Performance Evaluation and Asymptotics for Content Delivery Networks

Virag Shah and Gustavo de Veciana
ECE Dept., The University of Texas at Austin

***Abstract*—Large scale Content Delivery Networks (CDNs) are one of the key components of today's information infrastructure. This paper proposes and analyzes a simple stochastic model for a file-server system wherein servers can work together, as a pooled resource, to meet individual user requests. In such systems basic questions include: How and where to replicate files? What is the impact of dynamic service allocation across request types, and whether it can provide substantial gains over simpler load balancing policies? What are tradeoffs amongst performance, reliability and recovery costs, and energy? The paper provides both explicit and asymptotic approximations for large systems towards addressing these basic questions.**

I. Introduction

We are in the midst of a paradigm shift where the data from end users and content providers including videos, web content and files are stored in virtualized storage pools generally hosted by third parties operating large data centers. Although this type of infrastructure is prevalent, the design-space tradeoffs of such massive storage and delivery systems are not well understood. Some metrics of interest for such systems include: 1) service capacity available to end users and the resulting perceived performance; 2) reliability and recovery costs; and, 3) energy costs. Amongst these metrics there are various tradeoffs that can be realized. For example, by increasing the total number of active servers, or scaling the speed of individual servers, one can tradeoff energy cost with performance. A more subtle example, discussed further in the sequel, involves quantifying the trade off between spreading multiple copies of files across pools of servers so as to improve performance and cost in recovery from large-scale server loss events, e.g., power outages [5]. The focus of this paper lies in developing robust large scale models enabling a quantitative study of such tradeoffs.

Our contributions. The key challenge we tackle is the performance evaluation of large scale storage systems wherein multiple file copies are placed across pools of servers subject to stochastic loads. We consider a system model where arriving file requests/jobs/flows can be collectively served by servers, i.e., different chunks of each file can be concurrently downloaded from servers currently storing the file – this is akin to the service model in peer-to-peer systems. Since each server can store multiple files, which are themselves replicated across sets of servers, the service capacity available to serve requests for different files are dynamically coupled. Indeed, as explained in the sequel, ongoing file requests can share server capacity subject to various possible 'fairness' objectives rendering performance evaluation quite challenging.

978-1-4799-3360-0/14/$31.00

The main contributions of this paper can be summarized as follows. First, we propose a file-server model and show that the overall service capacity set has polymatroid structure. This in turn allows us to develop an explicit expression for the mean file transfer delay experienced by file requests. Second, we prove an asymptotic result for *symmetric* large-scale systems wherein the distribution concentrates at a mode. This result provides an easily computable approximation for the mean delay which is used to quantify system tradeoffs. Finally, these analytical results are used to develop and quantify three key insights regarding large file-server systems: *a)* We show how dynamic service capacity allocation across ongoing demands is impacted by the structure of overlapping resource pools (file placement) and quantify the substantial performance benefits over simpler load balancing strategies that assign file requests at random or to least loaded servers. *b)* We show that performance gains resulting from pooling of servers, although significant, quickly saturate as we increase the pool size. Thus one can engineer such systems so as to realize close to optimal performance while simultaneously achieving high reliability and thus low recovery costs. *c)* For a simple speed scaling policy where the processor runs at low speed (or halts) when idle and a high but fixed speed when busy, we show that dynamic service capacity allocation can achieve up to 70% energy saving as compared to simpler policies.

Related work. There are several works in the literature studying energy-performance tradeoffs, see eg [7], [12] and citations therein. In [7], the authors provide an approximation to the number of servers that should be active so as to optimize the energy-delay product. Similarly, [21] investigates speed scaling so as to optimize a weighted average of energy and mean delay for a single server system. In [12], the authors consider energy costs of switching servers on and off and provide an optimal online algorithm to optimize overall convex cost functions that can include performance and energy costs. In these works a server can handle any job request. By contrast in this paper we are particularly interested in the situations wherein pools of servers' capabilities are constrained (e.g., by the files they have available) and the coupling amongst these critically impacts energy-performance tradeoffs.

There has also been previous work considering file placement across servers. For example, [11] studies the file placement across servers so as to minimize bandwidth inefficiency when there are a fixed set of file requests. Another line of

work has focused on online packing/placement of dynamically arriving files/objects under constraints on available resources, e.g., [18]. By contrast with these works, we assume file placements across servers are fixed and we examine the performance impact of this when the system is subject to stochastic loads.

In the asymptotic large system regime we consider, our model is related to the super-market queueing model studied in [4], [20] where each arriving request is assigned to the least loaded of $d \geq 2$ randomly chosen servers. The key difference between the super-market model and our model is that, rather than assigning a file request to a single server, we allow a file request to be served by multiple servers simultaneously in a fashion similar to multipath routing architectures studied in [8]–[10]. Studies of the benefits of doing so have been previously carried out, e.g., [10], and show the benefit of coordinating rate over multiple paths in terms of the worst case rate achieved by the users in a static setting. In this paper we are also interested in quantifying the benefits of pooling service capacity from multiple servers but for a system subject to stochastic loads.

As will be discussed in more detail below this paper draws on, and extends, previous work on bandwidth sharing models; in particular "balanced fair" allocations, see e.g., [1]–[3]. Such allocations are a useful device in that they are amenable to analysis, are provably insensitive to job size distribution, and yet serve to approximate various forms of 'fair' resource sharing policies considered in the literature and in practice [1], [13].

Organization of the paper. In Section II we develop our system model for file server systems under stochastic loads. In Section III we discuss fairness based resource allocation and provide an exact analysis for mean delay in file transfers under balanced fair service allocation. Section IV provides an asymptotic expression for mean delay which we then use to compare the performance of our policy with other allocation policies. Section V includes a discussion of system tradeoffs involving mean delay, recovery costs and energy consumption. We conclude in Section VI.

II. System Model: file-server system, dynamics, and service capacity

Let F denote a set of files and S a set of servers in a file-server system where $|F| = n$ and $|S| = m$. For each file $i \in F$ let $S_i \subset S$ denote the set of servers that store, and thus can serve file i; thus $\mathcal{S} = (S_i : i \in F)$ captures a file replication policy. Suppose each server $s \in S$ has fixed service capacity of μ_s bits per second. For each $A \subset F$ let $S(A) \triangleq \cup_{i \in A} S_i$ and $\mu(A) \triangleq \sum_{s \in S(A)} \mu_s$ denote the set of servers capable of serving one or more of the files in A and the associated service capacity. In summary, $\left(F, S, \mu; \mathcal{S}\right)$ collectively define a *file-server system*.

Requests for file $i \in F$ arrive according to an independent Poisson process with rate λ_i. We shall use terms request, flow and job interchangeably. Similarly, we refer to each file $i \in F$ as a file or a class interchangeably. Each request has a service requirement corresponding to, for example, the number of bits it needs to download from the file-server system. Service requirements for requests for file $i \in F$ are i.i.d with mean ν_i bits. This can model, for example, requests for a part of a file. Let $\boldsymbol{\rho} = (\rho_i : i \in F)$, where $\rho_i = \lambda_i \nu_i$ denotes the load associated with class i.

Flows arrive to the system at total rate $\sum_{i \in F} \lambda_i$. Let u_k denote the flow corresponding to the k^{th} arrival after time $t = 0$. Let $q_i(t)$ denote the *set* of ongoing flows of class i at time t, i.e., flows which have arrived but have not completed service, and $\mathbf{q}(t) = (q_i(t) : i \in F)$. For each $A \subset F$, let $q_A(t) = \cup_{i \in A} q_i(t)$, i.e., the set of all active flows whose class is in A. Let $\mathbf{x}(t) = (x_i(t) : i \in F)$, where $x_i(t) \triangleq |q_i(t)|$, i.e., $\mathbf{x}(t)$ captures the *number* of ongoing flows in each class. We refer to $\mathbf{x}(t)$ as the state of the system at time t.

