Hindawi Publishing Corporation
Advances in Multimedia
Volume 2007, Article ID 84150, 12 pages
doi:10.1155/2007/84150

Research Article

Analysis and Implementation of Gossip-Based P2P Streaming with Distributed Incentive Mechanisms for Peer Cooperation

Sachin Agarwal,[1] Jatinder Pal Singh,[1] and Shruti Dube[2]

[1] *Deutsche Telekom AG Laboratories, Ernst-Reuter-Platz 7, 10587 Berlin, Germany*
[2] *Department of Computer Science and Engineering, Indian Institute of Technology, Kanpur 208 016, India*

Received 30 January 2007; Revised 29 June 2007; Accepted 15 August 2007

Recommended by Shigang Chen

Peer-to-peer (P2P) systems are becoming a popular means of streaming audio and video content but they are prone to bandwidth starvation if selfish peers do not contribute bandwidth to other peers. We prove that an incentive mechanism can be created for a live streaming P2P protocol while preserving the asymptotic properties of randomized gossip-based streaming. In order to show the utility of our result, we adapt a distributed incentive scheme from P2P file storage literature to the live streaming scenario. We provide simulation results that confirm the ability to achieve a constant download rate (in time, per peer) that is needed for streaming applications on peers. The incentive scheme fairly differentiates peers' download rates according to the amount of useful bandwidth they contribute back to the P2P system, thus creating a powerful quality-of-service incentive for peers to contribute bandwidth to other peers. We propose a functional architecture and protocol format for a gossip-based streaming system with incentive mechanisms, and present evaluation data from a real implementation of a P2P streaming application.

1. INTRODUCTION

Peer-to-peer (P2P) applications are widely used to distribute media files over the Internet and are sometimes referred to as "file-sharing" applications. P2P applications for real-time audio and video streaming have also been proposed and implemented in various academic and commercial settings (see Section 2). These applications are becoming popular with end users who wish to be content providers but do not have high-bandwidth upstream connections to stream their videos to other users on the Internet.

All P2P content delivery systems work on the premise that the peers will share their resources in order to increase the total service capacity of the P2P system. In the case of file sharing and streaming, the resource is the upstream bandwidth each peer contributes to distribute content to other peers. Peers download parts (blocks) of the media file or stream from various other peers and then reassemble these blocks. Thus P2P systems derive bandwidth from the participating peers who operate independently of each other. A mechanism, that creates compelling incentives for all peers to contribute resources and thus guards against bandwidth starvation in the P2P system, is needed to sustain peer interest in sharing bandwidth.

Many implemented P2P protocols have incentive mechanisms that encourage peers to contribute resources and participate in this "block sharing." For example, the BitTorrent file-sharing application's tit-for-tat incentive mechanism [1] rewards peers who share more bandwidth by allocating better download rates to them.

Unlike file sharing, live video streams have a time-limited "value" of a stream block, because older stream blocks become obsolete as time progresses, with older blocks no longer being "live." Even when a prerecorded stream is multicast and *played back concurrently* on the peers as they download the stream, peers seek to download contiguous blocks instead of random blocks in order to be able to play back the stream in real time as it is being downloaded. Thus stream blocks cannot be downloaded in a random order and BitTorrent's tit-for-tat scheme of bartering random blocks is not directly applicable for the case of live P2P streaming.

Some P2P streaming protocols for dissemination of video content distribution are centered around tree-based overlays, as surveyed in Section 2. A difficulty with most tree-based protocols is that the concept of "equality" amongst peers is inherently missing owing to the hierarchical tree structure, and so the formulation of a distributed incentive scheme is nontrivial. Another approach is gossip-based P2P video

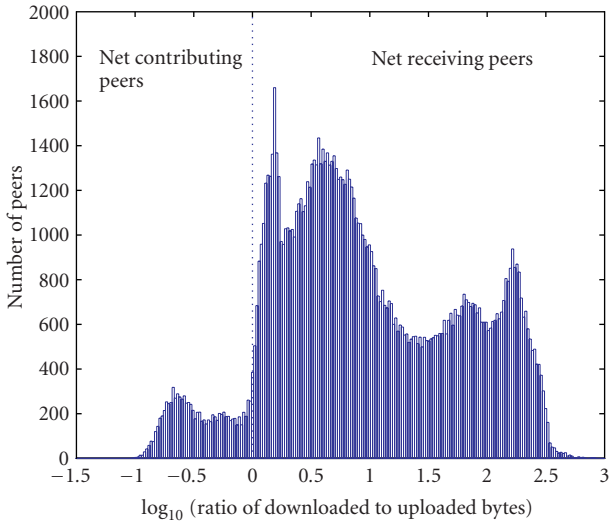

Figure 1: Histogram of the logarithm of the ratio of bytes sent to bytes received.

streaming, where stream blocks are "spread" amongst the peers using random gossip, as explained in Section 3. Creating incentive mechanisms for this class of protocols is one of the objectives of our work.

In order to further motivate our work, we obtained and analyzed log data from Roxbeam Media Networks (commercial offshoot of Cool Stream) to understand the overall distribution of peers' resource contributions to the P2P streaming effort. The particular log we analyzed was of a 4-hour baseball game streamed at 759 kbps video bitrate to about 120,000 unique IP addresses. The aim of this analysis was to highlight that many peers contribute very little in terms of bandwidth, and a few peers contribute a lot.

Figure 1 plots the distribution of the logarithm of the ratio of number of bytes downloaded to that uploaded by a peer. Most peers download more than they upload. Figure 1 clearly shows the inequality in P2P streaming systems: most peers contribute lesser than they download, and a few peers end up contributing the bulk of the bandwidth. We are motivated by this fact to develop an incentive system to reward cooperative, contributing peers in P2P streaming systems with better video streaming quality.

We provide a generic achievability argument for creating incentives to cooperate in gossip-based live P2P streaming. We do this by proving that after coupling a randomized optimistic unchoke component with an incentive mechanism, we can (still) distribute a stream block from the source to n peers in $O(\log n)$ time with high probability. This result is very significant because tree-based P2P streaming protocols also have logarithmic distribution times (because of the $O(\log n)$ tree height).

We employ a distributed incentive algorithm previously proposed for P2P file sharing and distributed file storage to demonstrate that this incentive mechanism is able to differentiate between peers on the basis of how much useful bandwidth each peer contributes, while still being in line with our analytical results for logarithmic completion time. Furthermore, we propose a functional architecture and protocol format for peers to implement and execute gossip-based streaming with incentive mechanisms. Finally, we provide simulation and real-implementation results to verify our claims.

We survey related work in overlay streaming multicast, gossip protocols, and distributed incentive mechanisms in Section 2. In Section 3, we elaborate on our model of a gossip-based overlay multicast protocol. The main result on the achievability of an incentive mechanism for gossip-based P2P streaming is provided in Section 4. We provide simulation results in Section 5 to verify the analytical results of Section 4. We then provide the functional architecture and protocol format for gossip streaming in Section 6. Finally, we provide performance results from implementation on a test bed in Section 7 and discuss some practical implementation aspects in Section 8. Conclusions and future work are presented in Section 9.

2. RELATED WORK

P2P systems are widely used to distribute content in the Internet, and it is estimated that a major portion of the bandwidth available on the consumer ISP networks carries P2P content [2]. It has been shown through analysis [3, 4], simulations, and measurements [5–7] that the P2P content delivery model scales gracefully with user demands in heterogeneous P2P networks. A majority of popular P2P systems are built around file sharing applications. These applications typically distribute stored and not live content, and the downloaded content is played back only after the entire file has been downloaded.

