DynaStream: Adaptive Overlay Management for Peer-to-Peer Video Streaming

Moonkyung Ryu and Umakishore Ramachandran
College of Computing
Georgia Institute of Technology
mkryu@gatech.edu, rama@cc.gatech.edu

***Abstract*—A number of mesh-based peer-to-peer video streaming systems have been proposed, however, they have not paid careful attention to the impact of the number of data streams on streaming quality. In this paper, we first explore the effect of the choice of the number of data streams to the streaming quality. We propose a simple, practical, and fully distributed mechanism, *Loss Rate Window*, that maintains the overlay topology adaptive to peers' uplink bandwidth and network fluctuation. The mechanism allows each node to make an entirely local decision to adaptively control the right number of data streams to support on its uplink without explicitly knowing or measuring its uplink bandwidth. This mechanism is general, and can be applied to any kind of mesh-based peer-to-peer video streaming system to improve the quality of the video streaming service.**

I. Introduction

Peer-to-Peer (P2P) technology has recently emerged as a promising technique for cost-effective and large-scale video streaming on the Internet. In mesh-based P2P video streaming systems [1]–[5], a video is divided into multiple blocks, and a source node generates the blocks. Peers have partial knowledge as to the membership who are watching a given video. A peer exchanges membership information with its neighbors. The peer chooses a subset of the neighbors as partners. A peer establishes a data stream with each partner, and exchanges video blocks with them. The partnership determines the overlay mesh structure. Each peer stores the received blocks in a buffer to deliver them in order to the client application that ultimately plays the video to the user.

The decision for fixing the number of data streams in previous systems has been *ad-hoc*. We claim that the number of data streams should be carefully chosen because it has a very close relationship with the streaming quality. Small number of data streams makes the overlay network sparse, and results in high paths between a source and a destination ultimately leading to the delivery of video blocks past the deadline to the peer. This phenomenon is called *Content Bottleneck* [4], [6]. Too many data streams, on the other hand, might incur network congestion. This phenomenon is called *Bandwidth Bottleneck* [4], [6]. Therefore, appropriate number of data streams should be chosen for each peer to avoid these bottlenecks.

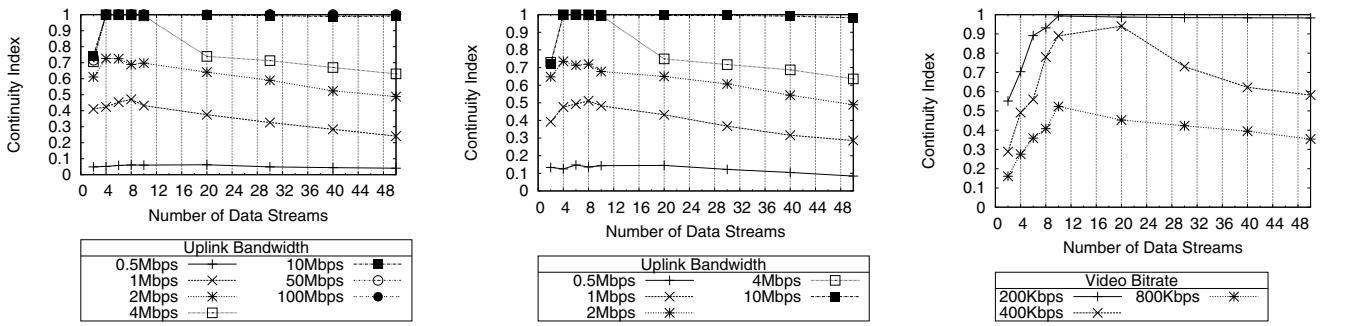
Previous mesh-based systems use *static* or *homogeneous* number of data streams. In this paper, we first investigate how static and homogeneous number of data streams affect the streaming quality over a range of different system parameters. We investigate the relationship between the playback continuity and the number of data streams by studying the bandwidth bottleneck and the content bottleneck. We observe that each peer should have sufficient number of data streams to create a sufficiently dense overlay network, which shorten the length of paths from a source to receivers, thus creating a large number of data flow routes to avoid the content bottleneck. However, too many data streams incur congestion when the peer tries to send larger volumes of data than the available bandwidth of its uplink, which would lead to bandwidth bottleneck. We propose a new P2P video streaming protocol, *DynaStream*, which has a *Loss Rate Window* mechanism at its heart that dynamically controls the number of data streams in the application layer to avoid both the bandwidth bottleneck and the content bottleneck. The mechanism manages the overlay topology to be adaptive to the peers' resources (e.g., access link bandwidth) and to the variations in the underlying network state. It does not need *a priori* knowledge of the peers' access link bandwidth or probing packets to measure the access link bandwidth. We demonstrate that DynaStream achieves better streaming quality than a mesh-based system that has static and homogeneous overlay independent of the system parameters, and improves the streaming quality by a factor of 3 in the worst case. Note that we will use the terms "number of data streams" and "number of partners" interchangeably for the rest of this paper.

