

# Gossiping with Multiple Messages

Sujay Sanghavi  
LIDS  
MIT

Email: sanghavi@mit.edu

Bruce Hajek  
CSL and ECE  
UIUC

Email: b-hajek@uiuc.edu

Laurent Massoulié  
Thomson Corp. Research  
France

Email: laurent.massoulié@thomson.net

**Abstract**—This paper investigates the dissemination of multiple pieces of information in large networks where users contact each other in a random uncoordinated manner, and users upload one piece per unit time. The underlying motivation is the design and analysis of piece selection protocols for peer-to-peer networks which disseminate files by dividing them into pieces. We first investigate one-sided protocols, where piece selection is based on the states of either the transmitter or the receiver. We show that any such protocol relying only on pushes, or alternatively only on pulls, will be inefficient in disseminating all pieces to all users. We propose a hybrid one-sided piece selection protocol – INTERLEAVE – and show that by using both pushes and pulls it disseminates  $k$  pieces from a single source to  $n$  users in  $10(k + \log n)$  time, while obeying the constraint that each user can upload at most one piece in one unit of time. An optimal, unrealistic centralized protocol would take  $k + \log_2 n$  time in this setting. Moreover, efficient dissemination is also possible if the source implements forward erasure coding, and users push the latest-released coded pieces (but do not pull). We also investigate two-sided protocols where piece selection is based on the states of both the transmitter and the receiver. We show that it is possible to disseminate  $n$  pieces to  $n$  users in  $n + O(\log n)$  time, starting from an initial state where each user has a unique piece.

## I. INTRODUCTION

Peer-to-peer systems are decentralized networks enabling users to contribute resources for mutual benefit. One of the main applications of such networks is the cost-effective distribution of bandwidth-intensive content from one source, or a few sources, to many users simultaneously. Peer-to-peer networks such as eDonkey and BitTorrent, which routinely serve files hundreds of megabytes in length to thousands of users, now account for a sizable share of all Internet traffic [1]. Examples of content distribution systems leveraging end-user resources are [2]–[5]. The service capacity in such systems can grow with the number of users, making them scalable and efficient in servicing a large number of users [6], [7].

File dissemination networks can be broadly categorized into *structured* and *unstructured* networks. Structured networks such as [3]–[5] rely on a specific structured pattern of interconnections among users to deliver the advantages of scalability and robustness. The structured pattern is first set up in a decentralized fashion, and data is then disseminated using this infrastructure. Unstructured networks have minimal infrastructure, and instead rely on randomization to provide load balancing, robustness, and scalability. For example, in the BitTorrent [2] system the only available infrastructure is a tracker of the addresses of users interested in obtaining the

file. Each user acquires a random list of other users from the tracker, who become neighbours. Each user’s actions are based on local information obtained from its neighbors.

This paper investigates data dissemination in unstructured networks. Initial unstructured approaches [8], [9] advocated uploading the whole file at one go. Users receiving the complete file would then upload it to other users chosen at random. These protocols were motivated in part by earlier theoretical work on random gossip models [10], [11] and epidemics [12]. However, for large files, making users wait to receive the entire file before they can start serving it becomes untenable for two reasons: (a) file transfer may take a long time, and during this time the upload capacity of downloading users is wasted, and (b) users who have received the file may depart before uploading a complete copy, resulting in the complete file being lost to others. Modern peer-to-peer file dissemination protocols such as BitTorrent take the following alternate approach to speed up dissemination: the file is divided into pieces, and users can start serving individual pieces once they are received, instead of waiting to obtain the entire file<sup>1</sup>.

When the file to be disseminated is divided into multiple pieces, each user has to carry out the task of *piece selection*: deciding which particular piece of the file is to be communicated at any given time, based on local information<sup>2</sup>. These local decisions have a significant impact on the global effectiveness of file dissemination, as the spread of one piece interacts with the spread of other pieces. The motivations of this paper are (1) to gain a quantitative analytical understanding of how splitting a file speeds up its dissemination in networks with random user contacts, and (2) to understand how the users’ local piece selection decisions impact the dissemination of multiple pieces.

In Section II, we develop a simple model of a peer-to-peer system relaying multiple pieces. We also explain how it captures the speedup obtained from file splitting, and enables us to compare the efficiency of various piece selection protocols. The user contact model is the same as in the classical random gossip process literature [8]–[11]. In Section III we state our main results, and discuss their implications. Sections IV and V contain the Lemmas and proof sketches of the main theorems. Section VI contains some simulation results

<sup>1</sup>The use of file splitting for multihop transfers is similar to the well-known technique of cut-through routing, see [13]

<sup>2</sup>In BitTorrent for example, this decision is made by each user based only on the information of its neighbours. Each peer polls its neighbors for their piece collections, and then downloads the *locally rarest* piece, i.e. the piece that has the lowest representation in the peer’s neighbors.

<sup>0</sup>This research was supported in part by the National Science Foundation under grant CNS 05-19691.

on protocol performance when some of the assumptions made in the model are relaxed.

## II. FRAMEWORK

We now present our peer-to-peer system model. A real-world deployed peer-to-peer network such as BitTorrent is an immensely complex system to model and analyze exactly, and we simplify some aspects for the purposes of tractability. In the following we first describe our framework, and then discuss the assumptions and approximations made.

Consider a network with  $n$  users, each of whom wants to receive an entire copy of a file. The file is divided into pieces. All users have the same upload bandwidth, and time is measured in slots. The length of each slot is the time one user takes to upload one piece. Any user receiving a piece in some slot  $t$  can upload that piece to other users beginning in slot  $t + 1$ . Thus the spread of one piece interacts with the spread of other pieces via the bandwidth constraint.

Throughout this paper, it is assumed that the users contact each other in the following manner: in each time slot each user chooses a *target*, which is another user chosen uniformly at random from the entire network, independently of any state, history, or other users' choices. Communication in that time slot occurs only between each user and its target. This is the contact and communication model used in the classical single-piece random gossip literature [8]–[11]. We provide bounds and performance guarantees that hold with high probability for large  $n$ . Our work goes beyond the classical random gossip literature in that it investigates the simultaneous spread of multiple pieces/messages.

Once a target is chosen, the user undertakes one of the following two possible actions

- *pull*: the user selects a piece it does not currently possess and requests it from the target.
- *push*: the user selects a piece it possesses, and transmits it to the target.

For either of the two actions above, the user needs to make a piece selection. This piece selection is said to be *one-sided* if it is based only on the user's own current state, and not that of the target. The piece selection is said to be *two-sided* if it is based on the current states of both the user and the target. In either case the selection is independent of system history or the states of other users. Different ways of making this choice correspond to different protocols. In this paper we will evaluate the performance of the protocols as measured by the *completion time*, which is the first time slot when all users have all pieces.