For any $v \in q_i(t)$, let $b_v(t)$ be the rate in bits per second at which flow v is served at time t by the file-server system. At any time t, we assume that the rates $b_v(t)$ for all $v \in q_F(t)$ depend only on $\mathbf{x}(t)$ and the classes to which they belong. Thus for any $i \in F$ and $u, v \in q_i(t)$ we have $b_u(t) = b_v(t)$. Further, let $r_i(\mathbf{x}')$ be the total rate at which class i flows are served at time t when $\mathbf{x}(t) = \mathbf{x}'$, i.e., at any time t, $r_i(\mathbf{x}(t)) = \sum_{v \in q_i(t)} b_v(t)$. Let $\mathbf{r}(\mathbf{x}') = (r_i(\mathbf{x}') : i \in F)$. To visualize this system, think of the system as consisting of n queues, one corresponding to each file, with coupled service rates $\mathbf{r}(\mathbf{x}(t))$. Each queue in turn allocates its rate among its active users equally akin to processor sharing, i.e., $b_v(t) = r_i(\mathbf{x}(t))/x_i(t)$ for each $v \in q_i(t)$, if $x_i(t) \neq 0$. For any $\mathbf{x}(t)$, let $A_{\mathbf{x}(t)}$ denote the set of active classes, i.e., the classes with at least one ongoing flow. If flow v arrives at time t_v^a and has service requirement e_v, then it departs at time t_v^d such that $e_v = \int_{t_v^a}^{t_v^d} b_v(t) dt$.

We will consider the case where the system's service capacity can be shared dynamically among ongoing requests. In particular multiple servers can contribute towards servicing a given request, e.g., by retrieving distinct chunks of a file from different servers, and thus allowing pooling of capacity of multiple servers. Let $b_{v,s}(t)$ be the rate at which server s serves request v at time t. A request v for file i, i.e., $v \in q_i(t)$, can only be served by servers which have that file, thus $b_{v,s}(t) = 0$ if $s \notin S_i$, subject to the following assumption.

Assumption 1. *Sharing of system service capacity among ongoing flows is such that:*

1) *Each server can concurrently serve multiple requests as long as $\sum_v b_{v,s}(t) \leq \mu_s$ for all t.*
2) *Multiple servers can concurrently serve a request v at time t giving a total service rate $b_v(t) = \sum_s b_{v,s}(t)$.*
3) *The service rate $b_{v,s}(t)$ allocated to a flow v at server s at time t depends only on its flow's class and the numbers of ongoing flows $\mathbf{x}(t)$. Thus a flow's overall service rate $b_v(t)$ as well as the aggregate service rate allocated to flows in each class $\mathbf{r}(\mathbf{x}(t)) = (r_i(\mathbf{x}(t)) : i \in F)$ depend only on the number of ongoing flows.*

Note that service rate allocations depend only on the queue length $\mathbf{x}(t)$ and thus can not depend on the residual file

sizes of ongoing flows. The constraints $\sum_v b_{v,s}(t) \leq \mu_s$ can model servers' processing capacity or their network access bottleneck. For simplicity, we shall refer to these as server capacity constraints.

Under Assumption 1 we will show that the set of feasible service-rate allocations across classes, i.e., the *capacity region*, is a polymatroid. We say a polytope $\hat{\mathcal{C}}$ is a *polymatroid* if there exists a set function $\hat{\mu}$ on F such that

$$\hat{\mathcal{C}} = \left\{ \mathbf{r} \geq \mathbf{0} : \sum_{i \in A} r_i \leq \hat{\mu}(A),\ \forall A \subset F \right\},$$

and if $\hat{\mu}$ satisfies the following properties:
1) Normalized: $\hat{\mu}(\emptyset) = 0$.
2) Monotonic: if $A \subset B$, $\hat{\mu}(A) \leq \hat{\mu}(B)$.
3) Submodular: for all $A, B \subset F$,

$$\hat{\mu}(A) + \hat{\mu}(B) \geq \hat{\mu}(A \cup B) + \hat{\mu}(A \cap B).$$

A function $\hat{\mu}$ satisfying the above properties is called a *rank function*. Polymatroids and submodular functions are well studied in the literature, see e.g., [16]. Each polymatroid $\hat{\mathcal{C}}$ has a special property that for any $\mathbf{r} \in \hat{\mathcal{C}}$, there exists $\mathbf{r}' \geq \mathbf{r}$ such that $\mathbf{r}' \in \mathcal{D} \triangleq \{\mathbf{r} \in \hat{\mathcal{C}} : \sum_{i \in F} r_i = \mu(F)\}$ [6]. Also, as evident from the definition, for any $A \subset F$ the set $\{\mathbf{r} \in \hat{\mathcal{C}} : r_i = 0, \forall i \notin A\}$ is also a polymatroid, with a rank function which is the restriction of μ to subsets of A.

Theorem 1. *Consider a file-server system defined by* $\big(F, S, \mu; \mathcal{S}\big)$ *and let*

$$\mathcal{C} \triangleq \{\mathbf{r} \geq \mathbf{0} : \sum_{i \in A} r_i \leq \mu(A),\ \forall A \subset F\}.$$

Then, the following hold
1) μ is a rank function.
2) $\mathcal{C}$ is the polymatroid capacity region associated with the file server system.

A sketch of proof of above theorem is provided in the Appendix due to space constraints. For detailed proof, see [17]. We say that a polymatroid capacity region is *symmetric* if $\mu(A) = h(|A|)$ for any $A \subset F$ where $h : \mathbb{Z}_+ \to \mathbb{R}_+$ is a non-decreasing function, i.e., $\mu(A)$ depends on A only through $|A|$. Conversely, it is easy to show that if $\mu(A) = h(|A|)$ for some non-decreasing concave function $h : \mathbb{R}_+ \to \mathbb{R}_+$ with $h(0) = 0$, then the capacity region is a symmetric polymatroid.

III. Resource allocation and performance analysis

There are several ways in which the capacity of a file-server system can be shared among a set of ongoing flows leading to potentially different user performance. For example, one may assign a *fixed* service capacity to each file to be exclusively shared by ongoing requests for that file. While this simplifies analysis by decoupling the dynamics across files, it results in wasted resources and poor performance. A better approach is to *dynamically* share service capacity across flow classes based on their load, e.g., queue lengths capturing the number of active flows.

Given the queue length $\mathbf{x}$ of the system at time t, one can consider allocating service capacity in various ways. For example, *α-fair rate allocation*, introduced in [15], allocates capacity based on maximizing a concave sum utility function subject to the systems capacity region. In our setting we can consider α-fair service rate allocation to flows subject to the the capacity region $\mathcal{C}$ given in Theorem 1. Formally, for any $\mathbf{x}$, the rate vector $\mathbf{r}(\mathbf{x})$ under α-fair allocation is given by

$$\mathbf{r}(\mathbf{x}) = \begin{cases} \arg\max_{\hat{\mathbf{r}} \in \mathcal{C}} \sum_{i \in F} \frac{x_i^\alpha\, \hat{r}_i^{1-\alpha}}{1-\alpha} & \text{for } \alpha \in (0, \infty) \backslash \{1\}, \\ \arg\max_{\hat{\mathbf{r}} \in \mathcal{C}} \sum_{i \in F} x_i \log(\hat{r}_i) & \text{for } \alpha = 1. \end{cases} \tag{1}$$

Note this generalizes various notions of fairness, e.g., *max-min fair* (MMF) and *proportional fair* (PF) allocations. Indeed PF and MMF are equivalent to α-fair policy for $\alpha = 1$ and $\alpha \to \infty$, respectively [15]. However, Theorem 2 below shows that on polymatroid capacity regions such allocations are equivalent – we omit proof for lack of space.

