mu_T}\right]^2 - (N^*)^2\mu_T}{\lambda N^* \mu_T}.\end{aligned}$$

Combining the expressions for $\overline{T}(\hat{\mathbf{p}}_{\mathbf{v}})$ and $\overline{T}(\mathbf{p}^*_{\mathbf{v}})$ and incorporating Eq. (25), we have

$$\begin{aligned}&\frac{\overline{T}(\hat{\mathbf{p}}_{\mathbf{v}})}{\overline{T}(\mathbf{p}^*_{\mathbf{v}})} \\ <\;& \frac{\lambda N^*}{\lambda + 2\left(N^* - \hat{N}\right)\sqrt{\mu_T}\left(\sum_{i=1}^{\hat{N}}\sqrt{\mu_i} - \hat{N}\sqrt{\mu_T}\right)} \\ <\;& N^*.\end{aligned}$$

From Eq. (28), we have

$$\frac{\overline{T}(\hat{\mathbf{p}})}{\overline{T}(\mathbf{p}^*)} < N^*,$$

which completes the proof of the second step. ∎

C. Bounding the Inefficiency of EQ_LOAD

In this section, we prove that the maximum ratio between EQ_LOAD's and OPT's average delays is exactly N. Thus, somewhat surprisingly, this worst-case ratio for EQ_LOAD is the same as that for EQ_DELAY.

Theorem 5: The worst case delay ratio between EQ_LOAD and OPT is exactly N:

$$\sup_{\boldsymbol{\mu},\lambda} \frac{\overline{T}(\tilde{\mathbf{p}})}{\overline{T}(\mathbf{p}^*)} = N. \qquad (30)$$

Proof: From Theorem 3, we know that $\overline{T}(\tilde{\mathbf{p}}) \geq \overline{T}(\hat{\mathbf{p}})$. Thus, according to Lemma 2, we must have

$$\sup_{\boldsymbol{\mu},\lambda} \frac{\overline{T}(\tilde{\mathbf{p}})}{\overline{T}(\mathbf{p}^*)} \geq N.$$

To complete the proof, we only need to show that

$$\frac{\overline{T}(\tilde{\mathbf{p}})}{\overline{T}(\mathbf{p}^*)} < N. \qquad (31)$$

First, from Eq. (10), we note that

$$\overline{T}(\tilde{\mathbf{p}}) \leq \frac{N}{\sum_{i=1}^{N^*}\mu_i - \lambda}.$$

Taking OPT's average delay expression, Eq. (5), into account, we can prove Eq. (31) by showing that the following expression holds

$$\frac{\frac{N}{\sum_{i=1}^{N^*}\mu_i - \lambda}}{\frac{\left(\sum_{i=1}^{N^*}\sqrt{\mu_i}\right)^2}{\lambda\left(\sum_{i=1}^{N^*}\mu_i - \lambda\right)} - \frac{N^*}{\lambda}} < N,$$

which can be simplified to

$$\left(\sum_{i=1}^{N^*}\sqrt{\mu_i}\right)^2 - N^*\left(\sum_{i=1}^{N^*}\mu_i - \lambda\right) - \lambda > 0. \qquad (32)$$

Now, we consider EQ_DELAY under the same parameters $\boldsymbol{\mu}$ and λ. If $\hat{N} = N^*$, then Eq. (32) is the same as Eq. (25), so Eq. (31) is proved.

Next, we note that Eq. (25) also holds when $\hat{N} < N^*$. To prove this, consider a hypothetical $\mathbf{w}$ system where all the servers $\hat{N}+1, \hat{N}+2, \ldots, N$ are removed. In the $\mathbf{w}$ system, OPT and EQ_DELAY have the same number of active servers,

$\hat{N}$, and Eq. (25) must be satisfied. Because Eq. (25) only provides a relation between the parameters $(\mu_1, \mu_2, \ldots, \mu_{\hat{N}})'$ and λ, it must hold as well for the original system with N servers.

We are now ready to use induction on the variable N' to prove that the following inequality holds for our original system:

$$\left(\sum_{i=1}^{N'} \sqrt{\mu_i}\right)^2 - N'\left(\sum_{i=1}^{N'} \mu_i - \lambda\right) - \lambda > 0, \ \hat{N} \le N' \le N^*. \tag{33}$$

First, we already know that Eq. (33) holds when $N' = \hat{N}$, which proves the induction basis. Now suppose Eq. (33) holds for some N', where $\hat{N} \le N' < N^*$. We want to show

$$\left(\sum_{i=1}^{N'+1} \sqrt{\mu_i}\right)^2 - (N'+1)\left(\sum_{i=1}^{N'+1} \mu_i - \lambda\right) - \lambda > 0, \tag{34}$$

which is in fact the case since

$$\begin{aligned}
& \left(\sum_{i=1}^{N'+1} \sqrt{\mu_i}\right)^2 - (N'+1)\left(\sum_{i=1}^{N'+1} \mu_i - \lambda\right) - \lambda \\
= & \left(\sum_{i=1}^{N'} \sqrt{\mu_i}\right)^2 + 2\sqrt{\mu_{N'+1}}\sum_{i=1}^{N'} \sqrt{\mu_i} + \mu_{N'+1} \\
& -N'\left(\sum_{i=1}^{N'} \mu_i - \lambda\right) - N'\mu_{N'+1} \\
& -\left(\sum_{i=1}^{N'+1} \mu_i - \lambda\right) - \lambda \\
> & \ 2\sqrt{\mu_{N'+1}}\sum_{i=1}^{N'} \sqrt{\mu_i} - N'\mu_{N'+1} - \left(\sum_{i=1}^{N'} \mu_i - \lambda\right) \qquad (35) \\
> & \left(\sum_{i=1}^{N'} \mu_i - \lambda\right) - N'\mu_{N'+1} \qquad (36) \\
\ge & \ 0. \qquad (37)
\end{aligned}$$