A large number of P2P streaming multicast solutions are based around setting up an overlay tree with the content source being the root and terminals being the other nodes of the tree [8–10]. This approach has the disadvantage of limited robustness against peer disconnections that disrupt stream reception on downstream neighbors. Many techniques [11] for alleviating this problem based on redundant data paths, multiple overlay trees, and multiple description coding have been proposed. We refer the interested reader to a good survey of P2P streaming in [12].

A comparison of the approaches and algorithms employed in various P2P streaming overlays can be found in [13, 14]. There have also been recent studies of commercial P2P streaming systems, for example [15–20], that study networking characteristics of some commercial P2P systems such as SopCast [21, 22], PPLive [23], Coolstreaming [24, 25] and Gridmedia [26].

Gossip protocols were first introduced as a means to achieve the lazy database replication [27]. The underlying architecture of BitTorrent [1] also uses a gossip-like randomized protocol to propagate data to peer hosts. The Pbcast protocol [28] uses the combination of a multicast tree and a gossip protocol to achieve robust content delivery. Recent work [29] confirms the utility of gossip-based protocols for media streaming applications. The Cool-stream technology [**?**] is a real-world implementation of such a gossip-based TV quality video stream delivery system.

3. GOSSIP-BASED P2P STREAMING

A gossip-based P2P streaming algorithm invokes a randomized gossip protocol on each stream block repeatedly to spread the blocks in the P2P network. Peers contact other peers randomly to spread the blocks through the network, much like an infectious virus spreads in a population or a rumor spreads in a crowd. This "gossiping" can be pull-based or push-based depending upon whether peers actively seek out the blocks they require (pull), or whether peers actively advertise (push) blocks to other peers.

For P2P streaming, the source peer only uploads blocks of the stream to randomly chosen peers. Unlike unicast, the source peer does not distribute a unique copy of the content to every recipient peer, and this characteristic underlines the P2P value: the source peer does not need a high-speed upload connection to simultaneously stream to a large population of recipient peers.

In the case of video streaming when recipients want to playback the received stream in real time (as it is being downloaded), recipient peers seek contiguous blocks of the stream so that the stream can be rebuilt from the downloaded blocks and played back instantaneously. We assume that each peer caches the downloaded stream and is capable of serving this stream to other peers. In case the stream is very long, only a certain stream window can be saved, but we discount this aspect in our analysis.

4. ANALYSIS

4.1. Incentives

The natural way of implementing an incentive mechanism is to make the bandwidth received by a peer a function of the bandwidth contributed by that peer. Ideally, the algorithm should be distributed with serving peers making decisions about how much bandwidth to allocate to other peers using local information only, without centralized control. We adopt the peer-wise proportional fairness scheme analyzed in [30, 31] to achieve this objective.

Let u_i be the total utility offered by peer i to the P2P network. At time t, peer i serves $u_{ij}(t)$ to peer j, computed on a peer-wise proportional basis, that is,

$$u_{ij}(t) = \frac{u_i}{\sum_{l=1}^{n} R_l(t) \sum_{k=0}^{t-1} u_{li}(k)} R_j(t) \sum_{k=0}^{t-1} u_{ji}(k), \qquad (1)$$

with the understanding that $0/0 = 0$ and $R_j(t)$ is set to 1 if peer j is requesting bandwidth from peer i, and 0 otherwise. (1) allocates peer i's offered utility u_i to all the other peers in the overlay requesting the utility based upon how much cumulative utility each peer has contributed to peer i in the past time.

The "utilities" in (1) can be the bandwidth (in terms of kbps, e.g.). In general, a peer i is free to assign any utility value to peer j depending on how peer i perceives the utility of peer j offered to itself (u_{ji}). For example, a noncooperative peer i may set all received utilities by setting u_{ji} to 0 for all j and then $u_{ij} = 0$, for all j according to (1), thus cutting off all contributions to other peers in the P2P network. Then the other peers receive no service from peer i and would decrease their contribution to peer i according to (1). Note that all peers only make local measurements of the utility they have received from other peers, thus making peer-wise proportional fairness a completely distributed algorithm.

In a live multimedia streaming application, peers gossip in order to discover the next useful (contiguous) block of the live stream amongst their peers. So even though a peer may be offering bandwidth to another peer, the utility of this bandwidth might be 0 in case the offering peer has no useful block to share with the other peer, as might be the case when new peers join the network. Thus per-stream fairness is not guaranteed by peer-wise proportional fairness. As we shall show later through simulations, the peer-wise proportional fairness algorithm is fair in the asymptotic sense (i.e., across multiple streams and over a long time).

The peer-wise proportional fairness criteria is biased toward peers who join the overlay streaming session early on and "build up credit" with other early peers. To allow newer peers to start downloading the stream, a fraction of the total utility of each peer called "optimistic unchoke" utility (borrowing a term from BitTorrent [1]) is offered to any requesting peer independent of the utility the requesting peer had offered previously. This optimistic unchoke utility also preserves the logarithmic completion time properties of gossip streaming, as we prove below.

4.2. Gossip under an incentive scheme

We model our P2P streaming system as a completely connected P2P network comprising n peers. For analysis, we assume that time is divided into discrete time slots (or rounds). Peer i can gossip with f_i other peers in one time slot (this is the fan-out of peer i, $f_i \geq 0$). Further, each peer assigns a finite fraction of f_i for uniform random gossip; this is the *optimistic unchoke* fraction δ_i of peer i.

We must show that the important property of a gossip protocol—in which each stream block is received with high probability by all peers in $O(\log n)$ time-slots—is valid with the incentive scheme, because our incentive scheme changes the *random* gossip model of classical gossip protocol analysis. This is important for comparing gossip-based streaming protocols to tree-based protocols, where the height of the tree ($\log n$) determines the worst case time before a stream block reaches every peer (for a balanced tree).

Formally, we seek to prove that under the assumption of all peers using a fraction of their fan-outs for optimistic unchoke, a stream block will reach all peers in $O(\log n)$ time slots with high probability. Without loss of generality, we consider the situation when the stream source has one block that it needs to spread to the other peers comprising the P2P network.

Let $I(t)$ denote the set of peers that have received the block at the end of t time slots (hence these peers are "infected"). Initially, only the stream source possesses the block, hence $I(0) = 1$. The number of peers that have not received the block after t time slots is denoted by $U(t) = n - I(t)$ (this quantity is the number of "uninfected" peers after time t).

We adopt the model of the block spreading from [32]. Specifically,

$$E\big(U(t+1) \mid I(t)\big) = U(t)\big(1-n^{-1}\big)^{\delta f I(t)} \quad (2)$$

$$\approx U(t)e^{-\delta f I(t)/n}. \quad (3)$$

As in [32], we neglect the fluctuation of $U(t+1)$ around its conditional expectation to get

$$U(t+1) = U(t)e^{-\delta f I(t)/n}. \quad (4)$$

Further, we define

$$S_n = \min\{t : I(t) = n\}, \quad (5)$$

so S_n denotes the number of time slots until everyone receives the stream block with high probability. Further, we define $\delta_m f_m = \min(\delta_i f_i)$ for all i. We now state and prove an important Lemma for proving an upper bound in the number of time slots S_n in Theorem 2. For two identical P2P networks α and β, let $U_\alpha(t)$ and $U_\beta(t)$ be the number of uninfected peers after time t in networks α and β, respectively.