Theorem 2. *All α-fair rate allocations are equivalent for polymatroid capacity regions.*

Note that while this is clear for a single server system where α-fair allocations reduce to equal share, it may be at first sight surprising in the multidimensional setting. Unfortunately, this does not characterize the performance users would see in a stochastic system and such results have been quite limited. What has been shown is that for such allocations, the performance is *sensitive* to the distribution of service requirements [2]. Thus, it is hard to make useful general claims.

By contrast, the *balanced fair* (BF) allocations introduced in [2] are 'insensitive', i.e., performance depends on the service distribution only through its mean. Moreover, BF has close structural relationship with proportional fairness, see e.g., [2], [13]. Additionally [1] studies several networks and shows a remarkable closeness in performance for balanced and proportional fairness motivating the use of BF as a mathematical tool for performance evaluation of stochastic networks under PF allocations.

Let us define BF rate allocation for our file-server system. Balanced fair rate allocation [2] for a polymatroid capacity region $\mathcal{C}$ can be defined as the service rate allocation $\mathbf{r}(\mathbf{x})$, where for any $\mathbf{x}$,

$$r_i(\mathbf{x}) = \frac{\Phi(\mathbf{x} - \mathbf{e}_i)}{\Phi(\mathbf{x})},\ \forall i \in F \tag{2}$$

where function Φ, called balance function, is defined recursively as follows: $\Phi(\mathbf{0}) = 1$ and $\Phi(\mathbf{x}) = 0$, $\forall \mathbf{x}$ s.t. $x_i < 0$ for some i, otherwise,

$$\Phi(\mathbf{x}) = \max_{A \subset F} \left\{ \frac{\sum_{i \in A} \Phi(\mathbf{x} - \mathbf{e}_i)}{\mu(A)} \right\}, \tag{3}$$

where $\mathbf{e}_i$ is a vector with 1 at i^{th} position and 0 elsewhere. As shown in [2], Eq. (2) ensures the important property of insensitivity, while (3) ensures that $\mathbf{r}(\mathbf{x})$ for each $\mathbf{x}$ lies in

the capacity region, i.e., the constraints $\sum_{i\in A} r_i(\mathbf{x}) \le \mu(A)$ are satisfied for each A. It also ensures that there exists a set $B \subset A_{\mathbf{x}}$ for which $\sum_{i\in B} r_i(\mathbf{x}) = \mu(B)$. In fact the BF allocation is the unique policy satisfying the above properties. As discussed in [1], [2], under BF allocation and a stability condition $\boldsymbol{\rho} \in$ Interior($\mathcal{C}$) the stationary distribution for the queue length process is given by

$$\pi(\mathbf{x}) = \frac{\Phi(\mathbf{x})}{G(\boldsymbol{\rho})} \prod_{i\in F} \rho^{x_i} \quad \text{where} \quad G(\boldsymbol{\rho}) = \sum_{\mathbf{x}'} \Phi(\mathbf{x}') \prod_{i\in F} \rho^{x'_i}.$$

An allocation of resources is said to be Pareto efficient if for any state $\mathbf{x}$, there does not exist an $\mathbf{r}' \in \mathcal{C}$ such that $r'_i \ge r_i(\mathbf{x})$, $\forall i \in A_{\mathbf{x}}$ with a strict inequality for at least one $i \in A_{\mathbf{x}}$. Pareto efficiency is a desirable property since it implies that the resource allocation is less wasteful. BF may not satisfy this property in general, e.g., see triangle networks studied in [2]. However, Theorem 3 below shows that BF is Pareto efficient when the capacity region is a polymatroid. For a polymatroid capacity $\mathcal{C}$, showing Pareto efficiency is equivalent to showing $\sum_{i\in A_{\mathbf{x}}} r_i(\mathbf{x}) = \mu(A_{\mathbf{x}})$. A proof of this theorem is provided in the Appendix.

Theorem 3. *For balanced fair rate allocations on polymatroid capacity regions we have $\sum_{i\in A_{\mathbf{x}}} r_i(\mathbf{x}) = \mu(A_{\mathbf{x}})$ for all $\mathbf{x}$.*

A similar result was proved in [3] for the special case of wireline networks with tree topology – indeed this is a special case of our result. This result serves as a basis to obtain an exact expression for the mean delay in our file-server system under BF rate allocation given by the following theorem. A proof is provided in the Appendix.

Theorem 4. *Consider a file-server system $(F, S, \mu; \mathcal{S})$ with load $\boldsymbol{\rho}$ and under balanced fair resource allocation. The mean delay for requests/flows of class i is given by*

$$E[D_i] = \frac{\nu_i \frac{\partial}{\partial \rho_i} G(\boldsymbol{\rho})}{G(\boldsymbol{\rho})} = \nu_i \frac{\partial}{\partial \rho_i} \log G(\boldsymbol{\rho}), \tag{4}$$

where $G(\boldsymbol{\rho})$ is given by,

$$G(\boldsymbol{\rho}) = \sum_{A\subset F} G_A(\boldsymbol{\rho}), \tag{5}$$

and where $G_\emptyset(\boldsymbol{\rho}) = 1$ and $G_A(\boldsymbol{\rho})$ can be computed recursively as

$$G_A(\boldsymbol{\rho}) = \frac{\sum_{i\in A} \rho_i G_{A\setminus\{i\}}(\boldsymbol{\rho})}{\mu(A) - \sum_{j\in A} \rho_j}. \tag{6}$$

Also, $\frac{\partial}{\partial \rho_i} G(\boldsymbol{\rho})$ can be recursively computed, without actually computing derivatives, as follows:

$$\frac{\partial}{\partial \rho_i} G(\boldsymbol{\rho}) = \sum_{A\subset F} \frac{\partial}{\partial \rho_i} G_A(\boldsymbol{\rho}), \tag{7}$$

where $\frac{\partial}{\partial \rho_i} G_\emptyset(\boldsymbol{\rho}) = 0$, and,

$$\frac{\partial}{\partial \rho_i} G_A(\boldsymbol{\rho}) = \frac{G_A(\boldsymbol{\rho}) + G_{A\setminus\{i\}}(\boldsymbol{\rho}) + \sum_{j\in A} \rho_j \frac{\partial}{\partial \rho_i} G_{A\setminus\{j\}}(\boldsymbol{\rho})}{\mu(A) - \sum_{j\in A} \rho_j}, \tag{8}$$

if $i \in A$ and 0 otherwise.

While the mean delay for systems with polymatroid capacity can be computed using (4) - (8), exact computation has a complexity which grows exponentially in the number of files n. If, however, the capacity region is given by a symmetric polymatroid and load vector $\boldsymbol{\rho}$ is homogenous, the complexity can be made linear in n. The following corollary, with proof sketched in the Appendix, details this result.

Corollary 1. *Consider a symmetric file-server system $(F, S, \mu; \mathcal{S})$ with homogenous load $\boldsymbol{\rho}$ and under balanced fair resource allocation, i.e., for each $A \subset F$, the rank function $\mu(A) = h(|A|)$ for some non-decreasing function $h : \mathbb{Z}_+ \to \mathbb{R}_+$ and for all $j \in F$ $\rho_j = \rho = \lambda\nu$. Then, the mean delay to serve the requests/flows of class i is given by,*

$$E[D_i] = \frac{\nu \hat{F}(\rho)}{F(\rho)}, \tag{9}$$

where, $F(\rho)$ and $\hat{F}(\rho)$ can be recursively obtained as follows:

$$F(\rho) = \sum_{k=0}^{n} F_k(\rho), \tag{10}$$

where, $F_0(\rho) = 1$, and for $k \ge 1$,

$$F_k(\rho) = \frac{(n-k+1)\rho F_{k-1}(\rho)}{h(k) - k\rho}. \tag{11}$$

Also,

$$\hat{F}(\rho) = \sum_{k=0}^{n} \frac{k}{n} \hat{F}_k(\rho), \tag{12}$$

where, $\hat{F}_0(\rho) = 0$, and for $k \ge 1$,

$$\hat{F}_k(\rho) = \frac{F_k(\rho) + \frac{n-k+1}{k} F_{k-1}(\rho) + \frac{(n-k+1)(k-1)}{k} \rho \hat{F}_{k-1}(\rho)}{h(k) - k\rho}. \tag{13}$$

IV. Performance evaluation and comparison

Using the results from previous section, we now develop an asymptotic result and compare the performance of various resource allocation policies.

