Optimal Content Replication and Request Matching in Large Caching Systems

Arpan Mukhopadhyay* , Nidhi Hegde† and Marc Lelarge‡
*IC/LCA-2, EPFL, Switzerland. Email: arpan.mukhopadhyay@epfl.ch
†Nokia Bell Labs, France. Email: nidhi.hegde@nokia-bell-labs.com
‡INRIA-ENS, Paris, France. Email: marc.lelarge@ens.fr

***Abstract*—We consider models of content delivery networks in which the servers are constrained by two main resources: memory and bandwidth. In such systems, the throughput crucially depends on how contents are replicated across servers and how the requests of specific contents are matched to servers storing those contents. In this paper, we first formulate the problem of computing the optimal replication policy which if combined with the optimal matching policy maximizes the throughput of the caching system in the stationary regime. It is shown that computing the optimal replication policy for a given system is an NP-hard problem. A greedy replication scheme is proposed and it is shown that the scheme provides a constant factor approximation guarantee. We then propose a simple randomized matching scheme which avoids the problem of interruption in service of the ongoing requests due to re-assignment or repacking of the existing requests in the optimal matching policy. The dynamics of the caching system is analyzed under the combination of proposed replication and matching schemes. We study a limiting regime, where the number of servers and the arrival rates of the contents are scaled proportionally, and show that the proposed policies achieve asymptotic optimality. Extensive simulation results are presented to evaluate the performance of different policies and study the behavior of the caching system under different service time distributions of the requests.**

I. Introduction

Recent years have seen an explosive growth in Internet traffic, stemming mainly from the transfer of multi-media contents, e.g., streaming videos, movies etc. It is expected that video streaming services and downloads will account for more than 81% of all Internet's traffic by 2021 [1]. Such growth in multi-media traffic has led to the emergence of content delivery networks (CDNs) and peer-to-peer systems, which support the demand for contents by replicating popular contents at the network periphery (e.g., boxes or servers). Popular video-streaming services such as Netflix, Youtube often use CDN's to serve the requests of their most popular contents.

Large CDNs usually consist of a central server, storing an entire catalogue of contents, and a large number of edge servers, each storing a small fraction of these contents in their caches and serving requests of the stored contents [2]. In such systems, it is assumed that access to the central server is expensive. Therefore, a large portion of the content requests must be served by the edge servers that are constrained by their limited memory and bandwidth capacities. In this paper, we model these servers as loss servers [3] and aim at minimizing the number of requests blocked at these servers (and thus need to be sent to the central server). We do not consider queueing of the requests since we focus on delay-sensitive streaming services, which comprise a large proportion of the Internet's traffic today [1].

Efficiency of such systems crucially depends on the replication (also called allocation) policy used to populate the caches of the servers and the request matching policy used to dispatch the incoming requests. In this paper, we first formulate the problem of computing the optimal allocation policy, which, if combined with the optimal matching policy (maximum matching), maximizes the number requests served per unit time by the caching system in the stationary regime. To the best of our knowledge, this is the first work that addresses this joint allocation-matching problem. The joint problem for finite systems is shown to be an NP-hard problem by reduction from the 3-partition problem. A polynomial-time greedy allocation scheme is proposed and it is shown to achieve a constant factor approximation of the optimal value.

Next, we turn our attention to the dynamics of the system which are determined by the matching policy in use. In earlier works, e.g., [4], [5] a maximum matching scheme has been considered to match incoming requests to servers. However, in the maximum matching scheme, the service of ongoing requests may be interrupted to incorporate a newly arrived request. Such interruption in service is clearly not desired and may cause significant delay in serving the requests. We thus propose a simple randomized policy for request matching which does not cause interruption to the already existing requests. We show that this matching scheme coupled with the proposed greedy replication policy or a previously studied [4], [6] 'proportional-to-product' replication policy is asymptotically optimal in a limiting regime where the number of cache servers and request arrival rates scale proportionally with each other and the number of contents remains fixed. Such a scaling regime corresponds to scenarios where a fixed number of most popular contents are served by a large number of cache servers. The proof of asymptotic optimality uses fluid limits of processes describing the dynamics of the system. The fluid limit in our case cannot be described by ordinary differential equations (ODE's) since it describes a non-smooth dynamical system. The novelty in our approach lies in describing the fluid limit as a solution to a differential inclusion (DI) system and showing that all trajectories of the solutions converge to the same global attractor. In summary, our main contributions are

as follows.

- We show that the joint problem is NP-hard and propose a polynomial-time approximation algorithm that achieves performance within a constant of the optimal value;
- We propose a randomized matching policy that avoids interruption of service and show that this policy is asymptotically optimal for large systems when combined with the proposed allocation policies.
- We employ a new approach based on the theory of differential inclusions to prove fluid limit results.
- We also present extensive simulation results to compare the different allocation and matching schemes discussed in this paper. Near insensitivity of the system to service time distributions is also studied.

Related Work: Content placement in caching systems has been the subject of study for many years now. Of the numerous papers on this topic, we now mention a few that are most relevant to our work. In [6]–[8], optimal cache allocation policy was designed without taking into account the bandwidth restrictions of the servers. A loss model of caching systems was first introduced in [4] in the context of peer-to-peer video on demand services. A replication policy in which the number of replicas is proportional to the arrival rate of the contents was proposed and analyzed in conjunction with the maximum matching algorithm which is different from the setting in our paper. In [5] a similar loss model was considered. However, the objective was to maximize the utilization of the resources as opposed to maximization of throughput. In [9], a discrete-time model similar to ours is considered. The objective there is to minimize the expected transmission rate from the main server in order to serve all requests in a time slot. A different scaling regime in which the number of contents is scaled is considered. We do not consider such a scaling regime in this paper since our focus is on a scenarios where only a fixed number of highly popular contents are present at any instant. An online matching policy similar to the proposed matching policy was considered in [10] for cloud computing systems. However, the setting there is completely different as the servers do not have any memory restrictions.

The remainder of the paper is organized as follows. Section II introduces the system model. In Section III we formulate the joint allocation and matching problem and analyze the complexity of the problem. In Section IV we present efficient approximate algorithms. In Section V we present a matching policy that does not involve repacking of ongoing requests, and analyze the system in the large-systems asymptotic regime. We prove optimality in the large-systems scaling regime. Section VI presents simulation results. Finally, the paper is concluded in Section VII.

II. System model

We consider a dynamic model of caching systems in which requests for contents arrive at random instants and are served by corresponding servers. The servers are assumed to be able to serve only a finite number of requests simultaneously.

A. Server and storage Model

The caching system consists of n servers and m contents indexed by the sets $S = \{1, 2, \ldots, n\}$ and $C = \{1, 2, \ldots, m\}$, respectively. We assume that each server $s \in S$ is capable of storing up to $d_s \geq 1$ contents in its cache and has a bandwidth of $U_s > 0$, i.e., it can serve U_s requests simultaneously. The *replication*, or *allocation* policy is represented by a binary matrix $A = (a_{sc})_{s\in S, c\in C} \in \{0,1\}^{nm}$, with $a_{sc} = 1$ if c is stored in the cache of s and $a_{sc} = 0$, otherwise. Thus, for any *feasible replication policy* $A = (a_{sc})_{s\in S, c\in C}$, we have

$$\sum_{c\in C} a_{sc} \leq d_s \text{ for all } s \in S \tag{1}$$

The set of all feasible replication policies is denoted as $\mathcal{A} \subset \{0,1\}^{nm}$. The cache of each server is populated at $t = 0$ according to some cache allocation policy $A \in \mathcal{A}$ and is kept unchanged for all $t \geq 0$. Servers are assumed to have different bandwidths and memory sizes from the sets $\{U_i, i \in \mathcal{I}\}$ and $\{d_j, j \in \mathcal{J}\}$, respectively, where $\mathcal{I}$ and $\mathcal{J}$ are some finite index sets. The fraction of servers having bandwidth U_i and memory size d_j is denoted as α_{ij} for each $(i, j) \in \mathcal{I} \times \mathcal{J}$.

B. Service model

Requests of contents are assumed to arrive one at a time according to simple point processes. The average number of requests of a content $c \in C$ arriving per unit time is denoted by λ_c and is called the *arrival rate* or the *popularity* of the content. The vector of content popularities is denoted as $\Lambda = (\lambda_c, c \in C)$ and is assumed to be a known constant throughout the paper. Typically, the popularities of items containing videos or movies vary over periods ranging from few hours to few weeks [11] (and have to be estimated periodically [12]) which are slower than the time scales of interest in the paper. For this reason we do not consider the effect of time-varying popularities and errors in estimation of the popularities in this paper.

