

PRIME: Peer-to-Peer Receiver-driven Mesh-based Streaming

Nazanin Magharei, Reza Rejaie
University of Oregon
{nazanin,reza}@cs.uoregon.edu

Abstract—The success of swarming content delivery has motivated a new approach to live Peer-to-Peer (P2P) streaming that we call mesh-based streaming. In this approach, participating peers form a random mesh and incorporate swarming content delivery to stream live content. Despite the growing popularity of this approach, neither the design tradeoffs nor the basic performance bottlenecks in mesh-based P2P streaming are well understood.

This paper presents PRIME, the first mesh-based P2P streaming for live content that effectively incorporates swarming content delivery. We identify two performance bottlenecks in a mesh-based P2P streaming, namely *bandwidth bottleneck* and *content bottleneck*. We derive proper peer connectivity to minimize bandwidth bottleneck as well as an efficient pattern of delivery for live content over a random mesh to minimize content bottleneck. We show that the pattern of delivery can be divided into diffusion and swarming phases and then identify proper packet scheduling algorithm at individual peers. Using ns simulations, we examine key characteristics, design tradeoffs and the relationship between main system parameters.

I. INTRODUCTION

Peer-to-Peer (P2P) overlays offer a promising approach to stream *live* video from a single source to a large number of receivers (or peers) over the Internet without any special support from the network. This approach is often called *P2P streaming*. The goal of P2P streaming mechanisms is to maximize delivered quality to individual peers in a scalable fashion while accommodating the heterogeneity and asymmetry of access link bandwidth and churn among participating peers. To effectively scale with the number of participating peers in a session, a P2P streaming mechanism should be able to utilize the contributed resources (namely outgoing bandwidth) by individual peers.

A well known approach to P2P streaming is to organize participating peers into multiple, diverse tree-shaped overlays where each specific sub-stream of the live content is *pushed* through a particular tree from source to all interested peers (e.g., [1]). This approach has the following potential limitations: (i) In the presence of churn, maintaining multiple diverse trees could be very challenging [2]. (ii) The rate of content delivery to each peer through individual trees is limited by the minimum throughput among the upstream connections. (iii) The outgoing bandwidth of those peers that do not have a sufficient number of child peers or an adequate amount of new content can not be effectively utilized. This in turn limits the scalability of the tree-based approaches.

An alternative approach to P2P streaming is *mesh-based P2P streaming*. In this approach participating peers form

a mesh-shaped overlay and incorporate *swarming* (or pull) content delivery. File swarming mechanisms (e.g., [3], [4]) leverage the elastic nature and the availability of the entire file at the source to distribute different pieces of a file among participating peers, enabling them to actively contribute their outgoing bandwidth through swarming. However, incorporating swarming content delivery into mesh-based P2P streaming mechanisms for “live” content is challenging for two reasons: (i) Accommodating the streaming constraint of in-time delivery for individual packets is difficult, and (ii) Since the content is progressively generated by a live source, the limited availability of future content limits the diversity of available pieces among participating peers which in turn degrades the utilization of their outgoing bandwidth.

Recently, several studies have proposed new P2P streaming mechanisms to address some of the above limitations. Cool-Streaming [5] is a data-driven approach where participating peers initially form a mesh. Once each peer identifies proper parents, it requests each parent to provide a specific sub-stream of the content. In essence, CoolStreaming eventually organizes participating peers into multiple trees and incorporates push-based content delivery [6]. ChunkySpread [7] is another protocol that forms multiple trees over a mesh and pushes each sub-stream through a separate tree. ChunkySpread uses a heavy-weight signaling mechanism to enable individual peers to frequently replace their low-performing parents. A couple of studies have proposed to add the notion of “delivery window” to Bittorrent in order to support “streaming” content delivery (e.g., [8]). These studies appear to be targeting playback streaming applications and have only examined a small number of simple and resourceful scenarios. Finally, several P2P streaming systems (e.g., www.sopcast.com) have become available for broadcasting popular live events such as World Cup 2006. However, no technical details about these systems is available. In summary, to our knowledge, previous studies have not answered the following important questions:

- How can swarming content delivery be properly incorporated into a live P2P streaming mechanism?
- What are the fundamental tradeoffs and limitations to incorporate swarming content delivery into mesh-based P2P streaming for live content?

This paper presents *PRIME*, the first mesh-based P2P streaming mechanism for delivery of *live* content that effectively incorporates swarming content delivery. We follow a problem-driven approach to design PRIME. Towards this end, first we identify two performance bottlenecks in mesh-based P2P streaming, namely *bandwidth bottleneck* and *content bot-*

This material is based upon work supported by the NSF CAREER Award CNS-0448639.

tleneck, that could limit the utilization of available resources. Then, we show how the incoming and outgoing degrees of individual peers should be determined in order to minimize the probability of bandwidth bottleneck. To design a proper *packet scheduling* algorithm for content delivery, we introduce the organized view of a random mesh and then derive the pattern of content delivery that minimizes the probability of content bottleneck. We demonstrate that the pattern of delivery for each segment should consist of *the diffusion* and *the swarming* phases, based on the direction that data flows. The notion of diffusion and swarming phases offers a powerful method to identify the performance bottlenecks in mesh-based P2P streaming. Leveraging this method, we derive the relationship between key parameters of the system and illustrate their impact on system performance through *ns* simulations. Our results not only reveal a few fundamental design tradeoffs and limitations in incorporating swarming content delivery into mesh-based P2P streaming for live content but also shed an insightful light on the dynamics of swarming content delivery in these systems. This paper builds and significantly expands on our earlier work on mesh-based P2P streaming [9].

The rest of this paper is organized as follows: In Sections II and III, we describe two key components of PRIME, namely overlay construction and content delivery mechanisms, respectively. Section IV presents simulation-based evaluations of PRIME and illustrates some of its key tradeoffs and limitations. Section V concludes the paper and sketches our future plans.

