Mode-Suppression: A Simple, Stable and Scalable Chunk-Sharing Algorithm for P2P Networks

Vamseedhar Reddyvari, *Member, IEEE*, Sarat Chandra Bobbili, Parimal Parag, *Senior Member, IEEE*, and Srinivas Shakkottai, *Senior Member, IEEE*

***Abstract*— The ability of a P2P network to scale its throughput up in proportion to the arrival rate of peers has recently been shown to be crucially dependent on the chunk sharing policy employed. Some policies can result in low frequencies of a particular chunk, known as the missing chunk syndrome, which can dramatically reduce throughput and lead to instability of the system. For instance, commonly used policies that nominally "boost" the sharing of infrequent chunks such as the well-known rarest-first algorithm have been shown to be unstable. We take a complementary viewpoint, and instead consider a policy that simply prevents the sharing of the most frequent chunk(s), that we call mode-suppression. We also consider a more general version that suppresses the mode only if the mode frequency is larger than the lowest frequency by a fixed threshold. We prove the stability of mode-suppression using Lyapunov techniques, and use a Kingman bound argument to show that the total download time does not increase with peer arrival rate. We then design versions of mode-suppression that sample a small number of peers at each time, and construct noisy mode estimates by aggregating these samples over time. We show numerically that mode suppression stabilizes and outperforms all other recently proposed chunk sharing algorithms, and via integration into BitTorrent implementation operating over the ns-3 that it ensures stable, low sojourn time operation in a real-world setting.**

***Index Terms*— Peer-to-peer networks, Lyapunov stability, chunk selection policy, Kingman bound.**

Manuscript received July 24, 2020; revised March 29, 2021 and June 8, 2021; accepted June 21, 2021; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor R. La. Date of publication July 5, 2021; date of current version December 17, 2021. The work of Vamseedhar Reddyvari and Srinivas Shakkottai was supported in part by NSF under Grant CNS-Intel 1719384, Grant CPS-2038963, and Grant CNS-1955696; and in part by Army Research Office (ARO) under Grant W911NF-19-1-0367, Grant W911NF-19-2-0243, and Grant W911NF-21-2-0064. The work of Sarat Chandra Bobbili and Parimal Parag was supported in part by the Science and Engineering Research Board (SERB) under Grant DSTO-1677, in part by the Department of Telecommunications, Government of India, under Grant DOTC-0001, in part by the Centre for Networked Intelligence (a Cisco corporate social responsibility (CSR) initiative) at IISc, Bengaluru, and in part by the Robert Bosch Centre for Cyber-Physical Systems at IISc. *(Corresponding author: Parimal Parag.)*

Vamseedhar Reddyvari was with the Texas A&M University, College Station, TX 77843 USA. He is now with the Google LLC, Sunnyvale, CA 94089 USA (e-mail: vamseedhar.reddyvari@gmail.com).

Sarat Chandra Bobbili was with the Indian Institute of Science, Bengaluru, Karnataka 560012, India. He is now with the Qualcomm India Pvt. Ltd., Hyderabad, Telangana 500081, India (e-mail: saratbobbili@iisc.ac.in).

Parimal Parag is with the Department of Electrical Communication Engineering, Indian Institute of Science, Bengaluru, Karnataka 560012, India (e-mail: parimal@iisc.ac.in).

Srinivas Shakkottai is with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843 USA (e-mail: sshakkot@tamu.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TNET.2021.3092008, provided by the authors.

Digital Object Identifier 10.1109/TNET.2021.3092008

I. INTRODUCTION

PEER-TO-PEER (P2P) file sharing networks such as BitTorrent [1] have been studied intensely—using analytical models, simulation studies, and large scale field experiments. After seeing much initial usage, declines of P2P applications were seen as cloud-based HTTP streaming services became available. However, with increasing fragmentation of content libraries over a plethora of paid content streaming services, yearly increases in P2P usage are being seen again. Increases in BitTorrent usage are measurable in the Asia-Pacific (APAC) and Europe-Middle-East-Africa (EMEA) regions, while it is likely that such applications are moving behind VPNs in the Americas according to a 2019 report from Sandvine [2]. Measurements from this study comparing BitTorrent and HTTP streaming, shown in Table I indicate robust BitTorrent usage.

Interest in the analysis of BitTorrent stems from a desire to understand the thought-provoking phenomenon of apparent scaling up of throughput as the number of peers grows, which enables it to effectively distribute content with low file-download times during high demand situations called *flash-crowds*. This scaling is related to the fact that a file is divided into fixed-size chunks, and a peer possessing a set of chunks can upload those chunks to other peers that need them. Once a peer has downloaded all chunks, it could continue to serve other peers or leave the system. A so-called *seed server* that possesses all chunks and never leaves is often used to ensure that no particular chunk ever goes missing. It is the feature of integrating the upload capacity of each peer into the system that is supposed to enable system-wide throughput scaling up with the number of peers. However, since peers can only share chunks that they possess, it is crucial to ensure the wide availability of all chunks to enable maximum usage of available upload capacity with each peer.

The problem of ensuring that all chunks are easily obtainable—ideally by engendering equal numbers of copies of each chunk over the network—was considered by the original designers of P2P networks. For example, BitTorrent, which is the most popular P2P network protocol, uses an algorithm called *rarest-first* (RF) to try to achieve this goal [1]. Here, the idea is to keep a running estimate of the frequency of all chunks in the system. When a peer has a chance to download a chunk, it chooses the least frequent (i.e., the "rarest") among all the chunks that it needs. In practice, peers keep track of the frequency of chunks in local subsets. Intuition suggests that such "boosting" of rare chunks might ensure a near-uniform empirical distribution of chunks.

TABLE I
TRAFFIC FRACTIONS OF BITTORRENT VS. HTTP STREAMING, 2019 [2]

	BitTorrent		HTTP Streaming	
	Upload	Download	Upload	Download
APAC	24.8%	4.5%	4.8%	10.9%
EMEA	44.2%	5.3%	10.6%	16.4%

Recent work has postulated that under some conditions, the rarest-first policy used by BitTorrent actually does not achieve its goal, and can actually be harmful to system performance. In particular, [3] studied a chunk-level model of P2P sharing under which new peers that do not possess any chunks arrive into the system at some rate, contacts between peers happen at random, and at each contact a chunk is transferred to a requesting peer under a given policy. Peers depart immediately after completing the file download. The objective was to determine if the system is *stable* under a given policy, i.e., at any time is the number of peers that have not yet received the whole file finite or is it exploding to infinity? The result was that under several policies including rarest-first and random chunk selection, a particular chunk can become very rare across the network—a phenomenon referred to as the *missing chunk syndrome*. This causes the creation of a large set of peers that are missing only that one chunk, referred to as the *one club*. In turn, the seed server must serve the missing chunk to almost all peers (which then depart), which means that the system is unstable unless the upload capacity of the seed server is of the order of the arrival rate of peers into the system. Thus, the phenomenon largely negates the value of the P2P system.

More recently, experimental studies have revealed that the missing piece syndrome is an observable phenomenon occurring in BitTorrent networks [4]. The results show that when the seed server has low or intermittent upload capacity, the throughput of the system saturates as the number of peers grows. In turn, this causes lengthened stay of peers in the system between arrival and completion, where an increasingly large number of peers are waiting to obtain the final chunk before leaving. In other words, designing policies that can ensure stability of a P2P network under a fixed seed server capacity for all peer arrival rates is practically relevant.

A. Related Work

There has been extensive work on P2P applications over the past few years. There are many examples, including but not limited to, content distribution [5], [6], crowd sensing [7], [8], energy trading [9]–[11], video streaming [12], [13] and models for peer selection [14]–[16]. In addition, with blockchain technology gaining a lot of traction, several works have focused on leveraging the distributed nature of P2P networked systems in its support [17]–[19].

Our specific problem relates to BitTorrent stability, and we refer below only to those directly relating to the scaling properties of a single swarm. A large system assumption was made in [20]–[22], and the evolution of peers and seeds is described using a system of differential equations. While [20], [21] study the stationary regime and indicate the stability of BitTorrent-like systems for all arrival rates, [22] considers the transient regime and studies how much seed server capacity is needed to attain a target sojourn time (the time between the arrival of a peer and its completing the file download). Results on stability and scaling here require that at least a fixed fraction of the peers' upload capacity can always be utilized—an implicit assumption of chunk availability. As shown in [3], this assumption need not hold for all chunk selection policies, and a chunk-level model is needed for accurate analysis.

