A New Stable Peer-to-Peer Protocol With Non-Persistent Peers: The Group Suppression Protocol

Omer Bilgen, *Member, IEEE*, and Aaron B. Wagner, *Senior Member, IEEE*

***Abstract*— Recent studies have suggested that the stability of peer-to-peer networks may rely on *persistent peers*, who dwell on the network after they obtain the entire file. In the absence of such peers, one piece becomes extremely rare in the network, which leads to instability. Technological developments, however, are poised to reduce the incidence of persistent peers, giving rise to a need for a protocol that guarantees stability with non-persistent peers. We propose a novel peer-to-peer protocol, *the group suppression protocol*, to ensure the stability of peer-to-peer networks under the scenario that all the peers adopt non-persistent behavior. Using a suitable Lyapunov potential function, the group suppression protocol is proven to be stable when the file is broken into two pieces, and detailed experiments demonstrate the stability of the protocol for arbitrary number of pieces. We define and simulate a decentralized version of this protocol for practical applications. Straightforward incorporation of the group suppression protocol into BitTorrent while retaining most of BitTorrent's core mechanisms is also presented. Subsequent simulations show that under certain assumptions, BitTorrent with the official protocol cannot escape from the missing piece syndrome, but BitTorrent with group suppression does.**

***Index Terms*— Peer-to-peer, BitTorrent, stability, file-sharing, content distribution, Lyapunov, stable protocol, missing piece, rarest.**

I. Introduction

IN A peer-to-peer network, a file is divided into a number of pieces, and each peer uploads the pieces obtained so far to other peers while it continues to download the remaining pieces. This results in high utilization of the bandwidth of all peers leading to improved scalability over a traditional server-client structure. Such systems have been deployed widely, as evident from their number of users and the fraction of overall internet traffic they command [1], and they are poised to become even more prominent with the rise of fog networking [2], [3]. A high level p2p protocol can be understood to be a set of rules that governs how a peer contacts another peer and chooses the piece to upload/download to/from that peer, depending on whether a "push" or "pull" policy is employed, respectively. *Network* and *system* will be used interchangeably to refer to the collection of all peers throughout the paper. Also we define a peer to be an *incomplete peer* if it does not have all the pieces of the file and a *seed* if it does.

Peer-to-peer networks with random peer and piece selection policies have been shown to struggle with what Hajek and Zhu [4] call *the missing piece syndrome* in which very few peers possess a missing piece while most of the peers, referred to as *the one club* in [4], have all the pieces but the missing piece [4], [5]. A qualitative argument why such an imbalanced state persists in the network is as follows: Assume that the one club currently dominates the network, uniformly random peer and piece selection policies are in effect, the peers are non-persistent in the sense that the peers depart as soon as they have the entire file, and the new peers arrive into the network with no pieces. As a result of the uniformly random contact policy, any new peer will contact peers mostly from the one club and thereby become a member of the one club before it can obtain the missing piece. Even when the new peers obtain the missing piece before they are pulled to the one club, they would download the rest of the file quickly thanks to the large upload capacity of the one club. Therefore, they would depart without disseminating the missing piece to other peers long enough to significantly decimate the one club. Hence, the number of peers in the system diverges to infinity once the system is overwhelmed by the one club. A more thorough discussion of the dynamics of the missing piece syndrome can be found in [4]. In the same paper, it has been proven that if the upload rate of the fixed seed is greater than the arrival rate of new peers, who carry no pieces under the model, the system is stable. On the other hand, if the arrival rate of peers exceeds the upload rate of the fixed seed, then the system experiences the missing piece syndrome and is unstable.

With an ideal p2p protocol the system should remain stable irrespective of the relationship between the arrival rate and the upload rate of the seed, as the arrival rate cannot be internally controlled by the protocol. Zhu and Hajek [6] establish that if peers linger in the system on average long enough to upload one piece, then the system is stable regardless of the arrival rate. This highlights the significance of having persistent peers from a stability point of view. Based on this study, one could reasonably posit that persistent peers are the main reason why

Manuscript received November 10, 2017; revised September 15, 2019; accepted September 20, 2019. Date of publication October 11, 2019; date of current version December 23, 2019. This work was supported by the U.S. National Science Foundation under Grant CCF-1617673. This article was presented at the 2017 IEEE International Conference on Computer Communications.

The authors are with the School of Electrical and Computer Engineering, Cornell University, Ithaca, NY 14853 USA (e-mail: ob65@cornell.edu; wagner@ece.cornell.edu).

Communicated by L. Ying, Associate Editor for Communication Networks.

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIT.2019.2946656

many experimental works evaluating actual trace logs of large p2p networks find their performance to be satisfactory.

As per capita media consumption increases [7] and moves to wireless devices [1], however, one expects the incidence of persistent peers to decline, for three reasons. First, data consumption on wireless devices is generally priced differently than for wired terminals. In particular, most wireless providers in the U.S. charge users or lower users' speed when their data usage exceeds a certain threshold, whereas most wired data plans allow for unlimited data transfers, constrained only by the rate. In the wireless case, uploading is counted toward the overall data usage as much as downloading is, and this constitutes a substantial incentive for a mobile p2p user to behave non-persistently. Second, power consumption is a significant concern for mobile users, and hence it is less likely for a mobile p2p user to dwell in the system after the download is complete than it is for a user of a wired system. Finally, increases in per-user data consumption [7], in the absence of attending increases in network capacity, will strain users' access and discourage them from being persistent peers. One should note that it is difficult for a protocol to penalize peers for being non-persistent since, by definition, non-persistent peers have left the p2p network and, therefore, lie outside its province.

There have been noteworthy attempts to devise stable protocols with non-persistent peers and arbitrary arrival rates. Reittu [8] suggests that all the peers sample three random peers and download a piece chosen uniformly at random from the set of pieces that are found in exactly one of these three peers, and they skip if the set is empty. Experimental results appear to demonstrate the stability of the protocol when the system with one fixed seed begins with the extreme case of all other peers lacking only one common piece, although no proof was provided for stability. In a follow-up work, Norros *et al*. [9] showed the global stability of this protocol under large deterministic system limits when the file is composed of two pieces. Later, global stability of the underlying stochastic system for arbitrary number of pieces was proved by Oğuz *et al*. in [10]. In the same work, Oğuz *et al*. [10] also provide a new provably stable scheme in which only new peers that arrive with no piece apply the rule in [8] and the peers lacking only one piece contact m peers and download the last piece only if every piece they have shows up at least twice in aggregate piece profile of these m peers. The protocol requires all other peers to contact a randomly chosen peer and download a piece chosen uniformly at random. The implementation of such a protocol burdens the one club peers in the sense that they are not only prevented from downloading the last piece and consequently departing, but also they are required to keep uploading to the network while waiting to obtain the last piece. Thus, concerns for power consumption of mobile devices and internet data usage may make this protocol unappealing to some p2p users. The same concern can be raised for the protocol in [8].

All of the above-mentioned protocols either rely on persistent peers [6] or on the willingness of peers to decline a useful piece that is offered to them [8]–[10]. We introduce a protocol that requires neither of these assumptions: the peers may be non-persistent, and they always download a piece when they connect with a peer who has a piece that they lack; we call peers that exhibit the latter behavior *opportunistic*. Instead, we modify the behavior of the transmitting peers: peers decline to transmit a piece when the transmission of this piece would harm stability. When all peers have perfect knowledge of the state of the network, it makes no difference whether the uploading peer or the downloading peer is charged with preventing undesirable piece exchanges. When there is imperfect state information, however, the uploading peer and downloading peer can have different views of the network and might reach different conclusions about the desirability of transmitting a particular piece. In such situations, it is preferable to rely on the uploading peer to decide whether the piece should be transmitted, due to a fundamental asymmetry in the roles of the two peers: the downloading peer has more at stake than the uploading peer does in the transaction. This is especially true when the downloading peer has all but one piece and the uploading peer has the piece that the downloading peer lacks. A downloading peer in this situation has a great incentive to accept the last piece, irrespective of what the protocol dictates, thereby completing its copy and allowing it to leave the network before its misbehavior can be detected.

Recognizing that the formation and persistence of the one club lies at the heart of instability as discussed in [4], we introduce the *group suppression protocol* that achieves stability by stopping the one club from recruiting new members. This goal is accomplished in the following way: First define a *group* as a collection of peers that share the same piece profile. And a group of peers whose population is strictly greater than any other group of peers in the network is called *the largest club*. The group suppression protocol dictates that a peer from the largest club uploads only to peers who hold greater number of pieces than it does and to refuse the upload to all the other peers when it contacts them. An upload from a peer in the largest club to any peer holding more pieces is allowed because such an upload does not draw new members into the largest club. To compare the group suppression protocol to the one suggested by Oğuz *et al*. in [10], consider a scenario in which the largest club consists of peers with all the pieces but one common piece, which is also the one club. In the former protocol, the largest club members download the last piece and leave as soon as they encounter a peer that has the last piece. Furthermore, they upload nothing since there are no peers that are eligible to receive a piece from them. In the latter protocol, when the largest club peers encounter the last piece, they may be prohibited from downloading it based on the piece profile of their m contacts, and they continue uploading while waiting for another contact. Also, when the largest club is very crowded, its peers' waiting times are consequently prolonged. The group suppression protocol compensates the largest club for their extended wait time in the network by allowing them to reduce their upload rate, which constitutes another desirable feature of the protocol.

In a similar spirit to the group suppression protocol, throttling the upload capacity of peers or even servers has been proposed to increase the performance of p2p systems.

Zhang *et al*. [11] considered a hybrid p2p system with a file consisting of a single piece and established via a fluid model that throttling the server capacity under certain conditions minimizes the number of peers compared with utilizing the entire capacity of the server. Their model, however, does not capture the missing piece syndrome. de Souza e Silva *et al*. [12] propose a protocol in which all the peers lacking only one piece, not necessarily the rarest piece, reduce their upload rate. Although this change in their closed system analysis where a departure causes an arrival leads to substantial gain in the maximum achievable throughput, the scheme requires fine-grained tuning. Subsequent to the publication of the conference version of this work, Reddyvari *et al.* [13] introduced a similar protocol, called mode-supression, that, like group-suppression, aims to block dissemination of common pieces. Mode-suppression yields similar sojourn times to group-supression in unstructured p2p networks. Although mode-suppression is provably stable for an arbitrary number of pieces, its incorporation into practical protocols such as BitTorrent is not explored in [13]. On the other hand, we show via simulations that, under certain assumptions, BitTorrent's official protocol does suffer from the missing piece syndrome and then demonstrate that straightforward incorporation of the group suppression protocol into BitTorrent, while retaining most of BitTorrent's core mechanisms, alleviates the missing piece syndrome.

Other related work is presented in I-A. A detailed system model, the stability theorem for two pieces, the stability conjecture for arbitrary number of pieces, and a discussion of the proof technique are given in Section II. Decentralized variant of the group suppression protocol is defined in Section III. Section IV offers simulation results confirming the stability theorem, the stability conjecture and the stability of the decentralized group suppression protocol. Section V contains the description and the simulations of BitTorrent modified to include the group suppression protocol. Finally, a discussion of contributions and findings as well as some future research directions is provided in Section VI.

A. Other Related Work

Massoulié and Vojnovic [14] proved global stability in a model with non-persistent peers if all new peers arrive into the system with one uniformly random piece of the file uploaded by the fixed seed. The fixed seed's upload rate, however, may not be large enough to allow this in practice.

One of the earliest p2p models suitable to analytic analysis appeared in [15]. The file consists of only one piece, which leads to a system that distinguishes peers based on whether they are seeds or not, rather than based on what fraction of the file they have. Furthermore, all incomplete peers are assigned the same effectiveness parameter and the incomplete peers exit the network according to a certain rate after becoming a seed, which means some of the peers are persistent. Under this model, they proved local stability of the associated deterministic system. Qiu and Sang [16] later established global stability under the same model. The model in [15] was extended by [17] and [18].

Considerable attention has been paid to improving the stability properties of p2p networks by connecting different *swarms* where swarm refers to the set of all peers that want to download the same particular file [19]–[21]. Bundling a number of files has been shown to increase the content availability and this benefit overcomes the burden of downloading unnecessary content for peers interested in unpopular files, which can be seen in reduced download time in [21]. Lacking any seeds, a network with non-persistent peers who aim to obtain some distinct files of size one was considered by Zhou *et al*. in [20]. Peers with caches large enough to hold all the files enter the network with a file that some of the other peers are interested in. One notable result of the paper is that at most one swarm out of all swarms in the network is unstable. Zhu *et al*. [19] introduce a model in which, unlike [20], there exists a fixed seed, and non-persistent peers join the network empty, and the sets of pieces corresponding to distinct swarms may or may not be disjoint [19]. In case of disjoint swarms, they determined that if the maximum of the arrival rates of all swarms are less than the upload rate of seed, then the overall system is stable and if it exceeds the upload rate of seed then the overall system is unstable. Zhang *et al*. [22] propose forming coalitions between peers that paves the way for different peer and piece selection policies that may surpass the performance of BitTorrent's peer and piece selection policies.

II. Problem Setup, The Group Suppression Protocol and Main Theorem

We first describe the protocol used in Hajek and Zhu [4]. We then define the group suppression protocol by describing how it deviates from the Hajek and Zhu protocol.

In Hajek and Zhu's model [4], which will be referred to as *the unstructured p2p protocol*, the file to be downloaded is divided into k pieces. Arrivals follow a Poisson process of rate λ. The new peers do not carry any pieces. The fixed seed remains in the system. All of the other peers are assumed to be non-persistent. The peer that initiates the contact uploads a piece to the other peer. In other words, the model employs a push policy instead of a pull policy. The uploading peer applies the "random useful piece selection" policy to determine the piece to upload, which amounts to selecting a piece among all useful pieces uniformly at random. If a contact occurs and the uploading peer has at least one piece that the other peer does not possess, we assume the upload of the randomly chosen piece takes place instantaneously. The time at which an incomplete peer contacts another peer is determined by Poisson processes with rate μ independent from an incomplete peer to incomplete peer. On the other hand, the seed contacts other peers according to a Poisson process of rate U_s. When it is time for a peer to contact another peer, it uniformly picks a peer from the set of all peers, which is called random peer selection policy.