II. OVERLAY CONSTRUCTION IN PRIME

In PRIME, participating peers form a *randomly* connected and *directed* mesh. Each participating peer in the overlay has multiple parents and multiple child peers. All connections in the overlay are congestion controlled (using RAP or TFRC) and are always initiated by the corresponding child peer. Each peer tries to maintain a sufficient number of parents that can collectively fill its incoming access link bandwidth. When a peer needs one (or more) new parent(s), it contacts a bootstrapping node to learn about a random subset of other participating peers in the system and then request those peers to serve as its parent. Such a mesh-based overlay is easy to maintain and is very resilient to churn. Furthermore, incoming and outgoing connections of each peer are more likely to have diverse paths which in turn reduces the probability of a shared bottleneck among them. The key design question for the overlay construction mechanism is “how to determine the incoming and outgoing degrees of individual peers in order to maximize the utilization of their incoming and outgoing access link bandwidth?”

Proper Incoming/Outgoing Peer Degree: Suppose that each peer always has some useful packets to be requested by its child peers. Then, the aggregate bandwidth to each child peer depends not only on its own incoming degree but also on the outgoing degree of its parent peers. Without loss of generality, we assume that congestion only occurs at the edge of the network, *i.e.*, at the incoming or outgoing access links of

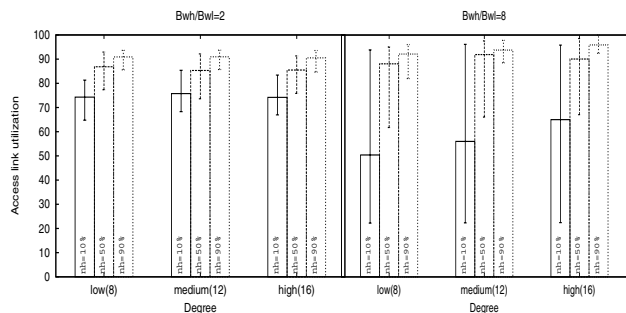


Fig. 1. Access link utilization

participating peers. Therefore, the average bandwidth for a connection between parent peer p to child peer c can be roughly estimated with $\text{MIN}(\frac{\text{outbw}_p}{\text{outdeg}_p}, \frac{\text{inbw}_c}{\text{indeg}_c})$ where outbw_p , outdeg_p , inbw_c , indeg_c denote the outgoing bandwidth and outgoing degree of peer p , and incoming bandwidth and incoming degree of peer c , respectively. If $\frac{\text{outbw}_p}{\text{outdeg}_p} < \frac{\text{inbw}_c}{\text{indeg}_c}$, the outgoing access link of the parent peer is the bottleneck and thus the incoming access link of the child peer may not be fully utilized. In contrast, if $\frac{\text{outbw}_p}{\text{outdeg}_p} > \frac{\text{inbw}_c}{\text{indeg}_c}$, the bottleneck is at the incoming access link of the child peer and the outgoing access link of the parent may not be fully utilized.

This observation suggests that to maximize the utilization of both incoming and outgoing access link bandwidth of all peers in a randomly connected overlay, the same bandwidth to degree ratio should be used for both the outgoing and incoming links of *all* participating peers. More specifically, the following condition should be satisfied for any two randomly selected peers i and j in the overlay: $\text{bwpf} = \frac{\text{outbw}_i}{\text{outdeg}_i} = \frac{\text{inbw}_j}{\text{indeg}_j}$. We call this *bandwidth-degree condition* and it implies that all connections in the overlay should have roughly the same bandwidth of bwpf , or bandwidth-per-flow. In essence, bwpf directly translates the (potentially heterogeneous and asymmetric) incoming and outgoing access link bandwidths of individual peers (and the source) to their incoming and outgoing degrees, respectively.

To examine the effect of bandwidth-degree condition on the utilization of access link bandwidth, we conduct *ns* simulation where 200 peers with heterogeneous incoming link bandwidth (bw_h and bw_l) form a random mesh. All peers use the same incoming and outgoing degree and all connections are congestion controlled using RAP. Figure 1 depicts the average utilization of incoming link bandwidth and its 10 and 90 percentiles (as bar) only among high bandwidth peers for two levels of bandwidth heterogeneity, *i.e.*, $\frac{\text{bw}_h}{\text{bw}_l}$ is equal to 2 and 8, respectively. We also examine each scenario with three different values of peer degree (namely 8, 12 and 16) and different fraction of high bandwidth peers (n_h) for each degree. Across all these simulations, the incoming link of low bandwidth peers has always achieved high utilization. Figure 1 shows that bandwidth heterogeneity can result in a poor utilization of access link bandwidth among high bandwidth peers especially when the fraction of high bandwidth peers is small. Setting the peer degree based on the bandwidth-degree condition (with a proper ratio) results in high utilization of

access link bandwidth among high bandwidth peers (>95%) with low variations (<3%) in all the above scenarios. The utilization of link bandwidth with bandwidth-degree condition is not shown in Figure 1 for clarity.

III. CONTENT DELIVERY IN PRIME

The content delivery mechanism in PRIME combines push reporting by parents with pull requesting by child peers. Each peer receives content from *all* of its parents and provides content to *all* of its child peers. The content is encoded with Multiple Description Coding (MDC) which enables each peer to maximize its delivered quality by pulling a proper number of descriptions. Each peer, as a parent, progressively reports its newly received packets to all of its child peers and as a child, periodically (*i.e.*, once per Δ second) requests an ordered list of packets from each one of its parents. Each parent peer delivers requested packets by each child peer in the provided order and at the rate that is determined by the congestion control mechanism. The requested packets from parent peers are determined by a *packet scheduling* mechanism at each child peer. The overall performance of content delivery depends on the collective behavior of the packet scheduling mechanism across all participating peers.

In the context of live P2P streaming applications, a new segment of length Δ is generated by the source every Δ seconds where a segment consists of a group of packets with consecutive timestamps ($[t_0, t_0 + \Delta]$) across all descriptions. To accommodate swarming, participating peers maintain a loosely synchronized playout time which is $\omega * \Delta$ seconds behind source's playout time. This provides roughly $\omega * \Delta$ seconds worth of content for swarming which has two implications: (i) each peer should buffer at least $\omega * \Delta$ seconds worth of content, and (ii) each packet should be delivered within $\omega * \Delta$ seconds from its generation time to ensure in-time delivery.

