PRIME: P2P Receiver-drIven MEsh-based Streaming

Nazanin Magharei
University of Oregon
nazanin@cs.uoregon.edu

June 9, 2006

Abstract

Multimedia services based on Peer-to-Peer live streaming such as IPTV, are getting increasingly popular among users specially with improvement of network bandwidth. The goal of these mechanisms is to maximize the delivered quality to individual peers with minimum buffer requirement in a scalable fashion. However existing P2P streaming schemes can not achieve this goal due to their inability to utilize available resources. In this paper we present the design and evaluation of a novel approach to live media streaming over P2P mesh-based overlay, called P2P Receiver-drIven-MEsh-based streaming or PRIME. In PRIME participating peers form a randomly connected and directed overlay mesh and incorporate a swarm-like content delivery to effectively contribute their outgoing bandwidth. We explore the design space of mesh-based P2P streaming mechanisms through PRIME which provides an insight on fundamental tradeoffs in design of mesh-based P2P streaming mechanisms. In particular, through extensive simulations we illustrate the effect of overlay properties, source behavior, per-peer packet scheduling and peer populations on system performance. Our evaluations show that PRIME can effectively provides high quality live P2P streaming to a large number of peers with minimum buffer requirement, without imposing an additional bandwidth requirement to source.

1 Introduction

During recent years, the increasing ability of average users to generate multimedia content coupled with the availability of high bandwidth connections especially to residential users, motivated research on streaming multimedia content over the Internet. A popular streaming application is one-to-many streaming of *live* video over the Internet (*e.g.*, IPTV [1]). IP multicast was considered a proper solution for live multimedia streaming over the Internet during the past decade. However, the limited deployment of IP multicast has motivated a new approach to streaming based on Peer-to-Peer overlays that is known as P2P streaming. In P2P streaming, participating peers form an overlay and contribute their outgoing bandwidth by sending the streaming content to other peers. A P2P streaming mechanism should accommodate the following challenging goals: (i) *effectively utilizing outgoing bandwidth of most of the participating peers*, (ii) *maximizing the delivered quality to individual peers considering heterogeneity and asymmetry of access link bandwidth among peers*, (iii) *minimizing buffer requirement and thus the playback delay at each peer*, (iv) *achieving scal-*

ability and (v) *providing resiliency to peer dynamics*.

Most of the existing P2P streaming applications form a tree-shaped overlay where the content is pushed through the overlay, from a source (*i.e.*, root) towards all peers. This approach cannot provide high quality stream to participating peers due to the following reasons: *(i)* It can not utilize outgoing bandwidth of all participating peers (particularly leaves of the tree). *(ii)* Delivered quality to each peer is limited to the minimum bandwidth among the upstream connections from source. *(iii)* Departure of individual peers could disrupt the delivered quality to its down stream peers. *(iv)* Maintaining an efficient tree is expensive due to the dynamics of peer participation. An extended version of this approach organizes peers into multiple trees where each peer is an internal node in only one tree and a leaf node in all other trees [2]. Then individual descriptions of a multiple description coded (MDC) stream is pushed through each tree. This approach improves utilization of resources and resiliency to peer dynamics, however due to the static mapping of content to trees, delivered quality to participating peers is still limited.

The limitations of tree-based approaches have motivated a new approach in which participating peers form a random mesh and the content delivery is pull-based. File swarming mechanisms such as BitTorrent have inspired this new approach. In file-swarming mechanisms, different pieces of a file are distributed among different peers which enables them to contribute their out-going bandwidth more effectively. Swarming is simple and efficient particularly for bulk data transfer. However incorporating swarm-like delivery into live P2P streaming applications is challenging due to two important reasons: *(i)* swarming can not guarantee the in-time delivery requirement of live streaming, *(ii)* limited availability of future content in live streaming could significantly affect the performance of swarming. A couple of recent studies showed that it is in fact feasible to incorporate swarming into P2P streaming in certain scenarios [3, 4, 5]. However, to our knowledge, none of the previous studies have examined the effect of the key parameters (*i.e.*, Overlay properties, source behavior, packet scheduling) on the performance (namely delivered quality and playback delay of individual peers) of swarm-incorporated live P2P streaming mechanisms. More specifically the following basic issues about P2P mesh-based streaming of live content have not been addressed: What are the performance bottlenecks and key design tradeoffs in a mesh-based P2P streaming? How to incorporating swarm-like content delivery into P2P live streaming?

In this project, we design a novel approach to live P2P streaming called Peer-to-peer Receiver drIven MEsh-based streaming or PRIME. We explore the design space of mesh-based P2P streaming mechanisms and identify fundamental design tradeoffs.

PRIME has two components: *(i)* overlay construction and *(ii)* content delivery. In PRIME, participating peers form a *directed and randomly connected mesh*. Each peer has multiple parent peers and multiple child peers. This simple approach to overlay construction results in an overlay which is resilient to churn and has diverse connections. We derive the proper incoming/outgoing peer degree based on incoming/outgoing bandwidth of individual peers to maximize the utilization of their access link bandwidth. The content delivery in PRIME is performed using progressively push reporting by parents coupled with periodically pull requesting by child peers. PRIME leverages MDC encoded content to accommodate bandwidth heterogeneity. Each peer receives content from all of its parents and provides content to all of its child peers. A packet scheduling mechanism at each peer determines the requested packets from each parent to maximize its delivered quality. We present a two phase content delivery mechanism that maximizes delivered quality to individual

peers. We develop a new evaluation methodology and extensively evaluate performance of PRIME using packet level simulations. In particular, we examine the effect of peer connectivities, per-peer packet scheduling, source behavior and peer populations on PRIME performance. Our results show some important design tradeoffs and relationships among key parameters. It also sheds an insightful light on dynamics of content delivery in mesh-based P2P streaming of live content.

The rest of this paper is organized as follows: Section II discusses the related work. Section III gives an overview of overlay construction and content delivery with two performance bottlenecks in P2P streaming mechanisms. Sections IV and V describe how PRIME addresses the mentioned performance bottlenecks. Section VI presents the behavior of source in the system. In Section VII, we present our evaluation methodology along with our simulation results. Section VIII concludes the paper.

2 Related Work

In recent years, streaming media over P2P overlays has received significant attention and many studies have been done on this topic. There exist two distinct categories of solutions: tree-based overlays and mesh-based overlays. In this section we give an overview of the existing solutions in the context of overlay based media streaming mechanisms.

2.1 Single/multiple tree(s) based overlays

Many existing P2P streaming systems deploy a tree structure. There are two types of tree-based protocols: *(i)* single tree [6] [7] [8] [9], and *(ii)* multiple tree protocols [2] [10]. The major issue in tree-based protocols is building and maintaining a single or multiple tree(s) with desired properties. Many of these approaches use delay or bandwidth as the single or primary metric in the tree construction, attempting to minimize the delay or maximize the bandwidth connectivity from source to individual peers.

2.1.1 Single tree overlays

Some of the single tree approaches such as Zigzag [6] and NICE [7] focus on maintaining a scalable tree using efficient distributed mechanisms while limiting the out-degree of peer nodes to satisfy the high bandwidth requirement of applications such as streaming. Both approaches organize members of a multicast group into a hierarchy of clusters to minimize transmission delay. SpreadIt [9] is another tree-based approach for small scale groups. It incorporates a peering layer among participating nodes that enables participating peers to redirect another peer to get the data from. The single tree overlay can not gracefully accommodate churn. Several tree maintenance mechanisms have been proposed but they often result in a considerable overhead for maintaining a tree with desired characteristics [11] [12] [6] [13] [14].

The single-tree approach has the following inherent limitations: *(i)* It is unable to utilize all outgoing bandwidth of leaf nodes, that limits its scalability, *(ii)* it can not gracefully accommodate churn and *(iii)* it can not accommodate heterogeneity of access link bandwidth among participating peers.

2.1.2 Multiple trees overlays

Limitations of the single tree approach have motivated the multiple tree approaches (*e.g.*, [2] and [10]). In CoopNet [2] and Splitstream [10], the main idea is to encode the stream into several descriptions and each description is delivered through a single multicast tree. The source as the root of all trees,

collects the information of all the nodes for tree construction and maintenance. To cope with dynamic peers, each node is an internal node in one tree and a leaf node in all others. Such a centralized complicated algorithm relies on a powerful and dedicated root node. The overhead of partitioning and reconstructing the desired trees specifically in presence of churn is huge.

In multiple-tree approach, there are some major limitations that affect the performance of the system: *(i)* static content-tree mapping affects the efficiency and limits the flexibility of requesting packets (specially losses) from multiple paths, *(ii)* bandwidth requirement of each connections through the trees limits the accommodation of bandwidth dynamics during a session (*e.g.*, bandwidth through each tree could not be less than a description bandwidth), *(iii)* complexity of tree construction and maintenance leads to a high overhead and *(iv)* the centralized approach that relies on root node, has limited scalability.