Users have limited upload bandwidth. In this paper this will be represented by either a *hard constraint* or a *soft constraint*. In the former, each user can upload at most one piece in any instant of time, while in the latter a user is allowed to upload (potentially) any number of pieces simultaneously. The fact that targets are chosen uniformly at random means that a network with  $n$  users will most likely have a maximum loading of at most  $\log n$  for the case of soft constraints. Soft constraints have previously been analyzed in the random gossip literature, see e.g. [9], [14].

The following simple calculation, similar to the one in [6], demonstrates the potential speedup that can be had from file splitting. Consider an initial condition where the file is divided into  $k$  pieces, and all are initially present at one user, called the source. It is easy to see that the completion time would be at least  $k + \log_2 n$  time slots<sup>3</sup>. It has been shown in [15] that a fully centralized scheme can achieve this bound for all  $n$  and  $k$ <sup>4</sup>. If, however, the file is not divided into pieces, each user can upload data only after receiving a copy of the entire file, which takes  $k$  time slots. In this case, complete file dissemination will take at least  $k \log_2 n$  time slots.<sup>5</sup> If we denote the ratio of the dissemination time for an unsplit file to the dissemination time of a file split into  $k$  pieces as the speedup achieved by splitting the file into  $k$  pieces, then

$$\text{optimal splitting speedup} = \frac{k \log_2 n}{k + \log_2 n}$$

From the above we see that if a decentralized protocol with random user contacts has a completion time close to  $k + \log n$  then its performance is close to that of a centralized optimal protocol, while if its completion time is closer to  $k \log n$  then it is performing badly, providing little speedup from file splitting.

For large networks, splitting a file into a large number of pieces will give significant speedup gains, but at the expense of increased overheads. For example, two-sided protocols may require users to maintain the current states of their neighbours. This may be hard when there are a large number of small pieces. This is the motivation for investigating piece-selection protocols relying on less than complete information, of which one-sided protocols represent an extreme case. Other overheads arise from network and system considerations (for example, whether the protocol uses TCP or UDP, etc.)

We now briefly discuss the modeling approximations made. Network effects including delays, packet losses due to congestion, and user heterogeneity have been abstracted away: we assume that communicating a piece always takes the same amount of time, between any two users. Also, real-world systems are typically open, with users joining and leaving, while our analysis assumes the simultaneous arrival of a large number of users who are present until system completion. The model would be a reasonable approximation for the servicing of a flash crowd<sup>6</sup>, which is when the scalability and efficiency of any file dissemination system is tested most severely. However, different models may be required for other situations. Also, each user may only have a limited view of the network, and may not be able to contact users chosen uniformly at random from the entire network. We relax this last assumption using simulations in Section VI.

<sup>3</sup>It takes at least  $k$  slots for the last piece to emerge from the source, and a further  $\log_2 n$  slots for that piece to reach all users. This  $k + \log_2 n$  lower bound holds even for systems that employ coding.

<sup>4</sup>[15] assumes exactly the same upload/download constraints as this paper. Other related work, with slightly different communication constraints, also point to optimal dissemination times that are close to  $k + \log_2 n$ : see for example [16], [17].

<sup>5</sup>Since the number of users having the file can at most double every  $k$  time slots.

<sup>6</sup>The phenomenon in which a large number of users arrive almost simultaneously, looking for a file initially present only at a few sources, due to the file suddenly being popular.

Finally, an important component of any peer-to-peer system is the incentive mechanism used to ensure users do not leech off the system. In this work we do not investigate incentives, but comment that it may be possible to design token-based incentive schemes that are compatible with our piece selection protocols.

### III. OUR RESULTS: PROTOCOLS AND THEIR PERFORMANCE

The main contributions of this work are outlined in this section. The piece selection protocols are described, along with corresponding performance bounds, in the following order: one-sided protocols using only pull, one-sided protocols using only push, the new hybrid one-sided protocol INTERLEAVE, and a two-sided protocol.

In our investigation of one-sided protocols, we assume that the file is divided into  $k = k(n)$  pieces, where  $k(n)$  is at most a polynomial function of  $n$ , and present results that hold with high probability for large  $n$ . Thus the relative number of pieces and users is allowed to vary over a broad range.

One-sided pull-based protocols are those where all communication occurs only via pulls, and piece selection is one-sided. Two examples within this class of protocols are:

**RANDOM PULL:** In each slot each user requests a random piece from the set of pieces it does not possess.

**SEQUENTIAL PULL:** Pieces are numbered  $1, \dots, k$ , and in each slot each user pulls the lowest numbered piece it does not possess.

The following three theorems hold for any one-sided pull-based protocol, and for hard or soft constraints. The first is a negative result showing that the time needed to disseminate a fixed fraction of the pieces to a small fraction of users grows as the product of the number of pieces  $k$  and  $\log n$ . This means that using one-sided pull during initial dissemination eliminates the potential speedup due to file splitting.

*Theorem 1:* For any  $0 < \beta < 1$ , from an initial state in which  $k$  pieces are each present in one user, let  $T^\beta$  be the time taken till at least  $\beta k$  pieces are present in  $\frac{n}{\log n}$  users each, using any one-sided pull-based protocol. Then, for any  $\epsilon > 0$ ,

$$P [ T^\beta \geq \beta(1 - \epsilon) k \log n ] \geq 1 - n^{-c}$$

for all  $c > 0$ ,  $k$  polynomial in  $n$ , and  $n$  large enough.

The next theorem shows that, starting from a state where each piece is present in a fraction of the nodes, any pull-based protocol will deliver all pieces to all users in  $O(k + \log n)$  time, with high probability. Thus, pull finishes dissemination within a constant factor of the time needed by the optimal centralized protocol.

*Theorem 2:* Let  $0 < \eta < 1$  and consider any pull-based protocol. From a state such that each piece is in  $\eta n$  users, if  $T_\eta$  is the time till all users have all pieces then

$$P \left[ T_\eta \leq \left( \frac{\log(1 + \frac{\epsilon}{\eta})}{\log(1 + \frac{\eta}{e})} \right) k + \left( \frac{1 + c}{\log(1 + \frac{\eta}{e})} \right) \log n \right] \geq 1 - n^{-c} \quad (1)$$

for all  $c > 0$ , and any  $n$  and  $k$ .

Intuitively, the reason that pull protocols are efficient from a starting state such as the one in Theorem 2 is that each pull request has a probability greater than  $\eta$  of targeting a user who will be able to service the request.

The next theorem gives an upper bound on the completion time for a pull protocol.

*Theorem 3:* Consider a network with  $n$  users and  $k$  pieces, each initially present at one user, implementing a pull-based protocol for piece dissemination. If  $T$  is the first time that all users have all pieces, then given any  $\delta > 0$ , any  $c > 0$ ,  $n$  large enough, and  $k$  arbitrary,

$$P [ T \leq 4e(1 + \delta)(k \log k + (1 + c)k \log n) ] \geq 1 - 2n^{-c}$$

If  $k$  grows polynomially in  $n$ , then  $k \log k$  is  $O(k \log n)$ , and Theorem 3 implies an upper bound of  $O(k \log n)$ . Thus, Theorems 1 and 3 together show that the completion time for any one-sided pull-based protocol is  $\Theta(k \log n)$  with high probability.