Lemma 1. *For randomized gossip in two identical P2P networks α and β, if $U_\alpha(t) \le U_\beta(t)$, then $U_\alpha(t+k) \le U_\beta(t+k)$ for $k \ge 0$.*

Proof. Since $U_\alpha(t) \le U_\beta(t)$, we have $I_\alpha(t) \ge I_\beta(t)$, because in identical networks α and β, $U(t) + I(t) = n$ for all t. Applying these inequalities in (3), we get $E(U_\alpha(t+1) \mid I_\alpha(t)) \le E(U_\beta(t+1) \mid I_\beta(t))$. Or equivalently from (4),

$$U_\alpha(t+1) \le U_\beta(t+1). \quad (6)$$

An induction argument then proves the statement of the lemma. □

An incentive model with optimistic unchoke can be thought of as overlaying the incentive-based (nonrandom) gossip protocol on top of an altruistic optimistic unchoke gossip protocol. Lemma 1 asserts that the altruistic optimistic unchoke mechanism will only speed up due to the increased number of infected peers from the overlaid incentive-based gossip.

Theorem 2. *In probability,*

$$S_n \le \log_{\delta_m f_m + 1} n + \big(\delta_m f_m\big)^{-1} \log n + O(1) \quad (7)$$

as $n \to \infty$.

Proof. We provide a sketch of the proof here. The proof follows on almost identical lines to a similar result for randomized gossip in [32], we reproduce an adapted version here for completeness. We show that if all peers only run the minimal altruistic gossip with fan-out $\delta_m f_m$, then the statement of Theorem 2 is satisfied in equality. Lemma 1 ascertains the direction of the inequality when additional infections occur under an incentive scheme.

Specifically, define

$$x(t+1) = f\big[x(t)\big], \quad t \ge 0,$$
$$f(x) = 1 - (1-x)e^{-\delta_m f_m x}, \quad (8)$$

where $x(t) = I(t)/n$ so that $x(0) = n^{-1}$. In terms of $x(\cdot)$,

$$S_n = \min\{t : x(t) > 1 - n^{-1}\}. \quad (9)$$

So, $f(0) = 0$, $f'(0) = 1 + \delta_m f_m$, and $f'(1) = e^{-\delta_m f_m}$.

Meaning that $x(t)$ grows exponentially in the beginning and increases slowly with a rate of $\approx e^{-\delta_m f_m}$ when most peers are already infected.

Let $\epsilon = \epsilon(n) \to 0$ and

$$\tau_1 = \tau_1(n) = \min\{t : x(t) > \epsilon\},$$
$$\tau_2 = \tau_2(n) = \min\{t > \tau_1 : x(t) > 1 - \epsilon\}. \quad (10)$$

Then,

$$\tau_1 = \log_{1+\delta_m f_m} \frac{\epsilon}{n^{-1}},$$
$$S_n - \tau_2 \approx \big(\delta_m f_m\big)^{-1} \log \frac{\epsilon}{n^{-1}} \quad (11)$$

since $f(x) > x$ for $0 < x < 1$,

$$\tau_2 - \tau_1 = o\big[\tau_1 + S_n - \tau_2\big]. \quad (12)$$

This means that the rumor spreads much more rapidly when the number of infected peers is neither too large nor too small. Combining (11) and (12) yields the proof of the theorem assuming $\epsilon \to 0$. □

The optimistic unchoke component ensures that the gossip protocol completes disseminating the block in *at least* $O(\log n)$ time and overcomes the clique tendency of the peer-wise proportional fairness incentive mechanism of (1) that would partition the network into peers that exchange blocks only among themselves and completely "ignore" peers who join the stream later and/or have a lower upload rate.

5. SIMULATIONS

We provide simulation experiments to verify three important properties of incentives in gossip-based streaming. Firstly, we demonstrate that the number of time slots before a stream block reaches all peers in a large P2P streaming network is logarithmic in the network size. Secondly, we show that a constant rate is achieved at the peers, meaning that they can play the stream in real time as it is being downloaded. Thirdly, our simulations demonstrate that the peer-wise proportional fairness of (1) rewards peers according to their contributions to other peers. In addition, we show the asymptotic fairness properties of peer-wise proportional fairness in Section 5.1.

We created a discrete time simulator that implemented the push-based gossip protocol on n peers in a completely connected network, that is, each peer could route blocks to any other peer. Peers ran the distributed peer-wise proportional fairness algorithm (1) to compute the (weighted and nonuniform) probability of assigning a part of their fan-out to all other peers in each time slot. We simulated two classes of peers: those which contribute bandwidth back to the network and those that do not contribute anything back to the P2P streaming network.

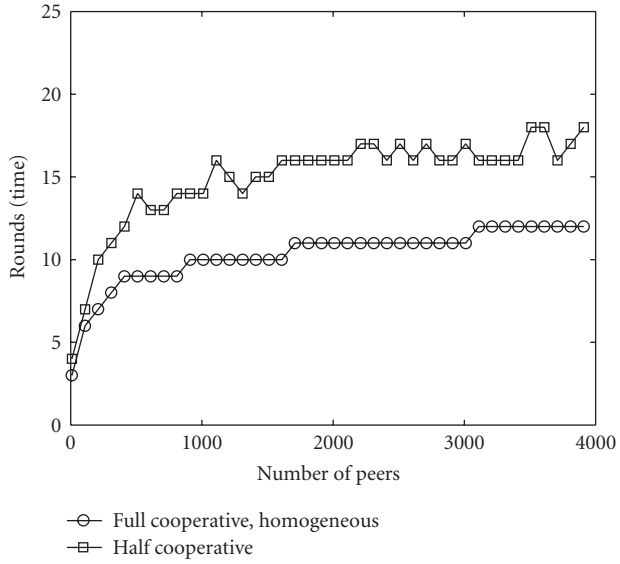

FIGURE 2: A stream block reaches all n peers in $O(\log n)$ time with the peer-wise proportional fairness incentive mechanism.

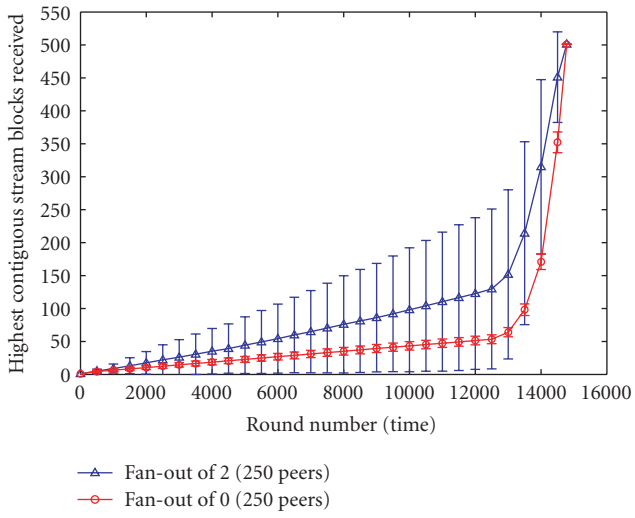

FIGURE 3: Optimistic unchoke fraction $\delta_i = 0.01$ for all i, 500 peers, 500 stream blocks. A constant streaming rate to peers is supported; and the peer-wise proportional fairness incentive algorithm differentiates peers according to their bandwidth contributions to the P2P network.

Figure 2 shows results for the number of time-slots (rounds) required to spread one block of a stream to all peers of the network with optimistic unchoke parameter $\delta_i = 0.01$ for all i. In the full-cooperative, homogeneous simulation of each peer had a fan-out of 3, meaning that each peer could support concurrent stream uploads to 3 other peers. In the "half-cooperative" mode, only half of the peers contributed to upload bandwidth. The plot confirms our analytical result for incentive-based gossip requiring $O(\log n)$ rounds.