2.2 Mesh based overlays

Due to the tree-based approach limitations, mesh-based protocols have been proposed, [3, 5, 15, 16, 17, 18] in which each peer receives media data from multiple peers and provides content to multiple peers.

Bullet [16] is a P2P content distribution mechanism that creates a mesh over a tree, to achieve a high bandwidth throughput. The root and intermediate nodes send disjoints set of data to their children. The children are getting data from their tree parent and allowed to search other peers with disjoint data for the remaining blocks using a distributed algorithm called RanSub [19]. Bullet focuses on distributing elastic contents and is unable to guarantee timing constraint of streaming content.

In PROMISE [15] participating peers construct a mesh overlay. Peers select their parents based on the best sending peers by a topology-aware selection technique. Each peer in PROMISE monitors the status of other peers and it might dynamically switch between senders to react to sudden bandwidth change. PROMISE by incorporating a large buffer and long start-up delay let the senders collectively send FEC-incorporated content to the receivers with the assigned rate.

A couple of recent studies inspired by BitTorrent [20] mechanism, incorporate swarm-like content delivery [3], [4]. File swarming mechanisms such as BitTorrent [20] achieves good performance by effectively utilizing out bandwidth of all participating peers [21]. In these systems the entire file which is available at the source, is broken into pieces that are independently distributed among peers. Due to the randomness of requesting and receiving pieces, they may be distributed our-of-order as the goal is to provide pieces as soon as they are received and there is no deadline for them. In bulk data transfer all pieces of the file are known priori which leads to an effective swarming.

Incorporating swarm-like content delivery into P2P live streaming is a challenging task due to: *(i)* The limited availability of content in live streaming affects the performance of swarming and *(ii)* In-time delivery requirement of live streaming can not be guaranteed by swarming. Despite these challenges several studies have incorporated swarm-like delivery into a P2P streaming mechanisms. CoolStreaming [3] is based on a random mesh-based overlay in which peer discovery is performed by a gossiping protocol. The content delivery part is receiver-driven and similar to the file swarming. The content delivery mechanism requests each packet from a parent for delivery with maximum bandwidth and enough available time among possible parents. Chainsaw [4] is a similar approach that works on top of an existing overlay. The content delivery mechanism in

Chainsaw, request each packet from a random parent for delivery. Some other studies modifies BitTorrent (*e.g.*, [22], [23] and [24]). All the above studies have illustrated the feasibility of incorporating swarm-like delivery into a P2P streaming mechanism.

However, to our knowledge none of the previous studies answered the fundamental questions: *(i)* What is the global pattern for streaming live content over a mesh-based overlay to effectively utilize outgoing bandwidth of most participating peers? *(ii)* How is the delivered quality to individual peers in a P2P streaming mechanism affected by peer connectivity (*i.e.*, peer degree, heterogeneity and bi- vs. uni-directional connections), packet scheduling, source behavior and peer populations?

3 PRIME Overview

Each P2P streaming system consists of two major components: (i) **Overlay Construction**, that organizes participating peers into an overlay and (ii) **Content Delivery**, that determines delivery of content into individual peers through the overlay. First peers form an overlay and then start delivering the content through that overlay. In this section, we present an overview of these two components in PRIME and then we discuss two potential performance bottlenecks in any P2P streaming mechanism.

3.1 Overlay Construction

The overlay construction in PRIME is very simple. Participating peers form a randomly connected and directed mesh. All connections are uni-directional *i.e.*, there is a parent-child relationship between connected peers. Each peer, has multiple parents and multiple children. Each peer as a child, identifies sufficient number of peers to utilize its incoming access link bandwidth. To discover parent peers, each peer contacts a bootstrapping node to learn about other existing peers in a demand-driven fashion. Such an overlay has several advantages: *(i)* Building and maintaining mesh-based overlays is simple and has low overhead, *(ii)* it is resilient to churn and *(iii)* connections are more diverse thus it is less likely that incoming connections from parents to a child peer share a bottleneck inside the network.

3.2 Content Delivery

Similar to other swarming mechanism, content delivery in PRIME incorporates push reporting coupled with pull requesting. To accommodate bandwidth heterogeneity among peers, the stream is encoded using a Multiple Description Coding (MDC) scheme at source. [1] All connections for content delivery are congestion controlled [33] to properly share bandwidth with other traffics. To send the reports to child peers, each peer as a parent progressively piggybacks a list of newly available packets to its child peers within each data packets. Each peer as a child periodically (every Δ seconds) sends a separate list of requested packets to each one of its parents. Requested packets are determined by a packet scheduling mechanism at each child peer. Parent peers deliver requested packets in the provided order and at the rate that is determined by the congestion control mechanism.

For live P2P streaming applications, source generates a new segment of length Δ, once every Δ seconds. Each 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 delayed by $\omega*\Delta$ seconds behind source playout time. This implies that individual

[1] In MDC coding, there is no decoding dependency among descriptions. Therefore any subset of descriptions can be decoded (viewed) by each peer.

peers should buffer at least $\omega*\Delta$ seconds of content and each packet should be delivered to each peer within $\omega*\Delta$ seconds after its generation time by source. The key component of content delivery is a packet scheduling at individual peers. The details of packet scheduling mechanism is discussed in Section V-B.

3.3 Performance Bottlenecks

The main design goal of a live P2P streaming mechanism is to maximize the delivered quality to individual peers while minimizing required buffering at each peer. There are two issues that can limit the delivered quality to individual peers:

- **Bandwidth bottleneck** occurs when the aggregate available bandwidth from all parents to a child peer can not fully utilize the child's incoming access link bandwidth.
- **Content bottleneck** occurs when the aggregate useful content of all parents of a child peer is not sufficient to fully utilize its available bandwidth.

We decouple these two factors to distinguish the contribution of each factor in delivered quality to each peer. Each parent sends packets to each child peer at a rate determined by a congestion control mechanism. At a packet transmission time to a particular child peer, if there is no outstanding packet to send, the parent sends a marked packet with the same size as data packet. This decouples bandwidth bottleneck from content bottleneck and enables child peers to quantify the contribution of bandwidth and content bottlenecks in the delivered quality. In the next sections, we discuss how to address bandwidth bottleneck and content bottleneck separately.

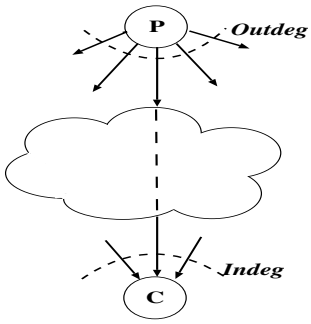
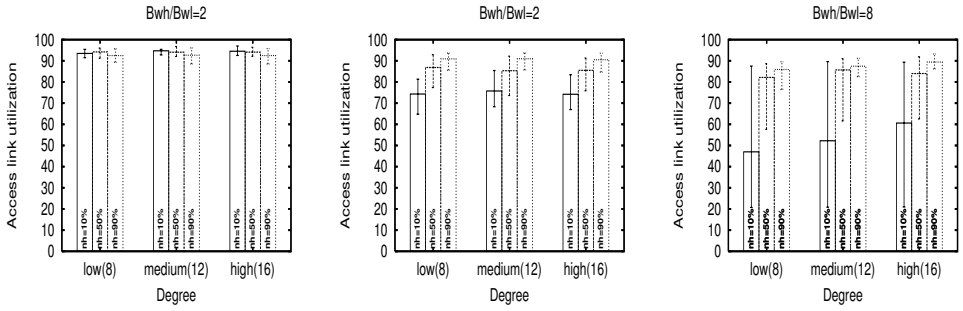


Figure 1: A connection from P to C with access link bandwidth $outbw_P$ and $inbw_C$ respectively

4 Addressing bandwidth bottleneck

The aggregate bandwidth to each child peer depends on *(i)* the number of its parent peers (*i.e.*, indegree) and *(ii)* the number of child peers for those parents (*i.e.*, parents' outdegree). Note that bandwidth bottleneck only depends on overlay properties. Figure 1 shows a child peer with incoming access link bandwidth of $inbw_c$ and indegree of $indeg_c$ as well as one of its parent peer with outgoing access link bandwidth of $outbw_p$ and outdegree of $outdeg_p$. Suppose that congestion occurs only at the edge of the network, *e.g.*, the incoming/outgoing access links of participating peers. The bandwidth connection between the child peer c and the parent peer p in Figure 1, can be estimated by $MIN(\frac{outbw_p}{outdeg_p}, \frac{inbw_c}{indeg_c})$. If the first ratio is smaller then the parent p outgoing access link is the bottleneck and the incoming access link of child c can not be fully utilized. On the other hand if the second ratio is smaller, the incoming access link of child c is the bottleneck. This suggests that to minimize the bandwidth bottleneck in a randomly connected mesh-based overlay, the same bandwidth to degree ratio should be used for all connections of all participating peers. We call this the $bandwidth - degree$ condition which should be satisfied by all participating peers i and j: $bwpf = \frac{outbw_i}{outdeg_i} = \frac{inbw_j}{indeg_j}$



(a) $bandwidth - degree$ condition enforced

(b) Fixed degree - Bandwidth heterogeneity ratio:2

(c) Fixed degree - Bandwidth heterogeneity ratio:8

Figure 2: Access link bandwidth utilization

This condition implies that all connections in the overlay should have the same bandwidth. Note that this can easily accommodate heterogeneity of bandwidth among peers by choosing a proper in/out degree to have the same $bwpf$ across all connections.

