Computer Communications 35 (2012) 1893–1901

Contents lists available at SciVerse ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Understanding the impact of neighboring strategy in peer-to-peer multimedia streaming applications

Xiangyang Zhang *, Hossam Hassanein

School of Computing, Queen's University, Kingston, Canada K7L 3N6

ARTICLE INFO

Article history:
Available online 30 May 2012

Keywords:
Multimedia streaming
Peer-to-peer networks
Network traffic

ABSTRACT

Peer-to-peer (P2P) multimedia streaming applications need to reduce network traffic to address ISPs' concerns without sacrificing the quality of users' viewing experience. Several studies on P2P file sharing applications propose that a peer only neighbors with nearby peers to reduce network traffic, but whether this strategy is applicable to P2P multimedia streaming applications remains an open issue. In this paper, we study packet propagation behavior and the impact of neighboring strategies on system performance in P2P multimedia streaming applications. We identify two "typical" schemes that capture the essential aspects of the swarm-based and tree-based P2P multimedia streaming schemes, respectively, and compare their performance on two types of neighboring overlays: a random overlay where a peer selects neighbors without considering their network locations, and a nearby overlay where a peer only neighbors with nearby peers. We first conduct simulation study and then provide models to analyze packet propagation behavior on a given overlay in the two typical schemes and the impact of the neighbor-with-nearby-peers strategy on system performance. We find that in the swarm-based scheme, packets propagate along short paths (in terms of hops) on the neighboring overlay, while in the tree-based scheme, peers select parents randomly with respect to their hop counts to the source peer. Applying the neighbor-with-nearby-peers strategy reduces network traffic but results in more lost packets because the nearby overlay has a larger diameter and clustering coefficient. This problem is more severe in the tree-based scheme than in the swarm-based scheme due to their different packet propagation behavior.[1]

1. Introduction

Peer-to-peer (P2P) multimedia streaming applications such as Internet television, live event broadcast, and remote education have gained great popularity during the past few years. While users are enjoying these applications (mostly for free), their enormous traffic causes Internet service providers (ISPs) great financial expenditure and threatens the quality of other Internet services. To avoid consequent unpleasant traffic throttling or blocking, P2P multimedia streaming applications need to address ISPs' concerns, i.e., to reduce inter- and intra-autonomous system (AS) traffic on the Internet.

In a P2P network, in order to reduce communication and processing overhead, a peer only maintains relationship with a limited number of peers, which we call the *neighbors* of the peer.[2] All the neighboring relationships form an overlay network, which we call the *neighboring overlay*. A peer exchanges packets only with its neighbors. The strategy by which a peer selects peers to neighbor with and the strategy by which a peer selects neighbors to exchange packets determine the network traffic. Several measurement studies, such as [2], on real-world P2P multimedia streaming deployments show that a peer does not consider other peers' network locations in the two strategies, and hence enormous traffic ensues.

Several studies [3–5] on P2P file sharing applications propose that a peer only neighbors with nearby peers in the construction of the neighboring overlay to reduce network traffic. However, compared with file sharing applications, live streaming applications have rigid delay constraints. Packets that arrive after playback deadlines are considered lost, and peers will not attempt to fetch packets that have missed or are about to miss playback deadlines. Whether the neighbor-with-nearby-peers strategy is applicable in the context of P2P multimedia streaming remains an open question in the literature.

In this paper, we study packet propagation behavior and the impact of neighboring strategies on system performance in P2P multimedia streaming applications. We attempt to answer the following questions. (1) Can we use the neighbor-with-nearby-peers

* Corresponding author. Tel.: +1 6135336052.
E-mail addresses: xiang@cs.queensu.ca (X. Zhang), hossam@cs.queensu.ca (H. Hassanein).

[1] An abbreviated version of this paper that includes the typical swarm-based and tree-based schemes and part of the simulation results appeared in [1].

[2] In some schemes, a peer maintains a membership table and selects some members to construct a neighbor table. In other schemes, a peer constructs the neighbor table directly.

0140-3664/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.comcom.2012.05.012

strategy in P2P multimedia streaming applications to reduce network traffic without causing unfavorable side-effects? (2) Given a neighboring overlay, will packets propagate along low-cost paths?

Since there exist a large number of P2P multimedia streaming schemes, we first identify "typical" schemes that capture the essential aspects of these schemes, especially those in actual deployment on the Internet. P2P multimedia streaming schemes can be classified into two categories: swarm-based schemes and tree-based schemes [6]. Swarm-based schemes employ the *swarm* technique used in BitTorrent [7]. A stream is split into chunks and peers pull missing chunks from neighbors. Our typical swarm-based scheme (called TS hereafter) employs the rarest-first chunk scheduling policy, which is used in most swarm-based schemes. In tree-based schemes, peers form parent–child relationship to build a tree along which a parent pushes packets to children. Our typical tree-based scheme (called TT hereafter) constructs the tree in a straight-forward manner—each peer subscribes to the neighbor with the latest position in the stream. We then compare system performance of the two typical schemes on two types of neighboring overlays: *random overlay* and *nearby overlay*. When constructing a random overlay, peers select neighbors without considering their network locations. When constructing a nearby overlay, peers use the neighbor-with-nearby-peers strategy and only select nearby peers as neighbors.

We study the impact of the neighbor-with-nearby-peers strategy by both simulation and analysis. We first develop a discrete-event simulator to study system performance of the two typical schemes on the two overlays. Based on the findings in the simulation, we then provide packet propagation models to analyze packet propagation paths on a given overlay and overlay models to characterize the random and nearby overlays and analyze their impact on system performance. We find that packets propagate in distinct manners in TS and TT although both schemes are *data-driven* (i.e., the propagation paths are determined by availability of packets at peers rather than network metrics). In TS, peers have a high probability to pull from neighbors that have fewer hops to the source peer; the set of paths a chunk traverses from the source to peers (called *propagation tree*) is comparable to a degree-bounded shortest path tree (in term of hops) on the neighboring overlay. In TT, peers select parents almost randomly with respect to their hop counts to the source, resulting in significantly taller propagation trees compared with TS. The neighbor-with-nearby-peers strategy reduces network traffic significantly, but also results in more lost packets in the presence of peer churn and substrate network errors because of the large diameter and clustering coefficient of the nearby overlay. This problem is more severe in TT than in TS because they have different packet propagation behaviors, which cause chunk to be lost in different ways.

The remainder of this paper is organized as follows. Section 2 introduces background and related work. Section 3 describes TS and TT. Section 4 studies TS and TT on the random overlay and nearby overlay by simulation. Section 5 presents packet propagation models in TS and TT and analyzes the impact of propagation behavior on system performance. Section 6 presents overlay models and analyzes the impact of the neighbor-with-nearby-peers strategy on system performance. Section 7 concludes this paper.

2. Background and related work

There are two basic approaches to P2P multimedia streaming. The first approach is inspired by the *swarm* technique used in the BitTorrent file sharing application [7]. A stream is split into chunks of a fixed size. A peer maintains a sliding window of recently received chunks. A peer advertises its buffer map, which describes the chunks the peer has, to neighbors, and exchanges chunks with neighbors. The second approach is inspired by IP multicast. A peer subscribes to a neighbor to form explicit parent–child relationship. All the parent–child relationships form a tree on the neighboring overlay. (A stream may be interleaved into multiple substreams and peers build a separate tree for each substream.) Upon receiving a packet, a peer immediately forwards the packet to its children. A P2P multimedia streaming scheme may use either approach or combine them together. Most deployments on the Internet use the first approach or both approaches.