Peers that contribute bandwidth back into the system were rewarded with higher-download rates, as Figure 3 indicates. In order to observe the streaming rate of a long stream,

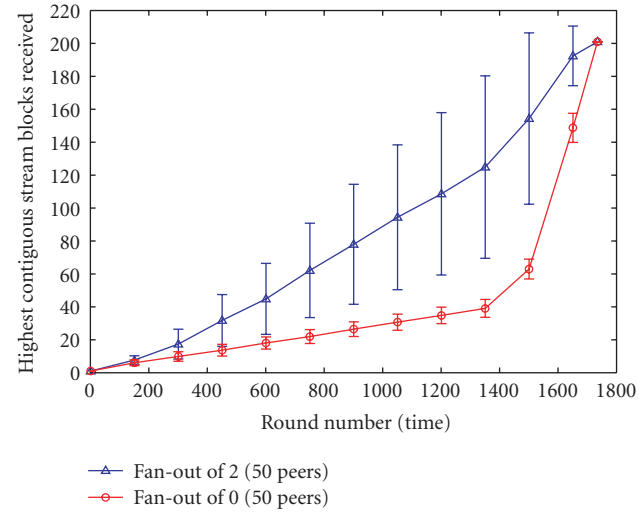

FIGURE 4: Optimistic unchoke fraction $\delta_i = 0.01$ for all i, 100 peers, 100 stream blocks.

our stream was divided into 500 stream blocks, each of which was gossiped in the P2P network. The knee in the graph indicates the time when most peers who were contributing more bandwidth (fan-out 2) to the system had completed the download and offered more bandwidth to all other peers. The distributed peer-wise proportional fairness algorithm correctly classifies peers according to how much they contribute back to the P2P network. Another important observation is that the graph is linear until the knee is reached, meaning that, on average, a constant download rate to peers is supported, which is of paramount importance for real-time playback of the video stream.

5.1. *Asymptotic properties of peer-wise proportional fairness*

The next few simulations shed light on the asymptotic properties of peer-wise proportional fairness. We simulated a P2P gossip streaming network with 100 peers, half of who contributed bandwidth (full cooperative, fan-out 2) while the other half did not contribute any bandwidth (noncooperative) to their peers.

As expected, peers who contributed bandwidth to the P2P system were rewarded with higher download rates, as Figure 4 indicates. But there was a large variance in the download rate amongst the full-cooperative (and to a lesser extent noncooperative) peers as indicated by the error bars of Figure 4. We therefore ran additional simulations in order to verify the asymptotic fairness of peer-wise proportional fairness across multiple stream downloads.

The bar graphs of Figures 5 and 6 show the average download time (rounds) taken by 50 full-cooperative and 50 noncooperative peers to receive the 100 stream blocks comprising a video stream. It is clear that, on average, all peers (full-cooperative or noncooperative) receive similar quality of service depending on how much they contribute back to

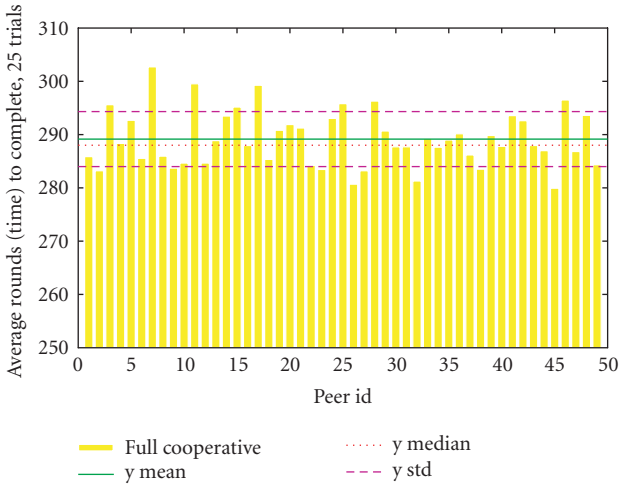

Figure 5: 50 full cooperative peers' download times (rounds), averaged over 25 streams.

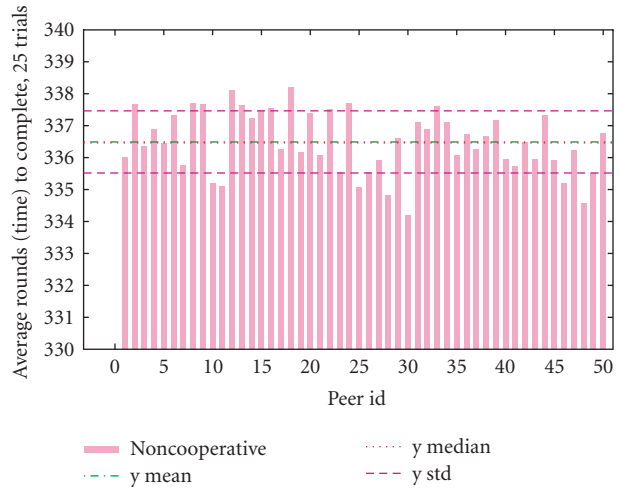

Figure 6: 50 full-noncooperative peers' download times (rounds), averaged over 25 streams.

the P2P system. Thus, the peer-wise proportional fairness scheme enables asymptotic fairness.

6. ARCHITECTURE AND PROTOCOL FORMAT

We next describe the architecture and functional components of the peers for gossip-based streaming. We also outline a protocol format that will be employed for implementation and evaluation in the next section.

6.1. Peer state space

Figure 7 shows the states of the peers as the incentive based P2P gossip streaming is executed by them in a distributed and asynchronous fashion.

(1) *Offline:* a peer in the offline state is not actively involved in downloading, uploading or otherwise participating in the media stream distribution.

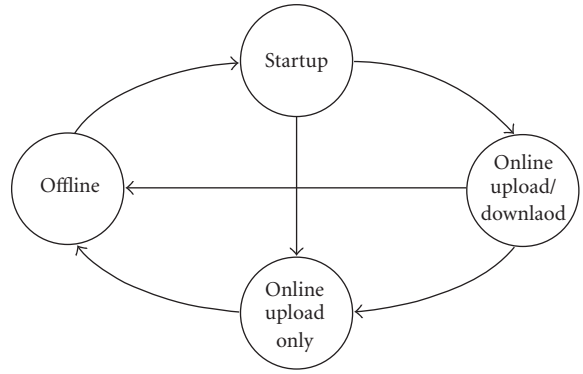

Figure 7: The state diagram of a peer.

(2) *Start-up:* a peer enters the start-up state when it joins the overlay P2P network. In case the peer is the source, it sets up the tracker that maintains a hostfile which contains the current list of all peers in the network. The nonsource peers on the other hand retrieve the hostfile from the tracker in the Start-up phase.
All peers fire-up a Listener component (Section 6.2) that listens to protocol messages. The streaming protocol parameters are also initialized during the Start-up phase. These include the gossip timeout, maximum block size, and upload bandwidth. The gossip timeout (ms) is the time between successive advertisement messages sent from a content uploading peer to a content downloading peer. Its value can be increased or decreased in order to slow or speed up the stream download rate, respectively, subject to network and content stream availability. The maximum block size (kB) is the size of the largest block transferred from one peer to another using the media streaming protocol.
The upload bandwidth is the upper limit on the amount of upload bandwidth dedicated to uploading stream blocks to requesting peers. The upper limit enforces a control over bandwidth that enables the users to run other applications without adversely affecting their online experience due to excessive utilization of the upload bandwidth of the network connections by the P2P streaming application.

(3) *Online upload/download:* following the completion of the startup phase, a peer enters the online upload/download state, during which the upload and download of data blocks happen. The source peer never enters this state as it does not download media content. A peer in online upload/download state can leave the overlay network and become offline.