Note that Eq. (35) holds because of the induction hypothesis. Eq. (36) holds since

$$\sqrt{\mu_{N'+1}} > \frac{\left(\sum_{i=1}^{N'} \mu_i - \lambda\right)}{\sum_{i=1}^{N'} \sqrt{\mu_i}}, \text{ for all } N' < N^*$$

according to Theorem 1. Eq. (37) holds since

$$\mu_{N'+1} \le \frac{\left(\sum_{i=1}^{N'} \mu_i - \lambda\right)}{N'} \text{ for all } N' \ge \hat{N}$$

according to Theorem 2. Thus, Eq. (32), being a special case of Eq. (33), is proven and the proof is complete. ■

Remark 2: Theorems 4 and 5 reveal that EQ_LOAD and EQ_DELAY have the same worst-case ratio against OPT in terms of average delay. However, it is worth noting that EQ_DELAY can achieve this ratio only at high load while EQ_LOAD can achieve it at both low and high loads. Indeed, according to Lemma 3, the ratio between EQ_DELAY's and OPT's average delays is always smaller than N^*. Thus, it can become large only at high load. On the other hand, at low load, the ratio between EQ_LOAD's and OPT's (or EQ_DELAY's) average delays can be as large as N, as indicated by Theorem 3.

V. Extensions for $M/G/1$-FCFS, $G/G/1$-FCFS and Multi-Class $M/G/1$-PS Models

In this section, we consider scenarios more general than the one studied so far. We will study the $M/G/1$-FCFS and $G/G/1$-FCFS models as well as multi-class $M/G/1$-PS networks where each SSN-server pair has its own service time.

A. The $M/G/1$-FCFS Model

We consider the same model as the one introduced in Section II, except that the scheduling policy at each server i is First-Come First-Serve instead of Processor Sharing. Following the notations of [32], we denote σ_i^2 as the variance of the service time at server i, $C_S(i) = \sigma_i/\bar{x}_i$ as the coefficient of variation of the service time, where we remind that $\bar{x}_i = 1/\mu_i$ is the mean service time at server i. In addition, we denote by d_i the network delay to each server i. We assume that d_i is a random variable with mean $\bar{d}_i$.

Using the above notation, we can express the average waiting time of a request at server i using the well-known Pollaczek-Khinchin (PK) formula [32]

$$\overline{W}_i(p_i) = \frac{K_i p_i \lambda}{\mu_i(\mu_i - p_i\lambda)}, \tag{38}$$

where $K_i \equiv \frac{1+C_S^2(i)}{2}$.

The total average delay of a request forwarded to server i consists of the sum of the waiting time, service time, and network delay. Thus,

$$\begin{aligned}
\overline{T}_i(p_i) &= \overline{W}_i(p_i) + \frac{1}{\mu_i} + \bar{d}_i \\
&= \frac{K_i p_i \lambda}{\mu_i(\mu_i - p_i\lambda)} + \frac{1}{\mu_i} + \bar{d}_i \\
&= \frac{K_i}{\mu_i - p_i\lambda} + \frac{1-K_i}{\mu_i} + \bar{d}_i. \qquad (39)
\end{aligned}$$

Our optimization problem can then be formalized as follows:

Problem 4: Find the optimal server access probability vector

$$\mathbf{p}^* = \arg\min_{\mathbf{p}\in\mathcal{P}} \overline{T}(\mathbf{p}), \tag{40}$$

where $\overline{T}(\mathbf{p}) = \sum_{i=1}^N p_i \overline{T}_i(p_i)$ and $\overline{T}_i(p_i)$ is given by Eq. (39).

It is generally infeasible to obtain a closed-form solution to Problem 4 and numerical approaches must be used to derive the optimal solution [26, 30, 33]. However, by analyzing the system behavior at high load, we can derive closed-form expressions for the optimal access probabilities in this regime. Our simulations in Section VI show that the asymptotically optimal server selection policy actually provides a solution close to the minimum average delay for a wide range of server utilizations.

B. High Load Analysis

In this section, we study Problem 4 under high load. High load is defined as the regime where for any feasible probability vector $\mathbf{p} \in \mathcal{P}$, each coordinate satisfies $\frac{p_i \lambda}{\mu_i} \to 1$, for all $i = 1, 2, \ldots, N$. Under this regime, the second and third terms in Eq. (39) become negligible compared to the first term and the total average delay becomes

$$\overline{T}(\mathbf{p}) \cong \sum_{i=1}^{N} \frac{K_i p_i}{\mu_i - p_i \lambda}. \tag{41}$$

Minimizing Eq. (41) is similar to obtaining the minimum average delay in Section III. We can compute the optimal server access probability p_i^* and the corresponding average delay $\overline{T}(\mathbf{p}^*)$ as the following:

$$p_i^* = \frac{\mu_i}{\lambda} - \frac{\left(\sum_{j=1}^{N} \mu_j - \lambda\right) \sqrt{K_i \mu_i}}{\lambda \sum_{j=1}^{N} \sqrt{K_j \mu_j}}, \quad 0 \le i \le N; \tag{42}$$

$$\overline{T}(\mathbf{p}^*) = \frac{\left(\sum_{i=1}^{N} \sqrt{K_i \mu_i}\right)^2}{\lambda \left(\sum_{i=1}^{N} \mu_i - \lambda\right)} - \frac{\sum_{i=1}^{N} K_i}{\lambda}. \tag{43}$$

The access probabilities and average delays for the EQ_DELAY and EQ_LOAD policies can be derived in a similar manner.

The following lemma generalizes Lemma 2 and provides a lower bound on the worst-case ratio between EQ_DELAY's (or EQ_LOAD's) and OPT's average delays for the $M/G/1$-FCFS case:

Lemma 4: For any given N servers, there exist parameters λ and $\mu_i, i = 1, 2, \ldots, N$, such that $\overline{T}(\tilde{\mathbf{p}})/\overline{T}(\mathbf{p}^*) = \overline{T}(\hat{\mathbf{p}})/\overline{T}(\mathbf{p}^*) \to \frac{\sum_{i=1}^{N} K_i}{K_1}$.

