Understanding Mesh-based Peer-to-Peer Streaming

Nazanin Magharei, Reza Rejaie
Department of Computer and Information Science
University of Oregon
{*nazanin,reza*}*@cs.uoregon.edu*

ABSTRACT

A common approach to peer-to-peer (P2P) streaming is to form a tree-based overlay coupled with push content delivery. This approach cannot effectively utilize the outgoing bandwidth of participating peers, and therefore it is not self-scaling. In contrast, swarm-like content delivery mechanisms exhibit the self-scaling property but incorporating them into *live* P2P streaming applications are challenging for two reasons: *(i)* in-time requirement of content delivery and *(ii)* the limited availability of future content.

In this paper, we examine the key design issues and tradeoffs in incorporating swarm-like content delivery into mesh-based P2P streaming of *live* content. We show how overlay properties and the global pattern of content delivery could lead to the bandwidth and content bottlenecks among peers, respectively. Leveraging an organized view of the overlay, we present a global pattern for streaming content over a mesh-based overlay that can effectively utilize the outgoing bandwidth of most participating peers. We conduct ns simulation to explore the impact of overlay properties on the global pattern of content delivery and thus delivered quality to individual peers. In particular, we show that for a given scenario, there is a sweet range for peer degree in the overlay that maximizes delivered quality to individual peers with minimum buffer requirement at each peer.

1. INTRODUCTION

Peer-to-Peer (P2P) overlays offer a promising approach to support one-to-many multimedia streaming applications 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 despite the heterogeneity and asymmetry of their access link bandwidth. To be truly "self-scaling", a P2P streaming mechanism should be able to effectively utilize outgoing bandwidth of most participating peers. This means that each peer should always be able to provide useful content to its connected peers in the overlay. In a nutshell, achieving self-scaling depends not only on the properties of the overlay topology but also on the global pattern of content delivery through the overlay.

A common 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 has 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 this approach organizes participating peers into multiple diverse trees. Then each description of a multiple description coded stream is pushed through one of the trees (*e.g.,* [3]). 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 to trees limit the delivered quality to individual peers.

The limitations of the tree-based overlay with push content delivery have motivated a new approach where participating peers form a mesh-based overlay and incorporate a swarm-like content delivery. This approach is inspired by file-swarming mechanisms (*e.g.,* BitTorrent or Bullet [2]). Leveraging the availability of the entire file, file-swarming mechanisms distribute pieces of a file among different peers which enables most peers to actively contribute their outgoing bandwidth. Incorporating swarm-like delivery into live P2P streaming applications is challenging due to two important reasons: *(i)* the streaming constraint of in-time arrival of individual packets, and *(ii)* the limited availability of future content in live streaming applications. A couple of recent studies (*e.g.,* [6]) have presented a mesh-based P2P streaming mechanism that incorporates swarm-like delivery. However, the following two basic issues about P2P streaming mechanisms have not been addressed:

- What is the global pattern for streaming *live* content over a mesh-based overlay that can effectively utilize the outgoing bandwidth of most participating peers while ensuring in-time delivery of individual packets despite limited availability of future content?
- How is the delivered quality to individual peers in a P2P streaming mechanism (*i.e.,* the global pattern for streaming) affected by key properties of an overlay mesh such as node degree, bandwidth heterogeneity among peers, and source bandwidth?

