



Active Reliable Multicast

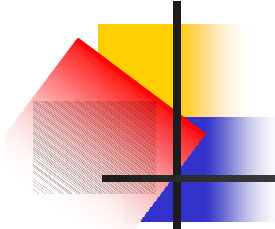
Li-wei H. Lehman, Stephen
J. Garland, and David L.
Tennenhouse

Presented by Youngsuk Jun



Multicast Routing

- Connection-oriented
- Only a single copy of datagram traverse a link
- Address indirection – class D multicast group
- IGMP – group membership protocol, operates locally
- Receiver-driven



Internet Philosophy

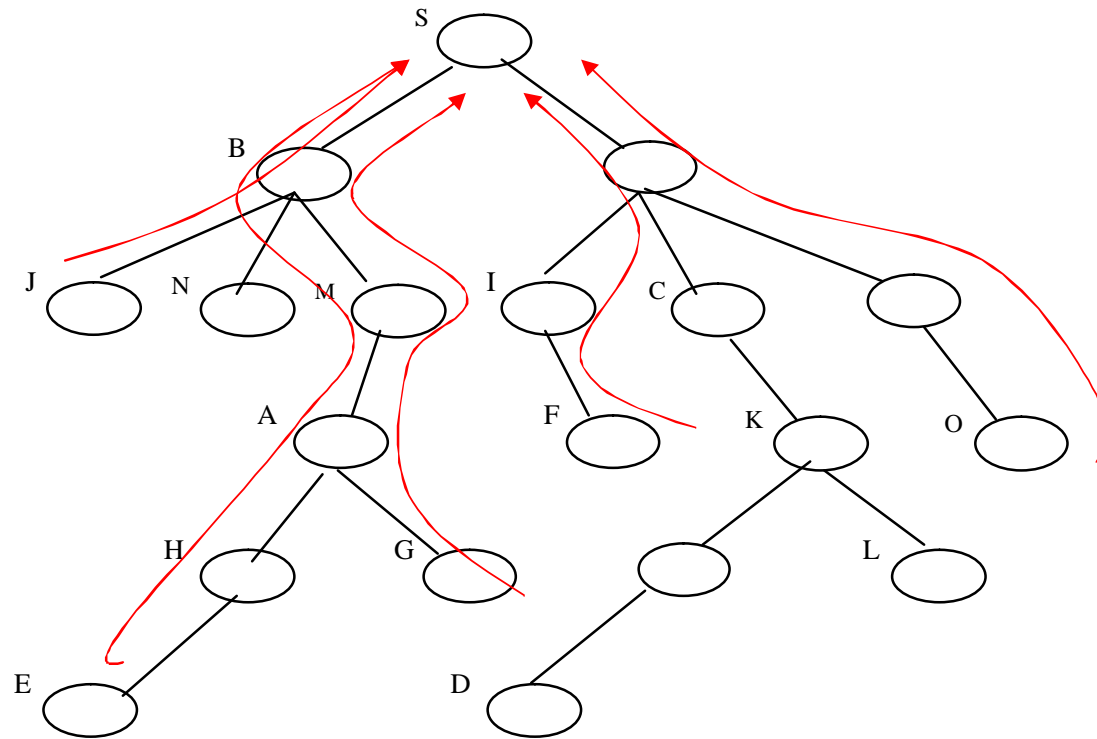
- “Whether the minimalist network layer philosophy will equally successful for the multicast service model?”
- “Do we need to move functionality from edge host to network layer?”



