Supporting Heterogeneity and Congestion Control in Peer-to-Peer Multicast Streaming*

Venkata N. Padmanabhan   Helen J. Wang   Philip A. Chou
Microsoft Research

Abstract— **We consider the problem of supporting bandwidth heterogeneity and congestion control in the context of P2P multicast streaming. We identify several challenges peculiar to the P2P setting including robustness concerns arising from peer unreliability and the ambiguity of packet loss as an indicator of congestion. We propose a hybrid parent- and child-driven bandwidth adaptation protocol that is designed in conjunction with a framework for robustness and that exploits application-level knowledge.**

I. Introduction

There has been a growing interest in peer-to-peer, or end host-based, multicast for streaming because of its advantages of being self-scaling, low cost (compared to infrastructure-based approaches), and easy to deploy (compared to IP multicast) (e.g., [4][7][13]). However, a key challenge in P2P multicast is robustness. Unlike routers in IP multicast or dedicated servers in an infrastructure-based content distribution network such as Akamai, peers or end hosts are inherently unreliable due to crashes, disconnections, or shifts in user focus (e.g., a user may hop between streaming sessions or launch other bandwidth-hungry applications). A natural way to achieve robustness is through redundancy, both in network paths and in data. Our work on CoopNet [13][12] has shown that resilient peer-to-peer streaming can be achieved by carefully constructing multiple diverse distribution trees spanning the interested peers, efficiently introducing redundancy in data using multiple description coding (MDC) [8], and striping the descriptions (i.e., substreams) across the diverse set of trees. A key property of MDC, which distinguishes it from traditional layering, is that with *any* subset of the descriptions, a receiver can reconstruct the stream with quality commensurate with the number of descriptions received.

A second key challenge in peer-to-peer multicast (as well as other forms of multicast) is accommodating bandwidth heterogeneity. Heterogeneity in bandwidth can be both static (e.g., due to differences in link speed) and dynamic (e.g., due to congestion). *It is desirable that the framework for congestion control and heterogeneity management build on top of and take advantage of the robustness scheme outlined above.* This then is the focus of the present paper.

A popular approach to supporting bandwidth heterogeneity in the context of unicast as well as multicast streaming is to offer multiple streams, each optimized for a specific bandwidth level. Clients tune in to the stream that best matches their bandwidth. While this approach has the advantage of being simple, it suffers from a number of drawbacks. It is wasteful of bandwidth on links shared by streams of different rates, it typically can only accommodate coarse-grained adaptation, and having clients switch between streams of different bandwidth in response to congestion may be quite disruptive.

An alternative and more elegant approach is the one advocated in the seminal work on Receiver-driven Layered Multicast (RLM) [10]. RLM tackles the heterogeneity and congestion control problems by combining a layered source coding algorithm with a layered transmission system that uses a separate IP multicast group for transmitting each layer of the stream. Receivers specify their level of subscription by joining a subset of the groups; at any point, a receiver's subscription must be a contiguous subset that includes the base layer group. By having receivers drop layers upon congestion and add layers to probe for additional bandwidth, RLM enables scalable and adaptive congestion control in an IP multicast setting.

A significant drawback of RLM, however, is that there is a fundamental mismatch between the ordering of layers based on importance and the lack of widespread support for differentiated treatment of packets in the Internet. In the face of congestion there is no mechanism to ensure that the network preferentially drops enhancement layer packets over the base layer ones.[1] Thus the support for heterogeneity and congestion control has to be coupled with mechanisms to ensure robustness to packet loss.

Furthermore, RLM cannot be readily applied to peer-to-peer multicast because of several differences compared to IP multicast. First, in P2P multicast, the interior

*Please visit the CoopNet project page at http://www.research.microsoft.com/projects/CoopNet/ for more information

[1]The RLM paper [10] actually argues that the lack of support for preferential dropping is an advantage since it discourages greedy behavior. However, we believe that this point is moot in the context of peer-to-peer multicast since there are more direct opportunities to cheat, for example, by failing to forward packets. Cooperative behavior is inherently assumed in such settings.

nodes as well as the leaves of the multicast tree are receivers. When an interior receiver adapts the bandwidth usage on its incoming link, the effect on its downstream receivers must be taken into account. Second, in P2P multicast, receivers that are interior nodes may also need to adapt bandwidth usage on their outgoing links, as these may be bottlenecks. And finally, in P2P multicast, the interior nodes of the multicast tree are a dynamic set of end hosts rather than a set of dedicated routers. Hence packet loss seen by a receiver arises not only from congestion in the network at large, but also from the unreliability of its peers, tree dynamics, and local congestion. Hence, it is not always possible to reduce packet loss by shedding traffic. In some cases the more appropriate response is to switch from the current parent to a better one.

