Queueing System Topologies with Limited Flexibility

John N. Tsitsiklis
MIT, LIDS
Cambridge, MA 02139
jnt@mit.edu

Kuang Xu
MIT, LIDS
Cambridge, MA 02139
kuangxu@mit.edu

ABSTRACT

We study a multi-server model with n *flexible* servers and rn queues, connected through a fixed bipartite graph, where the level of flexibility is captured by the average degree, $d(n)$, of the queues. Applications in content replication in data centers, skill-based routing in call centers, and flexible supply chains are among our main motivations.

We focus on the scaling regime where the system size n tends to infinity, while the overall traffic intensity stays fixed. We show that a large capacity region (robustness) and diminishing queueing delay (performance) are jointly achievable even under very limited flexibility ($d(n) \ll n$). In particular, when $d(n) \gg \ln n$, a family of random-graph-based interconnection topologies is (with high probability) capable of stabilizing all admissible arrival rate vectors (under a bounded support assumption), while simultaneously ensuring a diminishing queueing delay, of order $\ln n/d(n)$, as $n \to \infty$. Our analysis is centered around a new class of virtual-queue-based scheduling policies that rely on dynamically constructed partial matchings on the connectivity graph. [†]

Categories and Subject Descriptors

G.3 [**Probability and Statistics**]: Queuing theory; C.2.1 [**Network Architecture and Design**]: Distributed networks

General Terms

Performance, Theory

Keywords

queueing system, flexibility, partial resource pooling, random graph, expander graph, asymptotics

[†]Research supported in part by the National Science Foundation, under grant CMMI-1234062, and an MIT-Xerox Fellowship. The authors are grateful for the feedback from the anonymous reviewers.

1. INTRODUCTION

At the heart of many modern queueing networks lies the problem of allocating processing resources (e.g., manufacturing plants, web servers, or call-center staff) to meet multiple types of demands that arrive dynamically over time (e.g., orders, data queries, or customer inquiries). It is often the case that a *fully flexible* or *completely resource-pooled* system, where every unit of processing resource is capable of serving all types of demands, delivers the best possible performance. Our inquiry is, however, motivated by the unfortunate reality that such full flexibility is often infeasible due to overwhelming implementation costs (in the case of a data center) or human skill limitations (in the case of a skill-based call center).

What are the key benefits of flexibility and resource pooling in such queueing networks? Can we harness the same benefits even when the degree of flexibility is *limited*, and how should the network be designed and operated? These are the main questions that we address in this paper. While these questions can be approached from several different angles, we will focus on the metrics of *expected queueing delay* and *capacity region*; the former is a direct reflection of *performance*, while the latter measures the system's *robustness* against *demand uncertainties*, when the arrival rates for different demand types are unknown or likely to fluctuate over time. Our main message is positive: in the regime where the system size is large, improvements in both capacity region and delay are *jointly achievable* even under very limited flexibility, given a proper choice of the interconnection topology.

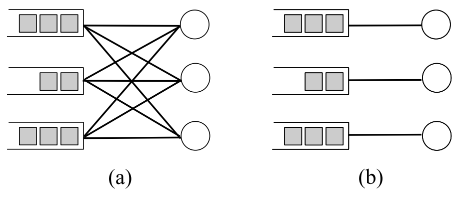

Figure 1: Extreme (in)flexibility: $d(n) = n$ vs. $d(n) = 1$.

Benefits of Full Flexibility. We begin by illustrating the benefits of flexibility and resource pooling using two simple examples.[1] Consider a system of n servers, each running at rate 1, and n queues, where each queue stores jobs of a particular demand type. For each $i \in \{1, \ldots, n\}$, queue i receives a Poisson arrival stream of rate $\lambda_i = \rho \in (0, 1)$, independently from all other queues. The job sizes are independent and exponentially distributed with mean 1.

[1]The lack of mathematical rigor in this section is intended to make the results easier to state. Formal definitions will be given in later sections.

For the remainder of this paper, we will use as a measure of flexibility the average number of servers that a demand type can receive service from, denoted by $d(n)$. Let us consider the two extreme cases: a fully flexible system, with $d(n) = n$ (Figure 1(a)), and an inflexible system, with $d(n) = 1$ (Figure 1(b)). Fixing the traffic intensity ρ, and letting the system size, n, tend to infinity, we observe the following qualitative benefits of full flexibility.

1. Diminishing Delay: Let W be the steady-state expected time in queue. In the fully flexible case and under any work-conserving scheduling policy[2], the *collection* of all jobs in the system evolves as an $M/M/n$ queue with arrival rate ρn. It is not difficult to verify that the expected total number of jobs in queue is *bounded* by a constant independent of n, and hence the expected waiting time in queue (the time from entering the queue to the initiation of service) satisfies $\mathbb{E}(W) \to 0$, as $n \to \infty$.[3] In contrast, the inflexible system is simply a collection of n *unrelated* $M/M/1$ queues, and hence the expected waiting time is $\mathbb{E}(W) = \frac{\rho}{1-\rho} > 0$, for all n. In other words, expected delay *diminishes* in a fully flexible system, as the system size increases, but stays bounded away from zero in the inflexible case.

2. Large Capacity Region: Suppose that we now allow the arrival rate λ_i to queue i to vary with i. For the fully flexible case, and treating it again as an $M/M/n$ queue, it is easy to see that the system is stable for all arrival rate vectors that satisfy $\sum_{i=1}^{n} \lambda_i < n$, whereas in the inflexible system, since all $M/M/1$ queues operate independently, we must have $\lambda_i < 1$, for all i, in order to achieve stability. Comparing the two, we see that the fully flexible system attains a much larger capacity region, and is hence more robust to uncertainties or changes in the arrival rates.

3. Joint Achievability: Finally, it is remarkable, though perhaps obvious, that a fully flexible system can achieve both benefits (diminishing delay and large capacity region) *simultaneously*, without sacrificing one for the other. In particular, for any $\rho \in (0, 1)$, the condition $\sum_{i=1}^{n} \lambda_i < \rho n$ for all n *implies* that $\mathbb{E}(W) \to 0$, as $n \to \infty$.

Preview of Main Result. Will the above benefits of flexibility continue to be present if the system is no longer fully flexible (i.e., if $d(n) \ll n$)? The main result of the paper (Theorem 1) shows that the objectives of diminishing delay and a large capacity region can still be jointly achieved, even when the amount of flexibility in the system is limited ($\ln n \ll d(n) \ll n$), as long as the arrival rates are appropriately bounded. However, when flexibility is limited, the interconnection topology and scheduling policy need to be chosen with care: our solution is based on connectivity graphs generated by the Erdös-Rényi random bipartite graph construction, combined with a new class of scheduling policies that rely on dynamically constructed partial matchings. Furthermore, the scheduling policies are completely oblivious to the exact values of λ_i, and adapt to them automatically.

1.1 Motivating Applications

We describe here some motivating applications for our model, which share a common overall architecture illustrated in Figure 2. **Content replication** is commonly used in data centers for bandwidth intensive operations such as database queries [2] or video streaming [9], by hosting the same piece of content on multiple servers. Here, a server corresponds to a physical machine in a data center, and each queue stores incoming demands for a particular piece of content (e.g., a video clip). A server j is connected to queue i if there is a copy of content i on server j, and $d(n)$ corresponds to the average number of replicas per content across the network. Similar structures also arise in **skill-based routing (SBR) in call centers**, where agents (servers) are assigned to answer calls from different categories (queues) based on their domains of expertise [15], and in **process-flexible supply chains** [3]-[8], where each plant (server) is capable of producing multiple product types (queues). In many of these applications, demand rates can be unpredictable and may change significantly over time (for instance, unexpected "spikes" in demand traffic are common in modern data centers [1]). The demand uncertainties make *robustness* an important criterion for system design. These practical concerns have been our primary motivation for studying the *trade-off* between robustness, performance, and the level of flexibility.

1.2 Related Work

Bipartite graphs serve as a natural model of the relationships between demand types and service resources. It is well known in the supply chain literature that limited flexibility, corresponding to a sparse bipartite graph, can be surprisingly effective in resource allocation even when compared to a fully flexible system [3, 4, 5, 6, 7]. The use of sparse random graphs or expanders as flexibility structures that improve robustness has recently been studied in [8] in the context of supply chains, and in [9] for content replication. Similar to the robustness results reported in this paper, these works show that expanders or random graphs can accommodate a large set of demand rate vectors. However, in contrast to our work, nearly all analytical results in this literature focus on static allocation problems, where one tries to match supply with demand in a single slot, as opposed to the queueing context, where resource allocation decisions need to be made dynamically over time.

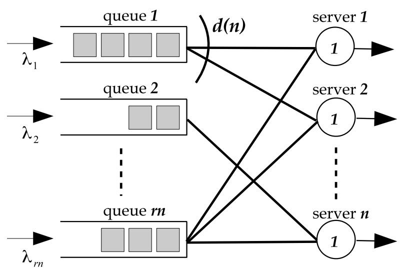

Figure 2: A processing network with rn queues and n servers.

In the queueing theory literature, the models we consider fall under the umbrella of so-called multi-class multi-server systems, where multiple queues and servers are connected through a bipartite graph. Under these (and similar) settings, complete resource pooling (full flexibility) is known to improve system performance [12, 13, 14], but much less is known when only limited flexibility is available, because systems with a non-trivial connectivity graph are very difficult to analyze, even under seemingly simple scheduling policies (e.g, first-come-first-serve) [10, 11]. Simulations in [15] show empirically that limited cross-training can be highly effective in a large call center under a skill-based routing algorithm. Using a very different set of modeling assumptions, [16] proposes a specific chaining structure with limited flexibility, which is shown to perform well under heavy traffic. Closer to the spirit of the current work is [17], which studies a partially flexible system where a fraction $p > 0$ of all processing resources are fully flexible, while the

[2]Under a work-conserving policy, a server is always busy whenever there is at least one job in the queues to which it is connected.