The proof is similar to that of Lemma 2. Because K_1 is finite (assuming the second moment exists) and $K_i \ge \frac{1}{2}$ for any $M/G/1$ queue, $\frac{\sum_{i=1}^{N} K_i}{K_1}$ becomes arbitrarily large as $N \to \infty$.

C. The $G/G/1$-FCFS Model

Consider the same model as in Section V-A, but assume now that the inter-arrival time between requests at each server follows a general *i.i.d.* distribution. Under such assumptions, each server can be modeled as a $G/G/1$-FCFS queue [32]. Unfortunately, there exists no simple, general expression for the mean waiting time in a $G/G/1$-FCFS and, therefore, even a numerical derivation of the optimal access probabilities becomes difficult.

However, in the high load regime, when $\frac{p_i \lambda}{\mu_i} \to 1$, for all $i = 1, 2, \ldots, N$, a simple, asymptotically exact expression for $\overline{W}_i$ can be provided. Specifically, define the coefficient of variation of the inter-arrival time at server i as $C_A(i)$, then [34]

$$\overline{W}_i(p_i) \cong \frac{C_A^2(i) + C_S^2(i)}{2\left(\mu_i - p_i \lambda\right)}. \tag{44}$$

As discussed earlier, at high load, the service time and network delay are negligible compared to the waiting time in the server. Using the notation $K_i = \frac{C_A^2(i) + C_S^2(i)}{2}$, we therefore have

$$\overline{T}(\mathbf{p}) \cong \sum_{i=1}^{N} \frac{K_i p_i}{\mu_i - p_i \lambda}, \tag{45}$$

which is the same as Eq. (41). Therefore, all the results derived in Section V-B can directly be applied to $G/G/1$-FCFS server selection as well.

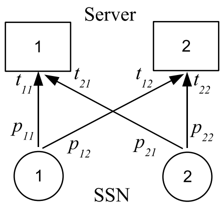


Fig. 2. A multi-class network with 2 SSNs and 2 servers

D. The Multi-Class $M/G/1$-PS Model

In this section, we extend our results to a multi-class $M/G/1$-PS model where each class is associated with a different SSN. Under this model, requests coming from different SSNs may have different average service times when accessing a given server j. Hence, the access probabilities may also differ for different SSNs. This model qualitatively captures the fact that requests originating from a SSN geographically close to a certain server experience smaller average delay than those originating from a remote SSN.

The model is defined by an arrival rate vector $\boldsymbol{\lambda}$, a service time matrix $\mathbf{t}$, and a server access probability matrix $\mathbf{P}$. As in Section II, we denote SSN i's request arrival rate as λ_i. The arrival rate vector in this multi-class network is defined as $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_M)'$. Suppose the service time for a request forwarded by SSN i to server j is t_{ij}. We denote $\mathbf{t}$ as the service time matrix, with t_{ij} at its i-th row and j-th column. The probability that SSN i selects server j is denoted by p_{ij}. Similar to the definition of $\mathbf{t}$, we define $\mathbf{P}$ as the server access probability matrix. Obviously, we have $\sum_{j=1}^{N} p_{ij} = 1$, $1 \le i \le M$.

We use the notation of $M \times N$ to represent a multi-class network with M SSNs and N servers. Fig. 2 illustrates the simplest network in this model, a 2×2 multi-class network, with the corresponding average service times and access probabilities.

The utilization of server j is computed by summing all SSNs' request rate on server j, weighted by individual service times:

$$\rho_j = \sum_{k=1}^{M} p_{kj} \lambda_k t_{kj}. \tag{46}$$

Then, SSN i's average delay at server j is [35, 36]

$$\overline{T}_{ij}(\mathbf{P}) = \frac{t_{ij}}{1 - \rho_j}. \tag{47}$$

The average delay of an $M \times N$ network $\overline{T}(\mathbf{P})$ is given by the following expression:

$$\begin{aligned}\overline{T}(\mathbf{P}) &= \frac{1}{\sum_{i=1}^{M} \lambda_i} \left[\sum_{i=1}^{M} \sum_{j=1}^{N} p_{ij} \lambda_i \overline{T}_{ij}(\mathbf{P}) \right] \\ &= \frac{1}{\sum_{i=1}^{M} \lambda_i} \left(\sum_{j=1}^{N} \frac{1}{1-\rho_j} - N \right) \qquad (48)\end{aligned}$$

Deriving an analytical solution for the server access probability matrix $\mathbf{P}$ in multi-class networks is far more difficult than computing the server access probability vector $\mathbf{p}$ as done in Section III, because the numbers of variables and boundary conditions increase significantly. For example, even in simple 2×2 networks, there exist nine possible configurations depending on whether specific network links connecting SSNs and servers are active or not [31]. To overcome this difficulty, we focus on the three policies' asymptotic performance at low load and high load, because as indicated in Section IV, these are the situations where EQ_DELAY and EQ_LOAD exhibit their largest inefficiencies in single-class networks.

For low load conditions, the following theorem demonstrates that the ratio of the average delays of EQ_LOAD to OPT in multi-class networks is always smaller than N. This result is consistent with single-class networks (i.e., multi-class does not perform worse). The proof of this theorem can be found in [31].