(4) *Online upload only:* in this state a peer only uploads media content. This is the state acquired by the source after startup. All other peers will eventually complete downloading the stream (in case of a fixed-length media stream), thus ending up with a cached copy of the entire media stream file just like the source host. These peers may decide to altruistically act as sources and

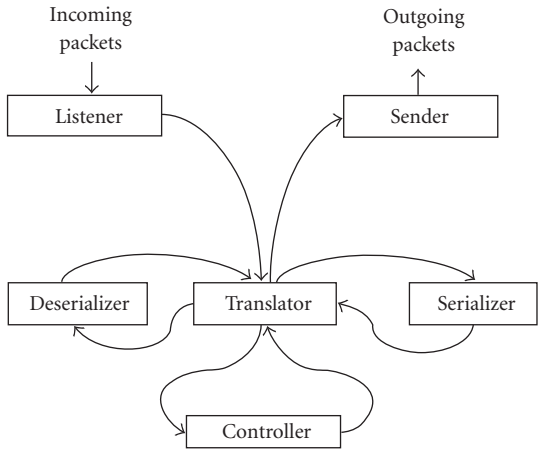

Figure 8: Functional components of a peer.

upload the stream to other requesting peers. The corresponding state of these peers is then online upload only.

The peers which act as content uploading peers or sources are called cooperative peers, whereas the peers which do not operate as content uploading peers are called noncooperative peers. The cooperative peers are able to achieve higher-download rates via the incentive-based mechanisms the implementation of which will be discussed in Section 6.4. A peer in online upload only state can leave the overlay network and enter the offline state.

6.2. Peer functional components

Figure 8 shows a schematic diagram of the functional components of a peer. The component listener listens on a specific port and receives transport layer packets from the media streaming application running on another peer: the payload of the packets being in a serialized suitable for transmission across the network, the listener passes the incoming packet's payload to a translator. The translator component deserializes the payload via the deserializer and stores it in a hash table which is then passed on to a controller. The translator is also responsible for converting a deserialized hash-table message understandable by the controller to a serialized message suitable for transmission in the payload of a packet. The serialization is performed via the serializer component. The serialized message is then transmitted by the component Sender to the listening port of another peer.

The controller comprises of a subcomponent called the Resource Broker that decides whether the media stream content should be sent to a soliciting peer. The Resource Broker makes the decision of sending based on the peer's available bandwidth and the incentive (credit) that the soliciting peer has earned by uploading to the peer.

6.3. Streaming operation

Figure 9 summarizes the operation of media streaming application for a nonsource peer. Push-based and pull-based gossip functionalities are depicted. A content downloading peer is said to be pulling a data block of the media data stream when it actively solicits the next contiguous block of the media data stream from a content uploading peer. The solicitation is performed via sending a request to send protocol message to a randomly chosen peer in every time slot. A content uploading peer is said to be pushing a data block when it actively advertises blocks of the media stream to content downloading peers via available messages. In general, the push-based or pull-based functionalities or a combination of the two may be employed for gossip-based media streaming.

Figure 10 depicts the operation of media streaming application for the source peer. The pull-based functionality is absent in this case since the source does not need to download content.

6.4. Realization of incentives

The choice of a peer for sending an available message (Figure 9) is made via an incentive-based mechanism. Each nonsource peer i maintains a vector U with elements designating the amount of data (in kB) received from each of the other peers. If the N peers are indexed, without loss of generality, as $0, 1, \ldots, N-1$, then the normalized vector

$$P = \frac{U}{\sum_{j=0}^{N-1} U[j]} \tag{13}$$

represents the probability mass function of a random number which represents the index of the peer to which available message is sent. Thus, the peer that contributed more earlier has a better chance of becoming a recipient of content.

Again, the Resource Broker in Figure 9 sends content to a peer according to the incentive-based approach. The size of a block sent to a peer k is ascertained as $P[k]B_{\max}$, where $B_{\max}$ is the maximum block size, as elaborated in Section 8.3.

7. IMPLEMENTATION AND EVALUATION

We implement the gossip streaming protocol as described in Section 6 and evaluate the streaming performance in a real test bed consisting of a small grid of computers networked via gigabit connectivity. We use a traffic shaping to limit the upload bandwidth assigned to the gossip streaming application on each peer. The scenario consists of 5 peers one of which acted as the content source. The source serves cached (prerecorded) audio or video files, although our implementation also supports live video capture via the java multimedia framework [33]. Peers use the implemented gossip streaming protocol in order to share and distribute blocks of the stream, with the objective of playing back the streamed file in real time (playback while the stream downloads). For the experiments, a 5.4 MB MPEG-2 video file is streamed from source peer. Figure 11 depicts the variation of time to download the file with the maximum block size. The gossip thread time-out is kept constant at 100 milliseconds and the upload bandwidth is fixed at 1200 kbps. As expected, the time to download the file decreases as the maximum block size was increased because the bigger block sizes resulted in

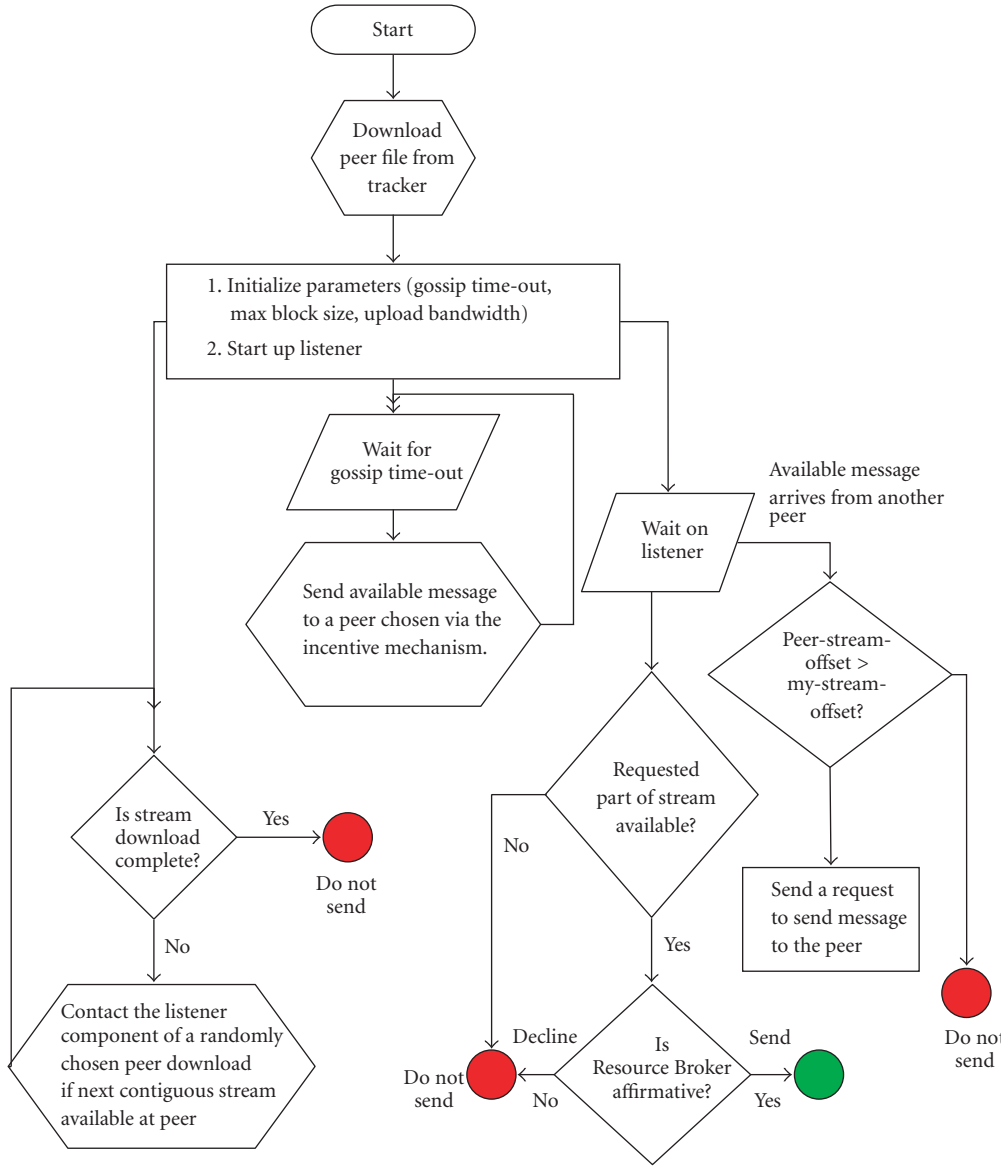

Figure 9: Streaming Operation of nonsource peer.