Reliable Multicast Problems

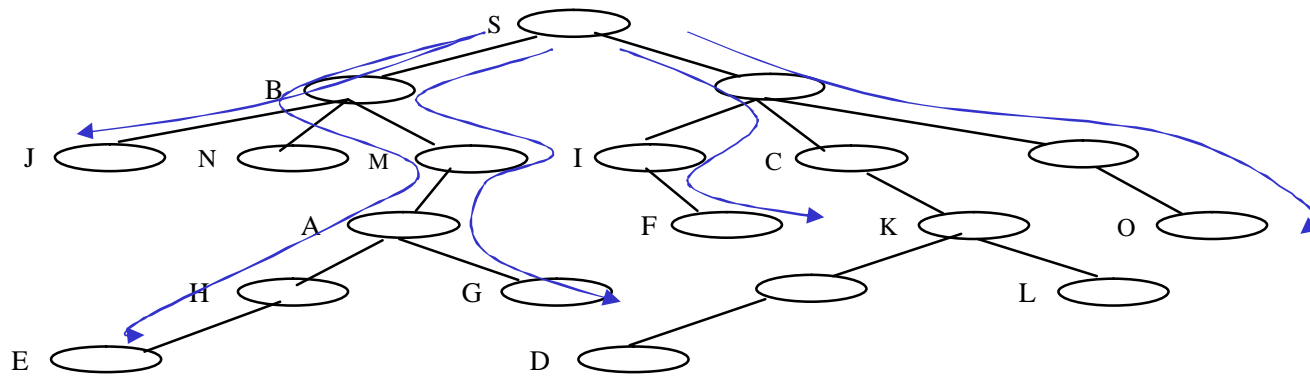
- Limited capacity for data loss report & response for Multicast
 - NACK Implosion
 - Host or Network Exposure
- Scale loss recovery to large multicast group
 - Control NACK Implosion
 - Distribute the load for retransmission
 - Limit the delivery scope of retransmitted packet
- Frequent group membership change

NACK Implosion

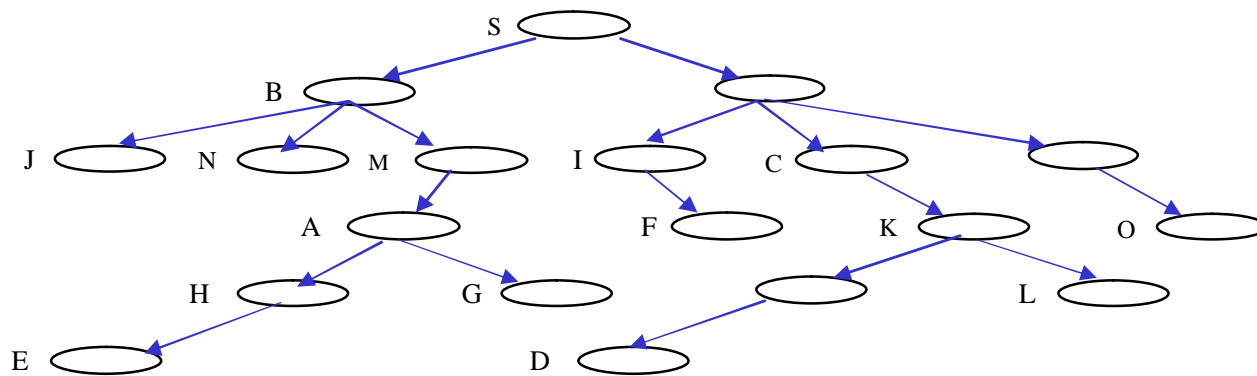


Implosion

Network & Host Exposure



Network Exposure



Host exposure

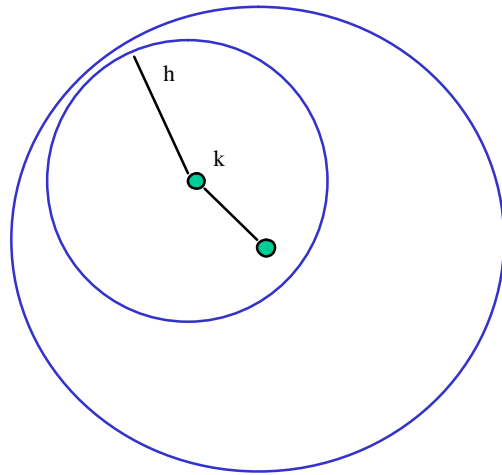


Existing Scheme

- Provide only partial solutions
 - SRM
 - trades recovery latency off against repair BW
 - NACK suppression: using random time back-off to avoid implosion
 - Any member can respond retransmission request
 - Local recovery in SRM is still an open issue
 - LBRM, TMTP, RMTP
 - Organize receivers in hierarchies
 - Scoped recovery using designated receiver
 - Less robust to topology change
 - Do not protect the proxy's bottleneck links from being overloaded

