PRIME: P2P Receiver-drIven MEsh-based Streaming

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

Abstract— **Peer-to-peer (P2P) streaming mechanisms support one-to-many delivery of streaming content without any especial support from the network. The goal of these mechanisms is to maximize delivered quality to individual peers with minimum buffer requirement at each peer in a scalable fashion. However, existing P2P streaming schemes can not achieve this goal because of their inability to effectively utilize the outgoing bandwidth of participating peers.**

This paper presents a new approach to live P2P streaming, called P2P Receiver-drIven MEsh-based Streaming, or PRIME. In PRIME, participating peers form a randomly connected and directed mesh-based overlay and incorporate a swarm-like delivery to effectively contribute their outgoing bandwidth. We present the design of PRIME and conduct detailed simulation-based evaluations. In particular, we illustrate the effect of per-peer packet scheduling, overlay properties, source behavior and peer population on system performance. Our evaluations shed an insightful light on fundamental tradeoffs in design of mesh-based P2P streaming mechanisms.

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 over the Internet without any special support from the network, called *P2P streaming*. The goal of P2P streaming mechanisms is to maximize delivered quality to individual peers in a scalable fashion. A P2P streaming mechanism can truly scale if a majority of participating peers can effectively participate in content delivery and contribute their outgoing bandwidth. This depends not only on the properties of the overlay topology but also on the overall pattern of content delivery through the overlay.

The traditional approach to P2P streaming is to organize participating peers into a single tree-structured overlay over which the content is pushed from the source towards all peers (*e.g.*, [1]). This approach suffers from two fundamental limitations: *(i)* the delivered quality to individual peers is limited by the minimum bandwidth among the upstream connections from the source. This problem is further aggravated by the heterogeneity and asymmetry of access link bandwidth among peers. *(ii)* more importantly, the content delivery mechanism can not utilize the outgoing bandwidth of a large fraction of peers that are leaves in the tree. An extension of tree-based approach organizes participating peers into multiple diverse trees [2], [3]. Each description of a Multiple Description Encoded (MDC) stream is pushed through one of the trees. This multiple-tree approach can utilize the outgoing bandwidth of participating peers more effectively. However, the limited available bandwidth to individual peers through each tree coupled with the static mapping of descriptions among trees limit the delivered quality to individual peers.

Recently, a new approach has gained popularity where participating peers form a mesh-based overlay and incorporate a swarm-like content delivery to effectively utilize outgoing bandwidth of participating peers [4]. File swarming mechanisms (*e.g.*, [5]) can leverage the availability of the entire file to distribute its pieces among different peers which enables them to actively contribute their outgoing bandwidth. However, incorporating swarm-like delivery into live P2P streaming applications is challenging for two reasons: *(i)* accommodating the streaming constraint for in-time delivery of individual packets is difficult, and *(ii)* the limited availability of future content in live streaming could limit the ability of the swarming mechanism to effectively utilize outgoing bandwidth of participating peers. A couple of recent studies have proposed a mesh-based P2P streaming mechanism and showed its feasibility for live streaming through experiment [4] or limited simulation-based evaluations [6]. Some studies have also proposed to use BitTorrent with minor modifications [7]. However, to our knowledge, none of the previous studies have clearly shown how (and under what conditions) their proposed approach can satisfy the above challenges for delivery of live streaming content. In a nutshell, some basic issues about P2P mesh-based streaming of live content have remained unanswered such as: How does the performance of these mechanisms change with key parameters such as heterogeneity of peer bandwidth, source bandwidth, peer degree, or packets scheduling? What are the performance bottlenecks and key design tradeoffs in mesh-based P2P streaming mechanisms? How well does this approach scale?

This paper presents a new mesh-based P2P streaming for scalable delivery of live multimedia content, called P2P Receiver-drIven MEsh-based Streaming, or PRIME. In PRIME, participating peers form a directed and randomly connected mesh and incorporate a swarm-like content delivery. Therefore, PRIME is able to effectively utilize the outgoing bandwidth of most participating peers and maximize their delivered quality with minimum buffer requirement at each peer. We present the overlay construction in PRIME and derive the require condition for incoming/outgoing degree of individual peers that maximizes their aggregate bandwidth. Then, we illustrate how a swarm-like delivery mechanism can be effectively incorporated in a mesh-based P2P streaming of live content and explain the packet scheduling mechanism and source functionality in PRIME. We present a new evaluation methodology for mesh-based P2P streaming and examine the effect of the following issues on PRIME performance through detailed ns simulations: peer connectivities and bandwidth heterogeneity, packet scheduling, source behavior and peer population. Our results not only reveal a few fundamental

design tradeoffs, inherent performance limitations and the relationship among key parameters but it also sheds an insightful light on the dynamics of content delivery in these systems.

In our earlier paper [8], we identified two main performance bottlenecks in mesh-based P2P streaming mechanisms and devised a global pattern of content delivery that can minimize these bottlenecks. This paper builds and significantly expands on our earlier paper. More specifically, we incorporate the proposed pattern of content delivery into a specific protocol, namely PRIME, evaluate its performance over a wide range of scenarios to identify key design tradeoffs and performance limitations as well as their underlying causes.

The rest of this paper is organized as follows: Section II and III describe overlay construction and content delivery components of PRIME protocol, respectively. In Section IV, we present our evaluation methodology and simulation-based performance evaluations. Section V concludes the paper.

II. Overlay Construction in PRIME

In PRIME, participating peers form a *randomly* connected and *directed* mesh that is used for content delivery to individual peers. Except for the source, each peer in the overlay has multiple parents and multiple child peers. Connections are established by child peers where each child maintains a sufficient number of parents whose aggregate bandwidth can fully utilize the child's incoming link. Toward this end, each peer contacts a bootstrapping node to learn about a random subset of participating peers in a demand-driven fashion. Such a mesh-based overlay is easy to maintain and is very resilient to churn. Furthermore, connections from different parents to each child peer are likely to have diverse paths, which in turn reduces the probability of a shared bottleneck.