In swarm-based schemes [8–12], every interval of length T, a peer selects the chunks to pull in the next interval and selects the neighbors to pull these chunks from. A peer may select chunks randomly [8], but in most schemes, a peer employs a *latest-first* or *rarest-first* policy [9–11]. (In multimedia streaming applications, latest chunks are also rarest chunks.) A peer may select neighbors in several ways to pull the selected chunks, such as randomly [8,10], according to neighbors' workload [9,11], according to the bandwidths and delays to neighbors, and according to data exchange history. A peer may implement an incentive mechanism similar to the *tit-for-tat* policy of BitTorrent to encourage peers to contribute upload bandwidth [11]. However, since multimedia streaming applications have stringent time constraints and a peer only buffers recently received chunks (compared with the whole file in BitTorrent), the efficacy of this policy is limited. In swarm-based schemes, propagation trees are determined by the availability of chunks at peers rather than by network metrics such as cost, delay, and bandwidth, hence swarm-based schemes are said to be "data-driven" [9].

In tree-based schemes, a peer may consider multiple factors when selecting neighbors to subscribe to. According to whether the main factors under consideration are network metrics or neighbors' buffer maps and data exchange history, we can classify tree-based schemes into two types: network-driven, such as [13–16], and data-driven, such as [17–20]. Early schemes [13,14] are usually network-driven with the objective of minimizing the tree cost. Recent schemes include both types, but data-driven schemes are more prominent in real-world deployments [17,18]. Data-driven schemes combine the swarm and tree-building techniques together. In data-driven schemes, a peer advertises its buffer map to neighbors, establishes parent-child relationships with neighbors, and pulls missing chunks from neighbors. Because packets (or chunks) are both pushed and pulled, data-driven tree-based schemes are also called push-pull hybrid schemes [17,18]. The main factors a peer considers when selecting parents are neighbors' proceeding positions in the stream and the data exchange history. For example, in [18], a peer tries to maintain that it advances at similar pace in each substream and its parent advances at similar pace as its neighbors. In [17], a peer subscribes to a neighbor that it has received more packets in the last subscription interval with a higher probability. In [20], a peer subscribes to neighbors with the latest positions in the stream and short packet delivery delays.

In most swarm-based schemes and data-driven tree-based schemes, like in P2P file sharing schemes, peers' network locations are not considered in the construction of the neighboring overlay. Several ISP-friendly schemes for P2P file sharing applications, such as [3–5], propose that a peer only neighbors with nearby peers to reduce inter-AS traffic. References [4,5] report reduced inter-AS hops and shorter file downloading time. We remark that a study of using the neighbor-with-nearby-strategy in P2P multimedia streaming applications does not exist in the literature.

We also remark that there are only a few analytical studies on P2P multimedia streaming applications and none of them studies the neighboring strategy or packet propagation behavior. Zhou et al. [21] compare the chunk delivery rate and delivery delay of the rarest-first and nearest-deadline-first chunk scheduling policies in swarm-based schemes. Picconi and Massoulié [22] study the maximum streaming rate achievable in swarm-based schemes.

3. Typical swarm-based and tree-based schemes

In this section, we identify a "typical" swarm-based (TS) scheme and a "typical" tree-based (TT) scheme to investigate the impact of the neighbor-with-nearby-peers strategy on system performance. TS and TT capture the essential aspects of swarm-based schemes and tree-based schemes, especially those that have been deployed on a large scale over the Internet. In TS, a peer employs the rarest-first chunk selection policy, which is in line with most swarm-based schemes. We use a pull target selection algorithm similar to that in [9,11], which we found achieves smoother playback than the algorithm in [10]. A peer grants pull requests if it has spare upload capacity, i.e., it does not employ the tit-for-tat incentive mechanism. This is in line with the observations on PPLive and Sopcast, two popular P2P-based Internet television services, in [2]. Because of the prominence of data-driven tree-based schemes in actual deployments [17,18], we devise TT to be data-driven. TT constructs the tree in a straight-forward manner—each peer subscribes to the neighbor with the latest position in the stream. We use this policy for two reasons. First, peers' positions in the stream are a major factor a peer considers when selecting parents. Even in schemes where a peer selects parents according to neighbors' exchange history, peers' positions in the stream indirectly play an important role. Second, this policy contrasts well with TS. When TS employs the rarest-first policy, a peer tends to pull chunks from neighbors that have the latest positions in the stream.

3.1. Overlay construction

The bootstrap and overlay construction in TS and TT are similar to other P2P multimedia streaming schemes. The system contains a video server (source peer), a tracker, and a number of peers. A new peer obtains the IP addresses of the source peer and the tracker using an out-of-band mechanism, such as by browsing a web page. Each peer maintains a membership table, which consists of peers in the system. A new peer obtains its initial membership table from the tracker and may contact the tracker for more peers if the table size drops to a certain level. A peer randomly selects peers from the membership table to neighbor with. The use of membership tables reduces the tracker's workload and communication overhead; otherwise peers have to contact the tracker every time they need a new neighbor.

When constructing a random overlay, the tracker randomly selects peers from the whole population to compose a membership table for the requesting peer. When constructing a nearby overlay, the tracker randomly selects peers that are close to the requesting peer. The particular scheme for locating nearby peers is irrelevant to our analysis. We simply assume that the tracker has global knowledge of the system. The number of neighbors a peer maintains is proportional to its upload capacity, with a minimum of 4. The minimum of 4 neighbors is a necessary measure to prevent system partitioning.

3.2. Typical swarm-based (TS) scheme

The stream is split into chunks of fixed size. Each chunk has a unique sequence number. Each peer maintains a buffer of size W_{buf} to hold recently received chunks. Every interval of length T_{bm}, each peer advertises its buffer map to neighbors. The message consists of the latest chunk's sequence number n and a vector, whose ith element indicates whether the peer has chunk $n-i$.

Table 1
System parameters.

Notation	Comment	Default
T_{bm}	Buffer map advertisement interval (Sec.)	1
T_{subs}	Subscription check interval (Sec.)	1
T_{pull}	Pull check interval (Sec.)	1
T_{late}	Margin to decide a chunk is late (Sec.)	1
T_{sm}	Margin to switch parents (Sec.)	1
W_{buf}	Buffer size (Chunks)	600
W_{urg}	Emergency window size (TT/TS)	9/5
W_{swa}	Swarming window size (TT/TS)	0/240
W_{bbp}	Number of chunks to buffer before playing	20
W_{bbs}	Number of chunks to buffer before skipping	10

The media player reads chunks in the buffer and plays them back. If the player is about to play a chunk but the chunk is unavailable, the player blocks. It resumes playing the chunk when the chunk arrives or skips to the next available chunk if the peer has W_{bbs} consecutive chunks after the chunk. Notations are shown in Table 1.

As in other swarm-based schemes, the buffer is sequentially divided into an emergency window, which consists of chunks approaching playback deadlines, and a swarming window. Every interval of length T_{pull}, a peer selects all the chunks in the emergency window and the rarest chunks in the swarming window to pull from neighbors. The pull target selection algorithm is similar to that in [9,11], which has a high pull success rate. Assume a peer wants to pull m chunks from n neighbors, and each neighbor has certain chunks. The algorithm starts with chunks that are available from only 1 peer, then chunks available from 2 peers, and so on. In each step, urgent chunks are considered before rare chunks, and a random neighbor is selected if more than one qualified neighbor exists. A new peer first swarms until $\frac{3}{4}$ of its swarming window is filled, then it starts playing after receiving W_{bbp} consecutive chunks.