the same 5.4 MB being downloaded with smaller number of (bigger) blocks, thus requiring fewer gossip transactions. Figure 12 depicts the time to download the file as a function of the gossip thread time-out. The bandwidth is kept constant at 1200 kbps and the block size is fixed at 25 KB. It can be seen that the time to download the stream on a recipient peer grows linearly with the gossip thread time out because the number of requests (random probes) sent out decreased (per time), slowing down the gossip protocol and increasing the download completion time. Figure 13 depicts the variation of the time to download the file with the upload bandwidth. As expected, the download time decreases with increasing amounts of bandwidth available to serve peers on the P2P system. We measure the control overhead of our implemented protocol in Figure 14. The number of control data messages sent over the network remain fairly constant irrespective of the gossip thread time. A smaller gossip thread time-out only leads to a quicker completion of the stream download (as Figure 12), and the control overhead remains constant for a fixed stream download size (5.4 MB). This is an important scalability strength of our incentive-based gossip streaming protocol.

8. DISCUSSION

We briefly describe three practical improvements while implementing the incentive mechanism in a real peer-to-peer streaming system. First, we propose the use of a more realistic utility function rather than simply using the bandwidth. Second, propose a decay component in the utility function in order to make the implemented system adapt to changes in the peer-to-peer network fast. Third, we suggest varying

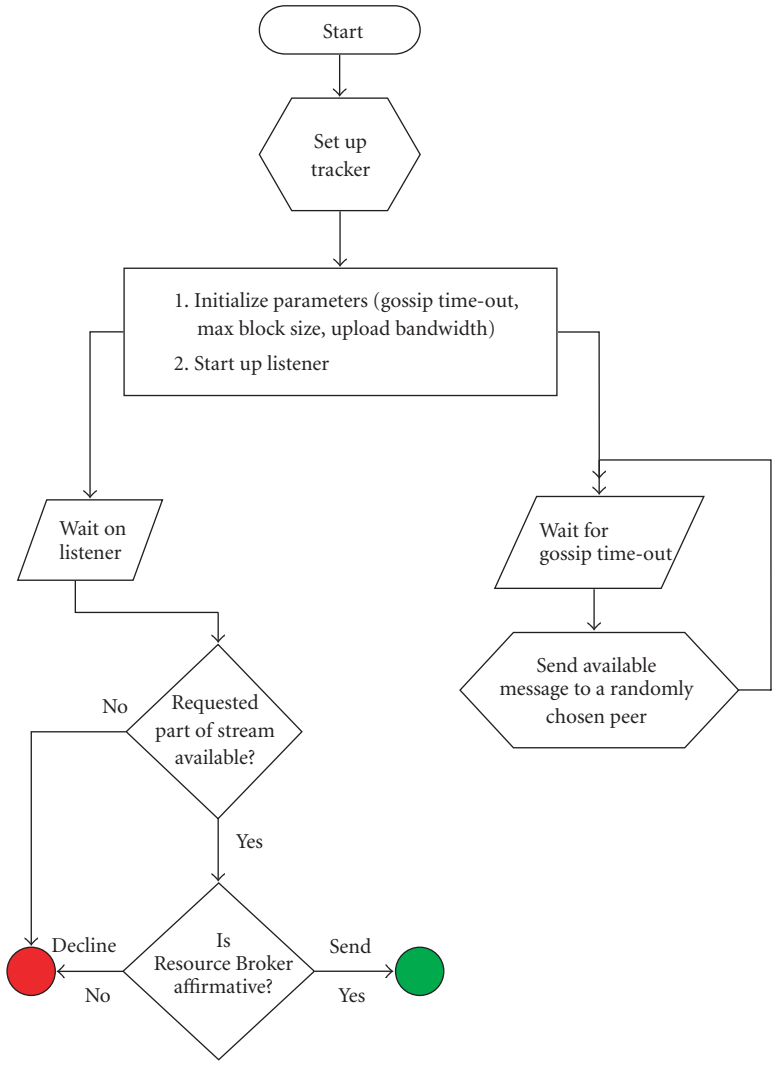

Figure 10: Streaming operation of the source.

block sizes as a supplementary measure to reward peers that contribute more bandwidth to the P2P system.

8.1. Realistic utility functions

In Sections 5, and 7 upload bandwidth was the primary barter quantity and a peer's utility to other peers was measured in terms of the upload bandwidth provided by the former to the latter. For example, the routing hop-distance (an indicator of delay and core network congestion), peer uptime (an indicator of peer reliability) can also be used by a peer while computing the utility of another peer. The modified utility function u'_{ij} in (1) can then be defined as

$$u'_{ij}(t) = \frac{u_{ij}(t) \cdot t_j}{h_{ji}}, \tag{14}$$

where u_{ij} is computed from (1), t_j is the time since peer i first communicated with peer j in the current session, and h_{ij} is the IP hop count as reported by the trace route IP routing tool.

Again, modifying the overlay characteristics by taking into account the underlying IP network and the up-time of peers breaks the "randomly connected" assumption of classical gossip analysis. By employing the optimistic unchoke argument of Section 4 we can still guarantee an $O(\log n)$ delivery time.

8.2. Exponential decay of utility

The expression for peer-wise proportional fairness provided in (1) may result in the distributed algorithm adapting slowly to peer churn. To see why this is case, consider peer i at time t that has received utility U_{ji} from peer i and U_{-j} from all other peers except peer j until time $t-1$. Then, the utility that $u_{ij}(t)$ assigns to peer j at time t is proportional to

$$u_{ij}(t) \propto \frac{U_{ji}}{U_{ji} + U_{-j}}. \tag{15}$$

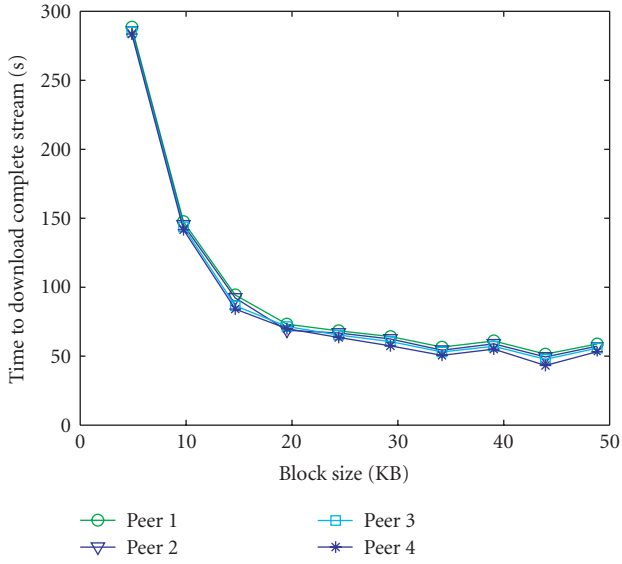

Figure 11: Time to download versus the block size.