Proper Incoming/Outgoing Peer Degree: The aggregate bandwidth to each child peer in a randomly connected mesh-based overlay depends not only on its own in-degree but also the out-degree of other participating peers. Suppose that congestion occurs only at the edge of the network, *i.e.*, the incoming/outgoing access links of participating peers. The average bandwidth for a connection between parent i to child peer j can be roughly estimated with $MIN(\frac{outbw_i}{outdeg_i}, \frac{inbw_j}{indeg_j})$ where $outbw_i$, $outdeg_i$, $inbw_j$, $indeg_j$ denote the outgoing bandwidth and outgoing degree of peer i, and incoming bandwidth and incoming degree or peer j, respectively. If the first term is smaller, the outgoing bandwidth of the parent peer is the bottleneck and thus the child's incoming access link may not be fully utilized. In contrast, if the second term is smaller, the bottleneck is at the incoming link of the child peer and the parent's access link may not be fully utilized. This observation suggests that to maximize the aggregate bandwidth to each individual peer in a randomly connected overlay the same bandwidth to degree ratio should be used for the outgoing and incoming links of *all* participating peers. More specifically, the following condition must be satisfied for any peer i and peer j: $bwpf = \frac{outbw_i}{outdeg_i} = \frac{inbw_j}{indeg_j}$

This constant ratio presents the average bandwidth of individual connections in the overlay and thus it is called *bandwidth-per-flow*, or $bwpf$. We call this the *bandwidth-degree condition* which implies that all connections in the overlay should have roughly the same bandwidth. $bwpf$ is indeed an important property of the system because it directly translates the (potentially heterogeneous and asymmetric) incoming and outgoing access link bandwidths of participating peers (and the source) to their incoming and outgoing degrees, respectively.

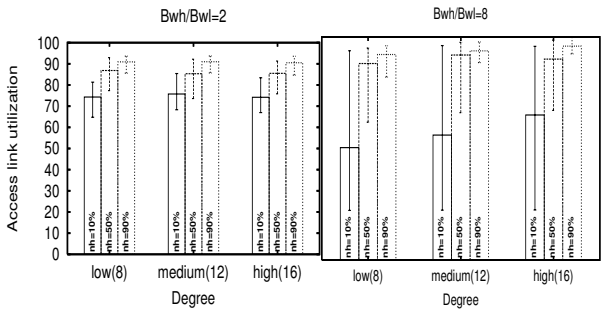


Fig. 1. Access link utilization

To examine the effect of enforcing the bandwidth-degree condition, we conduct ns simulation where 200 peers with heterogeneous access link bandwidth (bw_h and bw_l) form a random mesh with fixed degree. All connections are congestion controlled (RAP [9]). Figures II and II depict the average utilization of incoming access link bandwidth and its variations (as bar) among high bandwidth peers for two degrees of bandwidth heterogeneity (*i.e.*, $\frac{bw_h}{bw_l}$) 2 and 8, respectively. We also examined each scenario with three different degrees (*i.e.*, 8, 12 and 16) and different percentage of high bandwidth peers (n_h) for each degree. Across all these simulations low bandwidth peers always achieve a high access link utilization. This figure simply shows bandwidth heterogeneity can lead to poor utilization of access link bandwidth among high bandwidth peers. specially when the fraction of high bandwidth peer is small. Setting the peer degree based on the bandwidth-degree condition (with a proper ratio) results in high utilization of access link bandwidth for high bandwidth peers (average $>95\%$) with low variations ($<3\%$) in all the above scenarios (it is not shown in the figures).

In practice, some connections might experience bottleneck in the core rather than the edge of the network. This may affect the utilization of access link bandwidth for the child peers that receive content through these connections. This problem can be addressed by *(i)* allowing child peers with low in-bandwidth utilization to have extra parent peers and *(ii)* allowing parent peers with poor out-bandwidth utilization to accept extra child peers beyond the limit that is specified by the bandwidth-degree condition.

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 new packets to all of its child peers, and as a child, periodically (*i.e.*, once per Δ) requests an ordered list of packets from

its parents. Each parent peer delivers requested packets in the provided order and at the rate that is determined by the congestion control mechanism. The requested packets 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+Δ]) across all descriptions. To accommodate swarming, participating peers maintain a loosely synchronized playout time which is ω*Δ second behind source's playout time. This provides roughly ω*Δ seconds worth of content for swarming. It also implies individual peers must buffer at least ω*Δ seconds of content and each packet must be delivered within ω*Δ second from its generation time for in-time delivery of required packets.

Suppose all connections have roughly the same bandwidth ($bwpf$), then the amount of data that a child peer receives from each parent during an interval (Δ) can be estimated as $D = bwpf$*Δ. We call this 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 experiences *content bottleneck* and cannot fully utilize its access link bandwidth.

The goal of the packet scheduling mechanism at individual peers is to maximize their delivered quality while minimizing their buffer requirement. Achieving this goal is the same as minimizing the percentage of content bottleneck among participating peers which maximizes the utilization of the outgoing bandwidth among all peers (*i.e.*, self-scaling). The percentage 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 present a global pattern of content delivery that can achieve the above design goals. Then, we derive the required packet scheduling schemes at individual peers to achieve the desired global pattern.

A. Organized View of the Overlay Mesh

To identify the global pattern of content delivery over a mesh-based overlay, we present an organized view of a randomly connected, directed mesh. Towards this end, we define the distance of a peer p from the source as the shortest path (in hops) from the source to peer p through the connections in the overlay. Given this definition, peers in the overlay can be organized into separate *levels* (as shown in Figure III-B) based on their distance from source where level n consists of all peers that are exactly n hops away from source.

Consider the overlay consists of P homogeneous peers with the same in- and out-degree of deg and the source degree of deg_{src}. This organized view reveals three important properties of the overlay as follows [8]: *(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 can be approximated as $log_{deg}(P/deg_{src}) \leq depth$, *(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, each 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 or $n+1$. 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:

We describe the global pattern of content delivery for a single segment of content. Consecutive segments of the stream can be pipelined through the overlay by sequentially following 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 a 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 the new 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 all data units of a segment to different peers in level i is i*Δ seconds. This implies that each peer in the system has at least one data unit of the segment within $depth$*Δ seconds of it becoming available at the source.