We now build the group suppression protocol upon the unstructured p2p protocol by employing two modifications to it. The first and arguably less significant modification requires the fixed seed to select a peer uniformly at random among the peers with the least number of pieces rather than uniformly

at random among all peers and to upload a random useful piece. This seed policy is usually called the most deprived policy. It is sometimes combined with another rule that the seed upload the rarest piece rather than a random piece, which implies the seed has access to every peer's piece profile in the network [12], [23]. Our group suppression protocol allows the seed to upload a random useful piece.

To understand the second modification, recall from Section I that two peers belong to the same group if their piece profiles exactly match each other and the strictly most populous group is designated as the largest club. We assume for now that peers have access to this central knowledge. Notice that the network may not necessarily have a largest club. The backbone of the group suppression protocol relies on preventing a group of peers from dominating the network. In order to accomplish that, the second deviation from the unstructured p2p protocol, which earns the group suppression protocol its name, requires the peers in the largest club to deny the upload to any peer with a smaller or equal number of pieces whenever they initiate a contact with such a peer.

The group suppression protocol contributes to stability by curbing the growth of the largest club and by concentrating the seed's upload on the peers with the least number of pieces. Since the largest club rapidly recruits other peers once it starts to dominate the network, stopping the largest club requires eradication of its source of growth, which is uploads from the largest club to peers who hold a subset of the largest club's pieces, or more generally, all the peers except the ones holding a greater portion of the file. The peers with greater number of pieces than the largest club peers cannot be absorbed into the largest club. Therefore, the largest club is allowed to utilize its upload capacity for these peers but not for others. The protocol's rule concerning the seed originates from the intuition that if the seed uploads a rare piece to a peer, it is preferable that the peer be one who will remain in the system for as long as possible. A peer with the least number of pieces is presumably more likely to stay longer in the network than other peers.

We state the theorem regarding the stability of the group suppression protocol for $k = 2$ and also conjecture the stability of the protocol for arbitrary k.

Main Theorem: If the group suppression protocol is employed and $k = 2$, then the continuous-time Markov chain representing the network is positive recurrent for any $(\lambda > 0, U_s > 0, \mu > 0)$. That is, the p2p network with the group suppression protocol and $k = 2$ is stable for any $(\lambda > 0, U_s > 0, \mu > 0)$.

Conjecture 1: Under the group suppression protocol for arbitrary k, the continuous-time Markov chain representing the network is positive recurrent for any $(\lambda > 0, U_s > 0, \mu > 0)$. That is, the p2p network with the group suppression protocol and arbitrary k is stable for any $(\lambda > 0, U_s > 0, \mu > 0)$.

Sketch of the proof of Main Theorem: We employ the Foster-Lyapunov proposition to prove positive recurrence of the underlying Markov chain. To state the proposition, we need to define the drift of a potential function $V(\boldsymbol{s})$:

$$DV(\boldsymbol{s}) = \sum_{\boldsymbol{y}:\boldsymbol{y}\neq\boldsymbol{s}} q(\boldsymbol{s}, \boldsymbol{y})[V(\boldsymbol{y}) - V(\boldsymbol{s})], \tag{1}$$

where $q(\boldsymbol{s}, \boldsymbol{y})$ is the transition rate from state $\boldsymbol{s}$ to $\boldsymbol{y}$.

Foster-Lyapunov Proposition [24]**:** Let $\boldsymbol{\phi}$ be a time homogeneous, irreducible and continuous time Markov process and $\boldsymbol{S}$ be its state space. If there exists a finite set of states $\boldsymbol{C} \subset \boldsymbol{S}$, a potential function $V(\boldsymbol{s}) : \boldsymbol{S} \to (0, \infty)$ and some constants $b > 0$, $\epsilon > 0$, such that

$$\{\boldsymbol{s} : V(\boldsymbol{s}) < K\} \text{ is finite } \forall K \tag{2}$$

$$DV(\boldsymbol{s}) \leq -\epsilon + b\mathbf{1}_C(\boldsymbol{s}) \quad \forall \boldsymbol{s} \in \boldsymbol{S} \text{ where}$$

$$\mathbf{1}_C(\boldsymbol{s}) \triangleq \left\{ \begin{array}{ll} 1, & \text{if } \boldsymbol{s} \in C \\ 0, & \text{if } \boldsymbol{s} \notin C \end{array} \right\}, \tag{3}$$

then $\boldsymbol{\phi}$ is positive recurrent.

First note that the underlying Markov process for p2p network at hand is time homogeneous, irreducible and continuous time. To describe the particular potential function that satisfies Foster-Lyapunov proposition, we introduce the following notation for the system with $k = 2$ pieces:

type 0 peer: a peer that has no pieces
type 1 peer: a peer that has piece 1 but not piece 2
type 2 peer: a peer that has piece 2 but not piece 1
n_0 : the number of *type 0* peers.
n_1 : the number of *type 1* peers.
n_2 : the number of *type 2* peers.
$\boldsymbol{s}$: the state (n_0, n_1, n_2).
s : the number of peers in the system.

It is convenient to consider the fixed seed as not part of the system. Therefore, we have $s = n_0 + n_1 + n_2$

We choose the potential function as follows:

$$V = (a^+)^2 + (b^+)^2 + d^2, \tag{4}$$

where

$$a = n_0 + \min(n_0, n_1) + c_1(n_1 - n_0)^+ - c_2 n_2$$
$$b = n_0 + \min(n_0, n_2) + c_1(n_2 - n_0)^+ - c_2 n_1$$
$$d = c_3 n_0 + c_4 n_1 + c_4 n_2.$$

In the Appendix, where we postpone the rest of the proof of the Main Theorem, we specify a finite set of states $\boldsymbol{C}$ and a set of conditions on constants c_1, c_2, c_3, c_4, p for every (λ, U_s, μ), and then show that under the group suppression protocol, the potential function $V(\boldsymbol{s})$ in (4) and the specified $\boldsymbol{C}$ satisfies Foster-Lyapunov Proposition for any $(\lambda > 0, U_s > 0, \mu > 0)$.

We now discuss the choice of the potential function $V(\boldsymbol{s})$ in (4). As the complete reasoning behind the construction of the potential function is rather involved, consider a simple scenario where $c_1 = 1, c_2 = 1, c_3 = 1, c_4 = 1$, although this choice of constants violates the conditions imposed on these constants in the Appendix. Then, the elements of the potential function become:

$$a = n_0 + n_1 - n_2$$
$$b = n_0 + n_2 - n_1$$
$$d = n_0 + n_1 + n_2.$$

Then a is the *excess demand* for piece 2, i.e., the number of peers that demand piece 2 minus the number of peers that can supply it. Likewise, b is the excess demand for piece 1. The a and b terms in the Lyapunov function therefore penalize lopsided network states in which there is a dominant largest club. The d term evidently penalizes the overall network size.

III. The Decentralized Group Suppression Protocol

The group suppression protocol might appear to require a certain level of centralization because the seed must find the peers with smallest number of pieces in the system, and every peer needs to determine if they belong to the largest club when they contact another peer. However, both elements of centralization in the protocol can be relaxed for practical implementations of p2p systems. To remove centralization, we now define *the decentralized group suppression protocol*. The protocol is built on the unstructured p2p protocol as follows: Every incomplete peer considers the piece profiles of the last three peers it has contacted along with its own piece profile. If the multiplicity of its own piece profile is strictly more numerous than that of any other piece profiles in this set, then it designates itself as a largest club member. Note that any peer will always have at least two piece profiles including its own piece profile in that set. Second, the fixed seed receives the id of peers upon their arrival and stores the most recent 5 arrivals' ids, the most recent ranking first, and uploads to the highest ranking peer that is still in the network. If all 5 peers appear to have left and no arrival has happened yet, the fixed seed picks a peer uniformly at random among all peers. Thus, the decentralized group suppression protocol handles the largest club membership decision in a distributed fashion for incomplete peers and offers the fixed seed a decentralized way of finding the peers that are more likely to be among the peers with the smallest portion of the file. It should be noted that the effectiveness of both features of the protocol rely on the uniform peer selection policy.

IV. Simulations for the Main Theorem and Conjecture 1

This section simulates the unstructured p2p protocol and the group suppression protocol defined in Section II, and the decentralized group suppression protocol defined in Section III.

The following initial conditions and specifications of rates apply to both protocols: The seed contacts the other peers according to a Poisson process of rate 2. Every incomplete peer's contact process follows according to an independent Poisson process of rate 1. The seed's upload rate is chosen to be 2 instead of 1 because such choice captures the fact that the seed does not have to download anything and, therefore, it is likely to allocate more bandwidth to the upload than the incomplete peers. Moreover, increasing the seed's upload rate makes it more challenging for the largest club to pull the network towards instability, which nevertheless will happen when the unstructured p2p protocol is simulated. The simulations start with 500 peers at $t = 0$ and all of the peers except the fixed seed possess all the pieces of the file but the one common piece. And all incomplete peers depart the network as soon as they obtain the entire file. All simulations were done in Matlab.[1]

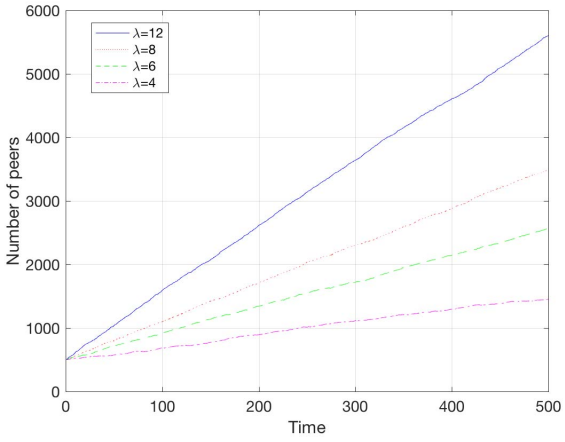

Fig. 1. The total number of peers in the network with the unstructured p2p protocol when $k = 2$ and λ is varying.

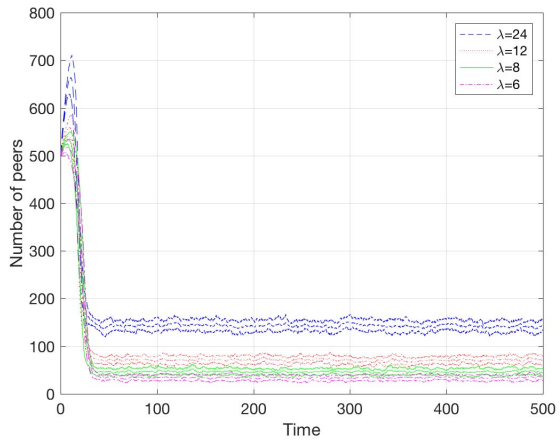

Fig. 2. The middle line in a color shows the mean number of peers in the network with the group suppression protocol when $k = 2$ and λ is varying. The remaining two lines in that particular color represent the mean $\pm$ the standard deviation.

Fig. 1-4 are intended to verify the Main Theorem and therefore the file is divided into only two pieces. Instability is seen in Figure 1 where the unstructured p2p protocol is applied and λ is varied. Since the arrival rate λ is greater than the seed's upload rate U_s, instability is expected and the result is consistent with the main theorem of Hajek and Zhu [4]. In Figure 1, the networks roughly grow with rate $\lambda - U_s$, which is consistent with the understanding that almost no peer except the seed can help another exit the system by supplying the missing piece. In fact, Figure 3 bears more direct evidence for the claim as the line that represent the number of the largest club peers closely trails the line that represents the total number of peers in the system. The total number of peers is shown in Figure 2 for varying λ values

[1]The codes used to generate the simulations are provided at provided at https://github.com/omerbilgen/p2p_project.

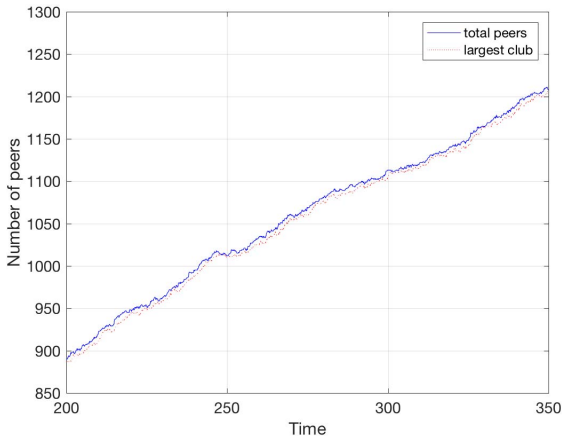

Fig. 3. Comparison of the total number of peers to population of the largest club peers in the network with the unstructured p2p protocol when $k = 2$ and $\lambda = 4$.

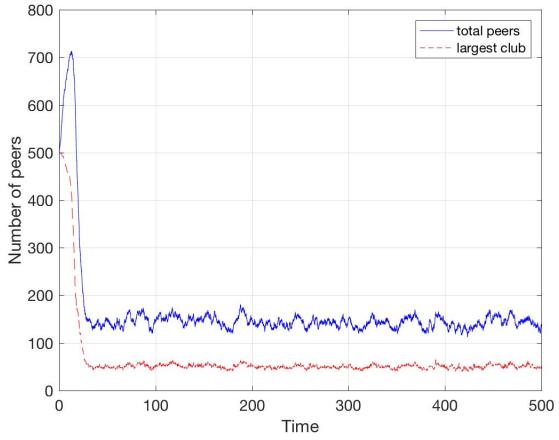

Fig. 4. Comparison of the total number of peers to population of the largest club peers in the network with the group suppression protocol when $k = 2$ and $\lambda = 24$.