[3]The diminishing expected waiting time follows from the bounded expected total number of jobs in steady-state, Little's Law, and the fact that the total arrival rate is ρn, which goes to infinity as $n \to \infty$.

remaining fraction, $1-p$, is dedicated to specific demand types, and shows an exponential improvement in delay scaling under heavy-traffic. However, both [16] and [17] focus on the heavy-traffic regime, which is different from the current setting where traffic intensity is assumed to be fixed while the system size tends to infinity, and provide analytical results for the special case of uniform demand rates. Furthermore, with a constant fraction of fully flexible resources, the average degree in [17] scales linearly with the system size n, whereas we are interested in the case of a much slower degree scaling. Finally, the expected delay in the architecture considered in [17] does not tend to zero as the system size increases.

At a higher level, our work is focused on the joint trade-off between robustness, delay, and the degree of flexibility in a queueing network, which is little understood in the existing literature, especially for networks with a non-trivial interconnection topology.

On the technical end, we build on several existing ideas. The techniques of batching (cf. [18, 19]) and the use of virtual queues (cf. [20, 21]) have appeared in many contexts in queueing theory, but the specific models considered in the literature bear little resemblance to ours. The study of perfect matchings on a random bipartite graph dates back to the seminal work in [22]; while it has become a rich topic in combinatorics, we will refrain from giving a thorough literature survey because only some elementary and standard properties of random graphs are used in the current paper.

Organization of the Paper. We describe the model in Section 2 along with the notation to be used throughout. The main result is stated in Section 3, with a discussion of potential improvements postponed till Section 7. Section 3.2 studies a specific flexibility structure and compares it against the architecture proposed in this paper. The rest of the paper is devoted to the proof of our main result (Theorem 1), with an overview of the proof strategy given in Section 3.3.

2. MODEL AND NOTATION

Notation. To avoid clutter, we will minimize the use of floor and ceiling notation throughout the paper (e.g., writing T_{rn} instead of $T_{\lfloor rn \rfloor}$). We will assume that all values of interest are appropriately rounded up or down to an integer, whenever doing so does not cause ambiguity or confusion. We denote by $\mathcal{G}_n$ the set of all $rn \times n$ bipartite graphs. For $g_n \in \mathcal{G}_n$, let $\deg(g_n)$ be the average degree among the rn left vertices. An $m \times n$ Erdös-Rényi random bipartite graph $G = (E, I \cup J)$,[4] where each of the mn edges is present with probability p, is referred to as an (m,n,p) random graph. We will use $\mathbb{P}_{n,p}(\cdot)$ to denote the probability measure on $\mathcal{G}_n$ induced by an (rn, n, p) random graph,

$$\mathbb{P}_{n,p}(g) = p^{|E|}(1-p)^{rn^2-|E|}, \quad \forall g \in \mathcal{G}_n, \tag{1}$$

where $|E|$ is the cardinality of the set of edges, E. For a subset of vertices, $M \subset I \cup J$, we will denote by $g|_M$ the graph induced by g on the vertices in M. Throughout, we will use the letter K to denote a generic constant.

The following short-hand notation for asymptotic comparisons will be used; here f and g are positive functions:

1. $f(x) \lesssim g(x)$ for $f(x) = \mathcal{O}(g(x))$, and $f(x) \gtrsim g(x)$ for $f(x) = \Omega(g(x))$.
2. $f(x) \gg g(x)$ for $\liminf_{x\to\infty} \frac{f(x)}{g(x)} = \infty$, and $f(x) \ll g(x)$ for $\limsup_{x\to\infty} \frac{f(x)}{g(x)} = 0$.
3. $f(x) \sim g(x)$ for $\lim_{x\to\infty} \frac{f(x)}{g(x)} = 1$.

[4]Throughout, we denote by E, I, and J the set of edges and left and right vertices, respectively.

When labeling a sequence, we will use the notation $a(n)$ if the value of a has a meaningful dependence on n, or a_n if n serves merely as an index. We will use Expo(λ) as a shorthand for the exponential distribution with parameter λ. The expression $X \overset{d}{=} Y$ means that X and Y have the same distribution. Whenever possible, we will use upper-case letters for random variables, and lower-case letters for deterministic values.

The Model. We consider a sequence of systems operating in *continuous time*, indexed by an integer n, where the nth system consists of n servers and rn queues, where r is a positive constant that is fixed as n varies (Figure 2). The connectivity of the system is encoded by an $rn \times n$ undirected bipartite graph $g_n = (E, I \cup J)$, where I and J represent the set of queues and servers, respectively, and E the set of edges between them.[5] A server $j \in J$ is *capable* of serving a queue $i \in I$ if and only if $(i,j) \in E$. We will denote by $\mathcal{N}(i)$ the set of servers in J connected to queue i, and similarly, by $\mathcal{N}(j)$ the set of queues in I connected to server j.

In the nth system, each queue i receives a stream of incoming jobs according to a Poisson process of rate $\lambda_{n,i}$, independent of all other streams. We will denote by $\boldsymbol{\lambda}_n$ the arrival rate vector, i.e., $\boldsymbol{\lambda}_n = (\lambda_{n,1}, \lambda_{n,2}, \ldots, \lambda_{n,rn})$. The sizes of the jobs are exponentially distributed with mean 1, independent from each other and from the arrival processes. All servers are assumed to run at a constant rate of 1. The system is assumed to be empty at time $t = 0$.

Jobs arriving at queue i can be assigned to any server $j \in \mathcal{N}(i)$ to receive service. The assignment is *binding*, in the sense that once the assignment is made, the job cannot be transferred to, or simultaneously receive service from, any other server. Moreover, service is *non-preemptive*, in the sense that once service is initiated for a job, the assigned server has to dedicate its full capacity to that job until its completion.[6] Formally, if a server j just completed the service of a previous job at time t, its available actions are: **1. Serve a new job**: Server j can choose to fetch a job from any queue in $\mathcal{N}(j)$ and immediately start service. The server will remain occupied and take no other actions until the current job is completed. **2. Remain idle**: Server j can choose to remain idle. While in the idling state, it will be allowed to initiate a service (Action 1) at any point in time.

The performance of the system is fully determined by a *scheduling policy*, π, which specifies for each server $j \in J$, when to remain idle, when to serve a new job, and from which queue in $\mathcal{N}(j)$ to fetch a job when initiating a new service.

We only allow policies that are causal, in the sense that the decision at time t depends only on the history of the system (arrivals and service completions) up to t. We allow the scheduling policy to be *centralized* (i.e., to have full control over all servers' actions), and to be based on the knowledge of the graph g_n and the past history of all queues and servers. On the other hand, a policy does *not* observe the actual sizes of jobs before they are served, and **does not know the arrival rate vector $\boldsymbol{\lambda}_n$**.

Performance metric: Let W_k be the waiting time in queue for the kth job, defined as the time from the job's arrival to a queue until

[5]For notational simplicity, we omit the dependence of E, I, and J on n.

[6]While we restrict ourselves to binding and non-preemptive scheduling polices in this paper, other common architectures where (a) a server can serve multiple jobs concurrently (processor sharing), (b) a job can be served by multiple servers concurrently, or (c) job sizes are revealed upon entering the system, are clearly more powerful than the current setting, and are therefore capable of implementing the scheduling policy we consider here. Hence the performance upper bounds developed in this paper also apply to these more powerful variants.

when it starts to receive service from some server. With a slight abuse of notation, we define the *expected waiting time*, $\mathbb{E}(W)$, as

$$\mathbb{E}(W) \overset{\triangle}{=} \limsup_{k\to\infty} \mathbb{E}(W_k). \tag{2}$$

Note that $\mathbb{E}(W)$ captures the *worst-case* expected waiting time across all jobs in the long run, and is always well defined, even under scheduling policies that do not induce a steady-state distribution.

We are primarily interested in the scaling of $\mathbb{E}(W)$ in the regime where the total traffic intensity (i.e., the ratio between the total arrival rate and the total service capacity) stays fixed, while the size of the system, n, tends to infinity.

3. MAIN THEOREM

Before stating our main theorem, we first motivate a condition on the arrival rate vector, $\boldsymbol{\lambda}_n$. Since we allow the arrival rates $\lambda_{n,i}$ to vary with i, and since on average each queue is connected to $d(n)$ servers, the range of fluctuations of the $\lambda_{n,i}$ with respect to i should not be too large compared to $d(n)$, or else the system would become unstable. We will therefore let the amount of *rate fluctuation* or *uncertainty* be bounded by some $u(n)$, an upper bound on $\lambda_{n,i}$ for all i. The following condition summarizes these points.

CONDITION 1. **(Rate Condition)** *We fix a constant $\rho \in (0,1)$ (referred to as the* traffic intensity*) and a sequence $\{u(n)\}_{n\ge 1}$ of positive reals with $\inf_n u(n) > 0$. For any $n \ge 1$, the arrival rate vector $\boldsymbol{\lambda}_n$ satisfies:*

1. $\max_{1\le i\le rn} \lambda_{n,i} \le u(n)$.

2. $\sum_{i=1}^{rn} \lambda_{n,i} \le \rho n$.

We denote by $\boldsymbol{\Lambda}_n \subset \mathbb{R}_+^n$ the set of all arrival rate vectors that satisfy the above conditions.

We denote by $\mathbb{E}_\pi(W \mid g_n, \boldsymbol{\lambda}_n)$ the expected waiting time (cf. Eq. (2)) under a scheduling policy π, a graph g_n, and an arrival rate vector $\boldsymbol{\lambda}_n$. The following is the main result of the paper.

THEOREM 1. *We assume that $d(n) \gg \ln n$ and $u(n) \ll \sqrt{\frac{d(n)}{\ln n}}$, and fix $\rho \in (0,1)$, $\gamma > 0$, and $n \ge 1$. Then, there exists a policy π_n such that for every arrival vector $\boldsymbol{\lambda}_n$ that satisfies Condition 1, the following hold.*

1. There exists a bipartite graph, $g_n \in \mathcal{G}_n$ (possibly depending on $\boldsymbol{\lambda}_n$) such that $\deg(g_n) \le (1+\gamma)d(n)$, *and*

$$\mathbb{E}_{\pi_n}(W \mid g_n, \boldsymbol{\lambda}_n) \le \frac{Ku^2(n)\ln n}{d(n)} = o(1), \tag{3}$$

where $K > 0$ is a constant independent of n, g_n and $\boldsymbol{\lambda}_n$.