To rapidly diffuse a new segment towards peers in lower levels, all the connections between peers in level n ($n<depth$) to their child peers in level $n+1$ should be exclusively used for diffusion of new data units. These connections are called *diffusion connections* and the corresponding parents are called *diffusion parents*. Diffusion connections are shown with straight arrows in Figure III-B. The number of diffusion connections into level n is at least equal to the population of peers in level n (*i.e.*, $deg_{src} * deg^{(n-1)}$) which is exponentially increasing with n.

During the diffusion phase of a segment, each peer p pulls a new data unit of the segment from its diffusion parent during an interval. During the next interval, the new data unit is pulled by all of p's child peers. This pattern of content diffusion has the following implications: First, the diffusion phase takes exactly $depth$ intervals. Second, each peer p in level 1 as well as all the peers in a sub-tree rooted in p receive the same data unit of each segment during their diffusion phase, but at different intervals depending on their levels. Each such sub-tree of peers 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}. Figure III-B shows a single diffusion sub-tree rooted at peer 1. Third, when the bandwidth of a diffusion connection is less than $bwpf$, all the downstream peers in the corresponding diffusion subtree experience a content bottleneck during the diffusion phase.

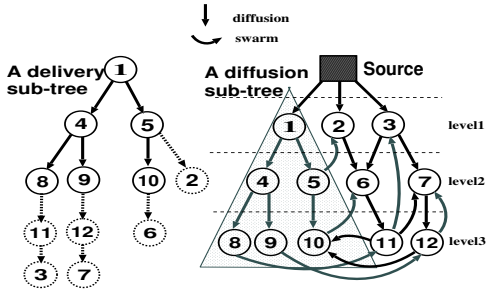


Fig. 2. Organized view of a mesh-based overlay with 12 peers

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 the 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 level i ($i \leq j$) are exclusively utilized for swarming. We call these *swarming connections*, and the corresponding parents are called *swarming parents*. These connections are shown with the curly arrows in Figure III-B. Note that most of the swarming connections are from peers in the bottom level to their child peers in same or higher levels This means that the outgoing bandwidth of peers in the bottom level is primarily utilized during the swarming phase of a segment.

We recall that all peers in the same diffusion sub-tree receive the same data unit. This implies that only a swarming parent that is located on a different diffusion sub-tree can provide a new data unit to a child peer at the end of the diffusion phase. For example, in Figure III-B, p_7 can effectively obtain a new data unit from p_{11} but not from p_12. This simple condition enables us to determine whether each peer experiences a content bottleneck during the swarming phase or not 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 interval, *e.g.*, p_{10} in Figure III-B. However, if two or more parents are located on the same diffusion sub-tree (or the subtree where the child peer is located), the child peer experiences a content bottleneck, *e.g.*, p_7 in Figure III-B. In such circumstances, a child peer requires more than one swarming interval to obtain its remaining ($indeg_i$-1) data units. During these extra intervals, some of its swarming parents will obtain new data units of the target segment, and can pass them to this child peer. For example, p_7 can receive a new data unit from p_{12} after one interval. Figure III-B shows the complete pattern for delivery of a single data unit as a tree that we call a *delivery sub-tree*.

In a randomly connected overlay, the probability of experiencing a content bottleneck among peers during the swarming phase depends on the ratio of the incoming degree of a given peer to the number of diffusion sub-trees with a unique data unit. For a given overlay, the minimum number of swarming intervals (k_{min}) should be determined such that nearly all peers can receive their maximum deliverable quality. This means that the required buffering intervals (ω) at individual peers should satisfy the following condition ($depth$+k_{min})$\leq\omega$.

C. 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, collectively. This quality is determined by *(i)* the aggregate throughput from the source to all of its child peers, and *(ii)* the utilization of its access link. The aggregate throughput from the source depends on its outgoing bandwidth coupled with its out degree which is determined by the bandwidth-degree conditions. 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 propagate through the levels. In practice, the following two factors can reduce the diffusion rate or skew 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.

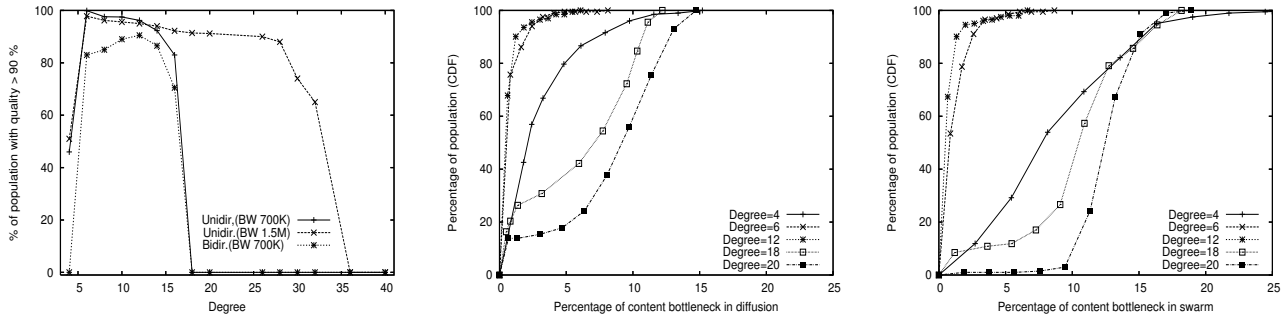
The source is the only common node among different diffusion subtrees. Therefore, it can minimize the overlap among the delivered data unit to different diffusion subtrees. Toward this end, the source implements two related mechanisms: First, it performs loss detection for delivered packets and keeps track of the number of actually delivered copies for each packet (*i.e.*, timestamp and description id). Second, any requested packet with timestamp of ts that has already been delivered, is swapped by a packet that has not been delivered (or the packet with the minimum number of delivered copies) within this window [ts-Δ, ts] ($\Delta >> RTT$), *i.e.*, swapping with the rarest packet. Performing loss detection ensures that the packet swapping mechanism behaves properly as shown in Section IV-C.