Suppose all connections have roughly the same bandwidth (*bw_{pf}*), then the amount of data that a child peer receives from each parent during an interval (Δ) can be estimated as $D = bw_{pf} * \Delta$. We call this volume of data a *data unit*. A data unit consists of several packets (possibly from different descriptions) that are selected by the packet scheduling mechanism at a child peer. When one (or multiple) parent(s) of a child peer does not have a useful data unit to offer during an interval, the child peer cannot fully utilize the bandwidth of the corresponding connection(s) and experiences *content bottleneck*.

The goal of the packet scheduling mechanism at individual peers is to maximize their delivered quality while minimizing their buffer requirement. Achieving these goals is the same as minimizing the probability of content bottleneck among participating peers which maximizes the utilization of the outgoing bandwidth among all peers and thus accommodates scalability. The probability of content bottleneck among peers depends on the availability of new data units at each parent peer which is determined by the global pattern of content delivery from the source to all peers in the overlay. Therefore, to design a content delivery mechanism, first we identify a global

pattern of content delivery that minimizes the probability of content bottleneck among peers. Then, we derive the required packet scheduling algorithm at individual peers that lead to the desired global pattern.

A. Organized View of a Random Mesh

To identify the desired global pattern of content delivery, we present an organized view of a randomly connected and directed mesh as shown in Figure 2. Towards this end, we define the distance of peer p from the source as the shortest path (in hops) from the source to peer p through the overlay. Given this definition, a group of peers that are exactly n hops away from source, can be grouped into *level n* .

Consider an overlay with P homogeneous peers where all peers have the same incoming and outgoing degree of deg and the source degree of deg_{src} . The organized view reveals three important properties of the overlay as follows [9]: (i) The population of peers at level n (or $pop(n)$) is limited to $pop(n) \leq deg_{src} * deg^{(n-1)}$, (ii) The number of levels, or *depth*, of such an overlay is limited to $log_{deg}(P/deg_{src}) \leq depth$, (iii) The probability of having a parent at level n is equal to $\frac{pop(n)}{P}$ for a given peer in the overlay. Typically, a peer in level n , except for peers in the bottom level, has a single parent in level $n - 1$, ($deg - 1$) parents in the same or lower levels, and deg child peers in level $n + 1$. Peers in the bottom level ($n = depth$) have a single parent in level $n - 1$, and deg child peers in the same or higher levels.

B. Global Pattern of Content Delivery

In this subsection, we derive the global pattern of content delivery for a single segment of content that minimizes the probability of content bottleneck. Consecutive segments of the stream can be delivered through the overlay using a roughly similar pattern. Intuitively, to minimize the number of intervals for delivery of a segment, first different data units of the segment should be rapidly delivered (or diffused) to different subset of peers. Then, participating peers can exchange (or swarm) their data units and contribute their outgoing bandwidth until each peer has a proper number of data units for the segment. The above observation motivates a two-phase approach for delivery of a segment as follows:

1) Diffusion Phase: Once a new segment becomes available at the source, peers in level 1 can collectively pull all data units of that segment during the next interval Δ . Then, peers in level 2 can collectively pull all data units of the new segment during the following interval and so on. Therefore, the fastest time for delivery of different data units of a segment to different peers in level i is $i * \Delta$ seconds. This implies that each peer in the system has at least one data unit of a new segment within $depth * \Delta$ seconds after it becomes available at the source.

To rapidly diffuse a new segment among peers in the overlay, all the connections between peers in level n ($n < depth$) to their child peers in level $n + 1$ should be exclusively used for the diffusion of new data units. These connections are called *diffusion connections* and the corresponding parents are called *diffusion parents*. Diffusion connections are shown with

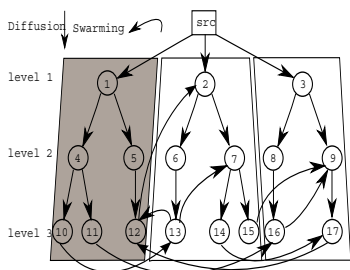


Fig. 2. Organized view of a mesh-based overlay with 17 peers

straight arrows in Figure 2. The number of diffusion connections into level n is less than or equal to the population of peers in level n (i.e., $deg_{src} * deg^{(n-1)}$) which is exponentially increasing with n .

The above pattern of content diffusion has the following implications: First, the diffusion phase takes exactly $depth$ intervals or $depth * \Delta$ seconds. Second, each peer p in level 1 as well as all of its descendant peers in a sub-tree rooted in p receive the same data unit of each segment during the diffusion phase of that segment, but at different intervals depending on their levels. Each such a sub-tree of peers that is rooted in a peer in level 1 is called a *diffusion sub-tree*. The number of diffusion subtrees in an overlay is equal to the population of peers in level 1, or deg_{src} . In Figure 2, one of the three diffusion sub-trees that is rooted at peer 1 is shaded. Third, when the bandwidth of a diffusion connection is less than bw_{pf} , all the downstream peers in the corresponding diffusion subtree experience content bottleneck during the diffusion phase.

2) Swarming Phase: At the end of the diffusion phase of a segment, all peers in the overlay have at least one data unit of that segment. During the swarming phase of a segment, participating peers pull the missing data units of the segment from their parents that are located in the same or lower levels. Therefore, all the connections from parent peers in level j to their child peers in the same or higher level i ($i \leq j$) are exclusively utilized for swarming. These connections are called *swarming connections* and shown with curly arrows in Figure 2. We also call their corresponding parents as *swarming parents*. Note that most of the swarming parents are located at the bottom level. This means that the outgoing bandwidth of peers at the bottom level is primarily utilized for the swarming of each segment.

We recall that all peers in the same diffusion sub-tree receive the same data unit of a segment during the diffusion phase. This implies that only those swarming parents that are located on different diffusion sub-trees can provide a new data unit to a child peer at the end of the diffusion phase. For example, in Figure 2, p_9 can obtain a new data unit from p_{15} but not from p_{16} . This simple condition enables us to determine whether each peer experiences a content bottleneck during the swarming phase based on the location of its swarming parents. If all swarming parents of a child peer are located at different diffusion sub-trees, the child peer can pull $(indeg_i - 1)$ new data units from all parents in a single swarming interval,

e.g., p_{12} in Figure 2. Otherwise, the child peer experiences a content bottleneck (e.g., p_9 in Figure 2) and thus requires more than one swarming interval to obtain the remaining data units. During these extra intervals, some of its swarming parents will obtain new data units of the target segment, and can pass them along. For example, p_{16} receives a new data unit from p_{11} after one interval and can pass it to p_9 in the next interval.