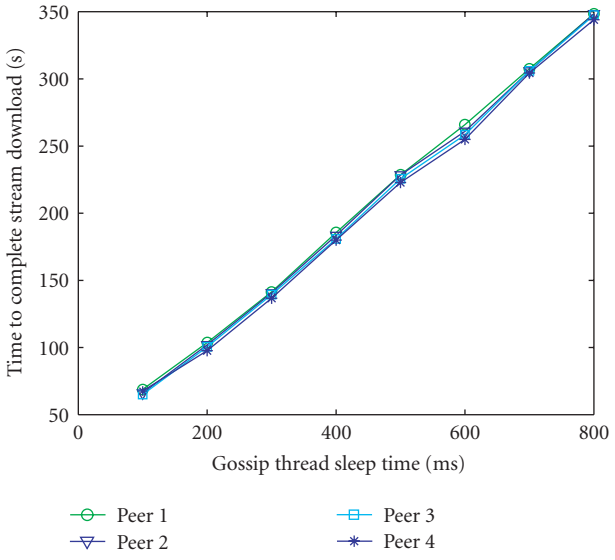

Figure 12: Time to download versus gossip thread time-out.

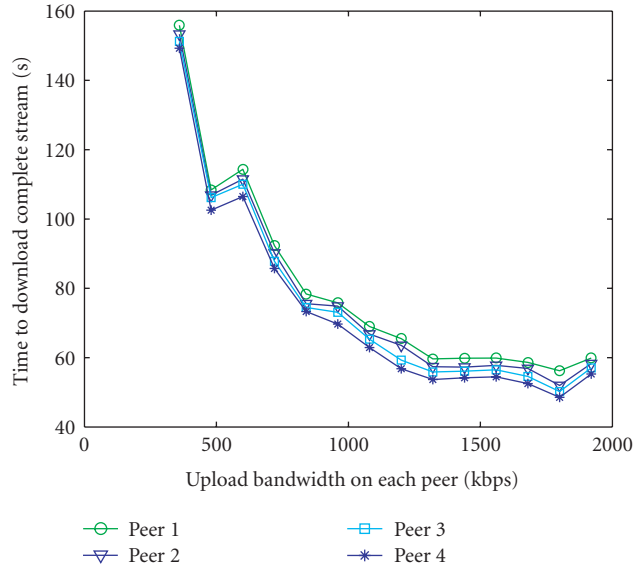

Figure 13: Time to download the complete stream (5.4 MB file) as a function of the upload bandwidth offered by each peer.

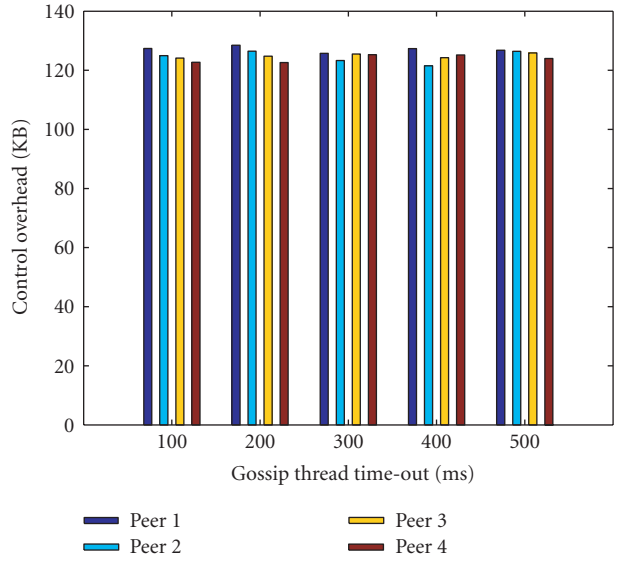

Figure 14: Control overhead remains constant across different gossip thread time-out.

Taking the derivative of $u_{ij}(t)$ with respect to U_{ji},

$$u'_{ij}(t) = \frac{U_{-j}}{(U_{ji} + U_{-j})^2}. \tag{16}$$

Since received utility is always nonnegative, the derivative $u'_{ij}(t)$ is nonnegative and upper-bounded by $1/U_{-j}$. This property makes $u_{ij}(t)$ adapt very slowly to changing peer characteristics after U_{-j} becomes sizeable (note that usually $U_{ji} \ll U_{-j}$). Thus while peer-wise-proportional fairness is asymptotically fair, it has slow dynamics in real systems.

In a real implementation, a decay mechanism to speed up the dynamics of the proposed method is incorporated in order to counter this problem. In particular, the utility values are decayed by a decay factor ρ ($0 \le \rho \le 1$) in each time slot (the time slot being an implementation-dependent parameter). Thus,

$$U(t+1) = \rho \cdot U(t). \tag{17}$$

This decay causes "historical" download information to become less relevant while recent download information becomes more relevant in the computation of utility for the incentive mechanism.

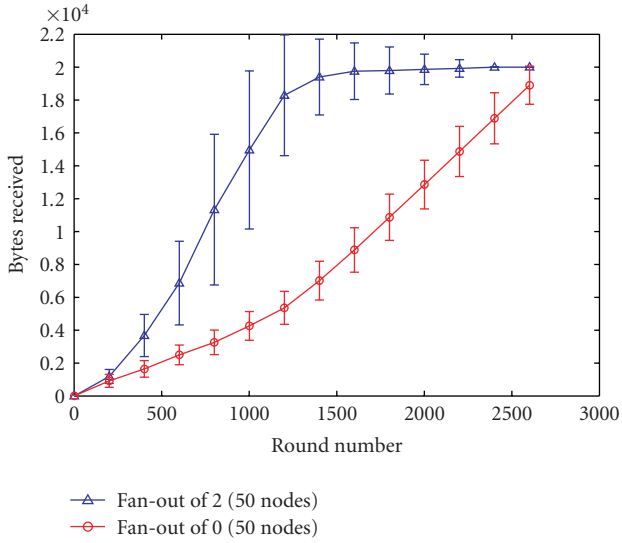

FIGURE 15: Simulation when stream block sizes are varied according to the incentive mechanism of (1).

8.3. *Stream block sizes as incentives*

A peer sending a stream block to a recipient peer can modulate the block size according to the latter's perceived utility as calculated on the sending peer (e.g., according to (1). Intuitively, a recipient peer that receives bigger blocks of the stream will download at a higher rate because fewer gossip transactions will be needed to download the whole stream.

We simulate these conditions in Figure 15. There are 100 peers, half of which contribute bandwidth while the other half do not contribute any bandwidth to the system. Peers push or upload stream blocks in sizes that are proportional to the utility of the recipient node calculated on the sending peer. Comparing Figure 15 to Figure 4, we observe that contributing peers achieve much higher rates as compared to their noncontributing counterparts.

9. CONCLUSIONS AND FUTURE WORK

Creating incentives for *live* video streaming is different than stored content incentives because the utility of the downloaded content is transient, with older content quickly becoming uninteresting to other peers, and hence unsuitable for barter. Conventional currency units (say dollars per downloaded MB) can be used as an instrument of barter, but the approach has the disadvantage of needing a centralized verification and transaction mechanism.

We have proved that by using an optimistic unchoke mechanism in a nonrandom gossip protocol we can achieve the important $O(\log n)$ time for dissemination of a stream block to all peers. Based on this result, an incentive mechanism to encourage peers to contribute resources such as bandwidth can be created while still preserving the logarithmic completion time property of gossip protocols.