2. Let G_n be a random graph in $\mathcal{G}_n$, chosen at random according to the Erdös-Rényi measure $\mathbb{P}_{n,\frac{d(n)}{n}}$ (cf. Eq. (1)). Then G_n has the property in Part 1, with high probability, uniformly over all $\boldsymbol{\lambda}_n \in \boldsymbol{\Lambda}_n$. Formally,

$$\mathbb{P}_{n,\frac{d(n)}{n}}\Big(G_n \textit{ has the property in Part 1}\Big) \ge 1-\delta_n, \quad \forall \boldsymbol{\lambda}_n \in \boldsymbol{\Lambda}_n,$$

where $\{\delta_n\}_{n\ge1}$ is a sequence wth $\lim_{n\to\infty}\delta_n = 0$.

The reader is referred to Section 7 for a discussion on potential improvements of the result.

3.1 Remarks on Theorem 1

Note that Part 1 of the theorem is a special case of Part 2, which states that for large values of n, and for a graph chosen according to the Erdös-Rényi model, the scheduling policy will have a large probability (with respect to the random graph generation) of achieving diminishing delay, as in Eq. (3). At a higher level, Eq. (3) relates three key characteristics of the system: *delay* $\mathbb{E}(W)$, *level of robustness* $u(n)$, and *degree of flexibility* $d(n)$. Furthermore, the scheduling policy only requires knowledge of ρ, not of the arrival rate vector; this is important in practice because this vector need not be known *a priori* or may change significantly over time.

The fact that a large capacity region (as given by Condition 1) is achievable when $d \ll n$ should not be too surprising: expander graphs (such as those obtained through a random graph construction) are known to have excellent max-flow performance. The more difficult portion of the argument is to show that a diminishing delay (a temporal property) can also be achieved without significantly sacrificing the range of admissible arrival rates, through appropriately designed scheduling policies. In our proof, the achievability of a large capacity region is not shown explicitly, but comes as a by-product of the delay analysis; in fact, the scheduling policy automatically finds, over time, a feasible flow decomposition over the connectivity graph (see Section 6).

Dependence on ρ.
It is possible to incorporate the traffic intensity ρ in the leading constant in Eq. (3), to obtain a bound of the form

$$\mathbb{E}_{\pi_n}(W \mid g_n, \boldsymbol{\lambda}_n) \le \frac{K'}{1-\rho} \cdot \frac{u^2(n)\ln n}{d(n)}, \tag{4}$$

where K' is a constant independent of ρ. This captures a trade-off between the traffic intensity and the degree of flexibility in the system. A proof of Eq. (4) is given in [26].

Adversarial Interpretation of Rate Robustness.
To better understand the power and limitations of the "rate robustness" entailed by Theorem 1, it is useful to view the system as a game between two players: an *Adversary* who chooses the rate vector $\boldsymbol{\lambda}_n$, and a *Designer* who chooses the interconnection topology g_n and the scheduling policy π_n. The goal of the Designer is to achieve small average waiting time (which would also imply stability of all queues), while the goal of the Adversary is the opposite. The following definition will facilitate our discussion.

DEFINITION 1. **(Good Graphs)** *Let us fix n, $d(n)$, $u(n)$, ρ, γ, and K, as in Theorem 1. We define the set* Good$(\boldsymbol{\lambda}_n)$ *of* good graphs *for $\boldsymbol{\lambda}_n$ as the subset of $\mathcal{G}_n$ for which* $\deg(g_n) \le (1+\gamma)d(n)$ *and the inequality in Eq.* (3) *holds, for some policy π_n.*

We now examine the robustness implications of Theorem 1 under different *orderings of the actions* in the game, listed according to increasing levels of difficulties for the Designer.

Level 1: Adversary acts first (weak adversary). The Designer gets to pick g_n and π_n *after* observing the realization of $\boldsymbol{\lambda}_n$. Note that this weak form of adversary fails to capture the rate uncertainty arising in many applications, where the arrival rates are unknown at the time when the system is designed. For this setting, it is not hard to come up with a simple deterministic construction of a good graph and a corresponding policy. Formally, the set Good$(\boldsymbol{\lambda}_n)$ is nonempty for every $\boldsymbol{\lambda}_n \in \boldsymbol{\Lambda}_n$. However, Part 1 of Theorem 1 actually makes a stronger statement. While the graph is chosen after observing $\boldsymbol{\lambda}_n$, the policy is designed without knowledge of $\boldsymbol{\lambda}_n$, albeit at the expense of a much more complex policy.

Level 2: Adversary and Designer act independently (intermediate adversary). Suppose that both the Adversary and the Designer make their decisions independently, *without* knowing each other's choice. This case models most practical applications, where $\boldsymbol{\lambda}_n$ is generated by some exogenous mechanism, unrelated to the design process, and is not revealed to the Designer ahead of time. Here, our randomization in the construction of the graph offers protection against the choice of the Adversary. Part 2 of the theorem can be rephrased into the statement that there exists a policy for which

$$\inf_{\boldsymbol{\lambda}_n \in \boldsymbol{\Lambda}_n} \mathbb{P}_{n, \frac{d(n)}{n}} \left(G_n \in \text{Good}\left(\boldsymbol{\lambda}_n\right) \right) \to 1. \tag{5}$$

As an extension, we may consider the case where the Adversary chooses $\boldsymbol{\lambda}_n$ at random, according to some probability measure μ_n on $\boldsymbol{\Lambda}_n$, but still independently from G_n. However, this additional freedom to the Adversary does not make a difference, and it can be shown, through an easy application of Fubini's Theorem, that

$$\inf_{\mu_n} \left(\mathbb{P}_{n, \frac{d(n)}{n}} \times \mu_n \right) \left(G_n \in \text{Good}\left(\boldsymbol{\lambda}_n\right) \right) \to 1, \tag{6}$$

and where $\times$ is used to denote product measure. The proof is given in [26].

Level 3: Designer acts first (strong adversary). In this case, the Adversary chooses $\boldsymbol{\lambda}_n$ *after* observing the g_n and π_n picked by the Designer. While such an adversary may be too strong for most practical applications, it is still interesting to ask whether there exist *fixed* g_n and π_n that will work well for *any* $\boldsymbol{\lambda}_n$ that satisfies Condition 1 (or even weaker conditions). We conjecture that this is the case.

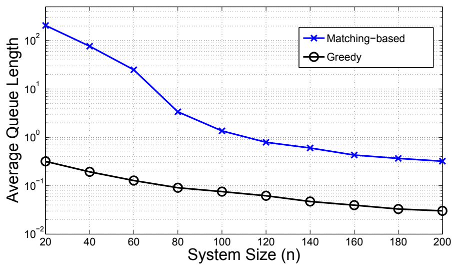

Figure 3: ***Diminishing Delay.*** **Simulations with** $n \times n$ **networks** ($r = 1$)**,** $\lambda_{n,i} = \rho = 0.95$**, and** $d(n) = \sqrt{n}$**.**

On Practical Scheduling Policies.

The scheduling policy that we will use in the proof of Theorem 1 was mainly designed for the purpose of analytical tractability, rather than practical efficiency. For instance, simulations suggest that a seemingly naive greedy heuristic, where any free server chooses a job from a longest queue to which it is connected, can achieve a superior delay scaling (see Figure 3).[7] Unfortunately, deriving an explicit delay upper bound for the greedy policy or other similar heuristics appears to be challenging. See also the discussion in Section 7.

3.2 A Comparison with Modular Architectures

Assume for simplicity that there is an equal number of queues and servers (i.e., $r = 1$). In a *modular architecture*, the designer partitions the system into $n/d(n)$ *separate* sub-networks. Each sub-network consists of $d(n)$ queues and servers that are fully connected within the sub-network (Figure 4). Note that this architecture guarantees a degree of exactly $d(n)$, by construction. Assume that all queues have the same arrival rate $\rho \in (0, 1)$ and that each server uses an arbitrary work-conserving service policy. Since each sub-network is *fully connected*, it is not difficult to see that the modular architecture achieves a diminishing queueing delay as $n \to \infty$, as long as $d(n) \to \infty$. This seems even better than the random-graph-based architecture, which requires $d(n)$ to scale at least as fast as $\ln n$. However, this architecture fares much worse in terms of robustness, as we now proceed to discuss.

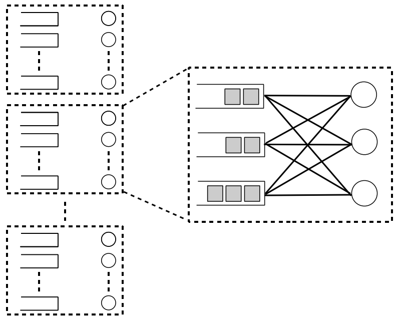

Figure 4: A modular architecture, consisting of $n/d(n)$ **discounted clusters, each of size** $d(n)$**. Within each cluster, all servers are connected to all queues.**

Since the sub-networks are completely disconnected from each other, the set of arrival rates that are admissible is severely reduced due to the requirement that the total arrival rate in *each* sub-network be less than $d(n)$, which is much more restrictive than the rate constraint given in Condition 1. Thus, the modular design may achieve a better delay scaling when the arrival rates are (almost) uniform, but at the risk of instability when the arrival rates are random or chosen adversarially, because the processing resources are not shared across different sub-networks.

3.3 Proof Overview

Sections 4 through 6 contain the proof of Theorem 1. We will provide an explicit construction of scheduling policies π_n, and then show that they achieve the delay scaling in Eq. (3). At the core of the construction is the use of *virtual queues*, whose operation is described in detail in Section 4.

The proof of Theorem 1 is completed in two steps. In Section 5, we first prove the result in the special case where all λ_i are equal to some $\lambda < 1/r$. The symmetry brought about by the uniform arrival rates will significantly simplify the notation, while almost all intuition developed here will carry over to the proof for the general case. We then show in Section 6 that, perhaps surprisingly, the original scheduling policies designed for the uniform arrival rate case can be applied directly to achieve the delay scaling in Eq. (3) for *any* $\boldsymbol{\lambda}_n$ satisfying Condition 1.

4. VIRTUAL QUEUE AND THE SCHEDULING POLICY

This section provides a detailed construction of a virtual-queue-based scheduling policy that achieves the delay scaling in Theorem 1. We begin by describing some high-level ideas behind our design.