Chunk-level models have considered the missing chunk problem from two angles. The first method is to explicitly insist that peers that have completed the download should stay in the system as servers for some period of time. For example, [23] presents results on fairness vs. system performance based on how long peers stay after completion. In a more recent work [24], it was analytically shown that the system is stable as long as peers stay long enough to serve of the order of one additional chunk after completion. Indeed, in the first BitTorrent implementation this often happened naturally, since most users manually stopped participation at some point after download was completed. Enforcing cooperation among peers to alleviate this issue has been proposed, typically using a specific initialization or an application-specific implementation [25]–[29]. However, the ability to leave after download has been available in multiple BitTorrent implementations for over fifteen years as a simple checkbox option. Given that users in many countries are subject to upload and download quotas, strategic user behavior by using this option suggests that the instability observed in [4] could easily affect system performance.

The second method is to assume that peers would leave immediately after completion, and to design the chunk sharing policy such that the missing chunk syndrome is avoided. Some algorithms of this nature are "boosting" policies that can be thought of as modified versions of rarest-first. For example, the *rare chunk* (RC) algorithm studied in [30]–[32] picks three peers at random and chooses a chunk that is available with exactly one of the selected peers (called a "rare" chunk). Also studied in [32] is a variant of this algorithm called the *common chunk* (CC) algorithm, which proceeds as in the RC algorithm when the peer has no chunks, then follows a policy of sampling a single peer with random selection among its required chunks until it only needs one more chunk, and then proceeds by sampling three peers and only downloading a chunk if every chunk with it appears at least twice with the sampled peers. However, although stable, these algorithms appear to have long sojourn times in some settings [33].

More recent work on chunk sharing policies [33] describes an algorithm called *group suppression* (GS), which is based on observations made in [3]. The policy is based on computing the empirical distribution of the states in the system, where a state of a peer is the set of chunks available with that peer. Peers that belong to the state with highest frequency are not allowed to upload chunks to peers that have fewer chunks than themselves, thus suppressing entry into the highest frequency group. Although this policy appears to have low mean sojourn times in simulations, it can have high variability. Also, this policy is complex since it requires the knowledge of the entire empirical state distribution. Furthermore, the authors are only

TABLE II
COMPARISON OF CHUNK SELECTION POLICIES

Policy	m = 2	m > 2	Information	Sojourn time	
				Choose from 1 Peer	*Choose from 3 Peers*
Random	Unstable	Unstable	None	N/A (unstable)	N/A (unstable)
Rarest-First (RF)	Unstable	Unstable	Chunk Frequency	N/A (unstable)	N/A (unstable)
Rare Chunk (RC)	Stable	Stable	3 Peers	Bad	Good
Common Chunk (CC)	Stable	Stable	3 Peers	Good	Bad
Group Suppression (GS)	Stable	Unknown	Complete Distribution	Good	Better
Mode-Suppression (MS T=1)	Stable	Stable	Chunk Frequency	Good	Better
Mode-Suppression (MS T=2m)	Stable	Stable	Chunk Frequency	Best	Best
Local Mode-Suppression (LMS)	Stable	Unknown	3 Peers	Better	Best
EWMA Mode-Suppression (MS-EWMA)	Unknown	Unknown	1 Peer	Better	Best

able to prove stability in a P2P network with exactly 2 chunks, while the stability of the general case is left as a conjecture.

B. Main Results

The nominal objective of Rarest-First is to ensure a uniform chunk distribution across the network boosting low-frequency chunks, which it actually does not achieve in all cases, causing instability as shown in [3]. To support rarest-first, BitTorrent implements a distributed algorithm for determining global chunk frequencies so that peers can determine which chunks are rarest. Our idea is to utilize the same chunk frequency information, but to simply prevent the most frequent chunk(s) from being shared, hence allowing less frequent chunks to catch up and drive the empirical distribution of chunks towards the desired uniform distribution. Implicitly, this would also remove a small fraction of the upload capacity, keeping peers in the system a little longer, and enabling them to share more copies of rare chunks.

Following this intuition, we propose a policy that we call *mode-suppression* (MS), which is based on terminology used in statistics in which the *mode* is the most frequent value(s) in a data set. In the basic version of this algorithm, we keep track of the frequency of chunks in the system, and when a peer contacts another peer, it is allowed to download any chunk except the one(s) belonging to the mode. Any chunk may be downloaded if all chunks are equally frequent (i.e., if all chunks belong to the mode). We extend this idea to a more general version where we do not insist on always suppressing the mode, but only do so when the highest frequency is greater than the lowest frequency by a fixed threshold.

We have two main analytical results under a random peer selection model. First, we show using a Lyapunov drift analysis that the general version of mode-suppression with any finite frequency difference threshold is *stabilizing under all peer arrival rates* in a system in which the *file is divided into any number of chunks*. Second, we show using Kingman bound arguments that for the general version of mode-suppression, *the sojourn time does not increase with peer arrival rate.* Hence, mode-suppression appears to be able to reduce chunk sharing just enough to maintain stability, without negatively affecting the sojourn time in a scaling sense.

We also construct two heuristic variants of the idea that only depend on limited observations. The first variant is mode-suppression that samples only one peer at a time and uses the history of interactions to compute a noisy mode, based on an exponentially weighted moving average estimate of chunk frequency (MS-EWMA). The second variant, *local mode-suppression* (LMS) samples 3 peers at a time, and uses a noisy mode constructed from only those samples. It is straightforward to show that LMS is stabilizing in the case of a system with two chunks following the proof in [32]. We primarily study these heuristic variants via simulations.

We simulate all the algorithms by starting the system in a corner case where one of the chunks is available only at the seed server, and observe the evolution of the system afterwards. An additional dimension that we explore is the impact on chunk diversity engendered by being able to pick a chunk from the set possessed across multiple peers, i.e., choice of one chunk from one randomly chosen peer, versus choice of one chunk from the chunk-set of three randomly chosen peers. We empirically find that MS attains its lowest sojourn time when we set the frequency difference threshold for suppression $T = 2m$, where m is the number of chunks that the file is divided into. We also find that the variants of MS preformed the best overall, and the case of choosing a chunk from the chunk-set of 3 random peers is near-optimal in terms of sojourn time. A comparison is presented in Table II.

We then replace rarest-first with mode-suppression in a BitTorrent implementation over the ns-3 simulation environment. This is straightforward, since mode-suppression uses exactly the same chunk frequency statistics as rarest-first, and hence can be integrated into BiTorrent with about 20 lines of code. The ns-3 experiments recover earlier instability issues of rarest-first, demonstrate that mode-suppression is stabilizing, and that it results in good sojourn time performance.

A preliminary version of this work was presented in [34], which only considered the stability of a basic version of mode-suppression. The current work derives a stability result for a generalized version of mode-suppression that has a frequency difference threshold, empirically determines the right threshold, and develops a Kingman-bound-based sojourn time scaling result. In this work, we also present an implementation of the main algorithm using BitTorrent over ns-3. It thus generalizes and adds to the methodological contributions, as well as to the empirical study.

II. SYSTEM MODEL

We consider a P2P file sharing system for a single file divided into m chunks. This file sharing system has a unique seed that has all m chunks, and the seed stays in the system indefinitely. Peers arrive according to a Poisson process with rate λ. Each incoming peer arrives without any chunks and

stays in the system till it obtains all m chunks of the file. In this model, a peer leaves as soon as it has all m chunks of the file. The peers can receive the chunks in two ways, either directly from the seed or from other peers.

A contact is said to occur whenever a seed or a peer contacts another peer. Hence, each peer and the seed have individual contact processes corresponding to the sequence of contact instants. Upon contact, the seed or the peer immediately transfers a missing chunk to the contacted peer, according to a *chunk selection policy*. When chunk selection policy depends solely on the current state of the system, it is said to be Markovian.

A. Contact Processes

The time interval between two contacts are assumed to be random, independent, and identically exponentially distributed, i.e., all contact processes are independent and Poisson. The Poisson contact rate for the seed is assumed to be U, and each peer is assumed to have a common contact rate of μ. We have also assumed that the transfer of chunk is immediate on the contact. In practice, contact times are small, and download times are significant. We have assumed random contact time and instantaneous download, for the ease of exposition. Our model is equivalent to random download time and instantaneous contact, where peers immediately contact once they have finished downloading the chunk. Effectively, the time interval between two contacts is modeling the chunk download time.

B. State Space

At any time t, the number of peers in the system with a proper subset of chunks $S \subset [m]$ is denoted by $X_S(t) \in \mathbb{N}_0 \triangleq \{0, 1, \dots\}$. The system state at time t can be represented by

$$X(t) = (X_S(t) : S \subset [m]).$$

The total number of peers at any time t is denoted by

$$|X(t)| = \sum_{S \subset [m]} X_S(t).$$

For any Markov chunk selection policy, the continuous time process $(X(t), t \geqslant 0)$ is Markov with countable state space $\mathcal{X} \triangleq \mathbb{N}_0^{\mathcal{P}([m]) \setminus [m]}$. The *stability region* is defined as the set of arrival rates λ, for which the continuous time Markov chain $X(t)$ is positive recurrent.