The unique contributions of our work are as follows:
(a) We carry out a systematic study, using a simulation framework, to validate the hypothesis that the number of data streams does affect the quality of P2P video streaming.
(b) We establish through the simulation framework that the reasons for the loss of quality is either due to under- or over-commitment of the available network resources.
(c) We propose a novel solution, namely, the *Loss Rate Window* mechanism. Users do not need to care about the number of data streams anymore since the system automatically adjusts the number adapting to the available network resources and to the fluctuating network state.

The rest of the paper is organized as follows. Related work is discussed in Section II. Section III explores how the system works over a large parameter space, raises the issue of the relationship between the number of data streams and the streaming quality, and reveals the crux of the problem that results in loss of video streaming quality. Section IV proposes our solution, *Loss Rate Window*. Section V presents concluding remarks.

II. Related Work

A number of different kinds of mesh-based P2P video streaming systems have been proposed. However, their focus has been different from ours; specifically they have not focused on adaptive control of the number of partners for ensuring good streaming quality. CoolStreaming [1], [2] requires peers to keep a static and homogeneous number of data streams.



(a) Homogeneous Symmetric

(b) Homogeneous Asymmetric. Downlink bandwidth is 5 times the uplink bandwidth.

(c) Heterogeneous Asymmetric. Downlink bandwidth is 5 times the uplink bandwidth.

Fig. 1. Continuity Index for a 200 node system with different bandwidth settings

Group	**Downlink**	**Uplink**	**Number of nodes** (Total=200 nodes)
A	2.5 Mbps	0.5 Mbps	120 (60%)
B	5 Mbps	1 Mbps	40 (20%)
C	50 Mbps	10 Mbps	40 (20%)
Average	12.5 Mbps	2.5 Mbps	

TABLE I
DISTRIBUTION OF HETEROGENEOUS ASYMMETRIC ACCESS LINKS INTO 3 DIFFERENT BANDWIDTH GROUPS

In the first version [1], all peers keep 4 partners, one partner corresponds to one data stream, and the authors claim that beyond 4 partners there is no improvement in streaming quality. In the second version [2], all peers keep 8 sub-streams, one sub-stream corresponds to one data stream, and their simulation results suggest that more than 8 sub-streams is superfluous. mTreebone [5] and AnySee [3] also have a similar mechanism to CoolStreaming, and keep a static and homogeneous number of partners. In PRIME [4], the degree of a node is determined by bandwidth-degree condition to let a node of more access link capacity have more degree and to maximize the bandwidth utilization. However, the degree is static, and PRIME requires a priori knowledge of the access link bandwidth. PPLive, the most popular mesh-based P2P video streaming application in China, leaves it up to the system administrator to use his or her experience in setting the number of data streams, statically [7]. Lobb et.al. [8] propose an adaptive overlay algorithm to dynamically build and maintain the overlay mesh topology. They address same problem of this paper, however, their approach and solution are different from ours. Moreover, they have not investigated reasons why different number of data streams for each peer lead to different performance.

Due to issues of NAT or Firewall, most commercial mesh-based P2P video streaming systems rely on TCP-friendly transport protocols (e.g., TCP-Friendly Rate Control (TFRC) [9]), which does not provide congestion control for the aggregate connections incident at a given peer. Transport layer protocols for supporting aggregate congestion control have been studied recently [10], [11]. For the trade-off between the content bottleneck and the bandwidth bottleneck, the application should be able to control the number of connections. However, these protocols do not offer a mechanism for the application to dynamically increase or decrease the number of connections. Our *Loss Rate Window* is an application layer mechanism which dynamically changes the number of data streams according to the aggregate TCP loss rate.

To the best of our knowledge, our work is the first to analyze the impact of the number of data streams on the performance of a mesh-based P2P video streaming system.