NOSSDAV '06 Newport, Rhode Island USA

In this paper, we answer the above two questions in the context of live P2P streaming applications. In Section 2, first we present an overview of mesh-based P2P streaming mechanisms that incorporate swarm-like delivery. Then, we describe an organized view of a randomly connected mesh and identify bandwidth and content bottlenecks as the two key performance bottlenecks in P2P streaming mechanisms. Using this organized view of the overlay, we illustrate how overlay properties and the global pattern of content delivery can minimize the probability of bandwidth and content bottlenecks, respectively. The key contribution of this paper is to present an optimal global pattern for scalable streaming live content over a mesh-based overlay that incorporates a swarm-like delivery to maximize delivered quality to individual peers with minimum buffer requirement at each peer. Understanding such a pattern sheds an insightful light on performance bottlenecks and design tradeoffs for mesh-based P2P streaming mechanisms. In Section 3, we present our preliminary simulation results that illustrate some of the issues and tradeoffs. In particular, we show that there is a sweet range of peer degree over which incorporating swarm-like delivery can maximize delivered quality to individual peers with a minimum buffer requirement at each peer. Section 4 concludes the paper and outlines our future plans.

2. MESH-BASED P2P STREAMING

Before we discuss the design issues in mesh-based P2P streaming mechanisms, we present common assumptions and two key components, namely overlay construction, and content delivery in these systems. To accommodate bandwidth heterogeneity among participating peers, the delivered stream is encoded with a multiple description coding (MDC) scheme at source. All pair-wise connections for content delivery between peers are congestion controlled (*e.g.,* [4]) to properly share resources with co-existing traffic.

Overlay Construction: In a mesh-based P2P streaming mechanism participating peers form a *randomly* connected and *directed* mesh (*i.e.,* unstructured overlay) that is used for content delivery to individual peers. The connection between each pair is uni-directional which means that data is delivered from a parent to a child peer. Except for the source, each peer in the overlay has multiple parents and multiple children. Maintaining such an overlay for content delivery has several advantages as follows: *(i)* overlay construction and maintenance are very simple, *(ii)* connections from different parents to each child peer are more likely to have a diverse path which in turn reduces the probability of a shared bottleneck between these connections, *(iii)* the resulting overlay is very resilient to churn. There are several approaches to form such an overlay. The simplest alternative is to use a bootstrapping node that maintains a list of participating peers and provides a random subset of participants to each new peer.

Content Delivery: Content delivery among peers is performed using push reporting by parents coupled with pull requesting by child peers. Each peer receives content from all of its parents and provides content to all of its child peers in the overlay. As a parent, each peer progressively reports its new packets to all of its child peers. As a child, each peer periodically (*i.e.,* once per Δ) requests a specific set of packets from each parent. Each parent peer simply delivers requested packets by individual child peers in the provided order at the rate that is determined by a congestion control mechanism. The requested packets from each parent are determined by a *packet scheduling* mechanism at each child peer. Given a peer's playout time as well as the available content and available bandwidth among its parents, this receiver-driven packet scheduling mechanism should select requested packets from each parent in order to maximize its delivered quality, *i.e.,* accommodating in-time delivery of requested packets while effectively utilizing available bandwidth from all parents. PALS is an example of such a receiver-driven packet scheduling mechanism from multiple parents [5].

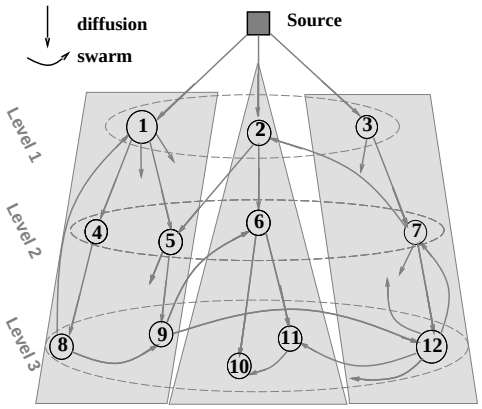


Figure 1: Organized view of a mesh-based overlay with 12 peers. Some connections were not shown for clarity of the figure.

2.1 Organized View of the Overlay Mesh

To discuss the proper pattern of streaming over an unstructured overlay and identify performance bottlenecks for content delivery, we present an "organized view" of peers in a randomly connected and 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 direction of connections in the overlay. Given this definition, peers in the overlay can be organized in to separate *levels* (as shown in Figure 1) based on their distance from source, *i.e.,* level n consists of all peers that are exactly n hops away from source and level 0 is the source itself.