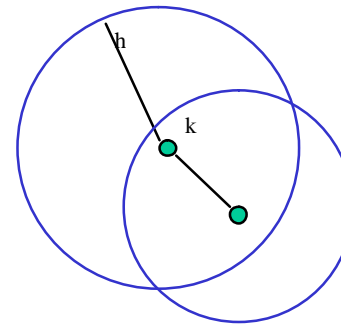


Local recovery in SRM



Request TTL h
Repair TTL $h+k$

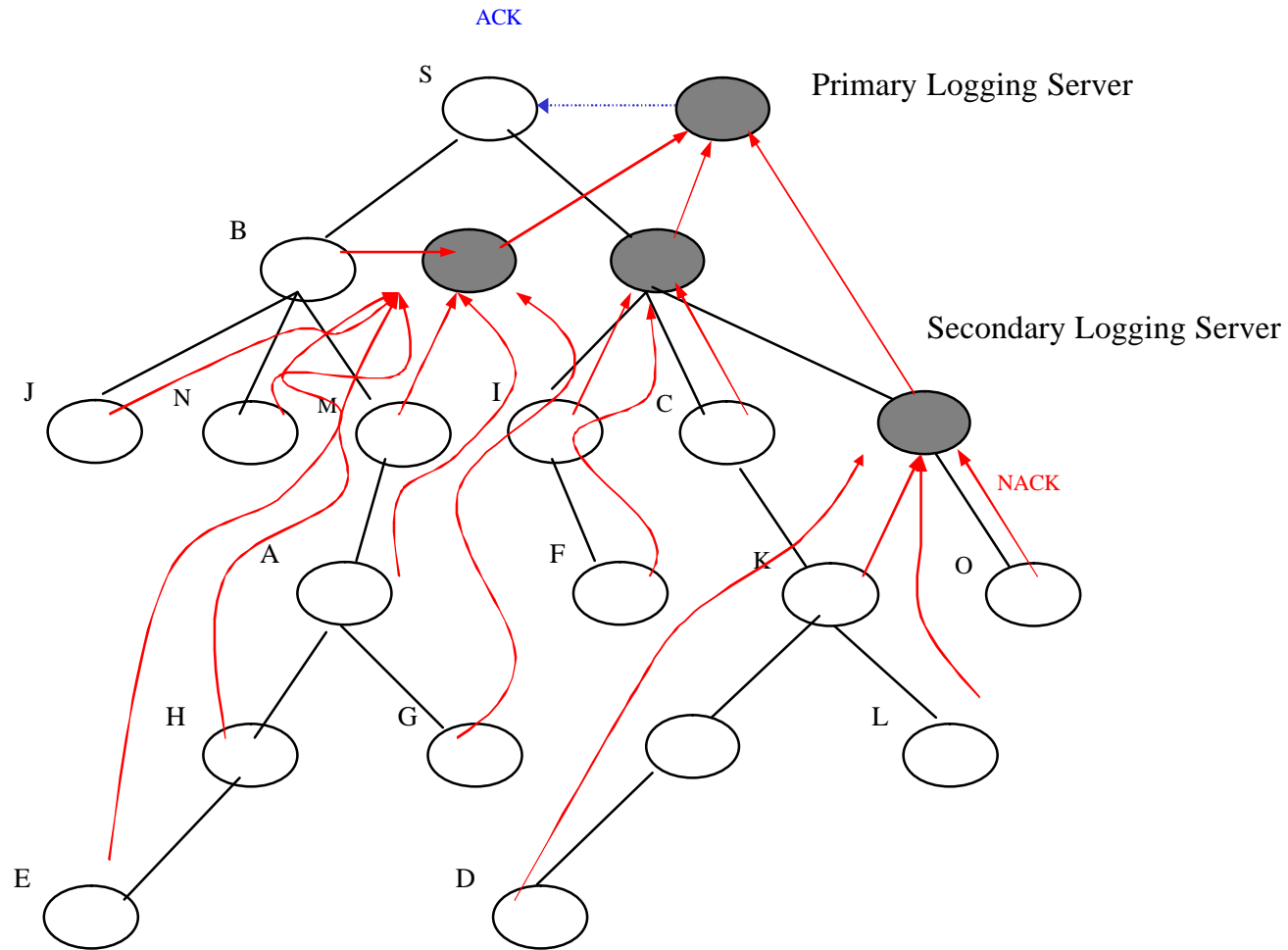