III. EVALUATING THE IMPACT OF NUMBER OF DATA STREAMS

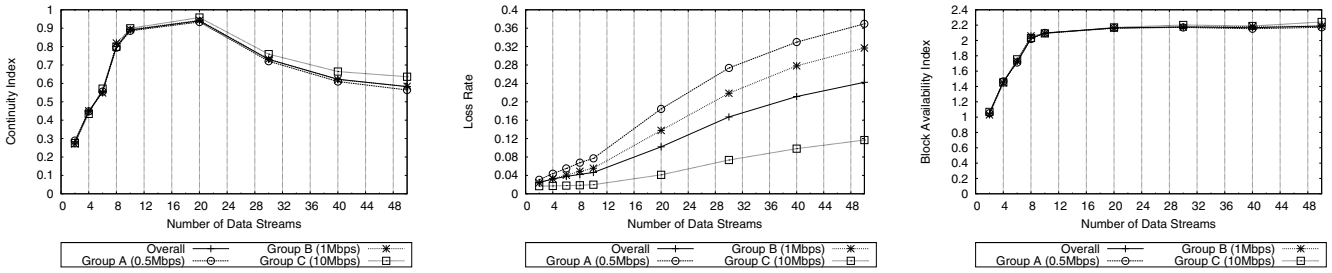
In this section, we analyze how the number of data streams at a peer affects the streaming quality in the mesh-based P2P video streaming system. We use CoolStreaming [1], [2] as an example state-of-the-art mesh-based P2P video streaming system to perform this study. Since video streaming is a bandwidth intensive application, we focus on two system parameters, which are access link bandwidth and video bitrate. We use a simulation methodology for the analysis to enable us to experiment with different bandwidth settings. However, we use a packet-level simulator, OMNeT++ and INET framework [12], to accurately capture the dynamics in a realistic network layer.

A. Simulation Configuration

In our simulation, the physical topology is generated by BRITE [13] using the following configuration parameters: 5 ISPs with 8 routers per ISP in top-down mode and 5 hosts are attached to each router, therefore the total number of hosts in the network is 200, and all of the hosts are used for experiments. The bandwidth between routers is uniformly distributed between 4Gbps and 10Gbps. The choice of parameters is consistent with realistic network configuration, wherein the core routers have more bandwidth than the edges [14].

We measure the performance over a large parameter space:
(i) *Homogeneous symmetric* in which uplink and downlink bandwidth of a peer are the same, and all peers have equivalent bandwidth setting.
(ii) *Homogeneous Asymmetric* in which uplink bandwidth is less than downlink bandwidth, and all peers have equivalent bandwidth setting.
(iii) *Heterogeneous Asymmetric* in which uplink bandwidth is less than downlink bandwidth, and each peer has different bandwidth setting.

We use 400Kbps bitrate in the homogeneous case. We use 200Kbps, 400Kbps, and 800Kbps bitrates in the heterogeneous



(a) Continuity Index as perceived by a node as the receiver

(b) Loss Rate as perceived by a node as the sender. Smaller uplink bandwidth group suffers higher loss rate than larger uplink bandwidth group.

(c) Block Availability Index as perceived by a node as the receiver. It is saturated after 10 data streams due to high packet loss rate after 10 data streams.

Fig. 2. Performance for a 200 node system with Heterogeneous Asymmetric access links for 400Kbps video bitrate

case to see how bitrate affects the performance of the system. We use TCP Reno [15] as a transport layer protocol, which is a widely deployed version of TCP in many operating systems. Peers buffer up 30 seconds of video data, and each run of the simulation is 10 minutes. Peers join the system according to Poisson arrival with an average inter-arrival time of 150 milliseconds. The size of a block is one second of video data, which translates to 50KBytes with 400Kbps video bitrate. A data stream is bi-directional, hence, is used for inbound and outbound data transmission.

We use following metrics to evaluate the CoolStreaming performance.

(i) *Continuity index*: The ratio of the number of blocks that arrive before the playback deadline to the total number of blocks, which is the main metric for the streaming quality. For instance, if a node A receives 90 blocks before the deadline, and misses or receives 10 blocks after the deadline out of a total 100 blocks during its playback session, the continuity index of A is 0.9. CoolStreaming [1], [2] defined and used this metric for the streaming quality in their paper.