To examine the effect of the $bandwidth - degree$ condition on utilization of access link bandwidth, we conduct ns simulations with 200 peers. Peers have heterogeneous access link bandwidth (bw_h and bw_l) and form a randomly connected mesh. We examine the following two scenarios: *(i)* all peers have the same fixed degree (8, 12 and 16) and *(ii)* the degree of all peers are set based on $bandwidth - degree$ condition (proportional to their access link bandwidth). We keep the total number of connections fixed in both scenarios for each degree for a fair comparison. All connections are congestion controlled [33]. We examine each scenario with two different ratios of bandwidth heterogeneity *i.e.*, $\frac{bw_h}{bw_l}$ (2 and 8) among peers while keeping the low bandwidth fixed as 700 kbps. We also change the percentage of high bandwidth nodes (nh) between 10%, 50% and 90%, to examine its impact on utilization of access link bandwidth. Figures 2(a), 2(b) and 2(c) show the average utilization of incoming access link bandwidth for 3 scenarios: *(a)* $bandwidth - degree$ condition, *(b)* fixed degree with $\frac{bw_h}{bw_l} = 2$ and *(c)* fixed degree with $\frac{bw_h}{bw_l} = 8$, respectively. Each figure shows bandwidth utilization in three different peer degree settings *i.e.*, low, medium and high. Within each degree setting, the three boxes show access link utilization for three bandwidth settings: 10%, 50% and 90% of the population being high bandwidth. The boxes represent the median of access link utilization among high bandwidth peers with its 10th and 90th percentile values as bars.

These figures illustrate the following points: *(i) Without $bandwidth - degree$ condition, utilization of access link among high bandwidth peers is not full and it has high variations.* The average access link utilization in scenario with enforced $bandwidth - degree$ condition is always more that 95% with low variation ($< 3\%$). *(ii)* Increasing percentage of high bandwidth peers (*i.e.*, $nh = 10\%$, 50% vs. 90%) , improves their access link bandwidth utilization due to the increasing number of connections among them. *(iii)* In fixed degree scenarios, increasing the degree of bandwidth heterogeneity (*i.e.*, $\frac{bw_h}{bw_l} = 2$ vs. 8), results in a larger drop in average utilization of access link among high bandwidth peers. *(iv)* Similarly

increasing peer degree results in higher utilization of access link among high bandwidth peers due to the larger number of connections among them.

In practice, some connections might experience bottleneck inside of the network. This causes under utilization of access link bandwidth of some peers that have connections through these links. To address this issue, child peers with low utilization of incoming access link bandwidth can select extra parent peers. Similarly parent peers with low utilization of outgoing access link bandwidth can accept extra child peers.

5 Addressing content bottleneck

Content bottleneck is a function of both content delivery and overlay properties (peer connectivities). In this section, we focus on the effect of content delivery on content bottleneck and discuss the impact of overlay properties in Section VII-A.

Suppose by enforcing the $bandwidth - degree$ condition, all connections have roughly the same bandwidth ($bwpf$). We define a data unit as the amount of data that a parent can send to each one of its child during one interval of Δ seconds. Thus, a data unit can be estimated as $D = bwpf{*}\Delta$. A data unit may consist of packets from different descriptions that are selected by the packet scheduling mechanism at a child peer. To avoid content bottleneck, each parent of a child peer must have at least one useful data unit to offer to each child peer during a Δ interval. The role of the packet scheduling at individual peers is to request the useful data units from its parents in order and to maximize its delivered quality to avoid content bottleneck. Achieving this goal is the same as maximizing utilization of the outgoing bandwidth of all parent peers which leads to self-scaling of resources. The collective behavior of packet scheduling at individual peers determines the overall performance of the system.

Considering the $\omega{*}\Delta$ seconds buffer size at each peer, different data units of each segment should be delivered to each peer within $\omega{*}\Delta$ seconds after it is generated at source. The global pattern of content delivery from the source to all peers through the overlay determines the availability of new data units at each parent peer and thus the probability of content bottleneck by individual peers. The global pattern of content depends on the collective behavior of packet scheduling mechanism at individual peers. We first derive the global pattern of content delivery that minimizes content bottleneck among peers. Then we explain per-peer packet scheduling that leads to such a desired global pattern.

5.1 Global Pattern of Content Delivery

The primary design goal of PRIME is to minimize the buffer size (*i.e.*, playback delay) while maximizing the delivered quality to individual peers. As we have discussed earlier, achieving this goal depends on the 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 to all peers, the pattern of delivery should have two phases as follows: first once a segment is generated at the source, all participating peers should receive a data unit of that segment as fast as possible (*i.e.*, diffuse the segment to all peers). Then peers can exchange (*i.e.*, swarm) their data units with each other to receive a number of data units for that segment corresponding to their desired quality. In a nutshell, content delivery for a segment occurs at 2 phases: *(i) Diffusion phase* and *(ii) Swarming phase*. To clarify the description of the global pattern of content delivery, we describe an organized view of the

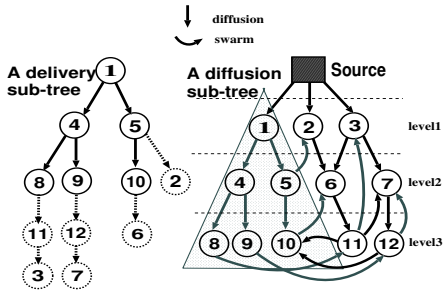


Figure 3: Organized view of a mesh-based overlay with 12 peers

mesh-based overlay.

Organized view of the overlay mesh: Organizing the mesh-based overlay simplifies the explanation of the global pattern of content delivery through a mesh-based overlay. Toward this end, we group peers into levels based on their shortest distance from source. Peers that are exactly one hop away from source, are grouped into level 1, peers that are two hops away from source are located in level 2 and so on (as shown in Figure 3). This view of the overlay reveals some simple but important properties of a mesh-based overlay. Consider the overlay that consists of P homogeneous peers with the same in- and out-degree deg and source degree of deg_{src}. Such an overlay exhibits the following properties: *(i)* The population of peers at level i ($pop(i)$) is at most $deg_{src}{*}deg^{(i-1)}$ and simply this reveals that by going down through the levels populations of increases. *(ii)* The number of levels ($depth$) of the overlay can be estimated as $log_{deg}(P/deg_{src}) \leq depth$. *(iii)* The probability of having a parent at level i is equal to $\frac{pop(i)}{P}$. Each peer in level i, typically has a single parent in level $i-1$ which we call diffusion parent, and $deg-1$ parents in the same or lower levels which we call swarming parents. In practice, small number of peers may have more than one parents in the higher level due to the random overlay construction, this reduces the populations of peers in their corresponding level and might increases the $depth$ of the overlay.

5.1.1 Diffusion phase of a segment

Considering the mentioned pattern of delivery, the first phase of delivery of a segment is *diffusion phase*. Upon generation of the segment in the source, all peers in level 1 should collectively pull all data units of that segment from source during the next interval Δ, this is the start of diffusion phase for this segment. Peers in level 2 at the next interval Δ should pull those data units from their *diffusion parents* in level 1 and so on. Therefore the fastest time for delivery of all different data units of the segment to the lowest level $depth$ is $depth{*}\Delta$ seconds. To achieve this minimum diffusion time, all connections from diffusion parents (diffusion connections) should be exclusively used for diffusion of new data units. These connections are shown by straight lines in Figure 3. The number of diffusion connections between each two levels is at least equal to the number of peers in lower level (due to the possibility of having multiple diffusion parents number of diffusion connections might be more that number of peers in lower level).

By explicitly using diffusion connections for diffusion of new data units, after $depth$ intervals each participating peer in the overlay has one data unit of the segment within $depth{*}\Delta$ seconds from its generation time. This restriction has the following implications: *(i)* Diffusion phase for a segment takes $depth$ intervals. *(ii)* each peer p in level 1 as well as all the peers in a subtree rooted at p receive the same data unit of the segment during the diffusion phase of that segment but at different intervals based on their level. We call these subtrees, diffusion subtrees. Figure 3 shows the diffusion subtree rooted at peer 1. *(iii)* The number of diffusion subtrees is equal to the deg_{src}, but the uniqueness of data units

in each subtree depends on source bandwidth that may cause redundancy between subtrees. *(iv)* When the bandwidth of a diffusion connection decreases to less than $bwpf$, all the downstream peers in that diffusion subtree experience content bottleneck during the diffusion phase.