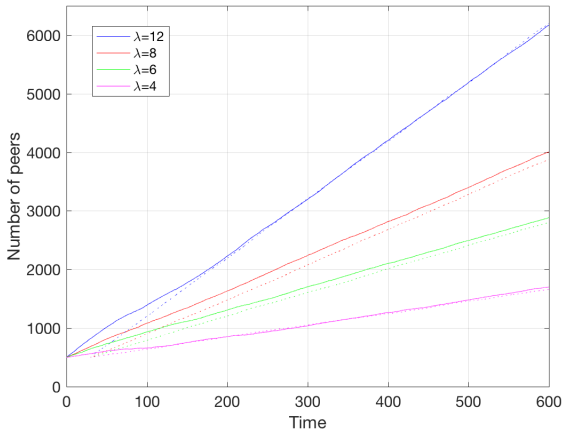

Fig. 5. When the unstructured p2p protocol is in effect with varying λ, the total number of peers are represented in solid line for $k = 48$ whereas the dashed line is a 30-time unit delayed version of the total number of peers for $k = 6$.

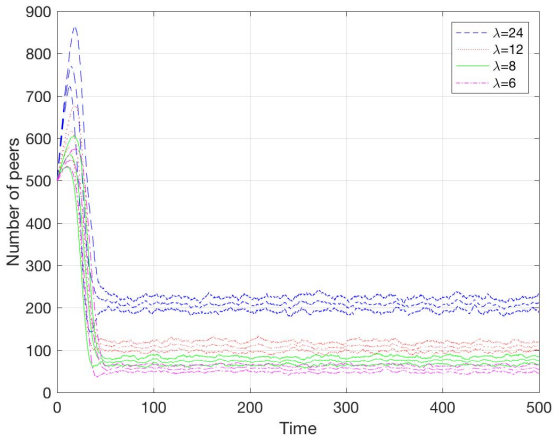

Fig. 6. The middle line in a color shows the mean number of peers in the network with the group suppression protocol when $k = 6$ and λ is varying. The remaining two lines in that particular color represent the mean $\pm$ the standard deviation.

when the group suppression protocol is applied. The networks for varying arrival rates, which are larger than the upload rate of the seed, recover from very skewed piece distribution and achieve stability. Figure 4 shows both the total number of peers and the largest club population when $\lambda = 24$. The apparent increases in the population of the largest club apparently give rise to the sudden jumps in the total number of peers. In the same vein, the downward trend in the population of the largest club is followed by the same trend in the total number of peers. Between the growth and the decline of the largest club, the group suppression protocol cuts off the supply of pieces from the largest club to any peer who does not hold a greater portion of the file, giving those peers more time to obtain and distribute the rarest piece without being absorbed into the largest club. Note that while Fig. 4 and 9 demostrate single realizations, all of the other figures in this section represent an average of 20 individual realizations of their corresponding stochastic network accompanied by one standard deviation above and below the average.

Conjecture 1 in Section II states that any p2p network with the group suppression protocol is stable regardless of the number of the pieces the file is divided into. Figure 5, where the unstructured p2p protocol is in effect, provides experimental evidence that when λ is greater than U_s, the missing piece syndrome phenomenon develops irrespective of the number of pieces k. Furthermore, the number of pieces in the file appears not to affect the slope of the total number of peers in the network because the total number of peers roughly grows with $\lambda - U_s$ per time unit for any k in Figure 5. The group suppression protocol is put into practice in Fig. 6-7 for various (λ, k) pairs. The stability follows a transient period and prevails permanently in all of different (λ, k) pairs. Therefore, Fig. 6-7 provide evidence that the stability conjecture holds true.

The simulations shown in Fig. 8-11 indicate that the decentralized group suppression protocol is stable for all (λ, k) pairs considered. Comparing the plots for the decentralized suppression protocol with those from the centralized version, we see that the former display more variability. Apart from this difference, the plots are similar. The only outliers are Fig. 8-9. First of all, the decentralized group suppression protocol in Fig. 8, where $k = 2$ and λ is large, appears to be

TABLE I

AVERAGE SOJOURN TIMES FOR DIFFERENT PROTOCOLS WHEN $\lambda = 6$ AND k IS VARIED

Protocol	Average sojourn time for $k = 25$	Average sojourn time for $k = 50$	Average sojourn time for $k = 100$
Group suppression protocol	28.95	54.50	106.11
Decentralized group suppression protocol	29.12	54.60	105.39
Waiting protocol [6]	29.18	54.92	105.69
Forced Friedman protocol [8]	30.54	55.89	105.84
Common chunk protocol [10] for m=5	30.39	57.63	110.92
Common chunk protocol [10] for m=3	36.88	67.39	126.02

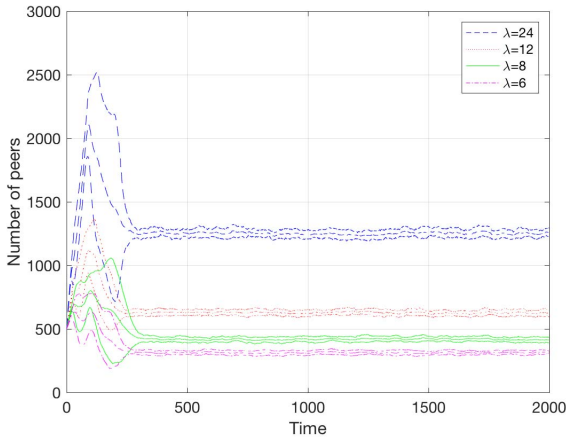

Fig. 7. The middle line in a color shows the mean number of peers in the network with the group suppression protocol when $k = 48$ and λ is varying. The remaining two lines in that particular color represent the mean $\pm$ the standard deviation.

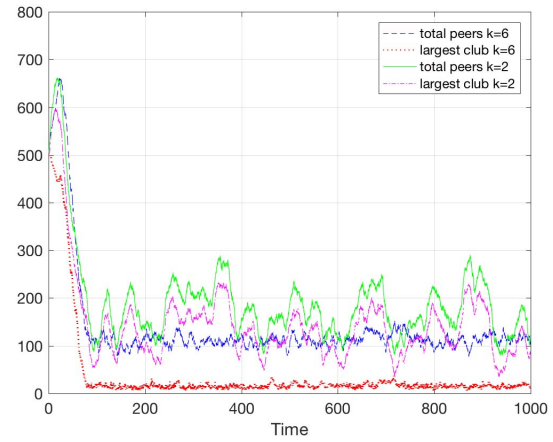

Fig. 9. Comparison of the total number of peers to the largest club when the decentralized group suppression protocol is in effect with $\lambda = 12$ and $k = 2, 6$.

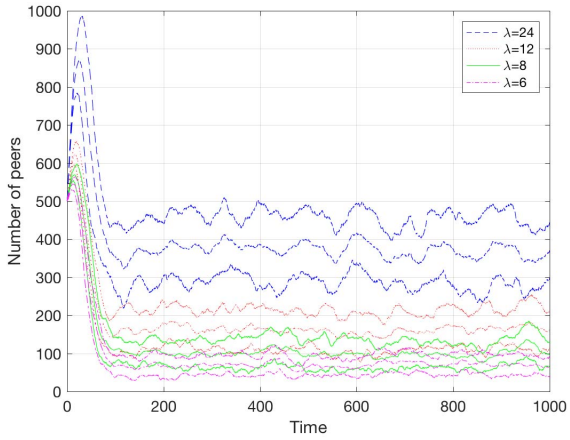

Fig. 8. The middle line in a color shows the mean number of peers in the network with the decentralized group suppression protocol when $k = 2$ and λ is varying. The remaining two lines in that particular color represent the mean $\pm$ the standard deviation.

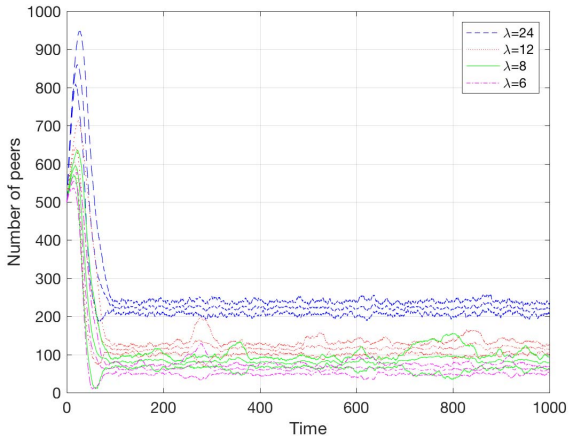

Fig. 10. The middle line in a color shows the mean number of peers in the network with the decentralized group suppression protocol when $k = 6$ and λ is varying. The remaining two lines in that particular color represent the mean $\pm$ the standard deviation.

more prone to spikes compared to the same setting with the group suppression protocol in Fig. 2 even though we take the average of 20 realizations. It stems from the higher probability of error in deciding the largest club membership due to the small number of type of peers in the network. Comparing Fig. 4 to Fig. 9, both of which demonstrate single realizations, underlines the difficulty of controlling the largest club when $k = 2$, which signals the largest club decision mistakes affect the network so as to create substantial ripples. Nevertheless, the largest club of the decentralized group suppression protocol behaves smoothly in Fig. 9 when k is increased to 6. Since k is usually large and the decentralized group suppression's goal is intended to be deployed for practical use, the spikes observed for $k = 2$ should not constitute a basis for concern. Indeed, none of average realizations in Figs. 10-11 for the decentralized protocol for k larger than 2 exhibit the spikes observed in Fig. 8.

TABLE II
PERCENTAGES OF BLOCKED UPLOADS BY THE GROUP SUPPRESSION PROTOCOL WHEN $\lambda = 8$ AND k IS VARIED

	$k = 2$	$k = 3$	$k = 6$	$k = 12$	$k = 24$
Percentage of uploads blocked by the group suppression protocol	$28.40\ (\pm 14\times 10^{-2})$	$18.82\ (\pm 15\times 10^{-2})$	$7.65\ (\pm 17\times 10^{-2})$	$1.70\ (\pm 6.5\times 10^{-2})$	$0.37\ (\pm 5.06\times 10^{-4})$

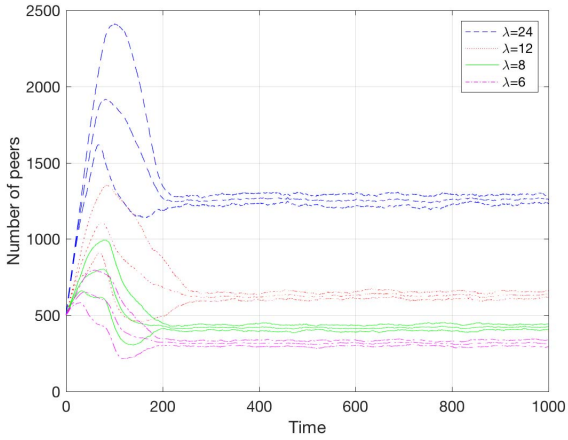

Fig. 11. The middle line in a color shows the mean number of peers in the network with the decentralized group suppression protocol when $k = 48$ and λ is varying. The remaining two lines in that particular color represent the mean $\pm$ the standard deviation.

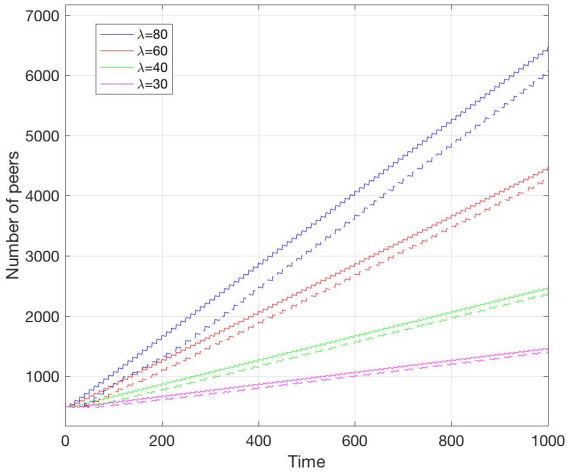

Fig. 12. In the network governed by the BitTorrent-like protocol, the total number of peers are represented in solid lines for $k = 12$ while the dashed lines show 30-second delayed versions of the total number of peers for $k = 48$.

After stability, the arguably most important measure of p2p system performance is the mean sojourn time. We compare different protocol's sojourn time performances to that of the group suppression protocol and to that of the decentralized group suppression protocol in Table I. One of the other protocols is due to Zhu and Hajek [6], which is referred to as the waiting protocol in Table I. This protocol assumes the peers stay in the network on average long enough to upload one piece after they become seeds in our implementation of the waiting protocol. The second protocol in the comparison is Oğuz *et al.*'s common chunk protocol in [10], which is simulated for both when the number of the contacts m that peers lacking only one piece make is three and five. The final protocol simulated is Reittu's forced Friedman protocol [8], which requires every peer to sample three other peers and allows the peer to download a random piece from the set of pieces that show up exactly once in the aggregate piece profile of these three sampled peers. If the set turns out to be empty, the peer skips without downloading any piece. We obtained 100 individual realizations for each protocol where the simulations started with 499 peers holding all pieces but one common piece and a fixed seed, and $k = 25, 50, 100$ and $\lambda = 6$. We also set $U_s = \mu = 1$ because the common chunk protocol and forced Friedman protocol do not distinguish between the upload capacity of a seed and an incomplete peer. In every realization, we collected the sojourn times of the first 500 peers to exit the system after we had waited 2000 time units, which we observed was enough for all networks to reach steady-state. The results in Table I show that the first four protocols including the group suppression protocol and the decentralized group suppression protocol have very close average sojourn times for all k values. Although the average sojourn times of both the waiting protocol and the forced Friedman protocol are on par with that of the group suppression protocol and the decentralized group suppression protocol, it is important to recall that the former relies on persistent peers and the latter depends on non-opportunistic peers. Both the group suppression protocol and the decentralized group suppression protocol assume non-persistence and opportunistic behavior. It is also notable that decentralizing the group suppression protocol does not lead to a sojourn time performance penalty.

To bring stability to a p2p network, the group suppression protocol blocks some of the piece uploads that would have otherwise occurred. It may be of interest to determine what percentage of the potential uploads, defined as contacts in which the contacting peer has at least one piece that the contacted peer lacks, are blocked by the group suppression protocol. Table II shows the result of running 20 different realizations for 20000 time units. As k increases, the percentage of blocked uploads rapidly declines. That $k = 2$ has the largest percentage of blocked uploads can be interpreted as it being the most challenging case to stabilize. Of course, performance is ultimately measured via stability and sojourn time.