(a) one-step repair



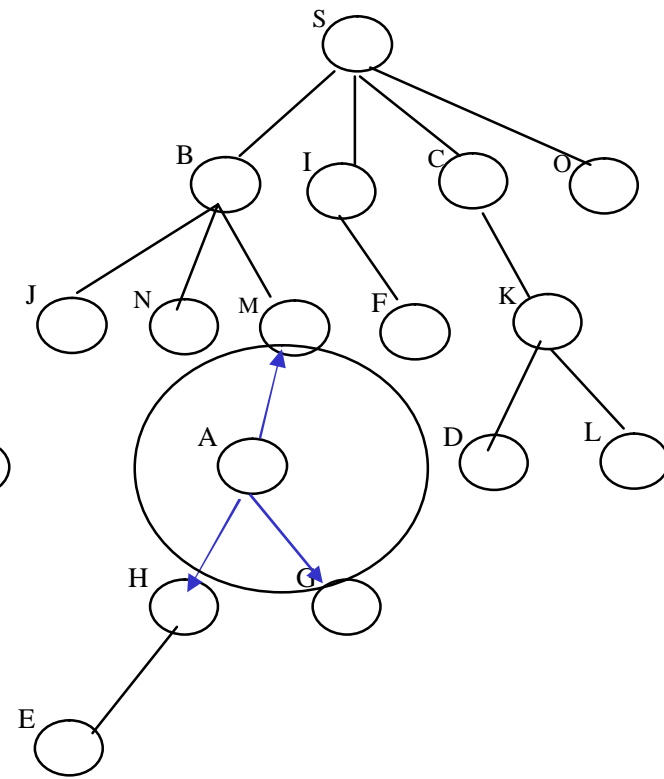
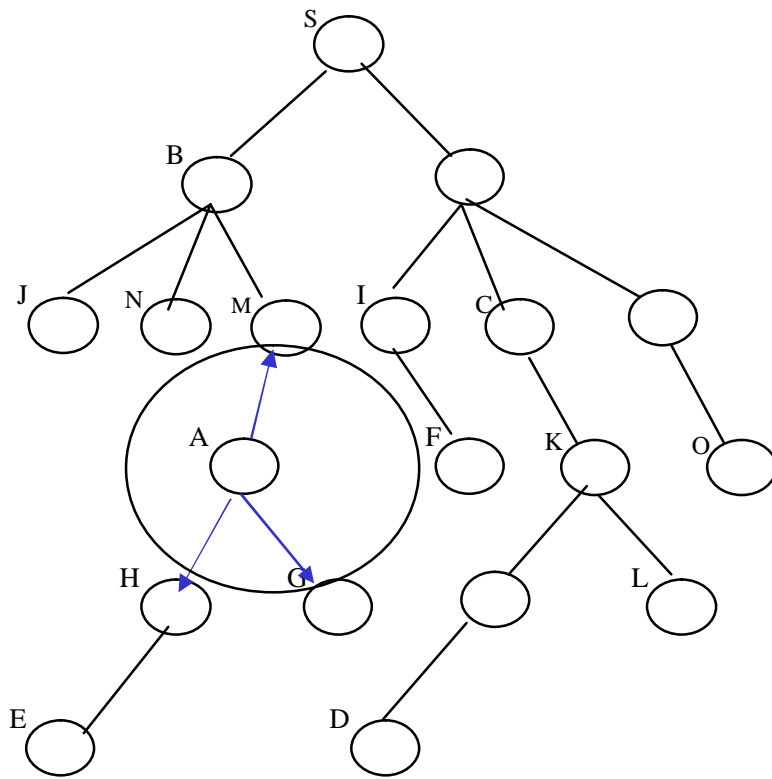
Request TTL h
Local repair TTL $> k$
Repair TTL h