5.1.2 Swarming phase of a segment

At the end of the diffusion phase of a segment all participating peers have one data unit of that segment. During the swarming phase, each peer should pull missing data units of the segment from its swarming parents. The number of unique data units that each peer should receive for each segment depends on its required quality (which is proportional to its available bandwidth). The connections from swarming parents are called swarming connections and are exclusively utilized for swarming. These connections are shown with curly arrows in Figure 3. As we have mentioned in Section V-A.1, all peers on a particular diffusion subtree receives the same data unit of a segment during its diffusion phase. Therefore only a swarming parent that is located at a different diffusion subtree can rapidly provide a useful data unit of that segment to a child peer. For example, in Figure 3, peer 12 is a swarming parent for peer 7 and both are located on the same diffusion subtree. On the other hand, peer 11 can provide a different data unit to peer 7.

To receive its maximum deliverable quality, each peer with in-degree deg should receive deg different data units. Ideally, if all swarming parents of a child peer located at $deg-1$ different diffusion subtrees, the child peer can pull $deg-1$ unique data units in the first interval of swarming phase. Due to the random connectivity among peers, some swarming parents of each peer may reside on the same diffusion subtree and this causes content bottleneck in swarming phase for each peer. However, during extra swarming intervals some of these swarming parents (on the same or different subtrees) will obtain new data units of the segment and can provide them to the child peer. This implies that the minimum number of required intervals to receive $deg-1$ unique data units of a segment (swarming phase) may be more than one for each peer, depending on the location of its swarming parents. The duration of swarming phase can be different for each peer (*e.g.*, peer 10 in Figure 3 requires only one swarming interval while peer 7 needs two intervals in swarming phase).

A complete pattern for delivery of a single data unit in both diffusion and swarming phases is called *delivery tree*. Figure 3 shows a *delivery tree* for a data unit diffused to peer 1, where solid and dashed lines show diffusion and swarming connections, respectively. This figure clearly illustrates that *(i)* the number of swarming intervals that each peer needs to receive a data unit, depends on the location of its swarming parents (*e.g.*, peer 3 receives the data unit after 2 swarming intervals while peer 6 after 1 interval). *(ii)* the depth of a *delivery tree* shows the total number of required intervals for delivery of a data unit to all participating peers (*e.g.*, in Figure 3 depth of delivery tree corresponding to that data unit is 4). For a given overlay, the minimum number of swarming intervals (K_{min}) should be determined such that nearly all peers can receive their maximum quality. This means that the amount of buffering intervals (ω) at each peer should be at least equal to the sum of diffusion and swarming intervals *i.e.*, $\omega >= depth + k_{min}$.

5.2 Packet scheduling

Packet scheduling at individual peers should behave such that their collective effect leads to the desired pattern of content delivery. This in turn minimizes content bottleneck among peers. To achieve this goal, packet scheduling should request 3 groups of

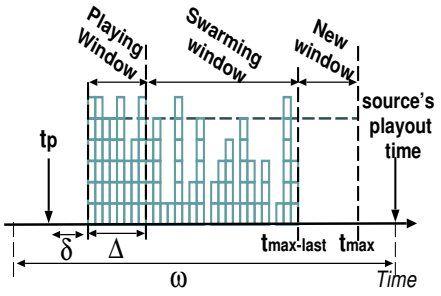


Figure 4: Buffer state at the sliding window time in a individual peer

packets as follows:

- *New packets*: newly generated packets from the diffusion parent to ensure rapid diffusion of new packets through the overlay.
- *Playing packets*: all missing packets that should be played within next Δ seconds from swarming parents to ensure playback requirement.
- *Other packets*: randomly selected packets that are missing but required from swarming parents to ensure content diversity among peers.

To group packets properly, each peer should maintain three windows of packets as shown in Figure 4. In Figure 4, t_p is the playout time and increases by stream rate, t_{max} is the maximum timestamp that is available among parents and $t_{max-last}$ is the t_{max} in the previous sliding window event. The three windows are as follows: *(i) The new window [$t_{max-last}$,t_{max}]:* This window consists of packets that are newly generated and are available at the diffusion parent(s).
(ii) The Playing window [t_p+δ,t_p+δ+Δ]: This window consists of all packets that should be played within next Δ seconds.
(iii) The swarming window [t_p+δ+Δ,$t_{max-last}$]: This window consists of packets that are in their swarming phase.
Windows slide in a step like fashion every Δ seconds. The packet budget for parent i could be estimated based on its exponential weighted moving average (ewma) bandwidth : $\frac{ewmabw_i * \Delta}{Packet-size}$. Each packet consists of a timestamp and a description. We use the term *useful packet* for a packet that is required by a child peer and is available at some parents of the peer.

Packet scheduling mechanism should identify timestamps, description of requested packets as well as their mapping to the parents. The scheduler selects timestamps in three steps and then identifies the description and finally the corresponding parent for each packet. [2] Timestamps of the packets are selected in three steps as follows:

1. Select all *useful packets* from the playing window to ensure playback continuity.
2. Select all *useful packets* in the new window to ensure rapid diffusion of new data unit.
3. Select all *useful packets* from the swarming window in a random order to exchange/swarm other data units.

After selecting timestamps, packet scheduling identifies description and corresponding parent for each selected timestamp. This can be done in two different methods/order for each timestamp ts, $parent - first$ and $description - first$. Within each method/order selection of the parent and description are based on some criteria. The two methods are:
$parent - first$*:* This scheme, first selects a parent i with some criteria from possibly multiple parents that have ts and have enough packet budget. Then packet scheduling selects a useful description among

[2] When a peer requires k new description for a particular timestamp, that timestamp could appear k times in the final list

available descriptions in the parent i for timestamp ts.
$description - first$: This scheme, first selects a description $Desc$ with some criteria from possibly multiple descriptions that are useful. Then it selects a parent i that can deliver the packet considering its packet budget.
In these mentioned methods, the selection of parents or descriptions can be done in different ways with different criteria. The criteria for description selection could be:
Random or rarest (among available descriptions for that timestamp).
The criteria for parent selection could be:
Random or a parent with minimum ratio of the assigned packets to packet budget. The latter proportionally balances the assigned packets among parents. These choices result in different packet scheduling mechanisms. In Section VII-B, we look at the effect of different packet scheduling mechanisms.

6 Source behavior

Source plays a key role in controlling the diffused content to different diffusion subtrees (*i.e.*, level 1 peers). The maximum available quality in the system is determined by the average number of descriptions for each timestamp that are delivered from the source to all peers in level 1. The delivered quality to level 1 is determined by *(i)* the aggregate throughput from source to its child peers and *(ii)* the number of unique descriptions from each timestamp that are delivered to peers in level 1. The aggregate throughput from source is a function of its outgoing access link bandwidth coupled with its outgoing degree which is determined by $bandwidth - degree$ condition. The number of unique descriptions from each timestamps and even distribution of diffused packets in the overlay depend on requested packets by peers in level 1. Due to independent requests by these peers from source, some packets may never be requested while some other packets may be requested multiple times. Thus source should coordinate among its children to minimize the redundancy in the requested packets. This in turn *(i)* maximizes the delivered quality to level 1 (guarantee the diffusion of at least one copy of all packets through the overlay) and *(ii)* evenly distributes delivered packets across different timestamps and descriptions. We introduce the term *diffusion rate* as the rate of delivery for new bits to peers in level 1. Ideally, *diffusion rate* should be equal to the stream quality and the number of copies to level 1 should be fairly even. To achieve this goal, source should perform *packet swapping*. By packet swapping source can minimize the overlap among delivered data units and evenly distribute delivered packets to different peers in level 1 which are roots of diffusion subtrees.

Packet swapping works as follows: Source keeps track of the number of delivered copies for each packet. Any requested packet with timestamp ts that has already been delivered, is swapped with a packet with the minimum number of delivered copies within timestamp window of [$ts-\Delta$,ts]. This strategy increases diffused quality from source through the overlay. However, because of packet loss source can not correctly estimate number of delivered copies for each packet. Therefore source implements *loss detection* by keeping track of the number of successfully delivered packets to peers in level 1. We examine the source behavior with or without packet swapping and loss detection in evaluation Section VII-C.