V. MODELS OF THE MODIFIED BITTORRENT PROTOCOLS AND THEIR SIMULATIONS

The section is organized as follows: First, the official BitTorrent protocol's fundamental algorithms, namely the rarest first and the unchoking algorithms, are briefly explained. Then a simplified yet rather loyal implementation of the official BitTorrent protocol, which lends itself to simulations

and we call the *BitTorrent-like protocol*, is introduced. We then construct *the BitTorrent-like protocol with group suppression* by incorporating the features of the group suppression protocol into the BitTorrent-like protocol. The section is concluded with simulation results of the BitTorrent-like protocol and the BitTorrent-like protocol with group suppression, all of which were done in Matlab.

Since BitTorrent's source code is no longer public, we consider one of the older versions of BitTorrent, namely version 4.20.0. In the official BitTorrent protocol, every peer maintains a set of peers, which are usually called the neighbor peers. The number of the neighbor peers cannot exceed some particular number. If a peer's number of neighbor peers falls below a pre-defined threshold, then that particular peer requests new peers from the tracker of the torrent. The file is divided into pieces and each piece is broken into a certain number of blocks. A peer cannot offer a partial piece. Furthermore, every peer knows the individual piece profiles of its neighbor peers at any time although they do not possess such knowledge for other peers in the network. Peer A is said to be unchoked by Peer B if Peer B decides to fulfill any download request Peer A may make. Peer A is called an interested peer by Peer B if Peer B has some piece that Peer A may want to request. The rarest first policy is employed, which simply means that when the peer is unchoked by one of its neighbors, the peer first requests from the neighbor the rarest piece among the pieces needed by the peer. The rarest piece is determined by the downloading peer based on all of its neighbors' piece profiles. Both the seed and incomplete peers could upload to a certain number of peers, which is at most and by default four. The unchoking policy varies based on whether the seed or an incomplete peer is unchoking. In every unchoking round of 10 seconds, an incomplete peer ranks all the interested neighbor peers in a descending order based on their upload rates to this incomplete peer and unchokes the first three peers. Once in every three rounds, an incomplete peer selects one additional peer among its neighbors uniformly at random and keeps it unchoked for three rounds, which is called optimistic unchoking. On the other hand, the seed ranks all the unchoked interested neighbor peers that have been unchoked in last 20 seconds based on when they have been unchoked, the latest being the first. In the first two rounds of seed's unchoking, it allocates one of its four upload slots to a uniformly chosen peer and the remaining three slots are filled from the top of the list of lately unchoked peers. In the last of every three rounds, all four unchoked peers are kept. The rarest first and the unchoking rules are the main policies of BitTorrent. Yet, there exist other rules governing BitTorrent networks such as strict priority, random first and endgame mode. Strict priority refers to requesting the remaining blocks of a partial block before making a request on the blocks of a new piece. According to random first policy, peers with less than four pieces select the piece to download randomly rather than based on rarity. Then they switch to the rarest first policy when they obtain four full pieces. Endgame mode kicks in when all the blocks have been requested by a peer. The mode enables the peer to request all the remaining blocks from every neighbor that is unchoking the peer and it also allows the peer to be interested in all the neighbors

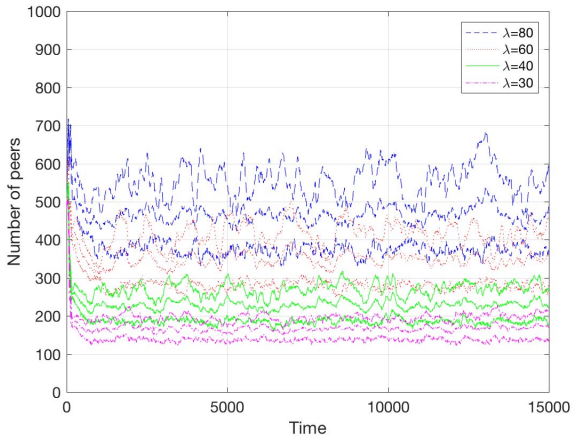

Fig. 13. The middle line in a color shows the mean number of peers in the network governed by the BitTorrent-like protocol with group suppression when $k = 12$ and λ is varied. The remaining two lines in that particular color represent the mean $\pm$ the standard deviation.

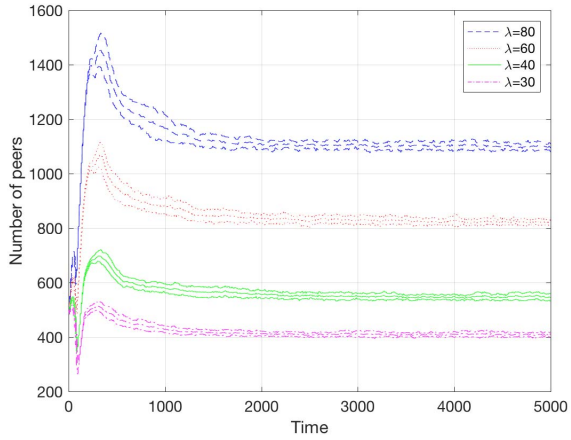

Fig. 14. The middle line in a color shows the mean number of peers in the network governed by the BitTorrent-like protocol with group suppression when $k = 48$ and λ is varied. The remaining two lines in that particular color represent the mean $\pm$ the standard deviation.

for the blocks that are currently choking it. Although the explanation of the official BitTorrent protocol here should be sufficient to understand the simulations, a reader is advised to consult [25], [26] and [27] for a deeper understanding of the official BitTorrent protocol.

We now define the BitTorrent-like protocol. The original rarest first and the unchoking algorithms of the official BitTorrent protocol are preserved in the BitTorrent-like protocol although a number of simplifications were made in order to render the large scale simulations practical. First, all the upload times, arrival times and departure times are discretized and synchronous. A fixed number of peers arrives into the system every 10 seconds rather than having random individual arrivals. Also, the upload of any piece takes every incomplete peer exactly 10 seconds and they have four upload slots, meaning that they can upload at most four pieces simultaneously. Every incomplete peer's download and upload speeds are upperbounded by 40 pieces and 4 pieces per 10 seconds, respectively, in order to mimic the incomplete peer's tendency

to allow much more bandwidth for the download than for upload. The seed, too, commands 4 upload slots but it finishes uploading a piece in exactly 2 seconds, which makes the seed's maximum upload speed five times an incomplete peer's maximum upload speed. In other words, the seed's maximum upload speed is 2 pieces per second and an incomplete peer's maximum upload speed is 0.4 pieces per second. Note that we do not apply strict priority, endgame mode and random first policies as our implementation operates on the level of pieces, not on the level of blocks. Out of these three features, one might suspect that endgame mode may influence stability because, after all, the missing piece syndrome results from not being able to download the last piece. However, endgame mode increases only the download speed of the piece but not its availability. Thus, it is reasonable to believe that not implementing endgame mode will not affect stability.

By the following modifications to some features of the original unchoking algorithm, we construct the BitTorrent-like protocol with group suppression: First, the seed ranks all interested neighbor peers in an ascending order of the number of pieces they hold. In case of a tie, the one with the larger upload ratio to the whole network is favored by the seed. It should be noted that this policy change does not add any complexity to the network as the seed already knows the piece profile of every neighbor peer. An incomplete peer first determines whether it belongs to the largest club or not based on the collection of its neighbors' piece profiles. If it does not turn out to be a member of the largest club, then the unchoking algorithm in original BitTorrent protocol proceeds. If it indeed belongs to the largest club, the incomplete peer creates the list according to the original unchoking algorithm of BitTorrent and then updates the list by removing peers that do not have more pieces than this peer. It then unchokes based on the updated list. Again, one should notice that for an incomplete peer to determine the largest club does not require any additional knowledge. Thus, the group suppression fits more naturally into BitTorrent than it does the unstructured p2p network.

The initial state of the simulation for the BitTorrent-like protocol includes one fixed seed, 494 peers holding all the pieces but the last piece and 5 peers possessing only the last piece; so there are 500 peers at $t = 0$. Note that these 5 peers act as an impediment to the largest club's gravitational effect on the new peers not only by supplying the last piece to the largest club and causing them to depart, but also by giving the last piece to the new peers, which guarantees that they will not be absorbed into the largest club. They, thus, have a stabilizing effect. To show the occurrence of the missing piece syndrome for the BitTorrent-like protocol in simulations, the constant number of arrivals per 10 second blocks is chosen from 30, 40, 60 and 80, which are equivalent to saying λ takes on values of 3, 4, 6 and 8 arrivals per second. Note that all values of λ are greater than the seed's maximum upload speed of 2 pieces per second. We also remark that while Fig. 15-16 represent single realizations, all the other figures of this section demonstrate the average of 20 individual realization of their corresponding stochastic network. Fig. 12 shows that the BitTorrent-like protocol falls victim to the missing piece

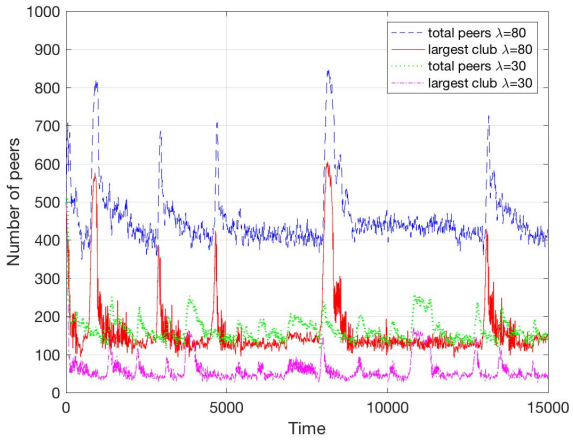

Fig. 15. In this realization of BitTorrent-like protocol with group suppression, $k = 12$ is fixed. Blue and green lines show the total number of peers and the largest club when $\lambda = 80$, respectively, while red and magenta lines show the total number of peers and the largest club when $\lambda = 30$.

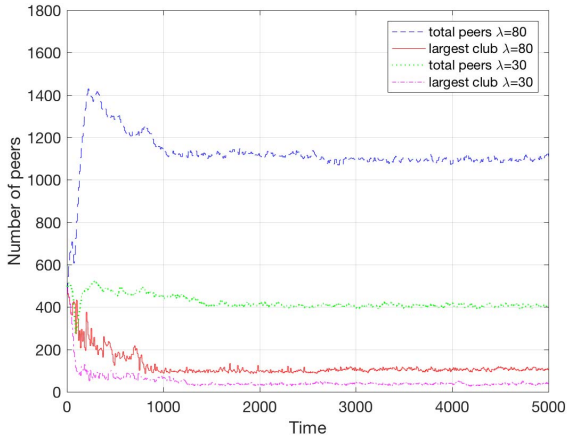

Fig. 16. In this realization of BitTorrent-like protocol with group suppression, $k = 48$ is fixed. The blue and green lines show the total number of peers and the largest club when $\lambda = 80$, respectively, while the red and magenta lines show the total number of peers and the largest club when $\lambda = 30$.

syndrome for all four λ values when k is fixed at both 12 and 48. The total number of peers in Fig. 12 grows roughly with rate $\lambda - U_s$ where U_s is naturally interpreted as the maximum upload speed of the seed per second. In conclusion, the BitTorrent-like protocol can be unstable when the arrival rate of new peers overwhelms the maximum upload speed of the seed, which in turn suggests that the original BitTorrent protocol may suffer from the missing piece syndrome if all peers choose to be non-persistent. We suspect that real-life BitTorrent networks enjoys stability only because of the presence of persistent peers. In fact, another study [28], which was published concurrently with the conference version of this paper [29], demonstrates that the missing piece syndrome unfolds in their closed BitTorrent network in which only a departure causes an arrival.

Aside from the occurrence of instability, there are two more observations worth mentioning regarding the BitTorrent-like protocol although we do not include the corresponding figures. First, increasing the number of pieces k in the file seems to

improve the stability for fixed upload rate of incomplete peers. But, if the upload rate of incomplete peers is made suitably large, then the increased upload capacity of the largest club would overcome the stability enhancing effect of increasing the number of pieces in the file. Second, unlike the unstructured p2p networks, increasing the arrival rate of peers beyond some point paradoxically tips the balance of the BitTorrent-like protocol into stability. This is more likely to be a specific result of discretized nature of arrivals and, therefore, this counter-intuitive observation of the BitTorrent-like network may not hold true for the continuous time arrivals that actually take place in real life BitTorrent.

For various (λ, k) pairs, we provide simulation results on the BitTorrent-like protocol with group suppression in Fig. 13-16. We modify the initial state by starting with 499 peers that hold all the pieces but the last piece and one fixed seed. Note that this tends to decrease the stability of the system. The stable trajectories of the total number of peers exhibited in Fig. 13-16 provides experimental support for the idea that the BitTorrent-like protocol with group suppression, as opposed to the BitTorrent-like protocol, is effective against the missing piece syndrome. The spikes in the total number of peers manifest themselves more intensely when k is smaller while they are insensitive to λ especially when k is large. Since k tends to be large in actual BitTorrent networks, the spikes should not be of any concern when it comes to stability. Yet, better understanding of why such spikes are observed could shed light on how the group suppression protocol achieves stability. It can be seen from examining Fig. 15 that a buildup of the largest club in the network triggers a spike of considerable magnitude in the total number of peers. Then it can be seen in the same figure that the growth of the largest club, thanks to the group suppression protocol, cannot gain further momentum and soon the largest club starts to fade away, paving the way for a steep decline in the total number of peers. Fig. 16 presents further evidence of the correlation of the spikes and the buildup of the largest club. Since no buildup of the largest club takes place after $t = 500$, there exists no spikes in the total number of peers in the figure. Thus, the simulations indicate that the BitTorrent-like protocol with group suppression is stable even the number of pieces is greater than two.