C. State Transitions

The generator matrix for the process $X(t)$ is denoted by Q. For this continuous time Markov chain, there can only be a single transition in an infinitesimal time. We denote the system state as $x \in \mathcal{X}$ just before any transition, and let e_S be the unit vector in the direction corresponding to a proper subset $S \subset [m]$.

There are three types of possible transitions. The first type of state transition is the arrival of a new peer, that leads to an increase in the number of peers with no chunks. The corresponding transition rate is denoted by

$$Q(x, x + e_\emptyset) = \lambda.$$

The second and third type of transitions occur when a peer with $S \subset [m]$ chunks receives a chunk $j \notin S$ from the contacting seed/peer. In both these cases, the next state is denoted by $\mathcal{T}_{S,j}(x)$. The second type of state transition occurs when the reception of new chunks doesn't lead to a departure. This transition is denoted by

$$\mathcal{T}_{S,j}(x) \triangleq x - e_S + e_{S \cup \{j\}}, \quad x_S > 0, |S| < m - 1.$$

The third type of state transition occurs for a peer with $m-1$ chunks, which departs the system after getting the last chunk upon contact. This transition is denoted by

$$\mathcal{T}_{S,j}(x) \triangleq x - e_S, \quad x_S > 0, |S| = m - 1.$$

As all possible state transitions fall into one of the above three cases, the rate of transition for any other pair of states would be 0. At a system state x, if the contacting source has B chunks and the contacted receiving peer has S chunks, then the set of available chunks that can be transferred is $B \setminus S$. Selection of which chunk to transfer is called the *chunk selection policy*, which governs the evolution of the process $X(t)$. In particular, the last two transition rates $Q(x, \mathcal{T}_{S,j}(x))$ can only be computed for a specific Markov chunk selection policy. We describe the proposed chunk selection policy and the corresponding transition rates in the following section.

D. Chunk Distribution Information and Chunk Selection

A chunk selection policy decides which chunk to download from a given peer. It needs to have some information on the chunk distribution in order to determine which chunks to prioritize or suppress. Specifically, rarest-first assumes that the chunk frequencies in the system are known to the peer. This information is obtained through a distributed algorithm in BitTorrent that samples a set of peers to obtain an estimate [1]. Mode Suppression assumes the availability of the same chunk frequency information available in BitTorrent implementations. Group Suppression assumes that the full joint state distribution is known to all peers [32], which might be hard to acquire in practice. We also design "local" variants of mode suppression, which estimate chunk frequencies by sampling a few peers at each time.

According to the model, when the exponential clock of a peer ticks, it chooses a target peer at random, obtains the list of chunks that the target peer has in its possession, applies the chunk selection policy to this set of chunks, selects at-most one chunk, and downloads it instantaneously. We also simulate a generalized version of this process where the peer can choose several target peers when its clock ticks, so that it can make a selection of at-most one chunk from among the chunk-set of all the target peers contacted. This approach yields a higher chunk diversity to choose from, in-spite of still having the same bandwidth limitation of one chunk download per clock tick, and so improves sojourn time performance considerably. Specifically, we will consider the cases wherein the peer selects one versus three target peers at each time.

Finally, we note that while the "local" mode suppression policies use the same number (one or three) of targets for both frequency estimation, as well as chunk-set to select from,

rarest-first, mode-suppression and group-suppression have an out-of-band algorithm for chunk distribution estimation as described above.

III. Mode-Suppression Policy

In this section, we describe the general version of the Mode-Suppression (MS) policy (with a finite threshold T) and derive its rate transition matrix. First, we establish some notation. The set of allowable transfers from a peer with set of chunks B to a peer with set of chunks S, is denoted by $A(x, B, S) \subseteq B \setminus S$. The cardinality of this set is denoted by $h(x, B, S)$, and it takes integer values between 0 and m. Recall that the seed has all the chunks, and hence the set of allowable chunk transfers by the seed is $A(x, [m], S)$. Below, we describe the specifics of selecting the set of allowable transfers.

If there are no peers in the system, there is no need for chunk transfer. Hence, without loss of generality, we consider the mode-suppression policy when there exist peers in the system, or $|x| > 0$. Here, we assume that each peer has the knowledge of all chunk frequencies in the system. The frequency of the jth chunk is

$$\pi_j(x) \triangleq \frac{\sum_{j\in S} x_S}{|x|}. \tag{1}$$

Let $\overline{\pi}(x)$ and $\underline{\pi}(x)$ denote the maximum and minimum chunk frequencies, respectively, in a state x. Then we have

$$\overline{\pi}(x) = \max\{\pi_j(x) : j \in [m]\} \quad \text{and}$$
$$\underline{\pi}(x) = \min\{\pi_j(x) : j \in [m]\}.$$

The chunk indices that attain the highest frequency $\arg\max\{\pi_j(x) : j \in [m]\}$ are called the modes of the chunk frequencies. The set of modes is defined as

$$\mathcal{M}(x) \triangleq \{j \in [m] : \pi_j(x) = \overline{\pi}(x)\}.$$

We denote the number of peers with chunk j as

$$y_j(x) \triangleq \sum_{S:j\in S} x_S = \pi_j(x)|x|. \tag{2}$$

We can also define the number of peers with the maximum and the minimum chunk frequency by $\overline{y}(x)$ and $\underline{y}(x)$ respectively. The number of chunks in the system is denoted by

$$r(x) \triangleq \sum_{S\subset[m]} |S|x_S = \sum_{S\subset[m]} x_S \sum_{j\in[m]} 1_{\{j\in S\}} = \sum_{j\in[m]} y_j(x).$$

We can lower bound the total number of chunks by the number of most popular chunk. Therefore,

$$r(x) = \sum_{j\in[m]} y_j(x) \geqslant \overline{y}(x). \tag{3}$$

For simplicity of presentation, we would drop the dependence on the state x for y, π, r when the underlying state x is clear from the context. We can find a bound on the fraction of peers with least popular chunk from its definition in Lemma 4 in Appendix A in the supplementary material.

Now, we will describe the mode-suppression policy. The mode-suppression policy *restricts transmission of any chunk that belongs to the set of modes if its count is greater than count of the least frequent chunk by at least $T > 0$ units*. Denote the set of suppressed chunks in state x by $D_T(x)$ for some threshold $T \in \mathbb{N}$. According to MS $D_T(x)$ is given by

$$D_T(x) = \big\{k \in \mathcal{M}(x) : y_k(x) \geqslant \underline{y} + T\big\}. \tag{4}$$

The allowable transfer set for MS policy is

$$A(x, B, S) = B\backslash(S \cup D_T(x)). \tag{5}$$

The steps of the mode-suppression policy are shown in Algorithm 1 for a generic peer p.

Algorithm 1 Mode-Suppression Policy for Peer p

$S \leftarrow$ Chunk profile of p
while $S \neq [m]$ **do**
  $t \leftarrow t + \tau$, where $\tau \sim \exp(\mu)$
  $x \leftarrow X(t)$
  $\forall j \in [m]$, compute $y_j(x)$ from (2) and $D_T(x)$ from (4)
  Pick a source peer (B) randomly
  Choose a chunk (j) randomly from $B\backslash\Big(S \cup D_T(x)\Big)$
  Update $S \leftarrow S \cup \{j\}$
end while

The policy of the seed will be similar except for two differences. First, since the contact rate is U, $\tau \sim \exp(U)$ and second, seed pushes the chunk to the peer instead of pulling.

A. Properties of the Suppressed set

Note that if we set $T = 1$, the policy strictly suppresses the mode, and as T becomes larger, we increasingly relax suppression. When $T \to \infty$, MS is equivalent to the Random Chunk selection policy as there will not be any suppression, and chunks are chosen uniformly and at random. When the difference in number of peers with different chunks are all within threshold T, no chunks are suppressed. In this case, $D_T(x) = \emptyset$, and we can upper bound the fraction of peers with most popular chunk in the Lemma 5 in Appendix A in the supplementary material.

We would like to make two important observations regarding the suppressed set $D_T(x)$. We first observe that depending on the threshold T, either all modes are suppressed or none of the modes are suppressed. When $\mathcal{M}(x) = [m]$, then $D_T(x) = \emptyset$ by definition. Hence, we consider $\mathcal{M}(x) \subset [m]$. Since $y_k = \bar{y}$ for all $k \in \mathcal{M}(x)$, we have

$$D_T(x) = \begin{cases} \mathcal{M}(x), & \text{if } \overline{y} \geqslant \underline{y} + T, \\ \emptyset, & \text{otherwise.} \end{cases}$$

We next observe that, using the definition of chunk frequency, the set of suppressed states can be written as

$$D_T(x) = \mathcal{M}(x)1_{\{\overline{\pi}\geqslant\underline{\pi}+\frac{T}{|x|}\}} + \emptyset 1_{\{\overline{\pi}<\underline{\pi}+\frac{T}{|x|}\}}.$$

That is, the mode-suppression threshold is a function of the peer population. As the peer population $|x|$ grows large, the policy strictly suppresses the mode for $|x| \geqslant T$. Contrastingly for small peer population $|x| = 1$, the policy is most relaxed.