D. Receiver-driven Packet Scheduling

The per-peer packet scheduling at individual peers should behave such that its collective effect lead to the desired pattern of content delivery. Each packet is identified by its timestamp and description id. Note that the diffusion parent(s) can be easily identified based on its distance from the source or its highest reported timestamp (t_{max}) among its available packets. The packet scheduling mechanism is invoked once every Δ seconds and takes the following steps:

I) Diffusion: it requests a collection of packets with highest timestamps from its diffusion parent(s) to fully utilize its bandwidth.

II) Swarming, Packet Selection: the scheduler determines the number of missing packets for all the swarming timestamps (*i.e.*, t_p+Δ<$ts \leq LAST(ts_{max})$) by simply comparing the target quality with the number of unique packets (from different descriptions) that has already been received for that timestamp. $LAST(ts_{max})$ denotes the highest timestamp that was reported during the last scheduling event from diffusion parent(s). t_p denotes the peer's playout time. This step generates a list of timestamps for required packets.



(a) Effect of *bwpf* on System Performance (b) Content Bottleneck in Diffusion (c) Content Bottleneck in Swarming

Fig. 3. Effect of various *bwpf*s on overall performance

III) Swarming, Packet Assignment: Given the average bandwidth from each parent, we can estimate the packet budget for 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*: Determine a proper description such that the corresponding packets (timestamp, description) is one of the useful packets among parents, and
- *Parent Selection*: Assign the packet to a parent that can provide it.

The description of a packet for a given timestamp could be selected 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 packet to its packet budget (*i.e.*, the fraction of its packet budget that has been assigned). This latter approach tends to proportionally balance the assigned packets among parents. These choices result in different variants of the scheduling mechanism depending on the criteria and ordering for description or parent selection. We examine these variants in Section IV-B.

IV. Performance Evaluation

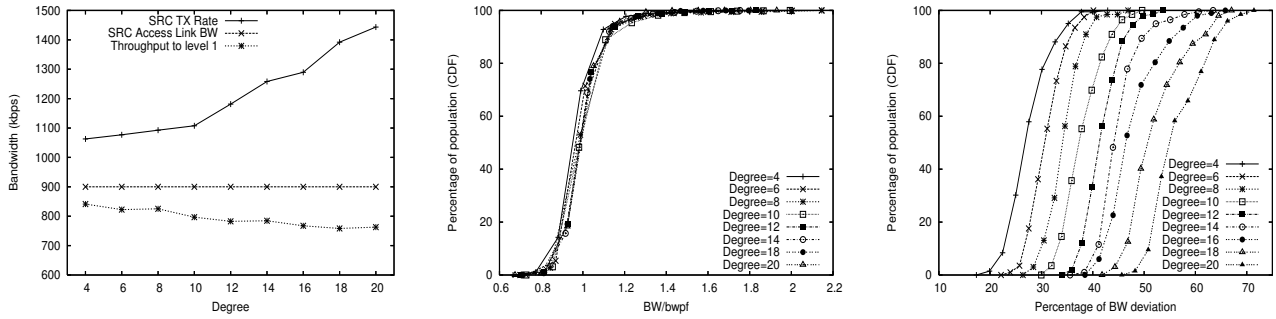
We use *ns* simulations to examine the effect of the following key components on PRIME performance: *(i)* Peer connectivities, *(ii)* Per-peer packet scheduling, *(iii)*Source behavior, *(iv)* Peer population. Using a packet level simulator has two important benefits compared to real world experiments or session level simulator: *(i)* it enables us to effectively control the degree of bandwidth heterogeneity and the location of bottlenecks, and *(ii)* it allows us to properly examine the effect of packet level dynamics and packet losses. In our simulations, the physical topology is generated with Brite, using the following configuration parameters: 15 AS with 10 routers per AS in top-down mode and RED queue management at all routers. The delay on each access link is randomly selected between [5ms, 25ms]. Core links have high bandwidth and thus all connections experience bottlenecks only on the access links. 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 our simulations over several random overlays with different seeds and the results were similar. All pairwise data connections use the RAP congestion control mechanism [9]. Δ = 6 seconds In our simulations Δ = 6 seconds and we assume that all descriptions have the same constant bit rate of C = 160 kbps. Each peer emulates the streaming consumption of delivered content (after $\omega * \Delta$ seconds delay) to simulate streaming delivery of live content with $\omega * \Delta$ seconds buffering.

We make the following assumptions in our simulations: the overlay is directed, the bandwidth-degree condition is satisfied, all access links are symmetrical. We do not model churn in our simulations since the dynamics of content delivery on the static overlay is sufficiently challenging and should be studied first. The following two overlays are used as the *reference scenarios* in our simulations: 200 homogeneous peers with *(i)* 700 kbps and *(ii)* 1.5 Mbps access link bandwidth. We also use the following methodology to decouple and separately quantify the respective impacts of bandwidth and content bottlenecks on delivered quality from each parent. We assume that each parent sends packets to each one of 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 certain child, if there is an outstanding list of requested packets from that child, the outgoing packet carries the first packet in the list. Otherwise, the parent sends a specially marked packet with the same size.

A. Peer Connectivities

Our goal is to examine how the connectivity of individual peers affect the performance of content delivery in PRIME. To isolate the effect of other factors on our evaluation, 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 highest incoming bandwidth in each scenario.