Arriving requests are matched or assigned to servers according to some matching scheme. Clearly, no matching policy can serve all arriving requests since the servers have limited storage and bandwidth capacities. The requests which cannot be served are *blocked* or dropped by the caching system and are immediately routed to a central server that stores all the contents. The accepted requests are assumed to stay in the system for a random amount of time, independent and identically distributed (i.i.d) with unit mean.

III. Optimal allocation: A static formulation

Our first objective is the design of an allocation policy which populates the caches at the start of the system. We seek an allocation A that solves the *static* optimization problem of maximizing the average number requests served per unit of time by the caching system operating in the stationary regime. To formulate the problem, we consider a time window of unit length. Let X_c, $c \in C$ denote the number of requests of content c arriving in this window and define $X := (X_c, c \in C)$. We make the following approximations: 1) the cache servers are

idle at the beginning of the time window, 2) all the requests that are accepted by the system in this time window stay in the system exactly till the end of the time window. Although these approximations do not capture dynamics of the departures of the requests in this window, they provide a good approximation of the stationary behavior of the caching system (since each request has an average service time of unit duration) and are appropriate when detailed statistics of the arrival processes and service times are not available. We analyze the dynamics of the caching system under specific assumptions on arrival processes and service time distributions later in this paper.

Let b_{sc} denote the total number of requests of content $c \in C$ matched/assigned to a server $s \in S$ in the given time window. Then, $B = (b_{sc})_{s \in Sc \in C}$ must satisfy

$$\sum_{c \in C} b_{sc} \leq U_s, \ \forall s \in S, \tag{2}$$

$$\sum_{s \in S} b_{sc} \leq X_c, \ \forall c \in C, \tag{3}$$

$$\mathbb{1}\left(b_{sc} > 0\right) \leq a_{sc}, \tag{4}$$

where $\mathbb{1}\left(\cdot\right)$ denotes the indicator function. Let $\mathcal{B}(A, X)$ denote the set of matchings B which satisfy (2)-(4). Thus, under allocation policy A, the maximum total number of requests that can be matched in this time window is $M(A, X) = \max_{B \in \mathcal{B}(A,X)} \sum_{s \in Sc \in C} b_{sc}$. Our goal is to find an allocation policy A for which $M(A, X)$ is maximized, i.e., we aim to find a solution to $\max_{A \in \mathcal{A}} M(A, X) = \max_{A \in \mathcal{A}} \max_{B \in \mathcal{B}(A,X)} \sum_{s \in S, c \in C} b_{sc}$.

We note that in the above formulation the random vector X appears in the constraints. Hence, the solution A^* of the above problem will only be optimal in windows where the demand vector is exactly equal to X. However, since random demands are unknown a priori and our goal is to find a solution which works well on average, we replace the vector X by its mean value, i.e., the *popularity vector* $\Lambda = (\lambda_c, c \in C)$. Hence, we consider the following problem:

$$\max_{A \in \mathcal{A}} M(A, \Lambda) = \max_{A \in \mathcal{A}} \max_{B \in \mathcal{B}(A,\Lambda)} \sum_{s \in S, c \in C} b_{sc} \tag{5}$$

We refer to the above as the joint allocation-matching (JAM) problem since in it the decision variable A also affects the matching $B \in \mathcal{B}(A, \Lambda)$. Our first result establishes the following equivalence.

Proposition 1. *Problem* (5) *is equivalent to the following problem*

$$\begin{aligned} \underset{Z=(z_{sc})_{s \in S, c \in C}}{\textit{Maximize}} \quad & \sum_{s \in S} \sum_{c \in C} z_{sc} \\ \textit{subject to} \quad & \sum_{c \in C} z_{sc} \leq U_s, \ \forall s \in S \\ & \sum_{s \in S} z_{sc} \leq \lambda_c, \ \forall c \in C \\ & \sum_{c \in C} \mathbb{1}\left(z_{sc} > 0\right) \leq d_s, \ \forall s \in S \\ & z_{sc} \in \mathbb{R}_+, \ \forall s \in S, \forall c \in C \end{aligned} \tag{6}$$

Furthermore, if $Z^* = (z^*_{sc})_{s \in S, c \in C}$ is an optimal solution of (6)*, then an optimal solution A^* of problem* (5) *can be found by setting $a^*_{sc} = \mathbb{1}\left(z^*_{sc} > 0\right)$, for all $(s, c) \in S \times C$.*

Proof. Let $Z = (z_{sc})_{s \in S, c \in C}$ be a feasible solution of (6). Set $a_{sc} = \mathbb{1}\left(z_{sc} > 0\right)$ and $b_{sc} = z_{sc}$ for all $s \in S$, $c \in C$. Then clearly we have $B = (b_{sc})_{s \in S, c \in C} \in \mathcal{B}\left(A, \lambda\right)$. Furthermore, $\sum_{s \in S} \sum_{c \in C} z_{sc} = \sum_{s \in S} \sum_{c \in C} b_{sc} \leq O_1$, where O_1 denotes the optimal value of problem (5). Taking the maximum of the LHS over the set of feasible solutions of (6) (this is possible since the feasible set of solutions is compact and the objective function is continuous) we have $O_2 \leq O_1$, where O_2 denotes the optimal value of problem (6).

Conversely, suppose that $A = (a_{sc})_{s \in S, c \in C} \in \mathcal{A}$, $B = (b_{sc})_{s \in S, c \in C} \in \mathcal{B}(A, \lambda)$. For all pairs $(s, c) \in S \times C$ such that $a_{sc} = 0$ we set $z_{sc} = 0$. For all other pairs $(s, c) \in S \times C$ we set $z_{sc} = b_{sc}$. Clearly, $(z_{sc})_{s \in S, c \in C}$ is a feasible solution of problem (6). Moreover, we have $\sum_{s \in S} \sum_{c \in C} b_{sc} = \sum_{s \in S} \sum_{c \in C} a_{sc} b_{sc} = \sum_{s \in S} \sum_{c \in C} z_{sc} \leq O_2$, Now taking the maximum of the LHS over all feasible solutions of problem (5) we obtain $O_1 \leq O_2$. Hence, we have $O_1 = O_2$. The first part of the proof also shows how to construct an optimal solution of (5) from that of (6). □

Theorem 1. *Problem* (6) *is strongly NP hard.*

The proof of Theorem 1 is provided in Appendix A. It uses a reduction from the 3-partition problem, which is a known NP-hard problem.

IV. Approximation algorithms

Since the joint allocation-matching problem is NP-hard, an efficient algorithm for finding an exact solution is out of reach (unless $P = NP$). We therefore look for allocation algorithms which provide approximate solutions and are easy to implement. Specifically, we consider the following allocation policies:

1) The `greedy` policy: : The `greedy` algorithm computes a feasible replication policy $A \in \mathcal{A}$ as follows: A flow is assigned to each server $s \in S$, each content $c \in C$. These flows are denoted as $\mathsf{flow}(s)$ and $\mathsf{flow}(c)$, respectively. Additionally, each server $s \in S$ is also assigned a degree, denoted as $\mathsf{deg}(s)$. Initially, we set $\mathsf{flow}(s) = U_s$, $\mathsf{deg}(s) = d_s$ for all $s \in S$ and $\mathsf{flow}(c) = \lambda_c$ for all $c \in C$. Then, in each iteration, a pair $(s, c) \in S \times C$ is found for which $\mathsf{flow}(s)\mathsf{deg}(s)\mathsf{flow}(c) > 0$ and which maximizes $\min\left(\mathsf{flow}(s), \mathsf{flow}(c)\right)$. If such a pair (s^*, c^*) is found then the flow of both s^* and c^* are decreased by an amount $\texttt{MatchedFlow} = \min\left(\mathsf{flow}(s^*), \mathsf{flow}(c^*)\right)$ and the degree of s^* is reduced by one. Furthermore, we set $a_{s^*c^*} = 1$ and $z_{s^*c^*} = \texttt{MatchedFlow}$. The iterations continue until no such pair (s^*, c^*) can be found. The pseudocode of the algorithm is given as Algorithm 1.