A. Asymptotics for 'averaged' RPBF file-server systems

We consider asymptotics for file-server systems wherein first the number of files n, and then the number of servers m become large. This serves as an approximation for systems with a large number of servers, which serve orders of magnitude larger numbers of files, e.g., $m \sim 10^2$ or 10^3 while $n \sim 10^6$ or more. We assume the service rate per server is fixed to ξ and total request rate on the system is $m\lambda$, i.e., grows linearly with m, resulting in a traffic load $m\rho = m\lambda\nu$ where ν is the mean service requirement per request. Suppose that c copies of each of the n files are placed independently and uniformly across m servers at random without replacement. Let $\left(F^{(n)}, S^{(m)}, \mu^{(m,n)}; \mathcal{S}_c^{(m,n)}\right)$ represent a realization of such random file-server system. For each realization, the service

capacity is allocated dynamically according to balanced fair allocations over the associated capacity region, see Sec. III. We call such a file server systems as one with *Random Placement with BF resource allocation (RPBF)*.

For a given realization of the random file placement, the rank function $\mu^{(m,n)}$ need not be symmetric. Exact performance expressions for such a system would require computation of the associated capacity region and evaluating the recursions developed in Sec. III both of which have exponential complexity in n. However, a key insight, we develop below, is that large RPBF systems exhibit the same performance.

To that end consider the averaged RPBF system having the "average capacity region". Let $M^{(m,n)}(.)$ denote the random rank function associated with an (m,n) RPBF file placement. Given a set of files A where $|A| = k \leq n$ one can show that

$$\bar{\mu}^{(m,n)}(A) \triangleq E[M^{(m,n)}(A)] = \xi m(1-(1-c/m)^k).$$

Indeed the probability that none of the c copies of a file are stored on a given server is $(1-c/m)$. Thus the probability that none of A' s k files is stored at the server is $(1-c/m)^k$. So $m(1-(1-c/m)^k)$ is the mean number of servers that can serve *at least* one file in A, and the above is their associated service capacity. The averaged capacity region is thus given by a *symmetric* polymatroid with rank function $\bar{\mu}^{(m,n)}(A) = h^{(m,n)}(|A|)$ where

$$h^{(m,n)}(k) \triangleq \xi m(1-(1-c/m)^k) \text{ for } k = 0,1,\ldots,n.$$

Below we let $\pi^{(m,n)}(\mathbf{x})$ denote the stationary distribution of the queue length process for the average RPBF system, i.e., using balanced fair allocations over the average capacity region. Also, let $E[D^{(m,n)}]$ be expected delay for a typical request in this system. The following result gives a simple expression for the expected delay in the asymptotic regime of interest. Its proof is sketched in Appendix. For a detailed proof, see [17].

Theorem 5. *Consider a sequence of (m,n) averaged RPBF file-server systems with symmetric polymatroid capacity with the rank function $\bar{\mu}^{(m,n)}(\cdot)$ given above and symmetric traffic load $\rho_i^{(m,n)} = m\rho/n$ for each file i where $\rho = \lambda\nu < \xi$. For given (m,n), let $\pi_k^{(m,n)} = \sum_{\mathbf{x}:|A_\mathbf{x}|=k} \pi^{(m,n)}(\mathbf{x})$ for $k = 0,1,2,\ldots,n$, and let*

$$\alpha^* \triangleq \frac{1}{c}\log\left(\frac{1}{1-\rho/\xi}\right). \tag{14}$$

Then, for each $\epsilon > 0$, we have that

$$\lim_{m\to\infty}\lim_{n\to\infty} \sum_{k=\lfloor\alpha^* m(1-\epsilon)\rfloor}^{\lfloor\alpha^* m(1+\epsilon)\rfloor} \pi_k^{(m,n)} = 1 \tag{15}$$

Also, under the same limits, the expected delay is given by

$$\lim_{m\to\infty}\lim_{n\to\infty} E[D^{(m,n)}] = \frac{\alpha^*}{\lambda} = \frac{1}{\lambda c}\log\left(\frac{1}{1-\rho/\xi}\right). \tag{16}$$

The intuition underlying this result is as follows. Eq. (15) indicates that the probability measure $\pi^{(m,n)}(\mathbf{x})$ concentrates on states $\mathbf{x}$ such that $|A_\mathbf{x}| \approx m\alpha^*$ where α^* is given by (14). For such states, we should equilibrate the total load to the system with the service capacity, i.e., for large enough m, $h^{(m,n)}(\alpha^* m) \approx m\rho$. Notice that $\lim_{m\to\infty}\frac{1}{m}h^{(m,n)}(\alpha^*) = \xi(1-e^{-c\alpha^*})$. Setting α^* such that $\xi(1-e^{-c\alpha^*}) = \rho$ recovers (14). As m,n become large, if the flows in the system are in states $\mathbf{x}$ for which $h^{(m,n)}(|A_\mathbf{x}|) \gg \rho m$ or $h^{(m,n)}(|A_\mathbf{x}|) \ll \rho m$, it will quickly drift towards equilibrated states.

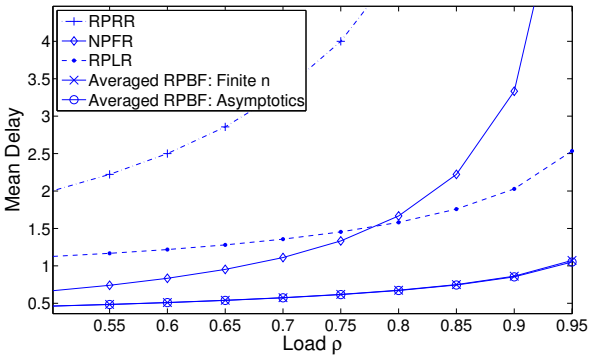

Fig. 1. Mean delay comparison of different systems with approximations for RPBF file-server system in the asymptotic regime: $\xi = 1$, $\nu_i = 1$ for all $i \in F$, $\rho_i = \rho m/n$ for all $i \in F$, and $c = 3$. Additionally, for finite n approximation of RPBF, $m = 400$, $n = 2\times 10^6$.

Fig. 1 exhibits plots for mean delay as a function of load for *averaged* RPBF systems. One corresponds to a finite system with $(m,n) = (400, 2\times 10^6)$ and computed using Corollary 1 while the other is based on the asymptotic result in Theorem 5. As can be seen the asymptotic formula is remarkably close. In next section we discuss why these expressions are good approximations for the actual performance in RPBF realizations.

B. Approximating performance of RPBF file-server system via 'averaged' RPBF.

In this subsection, we argue that the expression for mean delay given in Theorem 5 based on the averaged RPBF system can also be used to approximate the performance of realizations of a large RPBF file server systems.

Recall that $M^{(m,n)}(.)$ denotes the random rank function for our (m,n) RPBF system, and $\bar{\mu}^{(m,n)}(.)$ its mean over all random file placements. Below we provide an informal argument to show that, with high probability, the following is true for most sets A of size αm:

$$\frac{1}{m}M^{(m,n)}(A) \approx \frac{1}{m}\bar{\mu}^{(m,n)}(A) = \frac{1}{m}h^{(m,n)}(\alpha m)$$

This further suggests via Theorem 5 that asymptotically, even for asymmetric RPBF systems, concentration will once again happen at $\alpha^* m$ such that $\frac{1}{m}h^{(m,n)}(\alpha^* m) \approx \rho$.