1) Bandwidth-to-Degree Ratio: The bandwidth-to-degree ratio is a key aspect of peer connectivity that determines the value of bandwidth-per-flow or $bwpf$. We examine the impact of this ratio on the performance of content delivery in the two reference scenarios with 700 kbps and 1.5 Mbps bandwidth. 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



(a) Effect of Degree on loss rate

(b) CDF of ratio of BW to the *bwpf* for each connection

(c) CDF of BW Deviation

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

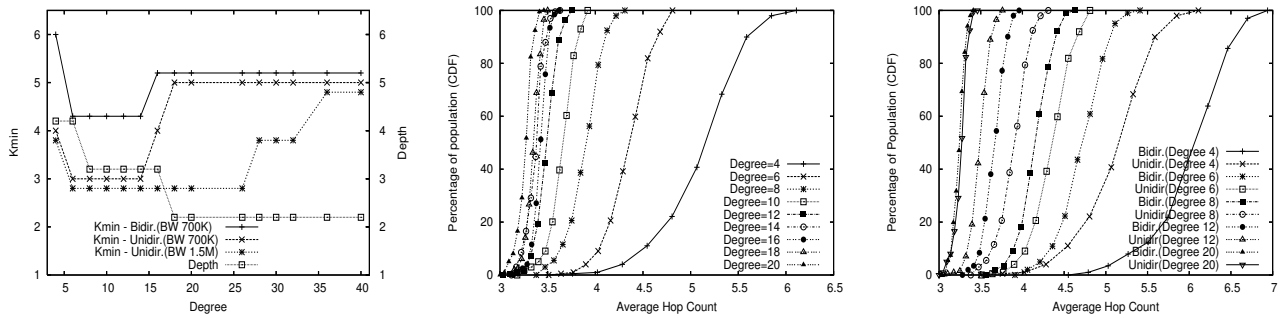
directly affects the depth of the overlay. For proper comparison we keep the number of swarming intervals constant across these simulations (K=3) by setting the value of ω as follows: $\omega = depth + 3$. Figure 3(a) shows two interesting points: *(i)* in each scenario, there is a sweet range of peer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

The poor performance of the system for small peer degree (degree =4) can be explained as follows: Given a fixed population of peers, increasing the peer degree increases the number of diffusion subtrees and thus decreases the population of peers in each diffusion subtree. This in turn proportionally reduces the probability that the randomly selected swarming parent for each peer would be located on the same diffusion tree, and therefore decreases the probability of content bottleneck during the swarming phase. In a nutshell, the diversity of swarming parents (across different diffusion subtrees) increases with peer degree for a fixed population of peers. This explanation implies that the lower bound of proper peer degree does not depend on peer bandwidth as shown in this figure. The rapid drop in the delivered quality for large peer degrees is the result of significant decrease in the throughput of individual connections, which depends on the bandwidth-to-degree ratio. Figure 3(a) illustrates that the upper bound for the scenario with peer bandwidth 1.5 Mbps is proportionally larger than the upper bound for peer bandwidth 700 kbps.

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 for a few degrees, respectively (only for a scenario with peer bandwidth 700 kbps). The percentage of content bottleneck in the diffusion (or swarming) phase is the percentage of bandwidth from the diffusion (or swarming) parent(s) that can not be utilized for content delivery. Comparing these figures shows that the percentage of content bottleneck is clearly higher in the swarming phase across all degrees as we discussed in Section III-B. Furthermore, as we increase the peer degree from 4 to 6, the percentage of content in both phases significantly decreases. But any further increase in peer degree (beyond 12) reverses the trends and rapidly increases the percentage of content bottleneck in both phases.

Loss Rate: To clearly illustrate the effect of peer degree on packet loss (and throughput of individual connection) Figure 4(a) plots the aggregate transmission rate from a parent to all its child peers, the parent's access link bandwidth and aggregate throughput to all child peers for a single peer. 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 associated bandwidth for 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 some connections are limited by the parent's outgoing link bandwidth while others are limited by the child's incoming access link bandwidth. This may seem surprising because the bandwidth-degree condition already limits individual connections' throughput by the parent's link. To investigate this issue, we examine the distribution of normalized average bandwidth (normalized by the corresponding $bwpf$) and its deviation across all connections for different peer degrees as shown 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 bandwidth across connections does not change but the distribution of bandwidth deviation shifts towards higher values. In a nutshell, the larger deviations with larger peer degree result in the bottleneck at both sender and receiver ends of individual connections. Note that session level simulators are unable to capture this important behavior.

Buffer Requirement: The poor performance outside the good operating region indicates that the number of swarming intervals is inadequate for delivery of the required quality 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 high a quality stream?"* Figure 5(a) depicts the number of diffusion intervals (*i.e.*, $depth$) and the minimum number of swarming intervals (K_{min} = ω_{min}-$depth$) as a function of peer degree in both reference



(a) Minimum K and depth

(b) Avgerage path length distribution with ω_{min}

(c) Bi- vs Uni-directional- avgerage path length distribution

Fig. 5. Effect of peer connectivity on system performance

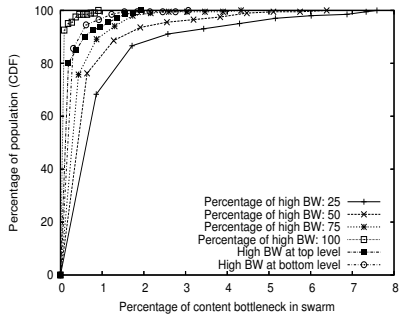
scenarios such that 90% of peers receive 90% of the maximum deliverable quality. Note that the depth of the overlay is independent of the peer bandwidth and gradually decreases with peer degree in a step-like fashion. As degree increases, K_{min} initially decreases from 4 to its minimum value (3). However, 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 (ω_{min}=$depth$+K_{min}) as a function or peer degree. It also suggests that there is a direct relationship between K_{min} and $bwpf$ in each scenario.

Average Path Length From Source: Another interesting issue is *"How does the average path length (in hops) of delivered packets to individual peers change as the overlay becomes more connected (*i.e.*, peer degree increases)?"* Figure 5(b) presents this information for several peer degrees over the reference scenario with bandwidth 700 kbps when the number of swarming intervals is equal to K_{min}. This figure shows 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 (*i.e.*, less skewed) due to the increase in the diversity of swarming parents (*i.e.*, availability of more shortcuts between diffusion subtrees). This also reduces the deviation of hop count across delivered packets to individual peers (not shown here). The rapid decrease in skewness of average path length by degree justifies that lost packets are asked from the same parents during the next swarming interval rather than through a longer path from other swarming parents.