3.3. Typical tree-based (TT) scheme

TT makes the following modifications to TS. Every interval of length T_{subs}, a peer checks whether to switch parents. Neighbors are ranked according to their positions in the stream. The neighbor with the latest position is the candidate parent. A peer switches parents only when the candidate parent has a margin of T_{sm} more than the current parent. A child must subscribe to its parent periodically to keep its status as a child. Peers employ connection admission control (CAC) and preemption when handling subscription requests. When a peer x receives a request from peer y, peer x grants the request if it has spare upload capacity. Otherwise, peer x compares the upload capacity of peer y with the child z that has the minimum upload capacity, and replaces child z with peer y if child z's capacity is smaller.

A peer pushes a chunk to its children immediately after receiving it and guarantees the capacity allocated to its children. Peers only pull urgent chunks and chunks that are T_{late} seconds later than their due time. A peer uses the same pull target selection algorithm as in TS. Note that a peer grants pull requests only if it has spare upload capacity after serving children. A new peer starts playing after having W_{bbp} consecutive chunks.

4. Simulation study

In this section we study system performance of TS and TT on the random and nearby overlay by simulation.

4.1. Performance metrics

We focus on three system performance metrics: chunk loss rate, playback delay, and propagation tree cost. The chunk (or packet) loss rate is the fraction of chunks that fail to arrive at a peer before playback deadlines; it reflects how smooth the playback is. Although audio streaming service is tolerative to chunk loss, video streaming service is very sensitive. Most users require smooth playback or do not use the service at all. Departures of unsatisfied users cause more lost chunks at satisfied users, resulting in a chain reaction. The playback delay is the time elapsed from when a chunk appears at the source to when the chunk is played at a peer. It consists of two parts: the chunk delivery delay, which is the time from the chunk appears at the source to the time the chunk arrives at the peer, and the chunk buffering delay, which is the time the chunk is buffered at the peer before being played. The amount of network traffic is measured by the average cost of propagation trees. This metric takes into account both inter-AS and intra-AS traffic. Inter-AS traffic usually incurs direct financial cost to an ISP, and intra-AS traffic threatens the quality of other services the ISP provides in addition to indirect financial cost. A propagation tree is a spanning tree of the neighboring overlay. In swarm-based schemes, packets of a chunk follow the same tree; in tree-based schemes, packets of a stream (or substream) follow the same tree. Because a packet may not reach all the peers, we normalize the tree cost by dividing with the number of peers the packet reaches. The tree cost is the sum of all the tree edges' costs. An edge's cost is the cost of the shortest IP path between the two end points. We remark that peers have no access to the underlying IP network link cost information. As is typical in tree-based schemes, we use an overlay edge's propagation delay as its cost because the two parameters are highly correlated.

As we will show later, system performance is highly related to the shape of propagation trees, which is described by the tree height and the average root path length of peers, both in terms of overlay hops. (In this paper, we say an overlay path is long or short depending on its hop count.) A peer's root path refers to the path from the source to the peer along the propagation tree. We only count propagation trees that have reached more than 90% of all the peers when taking averages.

4.2. Simulation setting

We use the following default settings unless specified otherwise. Default values for system parameters are shown in Table 1. We first use the transit-stub model of GT-ITM [23] to generate 10 substrate networks with 40,050 routers and about 200,000 links, then randomly connect 3000 peers to routers. The cost of router–router links is assigned by GT-ITM and ranges from 1 to 3000; the cost of peer–router links is 1. The purpose of using a large number of routers and links is to reflect the complexity of the Internet.

We assume that the bandwidths of pair-wise connections between peers are limited by access links rather than by the Internet core. This assumption is used for simulation in almost all of the studies on large-scale P2P multimedia streaming applications. Because users residing in different areas or using different streaming applications may have different access bandwidth to the Internet, we have simulated with three peers' upload capacity settings: $U(1,4)$, $U(0,6)$, and $U(1,9)$,[3] where $U(a,b)$ denotes uniform distribution over the range of a to b. Each peer attempts to maintain $1.5u_i$ neighbors within the range of 4 to 15, where u_i is peer x_i's upload capacity. On the nearby overlay, peers select randomly from the closest 10% of peers to neighbor with.

The streaming rate is 512 Kb/s. The chunk size is 16 KB. In TS, peers have a swarming window of 240 chunks and an emergency

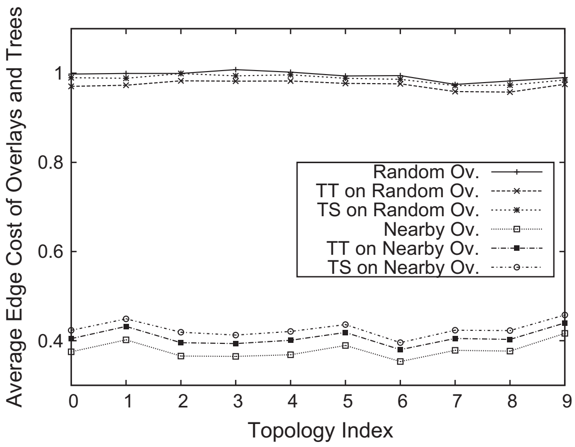

Fig. 1. Average overlay and propagation tree edge cost in TT and TS, normalized with the average cost of all the pair-wise edges between peers in the system.

window of 5 chunks. In TT, peers have no swarming window but need a larger emergency window to recover missing chunks. To emulate a practical setting where peers are asynchronous, we start peers' interval timers with a random offset. All the peers join the system at time 0 to emulate a "flash crowd", the source peer starts streaming at time 5, and the simulation ends at time 600. According to an empirical study [24], we set peer churn rate to 10% of the population per minute. From time 10, one peer is turned off every 0.2 seconds until 600 peers are turned off at time 130. Then every 0.2 seconds, one peer is turned off and one previously turned off peer is turned on. Statistics are collected on the 1500 peers that have never been turned off. At time 5, we dump the overlay graph to obtain its shortest path tree. We collect detailed log files of chunk arrivals and departures at each peer as well as the internal states of peers, such as their buffer maps, residual bandwidths, and outstanding subscription and pull requests. These internal information is necessary for analyzing the impact of scheduling policies and neighboring strategies on system performance. Each test is run for 10 times using the 10 substrate networks, and the average is reported.

4.3. Simulation results[4]

4.3.1. Tree cost

Fig. 1 shows the average edge cost of overlays and propagation trees. As expected, the neighbor-with-nearby-peers strategy greatly reduces the tree cost. TT has a slightly lower tree cost than TS on both overlays. This phenomenon is consistent on all of the ten topologies, indicating that TT has some capability to select low-cost paths. Also note that on the nearby overlay, the average cost of propagation tree edges is slightly higher than overlay edges for both TS and TT, indicating that high-cost edges are more heavily used than low-cost edges on the nearby overlay.

4.3.2. Playback delay and tree shape

Table 2 summarizes the playback delay and propagation tree height. The chunk delivery delay is similar on the two overlays. The chunk buffering delay is mainly a design choice and hence the playback delay is similar on the two overlays. In TS, a large chunk buffering delay has to be used because the chunk delivery delay has a large variance. In TT, the main purpose of the chunk buffering delay is to allow peers to pull chunks that are not pushed,

[3] In this paper, bandwidth is expressed in units of the streaming rate. The access links of broadband Internet users usually have upload capacities between 256 Kb/s and 5 Mb/s. Multimedia streaming applications usually have a streaming rate between 100 Kb/s and 1 Mb/s.