Theorem 6: Consider a general $M \times N$ $M/G/1$-PS networks, where $\lambda_i \to 0, 1 \le i \le M$. Then,

$$\frac{\overline{T}(\tilde{\mathbf{P}})}{\overline{T}(\mathbf{P}^*)} < N. \qquad (49)$$

Next, consider the case of high load conditions in 2×2 multi-class networks. We first let R_1 represent the service rate available to SSN 1 when SSN 2's arrival rate λ_2 is fixed at a constant a_2. Formally, R_1 is defined as follows:

$$R_1 = \sup_{\substack{\rho_1 < 1 \\ \rho_2 < 1 \\ \lambda_2 = a_2}} \lambda_1, \qquad (50)$$

where $0 \le a_2 < \frac{1}{t_{21}} + \frac{1}{t_{22}}$. Recall that ρ_i $(i = 1, 2)$ is server i's utilization as defined in Eq. (46).

We refer to all the cases where λ_1 approaches R_1 as *high load with respect to SSN 1*. The following theorem states that at high load with respect to SSN 1 (or SSN 2), the ratio of average delays of EQ_DELAY or EQ_LOAD to OPT does not exceed 2, as in the single-class case.

Theorem 7: In 2×2 $M/G/1$-PS networks, at high load with respect to SSN i, where $i = 1$ or 2, we have:

$$\frac{\overline{T}(\hat{\mathbf{P}})}{\overline{T}(\mathbf{P}^*)} < 2, \quad \text{and} \quad \frac{\overline{T}(\tilde{\mathbf{P}})}{\overline{T}(\mathbf{P}^*)} < 2. \qquad (51)$$

The above high load results (i.e., the ratio of the average delays does not exceed the number of servers in the network) can be extended to general $M \times N$ networks, under more specific conditions [31].

These results suggest that the largest ratio of the average delays of EQ_DELAY (or EQ_LOAD) to OPT in multi-class networks is the same as in single class ones. This conjecture is numerically verified in Section VI-E.

VI. Simulations

In this section, we perform extensive simulations to experimentally compare EQ_DELAY and EQ_LOAD versus the benchmark OPT policy. We first consider the case where the workload conforms to the $M/G/1$-PS model. We assume that the file size (and hence the service time) follows a heavy-tailed Pareto distribution with cumulative distribution function

$$F(x) = 1 - \frac{1}{(1+ax)^b} \quad x \ge 0, \qquad (52)$$

where b is the *Pareto tail index*. This choice is justified by the large number of experimental studies showing that Web file size distributions follow a Pareto distribution [37, 38]. Using this workload model, we compare the three policies' average delay and standard deviation of the delay (which relates to fairness). Next, we evaluate the performance of the three policies using synthetic traces generated by a Web workload generator, called ProWGen [12]. These traces exhibit temporal locality in file popularity and, therefore, differ from the $M/G/1$-PS model. Our simulations show that EQ_LOAD and EQ_DELAY are substantially suboptimal in both cases.

In addition, we simulate server selection for the case where servers are modeled by $M/G/1$-FCFS queues. We compare the average delay achieved by the (asymptotically optimal) OPT policy to the minimum average delay obtained through a numerical optimization procedure. The simulation results indicate that our analysis approximates the minimum average delay closely for a wide range of server utilizations.

Finally, we provide numerical results of a representative multi-class network setting, where each server has different service times for different SSNs. Our experiments indicate that the largest ratio of EQ_DELAY's (or EQ_LOAD's) average delay to that of OPT is likely to be the same as in single-class networks, in which case the general and tight bounds in Section IV can be used.

The simulations described in this section are representative of a variety of simulations that we performed for different network sizes (from 5 to 500 servers) and utilizations (from 0.05 to 0.95). Further, by Little's Law, we note that if the service rates of all servers are changed by a multiplicative constant, then the ratio of the average delays between the three policies remain the same.

A. Average Delay

In this section, we compare the three policies in the $M/G/1$-PS model using both analysis and simulations. The purposes of this experiment are to validate the correctness and accuracy of our analysis and to quantify the difference between the three policies at different load conditions. We define the aggregate utilization ρ as $\rho = \lambda / \sum_{j=1}^{N} \mu_j$. We simulate a content replication network with ten servers, with the service rate vector $(12, 12, 1.5, 1.5, 1.5, 1.5, 1.5, 1.5, 1.5, 1.5)$ req/sec. In this scenario, servers 1 and 2 are eight times faster than the other eight servers, which, when interpreted using Moore's

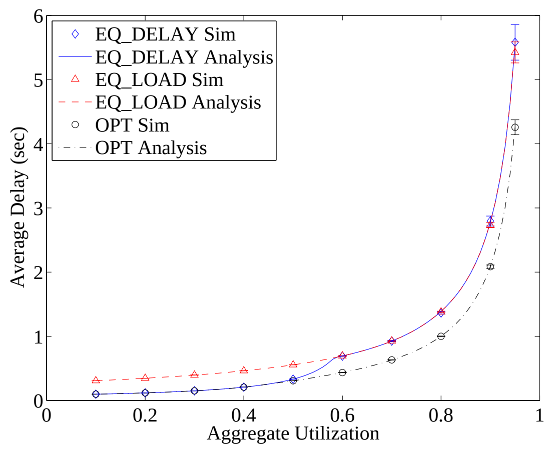


Fig. 3. Average delays of the three policies by simulation and analysis.

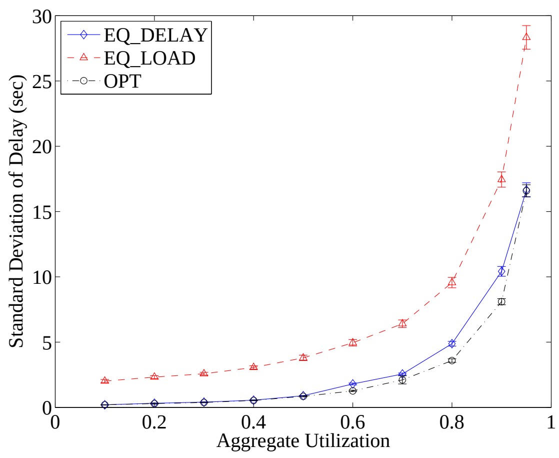


Fig. 4. Standard deviation of delay

Law, roughly represents the situation of adding two servers built with the latest hardware to an eight-server network built four or five years ago. The selection of the service rates is consistent with measurement results of Web sites that serve heavily dynamically generated content or large amount of multimedia content [22, 39].