Recall that $M^{(m,n)}(A) = \xi\sum_{s\in S^{(m)}} \mathbf{1}_{\left\{s\in S_c^{(m,n)}(A)\right\}}$, where $S^{(m)}$ and $S_c^{(m,n)}(A)$ are respectively the set of m servers, and the (random) set of servers where a copy of at least one of the files in A is stored. Suppose, for each (m,n), a subset of files $A_\alpha^{(m,n)}$ is selected uniformly at random amongst all $A \subset F^{(n)}$ such that $|A| = \lfloor\alpha m\rfloor$.

Suppose $S^{(m)} = \{s_1, s_2, \ldots, s_m\}$. Consider a random process $X^{(m,n)} = \left(X_1^{(m,n)}, X_2^{(m,n)}, \ldots, X_m^{(m,n)}\right)$ where

$$X_i^{(m,n)} = \mathbf{1}_{\left\{s_i \in S_c^{(m,n)}\left(A_\alpha^{(m,n)}\right)\right\}},\ \forall i \leq m$$

Then, $M^{(m,n)}\left(A_\alpha^{(m,n)}\right) = \xi \sum_{i=1}^m X_i^{(m,n)}$. We now study $\lim_{m\to\infty}\lim_{n\to\infty}\frac{1}{m}M^{(m,n)}\left(A_\alpha^{(m,n)}\right)$.

It can be checked that for each n, $X^{(m,n)}$ is a process of m exchangeable Bernoulli$\left(1-(1-c/m)^{\lfloor \alpha m \rfloor}\right)$ random variables, and so is $X^{(m,\infty)} \triangleq \lim_{n\to\infty} X^{(m,n)}$. Also, for any fixed set of l servers, say $\{s_1, s_2, \ldots, s_l\}$, $X_i^{(m,\infty)}$ for $i \in \{1, 2, \ldots, l\}$ become independent in the limit as $m \to \infty$. As was shown in [19], such asymptotic independence implies a law of large numbers to hold for a sequence of exchangeable random processes, which for our case implies that $\lim_{m\to\infty}\frac{1}{m}\sum_{i=1}^m X_i^{(m,\infty)} = \lim_{m\to\infty} E[X_1^{(m,\infty)}] = 1 - e^{-\alpha c}$ in probability. Thus, with high probability, $\frac{1}{m}M^{(m,n)}\left(A_\alpha^{(m,n)}\right) \approx \frac{1}{m}h^{(m,n)}(\alpha m)$ for almost all sets A of size αm, showing our claim. Also, in [17] we numerically check goodness of the approximation of mean delay using an 'averaged' polymatroid capacity for a file-server system with $m = 4$ and $n = 6$, and find that the performance of the exact and averaged systems are remarkably close.

C. Performance Comparisons

We now compare the performance of RPBF file-server system with several alternatives.

Random Placement with Random Routing (RPRR): Files are stored uniformly at random as in RPBF. However, upon arrival, a request for a given file is randomly routed to one of the c servers that has the file. Thus, each request is served by a single server, i.e., there is no pooling. As $n \to \infty$, the total load of ρm gets balanced across m servers and the system is equivalent to m independent $M/GI/1$ processor sharing systems with load ρ and service rate ξ.

Random Placement with Least loaded Routing (RPLR): Files are stored uniformly at random. Upon arrival, requests are routed to the server with least number of ongoing jobs among c servers that store the corresponding file. In the limit as $n \to \infty$, the load corresponding to each file tends to 0 and assigning c servers to each file at random beforehand is equivalent to assigning c servers to requests upon arrival. Thus, the allocation policy of this system is equivalent to the super-market model studied in [4], [14], [20] where upon each arrival, c servers are chosen at random and the request is assigned to the least loaded. As $m \to \infty$, for exponential service requirements one can show the mean delay is given by

$$E[D_{\text{RPLR}}] = \frac{1}{\lambda}\sum_{k=1}^{\infty}(\rho/\xi)^{\frac{c^k-1}{c-1}}. \tag{17}$$

Non-overlapping Pools with Fixed Routing (NPFR): The system's m servers are divided into m/c groups, each of size c. Each server group stores a mutually exclusive subset with nc/m files. Within a server group, all servers store the same files. Each ongoing file request is served by all the servers in the corresponding group. This system is thus equivalent to m/c independent $M/GI/1$ queues with load ρ and service rate ξc. Under the processor sharing discipline, the mean delay for this system is given by

$$E[D_{\text{NPFR}}] = \frac{\nu}{c\xi(1-\rho/\xi)}. \tag{18}$$

Contrast this with Theorem 5 where the mean delay increase is logarithmic in $1/(1-\rho/\xi)$.

In Fig. 1, we compare the performance of these three systems with RPBF. RPBF's performance is plotted using the approximations developed in the previous subsection. As expected, RPRR performs poorly as it does not exploit pooling or load dependent routing. RPLR and NPFR outperform each other in different regimes. NPFR performs better at lower loads where pooling works to its advantage. However, RPLR benefits from load balancing due to randomized storage as well as dynamic load balancing due to least loaded routing. As a result, it performs better at higher loads.

RPBF outperforms all these systems. Its performance improves by a factor of 2 over supermarket model (i.e., RPLR as $n \to \infty$) even at higher loads for $c = 3$. For larger values of c, the improvements are even greater. For example, for $c = 5$, the mean delay improves by a factor of up to 3. This is surprising since RPLR already enjoys the benefits of flow level load balancing. Also, at higher loads, one might expect that the gains of RPBF over RPLR due to pooling may be limited since load balancing in RPLR would ensure that most of the servers are busy serving requests for most of the time and are thus well utilized. The significant performance improvements of RRBF shows that the fairness based resource allocation is worthwhile to optimize performance in file-server systems.

V. System Tradeoffs

A. Recovery costs on correlated failure v/s Performance

We consider the cost of recovering files when there are large-scale correlated failures such as those occurring after power outages, see [5] for extensive discussion. It is not uncommon in datacenters that about 1% of servers fail to reboot after a power outage. The system then needs to recover data in these servers by retrieving copies from the servers that successfully rebooted. However, there might be some files for which no copy exists in the datacenter due to the failure of all servers in which it was stored. The probability of such an event occurring can be significant especially when the total number of files in the system is large.

When this occurs the system needs to locate and recover the lost files from a cold storage. Recovery of the files from cold storage may incur a high fixed cost but may not be greatly affected by the number of files lost. Thus in practice (as argued in [5]) it is desirable that the probability that one or more files are lost during power outage events be low. This can be achieved by constraining randomness in how files are copied

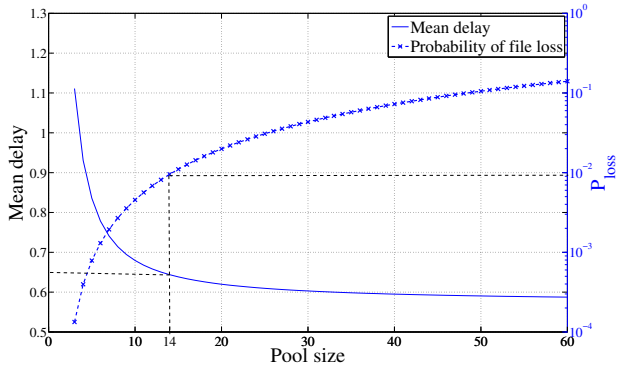

Fig. 2. Delay v/s reliability $n = 2 \times 10^6$, $m = 400$, $c = 3$, $\gamma = 0.01$, $\rho = 0.7$, and $\nu = 1$.