Motivated by the above considerations, we approach the problem as follows.

1) **Joint design of support for heterogeneity and robustness:** We design our adaptation scheme for heterogeneity in the context of a framework for robustness [13][12] that incorporates redundancy in network paths and in data. We use a *layered MDC* codec [6], which combines the robustness of MDC with the adaptability of layering.
2) **Hybrid parent- and child-driven adaptation:** Parents and children cooperatively determine the appropriate response to packet loss by exploiting path diversity to localize the cause of packet loss.
3) **Exploiting application-level knowledge for adaptation:** Both parents and children exploit their knowledge of the relative importance of the layered MDC substreams and the structure of the distribution trees to adapt to changing bandwidth in a way that minimizes the impact on the descendant nodes.

The specific novel contributions of this paper are the adaptation protocol and the application of layered MDC in this context. However, we believe that *our paper also makes a more general contribution by drawing attention to two observations*. First, it may often be advantageous to have the "routers" (i.e., the peers) in a P2P system use application-level knowledge to optimize performance, for example, by shedding less important data when there is congestion. There is no reason for the P2P nodes to simply mimic the "dumb" forwarding that IP routers do. Second, while network path diversity has been used in P2P systems for resilience [12] and for bandwidth management [4], it can also be exploited to give the peers greater visibility into the network using techniques such as tomography [3].

Before getting into layered MDC and our adaptation scheme, we briefly review our previous work on CoopNet, which provides the framework we build on.

II. CoopNet Background

As mentioned in Section I, CoopNet [13][12] employs redundancy in both network paths and in data to make P2P streaming robust to peer unreliability and failures. Rather than use a single distribution tree as in traditional multicast, CoopNet constructs multiple, diverse distribution trees, each spanning the set of interested peers. The trees are diverse in their structures; for instance node A could be the parent of B in one tree but be its child in another. Our experiments have suggested that having 8 trees works well. To ensure diverse and bushy (i.e., high fanout) trees, each peer is typically made an interior node (i.e., a "fertile" node) in a few trees and a leaf node (i.e., a "sterile" node) in the remaining trees. In the extreme case, a peer may be fertile (and have several children) in just one tree and be sterile in the remaining trees.

Tree management in CoopNet is done by a centralized tree manager. Our discussion here is agnostic of how exactly tree management is done, and we do not discuss the specifics of how nodes join and leave trees, and find themselves new parents.

The stream is encoded using multiple description coding (MDC). The MDC substreams, or descriptions, are all of equal importance and have the property that *any* subset of them can be used to reconstruct a stream of quality commensurate with the size of the subset. This is in contrast to layered coding, which imposes a strict ordering on the layers; for instance, an enhancement layer is useless in the absence of the base layer or a previous enhancement layer. The flexibility of MDC comes at a modest price in terms of bandwidth (typically around 20%). It is important to note, however, that MDC is optimized for the expected packet loss distribution. If few or no losses are expected, then MDC would adapt by cutting down the amount of redundancy and the bandwidth overhead.

The descriptions generated by the MDC codec are striped across the diverse set of trees. This ensures that each peer receives the substreams over a diverse set of paths, which makes it quite likely that it will continue to receive the majority of the descriptions (and hence be able to decode a stream of reasonable quality) even as other peers experience failures.

In our earlier CoopNet work, we assumed that all peers received streams of the same bandwidth. Also, we did not consider how peers might respond to congestion. We turn to these issues next.

III. Key Questions

Our discussion of an adaptation framework for accommodating bandwidth heterogeneity and congestion control is centered around the following key questions:

1) How should the stream data be organized to enable peers to subscribe to just the portion that matches their current available bandwidth?

2) How should peers respond to packet loss?
3) How should RLM-style adding and dropping of layers be done so that the impact on the other peers is minimized?

We discuss these questions in the sections that follow.