B. Transition Rates of the Contact Process

From the superposition of independent Poisson contact processes, the rate at which one of the peers with profile S contacts any other peer is also Poisson with the aggregate rate μx_S. The probability of contacting a source peer with profile B among all peers is $\frac{x_B}{|x|}$. From the thinning of Poisson process, we get that the Poisson contact process between any recipient peer with profile S and a source peer with profile B has rate $\mu x_S \frac{x_B}{|x|}$.

The contact process between seed and the peers is an independent Poisson process with rate U, where the seed contacts any peer at random. Hence, the Poisson contact process between seed and any peer with profile S occurs at rate $U x_S \frac{1}{|x|}$.

Since source peer has B chunks, then it can transfer one out of $h(x, B, S)$ available chunks to the destination peer with S chunks. The transition of type $\mathcal{T}_{S,j}$ occurs when one of the peers without chunk $j \notin S$ is contacted by seed or contacts a peer with chunks B, and receives the chunk j among all the possible choices. From the thinning and superposition of independent Poisson processes, we can write for $j \notin S$ and $x_S > 0$

$$Q(x, \mathcal{T}_{S,j}(x)) = \begin{cases} \dfrac{x_S}{|x|}\left(\dfrac{U}{h(x,[m],S)} + \mu \displaystyle\sum_{B: j\in B} \dfrac{x_B}{h(x,B,S)}\right) \\ \qquad \text{if } j \notin D_T(x), \\ 0 \quad \text{if } j \in D_T(x). \end{cases}$$

All other entries in the rate transition matrix other than the diagonal entries are 0, and the diagonal entries are equal to the negative sum of rest of the entries in that row.

It is difficult to work with exact transition rates for all transitions from state x to state $\mathcal{T}_{S,j}(x)$. We can lower bound the system performance by lower bounding the transition rates when $S \subset \{j\}^c$. To this end, we look at the the Poisson contact process of either the seed or one of the peers with chunk j with any peer, with the aggregate rate

$$R_j(x) \triangleq U + \mu \sum_{B: j\in B} x_B = U + \mu y_j(x)$$

from the superposition of independent Poisson contact processes. In Lemma 6 in Appendix A in the supplementary material, We can find a lower bound on the transition rates when $S \subset \{j\}^c$, and the exact transition rate when $S = \{j\}^c$, in terms of the rate $R_j = U + \mu y_j$.

IV. Stability Region of Mode-Suppression

In this section we characterize the stability region of mode-suppression. To prove the positive recurrence of the associated continuous time Markov chain $X(t)$, we employ the Foster-Lyapunov criteria.

Foster Lyapunov Criteria: Let ϕ be a time homogenous, irreducible and continuous time Markov process and $\mathcal{X}$ be its state space. If there exists a finite set of states $F \subset \mathcal{X}$, a Lyapunov function $V : \mathcal{X} \to (0, \infty)$ and some constants $b > 0$, $\epsilon > 0$, such that

$$QV(x) \leqslant -\epsilon + b1_{\{x\in F\}} \quad \forall x \in \mathcal{X},$$

then ϕ is positive recurrent [33], [35].

We consider the following Lyapunov function,

$$V(x) \triangleq \sum_{i=1}^{m} (\overline{y} - y_i)^2 + C_1(|x| - \overline{y}) + C_2(M - r)^+, \quad (6)$$

where, C_1, C_2 and M are positive constants that satisfies the constraints, $C_1 > (2T-1)(m-1)$, $C_2 \geqslant \frac{2m^2(C_1\lambda+\epsilon)}{U}$, $M > \max\{mN_{21}, N_{22}\}$, where $\epsilon > 0$ and N_{21}, N_{23} are positive constants defined in the equations (17) and (18) in the Appendix B of the supplementary material. Note that the explicit dependencies of $\pi(x)$ and $y(x)$ on x are not shown for simplicity.

The intuition behind this Lyapunov function is as follows. The nominal objective of MS is to approximately attain a uniform distribution of chunks (with the allowable error being related to the threshold value T). Hence, we should expect that the policy should promote negative Lyapunov drift whenever the current state differs from uniformity. Our Lyapunov function is designed to penalize three cases, namely, (i) where chunks have significantly differing frequency, (ii) where some might have zero frequency, and (iii) where all have zero frequency.

For a Markov process $X(t)$ with associated generator matrix Q, the expected rate of change of potential function from state x is called the mean drift from this state, and is given by

$$QV(x) \triangleq \sum_y Q(x,y)(V(y) - V(x)).$$

The mean drift from a state x for the Markov process $X(t)$ for the mode-suppression policy, in terms of its generator matrix Q can be written as

$$\begin{aligned} QV(x) = {} & Q(x, x+e_\emptyset)(V(x+e_\emptyset) - V(x)) \\ & + \sum_{j\in[m]} \sum_{S: j\notin S} Q(x, \mathcal{T}_{S,j}(x))(V(\mathcal{T}_{S,j}(x)) - V(x)). \end{aligned} \quad (7)$$

First, we compute the mean drift corresponding to a new peer arrival. The arrival of a new peer does not change the number of peers with chunk $j \in [m]$. However, it does lead to a unit increase in the number of peers in the system. That is,

$$Q(x, x+e_\emptyset)(V(x+e_\emptyset) - V(x)) = \lambda C_1.$$

We observe that the set of chunks S such that $j \notin S$ is identical to $S \subseteq \{j\}^c$. Hence, we can write the mean drift $QV(x)$ from state x in (7) to be equal to

$$\lambda C_1 + \sum_{j\in[m]} \left(\sum_{S\subset\{j\}^c} + \sum_{S=\{j\}^c} \right) Q(x,x')(V(x') - V(x)),$$

where $x' = \mathcal{T}_{S,j}(x)$.

We show in Theorem 13 in Appendix B in the supplementary material, that the Markov process $X(t)$ corresponding to the number of peers with different subsets of all chunk set $[m]$, is positive recurrent for all arrival rates $\lambda > 0$. To this end, we employ the Foster-Lyapunov criteria [35] for the Lyapunov function defined in (6). As a first step, we upper bound the difference in Lyapunov function between state $\mathcal{T}_{S,j}(x)$ and x in Lemma 7 in the Appendix B of the supplementary material. We can bound the aggregate transition rate $\sum_{S\subset\{j\}^c} Q(x, \mathcal{T}_{S,j}(x))$ by bounding the fraction of peers

without single chunk $\sum_{S\subset\{j\}^c}\frac{x_S}{|x|}$. This bound is presented in Lemma 9 in the Appendix B of the supplementary material.

We note that when the set of suppressed chunks $D_T(x)\neq\emptyset$ in state x, transition to state $\mathcal{T}_{S,j}(x)$ is possible only when the chunk $j\notin D_T(x)=\mathcal{M}(x)$. That is, the chunk $j\notin\mathcal{M}(x)$ for any transition to state $\mathcal{T}_{S,j}(x)$. When the set of suppressed chunks $D_T(x)=\emptyset$, transition to state $\mathcal{T}_{S,j}(x)$ is possible for all chunks $j\in[m]$. In particular, it is possible that the chunk $j\in\mathcal{M}(x)$. This leads to the Corollary 8 in the Appendix B of the supplementary material that upper bounds the difference between Lyapunov function evaluated at state $\mathcal{T}_{S,j}(x)$ and state x, for chunk set $S\subseteq\{j\}^c$ and peer $j\notin\mathcal{M}(x)$ and $j\in\mathcal{M}(x)$. From the bound on the Lyapunov difference $V(\mathcal{T}_{S,j})(x)-V(x)$ and the bound on aggregate transition rate $\sum_{S\subseteq\{j\}^c}Q(x,\mathcal{T}_{S,j}(x))$, we can bound the mean drift for non-empty and empty set of suppressed chunks in Proposition 10 and Proposition 11 in Appendix B of the supplementary material, respectively. Next, we can show in Lemma 12 in the Appendix B of the supplementary material that the mean drift is strictly negative when the number of peers with the most frequent chunk exceed a threshold. Together these results suffice to show the Foster-Lyapunov criteria can be applied to the Lyapunov function under consideration, and we can conclude that the Markov process $X(t)$ is positive recurrent.

V. Scaling of Swarm Size and Sojourn Time

Our next result is on the scaling properties of MS with respect to the peer arrival rate λ. We use the following Kingman moment bound to prove the properties.

