Loss Recovery in Application-Layer Multicast

Xing Jin, W.-P. Ken Yiu, and S.-H. Gary Chan
The Hong Kong University of Science and Technology

Application-layer multicast, sometimes called overlay multicast, can help circumvent the limitations in IP multicast and unicast.

Applications such as Internet TV, multiparty conferencing, and software distribution require point-to-multipoint Internet communication. In traditional IP multicast, routers replicate packets.[1] Despite its introduction more than a decade ago, IP multicast has not yet seen wide deployment, mainly because IP multicast requires multicast-capable routers, which are not scalable and need to maintain a per-group state for packet replication. Furthermore, IP multicast lacks large-scale commercial management functions, such as multicast address allocation and reliable transmission. It is for these reasons that many researchers have proposed using application-layer multicast (ALM) to move the multicast-related functionalities from routers to hosts.[2,3]

We compare IP multicast and ALM in Figure 1, where S is the source, H1 through H3 are receivers (hosts), and R1 through R5 are routers. Figure 1a shows the IP multicast case, where routers R1 to R5 replicate and forward data packets along the physical links in the router-level spanning tree rooted at S. Figure 1b shows ALM delivery, where S establishes unicast connections to H1 and H3, while H1 delivers data to H2. Therefore, hosts, instead of routers, replicate and forward packets. ALM achieves multicast via piece-wise unicast connections. In this example, hosts and unicast connections between them form an overlay tree for multicast. Other ALM schemes use multiple trees or meshes.[4,5]

There are two main types of ALM services: reliable service and loss-tolerant service. In reliable service, a source distributes data to a user pool; packets lost during this phase must be recovered. Examples include file distribution and online games. For these applications, lost packet recovery time should be as low as possible. In loss-tolerant service, a certain loss rate is acceptable. Examples include video and audio streaming. For these services, the residual loss rate should be kept below a certain threshold.

Loss recovery in ALM

In an ALM system, a host may suffer packet loss due to the following reasons:

- On the transport layer, a unicast connection can be either Transmission Control Protocol (TCP) or User Datagram Protocol (UDP). Packets can get lost when a path is congested (for a UDP connection) or fails (for both types of connections), or if the sender leaves the system or unexpectedly fails (for both types of connections).
- On the application layer, some services have requirements for data delivery. For example, a streaming application usually has a playback deadline by which data delivery and loss recovery have to be accomplished. Data received after the deadline are useless and regarded as a loss.

Therefore, timely loss recovery is important in ALM. Loss recovery in ALM is significantly different from that in IP multicast. IP multicast only supports UDP transmission, while ALM can use TCP or UDP connections. Moreover, in IP multicast, a host leaving or its failure doesn't affect other hosts. But in ALM, the leaving or failure of a host leads to packet loss at all its descendants. Clearly, ALM's high host dynamics puts more challenges on loss recovery. Furthermore, in IP multicast, any packet sent by the source will reach all the hosts in the multicast group (if there is no loss). As a result, if a host requests retransmission from the source, all the hosts in the group will receive the retransmitted packet. On the other hand, in ALM a host can freely select downstream hosts and control the transmission content. Loss recovery in ALM is therefore more flexible.

Loss recovery in ALM is also different from that in unicast. ALM works on the basis of unicast, and traditional recovery techniques for unicast can be directly applied to ALM. In

1070-986X/08/$25.00 © 2008 IEEE

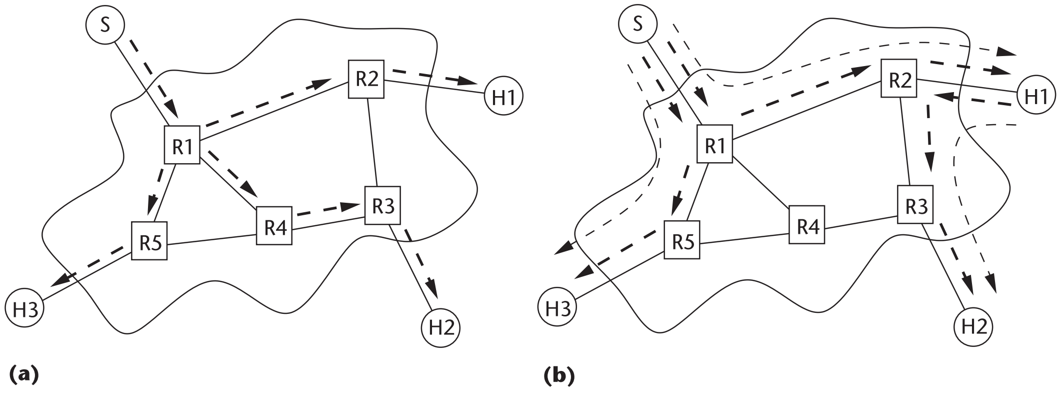


Figure 1. A comparison between (a) IP multicast and (b) application-layer multicast. In IP multicast, packets are replicated and forwarded by routers. But in application-layer multicast, this is done by hosts.

addition, the flexible overlays in ALM allow for more application-layer recovery mechanisms, such as multiple-path transmission or lateral error recovery. In ALM, hosts can cooperate to recover packet loss, and recovery is not limited to a single path.