IV. Layered MDC

A particularly efficient and practical MDC construction uses layered coding and Forward Error Correction (FEC) as building blocks. Layered coding is used to prioritize the data, while FEC, such as Reed-Solomon encoding, is then used to provide different levels of protection for the data units. Determining the protection level for the data units is an optimization procedure that is based on both the importance of the data units and the packet loss distribution across *all* clients.

When the clients' bandwidths are heterogeneous, either due to different link capacities or dynamic network conditions, MDC becomes less efficient. A naive way of supporting heterogeneity is to treat descriptions just as layers are treated in RLM: dropping descriptions upon congestion and adding descriptions to see if additional bandwidth is available. However, this approach is inefficient since the MDC construction is optimized for the entire ensemble of (heterogeneous) clients. For high-bandwidth clients, there is wasteful redundancy in the MDC, which unnecessarily degrades quality. For low-bandwidth clients, the redundancy would typically be insufficient to enable decoding the received stream.

In [6], we have developed a novel *layered MDC* scheme in which the descriptions are partitioned into layers. For our discussion here, we consider two layers, a base layer and an enhancement layer. The base layer descriptions are optimized for just the low-bandwidth clients. The enhancement layer descriptions are optimized for the high-bandwidth clients, while also providing additional protection for the less-protected data units in the base layer. This ensures that the more important data units have a higher probability of being successfully delivered over the best-effort Internet.

This layered MDC construction is clearly optimal for the low-bandwidth clients. Our experiments also indicate that high-bandwidth clients suffer a modest penalty of about 1.4 dB (in terms of distortion) compared to the case where all of the descriptions are optimized exclusively for the high-bandwidth clients. Thus layered MDC combines layering and robustness without sacrificing much in terms of efficiency.

In the context of P2P streaming with heterogeneous clients, the base layer alone is sent to the low-bandwidth clients while both the base and the enhancement layers are sent to the high-bandwidth clients. Clients adapt to dynamic fluctuations in bandwidth by adding/dropping descriptions in their current highest layer (or in the next layer above/below if the current layer is full/empty).

V. Inferring the location of congestion

When a node experiences packet loss in its incoming stream, the appropriate response depends on the reason for the packet loss. As discussed in Section VI, if there is congestion near the node's incoming link, then the node should shed some incoming traffic, while if there is congestion near its parent's outgoing link, then the parent should shed some outgoing traffic, possibly destined for another node. The interesting question then is how a node can determine where congestion is happening. In our framework (Section II), each node receives substreams from multiple parents. Thus each node is in a position to monitor the packet loss rate of the substream from each parent. *This loss rate information from a diverse set of network paths can be used to infer the likely location of network congestion using techniques akin to network tomography [3].* If a child node experiences significant packet loss in most or all trees, it can reasonably conclude that the congestion is occurring at or close to its incoming link. Likewise, if a parent node receives packet loss indications from most or all of its children, it can reasonably conclude that congestion is occurring at or near its outgoing link.

To evaluate the efficacy of this heuristic, we conducted a simple simulation experiment. We generated a 1000-node topology using the BRITE topology generator [11]. We used the preferential connectivity model provided by BRITE (based on the work of Barabasi et al. [2]), which helps capture the power-law distribution of node degrees observed in practice. The resulting topology had 662 leaves (i.e., nodes with degree 1), which we treat as candidates for peers (parents or children). In each run of the experiment, we pick a child node and 16 parent nodes (corresponding to 16 distribution trees) at random. We compute the shortest path routes from the parent nodes to the child node. Each link contained in one of these shortest paths is independently marked as "congested" with a probability of 10%. The substream from a parent node to the child is assumed to suffer congestion if one or more links along the path is marked as congested; we simply term the parent as "congested" in such a case.

Likewise, the path from a potential parent (i.e., a leaf node in the topology other than the chosen parents and the child) to the child is considered to be congested (and hence prone to packet loss) if it includes a congested link; again, for ease of exposition, we term the potential parent as being "congested" in such a case.