This organized view of the overlay enables us to derive some of its simple but important properties. Suppose that the overlay consists of P homogeneous peers where each peer has the same in- and out-degree of deg and the source degree is deg_{src}. This implies that there are deg_{src} peers in level 1, deg*deg_{src} peers in level 2 and so on. In summary, the population of peers at level n (or $pop(n)$) is limited to $pop(n) \leq deg_{src} * deg^{(n-1)}$. The number of levels, or $depth$, of such an overlay can be simply derived as $log_{deg}(P/deg_{src}) \leq depth$. The exponential distribution of total population across levels coupled with the random connectivity among peers imply that 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 *(i)* a single parent in level $n-1$, and $deg-1$ parents in the same or lower levels, and *(ii)* deg child peers in level $n+1$. Peers in the bottom level ($n = depth$) have a single parent in level $n-1$, and deg child peers in the same or higher levels. In practice, a small fraction of peers may have more than one parent in the higher level due to the random

connectivity among peers (*e.g.,* peer 5 in Figure 1). This in turn reduces the population of peers in their corresponding levels and may slightly increase the depth of the overlay.

2.2 Performance Bottleneck

The main design goal of a P2P streaming mechanism is to maximize delivered quality to individual peers with minimum buffer requirement at each peer while accommodating scalability. Delivered quality to individual peers can be limited for one of the following reasons:

- *A Bandwidth Bottleneck* occurs when the aggregate available bandwidth [1] from all parents to a given peer is not sufficient to fully utilize its incoming access link bandwidth.
- *A Content Bottleneck* occurs when useful content among some parents of a given peer is not sufficient to fully utilize their available bandwidth.

To decouple these two factors, 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 an especially marked packet with the same size. This approach enables each peer to quantify the contribution of bandwidth and content bottlenecks in delivered quality from each parent. In the following subsections, we address the underlying causes for each one of these performance bottlenecks and discuss how the probability of each type of bottleneck can be minimized.

2.3 Addressing Bandwidth Bottleneck

Available bandwidth to each peer *only* depends on properties of the overlay. Suppose that congestion occurs only at the edge of the network, *i.e.,* the incoming/outgoing access links of participating peers. Then 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 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 minimize the probability of bandwidth bottleneck among participating peers while fully utilizing their outgoing bandwidth, the following condition must be satisfied for any parent i and child peer j:

$$bwpf = \frac{outbw_i}{outdeg_i} = \frac{inbw_j}{indeg_j}$$

The random connectivity among peers implies that the ratio of outgoing bandwidth to outgoing degree, and incoming bandwidth to incoming degree should be the same for *all* participating peers. This constant ratio presents the average bandwidth of any connection 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 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 bandwidth of participating peers (and the source) to their incoming and outgoing degrees, respectively. Without loss of generality, we initially assume that accommodating the bandwidth-degree condition ensures that all connections in the overlay have roughly the same bandwidth (*i.e., bwpf*). This simplified view of the problem allows us to illustrate the effect of overlay properties on content delivery more clearly. In subsection 2.6, we consider the scenarios where individual connections have different bandwidth (*e.g.,* due to difference in their RTT) or experience bottleneck inside the network.

[1] Throughout this paper we use the terms congestion control bandwidth and available bandwidth interchangeably.

2.4 Addressing Content Bottleneck

Suppose all connections have roughly the same bandwidth ($bwpf$), then the amount of data that a child peer receives from each one of its parents during one interval (Δ) can be simply estimated as $D = bwpf * \Delta$. We call D 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. To minimize the probability of content bottlenecks in the system, all peers (as a parent) should have at least one useful data unit per interval Δ for each of their child peers. It is worth noting that minimizing content bottlenecks in the system is in essence the same goal as effectively utilizing the outgoing bandwidth of all peers.