We classify the ALM recovery mechanisms as being proactive or reactive, regardless of the underlying transport-layer connection. In a proactive approach, a host sends out redundant recovery packets in addition to the data packets. If there is data packet loss at the receiver end, the receiver can use the redundant packets to reconstruct the original data. Such approaches include erasure-resilient coding, probabilistic resilient multicast, and multiple-path transmission. A reactive approach retransmits the lost packet when loss occurs. Depending on the retransmission request's receiver, we can classify reactive approaches into source recovery, parent recovery, and recovery neighbor. Figure 2 (next page) shows the taxonomy of the loss recovery mechanisms in ALM.

An efficient loss recovery mechanism should achieve the following objectives:

- *Low residual loss rate.* Residual loss rate is the loss rate after recovery. As an analogy, we can define delivery ratio as the ratio of the amount of data packets a receiver successfully receives to the amount of data packets the sender sends. Delivery ratio is a complement of residual loss rate.

- *Low recovery latency.* Recovery latency is the time interval from the moment a loss is detected to when the repair is received. The latency should be as low as possible.

- *Low recovery overhead.* Recovery overhead is the ratio of redundant traffic to data traffic. It should be kept to a minimum.

- *Low deployment overhead.* The setup and deployment process should be as simple as possible. The costs of measurement and computation should be low.

Existing ALM schemes have different overlay reconstruction mechanisms to handle host leaving and overlay partition. This article doesn't discuss overlay reconstruction. Instead, it focuses on loss recovery on paths.

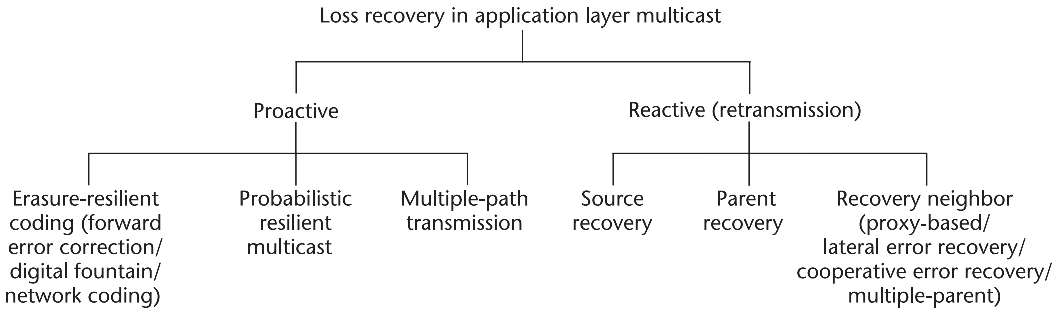
Transport-layer transmission

An ALM scheme can choose TCP or UDP connections to form the overlay, depending on its application requirements. UDP is unreliable and only provides best-effort delivery. Therefore, packets might get lost during transmission. On the contrary, TCP can provide full reliability in a connection through packet acknowledgment and retransmission.

To provide reliable data transfer among hosts, some ALM schemes employ TCP connections (for example, CoolStreaming[5]). However, TCP cannot provide end-to-end reliability for hosts. If a host leaves or fails, all the host's descendants need to reconnect to the remaining overlay and establish new TCP sessions from where they stopped. If a path fails, the receiver needs to connect to a new parent. While it's easy to reconnect to the overlay, it's not guaranteed that the data flows can be restarted from where they are stopped.

If the host buffers have finite sizes, the packets needed by the newly established TCP session might not be in the buffer. Even if the

Figure 2. A taxonomy of loss recovery mechanisms in ALM.


Loss recovery in application layer multicast
Proactive
Reactive (retransmission)
Erasure-resilient coding (forward error correction/ digital fountain/ network coding)
Probabilistic resilient multicast
Multiple-path transmission
Source recovery
Parent recovery
Recovery neighbor (proxy-based/ lateral error recovery/ cooperative error recovery/ multiple-parent)

buffer is infinitely large and hosts cache all the data they have received, hosts might have different receiving rates and the requested data in the new TCP session might not arrive at the new parent. Therefore, even when using TCP, an additional recovery mechanism is still necessary.

The same problems exist in UDP transmission, especially because UDP has high loss rate when paths congest. Whatever transport-layer connection is adopted, the issue of reducing loss rate remains in ALM systems.

Certainly, TCP and UDP connections have their own advantages and limitations. TCP is more resilient to path congestion, but it has higher overhead and a lower transmission rate than UDP. It therefore might not be applicable to high-bandwidth applications, such as real-time streaming. In streaming applications, if the transmission rate is not high enough, the application-layer loss will be high and the streaming quality will degrade. Using multiple-path transmission is a possible solution to this problem. A host can have multiple incoming connections and each connection is responsible for a portion of the data. Even though each connection has a low transmission rate, the aggregate incoming rate at the host can be high. Therefore, in bandwidth-demanding applications, TCP is often used with multiple-path transmission (for example, CoolStreaming).