One-sided push-based protocols are those where all communication occurs only via pushes, and piece selection is one-sided. An example of such a protocol is the following.

**RANDOM PUSH:** In each slot each user pushes a random piece from the set of pieces it possesses.

Unlike pull-based protocols, some push-based protocols may never reach completion. For example, if pieces are pushed in a strict predefined priority order by all users, then the spread of lower priority pieces may be suppressed by the spread of the higher priority ones. However, other push-protocols such as RANDOM PUSH will eventually reach completion. The following theorem shows that one-sided push-based protocols are slow in the final stages of dissemination.

*Theorem 4:* For  $0 < \beta < 1$ , from an initial state in which  $\beta k$  pieces are each absent in  $\frac{n}{\log n}$  users, let  $T^\beta$  be the time taken till all pieces are present in all users, using any push-based protocol. Then, for any  $\epsilon > 0$ ,

$$P [ T^\beta \geq \beta(1 - \epsilon) k \log n ] \geq 1 - n^{-c}$$

for all  $c > 0$ ,  $k$  polynomial in  $n$ , and  $n$  large enough.

Since the completion time grows linearly in the product of  $\beta k$  and  $\log n$ , Theorem 4 shows that purely push-based protocols provide no speedup from file splitting.

We now outline how the above results motivate the design of the hybrid efficient one-sided protocol INTERLEAVE. Theorems 1 and 4 show that any protocol relying on only one of the push or pull mechanisms can provide no speedup from file splitting. Also, pull protocols are inefficient near the start but efficient in the end, while push protocols are inefficient in the end. This indicates that it may be possible that a hybrid protocol, in which users execute pushes and pulls, can ensure efficient completion.

Theorem 2 shows that, from the viewpoint of achieving  $O(k + \log n)$  dissemination time, an intermediate state – in which each piece is present in  $\eta n$  users for some  $\eta > 0$  independent of  $n$  – is of fundamental importance. From such a state, user pulls in any hybrid protocol would enable

completion in  $O(k + \log n)$  time. To design a hybrid protocol whose overall completion time, from start to finish, is also  $O(k + \log n)$ , we would thus need to

- (a) design a push protocol that reaches the intermediate state in  $O(k + \log n)$  time, and
- (b) combine the above push protocol with a pull-based protocol in a decentralized way.

This is the idea underlying the design of the efficient protocol INTERLEAVE.

Working towards the first of the above two objectives, we notice that while Theorem 2 guarantees the efficiency of pulls in the end, there is no analogous theorem for the efficiency of push protocols in the beginning. In particular, some push protocols may not reach the intermediate state in  $O(k + \log n)$  time. We thus need to design a push protocol for this objective. Consider the one-sided push-based protocol PRIORITY PUSH.

**PRIORITY PUSH:**

- pieces are numbered  $1, 2, \dots$
- in each slot every user other than the source transmits a copy of the highest numbered piece it has received until that time.
- The source transmits piece number  $i$  in the time slots  $(i - 1)l + 1, \dots, il$ , such that  $l \geq 1$  is an integer called the *spacing* of the protocol.

*Theorem 5: Given any  $\delta > 0$  and  $0 < c < 1$ , if the PRIORITY PUSH protocol with spacing  $l$  is implemented, the probability that a given piece  $p$  reaches  $n(1 - e^{-l} - \delta)$  users within time  $(1 + \delta) \log_2 n$  after leaving the source is at least  $1 - 4n^{-c}$  for large enough  $n$ .*

Before we proceed with the design of an efficient hybrid one-sided protocol, we briefly comment on the use of PRIORITY PUSH for the case when the source has the ability to generate pieces that are forward-erasure-coded versions of the original file pieces. With forward erasure coding, each user now only needs to build a large enough set of distinct coded pieces to be able to recover the original file. A protocol based on a combination of rateless forward error correction at the source, as proposed for example in [18], and piece relay within the network using PRIORITY PUSH, would work as follows: the source pushes a new coded piece in every time slot, which is labeled with a piece number as required by the PRIORITY PUSH protocol. A user at any time would transmit the highest numbered (coded) piece it currently possesses. PRIORITY PUSH ensures that each user receives approximately 63.2% of the coded pieces emerging from the source, and each of these pieces will be received approximately  $\log_2 n$  time slots after it emerges from the source. This means that each user will be able to build up a large enough collection of coded pieces in a timely fashion, enabling the decoding of the source file. Such a combination of source coding and PRIORITY PUSH may be a good candidate in scenarios such that two-way communication between users is impossible or infeasible, as in this case the pulling of pieces by users would not be possible. The delay guarantee provided by PRIORITY PUSH means that it might also be a good candidate for the relaying of source-encoded

data that is of a streaming/real-time nature. Coded piece relay via PRIORITY PUSH is also compatible with other coding methods at the source.

We now return to the objective of designing a hybrid one-sided piece selection protocol that is efficient in the absence of coding. For  $k$  pieces and spacing  $l = 1$ , Theorem 5 implies that all but a vanishingly small fraction of the pieces will achieve good penetration within  $k + \log_2 n$  time slots. PRIORITY PUSH thus almost achieves objective (a) above. In particular, since the number of pieces that do not achieve good penetration is quite small, and so it may be possible to provably achieve efficient completion by combining PRIORITY PUSH with a suitable one-sided pull protocol. The SEQUENTIAL PULL protocol mentioned earlier is well-matched to PRIORITY PUSH: at any time, a pull request for a lower numbered piece is more likely to succeed than a request for a higher numbered piece, because in PRIORITY PUSH lower numbered pieces achieve good penetration before higher numbered ones. The choice of SEQUENTIAL PULL for interleaving with PRIORITY PUSH also enables us to provide delay guarantees for INTERLEAVE.

The performance guarantee of PRIORITY PUSH is more fragile than that of SEQUENTIAL PULL, and for this reason the hybrid protocol INTERLEAVE is designed so that the pulling does not interfere with the pushing. Interleaving in this non-interfering fashion can be done in a decentralized way, thus achieving objective (b) above.

**INTERLEAVE:**

- Pieces are numbered  $1, 2, \dots$
- In every odd time slot, the source pushes the piece with number one higher than the one it transmitted in the previous odd time slot. Every other user pushes the highest numbered piece it received in the previous odd time slots. The user may have a higher numbered piece obtained in an even time slot, but this is not the one chosen for transmission.
- In every even time slot, every user sends a pull request for the lowest numbered piece it does not already have. In this slot users do not distinguish pieces based on whether they were received in even or odd time slots.