Regularity vs. Discrepancies. Setting aside computational issues, an efficient scheduling policy is difficult to design because the future is unpredictable and random: one does not know *a pri-*

[7]The scheduling policy being simulated is a discrete-time version of the continuous-time policy analyzed in this paper.

ori which part of the network will become more loaded, and hence current resource allocation decisions must take into account all possibilities of future arrivals and job sizes.

However, as the size of the system, n, becomes large, certain *regularities* in the arrival processes begin to emerge. To see this, consider the case where $r = 1$, $\lambda_{n,i} = \lambda < 1$ for all n and i, and assume that at time $t > 0$, all servers are busy serving some job. Now, during the interval $[t, t+\gamma_n)$, "roughly" $\lambda n \gamma_n$ new jobs will arrive, and $n\gamma_n$ servers will become available. For this $[t, t+\gamma_n)$ interval, denote by Γ the set of queues that received a job, and by Δ the set of roughly $n\gamma_n$ servers that became available. If γ_n is chosen so that $\lambda n \gamma_n \ll n$, these incoming jobs are likely to spread out across the queues, so that most queues receive at most one job. Assuming that this is indeed the case, we see that the connectivity graph g_n restricted to $\Gamma \cup \Delta$, denoted $g_n|_{\Gamma\cup\Delta}$, is a subgraph sampled uniformly at random among all $(\lambda n\gamma_n \times n\gamma_n)$-sized subgraphs of g_n. When $n\gamma_n$ is sufficiently large, and g_n is *well connected* (as in an Erdös-Rényi random graph with appropriate edge probability), we may expect that, with high probability, $g_n|_{\Gamma\cup\Delta}$ admits a matching (Definition 4) that includes the entire set Γ, in which case *all* $\lambda n\gamma_n$ jobs can start receiving service by the end of the interval.

Note that when n is sufficiently large, despite the randomness in the arrivals, the symmetry in the system makes delay performance at a short time scale *insensitive* to the exact locations of the arrivals. Treated collectively, the structure of the set of arrivals and available servers in a small interval becomes less random and more "regular," as $n \to \infty$. Of course, for any finite n, the presence of randomness means that *discrepancies* (events that deviate from the expected regularity) will be present. For instance, the following two types of events will occur with small, but non-negligible, probability.

1. Arrivals may be located in a poorly connected subset of g_n.
2. Arrivals may concentrate on a small number of queues.

We need to take care of these discrepancies, and show that their negative impact on performance is insignificant.

Following this line of thought, our scheduling policy aims to use most of the resources to dynamically target the *regular* portion of the traffic (via matchings on subgraphs), while ensuring that the impact of the *discrepancies* is well contained. In particular, we will create two classes of *virtual queues* to serve these two objectives:

1. A single *Matching queue* that targets *regularity* in arrival and service times, and *discrepancies* of the first type.
2. A collection of rn *Residual queues*, one for each (physical) queue, which targets the discrepancies in arrival and service times of the second type.

The queues are "virtual," as opposed to "physical," in the sense that they merely serve to conceptually simplify the description of the scheduling policy.

Good Graphs: The management of the virtual queues must comply with the underlying connectivity graph, g_n, which is fixed over time. We informally describe here some desired properties of g_n; more detailed definitions and performance implications will be given in subsequent sections, as a part of the queueing analysis. We are interested in graphs that belong to a set $\mathcal{H}_n$, defined as the intersection of the following three subsets of $\mathcal{G}_n$:

1. $\hat{\mathcal{G}}_n$ (cf. Lemma 1): For the case $r = 1$, $g_n \in \hat{\mathcal{G}}_n$ if it admits a full matching. For the more general case, a suitable generalization is provided in the context of Lemma 1. This property will be used in both Matching and Residual queues to handle *discrepancies*.
2. $\tilde{\mathcal{G}}_n$ (cf. Lemmas 5-7) We have $g_n \in \tilde{\mathcal{G}}_n$ if a randomly sampled sublinear-sized subgraph admits a full matching, with high probability. This property will be used in the Matching queue to take advantage of *regularity*.
3. $\mathcal{L}_n$ (cf. Section 5.3) : We have $g_n \in \mathcal{L}_n$ if g_n has average degree approximately equal to $d(n)$. This property is to comply with our degree constraint.

Once the description of the policy is completed, the proof will consist of establishing the following:

(i) If $g_n \in \mathcal{H}_n$, then the claimed delay bound holds.

(ii) The probability that a random bipartite graph belongs to $\mathcal{H}_n$ tends to 1, as $n \to \infty$.

Inputs to the Scheduling Policy: Besides n and r, the scheduling policy uses the following inputs:

1. ρ, the traffic intensity as defined in Condition 1,
2. ϵ, a constant in $(0, 1-\rho)$,
3. $b(n)$, a batch size function,
4. g_n, the interconnection topology.

4.1 Arrivals to Virtual Queues

The arrivals to the Matching and Residual queues are arranged in *batches*. Roughly speaking, a batch is a set of jobs that are treated collectively as a single entity that arrives to a virtual queue. We define a sequence of random times $\{T_B(k)\}_{k\in\mathbb{Z}_+}$, by letting $T_B(0) = 0$, and for all $k \geq 1$,

$$T_B(k) \triangleq \text{time of the } k\frac{\rho}{r}b(n)\text{th arrival to the system,}$$

where $b(n) \in \mathbb{Z}_+$ is a design parameter, and will be referred to as the *batch parameter*.[8] We will refer to the time period $(T_B(k-1), T_B(k)]$ as the **kth batch period**.

DEFINITION 2. **(Arrival Times to Virtual Queues)** *The time of arrival of the kth batch to all virtual queues is $T_B(k)$, and the corresponding interarrival time is $A(k) \triangleq T_B(k+1) - T_B(k)$.*

While all virtual queues share the same arrival times, the contents of their respective batches are very different. We will refer to them as the *Matching batch* and *Residual batch*, respectively.

The kth *Matching batch* is the set of jobs that *first* arrive to their respective queues during the kth batch period. Since for each queue only the first job belongs to the Matching batch, it is convenient to represent the Matching batch as the set of queues that receive at least one job during the batch period.

The kth batch arriving to the ith Residual queue is the set of all jobs, except for the first one, that arrive to queue i during the kth batch.

DEFINITION 3. **(Size of a Residual Batch)**. *Let $H_i(k)$ be the total number of jobs arriving to queue i during the kth batch period. The size of the ith Residual batch is*

$$R_i(k) \triangleq (H_i(k) - 1)^+ . \tag{7}$$

A graphical illustration of the Matching and Residual batches is given in Figure 5.

[8]We chose not to absorb the constant ρ/r into $b(n)$ at this point because this will allow for simpler notation in subsequent sections.

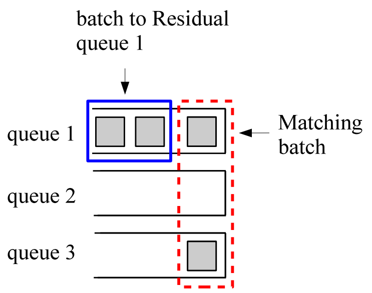

Figure 5: An illustration of the arrivals during a single batch period.

4.2 State Transitions and Service Rules

This section describes how the batches will be served by the physical servers. Before getting into the details, we first describe the general ideas.

1. The sizes of Residual batches are typically quite small, and the physical servers will process them using a (non-adaptive) randomized scheduling rule.
2. For each Matching batch, we will first "collect" a number of available servers, equal to the size of the batch:
(a) With high probability, all jobs in the Matching batch can be simultaneously processed by these servers through a matching over g_n.
(b) With small probability, some jobs in the Matching batch are located in a poorly connected subset of g_n and cannot be matched with the available servers. In this case, all jobs in the batch will be served, one at a time, according to a *fixed* server-to-queue assignment (a "clear" phase).

To implement the above queueing dynamics, we will specify the evolution of *states* and *actions* of all physical servers and virtual queues, to be described in detail in the remainder of this section.

4.2.1 States and Actions of (Physical) Servers and Residual Queues

We first introduce the notion of an assignment function, which will be used to schedule the physical servers. An assignment function L maps each queue $i \in I$ to a server $L(i) \in J$. We say that L is an *efficient assignment function* for the connectivity graph g_n if $(i, L(i)) \in E$ for all $i \in I$, and

$$\left|L^{-1}(j)\right| \le r + 1, \tag{8}$$

for all $j \in J$. As will become clear in the sequel, our scheduling policy will use an assignment function L to ensure that every Residual queue receives at least a "minimum service rate" from some server. Let $\hat{\mathcal{G}}_n$ be the set of all $g_n \in \mathcal{G}_n$ such that g_n has an efficient assignment function. With our random graph construction, an efficient assignment function exists with high probability. The proof can be found in [26].

LEMMA 1. *Let $p(n) = d(n)/n$ and assume that $d(n) \gg \ln n$. Then,*

$$\lim_{n\to\infty} \mathbb{P}_{n,p(n)}\left(\hat{\mathcal{G}}_n\right) = 1.$$

A physical server can be, at any time, in one of two states: "BUSY" and "STANDBY;" cf. Fig. 4.2.1. The *end* of a period in the BUSY state will be referred to as an *event point*, and the time interval between two consecutive event points an *event period*. At each event point, a new decision is to be made as to which virtual queue the server will choose to serve, during the next event period. All servers are initialized in a BUSY state, with the time until the first event point distributed as Expo (1), independently across all servers.

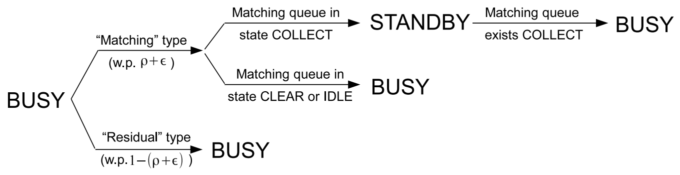
BUSY
"Matching" type
(w.p. ρ+ϵ)
Matching queue in state COLLECT
STANDBY
Matching queue exists COLLECT
BUSY
Matching queue in state CLEAR or IDLE
BUSY
"Residual" type
(w.p. 1−(ρ+ϵ))
BUSY

Figure 6: State evolution of a physical server.