In the context of live P2P streaming applications, a new segment of length Δ is generated by the source every Δ seconds where a segment consists of a group of packets with consecutive timestamps ($[t_0, t_0+\Delta]$) from all descriptions. In the absence of content bottlenecks, each peer i can obtain $indeg_i$ unique data units during each interval Δ. Because of the streaming nature of delivery, the required ($indeg_i$) unique data units for each segment must be delivered to peer i within a certain number of intervals (ω) after their generation time. Given an estimate for ω, participating peers can delay their playout time by at least $\omega * \Delta$ second behind source's playout time to ensure in-time delivery of required packets. This means that individual peers should buffer at least $\omega * \Delta$ seconds of content. The availability of useful data units at each parent peer and thus the value of ω, depends on the global pattern of content delivery from the source to all participating peers as we describe in the next subsection.

2.5 Global Pattern of Content Delivery

Our goal is to identify a global pattern of content delivery for each segment of streaming content that minimizes the required number of intervals (ω) for delivery of $indeg_i$ unique data units to all (or nearly all) participating peers. Consecutive segments of the stream can be simply pipelined through the overlay by sequentially following a roughly similar pattern. Participating peers in the system delay their playout time by at least $\omega * \Delta$ seconds behind source's playout time, to ensure in-time delivery of required packets.

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 until each peer has a proper number of data units for the segment. The above observation motivates a

two-phase approach to delivery of a segment as follows:

1) Diffusion Phase of a Segment: Suppose that all peers use the requesting interval of Δ. Once all data units of a new segment become 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 after it becomes available at the source.

To rapidly diffuse a new segment towards peers in lower levels, all the connections between all parent 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 1. 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.

The maximum available quality of each new segment 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 clearly limited by the aggregate throughput from the source to all of its child peers. Furthermore, to maximize the utilization of its throughput the source ensures that the overlap among delivered data units to different peers is minimal.

Assuming that all connections have the same bandwidth ($bwpf$), 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. Then, the new data unit is pulled by all of p's child peers during the next interval. This pattern of content diffusion has two obvious implications: First, peers do not experience content bottlenecks during the diffusion phase, and thus diffusion phase takes exactly $depth$ intervals; Second, each peer p in level 1 as well as all the peers in a sub-tree that is 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 a sub-tree of peers that is rooted in a peer in level 1 is called a *diffusion sub-tree*. The number of diffusion subtrees in an overlay is equal to the population of peers in level 1, or deg_{src}. Figure 1 depicts three diffusion sub-trees with a dark shading.

2) Swarming Phase of a Segment: 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≤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 1. Note that most of the swarming connections are from peers in the bottom level to their child peers in higher levels [2]. This means that the outgoing bandwidth of peers in the bottom level is primarily utilized during the swarming phase of a segment.

To achieve the above pattern of content delivery, the packet scheduling mechanism at each peer should pull data unit(s) of any new segment from its single diffusion parent in the higher level, and then pull ($indeg_i$−1) other data units of the segment from its swarming parents in the same or lower levels in order to maximize its delivered quality for each segment. We recall that all peers in the same diffusion sub-tree receive the same data unit. This implies 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 1, p_{11} can obtain a new data unit from p_{12} but p_7 cannot. 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_{11} in Figure 1. 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_{10} in Figure 1. 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_{10} can receive a new data unit from p_{11} after one interval. The flexibility for each child peer to receive any required packet from any *swarming* parents allows this approach to content delivery mechanism to utilize the outgoing bandwidth of participating peers more effectively than the multi-tree approaches (*e.g.,* CoopNet [3]).