In a randomly connected overlay, the probability of experiencing a content bottleneck for a given peer during the swarming phase depends on the ratio of its incoming degree to the number of diffusion sub-trees with a unique data unit. For a given overlay, the minimum number of swarming intervals should be set such that nearly all peers can receive their maximum deliverable quality. We call this K_{min} . In Section IV, we show how the value of K_{min} is affected by other system parameters. In summary, the required buffering at individual peers or their relative playout delay compare to source (i.e., $\omega * \Delta$ seconds) should satisfy the following condition: $(depth + K_{min}) \leq \omega$.

C. Receiver-driven Packet Scheduling

The packet scheduling algorithm at individual peers should determine the requested packets from each parent such that its collective behavior lead to the desired global pattern of content delivery. Suppose that packet size is fixed and thus individual packets are identified by their timestamp and description id. Each peer can identify its diffusion parent(s) based on their distance from the source or their highest reported timestamp (ts_{max}). The packet scheduling mechanism is invoked once every Δ seconds by each peer and takes the following steps: *I) Quality Adaptation:* it compares the smoothed aggregate rate of data arrival from all parents with the target quality (i.e., the number of requested descriptions) and adjust the target quality accordingly.

II) Diffusion: it requests a random subset of newly available packets with timestamp within the following range $LAST(ts_{max}) < ts < ts_{max}$. $LAST(ts_{max})$ denotes the highest timestamp that was reported by parents during the last scheduling event. This ensures rapid diffusion of new packets towards lower levels of the overlay.

III) Swarming - Packet Selection: the scheduler determines the number of missing packets for all the swarming timestamps (i.e., packets with timestamp within the following range $t_p + \Delta < ts \leq LAST(ts_{max})$) by simply comparing the target quality with the number of unique packets (from different descriptions) that it has already received for each timestamp. t_p denotes peer's playout time. This step generates a list of timestamps for packets that could be pulled from swarming parents.

IV) Swarming - Packet Assignment: Given the average bandwidth from each parent, we can estimate the number of delivered packets from each parent during one interval $(\frac{ewma_{bw}(i) * \Delta}{PktSize})$. Then, the scheduler shuffles the list of required timestamps and sequentially examines each timestamp by taking two related actions:

- *Description Selection*: Determining a proper description such that the corresponding packet (timestamp, description) is available among parents but missing at the child peer, and
- *Parent Selection*: Assigning the identified packet to a parent that can provide it.

The description for a given timestamp could be determined either randomly or by choosing the rarest description from the useful descriptions among parents. The parent can be selected either randomly or based on the minimum ratio of its assigned packets to its total packet budget (*i.e.*, the fraction of its packet budget that has been assigned). This latter criteria tends to proportionally balance the assigned packets among parents during the scheduling process. These choices result in different variants of the scheduling algorithm based on the selection criteria and the ordering of (description or parent) selection. We examine these variants in Section IV-C.

D. Source Behavior

The maximum available quality in the system is limited by the number of descriptions that are delivered from the source to all the peers in level 1. This quality is determined by (i) the aggregate throughput from the source to all of its child peers, and (ii) the utilization of source's access link bandwidth. The aggregate throughput from the source depends on its outgoing bandwidth as well as its outgoing degree which should be determined by the bandwidth-degree condition. We introduce the term *diffusion rate* as the rate of delivery for new bits from source to level 1. Ideally, the diffusion rate should be equal to the aggregate throughput from the source and the number of copies among delivered packets to level 1 should be fairly even. Satisfying these two conditions at level 1 ensures proper behavior across other levels since the packets are simply multiplied by degree as they are pulled towards lower levels. In practice, the following two factors can reduce the diffusion rate or unbalance the number of copies for delivered packets: (i) the independent packet scheduling by peers in level 1, and (ii) the random loss of delivered packets to level 1.

The source is the only node in the system that is aware of delivered data units to different diffusion subtrees. Therefore, it can minimize the potential overlap among the delivered data unit to different diffusion subtrees. In PRIME, the source implements two related mechanisms to achieve this goal: First, it performs loss detection for delivered packets and keeps track of the number of actually delivered copies for each packet (*i.e.*, timestamp and description id). Second, any requested packet with timestamp ts that has already been delivered to other peers, is swapped with a rarest packet within a recent window $[ts-\Delta, ts]$ where $\Delta \gg RTT$, *i.e.*, the requested packet is swapped with another packet that has not been delivered at all or has the minimum number of delivered copies. Performing loss detection ensures that the packet swapping mechanism behaves properly.

IV. PERFORMANCE EVALUATION

We use ns simulations to examine the effect of various key factors on PRIME performance including: (i) peer connectivity, (ii) packet scheduling, (iii) peer population and (iv) source behavior. Due to the limited space, we only present a subset of our results in this paper. Extensive evaluations of PRIME can be found in the related technical report [10]. In our simulations, the physical topology is generated with Brite [11] using the following configuration parameters: 15 AS with 10 routers per AS in top-down mode and RED queue management at all routers. We use the following default settings in our simulations: $\Delta = 6$ seconds, the overlay is directed, all access links are symmetrical, the bandwidth-degree condition is satisfied, the delay on each access link is randomly selected between [5ms, 25ms], core links have high bandwidth (ranging from 4 to 10 Gbps) and thus all connections experience bottlenecks only on the access links. All connections are congestion controlled using RAP. Furthermore, each stream has 10 descriptions and all descriptions have the same constant bit rate of $C = 160$ kbps. Each peer simulates the streaming consumption of delivered content after $\omega * \Delta$ seconds startup delay. The following two scenarios are used as the *reference scenarios* in our evaluations: 200 homogeneous peers with (i) 700 kbps and (ii) 1.5 Mbps access link bandwidth.