*Theorem 6: If  $T^{k_1}$  is the time INTERLEAVE takes to disseminate the  $k_1$  lowest numbered pieces, then given any  $s < \frac{1}{2}$  we have that*

$$P [T^{k_1} \leq 10k_1 + 2(1 + \epsilon) \log_2 n] \geq 1 - 5n^{-s}$$

*for any  $\epsilon > 0$ ,  $k_1$  polynomial in  $n$ , and  $n$  large enough.*

The above theorem implies that, with high probability, INTERLEAVE achieves complete dissemination in time that is within a factor of ten of what an optimal fully centralized protocol could achieve. This means that it will be able to provide a significant file-splitting speedup for large networks.

The fact that users communicate pieces in rough order, and the delay guarantee of Theorem 6 for the lowest numbered

pieces, suggests that users receive lower numbered pieces before higher numbered ones. This indicates that INTERLEAVE, or protocols of a similar design, would be useful in peer-to-peer networks in which the data to be disseminated is of a real-time nature.

It is interesting to contrast the above performance guarantee of INTERLEAVE with the single-piece results of Karp et. al. [9]. In that paper, the authors obtain a lower bound of  $\Omega(n \log \log n)$  on the number of transmissions of the single piece that need to occur for complete dissemination, for any protocol relying on random user contacts of the kind studied in our paper. However, when there are multiple pieces, protocols can save bandwidth by pipelining across pieces. INTERLEAVE manages to do this pipelining in a way that results in  $O(n)$  transmissions per piece, which is the optimal order.

We now move on to consider two-sided piece selection protocols. Users carry out pushes/pulls, but have knowledge of the target's current state. We consider an initial state where  $n$  distinct pieces are present in the system, one in each user.<sup>7</sup> For this state, consider the following two-sided pull protocol:

ADVOCATE: If the user does not already possess the target's initial piece, it downloads that piece. Else it pulls a random piece from among those present in the target but absent in the user.

In this protocol each user acts as an advocate for its initial piece. If each user is restricted to downloading at most one piece in every time slot, an optimal central protocol will take at least  $n - 1$  time slots to complete. The following theorem shows that the ADVOCATE protocol completes in time very close to this optimal, with high probability.

*Theorem 7: Starting from an initial state where each user has one unique piece, the ADVOCATE protocol operating under soft constraints finishes in  $n + O(\log n)$  time with high probability.*

In the above theorem the pre-constant 1 of  $n$  is the best possible. The above theorem means that for large  $n$  the fraction of wasted time slots will be negligible.

#### IV. ONE-SIDED PROTOCOLS

In this section we sketch the proofs of Theorems 1-6, which deal with one-sided protocols.

##### A. One-sided Pull-based Protocols

Since Theorem 1 is a lower bound on completion time, there is no loss of generality in assuming soft constraints. The idea behind the proof is to use a probabilistic counting argument to lower bound the number of pull requests needed per piece. Since at most  $n$  pull requests occur in any given time slot, such a lower bound on the number of requests needed yields a lower bound on the dissemination time.

*Lemma 1: Consider a system with soft constraints, and an initial system state such that a given piece  $p$  is present in only*

<sup>7</sup>Completion from such an initial state has been previously studied, often under the alternate title of "all-to-all communication".

*one user. Let  $A$  be the number of pull requests for  $p$  needed till it is present in  $\frac{n}{\log n}$  users. Then given any  $\epsilon > 0$  and  $c > 0$ ,*

$$P[A \leq (1 - \epsilon)n \log n] < \frac{n^{-c}}{k}$$

*for any  $k$  that grows polynomially in  $n$ , and  $n$  large enough.* **Sketch of proof of Lemma 1:** The probability of success of any pull request increases in the number of pull requests occurring strictly before it. It is thus sufficient to assume that the pull requests for piece  $p$  occur in strict sequence, with no two being simultaneous. For such a sequence,  $Geo(\frac{i}{n})$  pull requests for  $p$  are needed before its occupancy goes from  $i$  to  $i + 1$  users<sup>8</sup>. Thus  $A$  is stochastically larger than the sum  $Geo(\frac{1}{n}) + \dots + Geo(\frac{1}{\log n})$ . In turn, the probability that the sum is less than  $(1 - \epsilon)n \log n$  can be shown to be as small required by the lemma. ■

**Proof of Theorem 1:** For a given pull protocol, let  $A_p$  be the number of pull requests for a particular piece  $p$  until it is present in  $\frac{n}{\log n}$  users. Since there are at most  $n$  pull requests in one time slot, it follows that for any time  $t$ ,

$$\begin{aligned} P[T^\beta < t] &\leq P\left[A_p < \frac{nt}{\beta k} \text{ for some piece } p\right] \\ &\leq \sum_p P\left[A_p < \frac{nt}{\beta k}\right] \end{aligned}$$

From Lemma 1 we see that, if  $n$  is large enough, choosing  $\frac{nt}{\beta k} = (1 - \epsilon)n \log n$  yields  $\sum_p P\left[A_p < \frac{nt}{\beta k}\right] < n^{-c}$ . This proves the theorem. ■

We now turn our attention to Theorem 2. Here, we assume without loss of generality that the system is operating under hard constraints. Any user needs at most  $k$  successful pull requests until its collection is complete. From the initial state of Theorem 2, the time to the next successful pull is always stochastically less than  $Geo(\frac{\eta}{e})$ <sup>9</sup>. Each user can thus be shown to complete in  $O(k)$  time, and by a union bound the slowest of  $n$  users can be shown to finish in  $O(k + \log n)$  time.

**Sketch of Proof of Theorem 2:** Using the above arguments, it can be shown that for all times  $t$ , and any  $n$  and  $k$ ,

$$P[T_\eta > t] \leq n e^{-\theta t} \left( \frac{\rho e^\theta}{1 - (1 - \rho)e^\theta} \right)^k \quad (2)$$

for  $\rho = \frac{\eta}{e}$  and  $0 < \theta < \log\left(\frac{1}{1 - \rho}\right)$ . Setting  $\theta = \log(1 + \rho)$ , the value of  $t$  that makes the RHS of (2) to  $n^{-c}$  proves the Theorem. ■

If users are implementing SEQUENTIAL PULL, all that is required for equation (2) to hold is that piece  $i$  be present in  $\eta n$  users by time slot  $i$ , instead of being so from the beginning. This is because users pull in sequence, and so will not pull

<sup>8</sup>For any  $0 < \alpha < 1$ ,  $Geo(\alpha)$  represents a geometrically distributed random variable:  $P[Geo(\alpha) > m] = (1 - \alpha)^m$  for integer  $m \geq 0$

<sup>9</sup>The target has the requested piece w.p.  $\eta$ , and will not be simultaneously targeted by any other user w.p.  $(1 - \frac{1}{n})^{n-1} > \frac{1}{e}$

piece  $i$  before time  $i$ . Choosing  $\theta$  close to 0 gives the following lemma, which is used in Section IV-C.