Proactive recovery in ALM

Proactive recovery approaches actively send out redundant packets, which can help the receiver recover loss if some data packets get lost. Because there is no retransmission of packets, the recovery latency is low. However, the amount of redundancy is hard to tune. Although it can be adjusted dynamically to compensate for changes in the loss rate, adaptation is problematic when the network changes quickly.[6] A practical solution is to set the redundancy according to the worst-case scenario in the network, but this would result in high recovery overhead even when the loss rate gets low. Furthermore, proactive recovery cannot provide perfect reliability. Data packets might not be recovered if a large portion of packets is lost. Therefore, proactive recovery is applicable to streaming applications that don't require 100 percent reliability but require low recovery latency.

Erasure-resilient coding

In recovery based on erasure-resilient coding, original data are transmitted along with additional redundant data. A typical example is forward error correction. FEC uses (n, k) Reed-Solomon code, where k data packets are combined with ($n - k$) redundant recovery packets to form an n-packet transmission unit.[7] If a host receives more than k packets, it can recover all the data packets without error. Otherwise, it can achieve no recovery. Clearly, FEC's recovery overhead is (n - k)/k.

Another application of erasure-resilient coding is digital fountain, which uses Tornado and Luby Transform codes instead of the Reed-Solomon code.[8] In this approach, a host can reconstruct the original content of m packets from a subset of any m packets drawn from a large universe of encoding packets. In addition, another coding technique called network coding uses intermediate hosts to decode the encoded data blocks and recode them into another set of blocks before sending them out.[9] The randomization introduced by the coding process can ease data-propagation scheduling and improve system robustness.

In FEC, the selection of n and k depends on the actual loss rate on the path and the target residual loss rate. Because no recovery can be achieved when the loss is higher than a certain threshold, FEC often selects n and k to tolerate the highest possible loss rate. FEC hence has a high recovery overhead. To reduce the overhead, we can use FEC with packet retransmission. For example, OverQoS is a two-round procedure for data transmission.[6] In the first round, the method transmits the original data packets. If the loss is too high, OverQoS retransmits the lost packets with FEC protection in the second round. Because the amount of lost packets is usually small compared to the total amount of data packets, the recovery overhead is significantly reduced. However, the recovery latency is inevitably enlarged. Another approach is to use FEC and replicated packets together.[10] That is, after multicasting a set of data packets, the server continues multicasting the FEC recovery packets and some replications of the original data packets. If the FEC packets cannot reconstruct the original data, a host can selectively accept some replicated data packets.

The recovery schemes based on erasure-resilient coding transmit data and recovery packets along the same path. While this technique can address temporary packet loss on the path, it doesn't work if the path or the sender fails. An ALM scheme can combine erasure-resilient coding with multiple-path transmission to send the recovery and data packets along different paths.[11] As long as the multiple paths don't fail simultaneously, the host can achieve recovery with a high probability.

Probabilistic resilient multicast

PRM uses a randomized forwarding technique to address packet loss in ALM.[12] Hosts first form an overlay tree using any ALM scheme. Each host then randomly chooses a constant number of hosts as its extra children—besides its own children—in the overlay tree. A host forwards packets to each of these extra children with a certain probability (for example, 0.01 to 0.03). This randomized forwarding technique operates in conjunction with the normal data forwarding along the tree. Their theoretical analysis and simulation results show that careful selection of the extra children and the randomized forwarding probability lets hosts receive data with low residual loss rate and low recovery overhead.

A strength of PRM is that it works even when host fails. In a normal ALM tree, if a host fails, all its children cannot receive data until the host recovers the partition. With PRM, a host might be the extra child of other hosts. If its parent fails, it still can receive data from the extra paths with a certain probability. It's therefore applicable to streaming applications that tolerate certain data loss and require low recovery latency. Clearly, PRM has its own limitations. If the extra children are not properly selected, some hosts might encounter unbounded packet loss in a long period. And it's still not clear how the extra children should be selected to reduce the highest possible loss rate.

In fact, PRM can be regarded as a special case of multiple-path transmission. It uses one path for original data delivery and some other paths for redundant data delivery. Therefore, the path selection techniques for multiple-path transmission (for example, disjoint paths) can be applied to PRM to improve the recovery efficiency.

Multiple-path transmission

Because packets along a certain path might get lost, it's better to transmit them along multiple paths. As long as the multiple paths don't simultaneously fail or get congested, the receiver can receive at least some of the packets. Furthermore, a receiver can achieve a high aggregate incoming rate over multiple paths. This is especially useful for streaming applications, which often require high transmission rates that a single path cannot support.