(b) two-step repair

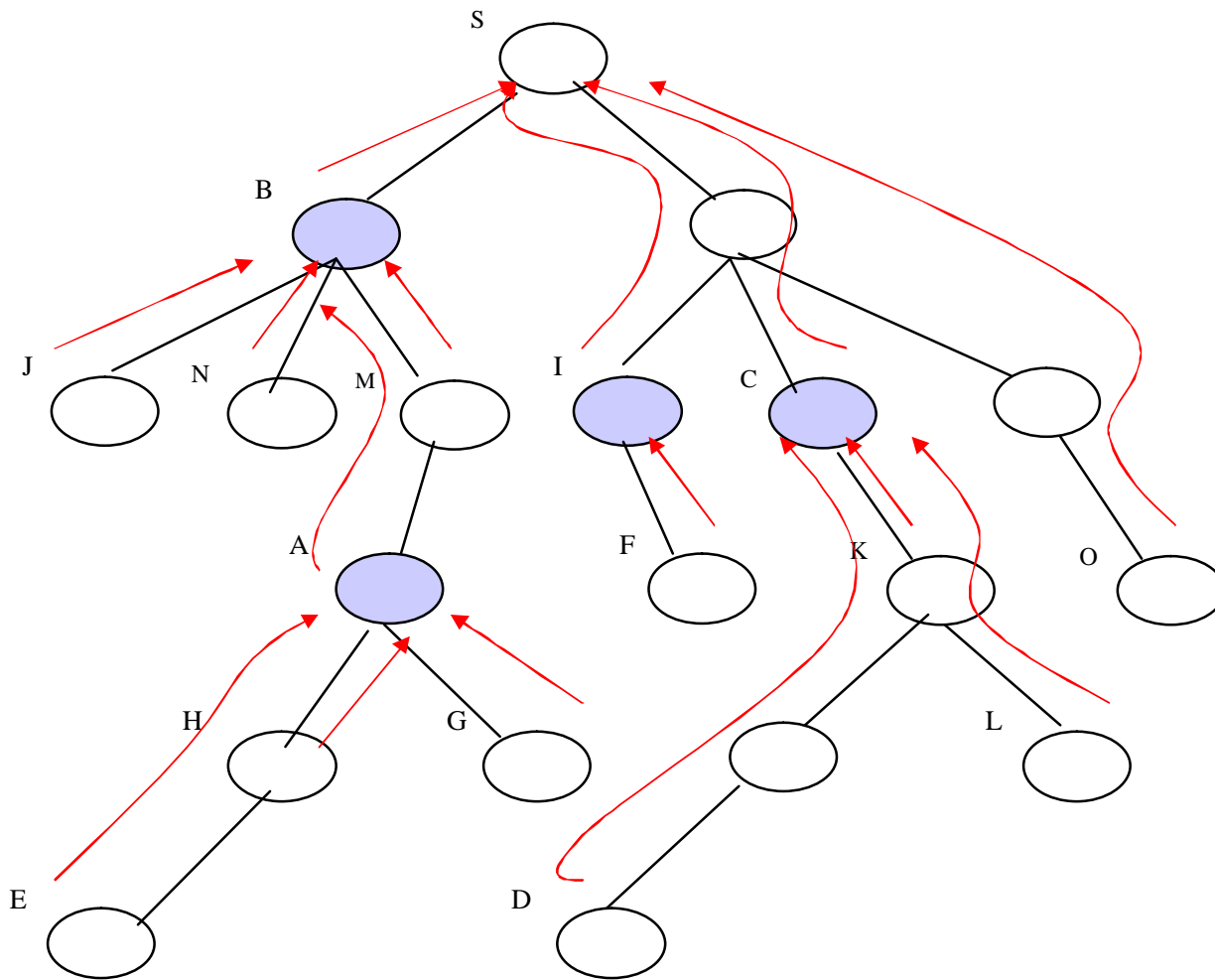
Log Based Reliable Multicast



Tree-based Multicast Transport Protocol



Reliable Multicast Transport Protocol





Network model

- End-points can take advantage of network-based processing and storage
- Active nodes provides fixed amount of “best-effort” soft-state storage and perform customized computation
- A tree rooted at the sender is formed to deliver multicast packet
- Paths for multicast routing correspond to the reverse paths for unicast routing



Active Reliable Multicast

- Receiver-reliable, NACK-based scheme
 - NACK-scalable, infinite buffer (heart-beat), quick loss detection
 - ACK-not scalable, easy to control buffer
- Receiver detect losses by sequence gaps in the data packets
- Multiple NACKs from different receivers are cached and fused at active routers along the multicast tree
- Sender responds to the first NACK by multicasting a repair to the group: ignores subsequent NACKs for this packet for fixed amount of time
- Sender receive higher NACK count than it maintained: retransmission is lost
- ARM is robust to group topology change



Intermediate active router functions

- Data caching for local retransmission
 - Best-effort caching(soft-state storage) of multicast data for possible retransmission
- NACK fusion/suppression
 - Control implosion by dropping duplicated NACKs and forwarding only one NACK upstream
- Partial multicasting for scoped retransmission
 - Repair packets delivered only to receivers that previously requested them



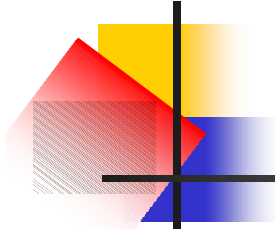
Caching data at router

- Caching can significantly reduce the recovery latency for distant receiver
- Distributing retransmission load: protect bottleneck link and sender
- Caching time: function of inter-packet sending rate and MAX RTT
- Most packet losses occurs at the edge: locating caches where BW is scarce



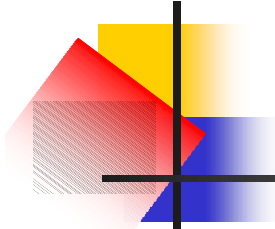