Figure 1 shows the fraction of congested potential parents versus the fraction of congested current parents over 1000 runs of the experiment. The scatter plot shows the results of the individual runs while the solid line shows the mean. It is clear that when a large fraction (say over 75%) of a node's current parents are congested, it is also likely that a large fraction of its potential parents would be congested. This is so because congestion is

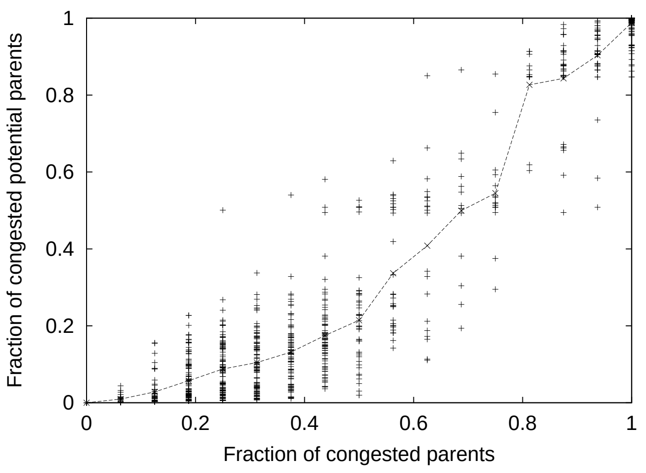

Fig. 1. Fraction of congested parents versus fraction of congested potential parents.

likely near the node's incoming link. In such a case, the node should shed incoming traffic to alleviate the congestion.

Conversely, the experiment also shows that if we swap the roles of parents and children, then when a large fraction of a node's children are congested, then there is the likelihood of congestion near the node's outgoing link. Thus, when a node receives packet loss indications from most or all of its children, it should shed outgoing traffic to alleviate the congestion.

Furthermore, *sharing of information between parents and children can lead to a more robust inference of the location of congestion*. For instance, consider a child that is unlucky to have several parents with congested uplinks.[2] The child can deduce that this is the case (and search for new parents) based on the knowledge of complaints that its parents are receiving from their other children. In the absence of complaint information from its parents, the child might have incorrectly concluded that the congestion is at its incoming link and hence proceeded to shed incoming traffic.

VI. Adaptation Protocol

We now discuss our adaptation protocol. As outlined in Section IV, we assume that the stream has been coded into sets of descriptions corresponding to each of the base layer and one or more enhancement layers; each description is termed a "substream" here.

As in RLM, there are two aspects to the adaptation protocol: shedding traffic when there is congestion, and adding traffic when there isn't congestion. We discuss each of these in turn. Our emphasis is on pointing out the unique opportunities for optimization in a P2P setting.

[2]This may happen in practice because congested links may be concentrated in the "last-mile" to/from peers rather than be spread uniformly throughout the network as in our simple experiment described above.

A. Shedding traffic

When congestion is encountered, the appropriate reaction depends on the location of the congested link(s). We consider three cases:

1) If congestion is at or near the outgoing link of a node, the node sheds outgoing traffic to alleviate the congestion. It does this by shedding children, who then have to look for new parents.
2) If congestion is at or near the incoming link of the node, the node sheds incoming traffic to alleviate the congestion. It does this by shedding parents. This entails also shedding any children that may have been receiving the now-discontinued substream(s).
3) If congestion is in the "middle", then the child node looks for new parents with a view to routing around or avoiding the point(s) of congestion.

We now turn to the interesting questions of how a congested parent picks children to shed and how a congested child picks parents to shed.

A congested parent preferentially sheds children that are receiving descriptions from the highest enhancement layer. Of such children, it preferentially sheds those that have no children or have few descendants in the tree of interest. (Recall from Section II that each peer is a leaf node in most of the trees.) The objective is to pick children that will be least affected by being orphaned because they are receiving substreams of the least importance from the congested parent and have few or no descendants dependent on them. Such *parent-driven* selective dropping results in better quality than a policy of randomly dropping packets across all children.

Likewise, a congested child preferentially sheds parents from whom it is receiving descriptions belonging to the highest enhancement layer. Of such parents, it preferentially sheds those that are sending it substreams for which it has no children or has few descendants. Such *child-driven* selective dropping likewise results in better quality than randomly dropping incoming streams.

This hybrid parent- and child-driven congestion control scheme elegantly addresses a key difficulty in using layered coding in today's Internet, viz., the mismatch between the prioritization of the layers and the lack of widespread support for service differentiation in the Internet.