In streaming applications, layered coding such as multiple description coding (MDC) is often used with multiple-path transmission.[13,14] MDC encodes data into several descriptions. When all the descriptions are received, the original data can be reconstructed without distortion. If only a subset of the descriptions are received, the quality of the reconstruction degrades gracefully. The more descriptions a host receives, the lower distortion the reconstructed data have. There are many schemes using MDC in ALM.[4] In these schemes, the server encodes its media content into *M* descriptions using MDC (where *M* is a tunable parameter), and transmits the descriptions along *M* different trees. Note that a host has different descendants in different trees. The descendant of a host in one tree is usually not the host's descendant in other trees. Therefore, packet loss at a host or failure of

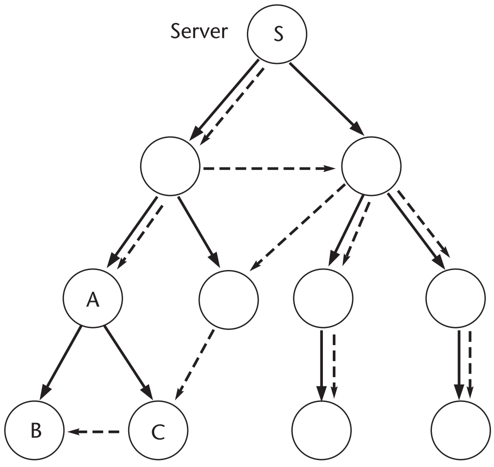


Figure 3. Building multiple trees in application-layer multicast.

the host only causes the loss of a single description (out of *M* descriptions) at each of its descendants. Figure 3 shows two multicast trees rooted at the server S, indicated by the solid and dashed lines, respectively. Each tree carries one description. If one host, say A, fails, its descendants B and C in the solid-line tree will lose one description. However, because B and C are not descendants of A in the dashed-line tree, they still can receive the other description. The transmission resilience is hence improved.

A key issue in these approaches is how to build the trees so that for any host in the system, its descendants in different trees have no or only small overlaps. One solution is to use a topology-aware hierarchical arrangement graph (THAG), where any interior host in one tree is a leaf host in all the other trees.[15] However, in each THAG tree, a large portion of hosts are leaves, which means that the interior hosts have heavy forwarding loads. A more general consideration of the issue is how to select multiple paths (or multiple parents) for a host. Many studies address this topic. For example, one study compared the performance of multiple application-, IP-, and autonomous-system-level disjoint paths in peer-to-peer (P2P) systems and recommended using autonomous-system-level disjoint paths.[16]

Multiple-path transmission has shown much lower residual loss rate then single-path transmission.[4] Most overlay networks are highly dynamic with the frequent joining and leaving of hosts. The underlying Internet is also dynamic. Routers and links might unexpectedly fail or get congested. Therefore, P2P file sharing and streaming applications have widely used multiple-path transmission for high resilience.[5] However, this approach has high control overhead for constructing multiple paths and synchronizing different data pieces. Furthermore, its transmission delay is high because the maximum path delay is always the bottleneck.

Reactive recovery in ALM

The automatic repeat request technique reactively retransmits lost packets to conduct recovery.[17] The data source usually identifies each data unit by increasing sequence numbers, while a host uses gaps in the sequence numbers to detect missing units. Depending on the retransmission request's receiver, we can develop different automatic repeat request schemes.

Source recovery

In source recovery, a host always asks the source for retransmission after a loss.[18] First developed for IP multicast, the scheme is suited for small-size ALM systems. A strength of this scheme is that the source is certain to have the lost packet. However, because packet loss at a host affects all its descendants and leads to a burst of retransmission requests, the retransmission requests might overwhelm the source if they are not aggregated. To alleviate the implosion problem at the source, a probabilistic model can delay each retransmission request by a random time.[19] However, doing so will increase recovery latency and might not work well with lots of hosts. Therefore, in large-scale ALM systems, this scheme is seldom used. Furthermore, if the source is far away, the recovery latency can be high.

Parent recovery

In parent recovery, a host requests retransmission from its parent.[20] If a parent doesn't have the requested packets, it turns to its own parent for retransmission (and so on, until the root is reached). Whenever a lost packet is recovered, the host immediately delivers it downstream. This retransmission mechanism, also known as hop-by-hop recovery, is similar to simplified TCP with no congestion control.[12]

Parent recovery is simple and effective. In parent recovery, the source has no implosion problem. Furthermore, the delay between a parent and a child is often low because most applications let hosts select close neighbors as parents. The recovery latency is not high. But this approach does not fully address all problems. Packet loss at a host leads to loss at all its descendants in the overlay tree. Therefore, loss at a downstream host is highly correlated to an upstream loss, and a problematic host's parent is likely in error. As a result, a retransmission request might get forwarded upstream multiple times before the packet is retransmitted. This leads to a large amount of retransmission requests and high recovery latency. Furthermore, parent recovery doesn't work for host or path failure. If an upstream host fails or leaves the system, or the transmission path fails, all of its descendants no longer can recover their loss and will experience service outage.

We can extend parent recovery so that a host requests retransmission not only from its parent, but also from its ancestors along the path from the source if its parent doesn't have the requested packet. This approach is more resilient to host or path failure, but still suffers the loss-correlation problem.