(ii) *Loss rate*: Aggregate packet loss rate for the set of TCP flows at a peer. We customize TCP Reno implementation of INET framework, and count the number of 3 duplications and timeouts over the total number of packets sent. Since a higher packet loss rate implies a higher congestion level on a wired network, and the congestion of a node can reflect the bandwidth bottleneck of the node, this metric is used for the indication of the bandwidth bottleneck.

(iii) *Block availability index*: For a given node, this metric quantifies the number of potential blocks that can be requested from its partners. For example, let's suppose that a node A has blocks 1 and 2, and does not have blocks 3,4,5 at some instant. If some of its partners have blocks 3 and 5, but none of the partners have block 4 in their buffer, the block availability of node A at this moment is 2. We define this metric for use with the content bottleneck.

B. Homogeneous Symmetric/Asymmetric

We measure the continuity index varying the number of data streams (2, 4, 6, 8, 10, 20, 30, 40, and 50) in 7 different bandwidth settings (0.5, 1, 2, 4, 10, 50, and 100Mbps). In the asymmetric case, the bandwidth parameter specifies the uplink, and the downlink bandwidth is 5 times the uplink bandwidth. The video bitrate is 400Kbps.

Figure 1(a), 1(b) shows that 4 to 8 data streams achieve the maximum continuity index for all bandwidth settings in the homogeneous case. This result is consistent with the claim of CoolStreaming that 4 to 8 is the right value for the number of data streams with homogeneous bandwidth setting.

C. Heterogeneous Asymmetric

Homogeneous bandwidth distribution is not realistic because the bandwidth of edge network is highly heterogeneous. To be more realistic, we use a bandwidth configuration as shown in Table I, which is a simplified setting based on real-life measurements of uplink capacity of peers in the BitTorrent network [16].

We measure the continuity index for this heterogeneous bandwidth setting with three different video bitrates; 200Kbps, 400Kbps, and 800Kbps. Figure 1(c) shows surprisingly different results from the homogeneous case. For 400Kbps bitrate, the continuity index of 4 data streams is around 0.49 and that of 8 data streams is 0.78, however the maximum continuity index of 0.94 is achieved with 20 data streams. The performance of 20 data streams is twice that of 4 data streams. For 200Kbps and 800Kbps bitrates, the maximum continuity index is achieved with 10 data streams. This result shows that arbitrarily choosing the number of data streams can lead to very poor performance depending on the available network bandwidth and/or video bitrates.

D. Analysis

This raises the question why the different number of data streams results in different streaming quality, and how we should control the number of data streams at each peer. To answer the question, we focus on the bandwidth bottleneck and the content bottleneck, which we alluded to earlier [4], [6]. Bandwidth bottleneck happens when a peer sends or receives data much larger than the available bandwidth between two peers leading to network congestion. Content bottleneck happens when a node cannot get blocks from its partners in a timely manner even though there is sufficient bandwidth between them. Inefficient overlay topology and/or poor scheduling algorithm could be reasons for the content bottleneck.

Since loss rate is a reasonable criterion for the congestion on a wired network, and congestion can reflect the bandwidth bottleneck of a peer, we measure loss rate of each peer as a metric for the bandwidth bottleneck. We have modified the TCP Reno implementation of INET framework, and let the TCP module notify packet loss and packet send events to the application layer. We use block availability index as a metric for the content bottleneck. The smaller the block availability index the lesser the number of blocks a node can get from its

partners. Ultimately, a small block availability index signals that the blocks could not be delivered throughout the overlay network at the right time.

We use the heterogeneous asymmetric bandwidth configuration of Table I with 400Kbps video bitrate, and measure the average loss rate and the average block availability index for each bandwidth group and for the overall system. The loss rate is measured for the data streams that are outbound from a node (i.e., when a node is acting as a sender), and the continuity index and the block availability index are measured when a node is acting as a receiver. From Figure 2(b), we can see the linear relationship of the loss rate and the number of data streams. The system incurs higher loss rate as peers in the system establish more data streams. A smaller uplink bandwidth group suffers higher loss rate than a larger uplink bandwidth group for a given number of data streams, and the smaller group has a steeper slope than the larger group. Figure 2(c) shows that the block availability index increases linearly with the number of data streams at first, but gets saturated after some point. The block availability should increase linearly with the number of data streams if there was no bandwidth bottleneck since more partners could build a denser mesh P2P network. By putting all the results from Figure 2 together, we can make the following observations: When a sending node is peered with a small number (less than 8) of receiving nodes (Figure 2(b)), it incurs a low loss rate; however, its available uplink bandwidth is not fully utilized. This leads to low average block availability index and low average continuity index in the system as a whole. On the other hand, when a node is peered with a large number (more than 20) of receivers, it suffers a high loss rate and the ensuing congestion prevents the data blocks from reaching their destination in a timely manner leading to poor block availability and low continuity index.