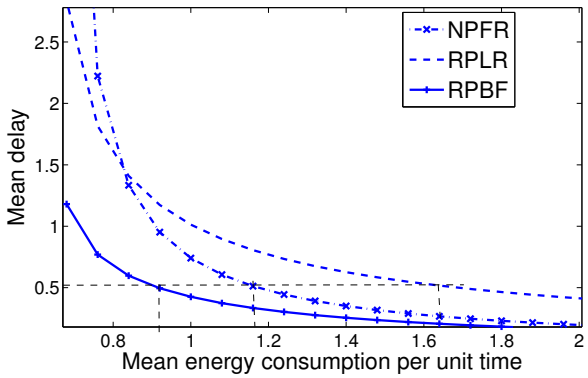

Fig. 3. Energy-delay tradeoff with varying server speed ξ for asymptotic regime where $m \to \infty$: load per server fixed at $\rho = 0.8$, $\nu = 1$, and $c = 3$.

across servers. The intuition from Section IV suggests that randomly 'spreading' the files across *all* the servers so that their pools overlap improves the user perceived performance. However, this may increase the probability of a file loss. To study how these quantities are related, consider a storage policy that divides m servers into independent pools of smaller size and restricts the copies of each file to be placed within a single pool, as follows.

Fix an integer κ such that $c \leq \kappa \leq m$. Suppose, for now, that number of servers m is divisible by κ and that number of files n is divisible by m/κ. Divide the set S of m servers into m/κ number of pools each of size κ. Similarly, divide the set F of n files into disjoint m/κ groups of size $n\kappa/m$. Associate each group of files with a distinct pool of servers. Then, for each file, independently store c copies by selecting c servers uniformly at random from the corresponding pool.

Suppose that upon a power outage, each server fails to reboot with probability γ independently. Then, for a pool of size κ, the probability that l servers fail is $\binom{\kappa}{l}\gamma^l(1-\gamma)^{\kappa-l}$, so the probability that one or more files are lost can be given by

$$P_{\text{loss}} = 1 - \left(\sum_{l=0}^{c-1} \binom{\kappa}{l} \gamma^l (1-\gamma)^{\kappa-l} + \sum_{l=c}^{\kappa} \binom{\kappa}{l} \gamma^l (1-\gamma)^{\kappa-l} \left(1 - \frac{\binom{l}{c}}{\binom{\kappa}{c}} \right)^{n\kappa/m} \right)^{m/\kappa}$$

For the general case where m is not divisible by κ or n is not divisible by m/κ, we can create non uniform pools and compute the corresponding loss probability. We use the above expression as a simpler approximation by using $\lfloor m/\kappa \rfloor$ and $\lfloor n\kappa/m \rfloor$ appropriately. Also, the performance within each pool can be computed using the expression of Corollary 1, which gives a reasonable approximation as explained in Sec. IV.

Fig. 2 plots mean delay and P_{loss} for $\gamma = 0.01$ for a system with $n = 2 \times 10^6$, $m = 400$, and $c = 3$ copies. The load per server is $\rho = 0.7$, i.e., the total load on the system is $m\rho = 280$ and is distributed uniformly across files. Also, $\nu_i = 1$ for all $i \in F$ and $\mu_s = 1$ for all $s \in S$. As can be seen, varying κ trades off performance with file loss probability. As κ increases mean delay decreases but quickly saturates at 0.57, which matches with the asymptotic limit as given by Theorem 5. At $\kappa = 14$, mean delay is 0.64 which is about 12% greater than the asymptotic value, while P_{loss} is less than 1%. Decreasing κ can further lower P_{loss} but at the cost of a significant increase in mean delay.

B. Energy-Delay tradeoffs

Next, we consider RPBF systems where servers' processing speed is a bottleneck. The processing speed can be improved by increasing clock frequency and voltage supply, which in turn increases energy consumption. This dependence is typically modeled through a polynomial relationship of power with ξ, i.e., when the service rate of a server is ξ, the power consumption is $f(\xi) = \xi^\alpha/\beta$ per unit time where $\alpha > 1$ and β is a positive constant [12]. In practice, even when ξ is set to 0, there is non-negligible leakage power consumption. Since our focus is on dynamic power, we ignore leakage power here. The choice of ξ trades off performance for energy consumption. Here, we consider a simple semi-static policy where each server operates at a fixed rate ξ when busy and rate 0 when idle, thus consuming negligible power when idle. For $M/GI/1$ queues, it was shown in [12] that such a simple policy, with ξ chosen judiciously, is close to an optimal policy for minimizing a weighted average of mean delay and energy consumption across all dynamic policies where ξ is allowed to vary with queue state.

Fig. 3 compares the energy-performance tradeoff for NPFR, RPLR, and RPBF where the plots are obtained by varying values of ξ. For RRBF, Theorem 5 is used to compute dependence of performance on ξ, whereas for NPFR and RPLR, (18) and (17), respectively, are used. Also, we assume that the power consumption as a function of ξ is given by $f(\xi) = \xi^2$. Since the fraction of time a server is busy in each system is ρ/ξ, the mean energy consumption is given by $E = \rho\xi$. To obtain performance of 0.5 for $\rho = 0.8$, the energy consumption for NPFR and RPLR systems is 20% and 70%

more than that for RPBF, respectively.

VI. Conclusions

In this paper, we propose a simple model to quantify design tradeoffs associated with large scale CDNs leveraging dynamic service allocation accross shared server/network resources. The benefits of doing so are significant, thus it is not surprising that current CDNs are moving in this direction increasingly incorporating such P2P like service models. Our work represents a first step towards developing the performance models needed for a disciplined engineering and optimization of such systems.

References

[1] T. Bonald, L. Massoulié, A. Proutiere, and J. Virtamo. A queueing analysis of max-min fairness, proportional fairness and balanced fairness. *Queueing Systems: Theory and Applications*, 53:65–84, 2006.

[2] T. Bonald and A. Proutiere. Insensitive bandwidth sharing in data networks. *Queueing Systems: Theory and Applications*, 44:69–100, 2003.

[3] T. Bonald and J. Virtamo. Calculating the flow level performance of balanced fairness in tree networks. *Perform. Eval.*, 58(1):1–14, Oct. 2004.

[4] M. Bramson, Y. Lu, and B. Prabhakar. Randomized load balancing with general service time distributions. In *Proceedings of the ACM Sigmetrics*, pages 275–286, 2010.

[5] A. Cidon, S. Rumble, R. Stutsman, S. Katti, J. Ousterhout, and M. Rosenblum. Copysets: Reducing the frequency of data loss in cloud storage. In *Usenix Annual Technical Conference*, pages 37–48, 2013.

[6] J. Edmonds. Submodular functions, matroids, and certain polyhedra. In *Proceedings of Calgary International Conference on Combinatorial Structures and Applications*, pages 69–87, 1969.

[7] A. Gandhi, V. Gupta, M. Harchol-Balter, and M. A. Kozuch. Optimality analysis of energy-performance trade-off for server farm management. *Perform. Eval.*, 67(11):1155–1171, Nov. 2010.

[8] H. Han, S. Shakkottai, C. V. Hollot, R. Srikant, and D. Towsley. Multi-path TCP: a joint congestion control and routing scheme to exploit path diversity in the internet. *IEEE/ACM Trans. Netw.*, 14(6):1260–1271, Dec. 2006.

[9] V. Joseph and G. de Veciana. Stochastic networks with multipath flow control: impact of resource pools on flow-level performance and network congestion. In *Proceedings of ACM Sigmetrics*, pages 61–72, 2011.

[10] P. Key, L. Massoulié, and D. Towsley. Path selection and multipath congestion control. *Commun. ACM*, 54(1):109–116, Jan. 2011.

[11] M. Leconte, M. Lelarge, and L. Massoulié. Bipartite graph structures for efficient balancing of heterogeneous loads. In *Proceedings of the 12th ACM Sigmetrics/Performance*, pages 41–52, 2012.