Clearly, the `greedy` algorithm terminates in at most $m+n$ iterations and returns a feasible replication policy in $\mathcal{A}$. Furthermore, in each iteration, the optimal pair (s^*, c^*) can be found in at most $O(m + n)$ steps. Thus, the worst case time complexity of the `greedy` algorithm is $O((m+n)^2)$. We refer to the allocation policy computed by the greedy algorithm as the `greedy` allocation policy.

Algorithm 1 $\texttt{greedy}(\mathcal{U}, \Lambda, \mathcal{D})$

Inputs: $\mathcal{U} = (U_s, s \in S)$, $\Lambda = (\lambda_c, c \in C)$, $\mathcal{D} = (d_s, s \in S)$
Output: $Z = (z_{sc})$, $A = (a_{sc})$
Initialize: $\mathsf{flow}(s) \leftarrow U_s, \mathsf{deg}(s) \leftarrow d_s, \forall s \in S$; $\mathsf{flow}(c) \leftarrow \lambda_c, \forall c \in C$

1: **while** $\exists (s,c) \in S \times C$ s.t $\mathsf{flow}(s)\mathsf{deg}(s)\mathsf{flow}(c) > 0$ **do**
2: $s^* \leftarrow \arg\max_{s \in S: \mathsf{deg}(s) > 0} (\mathsf{flow}(s))$
3: $c^* \leftarrow \arg\max_{c \in C} (\mathsf{flow}(c))$
4: $\texttt{MatchedFlow} \leftarrow \min(\mathsf{flow}(s^*), \mathsf{flow}(c^*))$
5: $\mathsf{flow}(s^*) \leftarrow \mathsf{flow}(s^*) - \texttt{MatchedFlow}$, $\mathsf{flow}(c^*) \leftarrow \mathsf{flow}(c^*) - \texttt{MatchedFlow}$, $\mathsf{deg}(s^*) \leftarrow \mathsf{deg}(s^*) - 1$
6: $a_{s^*c^*} \leftarrow 1$, $z_{s^*c^*} \leftarrow \texttt{MatchedFlow}$
7: **end while**

We now show that the greedy algorithm always achieves at least $1/2$ of the optimal value of problem (6). To state the result, we denote the instance of problem (6), defined by the vector of bandwidth capacities $\mathcal{U} = (U_s, s \in S)$, vector of arrival rates $\Lambda = (\lambda_c, c \in C)$, and the vector of memory sizes $\mathcal{D} = (d_s, s \in S)$ by $\texttt{JAM}(\mathcal{U}, \Lambda, \mathcal{D})$ and its optimal value by $\texttt{JAM}^*(\mathcal{U}, \Lambda, \mathcal{D})$. The following theorem, whose proof is given in Appendix B, provides a performance guarantee for the greedy algorithm.

Theorem 2. *The output of* greedy *on* $\texttt{JAM}(\mathcal{U}, \Lambda, \mathcal{D})$ *is at least* $\frac{1}{2}\texttt{JAM}^*(\mathcal{U}, \Lambda, \mathcal{D})$

2) Proportional to product (p2p) policy: Under the p2p policy, a server with memory size d_j, $j \in \mathcal{J}$, is allocated all the contents belonging to a set $K \subseteq C$ of size d_j with probability

$$p_{jK} = \frac{1}{Z_j} \prod_{c \in K} \hat{\lambda}_c, \tag{7}$$

independently of all other servers in the system, where $\hat{\lambda}_c = \lambda_c / \sum_{c' \in C} \lambda_{c'}$ denotes the normalized arrival rate of content c and $Z_j = \sum_{K \subseteq C, |K| = d_j} \prod_{c \in K} \hat{\lambda}_c$,. This scheme was proposed in [4]. Unlike the greedy policy, it does not take into account the bandwidths of the servers. Furthermore, it is difficult to provide any performance guarantee for this scheme. Nevertheless, we analyze the dynamics and evaluate the performance of the system under the p2p policy later in the paper.

3) Uniform (unif) policy: As a baseline for comparison, we consider a naive strategy for replication where a server with memory size d_j, $j \in \mathcal{J}$, is populated by all the contents of the set $K \subseteq C$ of size d_j with probability

$$p_{jK} = \frac{1}{\binom{m}{d_j}} \tag{8}$$

Note that (8) does not depend on the particular set K and is the same for all K of the same size. Furthermore, (8) follows from (7) if $\hat{\lambda}_c = 1/m$ for all $c \in C$. Thus, the unif policy treats all contents to be equally popular. It is easy to implement in practice since it does not require the knowledge of the popularities of the contents.

V. Dynamics of the system: Asymptotic optimality

We now analyze the dynamics of the system when the caches have been populated by one of the algorithms discussed in the previous section. To describe the dynamics of the caching system, we need also to specify the matching scheme used to assign each incoming request to one of the servers. In previous works e.g. [4], [5] the maximum matching algorithm has been considered. In this algorithm, an incoming request is accepted if a matching $B = (b_{sc})_{s \in S, c \in C}$ satisfying the constraints (2) and (4) can be found such that $\sum_{s \in S, c \in C} b_{sc}$ equals the total number of requests in the system including the new request. Such a matching, although optimal, is hard to compute in practice and may potentially involve *repacking* or reassignment of the ongoing requests to other available servers. Such repacking would cause undesirable interruptions of service and may lead to increased delay. We therefore look for matching policies which do not involve repacking of the ongoing requests. We present next a simpler matching strategy.

A. Random Available Server (RAS) matching policy

In this policy, each newly arrived content request is assigned to a server chosen uniformly at random from the set of all servers that store the content and are able to serve an additional request of the content. If no such server is available, then the request is blocked. This scheme can be implemented by maintaining a list of available servers for each content at a central job dispatcher that makes the matching decisions. Note that it is not necessary for the central job dispatcher to keep track of the number of jobs in each server. It is sufficient to just keep track of whether a server is operating at its maximum bandwidth capacity or not. This can be achieved by sending a message from a server back to the job dispatcher whenever a job leaves the server previously operating at its maximum bandwidth capacity. Since such updates occur in the background, when a new request arrives, it can be immediately matched to an available server. Unlike the maximum matching algorithm, the RAS scheme does not cause repacking of existing requests in the system. Next, we analyze the dynamics of the system under the RAS matching policy under specific assumptions on the arrival and service time processes for large system sizes.

B. Large system asymptotics

We assume that the requests of each content c arrive according to a Poisson process with rate λ_c, independent of all other processes. Furthermore, the service time of each request is assumed to be exponentially distributed with unit mean.[1] Let $\rho := \sum_{c \in C} \lambda_c / n \sum_{i \in \mathcal{I}, j \in \mathcal{J}} \alpha_{ij} U_i$ denote the load on the system.

We are specifically interested in an asymptotic scaling regime where the number of servers n goes to infinity keeping the load ρ and the proportions α_{ij}, $i \in \mathcal{I}, j \in \mathcal{J}$ fixed. This is achieved by keeping the same catalog C of contents but

[1] We later show numerically that our results do not depend on the type of service time distribution.

scaling the arrival rate of each content linearly with n, i.e., $\lambda_c = n\bar{\lambda}_c$ for all $c \in C$. We define $\bar{\lambda} = \sum_{c\in C} \bar{\lambda}_c$. Note that the normalized arrival rates $\hat{\lambda}_c$ remains the same. This represents a scenario where the system size scales with the demands of the contents.

Before analyzing the dynamics of the caching system, we first determine the fraction of servers in a particular cache configuration under a given allocation policy in the limiting system. Under a given allocation policy, let $q^{(n)}_{ijK}$ denote the fraction of servers having bandwidth U_i ($i \in \mathcal{I}$) and memory size d_j ($j \in \mathcal{J}$) that are storing all the contents belonging to the set $K \subseteq C$ in the nth system. We call these servers to be in *configuration* (i, j, K) and denote the set of all such configurations as $\mathcal{L}$. Define $q_{ijK} := \lim_{n\to\infty} q^{(n)}_{ijK}$ for each $(i, j, K) \in \mathcal{L}$, if the limit exists. Clearly, for the `p2p` and the `unif` allocation policies $q^{(n)}_{ijK} = q_{ijK} = p_{jK}$, where p_{jK} is defined in (7) and (8), respectively. The following lemma shows that q_{ijK} exists for the `greedy` policy and further characterizes it. We first denote by q_{ijc} the value of q_{ijK} for $K = \{c\}$ and define $\theta_c := \sum_{i\in\mathcal{I},j\in\mathcal{J}} \alpha_{ij}U_i q_{ijc}$ to be the normalized capacity allocated to content c. The lemma then states that for $\rho < 1$ the capacity allocated to a content is equal to the arrival rate of the content, and for $\rho > 1$, unpopular contents are allocated zero capacity. The proof of the lemma is given in Appendix C.