*Lemma 2:* Consider a scenario such that piece  $i$  is present in  $\eta n$  users by time slot  $i$ , for each  $i \in 1, \dots, k$ , and SEQUENTIAL PULL is implemented. Then, if  $T$  is the time till all users have all  $k$  pieces,

$$P \left[ T > \left( \frac{e}{\eta} + \epsilon \right) k + (1+c)M_\epsilon \log n \right] \leq n^{-c}$$

such that  $M_\epsilon$  is a constant. This is true for any  $n$  and  $k$ .

For the proof of Theorem 3, we first prove a stochastic upper bound on the number of pull requests for a given piece before it is present in  $\epsilon n$  users. We will then use this to upper bound the number of requests needed for all pieces to get to  $\epsilon n$  users each. Since at least  $n(1-\epsilon)$  pulls take place in every time slot until this state is reached, an upper bound on the total number of pull requests needed will provide a lower bound on the time taken for the system to reach such a state. For the remaining time to full completion, we will use Theorem 2.

*Lemma 3:* Let  $A$  be the number of pull requests for a piece  $p$  until it is present in all users. Then, for any  $c > 0$  and any  $k$  and  $n$ ,

$$P[A > (4e)n \log k + 4e(1+c)n \log n] < \frac{n^{-c}}{k}$$

**Sketch of proof of Lemma 3:** For any time  $t$  let  $a_t$  be the number of requests for  $p$  in that time slot, and  $N_t$  be the number of users who have  $p$  at time  $t$ . The variable  $a_t$  can vary greatly from slot to slot, because users can be pulling for other packets too. Let  $\mathfrak{F}_t$  denote the entire history until (and including) time  $t$ : it contains all the variables  $a_1 \dots a_t$  as well as  $N_1 \dots N_t$ .

Note that  $N_t \leq N_{t+1} \leq 2N_t$ . For any  $\theta > 0$ , and any  $0 < b \leq a \leq 2b$  we have that

$$\frac{1}{a^\theta} - \frac{1}{b^\theta} \leq \frac{-\theta(a-b)}{(2b)^{\theta+1}}$$

which implies

$$E \left[ \frac{1}{N_{t+1}^\theta} \middle| \mathcal{F}_t \right] - \frac{1}{N_t^\theta} \leq \frac{-\theta E[N_{t+1} - N_t | \mathcal{F}_t]}{(2N_t)^{\theta+1}}$$

Given  $N_t$ , the probability that any one of the  $a_t$  requests is successful is at least  $\frac{N_t}{ne}$ . This means that

$$\begin{aligned} E \left[ \frac{1}{N_{t+1}^\theta} \middle| \mathcal{F}_t \right] &\leq \frac{1}{N_t^\theta} - \frac{\theta}{(2N_t)^{\theta+1}} \frac{a_t N_t}{ne} \\ &\leq \left( e^{-\frac{\theta}{2^{\theta+1}e} \frac{a_t}{n}} \right) \frac{1}{N_t^\theta} \end{aligned} \quad (3)$$

Let  $\beta = \exp(\frac{\theta}{2^{\theta+1}en})$  and define the quantity  $M_t = N_t^{-\theta} \beta \sum_{s=1}^{t-1} a_s$ . Then, using (3) it can be shown that  $M_t$  is a supermartingale with respect to  $\mathfrak{F}_t$ .

Let  $T$  be the number of time slots required for  $n$  successes. Since  $M_1 = N_1 = 1$ ,

$$1 \geq E \left[ \frac{\beta \sum_{s=1}^T a_s}{N_T^\theta} \right] \geq E \left[ \frac{\beta \sum_{s=1}^T a_s}{n^\theta} \right]$$

For any number of requests  $F$ ,

$$P \left[ \sum_{s=1}^T a_s > F \right] \leq \frac{E[\beta \sum_{s=1}^T a_s]}{\beta^F} \leq \frac{n^\theta}{\beta^F}$$

Setting the RHS of the last inequality above to  $\frac{n^{-c}}{k}$ , setting  $\theta = 1$  and substituting the value of  $\beta$  yields

$$F = (4e)n \log k + 4e(1+c)n \log n$$

This proves the lemma.  $\blacksquare$

**Sketch of proof of Theorem 3:** Set  $\epsilon = \frac{\delta}{2+\delta}$ , and let  $T_\epsilon$  be the first time that each piece is in at least  $\epsilon n$  users. Until  $T_\epsilon$ , there will be at least  $n(1-\epsilon)$  pulls in each time slot. If  $t$  is chosen so that  $n(1-\epsilon)t = k((4e)n \log k + 4e(1+c)n \log n)$ , then there will be enough pulls for each piece to be disseminated to  $\epsilon n$  users with high probability, by Lemma 3. This gives an upper bound on  $T_\epsilon$ , which can be combined with the upper bound on  $T - T_\epsilon$  given by Theorem 2 to obtain the required upper bound on  $T$ .  $\blacksquare$

### B. One-sided Push-Based Protocols

The proof of Theorem 4 about the inefficiency of any one-sided push protocol in the final stages is similar to the proof of Theorem 1. Lemma 4 below is analogous to Lemma 1 for pull, can be proved in a similar fashion and leads to Theorem 4 in the same way that Lemma 1 leads to the proof of Theorem 1.

*Lemma 4:* Consider an initial system state such that a given piece  $p$  is absent in  $\frac{n}{\log n}$  users. Let  $A$  be the number of pushes for  $p$  needed till it is present in all users. Then given any  $\epsilon > 0$  and  $c > 0$ , for any pull-based protocol,

$$P[A \leq (1-\epsilon)n \log n] < \frac{n^{-c}}{k}$$

for  $k$  polynomial in  $n$  and  $n$  large enough.

*The PRIORITY PUSH Protocol:* The PRIORITY PUSH protocol is described in Section III. In this section we present the idea that simplifies its analysis when all pieces are initially at one source. Consider a particular piece  $p$ , and let time be counted so that the source first transmits  $p$  in time slot 1. Note that the spread of piece  $p$  and subsequent pieces is not affected by the spread of pieces preceding  $p$ . For any time  $t$ , let  $A_t$  be the number of users transmitting  $p$  in time slot  $t$ . We are interested in the process  $A$ , and for this purpose all higher numbered pieces are equivalent since they cause similar interference to the spread of  $p$ . Thus, let  $B_t$  be the number of users transmitting higher numbered pieces in time slot  $t$ . So, at any time, a user may be counted either in  $A$ , in  $B$ , or as not transmitting pieces that are numbered  $p$  or higher.

At this point it is useful to briefly recall the setting of the classical random gossip problem described in [10]. In this setting, there are  $n$  users and only one message, which is initially present with one user. Users with messages are called *informed*. In every time slot, every informed user contacts another user chosen uniformly at random from the set of all users and sends (pushes) the message to this user, who is then also informed. Let  $Y_t$  be the number of informed users at time

$t$ , and call the process  $(Y_t : t \geq 0)$  the *classical gossip process*, with the initial condition being  $Y_0 = 1$  as it is in [10].