Using recovery neighbors

To address the implosion and loss-correlation problems, a large number of approaches uses recovery neighbors, a model in which each host in the system must select a few other hosts or proxies as recovery neighbors. Whenever a loss is detected, the retransmission requests are sent to the corresponding recovery neighbors.

The selection of recovery neighbors is not trivial. A straightforward approach is to set some predeployed hosts (for example, proxies) as other hosts' recovery neighbors.[21] Whenever a host detects a loss, it sends a retransmission request to one or more of the predeployed hosts. In spite of its simplicity, this method introduces additional deployment overhead, which can be significantly high in large-scale systems. This method is therefore not scalable.

A more intelligent solution is lateral error recovery (LER), which employs a lateral retransmission instead of a vertical retransmission from a host's ancestors.[22] LER randomly divides hosts into several planes and independently builds an overlay tree in each plane. In a plane, a host acts as the multicast tree root and is called the plane source (usually the one closest to the original source). The original source sends data to all the plane sources, which then distribute data along their own trees. Each host selects some hosts in other planes as its recovery neighbors, which are sorted according to the estimated recovery latency and ended with the plane source and the original source.

Whenever a loss occurs, the host sends a retransmission request to its recovery neighbors in the sorted order. In the example shown in Figure 4 (next page), three planes are formed. Host B in plane 2 selects host A in plane 1 and host C in plane 3 as its recovery neighbors. If B encounters packet loss, it sends a retransmit request to the recovery neighbor with the smallest recovery latency. If the retransmission fails, host B resorts to the recovery neighbor with the second smallest recovery latency, and so on. If all the recovery neighbors fail to provide the lost packet, host B resorts to the plane source and the original source.

In LER, recovery no longer depends on upstream hosts, but relies on adjacent lateral hosts. It therefore addresses the loss-correlation problem. Furthermore, if a host fails or encounters packet loss and affects its descendants, the retransmission requests from the descendants are sent to different hosts. This addresses the implosion problem. A limitation of LER is that it takes high measurement and computation overheads to select proper recovery neighbors. Some techniques have been proposed to tune the tradeoff between measurement cost and the resulting recovery latency.[22]

Cooperative error recovery (CER) is a method that selects recovery neighbors in another way.[23] It first computes the loss correlation between two hosts according to how the paths from the tree root (which is also the data source) to the hosts overlap. It then divides hosts into multiple recovery groups, where the loss correlation among hosts in the same group is minimized. A host then serves as the recovery neighbor of other hosts in the same group. The advantage of CER is the small probability that a packet simultaneously gets lost at a host and its recovery neighbors. It does address the loss-correlation problem, but doesn't reduce recovery latency when selecting recovery neighbors. And in LER and CER, the recovery loads on hosts are not balanced. It's possible that many hosts might select a particular host as their recovery neighbor, in

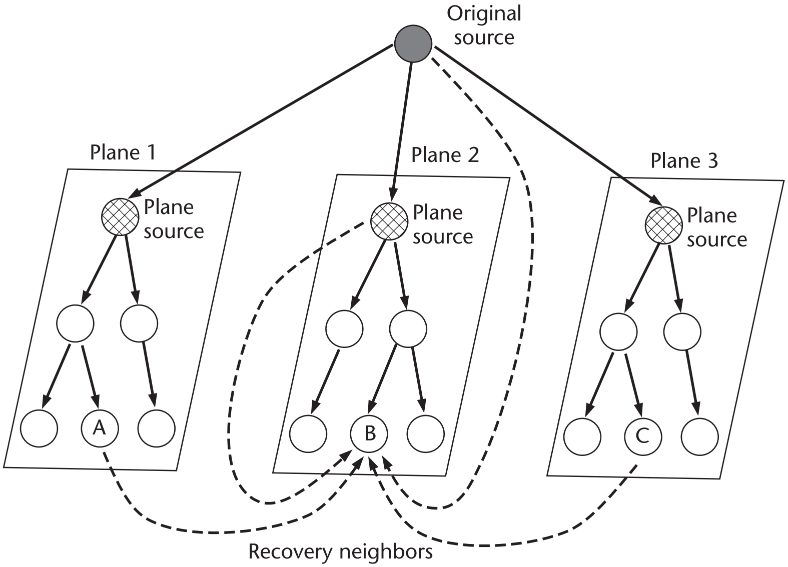


Figure 4. An example of lateral error recovery.

which case retransmission requests might overload such a host.

A general extension of the recovery neighbors is using multiple parents for data delivery. A typical example is mesh-based P2P streaming. In one study, hosts form a mesh using traditional gossip algorithms.[5] The hosts then periodically exchange data availability information with their neighbors and further exchange streaming data. In this approach, each neighbor becomes a recovery neighbor; so when a host identifies the data it lacks, it contacts its recovery neighbors for the data. This approach is resilient to system and network dynamics and has low residual loss rate. However, the recovery latency is high if the neighbors are far away—which is often the case in the randomly formed gossip mesh.

Discussion and comparison

Table 1 summarizes the qualitative comparison results of the recovery schemes we've discussed.