Each simulation was run for 400 seconds. Our results represent the behavior of the system during the steady state after all peers have identified their parents and their pair-wise connections have reached their average bandwidth. Furthermore, we have repeated individual simulations over several overlays with different random seeds and the results have been similar. We also use the following methodology to decouple and separately quantify the impacts of bandwidth and content bottlenecks on delivered content from each parent. Each parent always sends packet to its child peers at the rate that is determined by a congestion controlled mechanism regardless of its useful content. At each packet transmission time to a particular child, if there is an outstanding list of requested packets from that child, the outgoing packet carries the first requested packet in the list. Otherwise, the parent sends an especially marked packet with the same size.

A. Peer Connectivity

Our goal is to examine how the connectivity of individual peers affect the performance of content delivery in PRIME. To minimize the effect of other factors on our evaluation in this subsection, we use the best performing packet scheduling mechanism and ensure that the delivered quality to level 1 is equal to the maximum required quality for the peer with the highest incoming bandwidth in each scenario.

The bandwidth-to-degree ratio is a key aspect of peer connectivity that determines the value of bandwidth-per-flow or *bw/pf*. We examine the impact of this ratio on the performance of content delivery in the two reference scenarios. Figure 3(a) depicts the percentage of peers that received at least 90% of the maximum deliverable quality (*i.e.*, $\frac{inbw}{C}$) as a function of peer degree. Note that changing peer degree directly affects

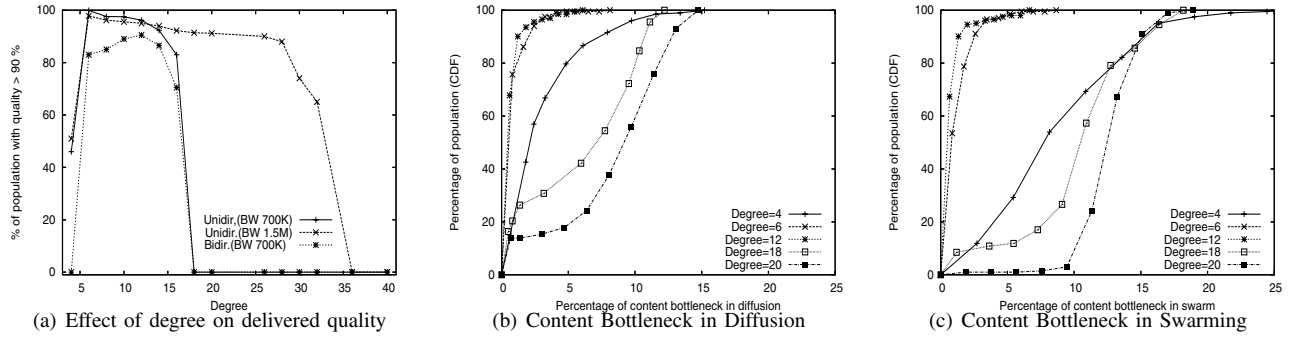


Fig. 3. Effect of peer connectivity on PRIME performance

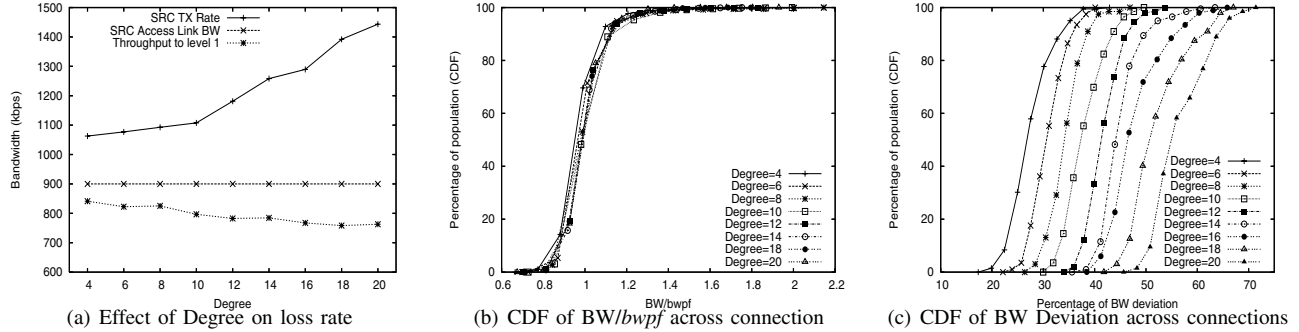


Fig. 4. Effect of *bwpf* on packet loss

the depth of the overlay. Therefore, for proper comparison, we keep the number of swarming intervals constant across these simulations ($K=3$) by setting the value of ω as follows: $\omega = depth + 3$. Figure 3(a) shows two interesting points: (i) in each scenario, there is a sweet range of peer degree over which a majority of peers receive a high quality stream, (ii) the proper range of peer degree has the same lower bound (degree = 6) in both scenarios but its upper bound depends on the bandwidth-degree ratio.

The poor performance of the system for small peer degrees (degree ≤ 4) is due to the limited diversity of swarming parents which leads to content bottleneck among participating peers. When peer degree is small, the number of diffusion subtrees will be small because of the bandwidth-degree condition. This in turn proportionally reduces the probability that the randomly selected swarming parents by each peer would be located on different diffusion subtrees and thus increases the probability of content bottleneck among peers regardless of peer bandwidth. The rapid drop in the delivered quality for large peer degrees is the result of significant increase in loss rate of individual connections which leads to a major drop in *bwpf*. Figure 3(a) clearly shows that the upper bound for the reference scenario with peer bandwidth 1.5 Mbps is almost twice the the upper bound for peer bandwidth 700 kbps. This demonstrates that the upper bound of the sweet range of peer degree is a function of loss rate rather than the peer degree. We examine the effect of loss rate for higher peer degrees in further details later in this section.

To verify our explanation, Figure 3(b) and 3(c) depict the distribution of content bottlenecks in the diffusion and swarming phases among participating peers with peer bandwidth 700

Kbps for a few peer degrees, respectively. The percentage of content bottleneck in the diffusion (or swarming) phase is the percentage of congestion controlled bandwidth from the diffusion (or swarming) parent(s) that can not be utilized for content delivery (i.e., the percentage of especially marked packets). Comparing these figures shows that the percentage of content bottleneck is clearly higher in the swarming phase across all degrees as we discussed in subsection III-B. Furthermore, as we increase the peer degree from 4 to 6, the percentage of content bottleneck in both phases significantly decreases. However, any further increase in peer degree (beyond 12) reverses this trend and rapidly increases the percentage of content bottleneck in both phases.