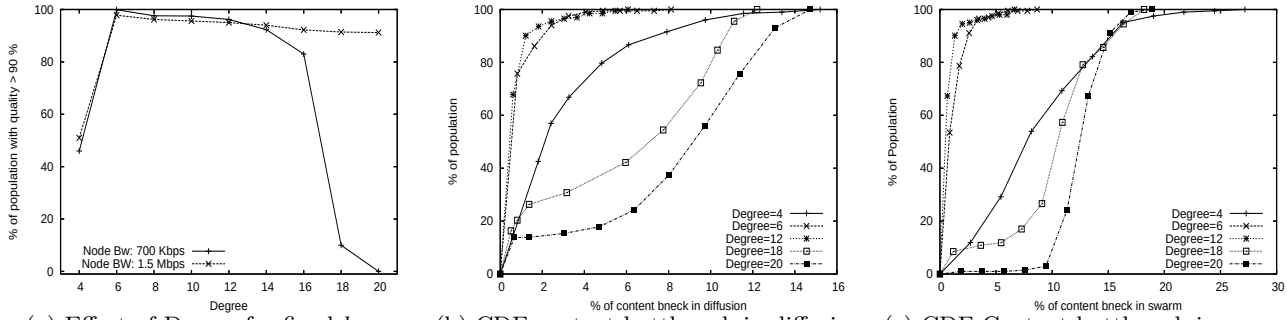
In a nutshell, a peer may complete its swarming phase in one interval or may experience a content bottleneck and thus require more swarming intervals depending on the location of its parents in the overlay. In a randomly connected overlay, the probability of experiencing a content bottleneck among peers 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 a majority of 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})≤ω.

Note that the directed nature of the overlay topology improves diversity of parents across different diffusion sub-trees. More specifically, when all connections in the overlay are bi-directional (similar to the connections in BitTorrent), then parent-child pairs are more likely to be part of the same diffusion sub-tree, *e.g.,* p_7 and p_{12} in Figure 1. This in turn increases the probability of content bottleneck among peers and thus the number of required swarming intervals (k_{min}).

2.6 Practical Considerations

We made two simplifying assumptions that may not hold in practice. First, some connections in the overlay may experience bottlenecks inside the network. This could affect the performance of the child peers that receive content through these connections if their incoming access link bandwidth cannot be fully utilized. This problem can be simply addressed by allowing child peers to have extra parent peers

[2]There is a small fraction of peers in other levels (n<$depth$) that have one or more child in the same or higher levels, *e.g.,* the connection from p_7 to p_2.



(a) Effect of Degree for fixed k_{min} (b) CDF content bottleneck in diffusion (c) CDF Content bottleneck in swarm

Figure 2: Effect of peer degree on delivered quality

beyond the limit that is specified by the bandwidth-degree condition. This enables the affected child peers to improve the utilization of their incoming access link bandwidth.

Second, the bandwidth of individual connections might be higher or lower than $bwpf$. This has two important effects on system performance. *(i)* when any diffusion connection has a bandwidth lower than $bwpf$, the diffusion sub-tree that is fed by this connection is more likely to experience a content bottleneck, *(ii)* more importantly, having different bandwidths for individual connections implies that some connections experience bottlenecks at the parent's access link while others experience bottleneck at the child's access link. Therefore, by increasing the peer degree, the loss rate at both the outgoing link of parent peers and the incoming link of child peers rapidly increase. This in turn leads to a rapid drop in the throughput from the source to level 1 which limits delivered quality to level 1, and thus all other peers. Similarly, the throughput from each peer to all of its children in the overlay drops by increasing the peer degree.