[12] M. Lin, A. Wierman, L. Andrew, and E. Thereska. Dynamic right-sizing for power-proportional data centers. In *Proceedings of IEEE Infocom*, pages 1098–1106, 2011.

[13] L. Massoulié. Structural properties of proportional fairness: Stability and insensitivity. *Annals of Applied Probability*, 17(3):809–839, 2007.

[14] M. D. Mitzenmacher. *The Power of Two Choices in Randomized Load Balancing*. PhD thesis, University of California, Berkeley, 1996.

[15] J. Mo and J. Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on Networking*, 8(5):556 –567, Oct. 2000.

[16] G. L. Nemhauser and L. A. Wolsey. *Integer and combinatorial optimization*, volume 18. Wiley New York, 1988.

[17] V. Shah and G. de Veciana. Performance evaluation and asymptotics for content delivery networks. Technical Report, available at http://users.ece.utexas.edu/∼gustavo/ShV13.pdf, 2013.

[18] A. L. Stolyar. An infinite server system with customer-to-server packing constraints. In *IEEE Annual Allerton Conference on Communication, Control, and Computing*, pages 1713–1720, 2012.

[19] A. S. Sznitman. Topics in propagation of chaos. In *Ecole d'Eté de Probabilités de Saint-Flour XIX1989*, pages 165–251. Springer, 1991.

[20] N. D. Vvedenskaya, R. L. Dobrushin, and F. I. Karpelevich. Queueing system with selection of the shortest of two queues: An asymptotic approach. *Problemy Peredachi Informatsii*, 32(1):20–34, 1996.

[21] A. Wierman, L. L. H. Andrew, and A. Tang. Power-aware speed scaling in processor sharing systems: Optimality and robustness. *Perform. Eval.*, 69(12):601–622, Dec. 2012.

Appendix

A. Sketch of proof of Theorem 1

We first show that μ is a rank function. By definition it is clear that $\mu(\emptyset) = 0$ and that μ is monotonic. To show that $\mu(.)$ is submodular we use the inclusion-exclusion principle to obtain

$$\begin{aligned}\mu(A) &= \sum_{s\in S(A)} \mu_s = \sum_{s\in S(A\cap B)\cup S(A\setminus B)} \mu_s \\ &= \sum_{s\in S(A\cap B)} \mu_s + \sum_{s\in S(A\setminus B)} \mu_s - \sum_{s\in S(A\cap B)\cap S(A\setminus B)} \mu_s.\end{aligned}$$

Similarly, one can obtain $\mu(B)$, $\mu(A\cup B)$, and $\mu(A\cap B)$, and show that $\mu(A)+\mu(B)-\mu(A\cup B)-\mu(A\cap B)\geq 0$ which shows that μ is submodular.

We then show that $\mathcal{C}$ is the capacity region. We show that each $\mathbf{r}\in\mathcal{C}$ is feasible by giving an achievability scheme for each extreme point of the dominant face $\mathcal{D} = \{\mathbf{r}\in\mathcal{C} : \sum_{i\in F} r_i = \mu(F)\}$ without violating Assumption 1.

B. Proof of Theorem 3

We prove this by induction on $|\mathbf{x}| \triangleq \sum_i x_i$. Clearly, the result is true when $|\mathbf{x}| = 1$. Lets assume that the claim is true for all $\mathbf{x}'$ such that $|\mathbf{x}'| < |\mathbf{x}|$ for a given $\mathbf{x}$. We show that it holds for $\mathbf{x}$ as well.

By definition of balanced fairness, i.e., by (2) and (3), there exists a B such that $\sum_{i\in B} r_i(\mathbf{x}) = \mu(B)$. Also, by monotonicity of $\mu(.)$, $B\subset A_{\mathbf{x}}$. If $B = A_{\mathbf{x}}$, then we are done. Suppose this is not the case. Then, from (2) and definition of B, we have

$$\Phi(\mathbf{x}) = \frac{\sum_{i\in B}\Phi(\mathbf{x}-\mathbf{e}_i)}{\mu(B)}. \tag{19}$$

Since the capacity condition $\sum_{i\in B} r_i(\mathbf{x}') \leq \mu(B)$ is satisfied for all states, we have $\sum_{i\in B} r_i(\mathbf{x}-\mathbf{e}_j)\leq\mu(B)$ for all $j\in A_{\mathbf{x}}\setminus B$. Using this in (19), we get

$$\Phi(\mathbf{x}) \leq \frac{\sum_{i\in B}\Phi(\mathbf{x}-\mathbf{e}_i)}{\sum_{i\in B} r_i(\mathbf{x}-\mathbf{e}_j)},\ \forall j\in A_{\mathbf{x}}\setminus B. \tag{20}$$

We now use this bound to compute one on the sum of all rates as follows:

$$\begin{aligned}\sum_{i\in A_{\mathbf{x}}} r_i(\mathbf{x}) &= \sum_{i\in B} r_i(\mathbf{x}) + \sum_{j\in A_{\mathbf{x}}\setminus B} r_j(\mathbf{x}), \\ &= \mu(B) + \sum_{j\in A_{\mathbf{x}}\setminus B} \frac{\Phi(\mathbf{x}-\mathbf{e}_j)}{\Phi(\mathbf{x})}, \\ &\geq \mu(B) + \sum_{j\in A_{\mathbf{x}}\setminus B} \frac{\sum_{i\in B} r_i(\mathbf{x}-\mathbf{e}_j)\Phi(\mathbf{x}-\mathbf{e}_j)}{\sum_{i\in B}\Phi(\mathbf{x}-\mathbf{e}_i)}, \\ &= \mu(B) + \sum_{j\in A_{\mathbf{x}}\setminus B} \frac{\sum_{i\in B}\Phi(\mathbf{x}-\mathbf{e}_j-\mathbf{e}_i)}{\sum_{i\in B}\Phi(\mathbf{x}-\mathbf{e}_i)},\end{aligned}$$

$$= \mu(B) + \frac{\sum_{i\in B}\sum_{j\in A_{\mathbf{x}}\setminus B}\Phi(\mathbf{x}-\mathbf{e}_j-\mathbf{e}_i)}{\sum_{i\in B}\Phi(\mathbf{x}-\mathbf{e}_i)},$$
$$\geq \mu(B) + \frac{\sum_{j\in A_{\mathbf{x}}\setminus B}\Phi(\mathbf{x}-\mathbf{e}_j-\mathbf{e}_{i^*})}{\Phi(\mathbf{x}-\mathbf{e}_{i^*})}, \quad (21)$$

where $i^* = \arg\min_{i\in B}\left\{\frac{\sum_{j\in A_{\mathbf{x}}\setminus B}\Phi(\mathbf{x}-\mathbf{e}_j-\mathbf{e}_i)}{\Phi(\mathbf{x}-\mathbf{e}_i)}\right\}$. In the last inequality (21), we have used the identity $\frac{a+b}{c+d} \geq \frac{a}{c}$ if $\frac{a}{c} \leq \frac{b}{d}$. Thus, we get the following inequality.

$$\sum_{i\in A_{\mathbf{x}}} r_i(\mathbf{x}) \geq \mu(B) + \sum_{j\in A_{\mathbf{x}}\setminus B} r_j(\mathbf{x}-\mathbf{e}_{i^*}). \quad (22)$$

We now only need to show $\mu(B) + \sum_{j\in A_{\mathbf{x}}\setminus B} r_j(\mathbf{x}-\mathbf{e}_{i^*}) \geq \mu(A_{\mathbf{x}})$. The following two cases are possible for the given $\mathbf{x}$.