Loss Rate: To further examine the effect of packet loss on system behavior for large peer degrees, Figure 4(a) plots (from top to bottom) the aggregate transmission rate from a parent to all of its child peers, the parent's access link bandwidth and aggregate throughput to all of its child peers. The gap between the top two lines shows the bandwidth associated with lost packets at the outgoing link of the parent peer whereas the gap between the bottom two lines represents the bandwidth associated with lost packets at the incoming access link of all child peers, collectively. This figure shows that the aggregate throughput from a parent peer to all of its children rapidly drops with increasing peer degree. More interestingly, while losses mostly occur at the parent's outgoing link, a non-negligible fraction of losses also occur at the incoming link of child peers as well. This suggests that throughput of some connections are limited by the parent's outgoing link bandwidth while others are limited by the child's incoming link bandwidth. This may seem surprising because

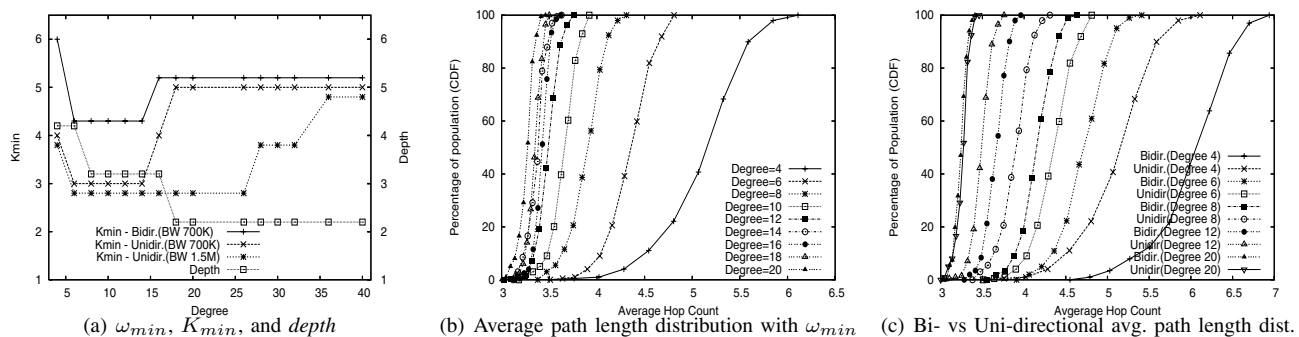


Fig. 5. Effect of peer connectivity on buffer requirement and pattern of delivery

the bandwidth-degree condition already limits individual connections' throughput at the parent's outgoing link.

To verify our hypothesis, we show the distribution of normalized average throughput (normalized by the corresponding bw_{pf}) and its deviation across all connections for different peer degrees in Figure 4(b) and 4(c), respectively. These two figures paint an insightful picture on how bandwidth dynamics affect the location of bottleneck for individual connections. As peer degree increases, the distribution of normalized average throughput across all connections does not change but the distribution of its deviation shifts towards higher values. In a nutshell, the larger deviations with larger peer degrees result in bottlenecks at both sender and receiver ends of individual connections. It is worth noting that session level simulators are unable to capture this important behavior.

Buffer Requirement: The poor performance outside the proper range of peer degree indicates that the number of swarming intervals is inadequate for the delivery of the required number of data units to most peers due to a content bottleneck. This raises the following question: "How many swarming intervals are required so that nearly all peers receive a high quality stream?". Figure 5(a) depicts the number of diffusion intervals (i.e., $depth$) and the minimum number of required swarming intervals ($K_{min} = \omega_{min} \cdot depth$) as a function of peer degree in both reference scenarios such that 90% of peers receive 90% of the maximum deliverable quality. As Figure 5(a) shows, the depth of the overlay is independent of the peer bandwidth and gradually decreases with peer degree in a step-like fashion. As degree increases, K_{min} initially decreases from 4 to its minimum value of 3 intervals within the proper range of peer degree. However, further increase of peer degree beyond a threshold results in a linear increase in K_{min} until it reaches the maximum value of 5. In essence, this figure represents the minimum buffer requirement at each peer in terms of number of intervals as a function of peer degree (i.e., $\omega_{min} = depth + K_{min}$). It also illustrates the direct relationship between K_{min} and bw_{pf} for different peer degrees.

Pattern of Delivery: To study the effect of peer degree on the pattern of content delivery, we examine the following question: "How does the distribution of the average path length (in hops) among delivered packets to individual peers change as peer degree increases (i.e., the overlay becomes more

connected)?" Figure 5(b) presents this distribution for several peer degrees in the reference scenario with peer bandwidth 700 Kbps when the number of swarming intervals is equal to K_{min} . This figure reveals the following two important changes in the average path length among peers as peer degree increases: (i) the average path length to individual peers monotonically decreases with peer degree primarily due to the decrease in overlay depth, (ii) the distribution of average path length among peers becomes more homogeneous due to the increase in the diversity of swarming parents which in turn evens out the probability of content bottleneck among peers. The increasing homogeneity of average path length with peer degree also implies that lost packets are requested from the same parent during the following swarming interval(s) rather than through a longer path from other swarming parents.

Bi- vs Uni-directional Connectivity: Maintaining uni- vs bi-directional connections between peers affect the nature of connectivity among peers and thus could impact the performance of content delivery mechanism. To investigate this issue, we examine the reference scenario with 700 Kbps bandwidth but enforce bi-directional connections among peers. The percentage of peers that receive 90% of the maximum deliverable quality as a function of peer degree is shown in Figure 3(a) when the number of swarming intervals is 3. This figure shows that the percentage of peers with high quality in a bi-directional overlay is 10%-20% lower compared to the unidirectional overlay over the sweet range of peer degree. Figure 5(a) also shows the value of K_{min} for these bidirectional overlays. This figure indicates that bi-directional overlays require at least one extra swarming interval for peer degrees between 4 and 16. To explain this result, we note that bi-directional connections reduce the number of swarming shortcuts among diffusion subtrees and thus increase the percentage of content bottleneck. More specifically, for each diffusion connection from a parent to a child, there is a swarming connection in the reverse direction that connects two peers within the same diffusion subtree which is not an effective swarming shortcut.