All the recovery schemes are applicable to UDP, because they all handle packet loss relating to path congestion. If TCP is used, packet loss due to path congestion is inherently solved by TCP retransmission mechanisms. We therefore only need to address path failure and sender failure issues. But only using erasure-resilient coding cannot address these two problems because they transmit data packets and redundant packets along the same path. If the path or the sender fails, no redundant packets can be received. Similarly, parent recovery is not applicable to TCP. All the remaining schemes can be jointly used with TCP because they all transmit data packets and recovery packets along different paths.

Proactive recovery schemes have no explicit recovery latencies because recovery packets are proactively sent with data packets, regardless of packet loss. We regard the recovery latency as the transmission latency of packets, which is small compared to reactive recovery because the receiver doesn't need to wait for packet retransmission. Among the three proactive recovery schemes, erasure-resilient coding has small recovery latency while PRM and multiple-path transmission have relatively large recovery latencies. The reason for the differences in latency is because in the latter two schemes, data is transmitted over multiple paths and the longest path enlarges the overall transmission delay.

In reactive recovery, parent recovery performs retransmission from the current parent, which is often close to the host. However, the loss-correlation problem enlarges the recovery latency. Using recovery neighbors also doesn't lead to high recovery latency because recovery neighbors often are close to the corresponding host. On the other hand, source recovery has high recovery latency because the source often is a fixed point on the Internet regardless of other hosts' location. If the source is far away, the recovery latency can be high.

Given a certain target residual loss rate, proactive approaches generally have higher recovery overhead than reactive approaches. This is because reactive approaches only need to retransmit lost packets, where the cost is proportional to the amount of lost packets. However, in proactive approaches, the parameters are often set according to the network's worst case. As a result, even when the loss rate becomes low, the recovery overhead is still high. Note that in multiple-path transmission using layered coding or MDC, the recovery overhead can be high because such coding is inefficient.[14]

In the proactive approaches, multiple-path transmission has the highest deployment overhead. It needs to construct and maintain multiple paths and schedule data delivery among the paths. In PRM, if each host only randomly selects extra children, the overhead isn't high. Its overhead will increase with more

Table 1. Comparison of loss recovery mechanimsms in ALM.

Scheme		Recovery mechanism	UDP/TCP applicable	Recovery latency		Recovery overhead	Deployment overhead
Proactive	Erasure-resilient coding	Encode data and use redundant packets	Yes/no	Equal to transmission latency	Small	Moderate	Moderate
	Probabilistic resilient multicast	Hosts selects other hosts as extra children	Yes/yes		Large	Moderate	Moderate
	Multiple-path transmission	Multiple paths	Yes/yes		Large	Moderate	High
Reactive	Source recovery	Send retransmission request to source	Yes/yes	High		Low	Low
	Parent recovery	Send retransmission request to parent	Yes/no	Moderate		Low	Low
	Recovery neighbor	Hosts send retransmission requests to recovery neighbors	Yes/yes	Moderate		Low	High

complicated selection mechanisms. In the reactive approaches, using recovery neighbors has high deployment overhead. It's not easy to select a qualified recovery neighbor with low delay and high uploading bandwidth. On the contrary, source recovery and parent recovery have low deployment overhead. A host only needs to know the source or the parent.

A combination of proactive and reactive approaches might make for an efficient recovery scheme in terms of both overhead and latency. Furthermore, some schemes have combined erasure-resilient coding with proactive or reactive multiple-path transmission. Multiple-path transmission is highly resilient to host and network dynamics, but it has high end-to-end delay and high control overhead. Using erasure-resilient coding (such as Luby Transform or network coding) can reduce both data redundancy over multiple paths and end-to-end delay (including recovery latency).[9,11] Using an overlay recovery tree is another way to reduce recovery latency in multiple-path transmission.[24] In this scheme, hosts use a mesh overlay for normal data transmission. Hosts also form a low-delay overlay tree. If a host fails to receive a data segment within a certain time, it resorts to its parent in the overlay tree for the missing segment. As the end-to-end delay in a tree is often smaller, the recovery latency is therefore reduced.

We quantitatively compared representative schemes through simulations, including PRM, source recovery, parent recovery and LER. We generated several Transit-Stub topologies with Georgia Tech Internetwork Topology Models. In these tests, we randomly connected a host to a stub router with a 1-millisecond delay, while the delays of core links are given by the topology generator. We choose the Delaunay Triangulation ALM scheme as the basis.[3] Packets are randomly dropped in a network link, and with probability 0.95 the loss rate is uniformly distributed between 0 and 1 percent. With probability 0.05, the loss rate is uniformly distributed between 5 and 10 percent. Each host has a certain failure probability of 5 percent. All these error hosts fail for a certain time, with the failure time uniformly and independently distributed within 640 seconds. For PRM, we set the parameters as in Banarjee et al.[12] That is, each host selects three extra children and forwards data to them with a probability of 0.01. For LER, we divide hosts into two planes and simulate two versions of LER, depending on the selection of recovery neighbors. We can use global network positioning to find the closest host in a plane as the recovery neighbor, or randomly ping 10 hosts in the plane and choose the closest one.[22]