This raises the question as to why congestion happens in the network given that all the peers use TCP connections, and TCP has a congestion control mechanism. Unfortunately, TCP or TCP-friendly transport protocols, while providing congestion control for individual connections, do not provide congestion control for *aggregate connections* associated with a given process. Thus when a node is serving too many peers it can lead to network congestion. On the other hand, too few connections lead to content bottleneck since blocks may not be received at a node in a timely manner despite the availability of network bandwidth between nodes. Note that even if the transport protocol is enhanced to deal with congestion control for aggregate connections, it does not have the knowledge at the transport level to decide on the number of data streams needed to deal with content bottleneck since that is an application level metric. Thus it is best to make this decision in the application layer.

From these simulation results, we get an invaluable insight. *A P2P video streaming application should have dynamic data stream control mechanism to make as many streams as possible to build a denser mesh network and to avoid the content bottleneck, but yet it should avoid the network congestion that could lead to the high packet loss rate.*

IV. DynaStream Protocol

From the analysis in Section III-D, we can deduce that a node should have as many data streams as possible so as not to incur the content bottleneck, but the number of streams should be limited so as not to incur the bandwidth bottleneck.

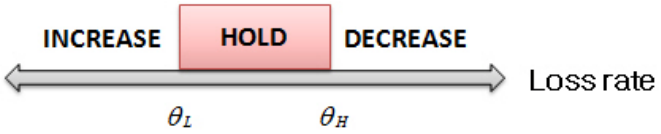


Fig. 3. Loss Rate Window

Bandwidth bottleneck occurs when the aggregate data streams in and out of a node are not adjusted dynamically respecting the state of congestion in the network.

Based on the analysis and observations from the previous section, we propose a new mesh-based P2P video streaming protocol called *DynaStream*. At the heart of this protocol is a novel mechanism, namely, *Loss Rate Window* (LRW).

A. Loss Rate Window

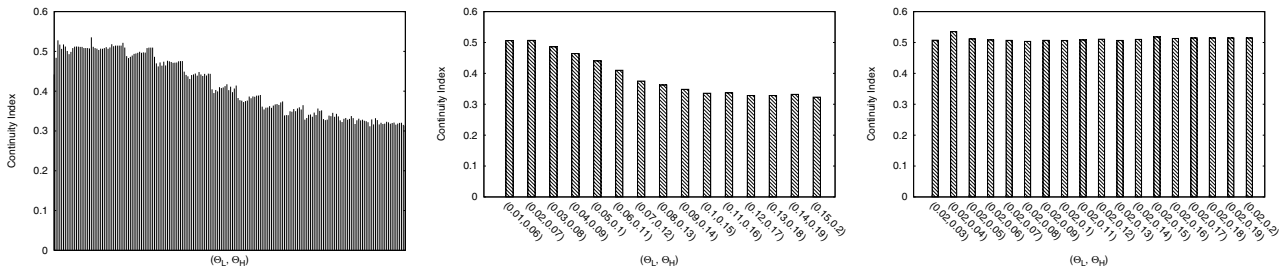
The goal of LRW is to bound the aggregate loss rate to a certain range by adjusting the number of data streams (Figure 3). An assumption here is that the scheduling algorithm does not control data flow rate. The scheduling algorithm simply sends a request for a block to a node that has it and fetches the block if available. By periodically altering the number of data streams, LRW indirectly controls the aggregate data flow rate and the aggregate loss rate.

The loss rate between two end hosts not only captures the congestion in the core network but also the congestion in the access link of the hosts. Therefore, controlling the number of data streams based on the loss rate observed at a peer simultaneously achieves the adaptivity of the protocol to both the traffic fluctuation in the network and the access link capacity of the peer.