In this experiment, we consider Pareto job size distribution with $b = 2.2$. We solve respectively Eqs. (5), (8), and (10) to analytically obtain average delays for the three policies and depict them in Fig. 3. Furthermore, we run an $M/G/1$-PS simulator 50 times for each of the three policies at each utilization point. We also depict the average delay for the 50 runs and the 95% confidence interval in Fig. 3 for the three policies.

From Fig. 3, we observe that the analysis and simulation results match very well. When comparing the three polices, we find that both OPT and EQ_DELAY perform much better than EQ_LOAD at low load. The reason is that, in this regime, OPT and EQ_DELAY forward requests only to the fastest server, while EQ_LOAD always sends a third of the requests to the slower servers. At high load, EQ_LOAD and EQ_DELAY achieve the same average delay, as proved in Section IV, and significantly underperform the OPT benchmark. For instance, at utilization $\rho = 0.9$, the average delay of EQ_DELAY and EQ_LOAD is approximately 30% higher than the optimal average delay.

B. *Standard Deviation of Delay and Fairness*

An apparent advantage of EQ_DELAY is to serve each request with the same average delay (at the cost of higher overall average delay). This might lead to the belief that it has better fairness properties than the other server selection policies. However, we show that this is actually not the case, at least with respect to OPT.

In the following set of simulations, we evaluate the standard deviation of the delay obtained with each policy. This metric has been recognized as one of the best ways to measure fairness in queues [40]. We consider a system consisting of $N = 40$ servers with $M/G/1$-PS queues. Among these servers, two have service rates of 20 req/sec, four at 10 req/sec, fourteen at 2 req/sec and twenty at 0.5 req/sec. The Pareto tail index is $b = 2.2$.

Fig. 4 depicts the deviation of delays of the three policies with 95% confidence interval of the 50 runs. We observe that OPT and EQ_DELAY perform very similarly, while the standard deviation of EQ_LOAD is noticeably higher at all utilization values. A few comments are in order here. First, even though the average delay at each server is the same for EQ_DELAY, the actual delay of a request is still a random variable. In particular, this delay depends on the number of other requests concurrently being served, which obviously varies over time. Therefore, the standard deviation of EQ_DELAY is non-zero and turns out to be on the same order as that of OPT. Second, the standard deviation of EQ_LOAD is higher than that of EQ_DELAY, even at high load. This result is somewhat surprising since the average delay of these two policies is identical in this regime. This discrepancy is resolved by noting that the average delay of requests at different servers is not the same for EQ_LOAD. Therefore, EQ_LOAD and EQ_DELAY are not statistically identical even at high load.

C. *ProWGen Simulations*

Our analytic model and solutions are based on the assumption that the service time of different requests is independent. On the other hand, some experimental studies have shown that Web requests exhibits temporal locality, see e.g. [41]. In this section, we compare the performance of EQ_DELAY and EQ_LOAD to that of OPT when request streams have short-term temporal correlation.

We use the Web workload generator ProWGen [12] to produce synthetic Web workload that exhibits temporal locality in file popularity. ProWGen, originally developed for Web cache performance evaluation, models the file popularity using the Zipf distribution, and the file size distribution using a combination of a lognormal body and a Pareto tail. It can also model positive or negative correlation between file size and popularity if needed. Furthermore, it can use an LRU stack to model request temporal locality.

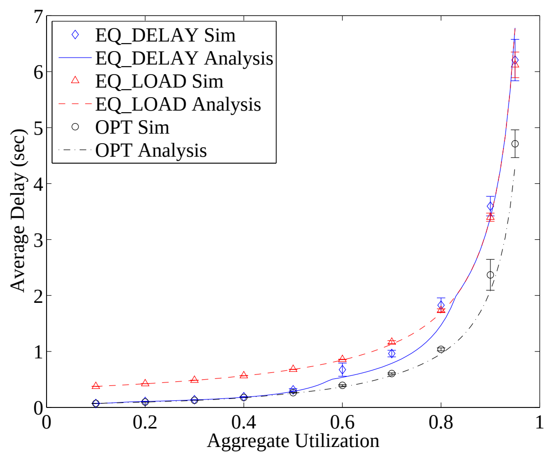


Fig. 5. Average delay with ProWGen traces

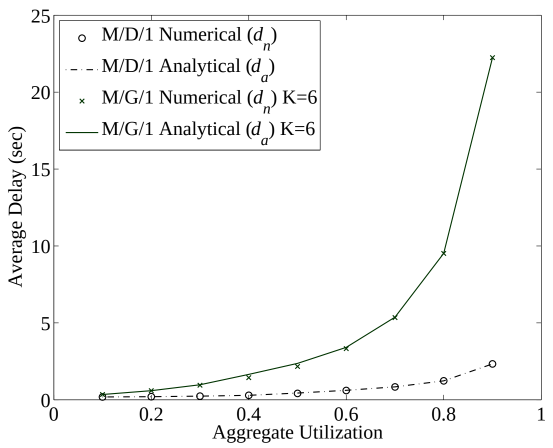


Fig. 6. Average delay of OPT for $M/G/1$-FCFS servers with deterministic and Pareto file-size distributions: analytical results vs. numerical results.

In our simulations, we use default values for most ProWGen parameters, with a Zipf slope of 0.75, Pareto Tail index of 1.2, lognormal mean of 7KB, standard deviation of 11KB, and tail cutoff size of 10KB. We also have a "dynamic" stack with a depth of 1000 requests to introduce temporal locality. Finally, the correlation between file size and popularity is set to zero, which is consistent with literature findings [42].