Lemma 1. *Under the* `greedy` *allocation policy, we have $q_{ijK} = 0$ for $|K| \geq 2$. Furthermore, for $\rho \leq 1$ we have $\theta_c = \bar{\lambda}_c$, $\forall c \in C$. For $\rho > 1$, we have the following: If the popularities are ordered as $\bar{\lambda}_1 > \bar{\lambda}_2 > \ldots > \bar{\lambda}_m > 0$ and with c^* such that $\frac{\bar{\lambda}}{\rho} \in \left(\sum_{c'=1}^{c^*-1} \bar{\lambda}_{c'} - (c^*-1)\bar{\lambda}_{c^*}, \sum_{c'=1}^{c^*} \bar{\lambda}_{c'} - c^*\bar{\lambda}_{c^*+1}\right]$ then*

$$\theta_c = \begin{cases} \bar{\lambda}_c - \frac{1}{c^*}\left[\sum_{c'=1}^{c^*} \bar{\lambda}_{c'} - \frac{\bar{\lambda}}{\rho}\right] & \text{if } 1 \leq c \leq c^* \\ 0 & \text{if } c^*+1 \leq c \leq m \end{cases} \tag{9}$$

Let $x^{(n)}_{i,j,K,r}(t)$, for $r \in [0, U_i]$, denote the fraction of servers in configuration $(i, j, K) \in \mathcal{L}$ serving at least r requests at time $t \geq 0$. Clearly, the process $x^{(n)} = (x^{(n)}_{i,j,K,r}(t), (i, j, K) \in \mathcal{L}, 0 \leq r \leq U_i, t \geq 0)$ is Markovian and takes values in the set $\mathcal{W}$, where

$$\mathcal{W} := \{w = (w_{i,j,K,r}, (i, j, K) \in \mathcal{L}, r \in \mathbb{Z}_+) : 1 = w_{i,j,K,0} \geq w_{i,j,K,1} \geq \ldots \geq w_{i,j,K,U_i} \geq 0 = w_{i,j,K,r}, \forall r > U_i\}.$$

It is easy to see that according to the RAS policy, the Markov process $x^{(n)}$ jumps from a given state $w \in \mathcal{W}$ to the state $w + e_{i,j,K,r}/n\alpha_{ij}q^{(n)}_{ijK} \in \mathcal{W}$ ($r \geq 1$) with rate

$$\sum_{c\in K} n\bar{\lambda}_c \frac{\alpha_{ij}q^{(n)}_{ijK}(w_{i,j,K,r-1} - w_{i,j,K,r})\mathbb{1}(w_{i,j,K,U_i} < 1)}{\sum_{K':c\in K'}\sum_{i,j}\alpha_{ij}q^{(n)}_{ijK'}(1 - w_{i,j,K',U_i})},$$

where $e_{i,j,K,r}$ denotes the unit vector with unity in position (i, j, K, r). The transition corresponds to an arrival of a request for content $c \in K$. Similarly, the rate of downward transition from $w \in \mathcal{W}$ to $w - e_{i,j,K,r}/n\alpha_{ij}q^{(n)}_{ijK}$ can be computed to be $rn\alpha_{ij}q^{(n)}_{ijK}(w_{i,j,K,r} - w_{i,j,K,r+1})$.

We are interested in the limiting behavior of the process $x^{(n)}$ as $n \to \infty$. We first notice from its transition rates, that $x^{(n)}$ is a *density dependent jump Markov process* [13]–[15] with a limiting ($n \to \infty$) *conditional drift* given by the mapping $h := (h_{i,j,K,r}, (i, j, K) \in \mathcal{L}, r \in \mathbb{Z}_+)$ on $\mathcal{W}$, defined as $h_{i,j,K,r}(w) = 0$ for $r \in \{0\} \cup [U_i + 1, \infty)$ and

$$h_{i,j,K,r}(w) = \sum_{c\in K} \bar{\lambda}_c \frac{(w_{i,j,K,r-1} - w_{i,j,K,r})}{\sum_{K':c\in K'}\sum_{i,j}\alpha_{ij}q_{ijK'}(1 - w_{i,j,K',U_i})} \times \mathbb{1}(w_{i,j,K,U_i} < 1) - r(w_{i,j,K,r} - w_{i,j,K,r+1}), \text{ for } r \geq 1, \tag{10}$$

for each $(i, j, K) \in \mathcal{L}$. For systems in which the limiting drift h is Lipschitz continuous, the classical results of Kurtz [13] imply that the process $x^{(n)}$ converges in distribution to the unique deterministic process $x = (x(t), t \geq 0)$ satisfying $\dot{x} = h(x)$. The process x is called the *mean field limit* or the *fluid limit* of the system. However, since in our case the RHS of (10) has discontinuities (due to the presence of the indicator terms), the solution to $\dot{x} = h(x)$ is not well defined. To overcome this, we define a process x as the solution of a *differential inclusion* (DI) given as $\dot{x} \in H(x)$, where H is set valued mapping on $\mathcal{W}$ defined as the cartesian product of set valued maps $H_{i,j,K,r}$ over all (i, j, K, r). For each $(i, j, K) \in \mathcal{L}$ we define $H_{i,j,K,r}(w) = \{0\}$ for $r \in \{0\} \cup [U_i + 1, \infty)$ and

$$H_{i,j,K,r}(w) = \left[0, \sum_{c\in K} \frac{\bar{\lambda}_c}{\alpha_{ij}q_{ijK}}\right] \mathbb{1}(w_{i,j,K,U_i} = 1) + \sum_{c\in K} \bar{\lambda}_c \frac{(w_{i,j,K,r-1} - w_{i,j,K,r})}{\sum_{K':c\in K'}\sum_{i,j}\alpha_{ij}q_{ijK'}(1 - w_{i,j,K',U_i})} \times \mathbb{1}(w_{i,j,K,U_i} < 1) - r(w_{i,j,K,r} - w_{i,j,K,r+1}), \text{ for } r \geq 1. \tag{11}$$

We note that that the set $H_{i,j,K,r}(w)$ is the convex hull of all the limit points of $h_{i,j,K,r}(w)$. In the next theorem, whose proof is given in Appendix D, we show that the DI $\dot{x} \in H(x)$ has well defined solutions and the process $x^{(n)}$ converges to one of the it solutions $n \to \infty$ in probability.

Theorem 3. *For any $x_0 \in \mathcal{W}$, the set $\mathcal{S}_{x_0}$ of solutions to the DI $\dot{x} \in H(x)$ with $x(0) = x_0$ is non-empty. Furthermore, if $x^{(n)}(0) \xrightarrow{p} x_0 \in \mathcal{X}$ as $n \to \infty$, then for all $T > 0$ we have $\inf_{x\in\mathcal{S}_{x_0}} \sup_{t\in[0,T]} \|x^{(n)}(t) - x(t)\| \xrightarrow{p} 0$ as $n \to \infty$, where the process x is a solution of the DI $\dot{x} \in H(x)$.*

Thus far we have seen that the limiting dynamics of the system for any finite time $t \geq 0$ is described by a solution of the DI $\dot{x} \in H(x)$. We are also interested in the stationary behavior of the limiting system, i.e., the behavior of $x^{(n)}(t)$ as both $n \to \infty$ and $t \to \infty$.[2] Specifically, we are interested in the total number of jobs processed by the system in the stationary regime. Let $Y^{(n)}(t) = n\sum_{i,j,K} \alpha_{ij}q^{(n)}_{ijK}\sum_{r=1}^{U_i} x^{(n)}_{i,j,K,r}(t)$ be the total number of requests in the nth system at time $t \geq 0$ and let $Y^{(n)}(\infty)$ denote its random stationary

[2]For finite n, the system always reaches stationarity because the process $x^{(n)}$ is irreducible on a finite state space.

value. Define $y^{(n)}(t) := Y^{(n)}(t)/n$ for all $t \in [0,\infty]$ and $y(t) := \sum_{i,j,K} \alpha_{ij} q_{ijK} \sum_{r=1}^{U_i} x_{i,j,K,r}(t)$. In the next theorem, whose proof is provided in Appendix E, we show that under the greedy and the p2p combined with the RAS matching policy, $y(\infty) = \lim_{t\to\infty} y(t) = \min\left(\bar{\lambda}, \bar{\lambda}/\rho\right)$ and $\lim_{n\to\infty} y^n(\infty) = y(\infty)$.