We evaluated the schemes for streaming, where the playback deadline is set to 2 seconds. Figure 5 (next page) shows the residual loss rate versus the number of hosts. Similar results have been presented elsewhere.[22] Generally, the loss rate increases with the number of hosts, because the tree is deeper and packets are more likely to be lost. The performance without loss recovery is clearly unacceptable, and all the recovery schemes can significantly reduce the residual loss rate. Among the

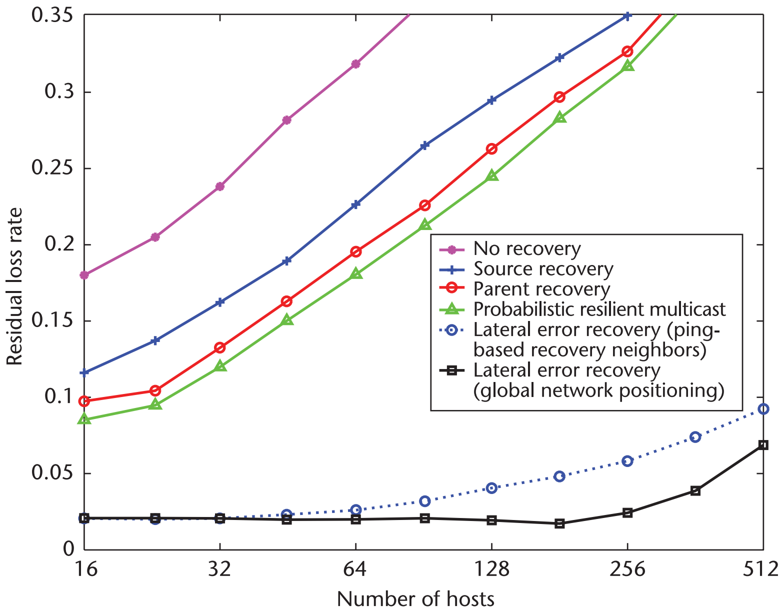


Figure 5. Residual loss rate for streaming applications.

schemes, source recovery has the highest residual loss rate, mainly due to the relatively long distance between the source and a host. Parent recovery and PRM are better.

PRM employs similar retransmission mechanism as in parent recovery. It also uses extra connections to enhance performance. It's therefore slightly better than parent recovery. LER achieves by far the lowest loss rate among the schemes. In addition, its results are the most stable when the host number increases. The two versions of LER trade off between residual loss rate and measurement cost. As shown, LER can address the loss correlation and implosion problems efficiently.

Conclusions

In this article, we discuss and compare different recovery mechanisms in ALM. The major challenge in loss recovery is how to achieve low residual loss rate with low recovery overhead. As discussed, a promising approach might be a combination of proactive and reactive techniques.

As wireless access becomes popular nowadays, there will be great demand for systems to accommodate mobile devices. This introduces more challenges for loss recovery, as connections in wireless networks are even more dynamic and unstable, and wireless transmission uses broadcast instead of unicast. Clearly, efforts are needed to cope with the challenges. **MM**

Acknowledgments

The Innovation and Technology Commission of the Hong Kong Special Administrative Region, China (GHP/045/05) supported this work, in part.

References

1. S.E. Deering, ''Multicast Routing in Internetworks and Extended LANs,'' *ACM Sigcomm Computer Comm. Review*, vol. 18, no. 4, 1988, pp. 55-64.
2. Y.H. Chu et al., ''A Case for End System Multicast,'' *IEEE J. Selected Areas Comm.*, vol. 20, no. 8, 2002, pp. 1456-1471.
3. J. Liebeherr, M. Nahas, and W. Si, ''Application-Layer Multicasting with Delaunay Triangulation Overlays,'' *IEEE J. Selected Areas Comm.*, vol. 20, no. 8, 2002, pp. 1472-1488.
4. V. Padmanabhan et al., ''Distributing Streaming Media Content Using Cooperative Networking,'' *Proc. ACM Int'l Workshop Network and Operating Systems Support for Digital Audio & Video* (Nossdav), ACM Press, 2002, pp. 177-186.
5. X. Zhang et al., ''CoolStreaming/DONet: A Data-Driven Overlay Network for Peer-to-Peer Live Media Streaming,'' *Proc. IEEE Infocom*, IEEE Press, 2005, pp. 2102-2111.
6. L. Subramanian et al., ''OverQoS: An Overlay Based Architecture for Enhancing Internet QoS,'' *Proc. Usenix Symp. Networked Systems Design and Implementation*, Usenix Press, 2004, pp. 71-84.
7. J. Nonnenmacher, E. Biersack, and D. Towsley, ''Parity-Based Loss Recovery for Reliable Multicast Transmission,'' *IEEE/ACM Trans. Networking*, vol. 6, no. 4, 1998, pp. 349-361.
8. J. Byers, M. Luby, and M. Mitzenmacher, ''A Digital Fountain Approach to Asynchronous Reliable Multicast,'' *IEEE J. Selected Areas Comm.*, vol. 20, no. 8, 2002, pp. 1528-1540.
9. C. Gkantsidis and P. Rodriguez, ''Network Coding for Large Scale Content Distribution,'' *Proc. IEEE Infocom*, IEEE Press, 2005, pp. 2235-2245.
10. S.-H. Chan et al., ''Video Loss Recovery with FEC and Stream Replication,'' *IEEE Trans Multimedia*, vol. 8, no. 2, 2006, pp. 370-381.
11. C. Wu and B. Li, ''rStream: Resilient Peer-to-Peer Streaming with Rateless Codes,'' *Proc. ACM Multimedia*, ACM Press, 2005, pp. 307-310.
12. S. Banerjee et al., ''Resilient Multicast Using Overlays,'' *IEEE/ACM Trans. Networking*, vol. 14, no. 2, 2006, pp. 237-248.
13. V. Goyal, ''Multiple Description Coding: Compression Meets the Network,'' *IEEE Signal Processing Mag.*, vol. 18, no. 5, 2001, pp. 74-93.