We find that a peculiarity of ProWGen is that popular files tend to be generated at the end of the trace, and "one-timers" (files accessed only once) are more likely to be generated earlier in the trace. To mitigate this artifact while still maintaining most of the short-term temporal locality, we divide the trace generated by ProWGen into segments each consisting of 4,000 requests and reshuffle the segments to obtain the trace used in our simulations. We scale the mean file size to 11 KB, and use the same service rate array as in Section VI-B. Each run contains about 550,000 requests, and the results are averaged over 50 runs.

The simulation results (with 95% confidence intervals) are shown in Fig. 5 together with analytical values based on our $M/G/1$-PS model. We observe that the average delays achieved by EQ_DELAY and EQ_LOAD are still higher than OPT, especially at high load. Other performance characteristics of EQ_DELAY and EQ_LOAD observed in Section VI-A are also present in this figure. Overall, the simulation results match the analysis fairly well for all the studied policies.

D. Performance of the OPT Policy for $M/G/1$-FCFS Servers

In Section V, we have shown that the OPT policy derived for $M/G/1$-PS queues is asymptotically optimal at high load for $G/G/1$-FCFS queues. The goal of this experiment is to evaluate the performance of the OPT policy for FCFS servers at light and moderate loads. We compare between the average delay achieved using the OPT policy and the minimum average delay. The value of the minimum average delay is obtained by solving Problem 4 using the `fmincon` routine of MATLAB.

In the following simulation, we use the service rate vector of $(6.06, 0.757, 0.757, 0.757, 0.757)$ req/sec. We consider two special cases of $M/G/1$-FCFS queues. In the first case, servers are modeled as $M/D/1$-FCFS queues. Thus, the file size is fixed and $K_i = 0.5$, for each server i. In the second case, the file size distribution is Pareto with tail index $b = 2.2$. In this case, it can be verified that $K_i = 1 + \frac{1}{b-2} = 6$, for each server i. We denote by d_a the average delay of OPT and d_n the minimum average delay computed numerically, and depict them in the above two cases in Fig. 6.

We observe from Fig. 6 that the OPT policy achieves delay close to the minimum over a wide range of server utilizations. We observe that the difference between d_n and d_a is quite small at both low- and high-loads. At its maximum, the difference between the two methods is 5% for $M/D/1$ (at utilization of 0.5) and 13% for $M/G/1$ with Pareto service distribution (at utilization of 0.4). This results illustrates the robustness and generality of our analytical model for a wide range of service distributions and queueing disciplines.

E. Multi-Class Networks

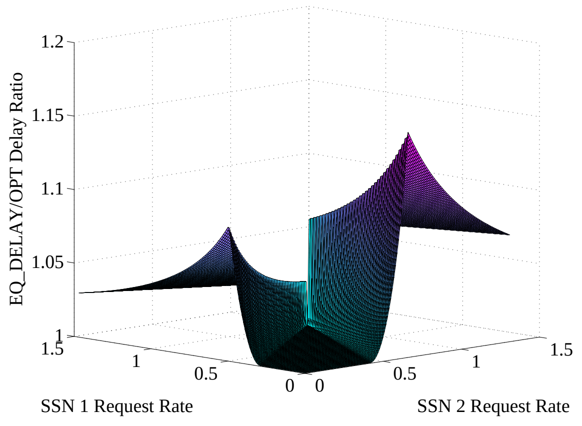
Now we examine multi-class networks that explicitly model delays specific to each SSN-server pair. To highlight the difference between single- and multi-class networks, we consider a 2×2 network with the service time matrix

$$\mathbf{t} = \begin{bmatrix} 1 & 2 \\ 3 & 1 \end{bmatrix} \text{sec.} \qquad (53)$$

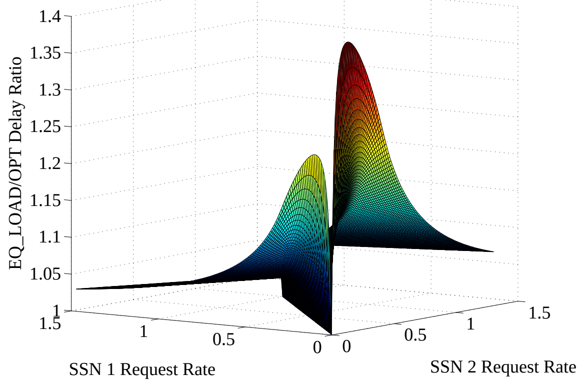
This means that for SSN 1, server 1's service time is only half that of server 2. Similarly, for SSN 2, server 2's service time is only one third that of server 1. In practice, this setup could represent the situation where SSN 1 is physically close to server 1, and SSN 2 is physically close to server 2.

We evaluate the average delays of the three policies as SSN 1's request rate increases from 0.01 to 1.5 req/sec, and SSN 2's increases from 0.01 to 1.33 req/sec. Fig. 7 depicts the ratios of EQ_DELAY's average delay (and that of EQ_LOAD) to OPT at different request rates.

Fig. 7(a) depicts the ratio of the average delay of EQ_DELAY to that of OPT. We observe that if SSN 1 and SSN 2 have comparable request rates, then they only access



(a) EQ_DELAY.



(b) EQ_LOAD.

Fig. 7. Average delay ratio for the multi-class service time matrix defined in Eq. (53).

their most efficient server (i.e., servers 1 and 2 respectively). In this case, EQ_DELAY and OPT use the same one server for each SSN, and have the same delay. The largest difference of the policies occurs when one SSN has very low request rate and the other SSN's access probabilities are different. In this case, the system degenerates to a single-class network. In this experiment, a maximum difference of 15% is achieved when SSN 1's rate is 0.02 req/sec and SSN 2's rate is 0.67 req/sec. When both SSNs generate substantial amount of traffic, the performance difference between the two policies decreases compared to the above single-class case.