Theorem 1 (Kingman Moment Bound [24]): Let X be a continuous-time, irreducible Markov process on a countable state space $\mathcal{X}$ with generator matrix Q. Suppose V, f, and g are nonnegative functions over the state space $\mathcal{X}$, and suppose $QV(x)\leqslant -f(x)+g(x)$ for all $x\in\mathcal{X}$. In addition, suppose X is positive recurrent, so that the means, $\bar{f}=\pi f$ and $\bar{g}=\pi g$ are well defined. Then $\bar{f}\leqslant\bar{g}$.

We then have the following scaling result.

Theorem 2: Under the Mode-suppression policy, the following statements are true.

1) **(Scaling of Swarm Size)** *The average number of peers in the system $L\leqslant C\lambda$, where C is a constant.*
2) **(Scaling of Sojourn time)** *The average sojourn time of the peers W, is bounded.*

Proof: See Appendix C-A in the supplementary material. ■

The scaling result implies that for the proposed mode-suppression policy, the average number of peers is growing at most linearly with the peer arrival rate in the system. That is, the average sojourn time of each peer remains bounded. It follows that the proposed policy scales well for all peer arrival rates. This also suggests that the sojourn time will remain bounded for variable arrival rates, as long as the arrival rate is finite at all times.

VI. Limited Observation Based Policies

Mode-suppression is simple to integrate into BitTorrent via utilizing the distributed approach to determining chunk frequencies available within BitTorrent. The procedure used by BitTorrent requires the sampling of many peers, and the question arises as to whether approaches to determining chunk frequencies that rely on a more limited set of observations can be devised? We shed some light on this question below.

A. Local Mode-Suppression Policy

Under *local mode-suppression* (LMS), a peer contacts three other peers at random, and among the chunks available with more than one peer, we define the *local mode* to be the chunk(s) with greatest frequency. The peer is allowed to download any chunk that is not part of the local mode. Any chunk may be downloaded if all chunks are equally frequent.

Let $B^j, j=1,2,3$ denote the chunk profiles of three selected peers and $B=\left\{B^1,B^2,B^3\right\}$, then we can write the modes

$$\mathcal{M}_{LMS}(x,B)=\left\{i\in[m]\middle|\sum_{j=1}^{3}B_i^j\geqslant\max_{k\in[m]}\sum_{j=1}^{3}B_k^j,\sum_{j=1}^{3}B_i^j>1\right\},\quad(8)$$

and we write the set of suppressed chunks (regardless of whether the chunk is downloaded from the seed or another peer) as

$$D_{LMS}(x,B)=\begin{cases}\mathcal{M}_{LMS} & \text{if } \mathcal{M}_{LMS}\neq[m],\\ \emptyset & \text{if } \mathcal{M}_{LMS}=[m].\end{cases}\quad(9)$$

The steps of the local mode-suppression policy are shown in Algorithm 2.

Algorithm 2 Local Mode-Suppression for Peer p

$S\leftarrow$ Chunk profile of p
while $S\neq[m]$ **do**
  $t\leftarrow t+\tau$, where $\tau\sim\exp(\mu)$
  $x\leftarrow X(t)$, $S\leftarrow$ Chunk profile of p
  Select three source peers (B^i) randomly
  Compute $D_{LMS}(x,B)$ from (9)
  Choose a chunk j randomly from $\cup_{i=1}^{i=3}B_i\backslash\Big(S\cup D_{LMS}(x,S)\Big)$
  Update $S\leftarrow S\cup\{j\}$
end while

Theorem 3: The stability region of Local Mode-Suppression (LMS) is $\lambda>0$ if $m=2,\mu>0$ and $U>0$.

Proof: The proof for $m=2$ chunks follows using the same Lyapunov function and steps as the proof of the Rare Chunk policy [32], and is hence omitted. ■

Stability for the case $m>2$ chunks is left as a conjecture.

B. EWMA Mode-Suppression

Under this policy, each peer calculates the empirical marginal chunk frequencies based only on the chunks possessed by all peers that it has met until (and including) the current time. The marginal chunk frequency is calculated using an Exponentially Weighted Moving Average (EWMA) taking into account both history and present, and the mode of this estimate is suppressed.

Let $n \in \mathbb{N}$ denote the index of Poisson ticks of a peer. We define empirical marginal chunk frequencies of a peer p with $\tilde{\pi}^n(p)$ and are computed as below for each chunk $j \in [m]$,

$$\tilde{\pi}_j^0(p) = 0, \quad \tilde{\pi}_j^n(p) = (1-\alpha)\tilde{\pi}_j^{n-1}(p) + \alpha B_j^n, \qquad (10)$$

where B^n denotes the chunk profile of the source peer selected at time slot n by peer p and $\alpha \in (0,1)$ is the exponential weighting parameter. The modes for this policy are defined as

$$\mathcal{M}_{EWMA}^n(p) = \left\{ i | \tilde{\pi}_i^n(p) \geq \max_{j \in [m]} \tilde{\pi}_j^n(p) \right\}, \qquad (11)$$

and the set of suppressed chunks (regardless of whether the chunk is downloaded from the seed or another peer) are denoted by

$$D_{EWMA}^n(p) = \begin{cases} \mathcal{M}_{EWMA}^n & \text{if } \mathcal{M}_{EWMA}^n \neq [m], \\ \emptyset & \text{if } \mathcal{M}_{EWMA}^n = [m]. \end{cases} \qquad (12)$$

The steps of EWMA Mode-Suppression Policy is shown in Algorithm 3.

Algorithm 3 EWMA Mode-Suppression for Peer p

$S \leftarrow$ Chunk profile of p, $n = 0$
while $S \neq [m]$ **do**
  $t \leftarrow t + \tau$, where $\tau \sim \exp(\mu)$, $n \leftarrow n + 1$
  $x \leftarrow X(t)$
  Pick a source Peer (B) randomly
  $\forall j \in [m]$, compute $\tilde{\pi}_j^n(p)$ from (10) and $D_{EWMA}^n(p)$ from (12)
  Choose a chunk j randomly from $B \backslash \left(S \cup D_{EWMA}^n(p) \right)$
  Update $S \leftarrow S \cup \{j\}$
end while

While we have proposed algorithms in this section that only exploit limited information, we have not provided their full analysis. We will study them empirically in the simulation setting to determine if they show sufficiently good performance that warrants further analysis in the future.

VII. Numerical Experiments

In this section, we show the results from numerical simulations that illustrate the performance of different chunk selection policies. Recall that our candidate policies are (i) random chunk selection, (ii) rarest-first, (iii) rare chunk, (iv) common chunk, (v) group suppression, (vi) mode-suppression, (vii) local mode-suppression, and (viii) mode-suppression-EWMA. Recall that the random chunk selection and the rarest-first chunk selection policies are known to be unstable [3] for large peer arrival rates, and hence we do not study the random chunk policy. A description of these policies can be found in Sections I, III, and VI. For all the simulations, we set the peer contact rate U and seed contact rate μ as 1. Each peer in the system, including the seed, generates an exponential random variable with mean $\frac{1}{\mu} = \frac{1}{U} = 1$, and the one with the smallest value gets a chance to contact another peer.

A. Chunk Frequency Evolution

A stable chunk selection policy has to be robust to the one-club state. In other words, a stable policy should be able to boost the frequency of a rare chunk. To see how different policies handle the one-club situation, we start the system with 500 peers that have all the chunks except first chunk (i.e., all peers are part of the one-club). In Figure 1, we plot the evolution of the chunk frequency for different policies under this initial condition. We see that when using the rarest-first policy, the rare chunk remains rare and abundant chunks remain abundant, which is a clear sign of instability. In all stabilizing policies, the rare chunk is made available by giving priority to that chunk in some way. For instance, in case of mode-suppression ($T = 1$), no other chunk will be transmitted until the frequency of the rare chunk is equal to the frequency of all other chunks. Once this happens, the frequencies of the different chunks remain almost same, and hence we only see a thin spread across the frequencies. Other policies also manage to bring the rare chunk back into circulation and the corresponding statistics become similar to all other chunks. We also observe that the *stabilization time* to increase the frequency of rare chunk to the same level as that of other chunk frequencies, is shorter for mode-suppression and local mode-suppression when compared to other algorithms.

B. Sojourn Times

In a stable system, an important performance metric is the sojourn time of a peer, which is defined as the amount of time a peer spends in the system collecting all chunks before leaving. For numerical illustration of sojourn time, we fix the peer arrival rate at $\lambda = 30$, and we calculate the mean stationary sojourn times of the peers under different policies, for different values of the number of file chunks m. The stationary sojourn times are obtained by running the system for a long period of time and ignoring the first 2000 peers that left the system. Our goal is to evaluate how effectively the algorithms use their information on chunk statistics.