VI. Conclusion

When a p2p system with uniform random peer and piece selection policies is composed of only non-persistent peers and a fixed seed, it was known that the p2p system suffers from instability if the arrival rate of new peers λ exceeds the seed's upload rate U_s [4]. Specifically, the network ends up producing a large group of peers who possess all the pieces except a common piece and the group grows roughly with rate $\lambda - U_s$. Motivated by the anticipation of p2p networks with largely non-persistent peers and the necessity of a stable protocol independent of the arrival rate of new peers and the seed's upload rate, we proposed *the group suppression protocol* that achieves stability with non-persistent peers for arbitrary arrival rate of new peers and arbitrary upload rate of the seed. The group suppression protocol aims to achieve roughly uniform piece distribution over the network by cutting off the upload from the peers with the most abundant piece profile to the peers who possess smaller or equal portion of the file. It was proven to be stable using a suitable Lyapunov function for any pair of the arrival rate of new peers and the upload rate of the seed when the file is divided into two pieces. We also conjectured that the group suppression protocol's stability can be extended to any number of the pieces in the file and we supported this conjecture with simulations.

The group suppression protocol contains some centralization elements, which may limit its practicality. According to the protocol, every incomplete peer needs to determine if it belongs to the largest club. This task cannot be carried out without the knowledge of piece profiles of every other peer in the network. Second, the protocol instructs the fixed seed to contact the peers with the smallest amount of the file. This, too, requires some central knowledge. We removed these centralization requirements by devising *the decentralized group suppression protocol*. In this protocol, an incomplete peer makes a decision on the largest club membership not comparing its piece profile to the entire network but to the last 3 contacts it has initiated. As for the fixed seed, we assume every new arrival provides the fixed seed with its id, which is quite realistic assumption for practical systems. The fixed seed uploads to the latest arrival. It stores the most recent 5 arrivals in case the newly arrived peer have already left the network before the fixed seed's Poisson upload time. The simulations showed that the protocol is stable for all (λ, k) pairs considered.

Hajek and Zhu's instability result [4] was proven in the context of p2p network models with uniform random peer selection policy and uniform random useful piece selection policy. Yet, most of the practical p2p protocols apply more sophisticated policies than uniform random peer selection policy and uniform random useful piece selection policy. Thus, one may suspect that the non-persistent behavior could pose no threat to the stability of some actual p2p networks that are equipped with more refined policies than uniform random selection policies. Called the BitTorrent-like protocol, a discretized BitTorrent protocol based on the official BitTorrent protocol was implemented to see if the practical p2p networks are vulnerable to instability when the peers adopt the non-persistent behavior. Extensive simulations showed that when a large fraction of the network was comprised of the one club peers at the start of simulations and the arrival rate of new peers exceeds the upload rate of the seed, the BitTorrent-like protocol with the non-persistent peers cannot prevent the one club from growing further without any bound, which leads to instability. Thus, this unstable behavior of the discretized BitTorrent raises legitimate questions on the stability of real-life BitTorrent networks if they cease to enjoy the existence of persistent peers.

We implemented the BitTorrent-like protocol with group suppression by employing a small number of changes to the unchoking algorithm of the official BitTorrent protocol. Remarkably, the official BitTorrent protocol readily lends itself into the implementation of the group suppression protocol because it already requires every peer to know the

piece profiles of all of its neighbors, which means that the largest club for a peer can be identified by the evaluation of its neighbors rather than the evaluation of every peer in the network. We ran detailed simulations of the BitTorrent-like protocol with group suppression by varying the arrival rate of new peers and the number of pieces in the file and observed that for all (λ, k) pairs we considered, the BitTorrent-like protocol with group suppression manages to remove the one-club, which comprised the whole network apart from the seed in the beginning. And the protocol does not let any group of peers dominate the network after it removes the one club. These two properties basically point to stability.

APPENDIX

For any given pair (U_s, λ), first pick constants c_1, c_2, c_3, c_4 and p such that the following conditions are satisfied simultaneously:

$$c_1, c_2, c_3, c_4, p > 0 \tag{5}$$
$$0 < p < \frac{1}{2} \tag{6}$$
$$c_4 < c_3 \tag{7}$$
$$2 < c_1 \tag{8}$$
$$3c_1 < c_2 \tag{9}$$
$$pc_2 < 1 - p \tag{10}$$
$$\lambda c_3^2 < [1 - p(1 + c_2)][U_s(1 + c_2) - 2\lambda] \tag{11}$$
$$2(c_1 - 1) < c_4(c_3 - c_4) \tag{12}$$
$$2c_3^2 < (c_1 - 2)^2 \tag{13}$$
$$\lambda c_3^2 < U_s \left(\frac{c_2 - 3c_1 + 4}{12} \right). \tag{14}$$

One can find such constants by first assigning some positive values to c_1, c_3 and c_4 so that the conditions (7), (8), (12), (13) are met. One such choice is (32,20,10) for (c_1, c_3, c_4). Set p to $\frac{1}{2(1+c_2)}$. With this assignment (6) and (10) are satisfied. Finally pick c_2 large enough that (9), (11) and (14) are met.

Before we further delve into the proof of the Main Theorem, some structural observations on the potential function are made to shorten the proof. Define $\tilde{a}$ and $\tilde{b}$ as the new a and b after one of the legitimate transitions occur in the system. Recalling $a = n_0 + \min(n_0, n_1) + c_1(n_1 - n_0)^+ - c_2 n_2$ and $b = n_0 + \min(n_0, n_2) + c_1(n_2 - n_0)^+ - c_2 n_1$, it is obvious that

$$a - 2 - 2c_1 - c_2 \le \tilde{a} \le a + 2 + 2c_1 + c_2 \tag{15}$$
$$b - 2 - 2c_1 - c_2 \le \tilde{b} \le b + 2 + 2c_1 + c_2. \tag{16}$$

These bounds on $\tilde{a}$ and $\tilde{b}$ are very loose and can be easily improved but they are good enough to serve our purposes. Observe that $V(\boldsymbol{s})$ in (4) is nonnegative for all $\boldsymbol{s}$. We proceed to show that $DV(\boldsymbol{s}) \le -\epsilon$ for all sufficiently large $s > 0$ as this almost proves the Main Theorem on its own. Fix some small $\epsilon > 0$.

Note that the group suppression protocol is applied to all cases in the Appendix.

Case 1: $n_0 \ge (1 - p)s$.

Under this case $n_0 \ge (1 - p)s > ps$ and $n_1 \le ps, n_2 \le ps$, which shows $n_0 > n_1, n_2$. Using the definition of a and b and (10),

$$a = n_0 + n_1 - c_2 n_2 \ge (1 - p - pc_2)s > 0 \text{ for all } s > 0, \tag{17}$$
$$b = n_0 + n_2 - c_2 n_1 \ge (1 - p - pc_2)s > 0 \text{ for all } s > 0. \tag{18}$$

By (17), (18), (15) and (16), we have

$$\tilde{a} \ge (1 - p - pc_2)s - 2 - 2c_1 - c_2 \tag{19}$$
$$\tilde{b} \ge (1 - p - pc_2)s - 2 - 2c_1 - c_2. \tag{20}$$

Therefore $(\tilde{a})^+ = \tilde{a}$ and $(\tilde{b})^+ = \tilde{b}$ for all sufficiently large s. We derive $DV(\boldsymbol{s})$ for large s in (21)–(27), shown at the bottom of this page.

Consider the term (24). (24) corresponds to an upload from *type* 2 peers to *type* 0 peers. Using $\tilde{a} > 0$ and $\tilde{b} > 0$ for large s,

$$\begin{aligned} V(\boldsymbol{y}) &= (\tilde{a}^+)^2 + (\tilde{b}^+)^2 + (\tilde{d})^2 \\ &= \tilde{a}^2 + \tilde{b}^2 + \tilde{d}^2. \end{aligned}$$

Thus,

$$\begin{aligned} &V(\boldsymbol{y}) - V(\boldsymbol{x}) \\ &= (\tilde{a}^2 + \tilde{b}^2 + \tilde{d}^2) - (a^2 + b^2 + d^2) \\ &= (\tilde{a}^2 - a^2) + (\tilde{b}^2 - b^2) + (\tilde{d}^2 - d^2) \\ &= [(a - 1 - c_2)^2 - a^2] + [b^2 - b^2] + [(d - c_3 + c_4)^2 - d^2] \\ &= (1 + c_2)(-2a + 1 + c_2) + (c_3 - c_4)(-2d + c_3 - c_4), \end{aligned}$$

and $q(\boldsymbol{y}, \boldsymbol{x}) = \frac{\mu n_2 n_0}{s}$. The other terms can be derived similarly.

$$DV(\boldsymbol{s}) = \lambda \left[(2a + 1) + (2b + 1) + c_3(2d + c_3) \right] \tag{21}$$
$$+ \frac{U_s}{2} \left[(1 + c_2)(-2a + 1 + c_2) + (c_3 - c_4)(-2d + c_3 - c_4) \right] \tag{22}$$
$$+ \frac{U_s}{2} \left[(1 + c_2)(-2b + 1 + c_2) + (c_3 - c_4)(-2d + c_3 - c_4) \right] \tag{23}$$
$$+ \frac{\mu n_2 n_0}{s} \left[(1 + c_2)(-2a + 1 + c_2) + (c_3 - c_4)(-2d + c_3 - c_4) \right] \tag{24}$$
$$+ \frac{\mu n_1 n_0}{s} \left[(1 + c_2)(-2b + 1 + c_2) + (c_3 - c_4)(-2d + c_3 - c_4) \right] \tag{25}$$
$$+ \frac{\mu n_1 n_2}{s} \left[c_2(2a + c_2) + c_4(-2d + c_4) + (-2b + 1) \right] \tag{26}$$
$$+ \frac{\mu n_2 n_1}{s} \left[c_2(2b + c_2) + c_4(-2d + c_4) + (-2a + 1) \right] \tag{27}$$

We claim that the sum of (24) and (26) and the sum of (25) and (27) are nonpositive for large s. To show this for sum of (24) and (26), note that for large s the only positive term inside the brackets of both expressions is $c_2(2a + c_2)$. Yet the first term of (24) eliminates this positive term for large s because

$$\left[\frac{\mu n_2\, n_0}{s}(1+c_2)(-2a+1+c_2)+\frac{\mu n_1\, n_2}{s}c_2(2a+c_2)\right]\le 0$$

for large s. One can justify omitting (25) and (27) in the same fashion. Omitting these terms, we are left with

$$DV(\boldsymbol{s}) \le$$
$$\lambda\left[(2a+1)+(2b+1)+c_3(2d+c_3)\right] \tag{28}$$
$$+\frac{U_s}{2}\left[(1+c_2)(-2a+1+c_2)+(c_3-c_4)(-2d+c_3-c_4)\right] \tag{29}$$
$$+\frac{U_s}{2}\left[(1+c_2)(-2b+1+c_2)+(c_3-c_4)(-2d+c_3-c_4)\right]. \tag{30}$$

We drop the second terms of (29) and (30), which are negative, and rearrange the inequality as follows:

$$DV(\boldsymbol{s}) < a\left[2\lambda - U_s(1+c_2)\right] \tag{31}$$
$$+ b\left[2\lambda - U_s(1+c_2)\right] \tag{32}$$
$$+ 2\lambda c_3 d + k, \tag{33}$$

where k is just some constant. (10) and (11) imply

$$2\lambda - U_s(1+c_2) < 0$$

and observe that

$$\min(a,b) \ge (1-p(1+c_2))s \text{ and } d \le c_3 s.$$

Then,

$$DV(\boldsymbol{s}) < s[1-p(1+c_2)]\left[2\lambda - U_s(1+c_2)\right] + s[1-p(1+c_2)]\left[2\lambda - U_s(1+c_2)\right] + 2\lambda c_3^2 s + k.$$

Since $[1-p(1+c_2)][U_s(1+c_2)-2\lambda] > \lambda c_3^2$ is met by (11), one can reduce the inequality to $DV(\boldsymbol{s}) < -K_1 s + k$ where K_1 is some positive constant. Thus, $DV(\boldsymbol{s}) \le -\epsilon$ for all sufficiently large s.

Case 2: $n_0 \ge n_1 \ge n_2$ and $n_i < (1-p)s$ for every i.

Case 2 implies the following:

$$\tfrac{s}{3} \le n_0 < (1-p)s$$
$$\tfrac{sp}{2} < n_1 \le \tfrac{s}{2}$$
$$0 \le n_2 \le \tfrac{s}{3}.$$

The potential function might reach boundaries and display piecewise linearity calling for careful handling of the drift. Thus, Case 2 will be broken into several subcases.

Case 2.1: Conditions of Case 2 and $n_0 > n_1 + 1$.

Under this case, $\tilde{a}$ and $\tilde{b}$ have the following forms regardless of whichever type of one-step transition occurs:

$$\tilde{a} = \tilde{n}_0 + \tilde{n}_1 - c_2\tilde{n}_2$$
$$\tilde{b} = \tilde{n}_0 + \tilde{n}_2 - c_2\tilde{n}_1,$$

where the tilded n_i's are the new numbers of *type* i peers in the next state. It is important to remember that the n_i's can change by 1 at most in absolute value for one state transition. We now derive the drift in (34)–(40), shown at the bottom of this page.