For the case of the ratio between EQ_LOAD and OPT, depicted in Fig. 7(b), it also appears that the largest difference occurs when one of the SSNs generates little traffic. When this SSN's request rate increases, the difference becomes less significant. Additionally, the observation we made in Section VI-A about EQ_LOAD's inefficiency at low load is also evident in the figure. The largest difference of 39% is achieved when SSN 1's rate is 0.01 request/sec and SSN 2's rate is 0.14 request/sec.

The above results are representative to 2×2 networks with $t_{11} < t_{12}$ and $t_{22} < t_{21}$. Additional numerical results for different network parameters can be found in [31]. They all indicate that in multi-class networks, the largest ratio between the average delay of EQ_DELAY (or EQ_LOAD) and that of OPT is likely to occur when the network degenerates to the single-class case.

VII. Conclusions

In this work, we analytically compared the performance of server selection policies in content replication networks. This problem has gained significant importance in recent years with the emergence of large-scale content delivery and peer-to-peer network architectures over the Internet. We introduced a mathematical framework, based on the $M/G/1$ Processor Sharing queueing model, which allowed us to provide quantitative, yet non-trivial, insight into the performance of server selection policies. Furthermore, we proved that our results are also applicable to the $G/G/1$-FCFS and multi-class $M/G/1$-PS queueing models, in certain asymptotic regimes.

Based on our model, we derived closed-form solutions for a hypothetical benchmark policy, called OPT, that achieves the minimum average delay. We used this benchmark policy to evaluate the performance of two server selection policies, generically referred to as EQ_DELAY and EQ_LOAD, which are representative of a large class of existing algorithms. We proved that EQ_LOAD's average delay is always larger than or equal to that of EQ_DELAY.

A major contribution of this paper is in the analytical quantification of the performance difference of EQ_DELAY and EQ_LOAD with regard to the minimal average delay. We analytically proved that, in an N-server system, the worst-case ratio of EQ_DELAY's (or EQ_LOAD's) average delay to the minimal average delay is exactly N. Moreover, we have provided analytical evidence that the same worst-case ratio is likely to prevail in multi-class settings, where different SSNs may favor different servers. We have shown that large inefficiency tends to occur in highly heterogeneous systems, but only under moderate and high utilization for the case of EQ_DELAY.

Simulations using our workload model as well as the ProWGen Web workload generator show substantial inefficiency, up to 30%, in realistic scenarios. This result is relatively insensitive to arrival and service time distributions as well as service disciplines. Other simulation results for workload exhibiting temporal locality, FCFS service models and multi-class settings show that our analytical results are general enough for a number of practical situations. Finally, we have provided a game-theoretic interpretation to our results, i.e., the price of anarchy in unbounded delay networks depends on the network topology, and the potential inefficiency of EQ_DELAY (or EQ_LOAD) grows with the scale of the network.

The purpose of the benchmark policy developed in this paper was to study the theoretical strengths and limitations of existing server selection policies. Clearly, to develop a comprehensive understanding of server selection, additional benchmarks related to other metrics (e.g., tail probabilities of delay) ought to be investigated. Another important open research area is to leverage the analytical insights provided by this work into the development of more efficient and robust server selection policies.

References

[1] K. Obraczka, P. Danzig, and D. DeLucia, “Massively replicating services in autonomously managed, wide-area internetworks,” University of Southern California, Tech. Rep. 93-541, 1993.

[2] A. Vakali and G. Pallis, “Content delivery networks: status and trends,” *IEEE Internet Computing*, vol. 7, no. 6, pp. 68–74, 2003.

[3] B. Yang and H. G. Molina, “Designing a super-peer network,” in *Proceedings of the Nineteenth International Conference on Data Engineering*, Bangalore, India, 2003.

[4] Cisco, “Cisco global site selector platforms: Chapter 1,” http://www.cisco.com/en/US/products/hw/contnetw/ps4162/products_configuration_guide_chapter09186a00800ca80e.html.

[5] V. Cardellini, E. Casalicchio, M. Colajanni, and P. Yu, “The state of the art in locally distributed Web-server systems,” *ACM Computing Surveys (CSUR)*, vol. 34, no. 2, pp. 263–311, 2002.

[6] E. Zegura, M. Ammar, Z. Fei, and S. Bhattacharjee, “Application-layer anycasting: a server selection architecture and use in a replicated Web service,” *IEEE/ACM Transactions on Networking*, vol. 8, no. 4, Aug. 2000.

[7] Y. Korilis, A. Lazar, and A. Orda, “Architecting noncooperative networks,” *IEEE Journal of Selected Areas in Communications*, vol. 13, no. 7, pp. 1241–1251, 1995.

[8] M. Stemm, R. Katz, and S. Seshan, “A network measurement architecture for adaptive applications,” in *Proceedings of IEEE INFOCOM*, Tel-Aviv,Israel, Mar. 2000.

[9] T. Ng, Y. Chu, S. Rao, K. Sripanidkulchai, and H. Zhang, “Measurement-based optimization techniques for bandwidth-demanding peer-to-peer systems,” in *Proceedings of IEEE INFOCOM*, San Francisco, CA, Apr. 2003.

[10] L. Qiu, Y. Yang, Y. Zhang, and S. Shenker, “On selfish routing in Internet-like environments,” in *Proceedings of ACM SIGCOMM*, Karlsruhe, Germany, Aug. 2003.

[11] D. Villela, P. Pradhan, and D. Rubenstein, “Provisioning servers in the application tier for e-commerce systems,” in *Proceedings of the Twelfth IEEE International Workshop on Quality of Service*, Montreal, Canada, June 2004.

[12] M. Busari and C. Williamson, “ProWGen: a synthetic workload generation tool for simulation evaluation of Web proxy caches,” *Computer Networks*, vol. 38, no. 6, 2002.

[13] J. Wardrop, “Some theoretical aspects of road traffic research,” *Proceedings of the Institute of Civil Engineers, Pt. II*, vol. 1, pp. 325–378, 1952.