Our first result is on determining the value of threshold T that minimizes the sojourn time under MS. Intuitively, the threshold is a way of allowing "noisy" suppression of the mode. It seems reasonable that as the number of chunks increases, the amount of noise permitted should also be allowed to increase in the interest of allowing more sharing to take place. Thus, we numerically studied different values of T that are increasing with the number of chunks m, and found empirically that setting $T = 2m$ appears to minimize the sojourn time under MS.

We also wish to study the effect of chunk diversity provided through the option to choose a chunk from the set of chunks possessed by 1 versus 3 peers. Thus, we have two versions of each algorithm that both use identical chunk statistics (obtained through sampling some or all peers as per the algorithm). However, the first version can obtain any one chunk from those possessed by 1 randomly selected peer, while the second can pick any one chunk from the set of chunks possessed by 3 randomly selected peers.

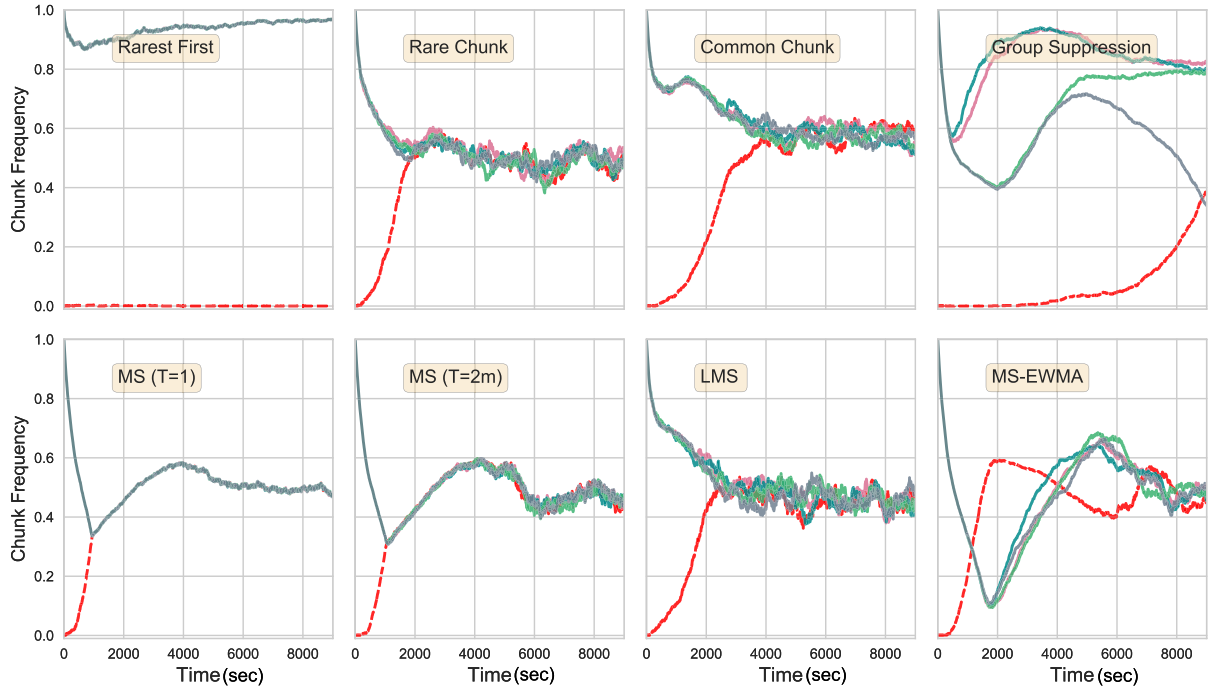


Fig. 1. Chunk frequency evolution in a system with $m = 5$ chunks under different policies when starting from the state of a "missing-chunk" (whose frequency is indicated by a red/dashed line). Rarest-first is clearly unstable, since it cannot recover, whereas the other protocols manage to bring the chunk back into peer circulation and stabilize the system. The time taken for a single chunk upload is taken as one second.

Recall that the qualitative comparison of sojourn time across the different algorithms appears in Table II. In Figure 2, we present a numerical comparison of sojourn times across the different algorithms. The increased sojourn times of RC and CC are visible, although increasing chunk diversity by sampling 3 peers improves RC considerably. GS has good performance, although the variability in sojourn time seen in the error bars (standard deviation) is high, noticeable when selection is from one peer or when m is large. The variants of MS all perform well, with the MS ($T = 2m$), LMS, and MS-EWMA, all showing low sojourn times. It is interesting to note that in the example, since the contact rate is 1, the best case sojourn time is equal to the number of chunks m. We see that for the case of sampling 3 peers, the mode-suppression variants MS ($T = 2m$), LMS, and MS-EWMA, attain a mean sojourn time that is very close to m, indicating that they achieve a near-optimal tradeoff between suppression (to keep peers in the system) and sharing (to enable peers to gather chunks).

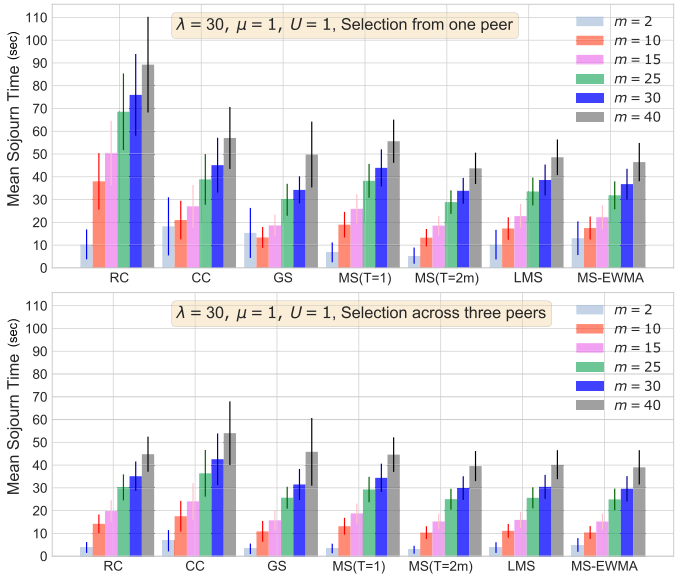


Fig. 2. Stationary mean sojourn times of stable policies for different values of m. The two regimes correspond to downloading a chunk from 1 peer, or downloading one chunk from the chunk set of 3 peers. The time taken for a single chunk upload is taken as one second.

VIII. BitTorrent Over NS-3 Experiments

In this section, we present P2P experiments over the ns-3 simulator [36]. We selected an open-source modular BitTorrent application model for ns-3 called VODSIM [37], which is fully compliant with the protocol description [38]. The framework of VODSIM over ns-3 allows us to easily collect micro benchmarks for each simulation run. Our goal is a comparison of the stability and sojourn time properties of rarest-first and mode-suppression policies.

As mentioned in Section I, BitTorrent already implements a distributed algorithm for identifying the chunk frequencies via observations from a subset of peers in the swarm, which we can directly leverage. Experiments with rarest-first are straightforward, since it is the default chunk selection policy of BitTorrent, and uses the chunk frequency observations to prioritize the lowest-frequency chunks for sharing. Mode-suppression is our custom modification that restricts the chunk set that may be shared at any time using the same chunk frequency information, and requires only about 20 lines of source code changes to implement.

We note a few other elements of BitTorrent that are relevant to our experiments. First, BitTorrent utilizes a Tit-for-Tat

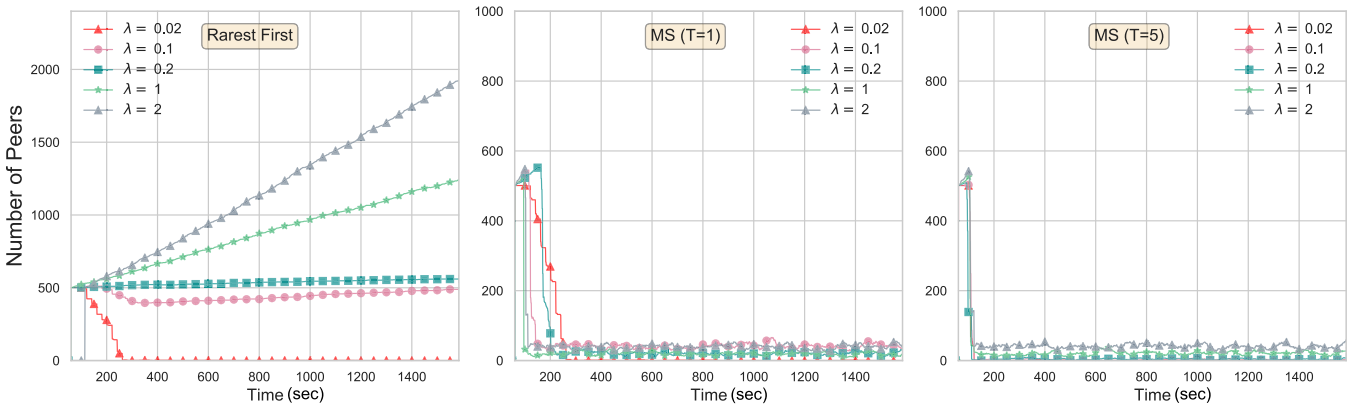


Fig. 3. Number of peers in the system when $m = 10$. Rarest-first becomes unstable for higher arrival rates, whereas mode-suppression is always stable.

mechanism that ensures strategic cooperation for chunk sharing via a scheme for "unchoking" an appropriate peer set, which we do not modify. Furthermore, it divides chunks into sub-blocks for sharing. We set the size of a sub-block as identical to that of a chunk for consistency with the analytical model, and to allow easier measurement of system state.