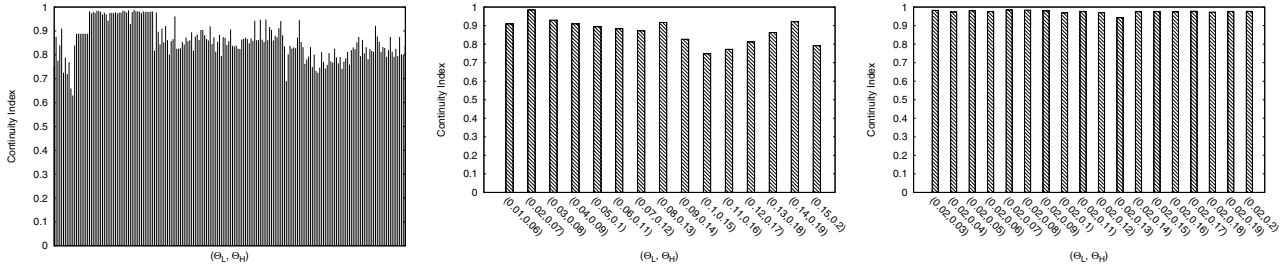
Section III-B conclusively shows that the uplink bandwidth, which is usually much smaller than the downlink bandwidth, is the source of the loss of quality in a P2P video streaming system. Therefore, LRW runs at the sender side and uses the aggregate loss rate observed on its uplink as an inference of network congestion.

LRW works as follows. The minimum number of data streams for a node is 4. If the total number of nodes, N, in the P2P network is more than 4, each node has N-1 data streams. Each node periodically measures the aggregate loss rate while they are transmitting packets, and adds or removes data streams according to the measured loss rate. If the loss rate is within a window, the node speculates that the current network state is stable, and does not change the number of data streams. If the loss rate gets larger than a threshold θ_H, the node guesses that the current network state is congested and removes δ data streams. On the other hand, if the loss rate is smaller than a threshold θ_L, the node speculates that the current network has enough room to accommodate more data streams and adds δ data streams. When removing data streams, we sort the data streams in the ascending order of observed throughput and get rid of data streams which have less throughput first. We choose this AIAD (Additive Increase Additive Decrease) strategy rather than AIMD (Additive Increase Multiplicative Decrease) [17] because simultaneous connection drops can lead to an effect similar to node churn, which is detrimental to the streaming quality. We use a default value of 1 for δ.

The virtue of LRW is that there are no probing packets, therefore, it does not consume any network bandwidth for its



(a) Homogeneous Asymmetric Uplink 1Mbps Bitrate 400Kbps



(b) Heterogeneous Asymmetric Uplink of Table I Bitrate 400Kbps

Fig. 4. Continuity Index against different values of θ_L and θ_H for various bandwidth settings with 200 nodes. Left graph varies θ_L from 0.01 to 0.19 and θ_H from θ_L+0.01 to 0.2 with a granularity of 0.01. Middle graph varies θ_L from 0.01 to 0.15 with fixed θ_H=θ_L+0.05. Right graph fixes θ_L to 0.02 and varies θ_H from 0.03 to 0.2.

adaptive behavior. It is a completely distributed mechanism, and thus is scalable. Moreover, LRW does not need any input for the uplink bandwidth or the downlink bandwidth from the application to determine the number of data streams.