With this background, the following two observations are easy to make:

- The process  $(A_{t+1} + B_{t+1} : t \geq 0)$  is stochastically identical to the classical gossip process  $(Y_t : t \geq 0)$ . In short,  $(A_{t+1} + B_{t+1} : t \geq 0) \stackrel{d}{=} (Y_t : t \geq 0)$ <sup>10</sup> and in particular  $A_{t+1} + B_{t+1} \stackrel{d}{=} Y_t$  for each  $t$ .
- The  $B$  process is also stochastically identical to a delayed  $Y$  process:  $(B_{t+l+1} : t \geq 0) \stackrel{d}{=} (Y_t : t \geq 0)$ .

Thus,  $A_t + B_t \stackrel{d}{=} B_{t+l}$  for all  $t \geq 1$ , and thus also  $E[A_t] = E[B_{t+l} - B_t]$ . It can be shown that the values of  $A_t$  and  $B_t$  are tightly concentrated around their expected values for all times  $t$ , and thus  $A_t \gtrsim (1 - \epsilon)(B_{t+l} - B_t)$  with high probability, until the time that  $B_t \approx n$ . Adding up over time, this means that with high probability

$$\sum A_t \gtrsim nl(1 - \epsilon) \quad (4)$$

Now, the quantity  $\sum A_t$  above is the total number of transmissions of  $p$  by time  $T$ . The target of each of these transmissions is chosen uniformly at random, independently of other transmissions. Thus the number of users who have piece  $p$  by time  $T$  will have the same distribution as the number of bins with at least one ball in an experiment such that  $\sum A_t$  balls are thrown into  $n$  bins. Since  $\sum_t A_t \gtrsim nl(1 - \epsilon)$ , this means that the number of users receiving  $p$  is  $\approx n(1 - e^{-l})$  with high probability. This is the idea behind the proof of Theorem 5.

### C. INTERLEAVE

In this section we analyze the performance of INTERLEAVE, and prove Theorem 6. The following two observations about INTERLEAVE facilitate its analysis:

- The pulling does not interfere with the pushing: if the system is sampled only in the odd time slots, the pieces pushed will be identical to an alternate system running only PRIORITY PUSH. In particular, the pushing of higher numbered pieces is unaffected by the spread of lower numbered pieces.
- Within the pulling in the even time slots, the spread of higher numbered pieces does not interfere with the pulling of lower numbered pieces.

Call a piece *failed* if it does not reach  $\frac{ne}{5}$  users within  $2(1 + \epsilon) \log_2 n$  of being pushed by the source. Note that  $\frac{e}{5} < 1 - e^{-1}$ , and hence the PRIORITY PUSH operating in the odd time slots will ensure that  $P[p \text{ fails}] < n^{-c}$  for  $0 < c < 1$ , and  $n$  large enough. The following lemma gives an upper bound on the dissemination time  $T^{k_1}$  of the  $k_1$  lowest-numbered pieces which have a given fixed number of failed pieces.

*Lemma 5: Let  $\mathcal{E}$  be the event that  $\{q_1, \dots, q_m\}$  is the set of all failed pieces numbered less than or equal to  $k_1$ , with  $q_i < q_{i+1}$ . Then, for any  $c > 0$  and  $n$  large enough,*

$$P \left[ T^{k_1} > \left( \begin{array}{l} 10k_1 + 2(1 + \epsilon) \log_2 n \\ + m\gamma(1 + c)(\log n + \log m) \end{array} \right) \middle| \mathcal{E} \right] \leq 2n^{-c}$$

<sup>10</sup>The symbol  $\stackrel{d}{=}$  denotes equality in distribution

such that  $\gamma$  is a constant independent of  $n$

**Proof of Lemma 5:** For each  $i \in 1, \dots, k$  let  $T^i$  be the first time that each piece in  $1, \dots, i$  is present in all users, and

$$\widehat{T}^i = \max \{T^i, 2i + 2(1 + \epsilon) \log_2 n\}$$

Consider any two successive failed pieces  $q_j$  and  $q_{j+1}$ , and let  $q_{j+1} - q_j = l$ . Then, for each  $1 \leq s \leq l - 1$ , piece number  $q_j + s$ , which has not failed, will be present in at least  $n(1 - e^{-1} - \epsilon)$  users by time  $\widehat{T}^{q_j} + 2s$ . Also, after  $\widehat{T}^{q_j}$  all users will be pulling pieces numbered  $q_j + 1$  or higher. For this scenario, Lemma 2 implies that all users will obtain all pieces  $i$  such that  $q_j < i < q_{j+1}$  within

$$\left( \frac{1}{\rho} + \epsilon \right) (q_{j+1} - q_j - 1) + M_\epsilon(1 + c) \log n$$

pull slots, with probability at least  $1 - n^{-c}$ , where  $\rho = \frac{1 - e^{-1} - \epsilon}{e}$ . Choosing  $\epsilon$  so small that  $\frac{1}{\rho} + \epsilon < 5$ , this implies that

$$\widehat{T}^{q_{j+1}-1} \leq \widehat{T}^{q_j} + 10(q_{j+1} - q_j) + 2M_\epsilon(1 + c) \log n$$

with probability at least  $1 - n^{-c}$ . Also, by time  $\widehat{T}^{q_{j+1}-1}$  all users will be pulling for pieces  $q_{j+1}$ , or higher pieces if they already have  $q_{j+1}$ . By Theorem 3 with  $k = 1$ ,

$$\widehat{T}^{q_{j+1}} - \widehat{T}^{q_{j+1}-1} \leq 8e(1 + \delta)(1 + c) \log n$$

with probability at least  $1 - n^{-c}$ . The last two inequalities above imply that

$$\widehat{T}^{q_{j+1}} \leq \widehat{T}^{q_j} + 10(q_{j+1} - q_j) + \gamma(1 + c) \log n \quad (5)$$

with probability at least  $1 - 2n^{-c}$ , such that  $\gamma = 8e(1 + \delta) + 2M_\epsilon$ . Defining  $\widehat{T}^{q_0} = 2(1 + \epsilon) \log_2 n$  and  $q_{m+1} = k_1$ , and summing (5) over all  $j$  shows that

$$\widehat{T}^{k_1} \leq 10k_1 + 2(1 + \epsilon) \log_2 n + m\gamma(1 + c) \log n$$

with probability at least  $1 - 2mn^{-c}$ . Replacing  $c$  by  $c + \frac{\log m}{\log n}$  proves the lemma. ■

We are now ready to prove the performance guarantee of the INTERLEAVE protocol as given in Theorem 6.