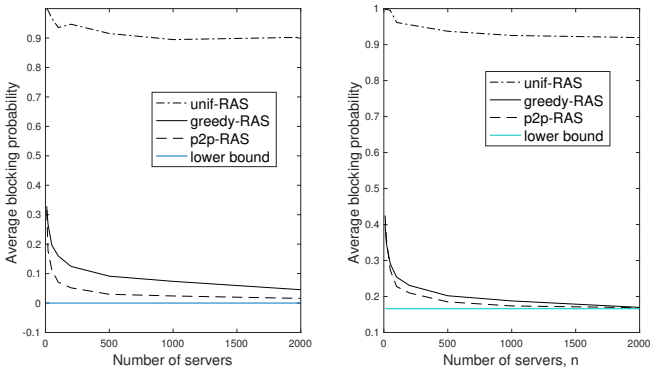
Theorem 4. *Under the greedy allocation policy combined with the RAS matching policy or under the p2p allocation policy with $d_s = 1$ for all $s \in S$ combined with the RAS matching policy, we have $y(\infty) = \lim_{t\to\infty} y(t) = \min(\bar{\lambda}, \bar{\lambda}/\rho)$. Furthermore, the sequence $(y^{(n)}(\infty))_n$ is tight and $y^{(n)}(\infty) \xrightarrow{p} y(\infty)$.*

C. Asymptotic optimality

The above theorem establishes that the p2p and greedy allocation schemes when combined with the RAS matching policy are optimal in the limiting system. To see this, we find an upper bound on $Y^{(n)}(t)$ for any combination of allocation scheme and matching scheme. A trivial upper bound on $Y^{(n)}(t)$ is clearly the total bandwidth capacity of the system, i.e., $Y^{(n)}(t) \leq n\sum_{ij} \alpha_{ij} U_i = n\bar{\lambda}/\rho$ for all $t \in [0,\infty]$. Another upper bound on $Y^{(n)}(\infty)$ can be obtained by comparing the system with an hypothetical caching system in which each server has infinite bandwidth and each content is stored in at least one server. Clearly, this system behaves as an $M/M/\infty$ system serving all incoming requests. Hence, the stationary number of requests $\bar{Y}(\infty)$ in this hypothetical system is a Poisson random variable with mean $n\bar{\lambda}$. By a simple coupling argument, it follows that $Y^{(n)}(\infty) \leq \bar{Y}(\infty)$ almost surely for all n. Thus, combining both upper bounds we have $y^{(n)}(\infty) \leq \min\left(\bar{Y}(\infty)/n, \bar{\lambda}/\rho\right)$. Hence, $\limsup_{n\to\infty} y^{(n)}(\infty) \leq \min\left(\bar{\lambda}, \bar{\lambda}/\rho\right)$. But Theorem 4 shows that for the proposed schemes $\lim_{n\to\infty} y^{(n)}(\infty) = \min\left(\bar{\lambda}, \bar{\lambda}/\rho\right)$. Hence, the proposed schemes are asymptotically optimal. It is easy to see that the corresponding optimal (minimal) blocking probability is given by $(1-\min(1, 1/\rho)) = (1-1/\rho)^+$.

VI. Numerical Results

We now numerically evaluate the performance of the caching system under the various allocation policies combined with the RAS matching policy for finite system sizes. Specifically, we consider a system with $d_s = 2, U_s = 1 \forall s \in S$, $m = 500$. The popularities of the contents are chosen according to a Zipf like distribution [9], [16], where the normalized arrival rate $\hat{\lambda}_c$ of any content $c \in C = \{1, 2, \ldots, m\}$ is chosen to be $\hat{\lambda}_c = c^{-\eta}/\sum_{c'\in C}(c')^{-\eta}$ for $\eta = 2$. The system is simulated for different values of n and ρ for 160000 arrivals. In Figures 1(a) and 1(b), we plot the stationary blocking probability of a request as a function the number of servers for different allocation policies combined with the RAS matching policy for $\rho = 0.8$ and $\rho = 1.2$, respectively. We observe that under both greedy and p2p policies the blocking probability approaches the optimal lower bound $(1-1/\rho)^+$ as n increases. In Table I, we show the difference between the blocking probability of the finite system $P^{(n)}_{\text{blocking}}$ and the optimal lower bound $P_{\text{opt}} = (1-1/\rho)^+$ as a function of the system size n for the p2p algorithm for $\rho = 0.8$. We observe that the distance decreases as $O(n^{-1/2})$. The same is observed for the greedy policy. Such rate of convergence is in accordance with the recent results in the literature on mean field convergence [17].



(a) $\rho = 0.8$ (b) $\rho = 1.2$

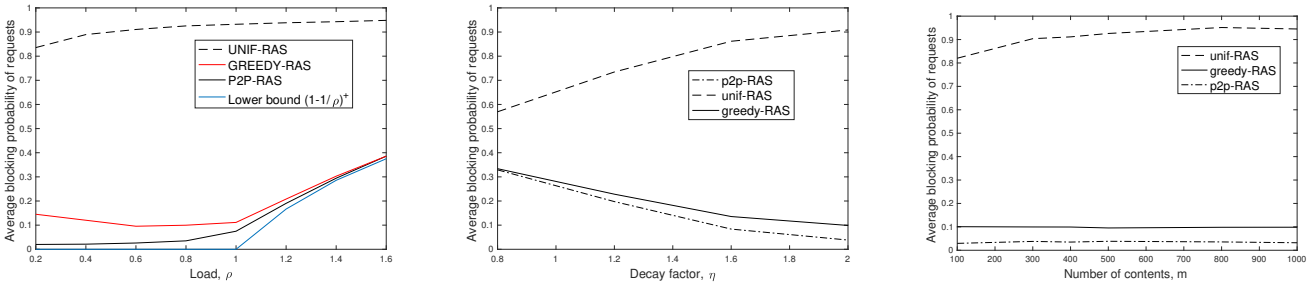
Fig. 1. Average blocking probability as a function of n

TABLE I
Convergence of $P^{(n)}_{\text{blocking}}$

n	$\lvert P^{(n)}_{\text{blocking}} - P_{\text{opt}}\rvert$
10	0.2612
20	0.1837
50	0.1130
200	0.0501
1000	0.0220
2000	0.0148

The blocking probability as a function of the load ρ is shown in Figure 2(a) for $n = 400$. We observe that for the given parameter setting the p2p policy performs better than the greedy policy, but the difference is not significant at high loads. In Figure 2(b), we plot the average blocking probability of requests as a function of the decay factor η of the popularity distribution for $\rho = 0.8$ and $n = 400$. We observe that as η increases and the popularity distribution becomes more skewed towards higher popularity contents, the performance of the unif policy degrades as it still treats all contents to be equally popular. On the other hand, for both greedy and p2p policies the performance improves. This implies that in these policies higher popularity contents are given more priority than lower popularity contents.

Next, we study the sensitivity of the system to the type of distribution of the service times of the requests. For this purpose, we consider the following types of service time distribution with unit mean: Exponential, Constant, Log-normal with probability density function (PDF) given by $f(x) = (1/x\sqrt{2\pi})e^{-(\ln x + 0.5)^2/2}$, and Pareto with PDF given by $f(x) = 10(0.9)^{10}/x^{11}$. For each type of distribution we simulate the system (for 160000 arrivals) with the same parameters as described above under the greedy algorithm



(a) Load (b) Popularity decay factor (c) Number of contents

Fig. 2. Average blocking probability as a function of (a) the load ρ, (b) the popularity decay factor η, and (c) the number of contents m, for the various allocation schemes.

TABLE II
SENSITIVITY TO THE TYPE OF SERVICE TIME DISTRIBUTIONS

(n, ρ)	Exponential	Constant	Lognormal	Pareto
(400, 0.4)	0.1200	0.1219	0.1241	0.1207
(400, 0.8)	0.0989	0.1028	0.1006	0.0999
(400, 1.2)	0.2084	0.2084	0.2110	0.2092
(400, 1.6)	0.3863	0.3854	0.3878	0.3874
(1000, 0.4)	0.0916	0.0912	0.0915	0.0919

(combined with the RAS matching). The blocking probability of requests is tabulated as a function of ρ in Table II for different distributions. We observe that for the same value of ρ the blocking probabilities are nearly the same for all distributions. The values become even more closer when the system size is increased. This seems to suggest the system approaches *insensitivity* as $n \to \infty$. Such asymptotic insensitivity is known to hold for similar systems [18], [19]. However, a proof remains an open problem.