[4] We report the results of using the $U(0,6)$ upload capacity setting as the base case, and report the results of using the other two upload capacity settings only when they lead to different conclusions.

Table 2
Average playback delays.

	Rand. Ov.		Nearby Ov.	
	TS	TT	TS	TT
Playback delay (Sec.)	62.5	6.2	65.3	6.1
Chunk delivery delay (Sec.)	18.8	1.5	18.8	1.3
Tree height (Hops)	9.0	13.4	11.0	16.5
Root path length (Hops)	5.9	7.3	6.7	8.6

and a delay of several seconds suffices. In our simulation, the chunk buffering delay in TS is almost an order of magnitude larger than in TT.

Propagation trees are shorter in TS than in TT on both overlays, and shorter on the random overlay than on the nearby overlay for both schemes. Peers' root path lengths exhibit a similar pattern. These observations suggest that the tree height and root path length have little impact on the playback delay and delivery delay for both schemes.

We are interested in how close propagation trees in TS can approximate the degree-bounded shortest path tree (DBSPT), in terms of hops, given that each peer's degree on the tree is bounded by its upload capacity. Because the DBSPT problem is NP-hard, we configure each peer to have enough upload capacity such that a shortest path tree (SPT) is also a DBSPT for this test. We first set peers upload capacity to 3 and let peers construct the overlays (a random overlay and a nearby overlay), then re-configure each peer's upload capacity to its degree minus one. There is no peer churn and the overlays remain unchanged during the simulation. We dump the overlay graphs and compute the SPT rooted at the source peer using the Dijkstra's algorithm. Fig. 2 shows that the cumulative distribution of peers' root path length on propagation trees in TS and on the SPT are close for the random overlay and nearby overlay. On average, a peer takes 0.6–0.7 more hops than the shortest path to the source peer. This result shows that the height of propagation trees in TS is comparable to that of a DBSPT.

We have observed an interesting phenomenon in TT during simulation. Fig. 3 shows the height of the propagation tree of each chunk on the random overlay. The propagation trees in TT are short initially but gradually grow to a large height, while in TS, the trees have a stable height. This phenomenon appears on both the random overlay and nearby overlay.

4.3.3. Chunk loss rate

Figs. 4 and 5 show the chunk loss rate when the substrate Internet links are error-free and noisy respectively. Different sources report wildly different link error rate (from less than 0.1% to more than 10%); we use an error rate of 1%. We first observe that, on both overlays, TS has fewer lost chunks than TT. Second, TT is more vulnerable to lossy Internet links. When Internet links have an error rate of 1%, as compared with when Internet links are error-free,

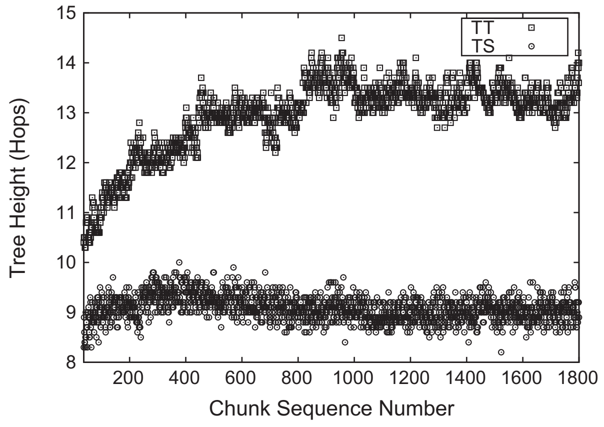

Fig. 3. Propagation tree height in TT and TS on random overlay.

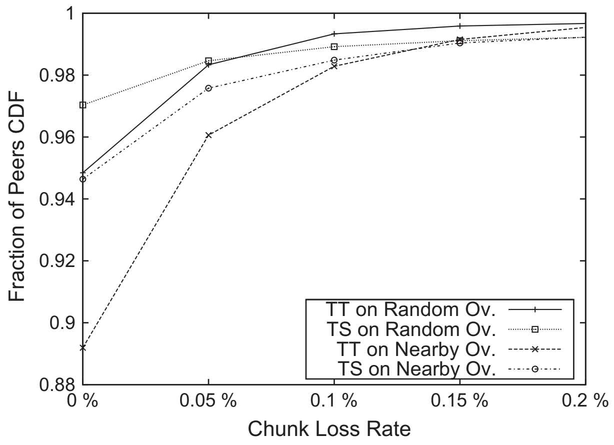

Fig. 4. Peers' chunk loss rate when substrate Internet links are error-free. The *y*-axis is the cumulative fraction of peers whose chunk loss rate is less than the value on the *x*-axis.

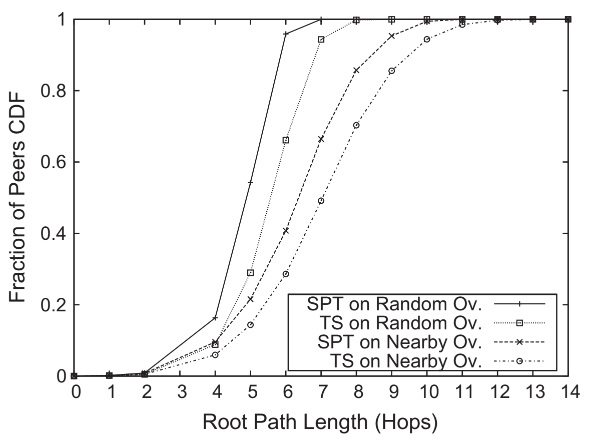

Fig. 2. Cumulative distribution of peers' root path length in TS when each peer's upload capacity is set to its degree minus 1. The *y*-axis is the fraction of peers whose root path length is no more than the value plotted on the *x*-axis.

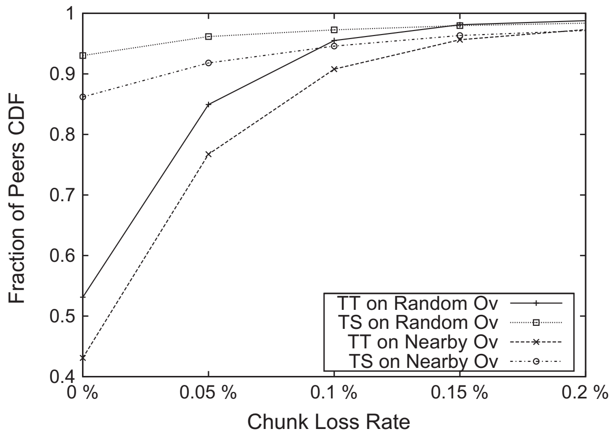

Fig. 5. Peers' chunk loss rate when substrate Internet links have an error rate of 1%. The *y*-axis is the cumulative fraction of peers whose chunk loss rate is less than the value on the *x*-axis.

many more chunks are lost in TT than in TS. Third, for both schemes, more chunks are lost on the nearby overlay than on the random overlay, indicating that the neighbor-with-nearby-peer strategy has a negative impact on chunk delivery. This problem is more severe in TT than in TS. In Fig.4, 5.6% fewer peers received all the chunks on the nearby overlay than on the random overlay in TT (2.4% in TS).

The chunk loss rate has a similar patterns when using the higher capacity setting of $U(1,9)$, but is unstable when using the lower capacity setting of $U(1,4)$ in TS. On the random overlay, the chunk delivery rate is less than 85% on 4 of the 10 topologies; on the nearby overlay, it is 99.9% on 9 topologies and 95.0% on 1 topology.