7 Performance Evaluation

We use ns-2 simulator to examine the effect of the following key parameters on PRIME: *(i)* Overlay

$Size$	200
$Bottleneck\ location$	edge
δ	0.3 sec
Δ	6 sec
$Delay\ range\ (accesslink)$	5ms-25ms
C	160 kbps
$NodeBW$	700 kbps/1.5 Mbps
$SRCBW$	900 kbps/1.8 Mbps
$Maximum\ Quality$	5/10 descriptions

Table 1: Summary of default simulation parameters

properties, *(ii)* Packet scheduling, *(iii)* Source behavior and *(iv)* Peer population. A packet level simulator enables us to properly examine the effect of packet level dynamics and packet loss. The topology in our simulations generated by Brite [34] with 15 AS and 10 routers per AS in a top-down model. We use RED queue management at all routers. The delay on each access link is randomly selected between [5ms, 25ms]. The bottlenecks are at the edge of the network by over provisioning the core of the network. We focus on the steady state behavior of the system in our results. We have run each simulation with different seeds over different random overlays and the results were similar. We have used RAP [33] as a congestion control mechanism for all connections. We assume that all descriptions of the MDC stream have the same bit rate of C = 160 kbps.

In all simulations, we enforced the $bandwidth - degree$ condition and all access links are symmetrical. We set $\Delta = 6$ seconds. 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. In most of the simulations, we use two default scenarios with 200 homogeneous peers with access link bandwidth of *(i)* 700 kbps or *(ii)* 1.5 Mbps. Table I summarizes the default values of parameters in simulations

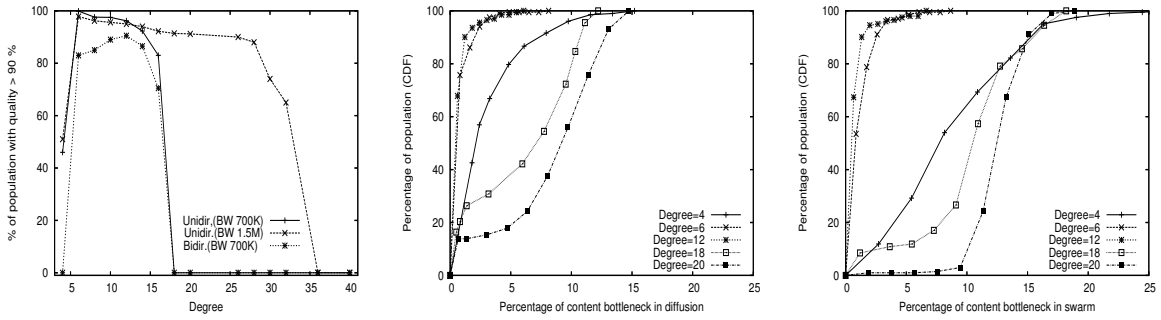
7.1 Overlay properties

First we examine the effect of overlay properties on the performance of content delivery in PRIME. To separate the effect of other factors, we choose the best packet scheduling mechanism and ensure that the delivered quality to level 1 is equal to the maximum stream quality. The maximum stream quality generated by source is equal to the delivered quality to the peers with highest access link bandwidth in each scenario.

7.1.1 Peer Degree and Bandwidth-to-Degree Ratio (*bwpf*)

We examine the impact of peer degree and *bwpf* in two default scenarios with 700 kbps and 1.5 Mbps bandwidth. Figure 5(a) shows the percentage of peers that receive at least 90% of the stream quality as a function of incoming peer degree. Note that increasing peer degree decreases the depth of the overlay. Therefore, for a fair comparison across different degrees, we keep the number of swarming intervals (K) fixed at 3 by setting the ω to $depth$+3. Figure 5(a) shows two interesting points: *(i)* There is a sweet range of peer degree based on peer bandwidth that the system performs best. *(ii)* The lower bound of this range does not depend on peer's bandwidth (*e.g.*, degree 6 is the start of good performance region for both bandwidth 700 kbps and 1.5 Mbps).

When the degree is very low, regardless of peer's bandwidth, delivered quality to half of the participating peers is less that 90% of the maximum quality. This is due to the lack of diversity between swarming parents. With a fixed peer population by decreasing peer degree, the number of diffusion subtrees decreases, thus the population of peers in each diffusion subtree increases. Therefore, the probability of having swarming parents on different subtrees increases. This in turn increases the duration



(a) Effect of *bwpf* on System Performance

(b) Content Bottleneck in Diffusion

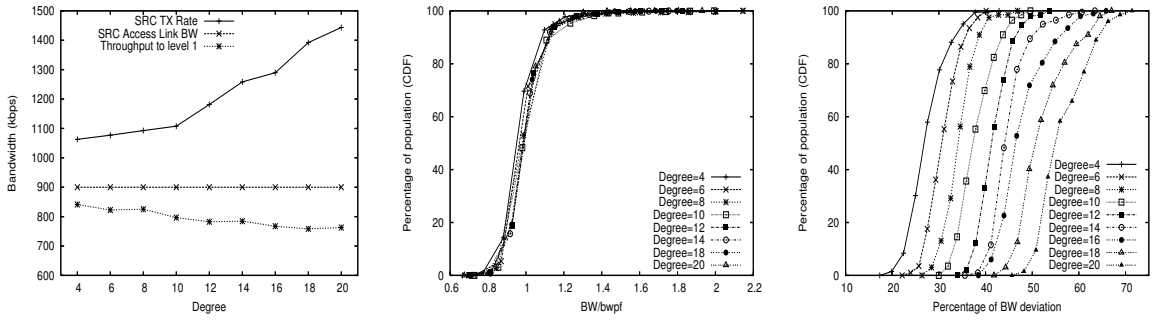
(c) Content Bottleneck in Swarming

Figure 5: Effect of *bwpf* on overall performance

of swarming phase. Clearly, this does not depend on peer's bandwidth. On the other hand, by increasing peer degree beyond a threshold, we observe a rapid drop in the delivered quality. This is due to a significant increase in loss rate for individual connections which agrees with the TCP equation. Figure 5(a) clearly shows that for high bandwidth scenario with 1.5 Mbps, the maximum degree is proportionally larger than the maximum degree for low bandwidth scenario of 700 kbps. This results in a wider operating region for high bandwidth scenario.

Figure 5(b) and 5(c) show the CDF of content bottlenecks from diffusion and swarming parents among participating peers for a few degrees, in 700 kbps scenario, respectively. The percentage of content bottlenecks from diffusion (or swarming) parent(s) is the percentage of available bandwidth of the diffusion (or swarming) parent(s) that can not be utilized. These figures show three points: *(i)* Overall, the percentage of content bottlenecks from swarming parents are higher than diffusion parents as we have discussed in Section V. *(ii)* By increasing the peer degree from 4 to 6, percentage of content bottleneck rapidly decreases due to higher diversity among parents. *(iii)* By increasing the peer degree beyond a threshold, loss factor becomes dominant and affects the content bottleneck in both diffusion and swarming phases.

Loss Rate: To investigate the effect of peer degree on loss rate, we focus on loss rate of connection from source as a representative peer in the overlay in the scenario with 700 kbps. Figure 6(a) shows the aggregate transmission rate from the source to all of its child peers, along with its access link utilization and its aggregate throughput to its child peers for different peer degree. This figure indicates that as degree increases, transmission rate significantly increases while access link utilization remains fixed. The gap between the top two lines represents the loss rate in outgoing link of source while the gap between the bottom two lines shows the aggregate loss rate in the incoming access link of peers in level 1. This clearly illustrates that loss rate at both end increases with the degree. While packet loss mostly occurs at the outgoing link of participating peers, some portion of it also occurs at their incoming links. Intuitively, one expects all losses to occur on outgoing bandwidth of all peers. This raises the question that



(a) Effect of Degree on loss rate

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

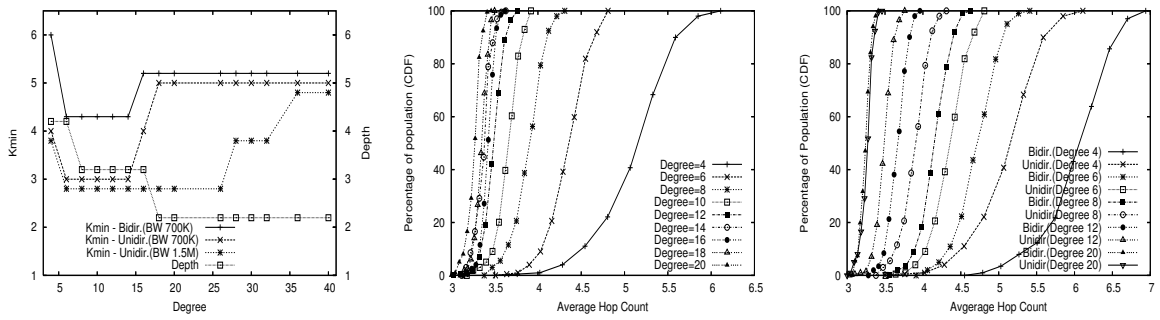
(c) CDF of BW Deviation

Figure 6: Effect of *bwpf* on packet loss

"Why do non-negligible losses occur on the incoming links of various peers?" To answer this we plot the distribution of normalized average bandwidth (by their corresponding $bwpf$) across all connections for different peer degree in Figure 6(b) and normalized deviation of bandwidth across all connections in Figure 6(c). As peer degree increases the distribution of normalized average bandwidth (Figure 6(b)) does not change but its deviation shifts towards larger values (Figure 6(c)). This implies that higher deviations of bandwidth in scenarios with higher degree, result in minor bottleneck (and thus loss) at the incoming link of participating peers.