This inequality differs substantially from the one presented in Case 1. Indeed, the expression in Case 1 was an equality

$$DV(\boldsymbol{s}) \le \lambda\left(\begin{bmatrix}(2a+1) & \text{if } a\ge 0\\ 1 & \text{if } -1\le a<0\\ 0 & \text{if } a<-1\end{bmatrix}+\begin{bmatrix}(2b+1) & \text{if } b\ge 0\\ 1 & \text{if } -1\le b<0\\ 0 & \text{if } b<-1\end{bmatrix}+c_3(2d+c_3)\right) \tag{34}$$
$$+\frac{U_s}{2}\left(\begin{bmatrix}(1+c_2)(-2a+1+c_2) & \text{if } a\ge 1+c_2\\ 0 & \text{if } 0\le a<1+c_2\\ 0 & \text{if } a<0\end{bmatrix}+(c_3-c_4)(-2d+c_3-c_4)\right) \tag{35}$$
$$+\frac{U_s}{2}\left(\begin{bmatrix}(1+c_2)(-2b+1+c_2) & \text{if } b\ge 1+c_2\\ 0 & \text{if } 0\le b<1+c_2\\ 0 & \text{if } b<0\end{bmatrix}+(c_3-c_4)(-2d+c_3-c_4)\right) \tag{36}$$
$$+\frac{\mu n_1 n_0}{s}\left(\begin{bmatrix}(1+c_2)(-2b+1+c_2) & \text{if } b\ge 1+c_2\\ 0 & \text{if } 0\le b<1+c_2\\ 0 & \text{if } b<0\end{bmatrix}+(c_3-c_4)(-2d+c_3-c_4)\right) \tag{37}$$
$$+\frac{\mu n_2 n_0}{s}\left(\begin{bmatrix}(1+c_2)(-2a+1+c_2) & \text{if } a\ge 1+c_2\\ 0 & \text{if } 0\le a<1+c_2\\ 0 & \text{if } a<0\end{bmatrix}+(c_3-c_4)(-2d+c_3-c_4)\right) \tag{38}$$
$$+\frac{\mu n_1 n_2}{s}\left(\begin{bmatrix}c_2(2a+c_2) & \text{if } a\ge 0\\ c_2^2 & \text{if } -c_2\le a<0\\ 0 & \text{if } a<-c_2\end{bmatrix}+\begin{bmatrix}-(2b-1) & \text{if } b\ge 1\\ 0 & \text{if } 0\le b<1\\ 0 & \text{if } b<0\end{bmatrix}+c_4(-2d+c_4)\right) \tag{39}$$
$$+\frac{\mu n_2 n_1}{s}\left(\begin{bmatrix}c_2(2b+c_2) & \text{if } b\ge 0\\ c_2^2 & \text{if } -c_2\le b<0\\ 0 & \text{if } b<-c_2\end{bmatrix}+\begin{bmatrix}-(2a-1) & \text{if } a\ge 1\\ 0 & \text{if } 0\le a<1\\ 0 & \text{if } a<0\end{bmatrix}+c_4(-2d+c_4)\right) \tag{40}$$

while we have inequality here. Therefore, a brief explanation is in order. Consider the first term of (34). (34) corresponds to an arrival of a *type* 0 peer and the first term in parenthesis of (34) corresponds to $(\tilde{a}^+)^2-(a^+)^2$. Observe that $\tilde{a}=a+1$ in this case. Therefore,

$$\begin{aligned}(\tilde{a}^+)^2-(a^+)^2&=((a+1)^+)^2-(a^+)^2\\&=\begin{bmatrix}(2a+1) & \text{if } a\geq 0\\ (a+1)^2 & \text{if } -1\leq a<0\\ 0 & \text{if } a<-1\end{bmatrix}\\&\leq\begin{bmatrix}(2a+1) & \text{if } a\geq 0\\ 1 & \text{if } -1\leq a<0\\ 0 & \text{if } a<-1\end{bmatrix}.\end{aligned}$$

The reader can follow the same procedure to derive the remainder of the inequality. The key part is to write $\tilde{a}$ and $\tilde{b}$ in terms of a and b, respectively.

We first give analysis for $\min(a,b)\geq 1+c_2$ as this is more involved regime to consider. Break down the first term of (37) as

$$-\frac{\mu n_1\ n_0}{s}2c_2b+\frac{\mu n_1\ n_0}{s}(-2b+c_2^2+2c_2+1).$$

Note that the positive term of (40) is $\frac{\mu n_1 n_2}{s}2c_2b+\frac{\mu n_1 n_2}{s}c_2^2$. Notice that $-\frac{\mu n_1\ n_0}{s}2c_2b+\frac{\mu n_1 n_2}{s}2c_2b\leq 0$. So we discard these two terms from the inequality to obtain another inequality. $\frac{\mu n_1 n_2}{s}c_2^2$ of (40) is put back into the parenthesis of (40). After removing the nonpositive middle term inside the parenthesis of (40), we obtain $c_2^2+c_4(-2d+c_4)\leq c_2^2-2c_4^2s+c_4^2<0$ for all sufficiently large s, which allows for the removal of (40) for large s. Similarly, we discard (39). For (38), its first term in parenthesis was broken down and used to eliminate (39)'s positive a term. (38) now becomes

$$\begin{aligned}&\frac{\mu n_2\ n_0}{s}[(-2a+c_2^2+2c_2+1)+(c_3-c_4)(-2d+c_3-c_4)]\\&\leq\frac{\mu n_2\ n_0}{s}[(c_2^2+2c_2+1)+(c_3-c_4)(-2c_4s+c_3-c_4)]\\&\leq 0\quad\text{for large } s.\end{aligned}$$

Therefore we also discard (38). We write the new inequality:

$$\begin{aligned}&DV(\boldsymbol{s})\\&\leq\lambda[(2a+1)+(2b+1)+c_3(2d+c_3)]\\&+\frac{U_s}{2}[(1+c_2)(-2a+1+c_2)+(c_3-c_4)(-2d+c_3-c_4)]\\&+\frac{U_s}{2}[(1+c_2)(-2b+1+c_2)+(c_3-c_4)(-2d+c_3-c_4)\\&+\frac{\mu n_1\ n_0}{s}[(-2b+c_2^2+2c_2+1)+(c_3-c_4)(-2d+c_3-c_4)].\end{aligned}$$

Notice that we have intentionally not discarded the rest of (37) because it is key to proving $DV(\boldsymbol{s})\leq\epsilon$ as can be seen in a moment. The drift can be reduced to the following for large s using $n_0\geq\frac{s}{3}$, $n_1>\frac{sp}{2}$ and $d>c_4s$:

$$DV(\boldsymbol{s})\leq\frac{-\mu s^2\ p(c_3-c_4)c_4}{3}+k_2s+k_3\leq-\epsilon$$

for all sufficiently large s, and where k_2,k_3 are some constants.

Suppose that $a<1+c_2$ or $b<1+c_2$ or both at the same time. Then (37) still admits a negative quadratic term. The positive terms inside the parenthesis of (39) and (40) are bounded by $c_2(2+3c_2)$ while the third terms inside the parenthesis of (39) and (40) do not exceed $-c_4^2s$. This makes (39) and (40) nonpositive for sufficiently large s and they can be discarded. Also we may omit (38) because it is nonpositive for large s. Thus $DV(\boldsymbol{s})\leq-\epsilon$ for all sufficiently large s.

Case 2.2: Conditions of Case 2 and $n_0=n_1+1$.

We have

$$\begin{aligned}b&=n_0+n_2-c_2n_1=s-n_1-c_2n_1=s-(1+c_2)n_1\\&\leq s-\left(\frac{s}{3}-1\right)(1+c_2)=s\left(1-\frac{1+c_2}{3}\right)+1+c_2.\end{aligned}\tag{41}$$

Thus, $b<0$ for all sufficiently large s. (41) also implies that $\tilde{b}\leq b+2+2c_1+c_2<0$ for all sufficiently large s. So we drop all b terms from the drift inequality and for large s we obtain (42)–(48), shown at the bottom of the next page.

If $a\geq 1+c_2$, the sum of (45) and (47) yields nonpositive result for all sufficiently large s by making use of $n_0=n_1+1$ and $d\geq c_4s$. If $a<1+c_2$, then the term in (47) does not exceed $\frac{\mu n_1 n_2}{s}[c_2(2+3c_2)+c_4(-2d+c_4)]\leq 0$ for all sufficiently large s. So discard (45), (47) and (48) in the large s regime. Note that $c_1-1<(c_3-c_4)c_4$ is implied by (10). Therefore, we write using $n_0=n_1+1\geq\frac{s}{3}$, $a=n_0+n_1-c_2n_2\leq s$ and $d\geq c_4s$:

$$DV(\boldsymbol{s})\leq\frac{2s^2[(c_1-1)-c_4(c_3-c_4)]}{9}+k_1s+k_2\leq-\epsilon$$

for all sufficiently large s.

Case 2.3: Conditions of Case 2 and $n_0=n_1$.

Under these constraints on n_0,n_1,n_2, we have $n_0=n_1\geq\frac{s}{3}$. For b,

$$\begin{aligned}b&=n_0+\min(n_0,n_2)+c_1(n_2-n_0)^+-c_2n_1\\&=n_0+n_2-c_2n_1=s-n_1-c_2n_1\leq s\left(1-\frac{1+c_2}{3}\right).\end{aligned}$$

Since $b\leq-ks$ for some $k>0$,

$$\tilde{b}\leq b+2+2c_1+c_2<0$$

for all sufficiently large s. Therefore all b terms disappear from the drift and we get (49)–(55), shown at the bottom of the next page.

If $a\geq c_2-c_1+2$, $\frac{\mu n_1 n_2}{s}2ac_2$ of (55) is cancelled out by $\frac{\mu n_2 n_0}{s}(-2c_2a)$ of (53). If not, the positive term of (55) can be upperbounded by $\frac{\mu n_2 n_0}{s}c_2(2(c_2-c_1+2)+c_2)$, which is eliminated by the second term of (55) in the large s regime. After $-2c_2a$ is used to eliminate the positive term of (55), we would like to make sure the remainder of (53) is negative for large s. Note that

$$\begin{aligned}a&=n_0+n_1-c_2n_2\leq s\\d&=c_3n_0+c_4n_1+c_4n_2>c_4s.\end{aligned}$$

Since $(c_1-2)<c_4(c_3-c_4)$ is satisfied by (12), the expression inside (53) is negative for large s. So we omit it along with (54) and (55). The expression in the parenthesis of (52) does not exceed $-ks$ for some $k>0$ for all sufficiently large s by (12) and, therefore, (52) admits the only quadratic term in the inequality, which is negative. Thus $DV(\boldsymbol{s})\leq-\epsilon$ for all sufficiently large s.

Case 3: $n_1 \geq (1-p)s$.
We have

$$a = n_0 + \min(n_0, n_1) + c_1(n_1 - n_0)^+ - c_2 n_2 \tag{56}$$
$$= c_1 n_1 - [(c_1 - 2)n_0 + c_2 n_2] \tag{57}$$
$$\geq c_1(1-p)s - pc_2 s = s[c_1(1-p) - pc_2] \tag{58}$$
$$> \frac{(c_1 - 2)s}{2}. \tag{59}$$

(57) is simply because of $n_1 \geq (1-p)s > ps \geq n_0$. In addition to utilizing $n_1 \geq (1-p)s$, (58) follows from the following thanks to conditions $3c_1 < c_2$ and $c_1 > 2$:

$$\max_{\substack{n_0+n_2 \leq ps \\ \min(n_0,n_2)\geq 0}} [(c_1 - 2)n_0 + c_2 n_2] = pc_2\, s.$$

And the last inequality is obtained by $pc_2 < 1 - p < 1$ and $1 - p > \frac{1}{2}$. Since $c_1 > 2$, then $a \geq ks$ for some $k > 0$.

$$DV(\boldsymbol{s}) \leq \lambda \left(\begin{bmatrix} (2a+1) & \text{if } a \geq 0 \\ 1 & \text{if } -1 \leq a < 0 \\ 0 & \text{if } a < -1 \end{bmatrix} + c_3(2d + c_3) \right) \tag{42}$$
$$+ \frac{U_s}{2} \left(\begin{bmatrix} (1+c_2)(-2a+1+c_2) & \text{if } a \geq 1 + c_2 \\ 0 & \text{if } 0 \leq a < 1 + c_2 \\ 0 & \text{if } a < 0 \end{bmatrix} + (c_3 - c_4)(-2d + c_3 - c_4) \right) \tag{43}$$
$$+ \frac{U_s}{2} \left(\begin{bmatrix} (c_1-1)(2a+c_1-1) & \text{if } a \geq 0 \\ (c_1-1)^2 & \text{if } -(c_1-1) \leq a < 0 \\ 0 & \text{if } a < -(c_1-1) \end{bmatrix} + (c_3 - c_4)(-2d + c_3 - c_4) \right) \tag{44}$$
$$+ \frac{\mu n_2 n_0}{s} \left(\begin{bmatrix} (1+c_2)(-2a+1+c_2) & \text{if } a \geq 1 + c_2 \\ 0 & \text{if } 0 \leq a < 1 + c_2 \\ 0 & \text{if } a < 0 \end{bmatrix} + (c_3 - c_4)(-2d + c_3 - c_4) \right) \tag{45}$$
$$+ \frac{\mu n_1 n_0}{s} \left(\begin{bmatrix} (c_1-1)(2a+c_1-1) & \text{if } a \geq 0 \\ (c_1-1)^2 & \text{if } -(c_1-1) \leq a < 0 \\ 0 & \text{if } a < -(c_1-1) \end{bmatrix} + (c_3 - c_4)(-2d + c_3 - c_4) \right) \tag{46}$$
$$+ \frac{\mu n_1 n_2}{s} \left(\begin{bmatrix} c_2(2a+c_2) & \text{if } a \geq 0 \\ c_2^2 & \text{if } -c_2 \leq a < 0 \\ 0 & \text{if } a < -c_2 \end{bmatrix} + c_4(-2d + c_4) \right) \tag{47}$$
$$+ \frac{\mu n_2 n_1}{s} \left(\begin{bmatrix} -(2a-1) & \text{if } a \geq 1 \\ 0 & \text{if } 0 \leq a < 1 \\ 0 & \text{if } a < 0 \end{bmatrix} + c_4(-2d + c_4) \right) \tag{48}$$