5. Analysis of chunk propagation

In this section, we study the propagation of chunks on a given overlay in TS and TT. Our focus is to study the probability that the "best" path is chosen over the other paths. We first study the simple case where only two disjoint paths exist from a source node x_0 to a target node x_n (see Fig. 6) and peers have sufficient upload capacity. We then extend the results to cases where multiple paths exist and peers have limited upload capacities.

5.1. Chunk propagation in typical swarm-based (TS) scheme

We model the propagation of a new chunk c as follows. Assume peers' upload capacity and overlay edges' bandwidth are infinite, i.e., there is no transmission and queuing delay.[5] Let d^X_{ij} denote the propagation delay between peers x_i and x_j (along path $x_i, x_{i+1}, \ldots, x_j$). Each peer advertises its buffer map and checks whether to pull chunks every interval of length T; the gap g from the checking time to the advertising time is constant. Peers work in an asynchronous manner, i.e., the checking time and advertising time of peers are uniformly distributed over $[0, T]$. Notations are described in Table 3.

A peer pulls a missing chunk immediately after it finds a neighbor having the chunk. Since there are no transmission or queuing delays, if peer x_i finds a missing chunk available at peer x_{i-1} at time 0, it will obtain the chunk at time $2d^X_{(i-1)i}$. To simplify the expression, we assume the interval T to be 1 time unit, i.e., time is expressed in units of T.

Preposition 1. *In the typical swarm-based (TS) scheme, given two disjoint paths $(x_0, x_1, \ldots, x_n)$ and $(x_0, y_1, \ldots, y_{m-1}, x_n)$ from a source peer x_0 (i.e., y_0) to a target peer x_n (i.e., y_m), the probability that peer x_n pulls chunks from x_{n-1} rather than from peer y_{m-1} is*

$$\widehat{P}_{n,m} = F_{m+n}(m - D^X_n + D^Y_m) \tag{1}$$

where F is the CDF of the sum of $m+n$ i.i.d $U(0,1)$, $D^X_n = ng + d^X_{0n}$, and $D^Y_m = mg + d^Y_{0m}$.

Proof. Let $\hat{\delta}^X_i$ be the random variable denoting the time when peer x_{i-1} obtains a chunk c to the time peer x_i obtains the same chunk c. Assume peer x_{i-1} obtains chunk c at time t. After fetching chunk c, it advertises its buffer map at time $t+g$. The message reaches peer x_i at time $t + g + d^X_{(i-1)i}$. The time peer x_i fetches chunk c is uniformly distributed over $(t + g + d^X_{(i-1)i}, t + g + d^X_{(i-1)i} + 1)$. Therefore $\delta^X_i = \hat{\delta}^X_i - (g + d^X_{(i-1)i})$ is a $U(0,1)$ random variable. Let X_i be the random variable denoting the time from peer x_0 obtains a chunk c to the time peer x_i obtains chunk c. Then $X_n - D^X_n$ is the sum of n i.i.d $U(0,1)$. Likewise, $Y_m - D^Y_m$ is the sum of m i.i.d $U(0,1)$. The sum of n i.i.d $U(0,1)$ has PDF [25]

$$f_n(t) = \frac{1}{2(n-1)!}\sum_{k=0}^{n}(-1)^k \binom{n}{k}(t-k)^{(n-1)}\mathrm{sgn}(t-k) \tag{2}$$

where sgn(·) is the sign function.

It can be shown that, for two i.i.d $U(0,1)$ random variables X and Y, $p_{(X-Y)}(t) = p_{(X+Y-1)}(t)$. Therefore, $p_{(X_n - D^X_n - Y_m + D^Y_m)}(t) = f_{m+n}(t+m)$. The probability that peer x_n pull chunk c from x_{n-1} rather than from y_{m-1} is

$$\widehat{P}_{n,m} = P\{X_n < Y_m\} = \int_{-\infty}^{0} p_{(X_n - Y_m)}(t)dt$$
$$= \int_{-\infty}^{0} f_{m+n}(t - D^X_n + D^Y_m + m)dt = F_{m+n}(m - D^X_n + D^Y_m) \quad \square \tag{3}$$

By decomposing $X_n - Y_m$, we can show the impact of each element. We are particularly interested in the cases when m and n are small since a long path can be divided into several short segments.

$$X_n - Y_m = (n-m)g + (d^X_{0n} - d^Y_{0m}) + (\Delta^X_n - \Delta^Y_m) \tag{4}$$

where $\Delta^X_n = \sum_{i=1}^{n}\delta^X_i$ and $\Delta^Y_m = \sum_{i=1}^{m}\delta^Y_i$. $\Delta^X_n - \Delta^Y_m$ has variance $\frac{m+n}{12}$; the other two elements are constant.

From Eqs. 1 and 4, we can draw the following conclusions. First, since the difference of propagation delays $d^X_{0n} - d^Y_{0m}$ is small compared with the interval T, it has little impact on path selection, i.e., peers cannot select low-cost paths in TS. Therefore, the average cost of propagation trees in TS is similar to that of the overlay edges in Fig. 1. Second, a short path has a significantly larger probability to be selected over a long path. When g and $d^X_{0n} - d^Y_{0m}$ are small enough to be ignored, $\widehat{P}_{n,m}$ has a closed form when $m = 1$, i.e., the probability that x_n retrieves chunk c from an n-hop path $(x_0, x_1, \ldots, x_n)$ rather than a 1-hop path (y_0, y_m) is $\widehat{P}_{n,1} = \frac{1}{(n+1)!}$. When $m > 1$, the probability can be calculated but has no closed form. Third, a large gap g can dramatically increase the probability of short paths being selected. For example, when g is set to its maximum value of 1 (i.e., the interval T), a 1-hop path will always be chosen over an n-hop ($n \geqslant 2$) path, and a 2-hop path is chosen over

[5] Queuing delay is the amount of time a packet waits at the sender to be transmitted; transmission delay is the amount of time required to push a packet onto a link; and propagation delay is the amount of time required for the head of a packet to travel from the sender to the receiver over a link.

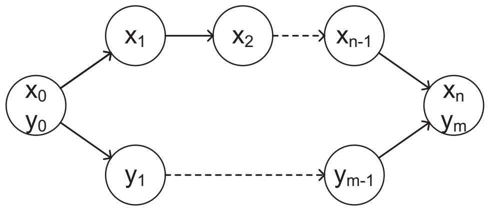

Fig. 6. Two disjoint paths with n and m hops, $n > m$. The leftmost peer is labeled as x_0 and y_0 and the rightmost peer as x_n and y_m for convenience.

Table 3
Notations

Notation	Comment
g	The gap from checking time to advertising time
δ	A $U(0,1)$ random variable
f_i, F_i	The PDF and CDF of the sum of i $U(0,1)$
$d^X_{ij}, (d^Y_{ij})$	The propagation delay from node x_i to node x_j (from node y_i to node y_j)
N	The population of the P2P system
M	The size of the set of nearby peers
S_i	The set of peers close to node x_i
w_i	The weight of peer x_i
r_i	The radius of set S_i

a 3-hop path with a probability of $\frac{23}{24}$ (compared with $\frac{31}{40}$ when g is 0). In our simulation, g is set to 1. Because of the second and third points, TS has short propagation trees (see Table 2 and Fig. 2).