[14] T. Roughgarden, *Selfish Routing and the Price of Anarchy*. MIT Press, 2005.

[15] E. Koutsoupias and C. Papadimitriou, “Worst-case equilibria,” in *Proceedings of Symposium on Theoretical Aspects in Computer Science*, Trier, Germany, Mar. 1999.

[16] T. Roughgarden and E. Tardos, “How bad is selfish routing?” *Journal of the ACM*, vol. 49, no. 2, pp. 236–259, 2002.

[17] J. Correa, A. Schulz, and N. Stier-Moses, “On the inefficiency of equilibria in congestion games,” http://jcorrea.uai.cl/papers/CSS2005.pdf, 2005.

[18] T. Roughgarden, “The price of anarchy is independent of the network topology,” *Journal of Computer and System Sciences*, vol. 67, no. 2, pp. 341–364, Sept. 2003.

[19] L. Kleinrock, *Queueing systems*. Wiley, 1976, vol. 2.

[20] S. Karlin and H. Taylor, *A First Course in Stochastic Processes*. Academic Press, 1975.

[21] Apache, “http://www.apache.org/.”

[22] S. Ranjan, R. Karrer, and E. Knightly, “Wide area redirection of dynamic content by Internet data centers,” in *Proceedings of IEEE INFOCOM*, Hong Kong, China, Mar. 2004.

[23] S. Iyer, S. Bhattacharyya, N. Taft, and C. Diot, “An approach to alleviate link overload as observed on an IP backbone,” in *Proceedings of IEEE INFOCOM*, San Francisco, CA, 2003.

[24] C. Fraleigh *et al.*, “Packet-level traffic measurements from the Sprint IP backbone,” *IEEE Network*, vol. 17, no. 6, pp. 6–16, 2003.

[25] S. Sen and J. Wang, “Analyzing peer-to-peer traffic across large networks,” *IEEE/ACM Transactions on Networking*, vol. 12, no. 2, pp. 219–232, Apr. 2004.

[26] A. Tantawi and D. Towsley, “Optimal static load balancing in distributed computer systems,” *Journal of the ACM*, vol. 32, no. 2, pp. 445–465, 1985.

[27] K. Ross and D. Yao, “Optimal load balancing and scheduling in a distributed computer system,” *Journal of the ACM*, vol. 38, no. 3, pp. 676–689, 1991.

[28] C. Kim and H. Kameda, “An algorithm for optimal static load balancing in distributed computer systems,” *IEEE Transactions on Computers*, vol. 41, no. 3, pp. 381–384, 1992.

[29] M. Gerla and L. Kleinrock, “On the topological design of distributed computer networks,” *IEEE Transactions on Communications*, vol. 25, no. 1, pp. 48–60, Jan. 1977.

[30] D. Bertsekas, *Nonlinear Programming*, 2nd ed. Athena Scientific, 1999.

[31] T. Wu, “An analytical study of server selection for scalable Internet services,” Ph.D. Dissertation, Boston University, 2007.

[32] L. Kleinrock, *Queueing systems*. Wiley, 1975, vol. 1.

[33] H. Kameda, J. Li, C. Kim, and Y. Zhang, *Optimal Load Balancing in Distributed Computer Systems*. Springer-Verlag, 1997.

[34] W. Marchal, “Some simpler bounds on the mean queueing time,” *Operations Research*, vol. 26, pp. 1083–1088, 1978.

[35] R. Wolff, *Stochastic Modeling and the Theory of Queues*. Prentice Hall, 1989.

[36] A. Zwart, “Sojourn times in a multiclass processor sharing queue,” in *Proceedings of the Sixteenth International Teletraffic Congress*, Edinburgh, UK, June 1999.

[37] M. Crovella and A. Bestavros, “Self-similarity in World Wide Web traffic: Evidence and possible causes,” *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 835–846, 1997.

[38] D. Starobinski and M. Sidi, “Modeling and analysis of power-tail distributions via classical teletraffic methods,” *Queueing Systems (QUESTA)*, vol. 36, no. 1–3, pp. 243–267, 2000.

[39] S. Ranjan, R. Swaminathan, M. Uysal, and E. Knightly, “DDoS-resilient scheduling to counter application layer attacks under imperfect detection,” in *Proceedings of IEEE INFOCOM*, Barcelona, Spain, Apr. 2006.

[40] B. Avi-Itzhak and H. Levy, “On measuring fairness in queues,” *Advances in Applied Probability*, vol. 36, no. 3, pp. 919–936, 2004.

[41] A. Mahanti, D. Eager, and C. Williamson, “Temporal locality and its impact on Web proxy cache performance,” *Performance Evaluation*, vol. 42, no. 2–3, pp. 187–203, 2000.

[42] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, “Web caching and Zipf-like distributions: evidence and implications,” in *Proceedings of IEEE INFOCOM*, New York, NY, Mar. 1999, pp. 126–134.

PLACE PHOTO HERE

Tao Wu (S’97-M’00) joined Nokia Research Center in 1999, where he is currently a Senior Research Engineer. At Nokia, he has led several research projects on mobile content delivery and multimedia management, and has over twenty papers and patents, awarded or pending, in these areas. His current research interests include human computer interface and multimedia networking and management.

Tao received a bachelor degree from Tsinghua University, an MS degree from Rice University, and a Ph.D. degree from Boston University, all in electrical engineering.

PLACE PHOTO HERE

David Starobinski (S’95-M’99-SM’07) received his Ph.D. in Electrical Engineering (1999) from the Technion-Israel Institute of Technology. In 1999-2000, he was a visiting post-doctoral researcher in the EECS department at UC Berkeley. Since September 2000, he has been at Boston University, where he is now an Associate Professor.

Dr. Starobinski received a CAREER award from the U.S. National Science Foundation and an Early Career Principal Investigator (ECPI) award from the U.S. Department of Energy. His research interests are in the modeling and performance evaluation of high-speed, wireless, and sensor networks.