So far we only examined the performance of the system with fixed K (number of swarming intervals) for a fair comparison. We observed that system does not perform well when degree is low or degree is too large, due to the low diversity and the high loss rate, respectively. This raises the question that *"How many extra swarming intervals are required to achieve higher quality in any scenario?"* Figure 7(a), shows the minimum number of swarming intervals where 90% of peers receive 90% of their maximum deliverable quality (K_{min}) as well as the $depth$ of the overlay as a function of different degrees for scenarios with 700 kbps and 1.5 Mbps. Note that $depth$ is the number of diffusion intervals that does not depend on peer bandwidth and decreases with peer degree. This figure shows that as the degree increases, K_{min} drops from 4 to its minimum value of 3 and then eventually grows to 5. The initial drop in K_{min} is due to the increasing diversity. The increase K_{min} starts later for higher bandwidth scenario due to their larger $bwpf$ at a same peer degree. This suggests that there is a direct relationship between K_{min} and $bwpf$ in each scenario.

Another interesting issue is to capture the pattern of content delivery from the source to individual peers. Figure 7(b) shows the CDF of average path length (in terms of hop count) for delivered packets from source to individual peers for different peer degrees. This figure shows that by increasing the degree the average path length decreases due to decrease in $depth$. On the other hand, the distribution of average path length becomes more homogeneous due to the increase in diversity among parents and richer connectivity among diffusion subtrees. This figure also supports our explanation for low perfor-



(a) Minimum K and depth

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

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

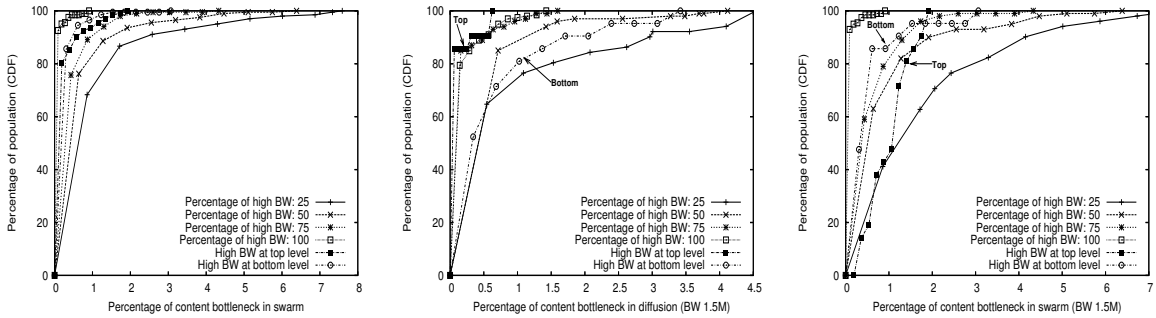
Figure 7: Effect of peer connectivity on system performance

mance in high degree scenarios. More specifically it shows that lost packets are requested from the same swarming parent again in the following swarming interval rather than through a longer path from other swarming parents.

7.1.2 Bi- vs. Uni-directional Connectivity

In this section, we look at the effect of bi- vs. uni-directional connectivity between peers. We examine the default scenario with 700 kbps peer bandwidth but enforced bi-directional connections between peers. The percentage of peers that receive 90% of stream quality is shown (with the lowest line) in Figure 5(a). This figure shows that the percentage of peers with high quality in a bi-directional overlay is 10%-20% lower than the uni-directional overlay. We have also shown the minimum number of swarming intervals (K_{min}) for this scenario in Figure 7(a). Figure 7(a) reveals that with bi-directional connections, at lease one extra swarming interval is required for peer degrees between 4 and 16. To explain this, note that in bi-directional overlays, for each diffusion connection from a parent to a child, there is a swarming connection from the child to the parent. This implies that all swarming connections from child peers to their parents can not provide a new data unit during the first swarming interval for the corresponding segment. We call these inefficient swarming connections. This primarily affects peers in higher levels since they need more swarming intervals as the packets should diffuse to the lowest level $depth$ and then swarm through the reverse connections to reach to the higher levels. The number of inefficient swarming connections that falls within the same diffusion subtrees is at least equal to the number of diffusion connections (roughly the same as peer population) in bi-directional scenario. When peer degree and therefore total number of connections is small, the larger fraction of swarming connections are inefficient. On the other hand, by increasing peer degree the extra connections establish useful swarming connections while the number of inefficient connections remains roughly the same.

Figure 7(c) shows CDF of average path length in bi-directional overlays (as well as the corresponding uni-directional overlays). The figure indicates that for small degree of 4, the average path length is 20% larger for bi-directional overlay compare to



(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)

Figure 8: Content bottleneck for high bandwidth peers in scenarios with heterogeneous bandwidth

uni-directional overlay. However, by increasing degree the gap in the path length between bi- and uni-directional overlays decreases.

7.1.3 Bandwidth Heterogeneity

Our goal is to answer the following questions of bandwidth heterogeneity: *(i) "How are the delivered quality and buffer requirements of high bandwidth peers affected by degree of bandwidth heterogeneity and the percentage of low bandwidth peers?"* and *(ii) "How does the location of high bandwidth peers in the overlay affect the percentage of content bottleneck among them?"* To examine the effect of heterogeneity of bandwidth, we consider the default scenario with peers' bandwidth of 1.5 Mbps (bw_h) and reduce the bandwidth of K_l percent of peers to bw_l.

Degree of heterogeneity: By enforcing the $bandwidth - degree$ condition, access link bandwidth utilization of all groups of peers remains high. The probability of content bottleneck for low bandwidth peers also decreases due to the fact that the available quality in the system and thus among their swarming parents is higher. We further discuss this issue in Section VII-C. Figure 8(a) shows the CDF for the percentage of content bottleneck from swarming parents among all peers where different percentage of peers (0%, 25%, 50%, 75%) are replaced with low bandwidth peers with 1 Mbps. This figure clearly shows that the percentage of high bandwidth peers does not have a significant effect on content bottleneck during swarming phase and thus, does not affect the overall performance of the system. Keeping the same $bwpf$ for different scenarios implies that by decreasing the number of high bandwidth peers, $depth$ of the overlay increases. Thus, when the percentage of high bandwidth peers is low, there is a minor increase in content bottleneck from diffusion parents as shown in Figures 8(b) (this is only for high bandwidth peers).

Figure 8(c) shows a *minor* increase in the percentage of content bottleneck from swarming parents when percentage of high bandwidth peers decreases. The reason is that the percentage of content bottleneck at each peer depends on the aggregate available quality of its swarming parents. When the percent-

Notation	Method	Parent selection criteria	Description selection criteria
$ParentMin. - Desc.Rare$	Parent first	Minimum $\frac{assigned}{totalbudget}$	Rarest
$ParentMin. - Desc.Random$	Parent first	Minimum $\frac{assigned}{totalbudget}$	Random
$ParentRandom - Desc.Random$	Parent first	Random	Random
$ParentRandom - Desc.Rare$	Parent first	Random	Rarest
$Desc.Rare - ParentMin.$	Description first	Minimum $\frac{assigned}{totalbudget}$	Rarest
$Desc.Random - ParentMin.$	Description first	Minimum $\frac{assigned}{totalbudget}$	Random

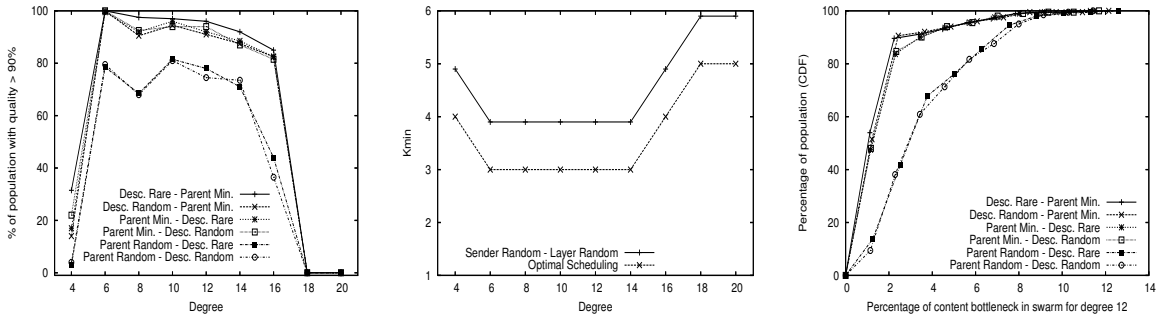
Table 2: Summary of different packet scheduling mechanisms

age of high bandwidth peers is small, a larger fraction of their swarming parents are low bandwidth peers with lower available quality. We have examined other heterogeneous scenarios by keeping the amount of the resources fixed with various combinations of peer bandwidth and observed the similar results.