Finally, in Figure 2(c) we plot the average blocking probability of the requests as a function of the number of contents for $\rho = 0.8$, $\eta = 2$, and $N = 400$. We observe that for the `p2p` and the `greedy` policies the average blocking probability remains almost constant with the variation of the number of contents. This is because even though the total number of contents is increasing, only a small fraction of them are highly popular. As a result the addition of more contents does not affect the performance of the system. This also justifies why we scale the arrival rates of the contents instead of the number of contents in Section V.

VII. CONCLUSIONS

We considered the joint problem of content placement and request matching in a distributed network of content servers. We formulated the problem in an optimization framework and showed that it is NP-hard. We then presented a polynomial-time greedy algorithm that approximates to a constant of the optimal value. We then considered a large systems scaling regime, where we showed that a simple greedy matching policy is asymptotically optimal. We employed a new approach based on the theory of differential inclusions to prove the fluid limit results. Many interesting avenues of future work exist. One such challenge is to find the combination of optimal allocation and matching algorithms for systems where both the number of servers and the number of contents scale proportionally to each other.

REFERENCES

[1] Cisco. Cisco visual networking index: Forecast and methodology, 2016–2021. Technical report, Cisco, 2017.

[2] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl. Globally distributed content delivery. *IEEE Internet Computing*, 6(5):50–58, 2002.

[3] F. Kelly. Loss networks. *Ann. Appl. Prob.*, 1(3):319–378, 1991.

[4] B. Tan and L. Massoulié. Optimal content placement for peer-to-peer video-on-demand systems. *IEEE/ACM Transactions on Networking*, 21(2):566–579, April 2013.

[5] M. Leconte, M. Lelarge, and L. Massoulié. Bipartite graph structures for efficient balancing of heterogeneous loads. *SIGMETRICS Perform. Eval. Rev.*, 40(1):41–52, June 2012.

[6] S. Tewari and L. Kleinrock. Proportional replication in peer-to-peer networks. In *Proc. IEEE INFOCOM*, 2006.

[7] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire. Femtocaching: Wireless video content delivery through distributed caching helpers. *IEEE Trans. Inf. Theory*, 59(12), 2013.

[8] M. Dehghan, A. Seetharam, B. Jiang, T. He, T. Salonidis, J. Kurose, D. Towsley, and R. Sitaraman. On the complexity of optimal routing and content caching in heterogeneous networks. In *Proc. IEEE INFOCOM*, 2015.

[9] S. Moharir and N. Karamchandani. Content replication in large distributed caches. arXiv: 1603.09153 [cs.NI].

[10] A. L. Stolyar. Large-scale heterogeneous service systems with general packing constraints. *Adv. Appl. Prob.*, 49(1), 2017.

[11] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire. Femtocaching: Wireless video content delivery through distributed caching helpers. In *Proc. IEEE INFOCOM*, pages 1107–1115, 2012.

[12] S. Moharir, J. Ghaderi, S. Sanghavi, and S. Shakkottai. Serving content with unknown demand: The high-dimensional regime. *SIGMETRICS Perform. Eval. Rev.*, 42(1):435–447, June 2014.

[13] T. G. Kurtz. Solutions of ordinary differential equations as limits of pure jump markov processes. *Journal of Applied Probability*, 7(1):49–58, 1970.

[14] M. Mitzenmacher. *The power of two choices in randomized load balancing*. PhD thesis, University of California at Berkeley, 1996.

[15] A. Mukhopadhyay, A. Karthik, and R. R. Mazumdar. Randomized assignment of jobs to servers in heterogeneous clusters of shared servers for low delay. *Stochastic Systems*, 6(1):90–131, 2016.

[16] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and zipf-like distributions: evidence and implications. In *Proc. IEEE INFOCOM*, 1999.

[17] L Ying. On the approximation error of mean-field models. *SIGMETRICS Perform. Eval. Rev.*, 44(1):285–297, June 2016.

[18] A. Mukhopadhyayay, A. Karthik, R. R. Mazumdar, and F. M. Guillemin. Mean field and propagation of chaos in multi-class heterogeneous loss models. *Perform. Eval.*, 91:117–131, September 2015.

[19] T. Vasantam, A. Mukhopadhyay, and R. R Mazumdar. Insensitivity of the mean-field Limit of Loss Systems Under Power-of-d Routing. arXiv: 1708.09328, August 2017.

[20] N. Gast and B. Gaujal. Mean field limit of non-smooth systems and differential inclusions. *SIGMETRICS Perform. Eval. Rev.*, 38(2):30–32, October 2010.

APPENDIX A
PROOF OF THEOREM 1

We prove this by reducing the 3-partition problem to a decision problem version of (6). This will show that the optimization version of (6) is NP-hard. The 3-partition problem is defined as follows: Given a finite set G of $3n$ elements, a number $L > 0$ and a mapping $\mathsf{size} : G \to \mathbb{R}_{++}$ satisfying $\sum_{g\in G} \mathsf{size}(g) = nL$, does their exist n disjoint subsets $G_1, G_2, \ldots, G_n$ of G, each containing three elements, such that for all $1 \le k \le n$, $\sum_{g\in G_k} \mathsf{size}(g) = L$?

We map each element $g \in G$ to a unique content $c \in C$ with $\lambda_c = \mathsf{size}(g)$. Thus we have $m = 3n$ contents. The sets $G_1, G_2, \ldots, G_n$ correspond to n servers each of which can store $d_s = 3$ contents and simultaneously serve $U_s = L$ requests. Clearly, for the above defined instance of problem (6), the optimal objective function value is bounded above by nL. We now show that the objective function value exactly equals the upper bound nL if and only if there exists a solution of the 3-partition problem.

Suppose that the optimal objective function value of the above defined instance of (6) is nL and it is achieved at $Z^* = (z^*_{sc})_{s\in S, c\in C}$, i.e., $nL = \sum_{s\in S c\in C} z^*_{sc}$. Since for each $s \in S$, $\sum_{c\in C} z^*_{sc} \le L$, and for each $c \in C$, $\sum_{s\in S} z^*_{sc} \le \lambda_c$, we must have $\sum_{c\in C} z^*_{sc} = L$, $\forall s \in S$ and $\sum_{s\in S} z^*_{sc} = \lambda_c > 0$, $\forall c \in C$ Hence, for each content $c \in C$ there must be one server $s \in S$ such that $z^*_{sc} > 0$, i.e., $\sum_{s\in S} \mathbb{1}(z^*_{sc} > 0) \ge 1$. The above implies $\sum_{c\in C}\sum_{s\in S} \mathbb{1}(z^*_{sc} > 0) \ge 3n$. But since every server can store at most three contents, we also have $\sum_{c\in C}\sum_{s\in S} \mathbb{1}(z^*_{sc} > 0) \le 3n$. Hence, we have $\sum_{c\in C}\sum_{s\in S} \mathbb{1}(z^*_{sc} > 0) = 3n$, which implies that $\sum_{s\in S} \mathbb{1}(z^*_{sc} > 0) = 1$, $\forall c \in C$ and $\sum_{c\in C} \mathbb{1}(z^*_{sc} > 0) = 3$, $\forall s \in S$. Hence, a solution of the 3-partition problem is found. The converse follows following the same line of arguments.

APPENDIX B
PROOF OF THEOREM 2

To prove the theorem we first introduce a slightly modified version of problem (6) by adding a constraint which requires that only pairs (s, c) belonging to a given set $\Gamma \subseteq S\times C$ can be assigned a non-zero value of z_{sc}, i.e., $z_{sc} = 0$ for all $(s, c) \notin \Gamma$. This modified problem is denoted as $\texttt{JAM}(\mathcal{U}, \Lambda, \mathcal{D}, \Gamma)$ and its optimal value is denoted as $\texttt{JAM}^*(\mathcal{U}, \Lambda, \mathcal{D}, \Gamma)$. We note that $\texttt{JAM}(\mathcal{U}, \Lambda, \mathcal{D})$ is a special case of $\texttt{JAM}(\mathcal{U}, \Lambda, \mathcal{D}, \Gamma)$ with $\Gamma = S \times C$. Clearly, if $\mathcal{U} \le \mathcal{U}'$, $\Lambda \le \Lambda'$, $\mathcal{D} \le \mathcal{D}'$, $\Gamma \subseteq \Gamma'$, then $\texttt{JAM}^*(\mathcal{U}, \Lambda, \mathcal{D}, \Gamma) \le \texttt{JAM}^*(\mathcal{U}', \Lambda', \mathcal{D}', \Gamma')$. Let $H^* = \{(s, c) \in S \times C : z^*_{sc} > 0\}$ denote the collection of (s, c) pairs that are assigned non-zero values of z^*_{sc} in the optimal solution of $\texttt{JAM}(\mathcal{U}, \Lambda, \mathcal{D})$. Clearly, we have $\texttt{JAM}^*(\mathcal{U}, \Lambda, \mathcal{D}, H^*) = \texttt{JAM}^*(\mathcal{U}, \Lambda, \mathcal{D})$.