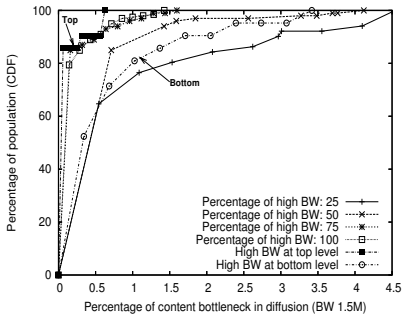
Bi- vs Uni-directional Connectivity: Maintaining uni-directional vs bi-directional connectivity between peers affect the nature of connectivity among peers and thus could impact the performance of content delivery mechanism. To investigate this issue, we examine the reference scenario with 700 kbps access link bandwidth but enforced 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 uni-directional overlay, over the sweet range of peer degree. Figure 5(a) also shows the value of K_{min} for the reference scenario (with 700 kbps) when connections are bidirectional. This figure indicates that bi-directional connections 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 (from child to parent) that connects two peers within the same diffusion subtree and thus it is not an effective shortcut. In a bidirectional overlay, swarming shortcuts between different sub-trees are established through connections between peers in the same level. Since most such intra-level connections are located at the bottom level, peers in higher levels of the overlay require a larger number of swarming intervals.

Figure 5(c) depicts the distribution of average path length in the above simulations with bidirectional connections. This figure also shows the hop count for the corresponding unidirectional scenario 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 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, as the peer degree increases (for a fixed population), the extra connections must establish useful swarming shortcuts (*i.e.*, bidirectional connections between peers in the same level). 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). Repeating these simulations with different peer bandwidth (1.5 Mbps), reveals that the performance of uni- and bi-directional overlays as a function of degree does not depend on peer bandwidth.

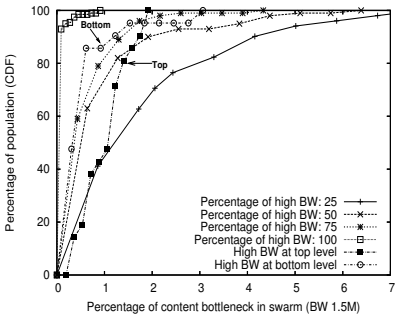
2) Bandwidth Heterogeneity: To investigate the effect of bandwidth heterogeneity, we consider the reference scenario with homogeneous peers and link bandwidth of 1.5 Mbps (bw_h) and reduce the link bandwidth for K_l percent of bandwidth to bw_l. The first question is: *"How are the delivered quality and buffer requirements of high bandwidth peers affected by bandwidth heterogeneity (*i.e., $\frac{bw_h}{bw_l}$*) and the*



(a) Percentage of content bottleneck in swarming

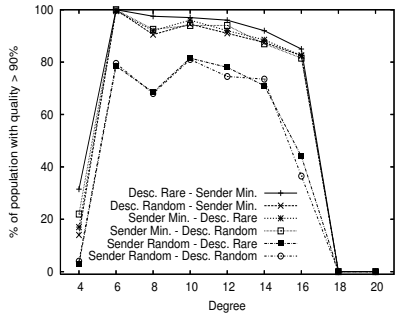


(b) Percentage of content bottleneck in diffusion for high BW peers

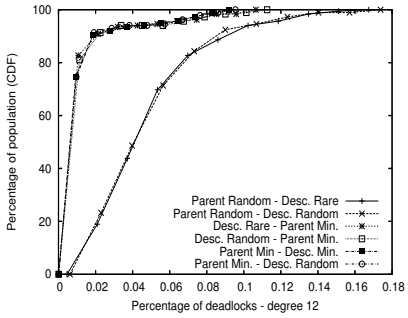


(c) Percentage of content bottleneck in swarming for high BW peers (1.5 Mbps)

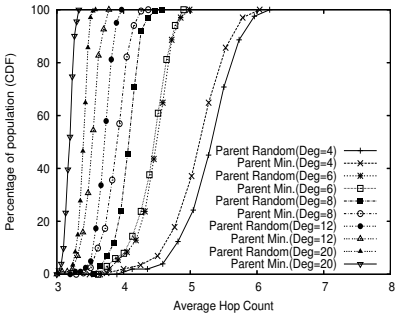
Fig. 6. Content bottleneck for high bandwidth peers in scenarios with heterogeneous bandwidth