Case 1 $x_{i^*} = 1$: Then, in state $\mathbf{x}-\mathbf{e}_{i^*}$, only classes in $A_{\mathbf{x}}\setminus\{i^*\}$ are active. Thus, we have,

$$\sum_{j\in A_{\mathbf{x}}\setminus B} r_j(\mathbf{x}-\mathbf{e}_{i^*}) + \mu(B)$$
$$= \mu(A_{\mathbf{x}}\setminus\{i^*\}) - \sum_{k\in B\setminus\{i^*\}} r_k(\mathbf{x}-\mathbf{e}_{i^*}) + \mu(B),$$
$$\geq \mu(A_{\mathbf{x}}\setminus\{i^*\}) - \mu(B\setminus\{i^*\}) + \mu(B),$$
$$\geq \mu(A_{\mathbf{x}}),$$

where the equality follows from induction hypothesis, the first inequality follows from the capacity constraint on set $B\setminus\{i^*\}$, and the last inequality follows from the submodularity of $\mu(.)$.

Case 2 $x_{i^*} > 1$: Here, all the classes in $A_{\mathbf{x}}$ are active in state $\mathbf{x}-\mathbf{e}_{i^*}$ as well, i.e., $A_{\mathbf{x}} = A_{\mathbf{x}-e_{i^*}}$. Thus, we have,

$$\sum_{j\in A_{\mathbf{x}}\setminus B} r_j(\mathbf{x}-\mathbf{e}_{i^*}) + \mu(B) \geq \sum_{i\in A_{\mathbf{x}}} r_i(\mathbf{x}-\mathbf{e}_{i^*})$$
$$= \mu(A_{\mathbf{x}}),$$

where the inequality follows from the capacity constraint on set B, and the equality follows from induction hypothesis. Thus, the result holds for both the cases.

C. Proof of Theorem 4

By Little's law,

$$E[D_i] = \frac{\sum_{\mathbf{x}} x_i\pi(\mathbf{x})}{\lambda_i} = \frac{\nu_i\frac{\partial}{\partial\rho_i}G(\boldsymbol{\rho})}{G(\boldsymbol{\rho})}. \quad (23)$$

Thus, to prove the result we only need to show (5). Equation (7) follows by taking derivative of (5) w.r.t. ρ_i. From Theorem 3 and (3) we have,

$$\Phi(\mathbf{x}) = \frac{\sum_{i\in A_{\mathbf{x}}}\Phi(\mathbf{x}-\mathbf{e}_i)}{\mu(A_{\mathbf{x}})}. \quad (24)$$

Let $G_A(\rho) = \sum_{\mathbf{x}:A_{\mathbf{x}}=A}\Phi(\mathbf{x})\prod_{i\in F}\rho^{x_i}$. Thus, we get , $G(\boldsymbol{\rho}) = \sum_{A\subset F} G_A(\boldsymbol{\rho})$ and

$$G_A(\boldsymbol{\rho}) = \sum_{\mathbf{x}:A_{\mathbf{x}}=A}\frac{\sum_{i\in A}\Phi(\mathbf{x}-\mathbf{e}_i)}{\mu(A)}\prod_{j\in F}\rho^{x_j},$$
$$= \frac{\sum_{i\in A}\sum_{\mathbf{x}:A_{\mathbf{x}}=A}\Phi(\mathbf{x}-\mathbf{e}_i)\prod_{j\in F}\rho^{x_j}}{\mu(A)},$$

Rearranging terms, we get,

$$\mu(A)G_A(\boldsymbol{\rho}) = \sum_{i\in A}\rho_i\sum_{\mathbf{x}:A_{\mathbf{x}}=A\setminus\{i\}}\Phi(\mathbf{x})\prod_{j\in F}\rho^{x_j}$$
$$+\sum_{i\in A}\rho_i\sum_{\mathbf{x}:A_{\mathbf{x}}=A}\Phi(\mathbf{x})\prod_{j\in F}\rho^{x_j},$$
$$= \sum_{i\in A}\rho_i G_{A\setminus\{i\}}(\boldsymbol{\rho}) + G_A(\boldsymbol{\rho})\sum_{i\in A}\rho_i,$$

further simplification of which gives the desired result.

D. Sketch of proof of Corollary 1

From symmetry it follows that $G_A(\boldsymbol{\rho})$ depends on A only through $|A|$. Then, eqns. (10) - (13) follow by letting $F_k(\rho) = \sum_{B:|B|=k} G_B(\boldsymbol{\rho})$ which equals $\binom{n}{k}G_A(\boldsymbol{\rho})$ for some A such that $|A| = k$, and similarly letting $\hat{F}_k(\rho) = \sum_{B:|B|=k}\frac{\partial}{\partial\rho_i}G_B(\boldsymbol{\rho})$ which equals $\binom{n-1}{k-1}\frac{\partial}{\partial\rho_i}G_A(\boldsymbol{\rho})$ for some A such that $|A| = k$ and $i \in A$, and further simplification.

E. Sketch of proof of Theorem 5

We first prove the following lemma by finding an explicit expression for $\pi_k^{(m,n)}$ for each k for given m and n and then taking the limit as $n \to \infty$ for a fixed m. Let $\lim_{n\to\infty}\pi_k^{(m,n)} = \pi_k^{(m,\infty)}$. Also let $h^{(m,\infty)}(k) = \xi m(1-(1-c/m)^k)$ for $k = 0, 1, 2, \ldots, \infty$.

Lemma 1. *For any fixed integers k_1 and k_2 such that $k_1 > k_2$, we have*

$$\frac{\pi_{k_1}^{(m,\infty)}}{\pi_{k_2}^{(m,\infty)}} = \frac{(m\rho)^{k_1-k_2}}{\prod_{l=k_2+1}^{k_1}h^{(m,\infty)}(l)} \quad (25)$$

Proof: Fix m and n. From definitions of $F_k(.)$ from the proof of Corollary 1 one can show that $\pi_k^{(m,n)} = \frac{F_k(m\rho/n)}{F(m\rho/n)}$ for $k = 1, \ldots, n$ where $F_k(m\rho/n)$ and $F(m\rho/n)$ are given by recursive expressions in the statement of Corollary 1. Thus, from (11), we get $\pi_0^{(m,n)} = 1/F(m\rho/n)$ and

$$\pi_k^{(m,n)} = \frac{(n-k+1)\frac{m\rho}{n}\pi_{k-1}^{(m,n)}}{h^{(m,n)}(k) - k\frac{m\rho}{n}}, \text{ for } k = 1, \ldots, n.$$

Thus, for any $k_1 > k_2$ we get

$$\frac{\pi_{k_1}^{(m,n)}}{\pi_{k_2}^{(m,n)}} = \frac{(n-k_2)!(\frac{m\rho}{n})^{k_1-k_2}}{(n-k_1)!\prod_{l=k_2+1}^{k_1}(h^{(m,n)}(l) - l\frac{m\rho}{n})}$$
$$\xrightarrow[n\to\infty]{} \frac{(m\rho)^{k_1-k_2}}{\prod_{l=k_2+1}^{k_1}h^{(m,\infty)}(l)}$$

■

Now, for any $\alpha > 0$, we have

$$\lim_{m\to\infty}\frac{1}{m}h^{(m,\infty)}(\lfloor\alpha m\rfloor) = \xi(1-e^{-\alpha c}).$$

Let k'_m be the largest k such that $h^{(m,\infty)}(k) \leq m\rho$. Thus, it is easy to show that $k'_m/m \to \alpha^*$ as $m \to \infty$ where α^* is given by (14). Further, we show that tail of the measure $\pi_k^{(m,\infty)}$ is lighter than geometric distribution for a given m. We also show that for any $\epsilon > 0$, for any k such that $k < (1-2\epsilon)k'_m$ or $k > (1+2\epsilon)k'_m$, $\pi_k^{(m,\infty)}$ decreases to 0 geometrically fast as $m \to \infty$, from which the result follows.