With the above simulation setup, we initialize a single seed that serves the file of interest, and a single Tracker that acts as a controller. Depending on the scenario, we initialize a fixed number of peers holding a subset of chunks of the file. Peers arrive at the BitTorrent swarm with an exponentially distributed inter-arrival time, which allows us to control the arrival rate of peers using a single parameter. Chunks are shared according to the BitTorrent protocol over the simulated network with the appropriate chunk sharing policy, and peers leave the system immediately after completion.

A. Stability

We first demonstrate the instability of the rarest-first policy for arrival rates larger than the seed service rate, and stability of the mode-suppression policy for any arrival rate. As in the previous section, we initialize the system state in one-club for both the rarest-first and the mode-suppression policy to observe the impact on the swarm size. Continual increase of the swarm size will indicate instability.

For the experiments, we chose a file of size 320 kB divided into 10 chunks of 32 kB each, and initialized the swarm with 500 peers as members of one-club. We repeated the experiment for values of the peer arrival rate into the swarm, $\lambda \in [0.02, 0.1, 0.2, 1, 2]$. We set the seed upload rate 100 kbps and the number of unchoked peers for the seed to 1. It follows that each client in the one-club takes $32 \times 8/100$ seconds to leave the system, which corresponds to an approximate service rate of $U \approx 0.4$ chunks per second.

We show the time evolution of number of peers in the swarm under the rarest-first policy in Figure 3 (left). We observe that the number of peers increases linearly in time, when the peer arrival rate λ exceeds the seed service rate U, indicating the instability. For lower arrival rates $\lambda < U$, the system is stable, as is evidenced by the gradual departure of clients from the one-club. In Figure 3 (middle-right), we also present our results under strict mode-suppression ($T = 1$) and mode-suppression with threshold $T = 5$, respectively under identical system parameters. Unlike rarest-first policy, mode-suppression is stable for all arrival rates.

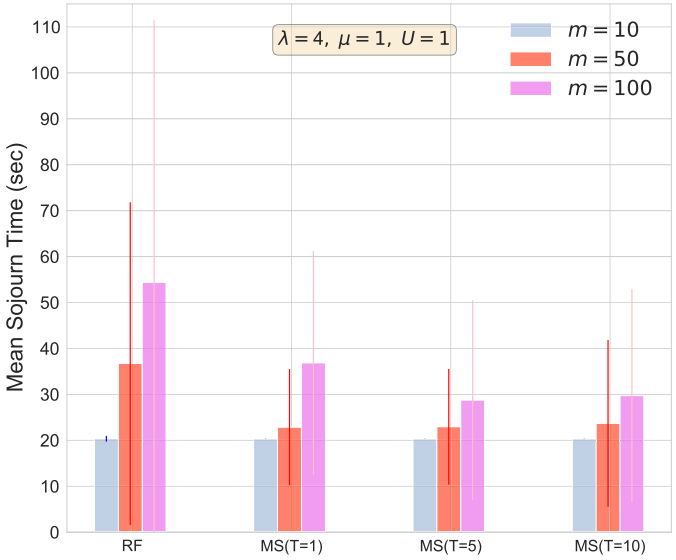


Fig. 4. Stationary mean sojourn times of stable policies for different values of m.

B. Mean Sojourn Time

We next present our simulation results for mean sojourn time. Here, we choose a peer arrival rate $\lambda = 4$, and the seed upload rate of $U = 1$ chunk per second, and the peer upload rate of $\mu = 1$ chunk per second. We compute the mean sojourn time for the mode-suppression policy with different thresholds $T \in \{1, 5, 10\}$ and the rarest-first policy, for varied values of the number of file chunks m.

In Figure 4 we compare the corresponding mean sojourn times for both algorithms. Under each policy, the increase in mean sojourn times as the number of file chunks increase is apparent. However, the mean sojourn time values achieved by mode-suppression are generally smaller than those of rarest-first. Note also that rarest-first is unstable here, and hence has continually increasing sojourn times, implying the the "mean" shown is simply the average of values observed thus far rather than representing the statistics of a stationary distribution.

So the error bars here keep increasing with the duration of the simulation.

We also observe that although a threshold value of $T = 2m$ appears to be optimal from our numerical studies in Section VII-B, such a choice might not always be reasonable in the real BitTorrent system. The reason is that in BitTorrent, the peer set of a client is limited to 50 peers. Recall that the suppressed set is non-empty only when the condition $T \leqslant (\overline{\pi} - \underline{\pi})|x|$ is satisfied, which in turn is limited by the fact that $|x| = 50$ in this case. Hence, setting large values of T as the number of chunks increases would imply no suppression, resulting in erratic behavior. In practice, we found that a values between $T = 5$ and $T = 10$ show good performance.

IX. Conclusion

In this work, we analyzed the scaling behavior of a P2P swarm with reference to its stability when subjected to an arbitrary arrival rate of peers. It has been shown earlier that not all chunk sharing policies are stable in such a regime, and our goal was to design a simple and stable policy that yields low sojourn times. Our main observation was that, contrary to the traditional approach of boosting the availability of rare chunks, preventing the spread of chunk(s) that are more frequent as compared to the lowest frequency chunks (where the maximum allowed threshold is a parameter of the algorithm) yields a simple and stable policy that we entitled mode-suppression (MS). We analytically proved its stability, and showed that the sojourn time under this algorithm does not scale up with increasing demand (peer arrival rate). Our results indicate that there is a delicate trade-off between sharing (i.e., uploading a useful chunk if at all possible) and suppression (i.e., trying to reduce chunk transfers to keep peers in the system so that they can help others). We showed in both numerical studies and simulations of BitTorrent over ns-3 that MS with an appropriately selected threshold results in stable behavior with low sojourn times.

Acknowledgment

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

References

[1] B. Cohen, "Incentives build robustness in BitTorrent," in *Proc. Workshop Econ. Peer-Peer Syst.*, vol. 6, 2003, pp. 68–72.

[2] *The Global Internet Phenomena Report: September 2019*, Sandvine, Waterloo, ON, Canada, 2019.

[3] B. Hajek and J. Zhu, "The missing piece syndrome in peer-to-peer communication," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2010, pp. 1748–1752.

[4] D. X. Mendes, E. de Souza e Silva, D. Menasché, R. Leão, and D. Towsley, "An experimental reality check on the scaling laws of swarming systems," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2017, pp. 1647–1655.

[5] M. Zhao *et al.*, "Peer-assisted content distribution in akamai netsession," in *Proc. Conf. Internet Meas. Conf. (IMC)*, Oct. 2013, pp. 31–42.

[6] A. Kobusińska, J. Brzeziński, J. Aftowicz, and G. Grzelachowski, "Distributed content dissemination with a rank function," in *Proc. IEEE Int. Conf. Comput. Inf. Technol. (CIT)*, Dec. 2016, pp. 468–475.

[7] X. Zeng, L. Gao, C. Jiang, T. Wang, J. Liu, and B. Zou, "A hybrid pricing mechanism for data sharing in P2P-based mobile crowdsensing," in *Proc. 16th Int. Symp. Modeling Optim. Mobile, Ad Hoc, Wireless Netw. (WiOpt)*, May 2018, pp. 1–8.

[8] C. Jiang, L. Gao, L. Duan, and J. Huang, "Scalable mobile crowdsensing via peer-to-peer data sharing," *IEEE Trans. Mobile Comput.*, vol. 17, no. 4, pp. 898–912, Apr. 2018.

[9] J. Dixon, T. Morstyn, L. Han, and M. McCulloch, "Flexible cooperative game theory tool for peer-to-peer energy trading analysis," in *Proc. IEEE Power Energy Soc. Gen. Meeting (PESGM)*, Aug. 2018, pp. 1–5.

[10] W. Tushar *et al.*, "Grid influenced peer-to-peer energy trading," *IEEE Trans. Smart Grid*, vol. 11, no. 2, pp. 1407–1418, Mar. 2020.