5.2. Chunk propagation in typical tree-based (TT) scheme

We model the tree-building and propagation of new chunks as follows. We also assume that peers' upload capacities and overlay edges' bandwidths are infinite. Each peer advertises its position in the stream and checks whether to switch parents every interval of length T; the gap g from the checking time to the advertising time is constant. If peer x_n finds neighbor x_{n-1} proceeding further in the stream than its parent y_{m-1}, it immediately switches to that neighbor, i.e., it becomes the child of peer x_{n-1} after a time of length $2d^X_{(n-1)n}$. A peer, after receiving a chunk, immediately pushes the chunk to its children. For example, if peer x_{i-1} obtains a chunk at time 0, its child x_i will receive the chunk at time $d^X_{(i-1)i}$. Peer x_0 begins streaming at time 0.

Preposition 2. *In the typical tree-based (TT) scheme, given two disjoint paths $(x_0, x_1, \dots, x_n)$ and $(x_0, y_1, \dots, y_{m-1}, x_n)$ from a source peer x_0 (i.e., y_0) to a target peer x_n (i.e., y_m), $(x_0, \dots, x_{n-1})$ and $(x_0, \dots, y_{m-1})$ are two tree branches rooted at x_0, the probability that peer x_n subscribes to x_{n-1} rather than to y_{m-1} is*

$$\check{P} = \begin{cases} 0, & \tau \geqslant 1, \\ \frac{1}{2} + \tau - \frac{\tau^2}{2}, & -1 \leqslant \tau < 1, \\ 1, & \tau < -1, \end{cases} \tag{5}$$

where $\tau = d^X_{0n} - d^Y_{0m}$.

Proof. Assume $d^X_{0n} \geqslant d^Y_{0m}$; the proof when $d^X_{0n} < d^Y_{0m}$ is the same. To simplify the expressions, we assume that peer x_0's latest position in the stream at time 0 is 0. Peer x_{n-1} advertises, at time t^X_1, its playback position $t^X_1 - d^X_{0(n-1)}$. The message arrives at peer x_n at time $t^X_2 = t^X_1 + d^X_{(n-1)n}$. Obviously, $t^X_1 - d^X_{0(n-1)} = t^X_2 - d^X_{0n}$. Similarly, peer y_{m-1} advertises at time t^Y_1, and the message arrives at peer x_n at time $t^Y_2 = t^Y_1 + d^Y_{(m-1)m}$, carrying peer y_{m-1}'s playback position $t^Y_2 - d^Y_{0m}$.

Since peers are asynchronous, $t = t^X_2 - t^Y_2$ follows the $U(0,1)$ distribution. Let t_c denote the time peer x_n attempts to switch parents. If $t_c \geqslant t^X_2$, $t_c \geqslant t^Y_2$, and $t^X_1 \geqslant t^Y_1$, then peer x_n will subscribe to peer x_{n-1}; otherwise peer x_n will subscribe to peer y_{m-1}. When $d^X_{0n} \geqslant d^Y_{0m} + 1$, the expression $t^X_1 \geqslant t^Y_1$ does not hold, hence peer x_n will always subscribe to peer y_{m-1}. When $d^X_{0n} < d^Y_{0m} + 1$, conditioning on t, the probability that peer x_n subscribes to peer x_{n-1} is

$$\check{P} = \int_{\tau}^{1} (1-t)dt = \frac{1}{2} - \tau + \frac{\tau^2}{2} \quad \square \tag{6}$$

Eq. 5 shows that peers in TT choose low-cost paths with a slightly higher probability (which explains why TT has slightly lower tree cost in Fig. 1), and the probability increases when using a smaller interval T. The gap g between peers' checking time and advertising time is irrelevant in path selection. In TT, the time difference between peer x_{i-1} and its child x_i obtain a chunk is only the propagation delay $d_{(i-1)i}$, which is one or two orders of magnitude smaller than the interval T. Compared with swarm-based schemes where each hop incurs a delay of more than $T/2$ on average, the edge (x_jx_j) seems "short-circuited". Because of this short-circuit effect, neighbors' root path lengths have no direct impact on peers' decisions, resulting in taller propagation trees (see Table 2).

The propagation model also explains the growth of propagation trees in TT in Fig. 3. In our simulation, because all the peers join before the streaming begins, for the first few chunks, the fewer hops a peer is from the source peer on the neighboring overlay, the earlier it attaches to the tree. Therefore, the tree height is small initially. With peer churn, peers switch parents. Because peers' root path lengths have no impact on parent selection (i.e., a peer selects parents randomly with respect to neighbors' root path lengths), the tree height grows.

5.3. Impact of multiple paths and limited upload capacity

Section 5.2 shows that in TT, peers select low-cost paths with a higher probability; we continue to investigate whether peers in TT can retain this capability when multiple paths exist and peers have limited upload capacity. Let $\Gamma_n = \{y, x^1, \dots, x^k\}$ be the set of non-child neighbors of peer x_n with lags $d^y < d^1 < \dots < d^k$ in the stream behind the source. Let p_i denote the probability that peer y is chosen over peer x^i when only neighbor x^i exists, then the probability that peer y is chosen when all the neighbors exist is $P = \prod_{i=1}^{k} p_i$. Therefore, the existence of multiple paths reduces peers' capability to choose low-cost paths in TT. When peers have limited upload capacity, the value of τ in Eq. 5 is augmented to include the queuing and transmission delays. When overlay edges have stable bandwidths and peers employ CAC, the queuing and transmission delays are proportional to path length. Therefore, peers capability to select low-cost paths decreases when peers have limited upload capacity, but the capability to select short paths increases. In practical settings, these changes are small because the interval T is typically one or two orders of magnitude larger than the propagation, queueing, and transmission delays. Therefore, peers still cannot select short paths and have a slightly higher probability to select low-cost paths.

Section 5.1 shows that TS can select short paths; we continue to investigate whether it can retain this capability when multiple paths exist and peers have limited upload capacity. Assume $\vec{y}, \vec{x}^1, \dots, \vec{x}^k$ are disjoint paths from x_0 to x_n, and let p_i denote the probability that path $\vec{y}$ is chosen over path $\vec{x}^i$ when only the two paths exist, then the probability that path $\vec{y}$ is chosen when all the paths exist is $P = \prod_{i=1}^{k} p_i$. When alternate paths are not disjoint, the probability cannot be expressed in a closed form. As a rule of thumb, adding cross edges between alternate paths $\vec{x}^1, \dots, \vec{x}^k$ decreases the probability that path $\vec{y}$ is chosen. Eq. 1 indicates that a long path has a significant lower probability to be chosen, and hence whether TS can select short paths depends on the number of short alternate paths. (We will discuss the number of alternative paths in Section 6). When peers have limited upload capacities, although the value of $X_n - Y_m$ in Eq. 4 is augmented to include queuing and transmission delays at each hop, they are too small to have any impact.

6. Analysis of the impact of overlays

In this section, we analyze the impact of the neighbor-with-nearby-peers strategy on system performance. We first introduce the random graph model that is used to construct the random and nearby overlays, then compare their properties that have the most impact on system performance, and finally, together with the packet propagation model, analyze these properties' impact on system performance in the two typical schemes.

6.1. Properties of random and nearby overlays

There exist a number of random graph models that produce different distributions of nodes and edges on the graph. Like most Internet topology generators, we use the $G(N,p)$ model to construct the random and nearby overlays. First N nodes are randomly placed on a plane, and then for each pair of nodes, an edge is added with probability p. The Euclidean distance on the plane between two nodes is their overlay edge's cost. Let w denote the desired average node degree on the neighboring overlay. On the random overlay, the probability p is set to $\mu = \frac{w}{N}$. The random graph is expected to have wN edges (including w self loops). On the nearby overlay, we only add edges between nearby nodes. Let S_i denote the set of M nodes that are closest to node x_i. Let r_i denote the distance between node x_i and the farthest node in S_i. For a pair of nodes x_i and x_j, the probability p is set to $v = \frac{w}{M}$ if $x_i \in S_j$ and $x_j \in S_i$ and to 0 otherwise. We remark that if peers greedily neighbor with nearby peers, the resulting overlay is inevitably partitioned; certain extent of randomness must be employed. Notations are shown in Table 3.