3. EVALUATIONS

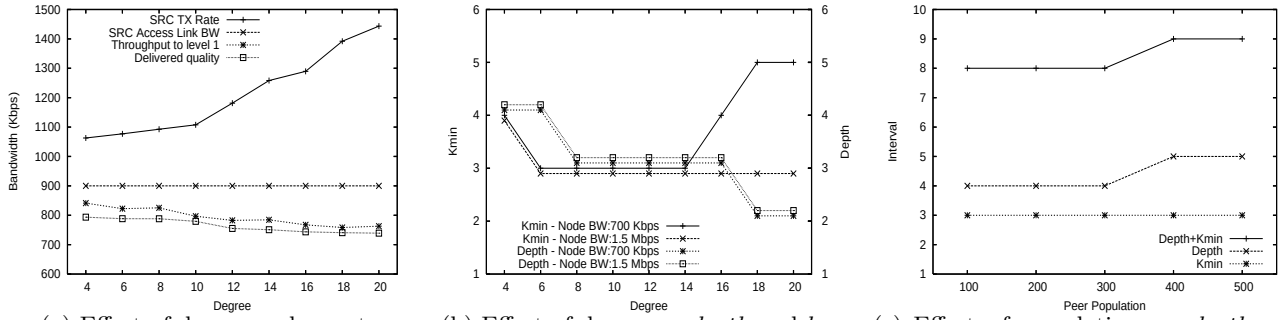
In this section, we use *ns* simulations to examine various design issues and tradeoffs in mesh-based P2P streaming, as we discussed in Section 2. Using a packet level simulator allows us to properly examine the effect of packet level dynamics, packet losses and difference in bandwidths for individual connections, which are not feasible with session level simulators.

Simulation Setup: 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 bottleneck only on the access links. To form a randomly connected and directed overlay, each peer contacts a bootstrapping node to learn about a random subset of participating peers until it identifies the specified number of parents. We did not model churn in our simulation and focused on the pattern of content delivery on the static overlay. The packet scheduling mechanism at individual peers implements the scheme that was described in Subsection 2.5 using requesting interval $\Delta = 6$ sec. Each simulation was run for 400 seconds. The presented results illustrate 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.

Fundamental Design Tradeoff: Given a set of peers with certain incoming/outgoing access link bandwidth, the fundamental question is "*what is a proper bandwidth-to-degree ratio (or $bwpf$) that maximizes the delivered quality to individual peers with minimum buffer requirement at each peer?*". Note that increasing the incoming/outgoing node degree of participating peers (or reducing $bwpf$) has two conflicting effects as follows: On the one hand, the required number of intervals for delivery of a segment (ω) decreases by reducing depth of the overlay and increasing the diversity of swarming connections for individual peers which decreases K_{min}. On the other hand, this will *exponentially* increase observed loss rate and thus decrease throughput of individual connections. These conflicting effects suggest that there is a limited range of peer degree (*i.e.,* sweet range of $bwpf$ values) over which the delivered quality to most peers can be maximized with minimum buffering.

To explore this issue, we examine a scenario with 200 peers with homogeneous and symmetric access link bandwidth for two different bandwidth values, 700 Kbps, and 1.5 Mbps. Figure 2(a) depicts the percentage of participating peers that receive 90% of the maximum deliverable quality as a function of peer degree. For proper comparison, we set the number of swarming intervals to 3 by adjusting the value of ω to $depth$+3 across these simulations. This figure clearly illustrates that there is a sweet range of peer degree (roughly between 6 to 14) where the delivered quality to the majority of peers is high. It also demonstrates that the sweet operating region is slightly wider when peers have higher bandwidth because they reach the lower bound of proper $bwpf$ values at a higher degree. The poor performance with small peer degree is due to the limited number of diffusion sub-trees. However, drop in performance for higher degrees is due to the high loss rate and the resulting drop in delivered quality to level 1.

To shed light on the underlying dynamics of content bottleneck among peers, we show the CDF of the un-utilized fraction of aggregate bandwidth from diffusion and swarming parents across all participating peers in the same simulations in Figure 2(b) and 2(c), respectively. These figures demonstrate that increasing the node degree from 4 to 6 significantly decreases the fraction of connections that experience content bottlenecks in both phases. However, further increase in the degree beyond 12, has a reverse effect and dramatically increases content bottlenecks among peers due to the significant increase in loss rate of individual connections. While the distribution of content bottleneck across swarming and diffusion connections may not be significantly different, the actual number of swarming connections and thus the number of content bottleneck events



(a) Effect of degree on loss rate (b) Effect of degree on *depth* and k_{min} (c) Effect of population on *depth* and k_{min}

Figure 3: Several aspects of performance in mesh-based P2P streaming

during swarming is larger.