Recall that ϵ be a parameter in $(0, 1-\rho)$. The state transitions defined below also involve the current state of the Matching queue, whose evolution will be given in Section 4.2.2. When a server $j \in J$ is at the kth event point:

1. With probability $\rho+\epsilon$, the kth event period is of type "matching" :
(a) If the Matching queue is in state COLLECT, server j enters state STANDBY.
(b) If the Matching queue is in state CLEAR, let B' be the Matching batch at the front of the Matching queue, and let $M' \subset I$ be the set of queues that still contain an unserved job from batch B'. Let $i^* = \min\{i : i \in M'\}$.
    i) If $L(i^*) = j$, then server j starts processing the job in queue i^* that belongs to B', entering state BUSY.
    ii) Else, server j goes on a vacation of length Expo (1), entering state BUSY.

   (c) If the Matching queue is in state IDLE, server j goes on a vacation of length Expo (1), entering state BUSY.
2. With probability $1-(\rho+\epsilon)$, the kth event period is of type "residual." Let i be an index drawn uniformly at random from the set $L^{-1}(j)$.
(a) If the ith Residual queue is non-empty, server j starts processing a job from the Residual batch that is currently at the front of the ith Residual queue, entering state BUSY.[9]
(b) Else, server j goes on a vacation of length Expo (1), entering state BUSY.

The above procedure describes all the state transitions for a single server, except for one case: when in state STANDBY, any server can be ordered by the Matching queue to start processing a job, or initiate a vacation period. The transition out of the STANDBY state will be specified in the next subsection.

4.2.2 States and Actions of the Matching Queue

As mentioned earlier, the Matching queue is not a physical entity, but a mechanism that coordinates the actions of the physical servers. The name "Matching" reflects the fact that this virtual queue is primarily focused on using matchings to schedule batches, where a matching is defined as follows.

DEFINITION 4. **(Matching in a Bipartite Graph)** *Let $F \subset E$ be a subset of the edges of $g = (E, I \cup J)$. We say that F is a* matching, *if $|\{j' : (i, j') \in F\}| \le 1$, and $|\{i' : (i', j) \in F\}| \le 1$, for all $i \in I, j \in J$. A matching F is said to* contain *$S \subset I \cup J$ if for all $s \in S$, there exists some edge in F that is incident on s, and is said to be* full *if it contains I or J.*

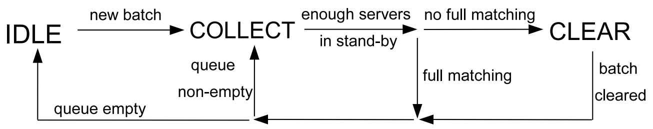

Figure 7: State evolutions of the Matching queue.

The Matching queue can be in one of three states: IDLE, COLLECT, and CLEAR. It is initialized to state IDLE.

1. If the Matching queue is in state IDLE, it remains so until the arrival of a Matching batch, upon which the Matching queue enters state COLLECT.
2. If the Matching queue is in state COLLECT, it remains so until there are exactly $(\rho/r)b(n)$ servers in state STANDBY.[10] Denote by $\Gamma \subset I$ the Matching batch currently at the front of the Matching queue, and by $\Delta \subset J$ the set of servers in STANDBY at the end of the COLLECT period. The Matching queue makes the following decisions:
   (a) If $g_n|_{\Gamma\cup\Delta}$ admits a full matching F:
   i) Let all matched servers in Δ start processing a job in the current Matching batch according to the matching F, entering state BUSY. If there are unmatched servers in Δ (which will occur if $|\Gamma| < |\Delta| = (\rho/r)b(n)$), let all unmatched servers enter state BUSY by initiating a vacation, with a length independently distributed according to $\text{Expo}(1)$.
   ii) This completes the departure of a Matching batch from the Matching queue. If the Matching queue remains non-empty at this point, it returns to state COLLECT (to deal with the rest of the Matching batch); else, it enters state IDLE.
   (b) If $g_n|_{\Gamma\cup\Delta}$ does not admit a full matching, the Matching queue enters state CLEAR. All servers in state STANDBY enter state BUSY by initiating a vacation, with a length independently distributed according to $\text{Expo}(1)$.
3. If the Matching queue is in state CLEAR, it remains so until all jobs in the current Matching batch have started receiving processing from one of the servers, upon which it enters state COLLECT if the Matching queue remains non-empty, or state IDLE, otherwise.

By now, we have described how the batches are formed (Section 4.1), and how each physical server operates (Section 4.2). The scheduling policy is hence fully specified, provided that the underlying connectivity graph g admits an efficient assignment function.

5. DYNAMICS OF VIRTUAL QUEUES – UNIFORM ARRIVAL RATES

We now analyze the queueing dynamics induced by the virtual-queue-based scheduling policy introduced in Section 4. In this section, we will prove Theorem 1 for the special case of uniform arrival rates, where

$$\lambda_{n,i} = \lambda < 1/r, \tag{9}$$

$i \in I$. In particular, we have $\rho = \lambda r < 1$. Starting with this section, we will focus on a specific batch size function of the form

$$b(n) = K\frac{n\ln n}{d(n)} = K\frac{n}{y(n)}, \tag{10}$$

[9]The Residual batch at the front of the Residual queue is defined to be the oldest Residual batch that contains any as yet unserved job.

[10]This particular number is equal to the total number of jobs arriving in a single batch period.

for a suitably chosen constant K, where $y(n)$ is defined by

$$y(n) = \frac{d(n)}{\ln n}. \tag{11}$$

The specifics of the choice of K will be given later. We have assumed that $d(n) \gg \ln n$, and hence $y(n) \to \infty$ as $n \to \infty$. Note that this implies that $b(n) \ll n$. As will be seen soon, this guarantees that only a small fraction of the jobs will enter the Residual queues.

The main idea behind the delay analysis is rather simple: we will treat each virtual queue as a $GI/GI/1$ queue, and use Kingman's bound to derive an upper bound on the expected waiting time in queue. The combination of a batching policy with Kingman's bound is a fairly standard technique in queueing theory for deriving delay upper bounds (see, e.g., [19]). Our main effort will go into characterizing the various queueing primitives associated with the virtual queues (arrival rates, traffic intensity, and variances of inter-arrival and processing times), as well as in resolving the statistical dependence induced by the interactions among the physical servers. We begin by stating a simple fact on the inter-arrival-time distribution for the virtual queues. Even though in this section we assume that $r = 1$, we state the lemma for the general case.

LEMMA 2. *The inter-arrival times of batches to the virtual queues, $\{A(k)\}_{k\geq 1}$, are i.i.d., with $\mathbb{E}(A(k)) = b(n)/(rn)$, and $\text{Var}(A(k)) \lesssim b(n)/n^2$.*

PROOF. By definition, $A(k)$ is equal in distribution to the time until a Poisson process with rate λrn records $\lambda b(n)$ arrivals. Therefore, $\mathbb{E}(A(k)) = (\lambda b(n))/(\lambda rn) = b(n)/(rn)$, and $\text{Var}(A(k)) = \lambda b(n) \cdot \frac{1}{(\lambda rn)^2} \lesssim b(n)/n^2$. $\square$

DEFINITION 5. **(Service Times in Virtual Queues)** *Consider the kth batch arriving at a virtual queue (Matching or Residual). Define the time of service initiation, E_k, to be when the batch first reaches the front of the queue, and the time of departure, D_k, to be when the last job in the batch starts receiving service from one of the physical servers. Let $D_k = E_k$ if the batch contains no jobs. The service time for the k batch is defined to be*

$$S(k) = D_k - E_k.$$

The interval $[E_k, D_k]$ is referred to as the service period of the kth batch.

We will denote by $S^M(k)$ and $S^R_j(k)$ the service time for the kth batch in the Matching queue and the jth Residual queue, respectively. Note that our scheduling policy (Section 4.2) induces interactions among the physical servers, and hence in general, the service times in a Residual queue are neither independent across batches or queues nor identically distributed.

5.1 Delays in Matching Queues

We first turn our attention to the service times $S^M(k)$ of the batches in the Matching queue. Based on our construction, it is not difficult to verify that the $S^M(k)$ are i.i.d. for any fixed graph, g_n. Furthermore, the value of $S^M(k)$ is equal to the length of a COLLECT period plus possibly the length of a CLEAR period, if the subgraph formed during the COLLECT period, $g_n|_{\Gamma\cup\Delta}$, fails to contain a matching that includes all queues in the current batch. We will therefore write

$$S^M(k) \overset{d}{=} S_{col} + X_{q(g_n)} \cdot S_{cle}, \tag{12}$$

where S_{col} and S_{cle} correspond to the length of a COLLECT and CLEAR period, respectively, and X_q is a Bernoulli random variable with $\mathbb{P}(X_q = 1) = q$. The value of $q(g_n)$ is defined by

$$q(g_n) = \mathbb{P}\{g_n|_{\Gamma\cup\Delta} \text{ does not admit a full matching}\}, \quad (13)$$

where the probability is taken over the randomness in Γ and Δ, but is conditional on $G_n = g_n$.

We begin by analyzing the properties of S_{cle}. Recall from Section 4.2.1 that, when the Matching queue is in state CLEAR, the jobs in the current Matching batch are to be served in a sequential fashion (Step 1-(b)) by a designated server. In particular, letting i^* be the smallest index of the queues that contain an unserved job from the current Matching batch, then the job in queue i^*, denoted by b, will be served by server $L(i^*)$ whenever that server enters an event period of type "matching." No other unprocessed jobs in the Matching batch can be served until job b begins to receive service, and the amount of time it takes until this occurs is exponentially distributed with mean $1/(\rho+\epsilon)$, since the probability for an event period at a physical server to be of type "matching" is $(\rho+\epsilon)$ (Step 1). Since there are at most $\lambda b(n)$ jobs in a Matching batch, the length of a CLEAR period, S_{cle}, is no greater than the amount of time it takes for a Poisson process of rate $\rho+\epsilon$ to record $\lambda b(n)$ arrivals. Arguing similar to the proof of Lemma 2, we have the following lemma. Note that the distribution of S_{cle} does not depend on the structure of g_n, as long as g_n admits an efficient assignment function (cf. Lemma 1).