Figure 5(c) depicts the distribution of average path length for the above bidirectional overlays, as well as the corresponding unidirectional overlays that we already presented in Figure 5(b) for easy comparison. This figure indicates that the distribution of average path length over the bi-directional overlay is around one hop (20%) longer than the uni-directional overlay

for peer degree of 4. However, the difference in path lengths between bi- and uni-directional overlays rapidly decreases with peer degree. Note that the number of ineffective swarming shortcuts is roughly equal to the number of peers. Therefore, for a fixed population, as the peer degree increases, the extra connections must establish useful swarming shortcuts. This in turn improves the diversity of swarming parents and reduces the average hop count (and its deviations) for individual peers as shown in Figure 5(c).

B. Bandwidth Heterogeneity

To investigate the effect of bandwidth heterogeneity, we consider the reference scenario with peer bandwidth 1.5 Mbps (bw_h) and reduce the link bandwidth for a fraction of peers to bw_l . As we showed in Section II, the bandwidth-degree condition ensures that the utilization of access link remains high when peers have heterogeneous link bandwidth. The probability of content bottleneck for low bandwidth peers in heterogeneous scenarios should be lower since the available quality among their swarming parents could be higher. Therefore, we focus on the high bandwidth peers. The first question is: “How the delivered quality and buffer requirements of high bandwidth peers is affected by the degree of bandwidth heterogeneity (i.e., $\frac{bw_h}{bw_l}$) and the percentage of low bandwidth peers?”.

Figures 6(a) and 6(b) show the distribution of content bottleneck among high bandwidth peers (1.5 Mbps) with different percentage of low bandwidth peers (1 Mbps) for diffusion and swarming phases, respectively. We use the same bandwidth-to-degree ratio ($\frac{1.5Mbps}{12} = \frac{1Mbps}{8}$) in different scenarios for a fair comparison. This in turn implies that the depth of the overlay increases as the number of high bandwidth peers decreases. These figures show that the percentage of high bandwidth peers has a minor impact on the content bottleneck in both phases. The minor increase in content bottleneck during the diffusion phase with the small percentage of high bandwidth peers (in Figure 6(a)) is due to the decrease in the total number of connections and the resulting increase in the overlay depth. In Figure 6(b), the minor increase in content bottleneck during the swarming phase when a small percentage of peers have high bandwidth can be explained as follows:

the percentage of content bottleneck at each peer depends on the aggregate available content among its swarming parents. When the percentage of high bandwidth peers is small, a larger fraction of their swarming parents consists of low bandwidth peers. This in turn reduces the aggregate available quality among their swarming parents and increases the probability of content bottleneck.

Location of High Bandwidth Peers: Another important question in an overlay with heterogeneous peers is: “Whether the location of high bandwidth peers in the overlay affects the percentage of content bottleneck among them?”. To examine this issue, we explore a heterogeneous scenario where only 10% of peers have link bandwidth of 1 Mbps and the remaining peers have link bandwidth of 1.5 Mbps. We modify the overlay construction mechanism to only place high bandwidth peers either in the top level (as source’s children) or at the bottom level. Figures 6(a) and 6(b) show the percentage of content bottleneck for these two cases (labeled as “top” and “bottom”) for comparison with previous scenarios. Placing the high bandwidth peers in non-bottom levels reduces the depth of the overlay and thus the minimum required number of diffusion intervals. However, it also reduces the connectivity among the diffusion subtrees and thus equally increases the minimum number of required swarming intervals (K_{min}). In contrast, placing high bandwidth peers at the bottom level slightly increases overlay depth and thus increases the required number of diffusion intervals. However, this effect is compensated by the higher connectivity among the diffusion subtrees which decreases the minimum number of required swarming intervals. *In summary, the location of high bandwidth peers does not have a significant impact on the minimum buffer requirement (i.e., ω).*

C. Packet Scheduling

To study the effect of packet scheduling algorithms on the performance of the PRIME protocol, we consider the reference scenario with link bandwidth 700 Kbps and assume that all peers use the same packet scheduling algorithm. We examine six different scheduling algorithms that represent various order of selection and different criteria for selecting description of a given timestamp (random or rarest) or assigning a packet to a parent (random or parent with minimum bandwidth utilization). Figure 7(a) depicts the percentage of peers that receive 90% of the maximum deliverable quality as a function of peer degree for these six packet scheduling algorithms where $\omega = depth + 3$. This figure illustrates two interesting points: First, except for two scheduling algorithms that randomly select the parent, the performance of other algorithms is very similar within the proper range of peer degree. This implies that neither the criteria for selecting the description of a packet nor the order of selection (between description and parent) significantly affects the performance. Second, the percentage of peers that receive a high quality stream in the two low-performing algorithms is very similar, and roughly 20% lower than other algorithms within the proper range of peer degree. Closer examination of K_{min} for these two scheduling schemes