$$DV(\boldsymbol{s}) \leq \lambda \left(\begin{bmatrix} (2a+1) & \text{if } a \geq 0 \\ 1 & \text{if } -1 \leq a < 0 \\ 0 & \text{if } a < -1 \end{bmatrix} + c_3(2d + c_3) \right) \tag{49}$$
$$+ \frac{U_s}{2} \left(\begin{bmatrix} (c_2 - c_1 + 2)(-2a + c_2 - c_1 + 2) & \text{if } a \geq c_2 - c_1 + 2 \\ 0 & \text{if } 0 \leq a < c_2 - c_1 + 2 \\ 0 & \text{if } a < 0 \end{bmatrix} + (c_3 - c_4)(-2d + c_3 - c_4) \right) \tag{50}$$
$$+ \frac{U_s}{2} \left(\begin{bmatrix} 2(c_1-1)(2a+2c_1-2) & \text{if } a \geq 0 \\ (2c_1-2)^2 & \text{if } -2(c_1-1) \leq a < 0 \\ 0 & \text{if } a < -2(c_1-1) \end{bmatrix} + (c_3 - c_4)(-2d + c_3 - c_4) \right) \tag{51}$$
$$+ \frac{\mu n_1 n_0}{s} \left(\begin{bmatrix} 2(c_1-1)(2a+2c_1-2) & \text{if } a \geq 0 \\ (2c_1-2)^2 & \text{if } -2(c_1-1) \leq a < 0 \\ 0 & \text{if } a < -2(c_1-1) \end{bmatrix} + (c_3 - c_4)(-2d + c_3 - c_4) \right) \tag{52}$$
$$+ \frac{\mu n_2 n_0}{s} \left(\begin{bmatrix} (c_2 - c_1 + 2)(-2a + c_2 - c_1 + 2) & \text{if } a \geq c_2 - c_1 + 2 \\ 0 & \text{if } 0 \leq a < c_2 - c_1 + 2 \\ 0 & \text{if } a < 0 \end{bmatrix} + (c_3 - c_4)(-2d + c_3 - c_4) \right) \tag{53}$$
$$+ \frac{\mu n_2 n_1}{s} \left(\begin{bmatrix} (-2a+1) & \text{if } a \geq 1 \\ 0 & \text{if } 0 \leq a < 1 \\ 0 & \text{if } a < 0 \end{bmatrix} + c_4(-2d + c_4) \right) \tag{54}$$
$$+ \frac{\mu n_1 n_2}{s} \left(\begin{bmatrix} c_2(2a+c_2) & \text{if } a \geq 0 \\ c_2^2 & \text{if } -c_2 \leq a < 0 \\ 0 & \text{if } a < 0 \end{bmatrix} + c_4(-2d + c_4) \right) \tag{55}$$

Recalling $\tilde{a}$ is the new a after one of the legitimate transitions occur, we get $\tilde{a} \geq a - 2 - 2c_1 - c_2 > 0$ for all sufficiently large s. Therefore we conclude that both a and $\tilde{a}$ are greater than any positive constant for sufficiently large s. For b,

$$\begin{aligned} b &= n_0 + \min(n_0, n_2) + c_1(n_2 - n_0)^+ - c_2 n_1 \\ &\leq c_1 ps - c_2 n_1 \quad \text{by} \quad c_1 > 2 \\ &\leq c_1 ps - c_2(1-p)s \\ &= s[c_1 p - c_2(1-p)] < 0 \quad \text{for all} \quad s > 0. \end{aligned}$$

Because $b \leq -ks$ for some $k > 0$, we get for $\tilde{b}$,

$$\tilde{b} \leq b + 2 + 2c_1 + c_2 < 0 \quad \text{for all sufficiently large } s.$$

That means we can drop all b terms from drift inequalities that we investigate in Case 3.

Case 3.1: Conditions of Case 3 and $n_0 > 0$.

For all sufficiently large s we can write (60)–(64), shown at the bottom of this page.

Observe that (63) and (64) are nonpositive terms for large s and we therefore omit them. Also $(2c_1-2)(2a+2c_1-2)+(c_2-c_1+2)(-2a+c_2-c_1+2) < 0$ for all sufficiently large s by (9) and the fact that $a \geq ks$. Thus we may omit (61) and (62). To get $DV(\boldsymbol{s}) \leq -\epsilon$ for all sufficiently large s, we need to satisfy $(c_1 - 2)2a > 2c_3 d$ so that $DV(\boldsymbol{s}) \leq \lambda(-ks)$ for some $k > 0$. Combined with $d \leq c_3 s$ and $a \geq \frac{c_1-2}{2}s$, (13) implies $(c_1 - 2)2a > 2c_3 d$.

Case 3.2: Conditions of Case 3 and $n_0 = 0$.

As there exists no *type* 0 peers in the system, our policy dictates the fixed seed selects a peer uniformly at random and uploads the piece that the peer needs. We have for large s

$$DV(\boldsymbol{s}) = \lambda\left[(c_1-2)(-2a+c_1-2) + c_3(2d+c_3)\right] \quad (65)$$

$$+ \frac{U_s n_1}{s}\left[c_1(-2a+c_1) + c_4(-2d+c_4)\right] \quad (66)$$

$$+ \frac{U_s n_2}{s}\left[c_2(2a+c_2) + c_4(-2d+c_4)\right] \quad (67)$$

$$+ \frac{\mu n_2 n_1}{s}\left[c_1(-2a+c_1) + c_4(-2d+c_4)\right]. \quad (68)$$

(68) may be omitted as it is nonpositive for large s. Also the sum of first terms of (66) and (67) is nonpositive because $c_1 n_1 - c_2 n_2 \geq c_1(1-p)s - c_2 ps \geq \frac{c_1-2}{2}s$ for all sufficiently large s. We are left with (65). We prove $DV(\boldsymbol{s}) \leq -\epsilon$ for all sufficiently large s in the same way as in Case 3.1

Case 4: $n_1 \geq n_0, n_1 \geq n_2$ and $n_i < (1-p)s \quad \forall i$.

Under this case, we observe that $n_1 \geq \frac{s}{3}$ and we have for b,

$$\begin{aligned} b &= n_0 + \min(n_0, n_2) + c_1(n_2 - n_0)^+ - c_2 n_1 \\ &\leq n_0 + \min(n_0, n_2) + c_1 n_2 - c_2 n_1 \\ &\leq n_0 + n_1 + c_1 n_2 - c_2 n_1 \\ &\leq n_1(c_1 + 2 - c_2) \leq \frac{s}{3}(c_1 + 2 - c_2) < 0 \quad \text{for all } s > 0. \end{aligned}$$

Therefore, $\tilde{b} \leq b + 2 + 2c_1 + c_2 < 0$ for all sufficiently large s. All b terms disappear in all subcases of Case 4.

Case 4.1: Conditions of Case 4 and $n_1 > n_0$ and $n_0 > 0$ and $n_1 > n_2$.

For this case, we get (69)–(73), shown at the bottom of this page.

$$DV(\boldsymbol{s}) = \lambda\left[(c_1-2)(-2a+c_1-2) + c_3(2d+c_3)\right] \quad (60)$$

$$+ \frac{U_s}{2}\left[(2c_1-2)(2a+2c_1-2) + (c_3-c_4)(-2d+c_3-c_4)\right] \quad (61)$$

$$+ \frac{U_s}{2}\left[(c_2-c_1+2)(-2a+c_2-c_1+2) + (c_3-c_4)(-2d+c_3-c_4)\right] \quad (62)$$

$$+ \frac{\mu n_2 n_0}{s}\left[(c_2-c_1+2)(-2a+c_2-c_1+2) + (c_3-c_4)(-2d+c_3-c_4)\right] \quad (63)$$

$$+ \frac{\mu n_2 n_1}{s}\left[c_1(-2a+c_1) + c_4(-2d+c_4)\right] \quad (64)$$

$$DV(\boldsymbol{s}) \leq \lambda\left(\begin{bmatrix}(c_1-2)(-2a+c_1-2) & \text{if } a \geq c_1-2 \\ 0 & \text{if } 0 \leq a < c_1-2 \\ 0 & \text{if } a < 0\end{bmatrix} + c_3(2d+c_3)\right) \quad (69)$$

$$+ \frac{U_s}{2}\left(\begin{bmatrix}(c_2-c_1+2)(-2a+c_2-c_1+2) & \text{if } a \geq c_2-c_1+2 \\ 0 & \text{if } 0 \leq a < c_2-c_1+2 \\ 0 & \text{if } a < 0\end{bmatrix} + (c_3-c_4)(-2d+c_3-c_4)\right) \quad (70)$$

$$+ \frac{U_s}{2}\left(\begin{bmatrix}(2c_1-2)(2a+2c_1-2) & \text{if } a \geq 0 \\ (2c_1-2)^2 & \text{if } -(2c_1-2) \leq a < 0 \\ 0 & \text{if } a < -2(c_1-1)\end{bmatrix} + (c_3-c_4)(-2d+c_3-c_4)\right) \quad (71)$$

$$+ \frac{\mu n_2 n_0}{s}\left(\begin{bmatrix}(c_2-c_1+2)(-2a+c_2-c_1+2) & \text{if } a \geq c_2-c_1+2 \\ 0 & \text{if } 0 \leq a < c_2-c_1+2 \\ 0 & \text{if } a < 0\end{bmatrix} + (c_3-c_4)(-2d+c_3-c_4)\right) \quad (72)$$

$$+ \frac{\mu n_2 n_1}{s}\left(\begin{bmatrix}c_1(-2a+c_1) & \text{if } a \geq c_1 \\ 0 & \text{if } 0 \leq a < c_1 \\ 0 & \text{if } a < 0\end{bmatrix} + c_4(-2d+c_4)\right) \quad (73)$$

We now investigate $DV(\boldsymbol{s})$ depending on different values of n_2. First, suppose $n_2 < \frac{s}{2c_2}$. Then,

$$a = n_0(2-c_1) + c_1 n_1 - c_2 n_2 \tag{74}$$
$$\geq n_0(2-c_1) + c_1 n_1 - \frac{s c_2}{2c_2} \tag{75}$$
$$\geq 2n_1 - \frac{s}{2} \tag{76}$$
$$\geq \frac{2s}{3} - \frac{s}{2} = \frac{s}{6}, \tag{77}$$

where (76) follows from $n_1 \geq n_0$, $c_1 > 2$ and (77) is obtained by $n_1 \geq \frac{s}{3}$.

We omit everything except the second term in (69) and first terms of (70) and (71) as the omitted terms are all negative for all sufficiently large s. To get $DV(\boldsymbol{s}) \leq -\epsilon$, the following condition suffices:

$$\lambda c_3^2 < U_s \left(\frac{c_2 - 3c_1 + 4}{12} \right).$$

Note that this is just (14).

If $n_2 \geq \frac{s}{2c_2}$, (73) admits a negative quadratic term. The only remaining term that may be quadratic in s is (72). However, the expression inside the parenthesis of (72) is negative for all sufficiently large s so it is omitted. So $DV(\boldsymbol{s}) \leq -\epsilon$ because of the negative quadratic coming from (73).

Case 4.2: Conditions of Case 4 and $n_1 > n_0$ and $n_0 = 0$ and $n_1 > n_2$.

Notice that $n_1 > \frac{s}{2}$ and $n_2 > sp$ and the fixed seed uploads to *type* n_1 and *type* n_2 peers selected uniformly at random. We derive (78)–(81), shown at the bottom of this page.

The first term of (81) is nonpositive so we may omit it. The second term of (81) admits negative quadratic expression because $\frac{\mu n_1 n_2}{s} < \frac{\mu s p}{2}$ and $-c_4 2d = -2c_4^2\, s$. Thus $DV(\boldsymbol{s}) \leq -\epsilon$ for all sufficiently large s.

Case 4.3: Conditions of Case 4 and $n_1 > n_0$ and $n_0 > 0$ and $n_1 = n_2$.

Note that $a = (2-c_1)n_0 + c_1 n_1 - c_2 n_2 < c_1 n_1 - c_2 n_2 = (c_1 - c_2) n_1 \leq (c_1 - c_2)\frac{s}{3} < 0$ and, therefore, $\tilde{a} \leq a + 2 + 2c_1 + c_2 < 0$ for all sufficiently large s. All a's are dropped from the drift. Recall that we dropped all b terms from the drift by the analysis given in the beginning of Case 4. Thus, we get for all sufficiently large s

$$DV(\boldsymbol{s}) = \lambda\left(c_3(2d + c_3)\right) \tag{82}$$
$$+ \frac{U_s}{2}\left((c_3 - c_4)(-2d + c_3 - c_4)\right) \tag{83}$$
$$+ \frac{U_s}{2}\left((c_3 - c_4)(-2d + c_3 - c_4)\right) \tag{84}$$
$$+ \frac{\mu n_2 n_0}{s}\left((c_3 - c_4)(-2d + c_3 - c_4)\right) \tag{85}$$
$$+ \frac{\mu n_1 n_0}{s}\left((c_3 - c_4)(-2d + c_3 - c_4)\right) \tag{86}$$
$$+ \frac{\mu n_2 n_1}{s}\left(c_4(-2d + c_4)\right) \tag{87}$$
$$+ \frac{\mu n_1 n_2}{s}\left(c_4(-2d + c_4)\right). \tag{88}$$

Observe that (85) and (86) are nonpositive for all sufficiently large s and (87), (88) admit negative quadratic terms. Therefore we have $DV(\boldsymbol{s}) \leq -\epsilon$.

Case 4.4: Conditions of Case 4 and $n_1 > n_0$ and $n_0 = 0$ and $n_1 = n_2$.

Since $a = c_1 n_1 - c_2 n_2 = (c_1 - c_2)\frac{s}{2}$, then $\tilde{a} \leq a + 2 + 2c_1 + c_2 < 0$ for all sufficiently large s. Therefore we have

$$DV(\boldsymbol{s}) = \lambda\left(c_3(2d + c_3)\right) \tag{89}$$
$$+ \frac{U_s n_1}{s}\left(c_4(-2d + c_4)\right) \tag{90}$$
$$+ \frac{U_s n_2}{s}\left(c_4(-2d + c_4)\right) \tag{91}$$
$$+ \frac{\mu n_2 n_1}{s}\left(c_4(-2d + c_4)\right) \tag{92}$$
$$+ \frac{\mu n_1 n_2}{s}\left(c_4(-2d + c_4)\right). \tag{93}$$

First note that (89)-(91) do not have any quadratic term. Also (92) and (93) admit negative quadratic terms, which implies $DV(\boldsymbol{s}) \leq -\epsilon$ for large s.