(a) Effect of packet scheduling on system performance



(b) Distribution of percentage of dead-locks for degree 12



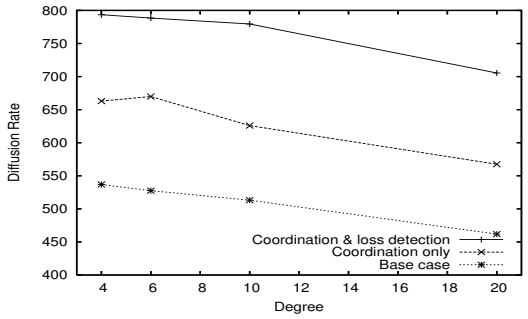
(c) Distribution of Average path length

Fig. 7. Packet scheduling effect

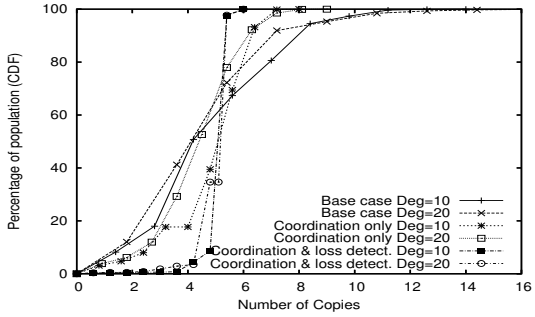
percentage of low bandwidth peers (K_l)?". As we showed in Section II, the bandwidth-degree condition ensures that the utilization of access link bandwidth remains high when peers have heterogeneous link bandwidth. The probability of content bottleneck for low bandwidth peers in this scenario is lower than the homogeneous scenario where all peers have the same bandwidth bw_l since the available quality among swarming parents is higher than the homogeneous scenario. We further discuss this issue in Section IV-C. Figures 6(b) and 6(c) show the percentage of content bottleneck among high bandwidth peers in a reference scenario (peer bandwidth 1.5 Mbps) for diffusion and swarming phases respectively where different percentage of peers (0%, 25%, 50% and 75%) are replaced with peers with a lower link bandwidth of 1 Mbps. We use the same bandwidth-to-degree ratio ($\frac{1.5Mbps}{12}$) in both scenarios for a fair comparison. However, this implies that the depth of the overlay increases as the number of high bandwidth peers decreases. Overall, these figures show that the percentage of high bandwidth peers does not have a significant impact on the content bottleneck in both phases, also the overall performance remains the same for all peers as in Figure 6(a). The minor increase in content bottleneck during the diffusion phase with small percentage of high bandwidth peers (in Figure 6(b)) is due to the decrease in the total number of connections and the resulting increase in the overlay depth, because each diffusion parent can always provide a new data unit as long as it can satisfy the bandwidth-degree condition.

In Figure 6(c) 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 swarming parents of each peer. 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. Examination of other heterogeneous scenarios (*e.g.*, various combinations of peers with 1.5 Mbps and 700 kbps bandwidth) similarly showed that the degree or percentage of bandwidth heterogeneity has a minor impact on the delivered quality to high bandwidth peers.

Location of High Bandwidth Peers: Another important question in an overlay with heterogeneous peers is: *"Does the location of high bandwidth peers in the overlay affects the percentage of content bottleneck among them?"* To examine this issue, we explore a scenario with heterogeneous bandwidth where only 10% of peers have a lower bandwidth of 1 Mbps. We modify the overlay construction mechanism to place high bandwidth peers either in the top level (level 1) or at the bottom level. Figures 6(b) and 6(c) show the percentage of content bottleneck for these two cases (with special labels) that allow comparison with other scenarios. Placing the high bandwidth peers at the top level slightly reduces overlay depth and leads to a minor decrease in content bottleneck during the diffusion phase, however it does not affect the swarming phase. In contrast, placing the high bandwidth peers at the



(a) Effect of Source coordination on Diffusion



(b) Distribution of packets in level 1

Fig. 8. Source Coordination

bottom level slightly increases overlay depth and causes a minor increase in content bottlenecks during the diffusion phase. However, this effect is compensated for the decrease in content bottleneck during the swarming phase due to a larger number of swarming shortcuts. *In summary, the location of high bandwidth peers does not have a significant impact on the minimum buffer requirement (*i.e.*, ω). However, placing the high bandwidth peers in non-bottom levels reduces the number of diffusion intervals while equally increases the number of swarming intervals and vice versa.* To justify this finding, we note that the effect of a high bandwidth peer on overall depth of the delivery tree (ω) does not depend on its location in the tree.

B. Packet Scheduling

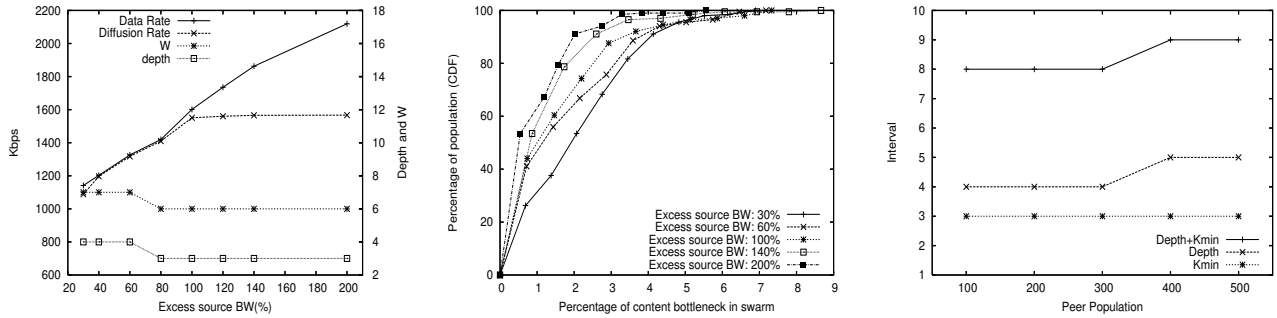
The receiver-driven packet scheduling mechanism at each peer is another important factor whose collective behavior across all peers determines the overall pattern of content delivery and thus the probability of content bottleneck at individual peers. To study this issue, we consider the reference scenario with access link of 700 kbps where all peers employ the same packet scheduling mechanisms. Figure 7(a) depicts the percentage of peers that receive 90% of the maximum deliverable quality for six different packet scheduling mechanisms where $\omega = depth + 3$ (*i.e.*, $K = 3$). 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 sweet range of degree. This implies that neither the criteria for selecting the description of a packet nor the order of selection (between description and parent) significantly affect the overall performance. Second, the percentage of peers that receive a high quality stream in the two algorithms that randomly select the parent for a packet is very similar, and roughly 20% lower than other algorithms within the sweet region. Examination of K_{min} for these two scheduling schemes revealed that their K_{min} value is always one interval larger than other schemes in a comparable scenario intuitively, those scheduling schemes that randomly select the parent might exhibit higher content bottleneck because they experience *deadlock* in packet assignment more frequently than other schemes. A deadlock occurs when a required packet is available among parents but it can not be requested since the bandwidth budget of those parents who can serve the content is already utilized for delivery of other packets. To verify this hypothesis, we examine the distribution of deadlock frequency (*i.e.*, the fraction of packets whose scheduling leads to a deadlock) among peers in the above scenario with 200 homogeneous peers. Figure 7(b) depicts the distribution of deadlock frequency for peer degree of 12 and confirms that scheduling schemes with random parent selection experience deadlock more frequently (*i.e.*, the median frequency is roughly four times higher for random selection strategies). Note that in a random parent scheduling all the unique contents (new data unit) of a parent may not be requested, because due to the random packet selection a portion of parent's budget may be used for requesting of packets available in other parent(s).