Location of High bandwidth peers: To answer the second question, we explore a scenario with heterogeneous bandwidth where only 10% of peer population are high bandwidth peers. We first place them in level 1 and then place all high bandwidth peers in the lowest level. Figures 8(b) and 8(c) show the percentage of content bottleneck for these two cases (for clarity lines represented locations are shown by specific arrows of Top and Bottom) that allow comparison with other scenarios. Placing the high bandwidth peers in the top level slightly reduces the $depth$ of the overlay decreases, due to their larger degree. Therefore as Figure 8(b) shows (for clarity lines represented locations are shown by specific arrows of Top and Bottom) percentage of content bottleneck in diffusion is slightly smaller when high bandwidth peers are in the top level. On the other hand placing high bandwidth peers at the bottom level leads to a minor decrease in swarming interval due to a larger number of swarming connections. In other words, the depth of the *delivery tree* (as shown in Figure 3) and thus the total number of intervals for delivery of a segment (ω) does not depend on the location of high bandwidth peers. However, the location of high bandwidth peers could slightly decrease the duration of one phase while results in an equal increase in the other phase of content delivery. In summary, *the location of high bandwidth peers does not have a significant impact on the minimum required buffer* (ω).

7.2 Packet Scheduling

In this subsection, we examine the effect of packet scheduling mechanism at individual peers on the overall system performance. As we've discussed in Section V-B, the collective behavior of packet scheduling at individual peers determines the global pattern of content delivery. Consider the default scenario with access link bandwidth of 700 kbps where all peers use the same packet scheduling mechanism. We examine the following scheduling mechanisms:
(i) $ParentMin. - Desc.Rare$,
(ii) $ParentMin. - Desc.Random$,
(iii) $ParentRandom - Desc.Random$,
(iv) $ParentRandom - Desc.Rare$,
(v) $Desc.Rare - ParentMin.$ and
(vi) $Desc.Random - ParentMin.$.
Table II summarizes the notations based on the method that is used and the corresponding criteria for different packet scheduling mechanisms. Figure 9(a) depicts the percentage of peers that receive 90% of the stream quality for six different packet scheduling mechanisms where $\omega = depth + 3$ (*i.e.*, $K = 3$). This figure shows two key points: first, except for the random selection of par-



(a) Effect of packet scheduling on system performance

(b) Packet scheduling- minimum k

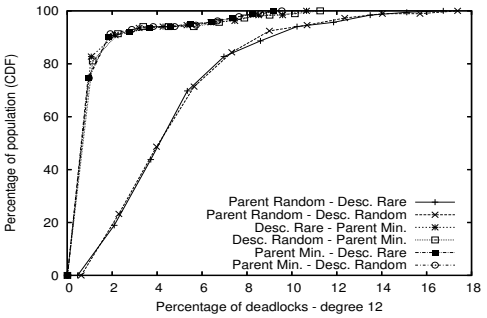
(c) Percentage of Content bottleneck in swarm phase, Degree 12

Figure 9: Effect of different packet scheduling mechanisms

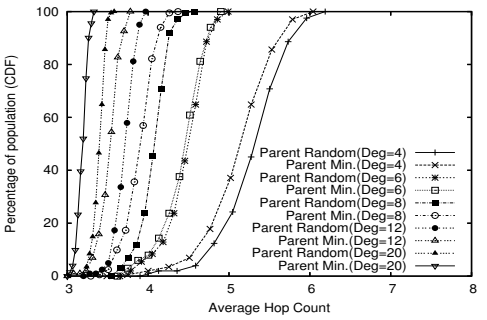
ents, all other mechanisms roughly exhibit the same behavior. This implies that neither the selection order (between parent and description) nor the selection criteria for description significantly affects the overall performance. Second, the random selection of parents regardless of the selection criteria for description performs roughly 20% lower than other mechanisms. Figure 9(b) shows the minimum swarming intervals (K_{min}) where 90% of peers receive 90% of their maximum deliverable quality. This figure revealed that the K_{min} value for $ParentRandom$ mechanisms is always one interval larger than other scheduling mechanisms in a comparable scenario. Figure 9(c) presents CDF of percentage of content bottleneck from swarming parents for different packet scheduling mechanisms for degree of 12. This figure shows that the percentage of content bottleneck from swarming parents for $ParentRandom$ selection mechanisms is significantly larger than other mechanisms.

To verify the underlying reasons of higher content bottleneck with $ParentRandom$ scheduling mechanisms, we define a notion of $deadlock$ in packet scheduling. A $deadlock$ occurs when a packet is available in some parents but the scheduling mechanism can not map it to the corresponding parent(s) since the packet budget of the parent(s) who can serve the packet is fully utilized for delivery of other packets. Figure 10(a) shows the distribution of fraction of packets whose scheduling leads to $deadlock$ among participating peers for peer degree of 12. It shows that $ParentRandom$ mechanism observes significantly more $deadlock$s. In a $ParentRandom$ mechanism, note that all the new data units of a parent may not be requested, since a portion of parent's packet budget may be used for requesting packets that are available in other parent(s).

Our finding raises the question that *"Are packets that experience* $deadlock$ *received through a longer path (*i.e.*, from a different parent in the next swarming interval)?"* Figure 10(b) depicts the distribution of average path length for two schemes of: $ParentRandom$ - $Desc.Random$ and $ParentMin.$ - $Desc.Random$ across different peer degrees. Interestingly, there is no major difference between these two schemes in their average path length across different peer degrees. This implies that the extra interval for the swarming phase is required to compensate for the bad scheduling and receive the $deadlock$ed packets from the same parents

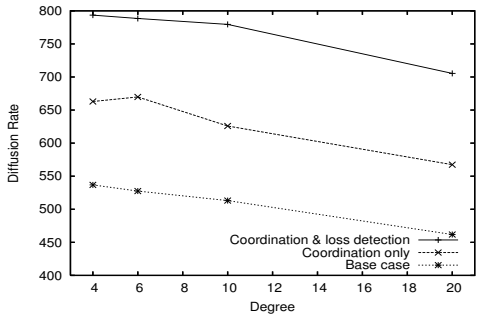


(a) Distribution of percentage of deadlocks for degree 12

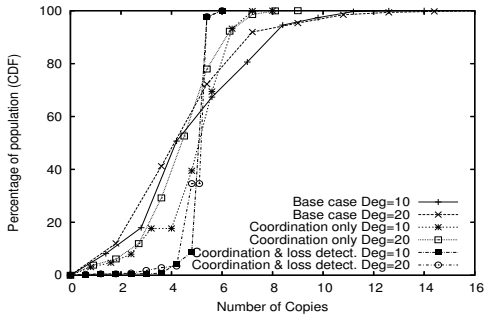


(b) Distribution of Average path length

Figure 10: Effect of various packet scheduling mechanisms on percentage of deadlocks and average path length



(a) Effect of Source coordination on Diffusion



(b) Distribution of packets in level 1

Figure 11: Source Coordination

in the next interval.

7.3 Source Behavior & Properties

In this section, we investigate the effect of source behavior on system performance. First, we look at the effect of source coordination, namely *packet swapping* and *loss detection*. Then, we explore the effect of source bandwidth on overall performance.