**Proof of Theorem 6:** Let  $m$  be the number of failed pieces in  $\{1, \dots, k_1\}$ . Then, by the Markov inequality we have that

$$P[m > k_1 n^{-s}] \leq \frac{E[m]}{k_1 n^{-s}}$$

By Theorem 5, if  $n$  is large enough the probability piece  $p$  fails is less than  $4n^{-2s}$  for  $s < \frac{1}{2}$ . This means that  $E[m] \leq 4k_1 n^{-2s}$  and so

$$P[m > k_1 n^{-s}] \leq 4n^{-s}$$

Now, in Lemma 5, if  $m \leq k_1 n^{-s}$  then the fact that  $k_1$  is polynomial in  $n$  implies that  $m\gamma(1 + c)(\log n + \log m)$  is  $o(k_1)$ : for  $n$  large enough its value will be less than  $\epsilon k_1$ . This means that

$$P \left[ T^{k_1} > 10k_1 + 2(1 + \epsilon) \log_2 n \mid m \leq k_1 n^{-s} \right] \leq 2n^{-s}$$

The theorem follows from the above two inequalities. ■

## V. TWO-SIDED PROTOCOLS

We sketch a proof for Theorem 7, on the performance of the ADVOCATE protocol. The idea is to break the evolution into stages, and show that each user is successful in each time slot in each stage.

By any given time, a user would have contacted a list of other users and would definitely be in possession of the corresponding pieces. The user's *primary collection* at that time consists of the pieces corresponding to users it has contacted directly by that time. All other pieces in its collection are defined to be *secondary*.

At the very beginning, users will contact a new other user with high probability, and hence their primary collections will grow at close to the rate of one per time slot. In fact, given some  $\delta < \frac{1}{4}$ , it can be shown that all users will have at least  $n^{\frac{1}{4}-\delta}$  pieces by time  $n^{\frac{1}{4}-\delta} + 2$ , with probability greater than  $1 - n^{-2\delta}$ .

After this time, users will start repeating contacts more often and their primary collections will grow more slowly. At time  $t$ , the size of a particular user's primary collection will be  $\approx n(1 - e^{-\frac{t}{n}})$ , and the size of the intersection of the primary collections of a user and its target will be  $\approx n(1 - e^{-\frac{t}{n}})^2$ . Thus, the number of pieces in the primary collection of the target that are not present in the primary collection of the user will be  $\approx n(1 - e^{-\frac{t}{n}})e^{-\frac{t}{n}}$ . As long as the size of the user's secondary collection is smaller than this number, the user will be able to download a new piece from the target. Now, a user's total collection has size at most  $t$ , and hence a useful transfer will occur as long as

$$t - n(1 - e^{-\frac{t}{n}}) < n(1 - e^{-\frac{t}{n}})e^{-\frac{t}{n}}$$

Or, stated another way, in each time slot each user will be able to download a new piece up until time  $n\hat{t}$ , for any  $\hat{t} \geq 0$  such that  $e^{-2\hat{t}} < 1 - \hat{t}$ .

We use the above argument to show that all users are successful in all or all but one time slots up until some time  $\frac{n}{2}$ . Then, let  $L = 8 \log n$ , and let  $\epsilon = \frac{3 \log n}{n}$  and let a set  $A$  of pieces with  $|A| = L$  be "bad" if it exists in less than  $n(1 - \epsilon)$  users. With these definitions, for any  $t \geq \frac{n}{2}$  it can be shown that the probability of there being a bad set at any time  $t$  is less than  $e^{-(\log n)^2}$ . A union bound over  $\frac{n}{2} \leq t \leq n$  shows that there will be no bad set in these times. This implies that each user will have at least  $n - 8 \log n$  pieces by time  $n - 4 \log n$  with high probability.

After this point, each user has been contacted by around  $n(1 - e^{-1})$  other users and hence each piece is in at least that many users. Thus each pull will be successful with probability at least  $1 - e^{-1}$ , and hence the  $n$  users will finish their collections in  $O(\log n)$  more time. This gives us an overall completion time of  $n + O(\log n)$ .

## VI. SIMULATIONS

In this section we investigate the performance of the PRIORITY PUSH and INTERLEAVE protocols using simulations. In the system model analyzed in this paper each user has the ability to communicate with another user chosen at random from the entire network. A more realistic assumption might be

to let each user have only a limited view of the network that does not change over time. Specifically, we assume that each user has a fixed list of a small number of other users, which we shall refer to as its *contact list*. A user only pushes to and pulls from other users in its contact list. Each contact list is generated randomly, independent of other contact lists. It remains constant for all time.

Consider now the case that users implement INTERLEAVE, but in each time slot a user communicates with a neighbor chosen at random from within its contact list. The source however still pushes pieces in every other time slot to another user chosen uniformly at random from the set of all users. Figure 1 displays the observed time taken for  $k = 1000$  pieces to be disseminated to  $n = 500$  users, versus the size of the contact lists. From this we can see that if each user has a contact list of size 8 or more, the completion time using INTERLEAVE is close to  $2(k + \log_2 n) \approx 2020$ , which is much better than the  $10k + 2 \log_2 n$  predicted by Theorem 6. This difference suggests that the proof technique for Theorem 6 is conservative.

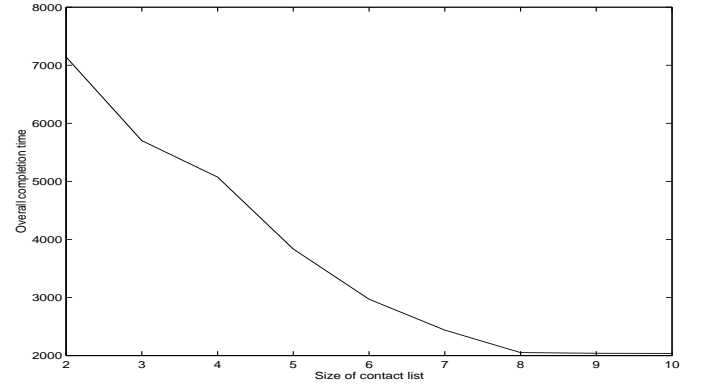


Fig. 1. The completion times of INTERLEAVE for  $n = 500$  users and  $k = 1000$  pieces, versus the size of the user contact lists.

Besides the overall completion time, we are also interested in the time a typical piece takes to get from the source to a typical user. Specifically, if a piece  $i$  emerges from the source at time  $t$  and reaches a user  $j$  at time  $t + d$ , we say that  $\text{delay}(i, j) = d$ . The *average delay profile*  $D(d)$  is the average of  $\mathbf{1}_{\text{delay}(i, j) \leq d}$  over all possible choices of user  $i$  and piece  $j$ :

$$D(d) = \frac{1}{nk} \sum_{i=1}^n \sum_{j=1}^k \mathbf{1}_{\text{delay}(i, j) \leq d}$$

where  $\mathbf{1}_{\text{delay}(i, j) \leq d}$  is 1 if and only if  $\text{delay}(i, j) \leq d$  and 0 otherwise.