LEMMA 3. *$\mathbb{E}(S_{cle}) = \frac{\lambda}{\rho+\epsilon}b(n)$, and $\mathbb{E}(S_{cle}^2) \lesssim b^2(n)$.*

To analyze S_{col}, we argue as follows. A COLLECT period consists of the time until a first server in Δ completes service, followed by the time until one of the remaining servers completes service, etc., until we have a total of $\lambda b(n)$ service completions. Thus the length of the period is the sum of $\lambda b(n)$ independent exponential random variables with parameters $\lambda(\rho+\epsilon)n$, $\lambda(\rho+\epsilon)(n-1), \ldots, \lambda(\rho+\epsilon)(n-\lambda b(n)+1)$. Using the fact that $b(n) \ll n$, this is essentially the same situation as in our analysis of S_{cle}, and we have the following result.

LEMMA 4. *$\mathbb{E}(S_{col}) \sim (\lambda/(\rho+\epsilon)) \cdot (b(n)/n)$ and $\mathbb{E}(S_{col}^2) \lesssim b^2(n)/n^2$.*

Finally, we want to focus our attention to a set $\tilde{\mathcal{G}}_n \subset \mathcal{G}_n$ of graphs that possess the following two properties.

1. The value of $\mathbb{E}(X_{q(g_n)}) = q(g_n)$ is small, for all $g_n \in \tilde{\mathcal{G}}_n$. This property will help us upper bound the service time $S^M(k)$, using Eq. (12) and the moment bounds for S_{cle} and S_{col} developed earlier.
2. The set $\tilde{\mathcal{G}}_n$ has high probability under the Erdös-Rényi random graph model.

We start with some definitions. For $m \le n$, let $\mathcal{M}_{m,n}$ be the family of all $m \times m$ subsets of $I \cup J$, that is, $h \in \mathcal{M}_{m,n}$ if and only if $|h \cap I| = |h \cap J| = m$. Let $m(n)$ be such that $m(n) \to \infty$ as $n \to \infty$. Let $\mathbf{P}_{\mathcal{M}_{m(n),n}}$ be a probability measure on $\mathcal{M}_{m(n),n}$. Let, for each $g \in \mathcal{G}_n$,

$$l(g) = \mathbf{P}_{\mathcal{M}_{m(n),n}}\big(\{h \in \mathcal{M}_{m(n),n} : g|_h \text{ does not admit a full matching}\}\big),$$

and $p(n) = d(n)/n$. We now define $\tilde{\mathcal{G}}_n$ as the set of graphs in $\mathcal{G}_n$ that satisfy

$$l(g) \le m^2(n)\,(1-p(n))^{m(n)}. \quad (14)$$

Informally, this is a set of graphs which, for the given measure $\mathbf{P}_{\mathcal{M}_{m(n),n}}$ on subgraphs, have a high probability that a random subgraph will have a full matching. Consistent with the general outline of the proof given in Section 4, we will show that a) graphs in $\tilde{\mathcal{G}}_n$ have favorable delay guarantees (Proposition 1) and b) that random graphs are highly likely to belong to $\tilde{\mathcal{G}}_n$ (Lemma 5).

LEMMA 5. **(Probability of Full Matching on Random Subgraphs)** *With $p(n) = d(n)/n$, we have*

$$\lim_{n\to\infty} \mathbb{P}_{n,p(n)}(\tilde{\mathcal{G}}_n) = 1, \quad (15)$$

Remark on Lemma 5: Eq. (15) states that graphs in $\tilde{\mathcal{G}}_n$ can be found using the Erdös-Rényi model, with high probability. This probabilistic statement is not to be confused with Eq. (14), which is a *deterministic* property that holds for all $g \in \tilde{\mathcal{G}}_n$. For the latter property, the randomness lies only in the sampling of subgraphs (via $\mathbf{P}_{\mathcal{M}_{m(n),n}}$). This distinction is important for our analysis, because the interconnection topology, g_n, stays fixed over time, while the random subgraph, $g_n|_{\Gamma\cup\Delta}$, is drawn independently for different batches.

Before proving Lemma 5, we state the following useful fact. The proof consists of a simple argument using Hall's marriage theorem and a union bound (cf. Lemma 2.1 in [23] for a proof of the lemma).

LEMMA 6. *Let G be an (n,n,p) random bipartite graph. Then*

$$\mathbb{P}_{n,p}(G \text{ does not admit a full matching}) \le 3n\,(1-p)^n. \quad (16)$$

PROOF. **(Lemma 5)** Let H be a random element of $\mathcal{M}_{m(n),n}$, distributed according to $\mathbf{P}_{\mathcal{M}_{m(n),n}}(h)$. Let G be an $(rn, n, p(n))$ random bipartite graph over $I \cup J$, generated independently of H. Note that the distribution of G restricted to any $m(n)$-by-$m(n)$ subset of $I \cup J$ is that of an $(m(n), m(n), p(n))$ random bipartite graph. Therefore, by Lemma 6, we have

$$\begin{aligned}\mathbb{E}(l(G)) &= \mathbb{P}(G|_H \text{ does not admit a full matching})\\ &\le 3m(n)\,(1-p(n))^{m(n)}, \quad (17)\end{aligned}$$

where $\mathbb{P}(\cdot)$ represents the distributions of both G and H. Eq. (17), combined with Markov's inequality, yields

$$\begin{aligned}\mathbb{P}_{n,p(n)}(\tilde{\mathcal{G}}_n) &= 1 - \mathbb{P}\left(l(G) > m^2(n)\,(1-p(n))^{m(n)}\right)\\ &\ge 1 - \frac{\mathbb{E}(l(G))}{m^2(n)\,(1-p(n))^{m(n)}} \ge 1 - \frac{3}{m(n)}, \quad (18)\end{aligned}$$

which converges to 1 as $n \to \infty$, because $m(n) \to \infty$. $\square$

Using Lemma 5, we can now obtain an upper bound on the value of $q(g_n)$. The proof is given in [26].

LEMMA 7. *Let $b(n) = Kn/y(n)$, as in Eq. (10), for some constant $K > 8/\lambda$. If $g_n \in \tilde{\mathcal{G}}_n$, then*

$$q(g_n) \le \frac{1}{y^2(n)} \cdot n^{-(\lambda K - 4)/2}.$$

We are now ready to bound the mean and variance of the service time distribution for the Matching queue, using Eq. (12) and Lemmas 3, 4, and 7. The proof is given in [26].

LEMMA 8. **(Service Time Statistics for the Matching Queue)** *Assume that $g_n \in \tilde{\mathcal{G}}_n$, and let $b(n) = Kn/y(n)$, for some constant $K > 8/\lambda$. The service times of the Matching batches, $S^M(k)$, are i.i.d., with*

$$\mathbb{E}(S^M(k)) \sim \frac{\lambda}{\rho+\epsilon} \cdot \frac{1}{y(n)}, \text{ and } \mathrm{Var}(S^M(k)) \lesssim \frac{1}{y^2(n)}.$$

Let W_M be the steady-state **waiting time of a batch in the Matching queue**, where waiting time is defined to be the time from when the batch is formed until when all jobs in the batch have started to receive service from a physical server. The following is the main result of this subsection.

PROPOSITION 1. **(Delays in the Matching Queue)** *Assume $g_n \in \hat{\mathcal{G}}_n \cap \tilde{G}_n$, where $\hat{\mathcal{G}}_n$ and $\tilde{G}_n$ were defined before Lemmas 1 and 5, respectively. We have*

$$\mathbb{E}(W_M) \lesssim \frac{1}{y(n)}. \tag{19}$$

PROOF. We use Kingman's bound, which states that the expected value of the waiting time in a $GI/GI/1$ queue, W, is bounded by $\mathbb{E}(W) \le \tilde{\lambda}\frac{\sigma_a^2+\sigma_s^2}{2(1-\tilde{\rho})}$, where $\tilde{\lambda}$ is the arrival rate, $\tilde{\rho}$ is the traffic intensity, and σ_a^2 and σ_s^2 are the variances for the interarrival times and service times, respectively. Using Lemmas 2 and 8, the claim follows by substituting

$$\begin{aligned}
\tilde{\lambda} &= \frac{1}{\mathbb{E}(A(k))} \lesssim \frac{n}{b(n)} = \frac{y(n)}{K},\\
\tilde{\rho} &= \frac{\mathbb{E}\left(S^M(k)\right)}{\mathbb{E}(A(k))} \le \frac{\frac{\lambda}{\rho+\epsilon}\frac{K}{y(n)}}{\frac{K}{ry(n)}} = \frac{\rho}{\rho+\epsilon} < 1,\\
\sigma_a^2 &= \mathrm{Var}(A(k)) \sim \frac{1}{\lambda}\frac{b(n)}{n^2} \lesssim \frac{1}{ny(n)},\\
\sigma_s^2 &= \mathrm{Var}\left(S^M(k)\right) \lesssim \frac{1}{y(n)^2},
\end{aligned} \tag{20}$$

for all sufficiently large values of n. We obtain

$$\mathbb{E}(W_M) \lesssim y(n)\left(\frac{1}{ny(n)} + \frac{1}{y(n)^2}\right) \lesssim \frac{1}{y(n)}.$$

□

5.2 Delays in a Residual Queue

The delay analysis for the Residual queues is conceptually simpler compared to that of the Matching queue, but requires more care on the technical end. The main difficulty comes from the fact that, due to the physical servers' interactions with the Matching queue, the service times in a Residual queue i, $\left\{S_i^R(k)\right\}_{k\ge0}$, are neither identically distributed nor independent over k (Section 4.2.1). To overcome this difficulty, we will use a trick to restore the i.i.d. nature of the service times. In particular, we will create a per-sample-path coupling of the $S_i^R(k)$'s to an i.i.d. sequence $\left\{\tilde{S}_i^R(k)\right\}_{k\ge0}$, such that $S_i^R(k) \le \tilde{S}_i^R(k)$ for all k, almost surely. If we pretend that the $\tilde{S}_i^R(k)$'s are indeed the true service times, an upper bound on the expected waiting time can be established using Kingman's bound. We finish the argument by showing that this upper bound applies to the expected value of the original waiting time. The following proposition is the major technical result of this section.