The overlay properties that have the most impact on system performance include edge cost, diameter, connectivity, and clustering coefficient. Obviously, the nearby overlay has a lower edge cost and a larger diameter than the random overlay. The random graph has a diameter of $O(\log N)$. The nearby overlay has a diameter of $O(N^{\frac{1}{\alpha}})$, where α is the dimension of the space in which nodes are placed (α equals 2 in our case). The connectivity of a graph measures how easily (or difficultly) a graph can be partitioned. The nearby overlay is more prone to partitioning than the random overlay, especially in the presence of peer churn. Temporary partitioning is tolerable in file sharing applications, but greatly degrades the quality of multimedia streaming service. We find that the system may be partitioned even when peers neighbor with the closest 10% of peers. The clustering coefficient measures the extent to which peers cluster together. It is defined as the number of closed triplets over the total number of triplets on a graph. (A triplet is three nodes that are connected by either two edges or three edges; in the latter case, it is called a closed triplet.) The nearby overlay has a much larger clustering coefficient than the random overlay.

When a neighboring overlay has a high clustering coefficient, for each edge, there tends to be more short alternate paths. As discussed in the propagation model, this will have an impact on the probability that a peer chooses the "best" path. We quantitatively analyze the number of short alternate paths on the random overlay and nearby overlay in the following two prepositions.

Preposition 3. *On a random overlay, the expected number of l-hop paths between two arbitrary nodes is* $O(\frac{1}{N})$ *when l is small and N is large.*

Proof. Let P be the matrix where the (i,j)-th element $p_{i,j}$ denotes the probability that an edge exists between nodes i and j. Let Q be the matrix where $q_{i,j}$ denotes the probability that a path of length n exists between node i and j. Let Q' be the matrix where $q'_{i,j}$ denotes the probability that a path of length $n+1$ exists between nodes i and j. Then $q'_{i,j} = \sum_{k=1}^{N} q_{i,k} p_{k,j}$, i.e., $Q' = QP$. Therefore, the probability $p^l_{i,j}$ that a path of length l exists between nodes i and j is the (i,j)-th element of P^l when $p^l_{i,j} < 1$ for all i and j. For the random overlay, when N is large and l is small, $p^l_{i,j} \approx \frac{w^l}{N}$, which is also the expected number of l-hop paths between nodes i and j. $\square$

Preposition 4. *On a nearby overlay, the expected number of l-hop paths between an arbitrary pair of adjacent nodes is* $O(1)$ *when l is small.*

Proof. Assume nodes x_i and x_j have w_i and w_j neighbors respectively. We first find the expected number of nodes that are in both S_i and S_j. Without loss of generality, we assume $r_i \geqslant r_j$. Let c_{ij} denote the distance (substrate IP path cost) between node x_i and x_j. For an arbitrary node in S_j, the probability ψ that it is in S_i is the proportion of the intersection of the two circles in Fig.7. When $c_{ij} = 0$, the two sets S_i and S_j are identical and ψ is 1. When $c_{ij} = r_i = r_j, \psi$ has its minimum value

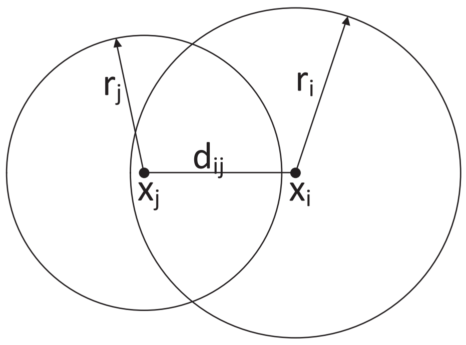

Fig. 7. Peer distribution on nearby overlay.

$$\psi = \frac{\frac{2}{3}\pi r^2 - \sqrt{3} r^2}{\pi r^2} \approx 0.12 \tag{7}$$

Assuming S_i and S_j have M' common nodes, the probability $Q_{2,k}$ that nodes x_i and x_j is connected via k 2-hop path is

$$Q_{2,k} \geqslant \binom{M'}{k} (v^2)^k (1 - v^2)^{M'-k}, \quad k \in [0, M'] \tag{8}$$

where the greater sign comes from the fact that nodes x_i and x_j may be connected via a node not in S_i or S_j. Since the distribution of nodes on the plane and the probability of an edge exiting between two peers are independent, the expected number of 2-hop paths is no less than $M'v^2 = \frac{\psi w_i w_j}{M}$, where $\psi \in [0.12, 1]$.

On average, node x_j has ψw_j adjacent nodes in the intersection area; each node has $M'v^2 = \frac{\psi w^2}{M}$ 2-hop paths to node x_i (assuming they have w neighbors). Although some paths are not edge-disjoint, the number of 3-hop edges is still $O(1)$. The proofs for $l = 2, 3, \ldots$ are similar. Note that the total number of edge-disjoint paths are bounded by $min\{w_i, w_j\}$. $\square$

6.2. Impact of neighbor-with-nearby-peers strategy

Since nearby overlay edges have a lower cost than random overlay edges, any spanning tree on the nearby overlay will have a smaller tree cost, and hence the propagation trees in both TS and TT will have smaller costs on the nearby overlay (see Fig. 1).

Since the nearby overlay has a larger diameter than the random overlay, the propagation tree is higher in both TT and TS on the nearby overlay (see Table 2). However, the difference in tree height between the two overlays is larger in TT than in TS because of the two schemes' different chunk propagation behaviors. In TT, the lower edge cost on the nearby overlay makes the value τ in Eq. 5 smaller, and hence peers select parents more randomly with respect to neighbors' root path lengths. In TS, because peers choose short paths with a significantly large probability, propagation trees are short unless a substantial number of short alternative paths exist between a pair of adjacent peers (see Section 5.3). On the random overlay, the number of short alternative paths goes to 0 when N goes to infinity. On the nearby overlay where peers neighbor with the closest 10% of peers, according to Proposition 4, the number is only an order of magnitude larger than on the random overlay, which will have little impact. Therefore, propagation trees in TS are comparable to the DBSPT on both the random overlay and nearby overlay (see Fig.2).

Despite the propagation tree being taller on the nearby overlay, the average playback delay and chunk delivery delay only differ slightly on the two overlays in both TT and TS (see Table 2). In TS, this is mainly because the queuing delay is shorter at each peer (due to less children). In TT, in addition to a shorter queuing delay, this is also because of the shorter propagation delays of nearby overlay edges.

Because TS uses a large buffering delay, it often has a lower chunk loss rate than TT and is more robust when Internet links are erroneous (see Figs. 4 and 5). Now we discuss the impact of the overlays on the chunk loss rate. When peers lack upload capacity to send all the chunks to all the requesting neighbors, or peer churn and substrate network errors exist, chunks propagate along different paths. When a neighboring overlay has a higher clustering coefficient, the root path of a peer tends to be more "similar", i.e., these paths share more common nodes and edges and have similar hop counts. Therefore, compared with the random overlay, on the nearby overlay, chunks tend to propagate along a group of longer but similar paths. This property has different impacts on the chunk loss rate in TT and TS because of their different chunk propagation behavior.