Different piece selection protocols have different average delay profiles. Also, for a given dissemination protocol, the average delay profile will vary with the size of each user's contact list. A delay profile rising further to the left implies, on average, faster dissemination of pieces. Figure 2 plots the average delay profiles for  $k = 1000$  pieces being disseminated to  $n = 500$  users, for different choices  $m$  of user contact list size. From the figure we see that users having contact lists of size two leads to poor performance, but with four or

five contacts the average delay profile is comparable to that achieved if each user were to have a complete view of the network.

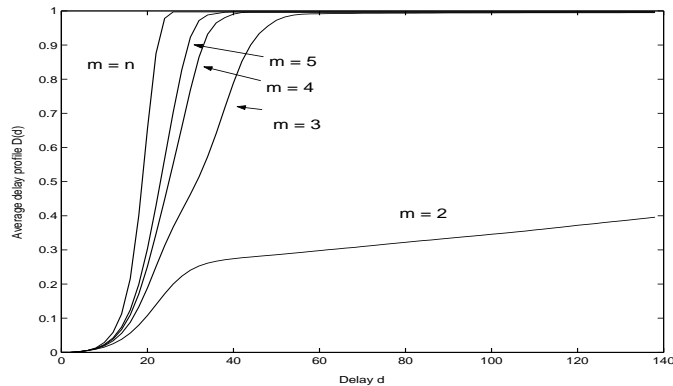


Fig. 2. The average delay profiles of INTERLEAVE for  $n = 500$  users and  $k = 1000$  pieces, for different sizes  $m$  of user contact lists.

We now turn our attention to the PRIORITY PUSH protocol. Figure 3 plots the average delay profiles for different choices of the spacing  $l$ , if each user has an entire view of the network. Recall that the source transmits a new piece every  $l$  time slots. The final limiting value of each delay profile represents the final fraction of users a typical piece reaches. This is equivalent to the fraction of pieces a typical user ultimately receives. As predicted by Theorem 5, a spacing of  $l$  has a limiting value of approximately  $1 - e^{-l}$ .

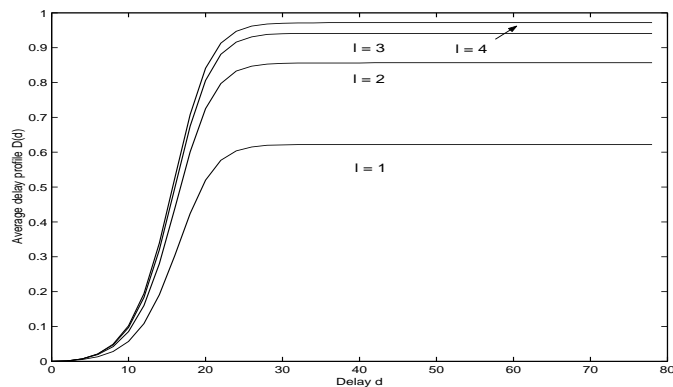


Fig. 3. The average delay profiles of PRIORITY PUSH for  $n = 500$  users and  $k = 1000$  pieces, for different values of spacing  $l$ .

## VII. DISCUSSION

In this work we

- investigated the speedup achieved in file dissemination by breaking the file into pieces.
- investigated the performance of one-sided piece selection protocols,
- designed the efficient piece selection protocol INTERLEAVE, and
- illustrated why the PRIORITY PUSH protocol would be useful for the relay of source-coded pieces.

- designed an efficient two-sided protocol for all-to-all exchange.

We believe that the techniques and results of this work will aid in the understanding of systems that involve the decentralized dissemination of large files. We would like to emphasize that the dissemination of multiple pieces of data over unstructured networks is significantly different from the dissemination of a single piece, but also note that insights gained from single-piece dissemination can be effectively leveraged to design protocols for multiple pieces. It would be interesting to investigate the spread of multiple pieces in unstructured networks using more detailed system models.

## REFERENCES

- [1] "Cachelogic research: Peer to peer in 2005," <http://www.cachelogic.com/research/p2p2005.php>.
- [2] B. Cohen, "Incentives build robustness in BitTorrent," *P2PECON Workshop*, 2003.
- [3] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: high-bandwidth multicast in cooperative environments," in *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*. New York, NY, USA: ACM Press, 2003, pp. 298–313.
- [4] S. Ratnasamy, M. Handley, R. M. Karp, and S. Shenker, "Application-level multicast using content-addressable networks," in *NGC '01: Proceedings of the Third International COST264 Workshop on Networked Group Communication*. London, UK: Springer-Verlag, 2001, pp. 14–29.
- [5] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron, "SCRIBE: A large-scale and decentralized application-level multicast infrastructure," *IEEE Journal on Selected Areas in communications (JSAC)*, vol. 20, no. 8, pp. 1489–1499, 2002.
- [6] X. Yang and G. de Veciana, "Service capacity of peer to peer networks," in *INFOCOM*, 2004, pp. 1–11.
- [7] D. Qiu and R. Srikant, "Modeling and performance analysis of BitTorrent-like peer-to-peer networks," *ACM SIGCOMM*, pp. 367–78, Sept. 2004.
- [8] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, "Epidemic algorithms for replicated database maintenance," in *PODC '87: Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*. New York, NY, USA: ACM Press, 1987, pp. 1–12.
- [9] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking, "Randomized rumor spreading," in *FOCS '00: Proceedings of the 41st Annual Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society, 2000, p. 565.
- [10] A. Frieze and G. Grimmett, "The shortest path problem for graphs with random arc lengths," *Discrete Applied Mathematics*, pp. 57–77, 1985.
- [11] B. Pittel, "On spreading a rumor," *SIAM Journal of Applied Mathematics*, vol. 47, no. 1, pp. 213–23, 1987.
- [12] N. Bailey, *The mathematical theory of infectious diseases and its applications*. Hafner Press, 1975.
- [13] P. Kermani and L. Kleinrock, "Virtual cut-through: A new computer communication switching technique," *Computer Networks*, vol. 3, pp. 267–286, January 1979.
- [14] S. Deb and M. Medard, "Algebraic gossip: A network coding approach to optimal multiple rumor mongering," *Allerton Conference on Communication, Control, and Computing*, 2004.
- [15] A. Bar-Noy and S. Kipnis, "Broadcasting multiple messages in simultaneous send/receive systems," *Discrete Applied Mathematics*, vol. 55, pp. 95–105, 1994.
- [16] E. Cockayne and A. Thomason, "Optimal multi-message broadcasting in complete graphs," *Proceedings of the eleventh SE conference on combinatorics, graph theory and computing*, pp. 181–99, 1980.
- [17] A. Farley, "Broadcast time in communication networks," *SIAM Journal of Applied Mathematics*, vol. 39, no. 2, pp. 385–90, 1980.
- [18] M. Luby, "Lt codes," in *The 43rd Annual IEEE Symposium on Foundations of Computer Science*, 2002, pp. 271–282.