PROPOSITION 2. *Assume $g_n \in \hat{\mathcal{G}}_n$, where $\hat{\mathcal{G}}_n$ was defined in Lemma 1. Fix any $i \in I$. Let k be the index for batches. Denote by $\left\{T_i^R(k)\right\}_{k\ge0}$ and $\left\{S_i^R(k)\right\}_{k\ge0}$ the sequences of interarrival and service times at the ith Residual queue, respectively, defined on a common probability space. There exists a sequence $\left\{\tilde{S}_i^R(k)\right\}_{k\ge0}$, defined on the same probability space, that satisfies:*
1. $\mathbb{P}\left(\tilde{S}_i^R(k) \ge S_i^R(k),\ \forall k \in N\right) = 1$.
2. Elements of $\left\{\tilde{S}_i^R(k)\right\}$ are i.i.d. over k, and independent of $\left\{T_i^R(k)\right\}$.
3. $\mathbb{E}\left(\tilde{S}_i^R(1)\right) \lesssim \frac{b^2(n)}{n^2}$, and $\mathbb{E}\left(\tilde{S}_i^R(1)^2\right) \lesssim \frac{b^2(n)}{n^2}$.

PROOF. The proof is given in [26]. It involves an explicit coupling construction of the sequence $\left\{\tilde{S}_i^R(k)\right\}$, which is then shown to satisfy all three claims. Technicalities aside, the proof makes use of the following simple observations: **(1)** the event periods can be "prolonged" via coupling to be made independent, without qualitatively changing the scaling of the service time, $\tilde{S}_i^R(k)$, and **(2)** the first and second moments of $S_i^R(k)$ are mainly influenced by the size of the Residual batch, $R(k)$, which is small ($\lesssim \frac{b(n)}{n}$) by design. □

Denote by W_i^R the steady-state **waiting times in the ith Residual queue**, where waiting time is defined to be the time from when the batch is formed till when all jobs in the batch have started to receive service. The next proposition is the main result of this subsection, which is proved using the stochastic dominance result of Proposition 2, and the same Kingman's bound as in Proposition 1. The proof is given in [26].

PROPOSITION 3. **(Delay in a Residual Queue)** *Assume $g_n \in \hat{\mathcal{G}}_n \cap \tilde{G}_n$, where $\hat{\mathcal{G}}_n$ and $\tilde{G}_n$ were defined before Lemmas 1 and 5, respectively. We have*[11]

$$\max_{i\in I}\mathbb{E}\left(W_i^R\right) \lesssim \frac{1}{y(n)}. \tag{21}$$

5.3 Proof of Theorem 1 Under Uniform Arrival Rates

PROOF. Let $\mathcal{H}_n' = \hat{\mathcal{G}}_n \cap \tilde{\mathcal{G}}_n$ for all $n \ge 0$, where $\hat{\mathcal{G}}_n$ and $\tilde{G}_n$ were defined before Lemmas 1 and 5, respectively. Since each job is served either through the Matching queue or a Residual queue, the total queueing delay is no more than that of the waiting time in a virtual queue plus the time to form a batch ($A(k)$). Hence, letting $b(n) = Kn/y(n)$, by Lemma 2, and Propositions 1 and 3, we have

$$\begin{aligned}
\mathbb{E}_\pi(W|g_n,\boldsymbol{\lambda}_n) &\le \mathbb{E}(A(1)) + \max\left\{\mathbb{E}\left(W^M\right),\mathbb{E}\left(W_1^R\right)\right\}\\
&\le \frac{K'}{y(n)} = K'\cdot\frac{\ln n}{d(n)},
\end{aligned} \tag{22}$$

if $g_n \in \mathcal{H}_n'$, where K' is a constant independent of n and g_n. It is not difficult to show, by the weak law of large numbers, that there exist $\epsilon_n \downarrow 0$, such that

$$\mathbb{P}\left(1-\epsilon_n \le \frac{\deg(G_n)}{d(n)} \le 1+\epsilon_n\right) = 1, \tag{23}$$

for all n, where G_n is a $\left(rn, n, \frac{d(n)}{n}\right)$ random graph. Let $\mathcal{L}_n = \left\{g_n \in \mathcal{G}_n : \frac{\deg(g_n)}{d(n)} \in [1-\epsilon_n, 1+\epsilon_n]\right\}$, and $\mathcal{H}_n = \mathcal{H}_n' \cap \mathcal{L}_n$. By Eq. (23), and Lemmas 1 and 7, we have

$$\mathbb{P}_{n,\frac{d(n)}{n}}(\mathcal{H}_n) \ge 1-\delta_n, \tag{24}$$

for all n, for some $\delta_n \downarrow 0$. Note that the definitions of $\hat{\mathcal{G}}_n$, $\tilde{\mathcal{G}}_n$ and $\mathcal{L}_n$ do not depend on the arrival rates $\boldsymbol{\lambda}_n$, and hence the value of δ_n also does not depend on $\boldsymbol{\lambda}_n$.

Eqs. (22) and (24) combined prove the first two claims of the theorem. Since all λ_i are equal in this case, the scheduling policy is oblivious to the arrival rates by definition. This completes the proof of Theorem 1 when $\lambda_{n,i} = \lambda < 1/r$ for all n and i. □

[11]The expectation is defined in the sense of Eq. (2).

6. GENERAL CASE AND ARRIVAL-RATE OBLIVIOUS SCHEDULING

We now complete the proof of Theorem 1 for the general case, for $\boldsymbol{\lambda}_n$ satisfying Condition 1. In particular, we will show that the original virtual-queue-based scheduling policy given in Section 4.2 automatically achieves the $u^2(n)\ln n/d(n)$ delay scaling, while being fully oblivious to the values of the $\lambda_{n,i}$. We will use a definition of $\mathcal{H}_n$ identical to that in the uniform case, so that the probability of this set will still converge to 1. To avoid replicating our earlier arguments, and since the delays in each of the virtual queues are completely characterized by the statistics of arrival times and job sizes, we will examine the changes in each of these queueing primitives as we move to the general case. Throughout the section, we will analyze a system where the arrival rate vector, $\boldsymbol{\lambda}_n$, satisfies Condition 1, and the scheduling rules are the same as before, with $b(n) = Kn/y(n)$.

1. Inter-arrival times of batches. The inter-arrival times to all virtual queues are equal to the time it takes to collect enough jobs to form a single batch. By the merging property of Poisson processes, the aggregate stream of arrivals to the system is a Poisson process of rate $\sum_{i\in I}\lambda_{n,i}$, with

$$\sum_{i\in I}\lambda_{n,i} \le \rho n. \tag{25}$$

We will make a further assumption, that there exists $\tilde{\rho}$, with $0 < \tilde{\rho} < \rho$, such that

$$\sum_{i\in I}\lambda_{n,i} \ge \tilde{\rho} n. \tag{26}$$

This assumption will be removed by the end of this section by a small modification to the scheduling policy. Given Eqs. (25) and (26), a result analogous to Lemma 2 holds:

$$\mathbb{E}\left(A(k)\right) = \frac{b(n)}{n}, \quad \text{and} \quad \mathrm{Var}\left(A(k)\right) \lesssim \frac{b(n)}{n^2}. \tag{27}$$

2. Service times for Residual queues. The distributions of the times at which a server tries to fetch a job from a Residual queue can vary due to the different values of the $\lambda_{n,i}$'s. However, it is not difficult to show that the *upper bound* on the service times at the Residual queues (Proposition 2) depend only on the batch size $\frac{\rho}{r}b(n)$, and are independent of the specific values of the $\lambda_{n,i}$'s. Therefore, the only factor that may significantly change the delay distribution at the ith Residual queue is its associated Residual batch size, $R_i(k)$ (Eq. (7)). Since the arrival rates are no longer uniform, this distribution will in principle be different from the uniform setting. However, we will show that the difference is small and does not change the scaling of delay.

Denote by p_i the probability that an incoming job to the system happens to arrive at queue i. We have that

$$p_i = \frac{\lambda_{n,i}}{\sum_{i\in I}\lambda_{n,i}} \le \frac{u(n)}{\tilde{\rho} n} = \frac{1}{\frac{\tilde{\rho}}{u(n)}n}, \tag{28}$$

where $u(n)$ is the upper bound on $\lambda_{n,i}$ defined in Condition 1, and the inequality follows from the assumptions that $\lambda_{n,i} \le u(n)$ and that $\sum_{i\in I}\lambda_{n,i} \ge \tilde{\rho}n$ (Eq. (26)). Using Eq. (28), and following the same steps as in the proof of Proposition 2, one can show that

$$\max_{i\in I}\mathbb{E}\left(\tilde{S}_i^R(k)\right) \lesssim \frac{b^2(n)u^2(n)}{n^2},$$

and

$$\max_{i\in I}\mathrm{Var}\left(\tilde{S}_i^R(k)\right) \lesssim \frac{b^2(n)u^2(n)}{n^2}.$$

We repeat the arguments in Proposition 3 by replacing, e.g., the bounds on $\mathrm{Var}\left(\tilde{S}_i^R(k)\right)$ with $\max_{i\in I}\mathrm{Var}\left(\tilde{S}_i^R(k)\right) \lesssim \frac{b^2(n)u^2(n)}{n^2}$.

Letting $b(n) = K\frac{n\ln n}{d(n)}$, we have that, whenever $u(n) \ll \sqrt{\frac{d(n)}{\ln n}}$,

$$\max_{i\in I}\mathbb{E}\left(W_i^R\right) \lesssim \frac{n}{b(n)}\left(\frac{b(n)}{n^2} + \frac{b^2(n)u^2(n)}{n^2}\right) \lesssim \frac{b(n)u^2(n)}{n}$$
$$\lesssim \frac{\ln n}{d(n)}u^2(n). \tag{29}$$

3. Service times for the Matching queue. Surprisingly, it turns out that the bounds on the service times statistics for the Matching queue in the uniform arrival-rate case (Lemma 8) will carry over to the general setting, unchanged. Recall from Eq. (12) that the service time of a Matching batch is composed of two elements: the length of a COLLECT phase, S_{col}, and, with probability $q(g_n)$, the length of a clearing phase S_{cle}. Since the size of the Matching batch is at most $\frac{\rho}{r}b(n)$ by definition, the characterizations of S_{col} and S_{cle} given by Lemmas 3 and 4 continue to hold, with λ being replaced by $\lambda = \frac{\rho}{r}$. It remains to verify that the bound on $q(g_n)$ stays unchanged. While the $\lambda_{n,i}$'s are now different and the subgraph induced by Δ and Γ is no longer uniformly distributed, the upper bound on $q(g_n)$ given in Lemma 7 still holds, because Lemma 5 applies to arbitrary distributions of subgraphs. Therefore, the scaling in Proposition 1

$$\mathbb{E}\left(W^M\right) \lesssim \frac{1}{y(n)} = \frac{\ln n}{d(n)}, \tag{30}$$

continues to hold under non-uniform arrival rates.