B. Experiments to Empirically Determine Thresholds

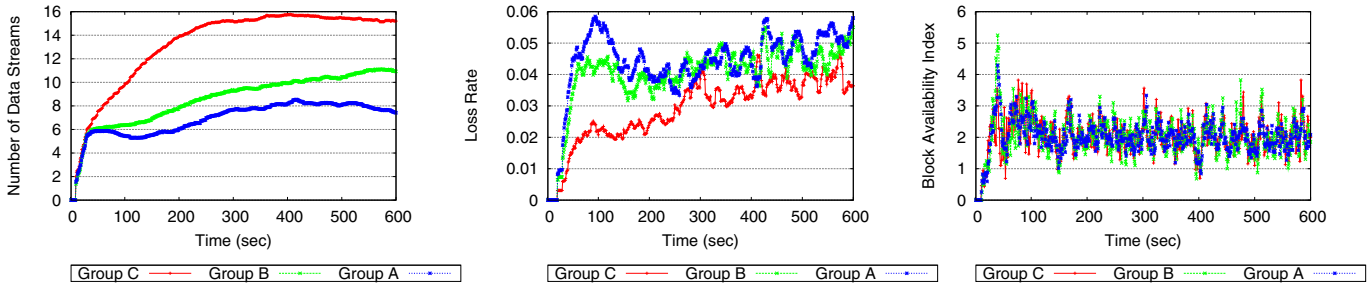
The two key parameters of LRW are θ_L and θ_H. To see how values of θ_L and θ_H affect the streaming quality, we measure the continuity index with a large number of combinations of θ_L and θ_H with all the different access link bandwidth settings and the different video bitrates that are used in Section III. However, in this paper due to space limitation we present results for only two representative configurations. We vary θ_L from 0.01 to 0.19 and θ_H from θ_L+0.01 to 0.2 with a granularity of 0.01(1% loss rate) giving rise to 190 combinations. From the large amount of simulation experiments (summarized in the graphs shown in Figure 4), we observe that there is a sweet-spot for the window independent of system parameters (namely access link bandwidth distributions or video bitrates). Figure 4 shows the sensitivity of the continuity index to values of θ_L and θ_H. In each horizontal row of the figure, the left-most graph shows the continuity index against 190 combinations of θ_L and θ_H; the middle graph shows the continuity index against different θ_L with fixed window length 0.05; and the right-most graph shows the continuity index against varying θ_H with θ_L fixed at 0.02. Regardless of the bandwidth distribution and video bitrates, we see couple of similarities. First, patterns of graphs on the left and the middle show that the continuity index is more sensitive to the value of θ_L, which is a criterion for adding data streams than θ_H, which is a criterion for removing data streams. Second, in all settings, maximum continuity index occurs when θ_L is 0.02. From these empirical results, we conclude that θ_L should be 0.02, and θ_H can be any value from 0.03 to 0.2. Based on these empirical results, we choose θ_L to be 0.02 and θ_H to be 0.07.

By virtue of the design of LRW, each node self-stabilizes to the right number of data streams it can support on its uplink given the system setting for the parameters θ_L and θ_H. This is illustrated in Figure 5 for three different uplink bandwidth groups (drawn from Table I). The figure shows how the system stabilizes to choosing the right number of data streams commensurate with the uplink bandwidth from system start-up in a fairly short amount of time. Group C (large uplink capacity) stabilizes at 15 data streams; group B stabilizes around 10 streams; and group A with the smallest uplink capacity stabilizes at 8 data streams. More importantly, a similar adjustment will happen automatically at each node whenever a node observes a change either due to the application dynamics (i.e., change in the video bitrate being served), or network dynamics (i.e., observed network congestion).

C. Comparison of DynaStream to Static Peer Group Setting

Figure 6 compares DynaStream against static and homogeneous setting of the peer group. We measure the continuity index of the heterogeneous asymmetric bandwidth setting of Table I with three different video bitrates. The continuity index is averaged over all 200 nodes. As can be in the figure, DynaStream outperforms static settings of 4 and 8 streams (suggested by CoolStreaming) for all three video bitrates. With 400Kbps bitrate, the continuity index of DynaStream is 0.998, that of static 4 is 0.491, and that of static 8 is 0.780. DynaStream improves the performance by a factor of two compared to static 4. With 800Kbps bitrate, the continuity index of DynaStream is 0.765, that of static 4 is 0.275, and that of static 8 is 0.408. DynaStream enhances the performance by a factor of three compared to static 4 and by a factor of two compared to static 8.

Note that the number of data streams is not a system parameter anymore in DynaStream. In other mesh-based P2P



(a) Average Number of Data Streams **(b)** Average Loss Rate **(c)** Average Block Availability Index

Fig. 5. Convergence of DynaStream with 200 nodes, 400Kbps bitrate, and Heterogeneous Asymmetric access links of Table I. DynaStream automatically and quickly converges to the optimal number of data streams for each node that maximizes video streaming quality. Average loss rate is bounded within thresholds (0.02, 0.07) after convergence. Block availability index is 2 on average after convergence, which means there are 2 blocks that can be requested from other peers in every second.

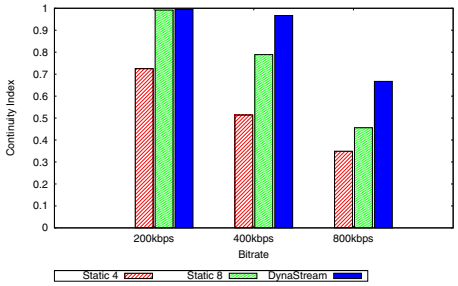


Fig. 6. Performance comparison of DynaStream to Static and Homogeneous Assignment of Peer-set

video streaming systems, it is a parameter which is hard to decide and is affected by other system configurations such as video bitrates or heterogeneous access link bandwidth of peers. DynaStream automatically and dynamically changes the number of data streams adapting to the available network resources.