B. Adding Traffic

Receivers not only need to adapt to worsening network conditions but also need to probe for newly available bandwidth, if any. When a receiver has not experienced any loss for a threshold period of time, it carries out a *join experiment*, as in RLM, by subscribing to an additional description in the current highest layer or one in the next higher layer if all of the descriptions in the current highest layer are already being received.

Subscribing to the new description involves joining the corresponding tree.

There is always the danger that a join experiment "fails" because the additional traffic congests a link that was operating almost at capacity. Such an unsuccessful join experiment could lead to packet loss and quality degradation at the receiver as well as other nodes (in particular, its descendants). A key advantage of using layered MDC over plain layered coding is that its inherent redundancy limits the damaged caused by an unsuccessful join experiments. For our discussion here, we assume that subscribing to an additional description causes the loss of at most one description's worth of data (basically, the additional data can at worst displace an equal amount of data previously subscribed to). If the losses are confined to the same layer as the new description, then we are no worse off than before the join experiment because all descriptions in a layer are equally valuable. Even if losses are suffered in a lower and hence more important layer, the redundancy in layered MDC can typically help recover the affected layer.

In contrast, RLM with plain layered coding is far more susceptible to the deleterious effects of failed join experiments, in addition to the deleterious effects of random packet loss. This is because there is nothing to mask packet loss suffered by a lower and hence more important layer, which can then render all subsequent layers of received data useless and degrade quality significantly.

To evaluate this, we compared the impact of join experiments with plain layered coding (RLM) to those with plain multiple description coding (MDC). We set the total number of substreams to be the same in both cases: 32 (thin) layers with RLM, and 32 descriptions with MDC. We assumed that layers (respectively, descriptions) are independently lost with probability 10%, and that the MDC system is optimized for this loss probability. Figure 2 shows for the RLM system (with circles) and the MDC system (without circles) typical video quality (measured as PSNR in dB[3]) as a function of the number of substreams to which a receiver is subscribed after performing the join in a join experiment. For the RLM system, even if the join succeeds (dotted line with circles), quality does not improve significantly since it is saturated at a low level due to the random packet loss frequently disrupting the more important layers; moreover, if the join fails (solid line with circles), quality falls even further. In contrast, for the MDC system, if the join succeeds (dotted line), quality is good for any number of substreams above 25, while if the join fails (solid line), quality remains the same as before the join. *Thus the loss resilience provided by MDC also enhances the robustness of congestion control.*

[3]Peak Signal-to-Noise Ratio in decibels is given by $10\log_{10}(255^2/D)$, where D is the mean squared error between the original and reconstructed luminance video pixels.

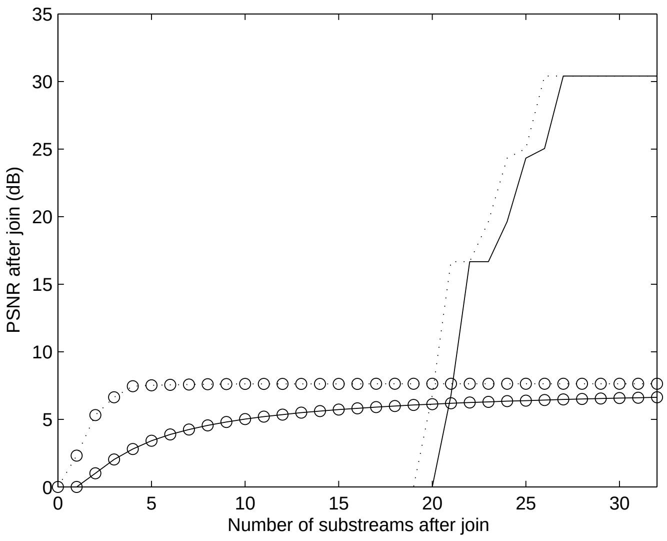

Fig. 2. Join Experiment for RLM vs. MDC: no collateral damage in MDC

VII. Related Work

There has been much work following up on and improving on RLM [10]. There has been work on adjusting the rate of each layer dynamically and adding/dropping layers in a manner that is TCP-friendly [9]. In our work, we do not advocate a specific policy in this regard and could leverage this related work. There has also been work on replacing the "thick" RLM layers with "thin" layers (called "thin streams") to enable more fine-grained adaptation [14]. Since in our scheme we add/drop individual descriptions rather than entire layers, we enjoy the same benefits as thin streams and in addition also have the benefit of robustness.