Combining Eqs. (25), (29) and (30), we have, analogous to Eq. (22), that for all $n \ge 1$,

$$\mathbb{E}_\pi\left(W|g_n,\boldsymbol{\lambda}_n\right) \le \mathbb{E}\left(A(1)\right) + \max\left\{\mathbb{E}\left(W^M\right),\mathbb{E}\left(W_1^R\right)\right\}$$
$$\le \frac{K_1\ln n}{d(n)} + \frac{K_2u^2(n)\ln n}{d(n)} \le \frac{Ku^2(n)\ln n}{d(n)}, \tag{31}$$

if $g_n \in \mathcal{H}_n$, where K_1, K_2 and K are positive constants independent of n, g_n and $\boldsymbol{\lambda}_n$.

Finally, we justify the assumption made in Eq. (26), by showing that the scheduling policy could be easily modified such that Eq. (26) always holds: we simply insert to each queue an independent Poisson stream of "dummy packets" at rate δ, where δ is chosen to be any constant in $(0, \frac{1-\rho}{r})$. The dummy packets are merely place-holders: when a server begins serving a dummy packet, it simply goes on a vacation of duration $\mathrm{Expo}(1)$. This trivially guarantees the validity of Eq. (26), with $\tilde{\lambda} = \delta$, and since $\sum_{i\in I}\lambda_{n,i} \le \rho n$, doing so is equivalent to having a new system with ρ replaced by $\rho' = \rho + \delta r < 1$. Hence, all results continue to hold.[12] This concludes the proof of Theorem 1.

7. CONCLUSIONS AND FUTURE WORK

The main message of this paper is that the benefits of diminishing delay and large capacity region can be jointly achieved in a system where the level of processing flexibility of each server is small compared to the system size. The main result, Theorem 1, proves this using a randomly constructed interconnection topology and a virtual-queue-based scheduling policy. As a by-product, it also provides an explicit upper bound (Eq. (3)) that captures a trade-off between the delay ($\mathbb{E}(W)$), level of robustness ($u(n)$), and degree of processing flexibility ($d(n)$).

[12]This maneuver of inserting dummy packets is of course a mere technical device for the proof. In reality, delay and capacity would both become much less of a concern when the traffic intensity, ρ, is too small, at which point less sophisticated scheduling policies may suffice.

The scaling regime considered in this paper assumes that the traffic intensity is fixed as n increases, which fails to capture system performance in the heavy-traffic regime ($\rho \approx 1$). It would be interesting to consider a scaling regime in which ρ and n scale simultaneously (e.g., the celebrated Halfin-Whitt regime [25]), but it is unclear at this stage what exact formulations and analytical techniques are appropriate. At a higher level, it would be interesting to understand how long-term input characteristics in a queueing network (e.g., how arrival rates are drawn or evolve over time) impact its temporal performance (e.g., delay), and what role the network's intrinsic structure (e.g., flexibility) has to play here.

Theorem 1 also leaves open several promising directions for future improvements and extensions, which we discuss briefly. It is currently necessary to have that $d(n) \gg \ln n$ in order to achieve a diminishing delay. The $\ln n$ factor is essentially tight if a random-graph-based connectivity structure is to be used, because $d(n) = \Omega(\ln n)$ is necessary for an Erdös-Rényi type random graph to be connected (i.e., not have isolated queues). We suspect that the $\ln n$ factor can be reduced using a different graph generation procedure.

Fixing $u(n)$ to a constant, we observe that when $d(n) = n$ (fully connected graph), the system behaves approximately as a $M/M/n$ queue with total arrival rate $n\rho$; if ρ is held fixed, then the expected delay is known to vanish exponentially fast in n. Thus, there is a gap between this fact and the polynomial scaling of $\mathcal{O}(\ln n/d(n))$ given in Theorem 1. We believe that this gap is due to an intrinsic inefficiency of the batching procedure used in our scheduling policy. The batching procedure provides analytical convenience by allowing us to leverage the regularity in the arrival traffic in obtaining delay upper bounds. However, the use of batching also mandates that all jobs wait for a prescribed period of time before receiving any service. This can be highly inefficient in the scaling regime that we consider, where the traffic intensity is fixed at ρ, because a $1-\rho$ fraction of all servers (approximately) are actually idle at any moment in time. We therefore conjecture that a scheduling policy that tries to direct a job to an available server immediately upon its arrival (such as a greedy policy) can achieve a significantly better delay scaling, but the induced queueing dynamics, and the appropriate analytical techniques, may be very different from those presented in this paper.

Finally, the parameter $u(n)$ captures the level of robustness against uncertainties in the arrival rates. In terms of inducing a diminishing waiting time as $n \to \infty$, there is still a gap between the current requirement that $u(n) \ll \sqrt{d(n)/\ln n}$ and the upper bound that $u(n) = \mathcal{O}(d(n))$ (since each queue is connected to only $d(n)$ servers on average). The $u^2(n)$ factor in the scaling of $\mathbb{E}(W)$ is due to the impact on delay by the arrival rate fluctuations in the Residual queues (cf. Eq. (29)). It is unclear what the optimal dependence of $\mathbb{E}(W)$ on $u(n)$ is. It is also possible that the condition $u(n) \ll d(n)$ alone is sufficient for achieving a diminishing delay; we conjecture that this is the case.

8. REFERENCES

[1] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, R. Chaiken, "Nature of datacenter traffic: measurements and analysis," *IMC*, 2009.

[2] G. Soundararajan, C. Amza, and A. Goel, "Database replication policies for dynamic content applications," *Proc. of EuroSys*, 2006.

[3] W. Jordan and S. C. Graves, "Principles on the benefits of manufacturing process flexibility," *Management Science*, 41(4):577–594, 1995.

[4] S. Gurumurthi and S. Benjaafar, "Modeling and analysis of flexible queueing systems," *Management Science*, 49:289–328, 2003.

[5] S. M. Iravani, M. P. Van Oyen, and K. T. Sims, "Structural flexibility: A new perspective on the design of manufacturing and service operations," *Management Science*, 51(2):151–166, 2005.

[6] M. Chou, G. A. Chua, C-P. Teo, and H. Zheng, "Design for process flexibility: efficiency of the long chain and sparse structure," *Operations Research*, 58(1):43–58, 2010.

[7] D. Simchi-Levi and Y. Wei, "Understanding the performance of the long chain and sparse designs in process flexibility," submitted, 2011.

[8] M. Chou, C-P. Teo, and H. Zheng, "Process flexibility revisited: the graph expander and its applications," *Operations Research*, 59:1090–1105, 2011.

[9] M. Leconte, M. Lelarge and L. Massoulie, "Bipartite Graph Structures for Efficient Balancing of Heterogeneous Loads," *ACM Sigmetrics*, London, 2012.

[10] R. Talreja, W. Whitt, "Fluid models for overloaded multiclass many-service queueing systems with FCFS routing," *Management Sci.*, 54(8):1513-1527,2008.

[11] J. Visschers, I. Adan, and G. Weiss, "A product form solution to a system with multi-type jobs and multi-type servers," *Queueing Systems*, 70:269-298, 2012.

[12] A. Mandelbaum and M. I. Reiman, "On pooling in queueing networks," *Management Science*, 44(7):971-981, 1998.

[13] J. M. Harrison and M. J. Lopez, "Heavy traffic resource pooling in parallel-server systems," *Queueing Systems*, 33:39-368, 1999.

[14] S. L. Bell and R. J. Williams, "Dynamic scheduling of a system with two parallel servers in heavy traffic with resource pooling: asymptotic optimality of a threshold policy," *Ann. Appl. Probab.*, 11(3): 608-649, 2001.

[15] R. Wallace and W. Whitt, "A staffing algorithm for call centers with skill-based routing," *Manufacturing and Service Operations Management*, 7:276–294, 2005.

[16] A. Bassamboo, R. S. Randhawa, and J. A. Van Mieghem, "A little flexibility is all you need: on the asymptotic value of flexible capacity in parallel queuing systems," submitted, 2011.

[17] J. N. Tsitsiklis and K. Xu, "On the power of (even a little) resource pooling," *Stochastic Systems*, 2: 1–66 (electronic), 2012.

[18] M. Neely, E. Modiano and Y. Cheng, "Logarithmic delay for n×n packet switches under the crossbar constraint," *IEEE/ACM Trans. Netw.*, 15(3):657–668, 2007.

[19] J. N. Tsitsiklis and D. Shah, "Bin packing with queues," *J. Appl. Prob.*, 45(4):922–939, 2008.

[20] N. McKeown, A. Mekkittikul, V. Anantharam and J. Walrand, "Achieving 100% throughput in an input-queued switch," *IEEE Trans. on Comm.*, 47(8):1260–1267, 1999.

[21] S. Kunniyur and R. Srikant, "Analysis and design of an adaptive virtual queue," *ACM SIGCOMM*, 2001.

[22] P. Erdös and A. Rényi, "On random matrices," *Magyar Tud. Akad. Mat. Kutato Int. Kozl*, 8:455–461, 1964.

[23] S. R. Bodas, "High-performance scheduling algorithms for wireless networks," Ph.D. dissertation, University of Texas at Austin, Dec. 2010.

[24] V. F. Kolchin, B. A. Sevast'yanov, and V. P. Chistyakov, *Random Allocation*, John Wiley & Sons, 1978.

[25] S. Halfin and W. Whitt, "Heavy-traffic limits for queues with many exponential servers," *Operations Research*, 29: 567-588, 1981.

[26] J. N. Tsitsiklis and K. Xu, "Queueing system topologies with limited flexibility," *Technical Report*, `http://web.mit.edu/jnt/www/Papers/c-13-SigRep.pdf`.