Processing multicast data packet

```
Algorithm for data packet DP:
If (cache is available at this node) {
    Store DP in cache;
    Set DP's cache TTL to DP.t;
}
For each (outgoing link) {
    If (downstream receivers are subscribed
        to DP.group) {
        Forward DP down link;
    }
}
```

NACK suppression and local retransmission

- Prevent unnecessary traffic from propagating beyond an active node
- Provide subscription info for repairs
- Trigger local retransmission
- Cache a NACK record, a REPAIR record, and data packet itself



NACK suppression and local retransmission

- NACK record
 - Suppress subsequent duplicate NACKs
 - Determine outgoing links of subsequent repair packet
- REPAIR record
 - Indicate outgoing links on which repair packet already forwarded
 - Suppress NACKs before receiver receive repair packet in transit
- Preparation of scoped retransmission
- Caching time: RTT between sender and “farthest” receiver in group



Processing NACK packet

```
Algorithm for NACK packet NP arriving on
link k:
Look up unexpired NACK record NR, REPAIR
record RR, and data or repair packet DP
for (NP.group, NP.source, NP.seqNumber);

If (RR found && RR's NACK count for link k
    >= NP.nackCount) {
    // Do nothing
} Else If (DP found) {
    Set DP's packet type to REPAIR;
    Set DP's NACK count to NP.nackCount;
    Deliver DP down link k;
    If (RR not found) {
        Create REPAIR record RR for
        (DP.group, DP.source, DP.seqNumber);
    }
    Set RR's cache TTL to NP.t;
    Set RR's NACK count for link k to
    NP.nackCount;
} Else If (NR found &&
    NR contains subscribed link &&
    NR.nackCount >=
    NP.nackCount) {
    Subscribe link k to repair packet;
} Else {
    If (NR not found) {
        Create NACK record NR for
        (NP.group, NP.source, NP.seqNumber);
    }
    Set NR.nackCount to NP.nackCount;
    Set NR's cache TTL to NP.t;
    Subscribe link k to repair packet;
    Forward NP toward the source;
}
```