This raises the question of how this difference in the frequency of deadlock events affect the average path length in the overall pattern of content delivery, *i.e.*, the reason for larger K_{min} is due to a longer path during the swarming phase? Figure 7(c) depicts the distribution of average hop-count for the $ParentMin. - LayerRandom$ and $ParentRandom - LayerRandom$ scheduling schemes across different peer degrees from the range of degrees (4 to 20). Interestingly, the distribution of hop-count for both scheduling schemes are very similar across this range. This means that the extra interval for the swarming phase is only required to compensate for the poor scheduling and receive the deadlocked packets from the same parents during the next swarming interval (*i.e.*, rather than through a longer path from other swarming parents).

C. Source Behavior

In this section, we quantify the effect of the following two orthogonal aspects of source behavior on system performance: *(i)* Packet swapping and loss detection, and *(ii)* Source bandwidth.

Packet Swapping & Loss Detection: We explore the effect of source coordination in the reference scenario with 700 kbps link bandwidth where source bandwidth and K_{min} are 900 kbps and 5, respectively. This configuration ensures all peers receive a high quality stream. Figure 8(a) shows the delivered quality to the system (*i.e.*, diffusion rate to level 1) as a function of peer degree in three different cases: *(i)* source without any coordination, *(ii)* source with packet swapping, and *(iii)* source with packet swapping as well as loss detection. This figure demonstrates that while the diffusion rate slowly decreases with peer degree in all three cases, incorporating packet swapping significantly increases the diffusion rate, and



(a) Effect of Source BW

(b) Effect of Source BW on bottleneck in swarming

(c) variations of per-peer buffer size with population

Fig. 9. Source Bandwidth & Scalability

adding loss detection leads to further improvement in the diffusion rate.

Note that the source throughput (*i.e.*, the rate of delivered packets) to level 1 is not affected by source coordination. Figure 8(b) depicts the distribution of the number of delivered copies for individual packets to level 1 in these three cases for two different peer degrees (10 and 20). This figure reveals that the addition of packet swapping and then loss detection progressively improves the uniformity of the number of copies to level 1 for both degrees. When both mechanisms are used, the number of delivered copies are very even. In essence, incorporating these two mechanisms enables us to deliver the same quality with lower source bandwidth or to improve delivered quality for a given source bandwidth.

Source Bandwidth: The next key question is *"How does source bandwidth affect delivered quality and buffer requirement at individual peers?"*. Figure 9 shows the effect of excess source bandwidth (beyond the stream required bandwidth of 700 kbps) on the following properties in the reference scenario with 700 kbps and peer degree of 6: throughput to level 1, diffusion rate, overlay $depth$ and ω. The values on the X axis represent the normalized value of excess source bandwidth, *i.e.*, $\frac{SourceBW-700kbps}{700kbps}$. This figure shows that increasing source bandwidth has two effects: First, it increases the source degree (due to the bandwidth-degree condition) and thus reduces the overlay depth which slowly decreases the buffer requirement at individual peers as shown in Figure 9. Second, increasing source bandwidth (with packet swapping and loss detection) increases the number of diffusion subtrees with unique content and thus improves the delivered quality to level 1 (*i.e.*, the diffusion rate) until it reaches the maximum available quality at the source. This in turn increases the diversity of swarming shortcuts among subtrees and reduces the percentage of content bottleneck among peers during the swarming phase as shown in Figure 9(b). Once the delivered quality to level 1 is maximized, further increasing the source bandwidth results in adding redundant diffusion subtrees (that do not have unique content). This reduces overlay depth and slightly reduces content bottleneck during the diffusion phase

D. Peer Population

The final issue is the scalability of PRIME mechanism. More specifically, *"How does the delivered quality and buffer requirement at individual peers change with peer population?"*. Figure 9(c) shows overlay depth, K_{min} and ω as a function of peer population in our reference scenario with peer degree of 6. This figure illustrates the scalability of PRIME protocol. As the 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 increasing population. *This result implies that a system that operates in the sweet region of peer degree can easily accommodate a major increase in peer population while slowly increases peer buffer size.*

V. Conclusions

In this paper, we presented a new mesh-based P2P streaming mechanism for delivery of live content, called PRIME. PRIME is unique because it effectively incorporates a swarm-like delivery to utilize the outgoing bandwidth of all participating peers in a scalable fashion. We described the peer connectivity and packet scheduling mechanism in PRIME that minimize bandwidth bottleneck and content bottleneck at individual peers, respectively. Through extensive ns simulations, we showed the effect of key components on PRIME performance and identified a few fundamental design tradeoffs in mesh-based P2P streaming.

References

[1] Y. Chu, S. G. Rao, S. Seshan, and H. Zhang, "Enabling conferencing applications on the internet using an overlay multicast arcitecture," in *ACM SIGCOMM*, Aug. 2001.

[2] V. N. Padmanabhan, H. J. Wang, and P. A. Chou, "Resilient peer-to-peer streaming," in *IEEE ICNP*, 2003.

[3] M. Castro, P. Druschel, A.-M. Kermarrec, A. R. A. Nandi, and A. Singh, "SplitStream: High-bandwidth content distribution in a cooperative environment," in *ACM SOSP*, 2003.

[4] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "Coolstreaming: A data-driven overlay network for live media streaming," in *IEEE INFOCOM*, 2005.

[5] B. Cohen, "Bittorrent." [Online]. Available: http://www.bittorrent.com

[6] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. E. Mohr, "Chainsaw: Eliminating Trees from Overlay Multicast," in *International Workshop on Peer-to-Peer Systems*, 2005.

[7] A. Vlavianos, M. Iliofotou, and M. Faloutsos, "Bitos; enhancing bittorrent for supporting streaming applications," in *Global Internet*, 2006.

[8] N. Magharei and R. Rejaie, "Understanding Mesh-based Peer-to-Peer Streaming," in *NOSSDAV*, 2006.

[9] R. Rejaie, M. Handley, and D. Estrin, "RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the internet," in *IEEE INFOCOM*, 1999.