14. B. Li and J. Liu, ''Multirate Video Multicast over the Internet: An Overview,'' *IEEE Network*, vol. 17, no. 1, 2003, pp. 24-29.
15. R. Tian et al., ''Robust and Efficient Path Diversity in Application-Layer Multicast for Video Streaming,'' *IEEE Trans. Circuits and System for Video Technology*, vol. 15, no. 8, 2005, pp. 961-972.
16. T. Fei et al., ''How to Select a Good Alternate Path in Large Peer-to-Peer Systems?'' *Proc. IEEE Infocom*, IEEE Press, 2006, pp. 1-13.
17. S. Lin, D.J. Costello, and M.J. Miller, ''Automatic-Repeat-Request Error-Control Schemes,'' *IEEE Comm. Mag.*, vol. 22, no. 12, 1984, pp. 5-17.
18. M.S. Lacher, J. Nonnenmacher, and E.W. Biersack, ''Performance Comparison of Centralized Versus Distributed Error Recovery for Reliable Multicast,'' *IEEE/ACM Trans. Networking*, vol. 8, no. 2, 2000, pp. 224-238.
19. J. Nonnenmacher and E.W. Biersack, ''Scalable Feedback for Large Groups,'' *IEEE/ACM Trans. Networking*, vol. 7, no. 3, 1999, pp. 375-386.
20. P. Radoslavov et al., ''A Comparison of Application-Level and Router-Assisted Hierarchical Schemes for Reliable Multicast,'' *IEEE/ACM Trans. Networking*, vol. 12, no. 3, 2004, pp. 469-482.
21. J. Gemmell et al., ''The PGM Reliable Multicast Protocol,'' *IEEE Network*, vol. 17, no. 1, 2003, pp. 16-22.
22. W.-P. Yiu et al., ''Lateral Error Recovery for Media Streaming in Application-Level Multicast,'' *IEEE Trans. Multimedia*, vol. 8, no. 2, 2006, pp. 219-232.
23. G. Tan and S.A. Jarvis, ''Improving the Fault Resilience of Overlay Multicast for Media Streaming,'' *Proc. IEEE/IFIP Int'l Conf. Dependable Systems and Networks*, IEEE Press, 2006, pp. 558-567.
24. M. Zhou and J. Liu, ''A Hybrid Overlay Network for Video-on-Demand,'' *Proc. IEEE Int'l Conf. Communications*, IEEE Press, 2005, pp. 1309-1313.

Xing Jin is working toward a PhD in the Department of Computer Science and Engineering at the Hong Kong University of Science and Technology, Kowloon. His research interests include overlay multicast, Internet topology inference, end-to-end measurements, and peer-to-peer streaming. Jin has a B.Eng. in computer science and technology from Tsinghua University, Beijing. He received a Microsoft Research Fellowship in 2005. He is a junior editor of the *Journal of Multimedia* and a student member of the IEEE Computer Society. Contact him at csvenus@cse.ust.hk.

W.-P. Ken Yiu is working toward a PhD in the Department of Computer Science and Engineering at the Hong Kong University of Science and Technology, Kowloon. His research interests include computer networks, peer-to-peer systems, multimedia networking, and network security. Yiu has a B.Eng. and M.Phil. in computer science from the Hong Kong University of Science and Technology, Kowloon. He received the Academic Achievement Medal from HKUST in 2002, and the Sir Edward Youde Memorial Fellowship in 2005 and 2006. He is a student member of the IEEE Computer Society. Contact him at kenyiu@cse.ust.hk.

S.-H. Gary Chan is an associate professor in the Department of Computer Science and Engineering at the Hong Kong University of Science and Technology, Kowloon. His research interests include multimedia networking, peer-to-peer technologies and streaming, and wireless communication networks. Chan has a PhD from Stanford University in electrical engineering. He is a member of Tau Beta Pi, Sigma Xi, and Phi Beta Kappa, and a senior member of the IEEE Computer Society. Contact him at gchan@cse.ust.hk.