V. Conclusions

Previous mesh-based systems use static or homogeneous number of data streams. First, we carry out a systematic study to validate our conjecture that static and homogeneous number of data streams leads to the poor quality as measured by continuity index, block availability index, and packet loss rate. Second, we establish through the simulation framework that the reasons for the loss of quality is either due to sub-optimal choice of number of peers (i.e., under commitment of available network resources) or over-commitment leading to network congestion. Based on these extensive measurements and analysis, we propose a new mesh-based P2P video streaming protocol called *DynaStream*. At the heart of the protocol is a simple and completely distributed mechanism, namely, *Loss Rate Window* that allows each node to locally adjust the number of data streams it is willing to support on its uplink to avoid the pitfalls of under- or over-commitment of network resources. We show that DynaStream consistently outperforms static and homogeneous assignment of data streams. The virtue of DynaStream is that each node makes an entirely local decision to determine the right number of data streams based on the aggregate packet loss rate. By exploiting the loss rate, DynaStream can maintain the overlay topology to be adaptive to the heterogeneous peers' uplink bandwidth and to the variations of the network state without explicitly knowing or measuring peers' uplink bandwidth. Our future work includes real implementation and evaluation on a large-scale testbed such as the PlanetLab.

References

[1] X. Zhang, J. Liu, B. Li, and T. S. P. Yum, "DONet/Coolstreaming: A data-driven overlay network for live media streaming," in *Proceedings of IEEE INFOCOM*, Miami, FL, USA, March 2005.

[2] B. Li, Y. Qu, Y. Keung, S. Xie, C. Lin, J. Liu, and X. Zhang, "Inside the New Coolstreaming: Principles, measurements and performance implications," in *Proceedings of IEEE INFOCOM*, 2008.

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

[4] N. Magharei and R. Rejaie, "Prime: Peer-to-peer receiver-driven mesh-based streaming," in *Proceedings of IEEE INFOCOM*, Anchorage, Alaska, USA, May 2007.

[5] F. Wang, Y. Xiong, and J. Liu, "mTreebone: A hybrid tree/mesh overlay for application-layer live video multicast," in *Proceedings of IEEE ICDCS*, 2007.

[6] N. Magharei and R. Rejaie, "Understanding mesh-based peer-to-peer streaming," in *Proceedings of NOSSDAV*, Newport, Rhode Island, USA, 2006.

[7] Y. Huang, T. Fu, D. Chiu, J. Lui, and C. Huang, "Challenges, design and analysis of a large-scale p2p-vod system," in *Proceedings of ACM SIGCOMM*, Seattle, WA, USA, 2008.

[8] R. J. Lobb, A. P. C. da Silva, E. Leonardi, M. Mellia, and M. Meo, "Adaptive overlay topology for mesh-based p2p-tv systems," in *Proceedings of NOSSDAV*, Williamsburg, Virginia, USA, 2009.

[9] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *Proceedings of ACM SIGCOMM*, Stockholm, Sweden, 2000.

[10] L. Ma and W. T. Ooi, "Congestion control in distributed media streaming," in *Proceedings of IEEE INFOCOM*, Anchorage, Alaska, USA, May 2007.

[11] D. E. Ott, T. Sparks, and K. Mayer-Patel, "Aggregate congestion control for distributed multimedia applications," in *Proceedings of IEEE INFOCOM*, Hong Kong, China, March 2004.

[12] "OMNeT++," http://www.omnetpp.org/.

[13] "BRITE," http://www.cs.bu.edu/brite/.

[14] A. Akella, S. Seshan, and A. Shaikh, "An empirical evaluation of wide-area internet bottlenecks," in *Proceedings of IMC*, 2003.

[15] M. Allman, V. Paxson, and W. Stevens, "Tcp congestion control," RFC 2581, April 1999, http://www.ietf.org/rfc/rfc2581.txt.

[16] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "Do incentives build robustness in bittorrent?" in *Proceedings of NSDI*, Cambridge, MA, USA, 2007.

[17] D. Chiu and R. Jain, "Analysis of the increase and decrease algorithm for congestion avoidance in computer networks," *Computer Networks and ISDN Systems*, vol. 17, 1989.