In term of support for heterogeneity in P2P multicast streaming, we are aware of a couple of different approaches. In end system multicast [1], clients choose between separate (non-layered) low-bandwidth (100 Kbps) and high-bandwidth (300 Kbps) streams. In SplitStream [4], the stream is divided into substreams that are striped across multiple trees. The number of stripes that a host subscribes to is a function of its bandwidth. The focus is on accommodating static bandwidth heterogeneity rather than dynamic fluctuations caused by congestion. There has also been work on exploiting heterogeneity to improve the efficiency and scalability of P2P overlays by assigning a greater share of the work to the more resourceful peers [5]. This is an orthogonal problem to heterogeneity support and congestion control for data transmission over such P2P overlays.

VIII. Conclusions

In this paper, we have presented the design of a bandwidth adaptation protocol for P2P multicast streaming with several novel features. First, the adaptation protocol is designed jointly with a framework for robustness that incorporates redundancy in network paths and in data. Second, parent and child nodes work in conjunction to determine the appropriate response to packet loss by exploiting tree diversity to localize the cause of packet loss. Third, both parents and children exploit knowledge of the relative importance of layered MDC substreams and the structure of the distribution trees to adapt to changing bandwidth in a way that minimizes the impact on the descendant nodes.

Although our discussion here has focussed on CoopNet for the sake of concreteness, many of the ideas have general applicability to multicast and non-multicast P2P settings. For instance, the robustness of join experiments with layered MDC would be advantageous in any RLM-like setting, even one based on a single distribution tree. Inferring the location of congestion could be useful even in an on-demand (non-multicast) streaming scenario where the receiver requests different substreams from different peers.

REFERENCES

[1] End System Multicast. http://esm.cs.cmu.edu/.

[2] BARABASI, A. L., AND ALBERT, R. Emergence of Scaling in Random Networks. *Science, 286* (Oct. 1999), 509–512.

[3] BU, T., DUFFIELD, N., PRESTI, F. L., AND TOWSLEY, D. Network Tomography on General Topologies. In *Proc. ACM SIGMETRICS* (June 2002).

[4] CASTRO, M., DRUSCHEL, P., KERMARREC, A.-M., NANDI, A., ROWSTRON, A., AND SINGH, A. SplitStream: High-bandwidth Content Distribution in a Cooperative Environment. In *Proc. SOSP* (Oct. 2003).

[5] CHAWATHE, Y., RATNASAMY, S., BRESLAU, L., AND SHENKER, S. Making Gnutella-like P2P Systems Scalable. In *Proc. ACM SIGCOMM* (Aug. 2003).

[6] CHOU, P. A., WANG, H. J., AND PADMANABHAN, V. N. Layered Multiple Description Coding. In *Proc. Packet Video Workshop* (Apr. 2003).

[7] CHU, Y., RAO, S. G., SESHAN, S., AND ZHANG, H. Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture. In *Proc. ACM SIGCOMM* (Aug. 2001).

[8] GOYAL, V. K. Multiple Description Coding: Compression Meets the Network. *IEEE Signal Processing Mag.* (Sept. 2001), 74–93.

[9] LIU, J., LI, B., AND ZHANG, Y. A Hybrid Adaptation Protocol for TCP-friendly Layered Multicast and Its Optimal Rate Allocation. In *IEEE Infocom* (June 2002).

[10] MCCANNE, S. R., JACOBSON, V., AND VETTERLI, M. Receiver-driven Layered Multicast. In *Proc. ACM SIGCOMM* (Aug. 1996).

[11] MEDINA, A., LAKHINA, A., MATTA, I., AND BYERS, J. BRITE: An Approach to Universal Topology Generation. In *Proc. MASCOTS* (Aug 2001).

[12] PADMANABHAN, V. N., WANG, H. J., AND CHOU, P. A. Resilient Peer-to-Peer Streaming. In *Proc. IEEE ICNP* (Nov. 2003).

[13] PADMANABHAN, V. N., WANG, H. J., CHOU, P. A., AND SRIPANIDKULCHAI, K. Distributing Streaming Media Content Using Cooperative Networking. In *Proc. NOSSDAV* (May 2002).

[14] WU, L., SHARMA, R., AND SMITH, B. Thin Streams, An Architecture for Multicasting Layered Video. In *Proc. NOSSDAV* (1997).