Scoped retransmissions

- By looking up subscription bitmap
- Forwards a repair only to subscribed links
- Cache repair packet than original packet

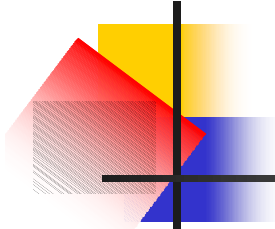


Processing repair packet

```
Algorithm for repair packet RP:
Look up unexpired NACK record NR and
REPAIR record RR for
(RP.group, RP.source, RP.seqNumber);

If (cache available at this node) {
  Store RP in cache;
  Set RP's cache TTL to RP.t;
  If (NR found) {
    Forward RP down each subscribed
      link in NR;
    Remove NR;
  }
} Else {
  For each (outgoing link) {
    If (downstream receivers are subscribed
      to RP.group) {
      Forward RP down link;
    }
  }
}

If (RR not found) {
  Create REPAIR record RR for
    (RP.group, RP.source, RP.seqNumber);
  Set RR's cache TTL to RP.t;
}
For (each link i on which RP
  was forwarded) {
  Set NACK count for i in RR to
    RP.nackCount;
}
```



Implementation

- Language: Java
- O/S: Solaris 2.5
- Prototyping: Active Node Transport System(ANTS)
- Capsule code fragment can be “demand-loaded”: loaded into active node during multicast session initialization
- LRU-based soft state storage
- Shortest path based multicast
- Communicate through UDP



Simulations

- Recovery Latency
 - Improvement in recovery latency can be obtained when fewer than 20% of the routers cache fresh multicast data packets
- Implosion Control
 - Able to control implosions when fewer than 50% of routers are active
- Bandwidth Consumption
 - NACK packets consume less BW in ARM, which performs well when fewer than 50% of routers are active
 - Repair packets also consume less BW in ARM.
 - Significant benefits are obtained from scoped retransmission and cached repairs when only 40% of routers are active