We now generate a sequence of tuples $(\mathcal{U}_k, \Lambda_k, \mathcal{D}_k, H^*_k, G_k)$ for each iteration $k \ge 0$ of the `greedy` algorithm (applied on $\texttt{JAM}(\mathcal{U}, \Lambda, \mathcal{D})$) as follows: For $k = 0$ we set $\mathcal{U}_0 = \mathcal{U}, \Lambda_0 = \Lambda, \mathcal{D}_0 = \mathcal{D}, H^*_0 = H^*$, $G_0 = S \times C$. For subsequent values of k, let $f_k (= \texttt{MatchedFlow})$ denote the flow found by the `greedy` algorithm in its k^{th} iteration and let (s_k, c_k) $(= (s^*, c^*))$ denote the corresponding server-content pair. We set $G_{k+1} = G_k - (s_k, c_k)$, $mcU_{k+1} = \mathcal{U}_k - f_k e^{(n)}_{s_k}$, $\Lambda_{k+1} = \Lambda_k - f_k e^{(m)}_{c_k}$, $\mathcal{D}_{k+1} = \mathcal{D}_k - e^{(n)}_{s_k}$, and $H^*_{k+1} = H^*_k - (s_k, c_k)$ where $e^{(l)}_r$ denotes the standard l-dimensional unit vector having one in the r^{th} component.

Note that it may so happen that $(s_k, c_k) \notin H^*_k$ for some k (but (s_k, c_k) will always be in G_k). In such cases, the above update sets $H^*_{k+1} = H^*_k$. Hence, $H^*_k \subseteq G_k$ is maintained for all k. Therefore, we have $\texttt{JAM}^*(\mathcal{U}_k, \Lambda_k, \mathcal{D}_k, H^*_k) \le \texttt{JAM}^*(\mathcal{U}_k, \Lambda_k, \mathcal{D}_k, G_k)$. If the `greedy` algorithm terminates in the q^{th} iteration, then we must have $f_q = 0$ and $\texttt{JAM}^*(\mathcal{U}_q, \Lambda_q, D_q, G_q) = 0$. Therefore, $\texttt{JAM}^*(\mathcal{U}_q, \Lambda_q, \mathcal{D}_q, H^*_q) = 0$. Furthermore, for each iteration k we prove that the following inequality

$$\texttt{JAM}^*(\mathcal{U}_k, \Lambda, \mathcal{D}_k, H^*_k) - \texttt{JAM}^*(\mathcal{U}_{k+1}, \Lambda_{k+1}, \mathcal{D}_{k+1}, H^*_{k+1}) \le 2f_k, \quad (12)$$

holds. To see the above, first let us consider the case when $(s_k, c_k) \in H^*_k$. Hence, $(s_k, c_k) \notin H^*_{k+1}$. Let $v \ge 0$ be the flow assigned to the (s_k, c_k) pair in the optimal solution of $\texttt{JAM}(\mathcal{U}_k, \Lambda_k, \mathcal{D}_k, H^*_k)$. We have $\texttt{JAM}^*(\mathcal{U}_k, \Lambda_k, \mathcal{D}_k, H^*_k) = v + \texttt{JAM}^*(\mathcal{U}_k - ve^{(n)}_{s_k}, \Lambda_k - ve^{(m)}_{c_k}, \mathcal{D}_{k+1}, H^*_{k+1}) \le v + \texttt{JAM}^*(\mathcal{U}_{k+1}, \Lambda_{k+1}, \mathcal{D}_{k+1}, H^*_{k+1}) + 2(f_k - v)$. The last inequality holds since no more than $2(f_k - v) \ge 0$ additional flow can be matched under constraints given by the tuple $(\mathcal{U}_k - ve^{(n)}_{s_k}, \Lambda_k - ve^{(m)}_{c_k}, \mathcal{D}_{k+1}, H^*_{k+1})$ as compared to constrains given by the tuple $(\mathcal{U}_{k+1}, \Lambda_{k+1}, \mathcal{D}_{k+1}, H^*_{k+1})$. Inequality (12) hence follows for $(s_k, c_k) \in H^*_k$. Now consider the case when $(s_k, c_k) \notin H^*_k$. Hence, $H^*_k = H^*_{k+1}$. Clearly, no more than $2f_k$ additional flow can be matched under the constraints given by $(\mathcal{U}_k, \Lambda_k, \mathcal{D}_k, H^*_k)$ as compared to the constraints given by $(\mathcal{U}_{k+1}, \Lambda_{k+1}, \mathcal{D}_{k+1}, H^*_{k+1})$. Hence, (12) holds in this case also. Summing (12) for $k = 0, 1, \ldots, q-1$ we obtain $\texttt{JAM}^*(\mathcal{U}_0, \Lambda_0, \mathcal{D}_0, H^*_0) - \texttt{JAM}^*(\mathcal{U}_q, \Lambda_q, \mathcal{D}_q, H^*_q) \le 2\sum_{k=0}^{q-1} f_k$. This completes the proof since $\sum_{k=0}^{q-1} f_k$ is the output of the `greedy` algorithm, $\texttt{JAM}^*(\mathcal{U}_0, \Lambda_0, \mathcal{D}_0, H^*_0) = \texttt{JAM}^*(\mathcal{U}, \Lambda, \mathcal{D})$, and $\texttt{JAM}^*(\mathcal{U}_q, \Lambda_q, \mathcal{D}_q, H^*_q) = 0$. □

APPENDIX C
PROOF OF LEMMA 1

First we note that for sufficiently large n, we have $n\bar{\lambda}_c > U_i$ for all $c \in C$ and all $i \in \mathcal{I}$. Hence, to allocate more than one content to the cache of a server the `greedy` algorithm must reach a stage where the remaining flow of each content is less than $U_{\max} := \max_{i\in\mathcal{I}} U_i$. Since at least $U_{\min} := \min_{i\in\mathcal{I}} U_i$ flow can be matched to each server, the maximum number of servers which can be assigned a non-zero flow after this stage is $mU_{\max}/U_{\min}$. These are the only servers which can be allocated more than one contents. Therefore, the fraction of servers assigned more than two contents is at most $mU_{\max}/nU_{\min}$ which approaches zero as $n \to \infty$. This shows

that the probability of a server storing more than one content approaches zero as $n \to \infty$, i.e., $q_K^{(ij)} = 0$ for $|K| \geq 2$.

Next, consider the case $\rho \leq 1$. Again, for sufficiently large n, we have $n\bar{\lambda}_c > U_{\max}$ for all $c \in C$. In this case, the `greedy` algorithm cannot terminate before the remaining flow for each content becomes less than or equal to $U_{\max}$. To prove this, let us assume the converse, i.e., the greedy algorithm terminates when the remaining flows for some contents are still strictly above $U_{\max}$. This implies that the `greedy` algorithm terminated because the remaining flows of each server has become zero. Clearly, for this to happen we must have $n\bar{\lambda} > n\sum_{i,j} \alpha_{ij} U_i$, i.e., $\rho > 1$, which is a contradiction. Therefore, the `greedy` algorithm terminates with less than $U_{\max}$ remaining flow for each content. Thus, the fraction of the total flow $n\bar{\lambda}$ which remains unmatched is at most $\frac{mU}{n\bar{\lambda}}$, which approaches to zero as $n \to \infty$. Hence, in the limiting system, the whole flow $n\bar{\lambda}_c$ of each content $c \in C$ is matched. Since $n\theta_c$ denotes the total flow of content c assigned to all the servers combined, we must have $\theta_c = \bar{\lambda}_c$.