Figure 3(a) depicts the aggregate transmission rate, source access link bandwidth, aggregate throughput and delivered quality from source to all peers in level 1 as a function of source degree (shown with lines from top to bottom) in a simulation with 200 homogeneous peers with 700 Kbps access link bandwidth. The difference between the top two lines shows the loss rate at the source access link bandwidth which is dramatically increasing with peer degree. The difference between source access link bandwidth and throughput to level 1 shows the average loss rate across the incoming links of all peers in level 1 which is increasing with the peer degree but at a slower pace. Finally the gap between the throughput and delivered quality to level 1 shows how efficiently source throughput has been utilized. Note that the aggregate transmission rate from each peer to all its children exhibits the same behavior. This figure illustrates the effect of the peer degree on the aggregate loss rate and its break down across different locations between a parent peer and its children throughout the overlay.

Figure 3(b) shows the required number of intervals for diffusion phase (*i.e.,* the overlay depth) and swarming phase (K_{min}) in a scenario with 200 homogeneous peers for two different peer bandwidth values, 700 Kbps and 1.5 Mbps. In essence, this figure presents the minimum buffering requirement at each peer ($\omega = K_{min} + depth$) to achieve high quality for a given peer degree. As expected, the overlay depth is slowly decreasing with the peer degree. Increasing the peer degree initially results in drop in k_{min}. However, further increase of the peer degree beyond a threshold (14 for 700 Kbps and 16 for 1.5 Mbps bandwidth) leads to an increase in k_{min} which eventually results in larger ω. The larger number of swarming intervals for higher peer degrees is mainly due to the rapid increase in loss rate for individual connections.

Scalability: Another key question is how a mesh-based P2P streaming mechanism scales with the number of participating peers. Figure 3(c) shows the effect of peer population on both the overlay depth and k_{min} for a group of homogeneous peers with access link bandwidth of 700 Kbps and peer degree of 6. This figure shows that as peer population increases the overlay depth slowly grows but the required number of swarming intervals (K_{min}) remains unchanged. We note that both the peer degree and the number of diffusion subtrees remain fixed. Increasing the number of participating peers gradually increases the depth of the overlay but it does not have any effect on the probability of content bottleneck and thus it does not change the minimum duration of the swarming phase (K_{min}). To accommodate a larger number of peers, the required buffering at each peer (ω) should gradually increase equal to the growth in the overlay depth.

4. CONCLUSIONS AND FUTURE WORK

This paper examined two key issues in design of scalable mesh-based P2P streaming for live content that incorporates swarm-like delivery: *(i)* what is a global pattern of content delivery that effectively utilizes outgoing bandwidth of participating peers?, *(ii)* how does the connectivity in an overlay mesh affect the delivered quality to individual peers? We leveraged an organized view of a random mesh and presented a global pattern of content delivery that is able to effectively utilize outgoing bandwidth of most peers while maximizing delivered quality with minimum buffer requirement. Using ns simulations, we examined the interactions between overlay properties and the pattern of content delivery in mesh-based streaming and illustrated some of the key design stradeoffs.

We are currently conducting a comprehensive evaluation of mesh-based P2P streaming through simulations. Besides the issues we discussed in this paper, In particular, we examine the effect of heterogeneity in peer properties (*e.g.,* access link bandwidth) and peer behavior (*e.g.,* packet scheduling strategy) as well as churn on overall performance of the system. We are also working on a prototype implementation of a mesh-based P2P streaming mechanism to conduct experiments over PlanetLab in a near future.

5. REFERENCES

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

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

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

[4] 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.

[5] R. Rejaie and A. Ortega. PALS: Peer-to-Peer Adaptive Layered Streaming. In *NOSSDAV*, 2003.

[6] 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.