7.3.1 Coordination (packet swapping) and Loss Detection

We examine the effect of source coordination in the default scenario with 700 kbps link bandwidth where source bandwidth is 900 kbps and $\omega = depth + 5$ to that ensure all peers receive a high quality stream. Figure 11(a) shows the *diffusion rate* (delivered quality to level 1) as a function of peer degree in three different cases: *(i)* source without any coordination (base case), *(ii)* source with coordination only (*packet swapping*) and *(iii)* source with coordination as well as *loss detection*. This figure shows while the *diffusion rate* slowly decreases with peer

degree, incorporating *packet swapping* significantly increases the *diffusion rate* and adding *loss detection* leads to further improvement. Note that source throughput is not a function of source coordination and the decrease in *diffusion rate* is due to higher redundancy among delivered packets to level 1. Figure 11(b) shows the distribution of the number of delivered copies for each packets to level 1. This figure shows that the CDF becomes progressively uniform across different packets. By incorporating both *packet swapping* and *loss detection*, the number of delivered copies to level 1 reaches to perfectly even distribution across delivered packets. *Incorporating these two mechanisms enables the source to deliver the highest achievable quality (considering packet losses) for a given bandwidth to peers in level 1, and thus the system.*

7.3.2 Source Bandwidth

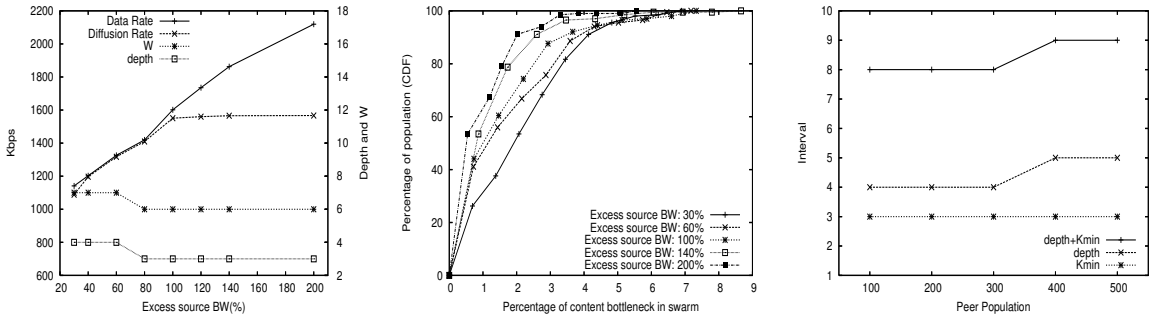
In this subsection, we examine the effect of source bandwidth on delivered quality and buffer requirement at individual peers. We use the default scenario with peer degree of 6 and examine the effect of excess source bandwidth (beyond the peer bandwidth of 700 kbps). Figure 12(a) shows the effect of source bandwidth on its aggregate throughput to level 1, *diffusion rate* (delivered quality to level 1), ω and $depth$. Values on the X axis represent the normalized values of the excess source bandwidth, *i.e.*, $\frac{SourceBW-700kbps}{700kbps}$. This figure reveals two points as follows: *(i)* because of enforcement of $bandwidth - degree$ condition, increasing source bandwidth increases the source degree. This in turn, results to gradual decrease in the $depth$ of the overlay (*i.e.*, smaller number of diffusion intervals) and decreases the buffer requirement (ω). *(ii)* increasing source bandwidth increases *diffusion rate* up to the maximum stream quality . Further increase of source bandwidth beyond this point does not affect the *diffusion rate*, but the aggregate throughput from source to level 1 increases proportionally. Increase in the *diffusion rate* increases the number of diffusion subtrees with unique content and thus improves the diversity of swarming parents. As a results the percentage of content bottleneck among peers from swarming parents slightly reduces with source bandwidth which is as shown in Figure 12(b). Once the delivered quality to level 1 reaches the maximum stream quality further increase in source bandwidth results in adding redundant diffusion subtrees without affecting the diversity of their content.

7.4 Peer Population

The final issue to explain is the scalability of PRIME with peer population. Figure 12(c) shows $depth$, K_{min} and ω as a function of peer population in the default scenario with link bandwidth of 700 kbps and peer degree of 6. This figure shows that as the population increases, $depth$ of the overlay gradually increases but duration of swarming phase (K_{min}) remains constant. This is due to the fixed number of diffusion subtrees for different peer populations which results in the same diversity among swarming parents. *This implies that a system that operates in the sweet range of peer degree ($bwpf$), can easily accommodate a larger peer population by increasing peer buffer size proportional to the $depth$ of the overlay.*

8 Conclusion

Incorporating swarm-like content delivery into mesh-based live P2P streaming in scalable fashion is a challenging task. In this paper, we presented the design of PRIME, a novel mesh-based live P2P streaming protocol. We discussed the key design issues of PRIME and identified two key performance



(a) Effect of Source BW

(b) Effect of Source BW on bottleneck in swarming

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

Figure 12: Source Bandwidth & Scalability

bottlenecks in content delivery. Then we proposed a global pattern of content delivery along with a corresponding packet scheduling that leads to effective utilization of out-going bandwidth across all participating peers. Through extensive performance evaluation, we showed important design tradeoffs and relationships among key parameters. More specifically, we showed that there is a sweet range of bandwidth per connection, over which most of the participating peers are able to obtain maximum deliverable quality with minimum buffer (playback delay) requirement in a scalable fashion.

As a future work, we plan to compare the performance of different class of live P2P streaming mechanisms namely tree-based approach and mesh-tree approach to quantify their differences across the evaluation space. Furthermore we plan to examine the performance of PRIME with the presence of churn.

References

[1] B. Alfonsi, "I want my iptv: Internet protocol television. predicted a winner," in *IEEE Distributed Systems Online*, 2005.

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

[3] 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.

[4] 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.

[5] X. Jiang, Y. Dong, D. Xu, and B. Bhargava, "GnuStream: A P2P Media Streaming System Prototype," in *IEEE International Conference in Multimedia and Expo*, 2003.

[6] D. A. Tran, K. A. Hua, and T. Do, "Zigzag: An efficient peer-to-peer scheme for media streaming," in *IEEE INFOCOM*, 2003.

[7] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *Proceedings of the ACM SIGCOMM*, 2002.

[8] Y. Guo, K. Suh, J. Kurose, and D. Towsley, "P2cast: Peer-to-peer patching scheme for vod service," in *Proceedings of World Wide Web Conference*, 2003.

[9] H. Deshpande, M. Bawa, and H. Garcia-Molina, "Streaming live media over a peer-to-peer network," Tech. Rep., 2001.

[10] 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.

[11] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan, "Resilient overlays using multicast," in *ACM/IEEE Transactions on Networking*, 2005.

[12] M. Yang and Z. Fe, "A proactive approach to reconstructing overlay multicast trees," in *IEEE INFOCOM*, 2004.

[13] A. Fei, J. Cui, M. Gerla, and D. Cavendish, "A dual-tree scheme for fault-tolerant multicast," in *IEEE International Conference on Communications*, 2001.

[14] W. Jia, W. Zhao, D. Xuan, and G. Xu, "An efficient fault-tolerant multicast routing protocol with core-based tree techniques," in *IEEE International Conference on Communications*, 1999.

[15] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava, "Promise: Peer-to-peer media streaming using collectcast," in *ACM Multimedia*, 2003.

[16] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High bandwidth data dissemination using an overlay mesh," in *SOSP*, 2003.

[17] X. Liao, H. Jin, Y. Liu, L. M. Ni, and D. Deng, "Anysee: Scalable live streaming service based on inter-overlay optimization," in *IEEE INFOCOM*, 2006.

[18] S. Annapureddy, C. Gkantsidis, P. Rodriquez, and L. Massoulie, "Providing video-on-demand using peer-to-peer networks," in *Internet Protocol TeleVision (IPTV) workshop in conjunction with WWW*, 2006.

[19] D. Kostic, A. Rodriguez, J. Albrecht, A. Bhirud, and A. Vahdat, "Using Random Subsets to Build Scalable Network Services," in *USENIX Symposium on Internet Technologies and Systems*, 2003.

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

[21] A. Bharambe, C. Herley, and V. Padmanabhan, "Analyzing and improving bittorrent performance," Microsoft Research, Tech. Rep., 2005.

[22] C. Dana, D. Li, D. Harrison, and C. Chuah, "Bass: Bittorrent assisted streaming system for video-on-demand," in *IEEE Signal Processing Soc. Workshop on Multimedia Signal Processing*, 2005.

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

[24] F. Pianese, J. Keller, and E. W. Biersack, "Pulse, a flexible p2p live streaming system," in *Global Internet Workshop*, 2006.

[25] W. T. Ooi, "Dagster: Contributor Aware End-Host Multicast for Media Streaming in Heterogeneous Environment," in *SPIE Multimedia Computing and Networking*, Jan. 2005.

[26] K. Sripanidkulchai, A. Ganjam, and B. Maggs, "The Feasibility of Supporting Large-Scale Live Streaming Applications with Dynamic Application End-Points," in *ACM SIGCOMM*, 2004.

[27] M. Guo and M. Ammar, "Scalable live video streaming to cooperative clients using time shifting and video patching," in *IEEE INFOCOM*, 2004.

[28] Y. Cui, B. Li, and K. Nahrstedt, "oStream: Asynchronous Streaming Multicast in Application-Layer Overlay Networks," in *IEEE Journal of Selected Areas in Communication*, 2004.

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

[30] M. Adler, R. Kumar, K. W. Ross, D. Rubenstein, T. Suel, and D. D. Yao, "Optimal Peer Selection for P2P Downloading and Streaming," in *IEEE INFOCOM*, 2005.

[31] J. Liu, B. Li, T. Hou, and I. Chlamtac, "Dynamic Layering and Bandwidth Allocation for Multi-Session Video Broadcasting with General Utility Functions," in *IEEE INFOCOM*, 2003.

[32] N. Magharei and R. Rejaie, "Adaptive Receiver-Driven Streaming from Multiple Senders," *ACM Multimedia Systems Journal*, 2005.

[33] 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.

[34] 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.