For $\rho > 1$, it is easy to see that the `greedy` algorithm terminates when remaining flows of all the servers become zero. Furthermore, at termination, all the contents, which have been chosen at least once by the algorithm in some iteration, have the same remaining flow. Let this flow be equal to f and let the contents chosen by the algorithm at least once be $c = 1, 2, \ldots, c^*$ for some $c^* \leq m$. Then, we must have $n\bar{\lambda}_{c^*+1} \leq f < n\bar{\lambda}_{c^*}$. Also, since the total matched flow $n\sum_{c'=1}^{c^*} \bar{\lambda}_{c'} - kf$ combining all the contents is equal to the total capacity $\bar{\lambda}/\rho$ of the system, we have $f = \frac{n}{c^*}\left(\sum_{c'=1}^{c^*} \bar{\lambda}_{c'} - \frac{\bar{\lambda}}{\rho}\right)$. The matched flow for each content $c = 1, 2, \ldots, c^*$ is therefore $n\theta_c = n\bar{\lambda}_c - f$ and for each content $c > c^*$ is $\theta_c = 0$. This completes the proof of the lemma. □

Appendix D
Proof of Theorem 3

We recall from Theorem 1 of [20] that the DI $\dot{x} \in H(x)$ has at least one solution x with $x(0) = x_0 \in \mathcal{W}$ if 1) for each $w \in \mathcal{W}$ the set $H(w)$ is non-empty, closed, convex; 2) $\|H(w)\| := \sup\{\|z\|_2 : z \in H(w)\} < D(1 + \|w\|)$ for some constant $D > 0$; 3) H is upper semi-continuous[3]. Furthermore, the density dependent Markov process $x^{(n)}$ with limiting drift h converges in probability to the solution of the DI $\dot{x} \in H(x)$ if $H(w) = \mathsf{conv}\left(\mathsf{acc}_{w_k \to w} h(w_k)\right)$, where $\mathsf{conv}(V)$ denotes the closure of convex hull containing the set V and $(\mathsf{acc}_{w_k \to w} h(w_k))$ denotes the set of accumulation points of the sequence $(h(w_k))$ for $w_k \to w$.

From (11), it follows directly that $H(w)$ is nonempty, closed and convex for each $w \in \mathcal{W}$. Furthermore, for each $(i, j, K) \in \mathcal{L}$ and $r \in \mathbb{Z}_+$ we have $0 \leq \bar{\lambda}_c \frac{(w_{i,j,K,r-1} - w_{i,j,K,r})}{\sum_{K': c \in K'} \sum_{i,j} \alpha_{ij} q_{ijK'} (1 - w_{i,j,K',U_i})} \leq \bar{\lambda}_c / \alpha_{ij} q_{ijK}$. Hence, from (11) we have

$$H_{i,j,K,r}(w) \leq D_{i,j,K} := \sum_{c \in K} \frac{\bar{\lambda}_c}{\alpha_{ij} q_{ijK}} + \max_{i \in \mathcal{I}} U_i$$

Therefore, $\|H(w)\| \leq D \leq D(1 + \|w\|_2)$, where $D := \sqrt{\sum_{(i,j,K) \in \mathcal{L}} (1 + U_i D_{i,j,K}^2)} > 0$. We also note that $H_{i,j,K,r}(w)$ is continuous if $w_{i,j,K,U_i} < 1$ and the compact set $[0, \bar{\lambda}_c / \alpha_{ij} q_{ijK}]$ contains all limit points of $\bar{\lambda}_c \frac{(w_{i,j,K,r-1} - w_{i,j,K,r})}{\sum_{K': c \in K'} \sum_{i,j} \alpha_{ij} q_{ijK'} (1 - w_{i,j,K',U_i})}$ as $w_{i,j,K,U_i} \to 1$. Hence, the set valued mapping H is upper-hemicontinuous. By definition it follows that $H(w) = \mathsf{conv}\left(\mathsf{acc}_{w_k \to w} h(w_k)\right)$. Therefore, the statement of the theorem follows from Theorem 1 of [20]. □

Appendix E
Proof of Theorem 4

Let $y_c(t) := \sum_{K: c \in K} \sum_{i,j} \alpha_{ij} q_{ijK} \sum_{r=1}^{U_i} x_{i,j,K,r}(t)$ for all $t \geq 0$. For the `greedy` algorithm, we have from Lemma 1 that $q_{ijK} = 0$ for $|K| > 1$. Hence, we have $y(t) = \sum_{c \in C} y_c(t)$. Further, in (11) substituting $K = \{c\}$ and summing we obtain

$$\dot{y}_c(t) \in H_c(y_c) := [0, \bar{\lambda}_c] \mathbb{1}(y_c = \theta_c) + \bar{\lambda}_c \mathbb{1}(y_c < \theta_c) - y_c. \tag{13}$$

It can be easily verified that $(z_1 - z_2)(w_1 - w_2) \leq 0$ for all $w_1, w_2 \in \mathbb{R}$ and for all $z_1 \in H_c(w_1), z_2 \in H_c(w_2)$. In other words, H_c is one-sided Lipschitz (OSL) with Lipschitz constant $L = 0$. Therefore, the DI (13) has a unique solution. It can also be verified that for any $y(0) = y_0 \in [0, \theta_c]$, $y_c(t) = \tilde{y}_c(t) \mathbb{1}(\tilde{y}_c(t) < \theta_c) + \theta_c \mathbb{1}(\tilde{y}_c(t) \geq \theta_c)$, where $\tilde{y}_c(t) = \bar{\lambda}_c + (y_0 - \bar{\lambda}_c) e^{-t}$ is a solution of the DI (13). Hence, it must be the only solution. Since from Lemma 1 we have for $\rho \leq 1$, $\theta_c = \bar{\lambda}_c$ for all $c \in C$, it follows that $y_c(\infty) = \lim_{t \to \infty} y_c(t) = \bar{\lambda}_c$, or $y(\infty) = \bar{\lambda}$. For $\rho > 1$ again from Lemma 1 we have that $\theta_c < \bar{\lambda}_c$ for all $c = 1 : c^*$ and $\theta_c = 0$ for $c = c^* + 1 : m$. Hence, $y_c(\infty) = \theta_c$ for $c = 1 : c^*$ and $y_c(\infty) = 0$ for $c = c^* + 1 : m$. Thus, $y(\infty) = \sum_{c \in C} y_c(\infty) = \sum_{c=1}^{c^*} \theta_c = \bar{\lambda}/\rho$.

Now, we consider the `p2p` algorithm with $d_s = 1$ for all $s \in S$. For this case we have $q_{ijc} = \hat{\lambda}_c$ for all $c \in C$ and $q_{ijK} = 0$ if $|K| > 1$. Hence, as before we have $y(t) = \sum_{c \in C} y_c(t)$ and $y_c(t) = \tilde{y}_c(t) \mathbb{1}(\tilde{y}_c(t) < \theta_c) + \theta_c \mathbb{1}(\tilde{y}_c(t) \geq \theta_c)$. In this case, $\theta_c = \bar{\lambda}_c/\rho$. We thus have, for $\rho \leq 1$, $\bar{\lambda}_c \leq \theta_c$ and for $\rho \leq 1$ $\theta_c < \lambda_c$. Hence, $y_c(\infty) = \lim_{t \to \infty} y_c(t) = \lambda_c$ for $\rho \leq 1$ and $y_c(\infty) = \theta_c = \lambda_c/\rho$ for $\rho > 1$. Thus, $y(\infty) = \min\left(\bar{\lambda}, \bar{\lambda}/\rho\right)$.

Since $y^{(n)}(\infty) \leq \bar{\lambda}/\rho$ uniformly for all n, it follows that the sequence $(y^{(n)}(\infty))_n$ is tight. From Theorem 3 and the uniqueness of solution of $y(t)$ we have that $y^{(n)} \to y$. Hence, every limit point of the sequence of stationary measures of $y^{(n)}$ must be an invariant measure of the process y. Since $y(\infty)$ is the unique, globally asymptotically stable stationary point of the process y it follows that the only invariant measure for the process y is the Dirac measure concentrated at $y(\infty)$. Thus, all limit points of the sequence of stationary measures of $y^{(n)}$ must coincide with the Dirac measure at $y(\infty)$, i.e., $\lim_{n \to \infty} y^{(n)}(\infty) = y(\infty)$. □

[3]The set valued mapping F is said to be upper-hemicontinuous at w if $\forall w_n, \forall z_n \in F(w_n)$, $\lim_{n \to \infty} w_n = w$ and $\lim_{n \to \infty} z_n = z$ implies $z \in H(w)$.