We employed the distributed peer-wise proportional fairness algorithm for creating incentives for live video streaming. This algorithm achieves fairness in an asymptotic regime, as proved in [30, 31] and shown through simulations in Section 5.1. Short term fairness is not guaranteed however, as indicated from the error bars in Figure 3. Research in other fairness and incentive algorithms for P2P streaming remains an important part of future work.

Gossip protocols incur an extra factor of $\log n$ overhead as compared to tree-based P2P networks because the number of blocks transmitted in order to spread one block from the source to all other peers is $O(n \log n)$ in the Gossip-based approach as compared to $O(n)$ in the tree-based approach. The trade-off is between higher robustness (gossip protocols) and lower message overhead (tree-based protocols). Recent results [34] combine push- and pull-based gossip protocols to reduce the additional overhead factor to $O(n \log \log n)$. Our implementation uses such a combination of push- and pull-based gossip.

We have also delineated the functional components and protocol format that can be employed for the implementation of gossip-based streaming. Preliminary results from the implementation confirm the ability of achieving a constant throughput at the peers, a prime objective of any live streaming scheme. Experiments are underway to extend the results over a much larger test bed.

ACKNOWLEDGMENTS

The first author thanks Ari Trachtenberg, Moshe Laifenfeld, and Murat Alanyali for preliminary discussions on peer-wise proportional fairness. The first author also thanks Shruti Arun Gupta for editorial inputs. The authors sincerely thank the reviewers whose valuable suggestions helped in improving this work.

REFERENCES

[1] B. Cohen, "Bittorrent," http://bitconjurer.org/BitTorrent.
[2] J. Glasner, "P2P fuels global bandwidth binge," *Wired News*, April 2005.
[3] R. T. B. Ma, S. C. M. Lee, J. C. S. Lui, and D. K. Y. Yau, "A game theoretic approach to provide incentive and service differentiation in P2P networks," *ACM SIGMETRICS Performance Evaluation Review*, vol. 32, no. 1, pp. 189–198, 2004.
[4] Z. Ge, D. R. Figueiredo, S. Jaiswal, J. Kurose, and D. Towsley, "Modeling peer-peer file sharing systems," in *Proceedings of the 22nd IEEE Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '03)*, vol. 3, pp. 2188–2198, San Francisco, Calif, USA, March-April 2003.
[5] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A measurement study of peer-to-peer file sharing systems," in *Multimedia Computing and Networking*, vol. 4673 of *Proceedings of SPIE*, pp. 156–170, San Jose, Calif, USA, January 2002.
[6] D. Qiu and R. Srikant, "Modeling and performance analysis of BitTorrent-like peer-to-peer networks," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (ACM SIGCOMM '04)*, pp. 367–378, Portland, Ore, USA, August-September 2004.
[7] T. Ng, Y. Chu, S. Rao, K. Sripanidkulchai, and H. Zhang, "Measurement-based optimization techniques for bandwidth-demanding peer-to-peer systems," in *Proceedings of the 22nd*

Annual Joint Conference on the IEEE Computer and Communications Societies (INFOCOM '03), vol. 3, pp. 2199–2209, San Francisco, Calif, USA, March-April 2003.

[8] Carnegie Mellon University, "End system multicast," http://esm.cs.cmu.edu/.

[9] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," in *Proceedings of the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '02)*, pp. 177–186, Miami, Fla, USA, May 2002.

[10] Y. Chawathe, "Scattercast: an adaptable broadcast distribution framework," *Multimedia Systems*, vol. 9, no. 1, pp. 104–118, 2003.

[11] J. Silber, S. Sahu, J. P. Singh, and Z. Liu, "Augmenting overlay trees for failure resiliency," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '04)*, vol. 3, pp. 1525–1531, Dallas, Tex, USA, November-December 2004.

[12] C. L. Abad, W. Yurcik, and R. H. Campbell, "A survey and comparison of end-system overlay multicast solutions suitable for network-centric warfare," in *Battlespace Digitization and Network-Centric Systems IV*, vol. 5441 of *Proceedings of SPIE*, pp. 215–226, Orlando, Fla, USA, April 2004.

[13] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *IEEE Communications Surveys & Tutorials*, vol. 7, no. 2, pp. 72–93, 2005.

[14] N. Magharei, R. Rejaie, and Y. Guo, "Mesh or multiple-tree: a comparative study of live P2P streaming approaches," in *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM '07)*, pp. 1424–1432, Anchorage, Alaska, USA, May 2007.

[15] T. Silverston and O. Fourmaux, "P2P IPTV measurement: a comparison study," preprint, 2006.

[16] A. Ali, A. Mathur, and H. Zhang, "Measurement of commercial peer-to-peer live video streaming," in *The 1st International Workshop on Recent Advances in Peer-to-Peer Streaming in Conjunction with the 3rd International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks*, Waterloo, Ontario, Canada, August 2006.

[17] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "Insights into PPLive: a measurement study of a large-scale P2P IPTV system," in *Proceedings of the IPTV Workshop in Conjunction with the International World Wide Web Conference*, Edinburgh, UK, May 2006.

[18] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "A measurement study of a large-scale P2P IPTV system," Tech. Rep., Department of Computer and Information Science, Polytechnic University, New York, NY, USA, 2006.

[19] X. Zhang, J. Liu, and B. Li, "On large-scale peer-to-peer live video distribution: coolstreaming and its preliminary experimental results," in *Proceedings of the IEEE International Workshop on Multimedia Signal Processing (MMSP '05)*, Shanghai, China, October-November 2005.

[20] M. Zhang, L. Zhao, Y. Tang, J.-G. Luo, and S.-Q. Yang, "Large-scale live media streaming over peer-to-peer networks through global internet," in *Proceedings of the ACM Workshop on Advances in Peer-to-Peer Multimedia Streaming (P2PMMS '05)*, pp. 21–28, Singapore, November 2005.

[21] SOPCast, "SOPCast," http://www.sopcast.org/.

[22] G. A. Fowler and S. McBride, "Newest export from China: pirated pay TV," *Wall Street Journal*, 2005.

[23] PPLive, "PPLive," http://www.pplive.com/.

[24] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming," in *Proceedings 24th IEEE Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '05)*, vol. 3, pp. 2102–2111, Miami, Fla, USA, March 2005.

[25] Coolstreaming, "Coolstreaming," http://www.coolstreaming.us/.

[26] Gridmedia, "Gridmedia," http://www.gridmedia.com.cn/.

[27] A. Demers, D. Greene, C. Hauser, et al., "Epidemic algorithms for replicated database maintenance," in *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing*, pp. 1–12, Vancouver, British Columbia, Canada, August 1987.

[28] K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky, "Bimodal multicast," *ACM Transactions on Computer Systems*, vol. 17, no. 2, pp. 41–88, 1999.

[29] S. Verma and W. T. Ooi, "Controlling gossip protocol infection pattern using adaptive fanout," in *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS '05)*, pp. 665–674, Columbus, Ohio, USA, June 2005.

[30] X. Yang and G. de Veciana, "Service capacity of peer-to-peer networks," in *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '04)*, vol. 4, pp. 2242–2252, Hong Kong, March 2004.

[31] S. Agarwal, M. Laifenfeld, A. Trachtenberg, and M. Alanyali, "Fast data access over asymmetric channels using fair and secure bandwidth sharing," in *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS '06)*, p. 58, Lisboa, Portugal, July 2006.

[32] B. Pittel, "On spreading a rumor," *SIAM Journal on Applied Mathematics*, vol. 47, no. 1, pp. 213–223, 1987.

[33] Sun-Developer-Network, "Java multimedia framework," http://java.sun.com/products/java-media/jmf/.

[34] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking, "Randomized rumor spreading," in *Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS '00)*, pp. 565–574, Redondo Beach, Calif, USA, November 2000.