Case 4.5: Conditions of Case 4 and $n_1 = n_0 \geq n_2$.

Under this case, $\frac{s}{3} \leq n_1 \leq \frac{s}{2}$ and $n_2 \leq \frac{s}{3}$. Then

$$a = (2-c_1)n_0 + c_1 n_1 - c_2 n_2 = 2n_1 - c_2 n_2 \leq s$$

and we write (94)–(100), shown at the top of the next page.

First of all, using $a \leq s$ and (12) we get

$$(2c_1 - 2)2a + (c_3 - c_4)(-2d) \leq (2c_1 - 2)2s - (c_3 - c_4)2c_4 s \leq -ks$$

$$DV(\boldsymbol{s}) \leq \lambda\left(\begin{bmatrix} (c_1-2)(-2a + c_1 - 2) & \text{if } a \geq c_1 - 2 \\ 0 & \text{if } 0 \leq a < c_1 - 2 \\ 0 & \text{if } a < 0 \end{bmatrix} + c_3(2d + c_3)\right) \tag{78}$$
$$+ \frac{U_s n_1}{s}\left(\begin{bmatrix} c_1(-2a + c_1) & \text{if } a \geq c_1 \\ 0 & \text{if } 0 \leq a < c_1 \\ 0 & \text{if } a < 0 \end{bmatrix} + c_4(-2d + c_4)\right) \tag{79}$$
$$+ \frac{U_s n_2}{s}\left(\begin{bmatrix} c_2(2a + c_2) & \text{if } a \geq 0 \\ c_2^2 & \text{if } -c_2 \leq a < 0 \\ 0 & \text{if } a < -c_2 \end{bmatrix} + c_4(-2d + c_4)\right) \tag{80}$$
$$+ \frac{\mu n_1 n_2}{s}\left(\begin{bmatrix} c_1(-2a + c_1) & \text{if } a \geq c_1 \\ 0 & \text{if } 0 \leq a < c_1 \\ 0 & \text{if } a < 0 \end{bmatrix} + c_4(-2d + c_4)\right) \tag{81}$$

$$DV(\boldsymbol{s}) \le \lambda \left(\begin{bmatrix} (2a+1) & \text{if } a \ge 0 \\ 1 & \text{if } -1 \le a < 0 \\ 0 & \text{if } a < -1 \end{bmatrix} + c_3(2d + c_3) \right) \tag{94}$$

$$+ \frac{U_s}{2} \left(\begin{bmatrix} (2c_1-2)(2a+2c_1-2) & \text{if } a \ge 0 \\ (2c_1-2)^2 & \text{if } -(2c_1-2) \le a < 0 \\ 0 & \text{if } a < -(2c_1-2) \end{bmatrix} + (c_3-c_4)(-2d+c_3-c_4) \right) \tag{95}$$

$$+ \frac{U_s}{2} \left(\begin{bmatrix} (c_2-c_1+2)(-2a+c_2-c_1+2) & \text{if } a \ge c_2-c_1+2 \\ 0 & \text{if } 0 \le a < c_2-c_1+2 \\ 0 & \text{if } a < 0 \end{bmatrix} + (c_3-c_4)(-2d+c_3-c_4) \right) \tag{96}$$

$$+ \frac{\mu n_1 n_0}{s} \left(\begin{bmatrix} (2c_1-2)(2a+2c_1-2) & \text{if } a \ge 0 \\ (2c_1-2)^2 & \text{if } -(2c_1-2) \le a < 0 \\ 0 & \text{if } a < -(2c_1-2) \end{bmatrix} + (c_3-c_4)(-2d+c_3-c_4) \right) \tag{97}$$

$$+ \frac{\mu n_2 n_0}{s} \left(\begin{bmatrix} (c_2-c_1+2)(-2a+c_2-c_1+2) & \text{if } a \ge c_2-c_1+2 \\ 0 & \text{if } 0 \le a < c_2-c_1+2 \\ 0 & \text{if } a < 0 \end{bmatrix} + (c_3-c_4)(-2d+c_3-c_4) \right) \tag{98}$$

$$+ \frac{\mu n_1 n_2}{s} \left(\begin{bmatrix} c_2(2a+c_2) & \text{if } a \ge 0 \\ c_2^2 & \text{if } -c_2 \le a < 0 \\ 0 & \text{if } a < -c_2 \end{bmatrix} + c_4(-2d+c_4) \right) \tag{99}$$

$$+ \frac{\mu n_2 n_1}{s} \left(\begin{bmatrix} (-2a+1) & \text{if } a \ge 1 \\ 0 & \text{if } 0 \le a < 1 \\ 0 & \text{if } a < 0 \end{bmatrix} + c_4(-2d+c_4) \right) \tag{100}$$

for some $k > 0$. This implies that (97) admits a negative quadratic term as $n_1, n_0 \ge \frac{s}{3}$. Also omit (100) as it is nonpositive for all sufficiently large s. Combining (98) and the first term of (99) without constants, we have

$$\begin{aligned} &\frac{\mu n_2\, n_0}{s}[(c_2-c_1+2)(-2a)+(c_3-c_4)(-2d)] + \frac{\mu n_1 n_2}{s} 2ac_2 \\ &= \frac{\mu n_1 n_2}{s}[(c_1-2)(2a)+(c_3-c_4)(-2d)] \\ &\le \frac{\mu n_1 n_2}{s}[(c_1-2)(2s)+(c_3-c_4)(-2c_4 s)] \\ &\le \frac{\mu n_1 n_2}{s}(-k_1 s) \quad \text{for some } k_1 > 0, \end{aligned}$$

where we used $(c_3 - c_4)c_4 > 2(c_1 - 1)$ and $c_1 > 2$ to get $(c_1 - 2) < c_4(c_3 - c_4)$. Thus (98) and (99) are omitted as well. Since we have only one quadratic term that is negative, $DV(\boldsymbol{s}) \le -\epsilon$ for all sufficiently large s.

Case 5: $n_0 \ge n_2 \ge n_1$ and $n_i < (1-p)s$ for every i.

Note that Case 5 and Case 2 are symmetric.

Case 6: $n_2 \ge (1-p)s$.

Note that Case 6 and Case 3 are symmetric.

Case 7: $n_2 \ge n_0$, $n_2 \ge n_1$ and $n_i < (1-p)s \quad \forall i$.

Note that Case 7 is symmetric to Case 4.

By Case 1-7, we proved the existence of some $s_0(\epsilon) > 0$ such that $DV(\boldsymbol{s}) \le -\epsilon$ for all $s \ge s_0(\epsilon)$. Now we let $b = \max\{DV(\boldsymbol{s}) : s < s_0)\} + 1$ and $\boldsymbol{C} = \{\boldsymbol{s} : s < s_0\}$, thus (3) is achieved and $\boldsymbol{C}$ is a finite set as desired. Finally we appeal to $V(\boldsymbol{s}) \ge (c_4\, s)^2$ to show (2).

ACKNOWLEDGMENT

The authors wish to thank Ebad Ahmed for his contribution to the early stages of this project.

REFERENCES

[1] "Cisco visual networking index: Forecast and methodology, 2015–2020," Cisco Syst., San Jose, CA, USA, Tech. Rep., Jun. 2016. [Online]. Available: http://www.webvideomarketing.org/pdf/Cisco_Video_and_Visual_Networking_Index_Report_8.10.16.pdf

[2] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st Ed. MCC Workshop Mobile Cloud Comput. (MCC)*. New York, NY, USA: ACM, 2012, pp. 13–16.

[3] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 27–32, Oct. 2014.

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

[5] F. Mathieu and J. Reynier, "Missing piece issue and upload strategies in flashcrowds and p2p-assisted filesharing," in *Proc. Adv. Int. Conf. Telecommun. Int. Conf. Internet Web Appl. Services (AICT-ICIW)*, Feb. 2006, p. 112.

[6] 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.

[7] R. E. Bohn and J. E. Short, "How much information? 2009 report on American consumers," Global Inf. Ind. Center, Univ. California San Diego, La Jolla, CA, USA, Tech. Rep., 2009. [Online]. Available: http://hmi.ucsd.edu/pdf/HMI_2009_ConsumerReport_Dec9_2009.pdf

[8] H. Reittu, "A stable random-contact algorithm for peer-to-peer file sharing," in *Proc. 4th IFIP TC Int. Workshop Self-Organizing Syst. (IWSOS)*. Berlin, Germany: Springer-Verlag, 2009, pp. 185–192.

[9] 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.

[10] B. Oğuz, 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.

[11] B. Zhang, S. C. Borst, and M. I. Reiman, "Optimal server scheduling in hybrid p2p networks," *Perform. Eval.*, vol. 67, no. 11, pp. 1259–1272, Nov. 2010.

[12] E. de Souza e Silva, R. M. M. Leão, D. S. Menasché, and D. Towsley, "On the scalability of P2P swarming systems," *Comput. Netw.*, vol. 151, pp. 93–113, Mar. 2019.

[13] 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.
[14] L. Massoulié and M. Vojnovic, "Coupon replication systems," *IEEE/ACM Trans. Netw.*, vol. 16, no. 3, pp. 603–616, Jun. 2008.
[15] D. Qiu and R. Srikant, "Modeling and performance analysis of BitTorrent-like peer-to-peer networks," in *Proc. Conf. Appl., Technol., Architectures, Protocols Comput. Commun. (SIGCOMM)*, New York, NY, USA: ACM, 2004, pp. 367–378.
[16] D. Qiu and W. Sang, "Global stability of peer-to-peer file sharing systems," *Comput. Commun.*, vol. 31, no. 2, pp. 212–219, Feb. 2008.
[17] Y. Tian, D. Wu, and K. W. Ng, "Modeling, analysis and improvement for BitTorrent-like file sharing networks," in *Proc. 25th IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2006, pp. 1–11.
[18] B. Fan, D.-M. Chiu, and J. C. S. Lui, "Stochastic analysis and file availability enhancement for bt-like file sharing systems," in *Proc. 14th IEEE Int. Workshop Qual. Service*, Jun. 2006, pp. 30–39.
[19] J. Zhu, S. Ioannidis, N. Hegde, and L. Massoulié, "Stable and scalable universal swarms," in *Proc. ACM Symp. Princ. Distrib. Comput. (PODC)*. New York, NY, USA: ACM, 2013, pp. 260–269.
[20] X. Zhou, S. Ioannidis, and L. Massoulié, "On the stability and optimality of universal swarms," in *Proc. ACM SIGMETRICS Joint Int. Conf. Meas. Modeling Comput. Syst. (SIGMETRICS)*. New York, NY, USA: ACM, 2011, pp. 341–352.
[21] D. S. Menasche, A. A. De A. Rocha, A. Antonio, B. Li, D. Towsley, and A. Venkataramani, "Content availability and bundling in swarming systems," *IEEE/ACM Trans. Netw.*, vol. 21, no. 2, pp. 580–593, Apr. 2013.
[22] H. Zhang, S. Vasudevan, R. Li, and D. Towsley, "Coalitions improve performance in data swarming systems," *IEEE/ACM Trans. Netw.*, vol. 23, no. 6, pp. 1790–1804, Dec. 2015.
[23] E. E. de Souza Silva, R. M. M. Leão, D. S. Menasché, and A. A. de A. Rocha, "On the interplay between content popularity and performance in p2p systems," in *Proc. 10th Int. Conf. Quant. Eval. Syst. (QEST)*. Berlin, Germany: Springer-Verlag, 2013, pp. 3–21.
[24] B. Hajek, *Random Processes for Engineers*. Cambridge, U.K.: Cambridge Univ. Press, 2015, ch. 6, p. 194.
[25] B. Cohen, "Incentives build robustness in bittorrent," in *Proc. Workshop Econ. Peer-to-Peer Syst.*, vol. 6, 2003, pp. 68–72.
[26] A. Legout, N. Liogkas, E. Kohler, and L. Zhang, "Clustering and sharing incentives in BitTorrent systems," in *Proc. ACM SIGMETRICS Int. Conf. Meas. Modeling Comput. Syst. (SIGMETRICS)*. New York, NY, USA: ACM, 2007, pp. 301–312.
[27] A. Legout, G. Urvoy-Keller, and P. Michiardi, "Rarest first and choke algorithms are enough," in *Proc. 6th ACM SIGCOMM Conf. Internet Meas. (IMC)*. New York, NY, USA: ACM, 2006, pp. 203–216.
[28] D. X. Mendes, E. E. de Souza Silva, D. S. 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.
[29] O. Bilgen and A. B. Wagner, "A new stable peer-to-peer protocol with non-persistent peers," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2017, pp. 1782–1790.

Omer Bilgen received the B.S. degree in Electrical and Electronics Engineering from Bilkent University, Ankara, Turkey. He is currently pursuing the Ph.D. degree in Electrical and Computer Engineering at Cornell University, Ithaca, NY, USA where he received the M.S. degree in the same field. His research interests include source coding, network protocol design and machine learning.

Aaron B. Wagner (S'00–M'05–SM'13) received the B.S. degree in Electrical Engineering from the University of Michigan, Ann Arbor, in 1999 and the M.S. and Ph.D. degrees in Electrical Engineering and Computer Sciences from the University of California, Berkeley, in 2002 and 2005, respectively. During the 2005-2006 academic year, he was a Postdoctoral Research Associate in the Coordinated Science Laboratory at the University of Illinois at Urbana-Champaign and a Visiting Assistant Professor in the School of Electrical and Computer Engineering at Cornell University. Since 2006, he has been with the School of Electrical and Computer Engineering at Cornell, where he is currently a Professor.

He has received the NSF CAREER award, the David J. Sakrison Memorial Prize from the U.C. Berkeley EECS Dept., the Bernard Friedman Memorial Prize in Applied Mathematics from the U.C. Berkeley Dept. of Mathematics, the James L. Massey Research and Teaching Award for Young Scholars from the IEEE Information Theory Society, and teaching awards at the department, college, and university level at Cornell. He is distinguished lecturer for the IEEE Information Theory Society for 2018-2019.