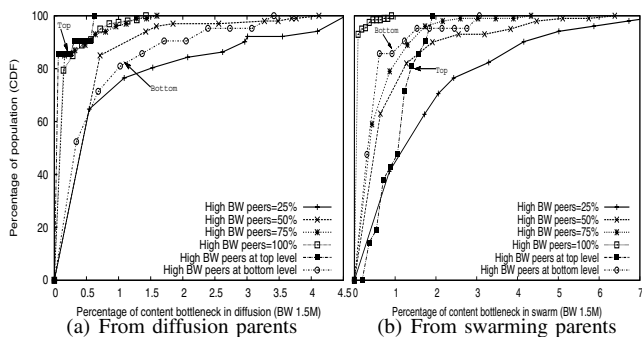


Fig. 6. Content bottleneck among high BW peers in heterogeneous scenarios

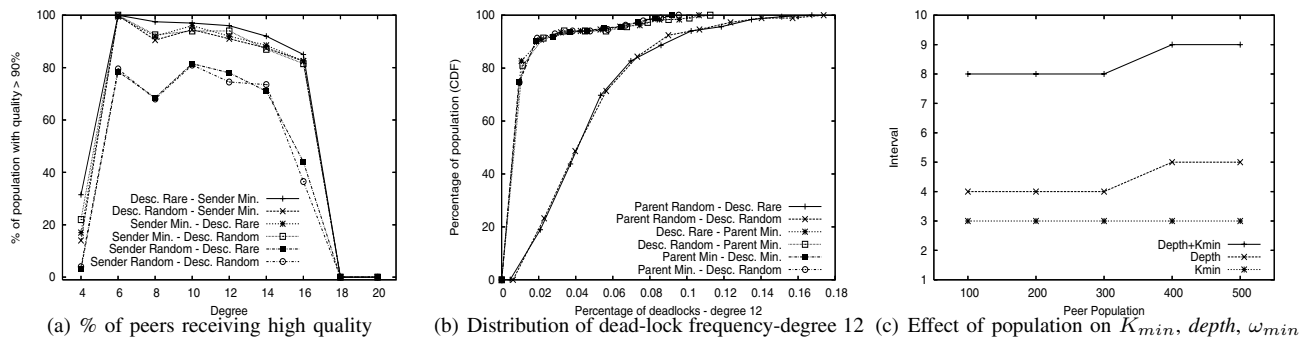


Fig. 7. The effect of packet scheduling and scalability on PRIME

revealed that their K_{min} value is always one interval larger than other schemes in a comparable scenario. Intuitively, those scheduling schemes that assign a packet to a random parent are more likely to experience content bottleneck due to the higher frequency of *deadlock* during parent selection. Here by *deadlock*, we are referring to an event when a required packet is available among some parents but it can not be requested since the bandwidth budget of those parents are fully used for delivery of other packets. To verify this hypothesis, Figure 7(b) depicts the distribution of deadlock frequency (*i.e.*, the fraction of packets whose scheduling leads to a deadlock) among peers in the above scenario when peer degree is 12. Figure 7(b) clearly shows that the median deadlock frequency is roughly four times higher for scheduling algorithms that use random parent selection. In the random parent selection strategy all unique packets of a parent may not be requested. Therefore, a fraction of parent’s bandwidth budget is used for the delivery of packets that are available at other parents.

D. Peer Population

We finally examine the scalability of PRIME protocol by addressing the following question: “How does the delivered quality and buffer requirement at individual peers change with peer population?”. Figure 7(c) shows the duration of diffusion phase (or overlay depth) and the minimum duration of swarming phase (K_{min}) and the minimum buffer requirement (or ω) as a function of peer population in the reference scenario with link bandwidth 700 Kbps and peer degree 6. This figure provides a good evidence on the scalability of PRIME protocol. As the peer population increases, overlay depth slowly grows but the duration of the swarming phase (with a proper peer degree) remains constant. To explain this, we note that increasing peer population does not affect the number of diffusion subtrees. Therefore, the diversity of swarming parents for individual peers does not change with peer population. *This result indicates that within the proper range of peer degree, PRIME can effectively utilize available resources in the system and accommodate scalability if the buffer size is sufficiently large.*

V. CONCLUSIONS AND FUTURE WORK

In this paper, we presented PRIME, the first mesh-based P2P streaming mechanism for live content that can effectively incorporate swarming content delivery. We argued that the

bandwidth-degree condition should be satisfied by the overlay construction mechanism in order to minimize the bandwidth bottleneck among participating peers. We also derived the pattern of content delivery that can incorporate swarming in order to effectively utilize the outgoing bandwidth of participating peers and thus minimize the content bottleneck in the system. This in turn led us to the desired packet scheduling algorithm at individual peers. Through extensive ns simulations, we examined the effect of key factors on PRIME performance and identified a few fundamental design tradeoffs.

We are currently extending this work along several dimensions. First, we are examining the effect of churn on PRIME performance, in particular on ensuring the bandwidth-degree condition. Second, we are evaluating PRIME performance in scenarios where the distribution of outgoing bandwidth is very skewed or in the presence of free-loaders [12]. Third, we also use PRIME to conduct systematic comparison between tree-based and mesh-based P2P streaming mechanism [2]. Fourth, we have prototyped PRIME and currently conducting experiments over PlanetLab. Finally, we plan to incorporate the notion of “contribution awareness” into PRIME.

REFERENCES

- [1] V. N. Padmanabhan, H. J. Wang, and P. A. Chou, “Resilient peer-to-peer streaming,” in *ICNP*, 2003.
- [2] N. Magharei, R. Rejaie, and Y. Guo, “Mesh or Multiple-Tree: A Comparative Study of P2P Live Streaming Services,” in *INFOCOM*, 2007.
- [3] B. Cohen, “Bittorrent.” [Online]. Available: <http://www.bittorrent.com>
- [4] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, “Bullet: High bandwidth data dissemination using an overlay mesh,” in *SOSP*, 2003.
- [5] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, “Coolstreaming: A data-driven overlay network for live media streaming,” in *INFOCOM*, 2005.
- [6] S. Xie, B. Li, G. Keung, and X. Zhang, “Large Scale Peer-to-Peer Live Video Streaming: Theory and Practice,” Tech. Rep., 2006.
- [7] V. Venkataraman, K. Yoshida, and P. Francis, “Chunkyspread: Heterogeneous Unstructured End System Multicast,” in *ICNP*, 2006.
- [8] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. Mohr, “Chainsaw: Eliminating Trees from Overlay Multicast,” in *IPTPS*, 2005.
- [9] N. Magharei and R. Rejaie, “Understanding Mesh-based Peer-to-Peer Streaming,” in *NOSSDAV*, 2006.
- [10] N. Magharei and R. Rejaie, “Peer-to-Peer receiver-driven mesh-based streaming: Design and Evaluation,” Tech. Rep. CIS-TR-06-05, 2006. [Online]. Available: <http://mirage.cs.uoregon.edu/pub/tr06-05.pdf>
- [11] A. Medina, A. Lakhina, I. Matta, and J. Byers, “BRITE: An Approach to Universal Topology Generation,” in *International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems*, 2001.
- [12] N. Magharei, Y. Guo, and R. Rejaie, “Issues in Offering Live P2P Streaming Service to Residential Users,” in *IEEE CCNC*, 2007.