[11] K. Zhang, S. Troitzsch, S. Hanif, and T. Hamacher, "Coordinated market design for peer-to-peer energy trade and ancillary services in distribution grids," *IEEE Trans. Smart Grid*, vol. 11, no. 4, pp. 2929–2941, Jul. 2020.

[12] R. Boussaha, Y. Challal, A. Bouabdallah, D. Ighit, and L. Tairi, "Peer-to-peer collaborative video-on-demand streaming over mobile content centric networking," in *Proc. IEEE 32nd Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, May 2018, pp. 1053–1059.

[13] J. Kim, G. Caire, and A. F. Molisch, "Quality-aware streaming and scheduling for device-to-device video delivery," *IEEE/ACM Trans. Netw.*, vol. 24, no. 4, pp. 2319–2331, Aug. 2016.

[14] L. Qing, F. Xuan-Li, H. Yu-Ke, and H. Wan-Jie, "Super-peer selection algorithm based on AHP in mobile peer-to-peer network," in *Proc. 7th Int. Conf. Inf. Technol., IoT Smart City*, Dec. 2019, pp. 305–309.

[15] Z. Wen, N. Liu, K. L. Yeung, and Z. Lei, "Closest playback-point first: A new peer selection algorithm for P2P VoD systems," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, Dec. 2011, pp. 1–5.

[16] T. Naito and S. Fujita, "Acquiring nearly optimal peer selection strategy through deep Q-network," in *Proc. 6th Int. Symp. Comput. Netw. (CANDAR)*, Nov. 2018, pp. 120–125.

[17] Y. Yuan and F.-Y. Wang, "Blockchain and cryptocurrencies: Model, techniques, and applications," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 48, no. 9, pp. 1421–1428, Sep. 2018.

[18] R. K. Raman and L. R. Varshney, "Dynamic distributed storage for blockchains," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 2619–2623.

[19] M. Sabounchi and J. Wei, "Towards resilient networked microgrids: Blockchain-enabled peer-to-peer electricity trading mechanism," in *Proc. IEEE Conf. Energy Internet Energy Syst. Integr. (EI2)*, Nov. 2017, pp. 1–5.

[20] D. Qiu and R. Srikant, "Modeling and performance analysis of BitTorrent-like peer-to-peer networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, pp. 367–378, Aug. 2004.

[21] X. Yang and G. de Veciana, "Performance of peer-to-peer networks: Service capacity and role of resource sharing policies," *Perform. Eval.*, vol. 63, no. 3, pp. 175–194, Mar. 2006.

[22] S. Shakkottai and R. Johari, "Demand-aware content distribution on the Internet," *IEEE/ACM Trans. Netw.*, vol. 18, no. 2, pp. 476–489, Apr. 2010.

[23] B. Fan, J. C. S. Lui, and D.-M. Chiu, "The design trade-offs of BitTorrent-like file sharing protocols," *IEEE/ACM Trans. Netw.*, vol. 17, no. 2, pp. 365–376, Apr. 2009.

[24] J. Zhu and B. Hajek, "Stability of a peer-to-peer communication system," *IEEE Trans. Inf. Theory*, vol. 58, no. 7, pp. 4693–4713, Jul. 2012.

[25] L. Massoulie and M. Vojnovic, "Coupon replication systems," *IEEE/ACM Trans. Netw.*, vol. 16, no. 3, pp. 603–616, Jun. 2008.

[26] K. Shin, D. S. Reeves, and I. Rhee, "Treat-before-trick: Free-riding prevention for BitTorrent-like peer-to-peer networks," in *Proc. IEEE Int. Symp. Parallel Distrib. Process.*, May 2009, pp. 1–12.

[27] J. K. So and D. S. Reeves, "Adaptive neighbor management for cooperative P2P video-on-demand streaming," in *Proc. 30th IEEE Int. Perform. Comput. Commun. Conf.*, Nov. 2011, pp. 1–10.

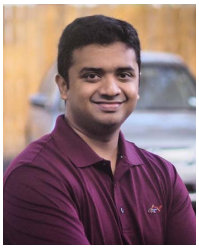
[28] J. Dharanipragada and H. Haridas, "Stabilizing peer-to-peer systems using public cloud: A case study of peer-to-peer search," in *Proc. 11th Int. Symp. Parallel Distrib. Comput.*, Jun. 2012, pp. 135–142.

[29] A. L. Jia, R. Rahman, T. Vinkó, J. A. Pouwelse, and D. H. J. Epema, "Fast download but eternal seeding: The reward and punishment of sharing ratio enforcement," in *Proc. IEEE Int. Conf. Peer-Peer Comput.*, Aug. 2011, pp. 280–289.

[30] H. Reittu, "A stable random-contact algorithm for peer-to-peer file sharing," in *Proc. IFIP Int. Workshop Self-Organizing Syst. (IWSOS)*, Dec. 2009, pp. 185–192.

[31] I. Norros, H. Reittu, and T. Eirola, "On the stability of two-chunk file-sharing systems," *Queueing Syst.*, vol. 67, no. 3, pp. 183–206, Mar. 2011.

[32] B. Oguz, V. Anantharam, and I. Norros, "Stable distributed P2P protocols based on random peer sampling," *IEEE/ACM Trans. Netw.*, vol. 23, no. 5, pp. 1444–1456, Oct. 2015.
[33] O. Bilgen and A. B. Wagner, "A new stable peer-to-peer protocol with non-persistent peers: The group suppression protocol," *IEEE Trans. Inf. Theory*, vol. 66, no. 1, pp. 614–632, Jan. 2020.
[34] V. Reddyvari, P. Parag, and S. Shakkottai, "Mode-suppression: A simple and provably stable chunk-sharing algorithm for P2P networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2018, pp. 2573–2581.
[35] S. P. Meyn and R. L. Tweedie, *Markov Chains and Stochastic Stability*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2009.
[36] *ns-3 Network Simulator*. Accessed: Jun. 30, 2021. [Online]. Available: https://www.nsnam.org
[37] E. Weingärtner, R. Glebke, M. Lang, and K. Wehrle, "Building a modular BitTorrent model for ns-3," in *Proc. 5th Int. Conf. Simulation Tools Techn.*, 2012, pp. 337–344.
[38] *BitTorrent Protocol Specification V1.0*. Accessed: Jun. 30, 2021. [Online]. Available: https://wiki.theory.org/index.php/BitTorrent Specification



Vamseedhar Reddyvari (Member, IEEE) received the bachelor's degree in electrical engineering from the Indian Institute of Technology Bombay, India, in 2009, the M.Tech. degree in electrical engineering from the Indian Institute of Technology Kanpur, India, in 2012, and the Ph.D. degree from Texas A&M University, College Station, TX, USA, in 2018. He is currently working with Google. His research interests include P2P networks, game theory, and BGP security.

Sarat Chandra Bobbili received the B.E. degree (Hons.) in electronics and communication engineering from the Birla Institute of Technology and Science Pilani, India, in 2015, and the M.Tech. (Research) degree in electrical communication engineering from the Indian Institute of Science, Bengaluru, India, in 2020. He is currently with Qualcomm India Pvt. Ltd., Hyderabad, as a Modem Firmware Developer. His research interests include reinforcement learning, communication systems, game theory, and online learning.

Parimal Parag (Senior Member, IEEE) received the B.Tech. and M.Tech. degrees from IIT Madras in 2004 and the Ph.D. degree from Texas A&M University in 2011, all in electrical engineering.

He joined the Indian Institute of Science in 2014, where he is currently an Associate Professor with the Department of Electrical Communication Engineering. Prior to that, he was a Senior System Engineer (Research and Development) with ASSIA, Inc., Redwood City, CA, USA, from 2011 to 2014. His research interests include the design and analysis of large scale distributed systems. He was the coauthor of the 2018 IEEE ISIT Student Best Paper, and a recipient of the 2017 Early Career Award from the Science and Engineering Research Board.

Srinivas Shakkottai (Senior Member, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of Illinois at Urbana-Champaign in 2007.

He was a Post-Doctoral Scholar of Management Science and Engineering with Stanford University in 2007. He joined Texas A&M University in 2008, where he is currently a Professor of computer engineering with the Department of Electrical and Computer Engineering. His research interests include caching and content distribution, wireless networks, multi-agent learning, and game theory, as well as network data collection and analytics. He was a recipient of the Defense Threat Reduction Agency Young Investigator Award in 2009 and the NSF Career Award in 2012, as well as Research Awards from Cisco in 2008 and Google in 2010. He received the Outstanding Professor Award from Texas A&M University in 2013, the Select Young Faculty Fellowship from Texas A&M University in 2014, and the Engineering Genesis Award from Texas A&M University in 2019.