In TT, a peer attempts to pull all the chunks not pushed by its parent. The fewer chunks that need to be pulled and the higher the pull success rate, the higher the chunk delivery rate. Because TT has a short playback delay, if a chunk is late at a peer, its descendants may also need to pull the chunk. On the random overlay, these peers have different neighbors to pull the chunk from. On the nearby overlay, these peers share many common neighbors; it is hard for them to find neighbors with the chunk and spare upload capacity in a short time. Therefore, the pull success rate is lower and hence more chunks are in TT on the nearby overlay (see Figs.4 and 5).

In TS, for a chunk *c*, at each hop and in each interval, the peer may decide to pull other chunks rather than chunk *c*, causing chunk *c* to miss its playback deadline. Therefore, longer root paths on the nearby overlay usually cause more lost chunks in TS. However, because swarm-based schemes typically have a large chunk buffering delay, the increase is usually not severe (see Figs. 4 and 5).

Because chunks propagate along a group of similar paths on the nearby overlay, the chunk delivery delay has a smaller variance than on the random overlay. If the chunk buffering delay is set to a value that cannot absorb the variance, the nearby overlay may results in fewer lost chunks. This is the reason for the unstable performance in TS when using the lower setting of $U(1,4)$. By examining the detailed log file, we find that on the random overlay, peers' root paths have more diverse hop counts, especially peers within a few hops to the source peer on the neighboring overlay, causing many chunks to miss playback deadlines. On the nearby overlay, peers' root paths are longer but have smaller variance. Chunks arrive in time regardless of which paths they take.

7. Conclusion

In this paper, we studied packet propagation behavior and the impact of neighbor-with-nearby-strategy on system performance in P2P multimedia streaming applications. We identified two "typical" schemes, a swarm-based and a tree-based, and compared their performance on random overlays and nearby overlays. We first conducted simulation study and then provided models to analyze packet propagation paths in the two typical schemes on a given overlay and models to analyze the impact of the two types of overlays on system performance. We found that chunks propagate in different manners in the two schemes. In the swarm-based scheme, peers have a large probability to pull chunks from short paths, and propagation trees are comparable to DBSPT. In the tree-based scheme, because of the short-circuit effect, peers tend to select parents randomly with respect to neighbors' hop counts to the source peer on the neighboring overlay, and hence propagation trees have a large height. The neighbor-with-nearby-peers strategy reduces the propagation tree cost but increases the risk of system partitioning and causes more lost chunks. The different chunk propagation behaviors of the swarm-based and tree-based schemes cause chunk loss in different ways in the two schemes. The neighbor-with-nearby-peers strategy causes more lost chunks in the tree-based schemes than in the swarm-based schemes.

References

[1] X. Zhang, H. Hassanein, On network utilization of peer-to-peer video live streaming on the internet, in: Proceedings of International Conference On Communications (ICC), 2011, pp. 1–5.

[2] S. Ali, A. Mathur, H. Zhang, Measurement of commercial peer-to-peer live video streaming, in: Proceedings of Workshop in Recent Advances in Peer-to-Peer Streaming, 2006.

[3] V. Aggarwal, A. Feldmann, C. Scheideler, Can ISPS and P2P users cooperate for improved performance?, ACM SIGCOMM Computer Communication Review 37 (3) (2007) 29–40.

[4] H. Xie, Y. Yang, A. Krishnamurthy, Y. Liu, A. Silberschatz, P4P: provider portal for P2P applications, in: Proceedings of ACM SIGCOMM Conference on Data Communication, 2008, pp. 351–362.

[5] D. Choffnes, F. Bustamante, Taming the torrent, in: Proceedings of ACM SIGCOMM, 2008.

[6] J. Liu, S.G. Rao, B. Li, H. Zhang, Opportunities and challenges of peer-to-peer internet video broadcast, Proc. IEEE 96 (1) (2008) 11–24.

[7] <http://www.bittorrent.com>, Accessed Mar. 2011.

[8] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, A.E. Mohr, Chainsaw: eliminating trees from overlay multicast, in: Proceedings of International Workshop on Peer-to-Peer Systems, 2005.

[9] X. Zhang, J. Liu, B. Li, T.S.P. Yum, Coolstreaming/donet: a data-driven overlay network for efficient live media streaming, in: Proceedings of IEEE INFOCOM, vol. 3, 2005, pp. 13–17.

[10] N. Magharei, R. Rejaie, Prime: peer-to-peer receiver-driven mesh-based streaming, in: Proceedings of IEEE INFOCOM, 2007, pp. 1415–1423.

[11] F. Pianese, D. Perino, J. Keller, E. Biersack, PULSE: an adaptive incentive-based unstructured P2P live streaming system, IEEE Transactions on Multimedia 9 (8) (2007) 1645–1660.

[12] Z. Liu, Y. Shen, K. Ross, S. Panwar, Y. Wang, LayerP2P: using layered video chunks in p2p live streaming, IEEE Transactions on Multimedia 11 (7) (2009) 1340–1352.

[13] Y. Chu, S. Rao, S. Seshan, H. Zhang, A case for end system multicast, IEEE Journal on Selected Areas in Communications 20 (8) (2002) 1456–1471.

[14] S. Banerjee, B. Bhattacharjee, C. Kommareddy, Scalable application layer multicast, in: Proceedings of ACM SIGCOMM Conference on Applications Technologies Architectures and Protocols for Computer Communication, vol. 32 , 2002, pp. 205–217.

[15] F. Wang, Y. Xiong, J. Liu, B. Burnaby, C. Beijing, mTreebone: a hybrid tree/mesh overlay for application-layer live video multicast, in: Proceedings of ICDCS, 2007, p. 49.

[16] D. Ren, Y.-T. Li, S.-H. Chan, Fast-mesh: a low-delay high-bandwidth mesh for peer-to-peer live streaming, IEEE Transactions on Multimedia 11 (8) (2009) 1446–1456.

[17] M. Zhang, L. Zhao, Y. Tang, J.G. Luo, S.Q. Yang, Large-scale live media streaming over peer-to-peer networks through global Internet, in: Proceedings of ACM Workshop on Advances in Peer-to-Peer Multimedia Streaming, 2005, pp. 21–28.

[18] S. Xie, B. Li, G.Y. Keung, X. Zhang, Coolstreaming: design theory and practice, IEEE Transactions on Multimedia 9 (8) (2007) 1661–1671.

[19] Z. Liu, Y. Shen, K. Ross, S. Panwar, Substream trading: towards an open p2p live streaming system, in: IEEE International Conference on Network Protocols, 2008, pp. 94–103.

[20] K.-H. Chan, S.-H. Chan, A. Begen, Spanc: Optimizing scheduling delay for peer-to-peer live streaming, IEEE Transactions on Multimedia 12 (7) (2010) 743–753.

[21] Y. Zhou, D. Chiu, J. Lui, A simple model for analyzing p2p streaming protocols, in: IEEE International Conference on Network Protocols, ICNP 2007, 2007, pp. 226–235.

[22] F. Picconi, L. Massoulié, Is there a future for mesh-based live video streaming? in: Eighth International Conference on Peer-to-Peer Computing P2P'08 (2008), 2008. pp. 289–298.

[23] <http://www.cc.gatech.edu/projects/gtitm/>, Accessed Mar. 2011.

[24] B. Li, S. Xie, G.Y. Keung, J. Liu, I. Stoica, H. Zhang, X. Zhang, An empirical study of the Coolstreaming system, IEEE Journal on Selected Areas in Communications 25 (9) (2007) 1627–1639.

[25] <http://mathworld.wolfram.com/uniformsumdistribution.html>, Accessed Aug. 2010.