Loss recovery delay

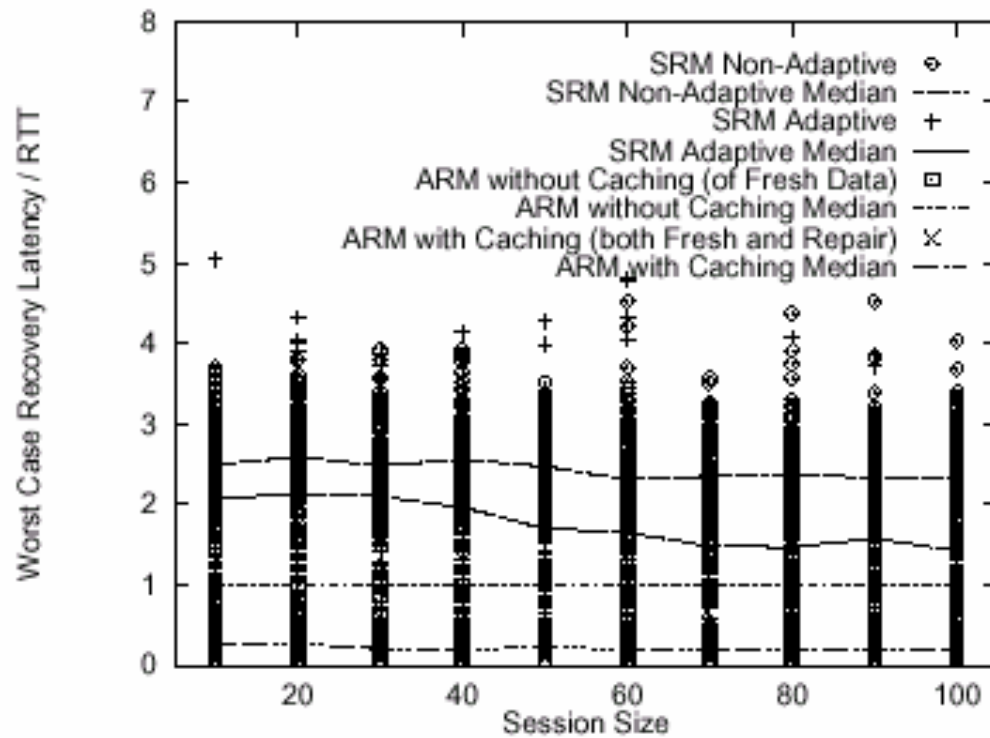


Fig. 1. ARM vs. SRM worst case recovery delay (random loss, 1000 nodes, degree 4). ARM results shown when router caches of fresh multicast data are enabled and disabled. SRM results shown for both non-adaptive and adaptive algorithms.

End-to-end recovery latency

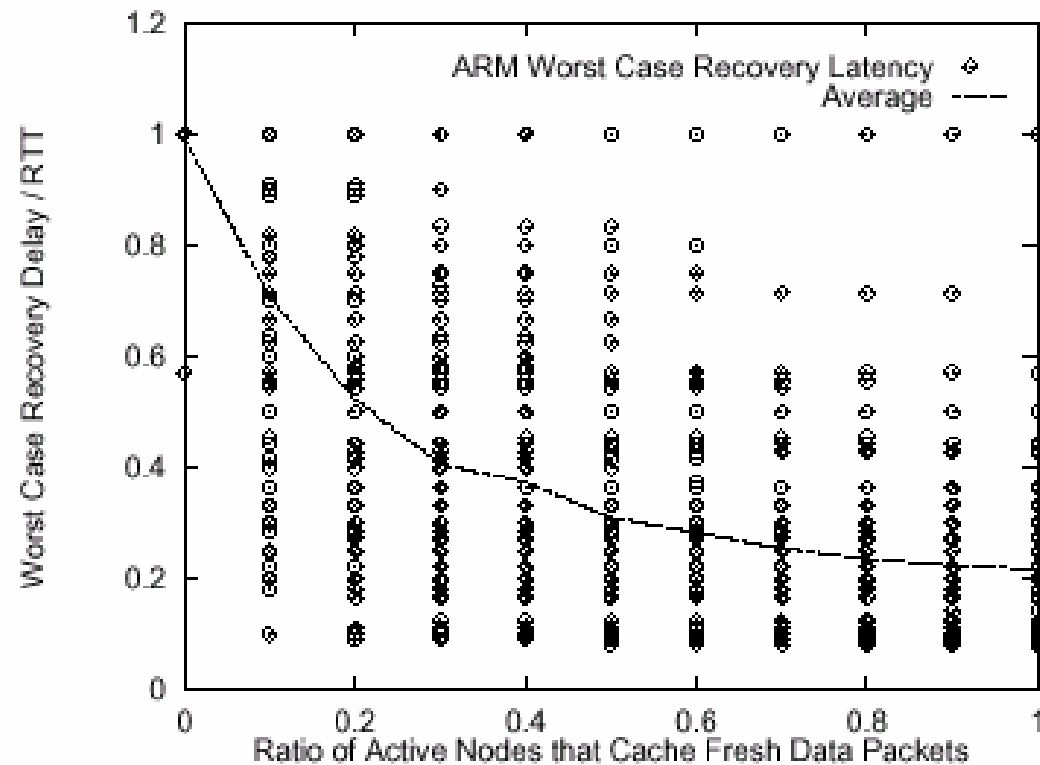


Fig. 2. ARM tradeoff between caching of fresh multicast data and latency (random loss, group size 100, 1000 nodes, degree 4). All non-leaf nodes in multicast tree are active; caching of repair packets is enabled at all nodes.

NACK implosion control

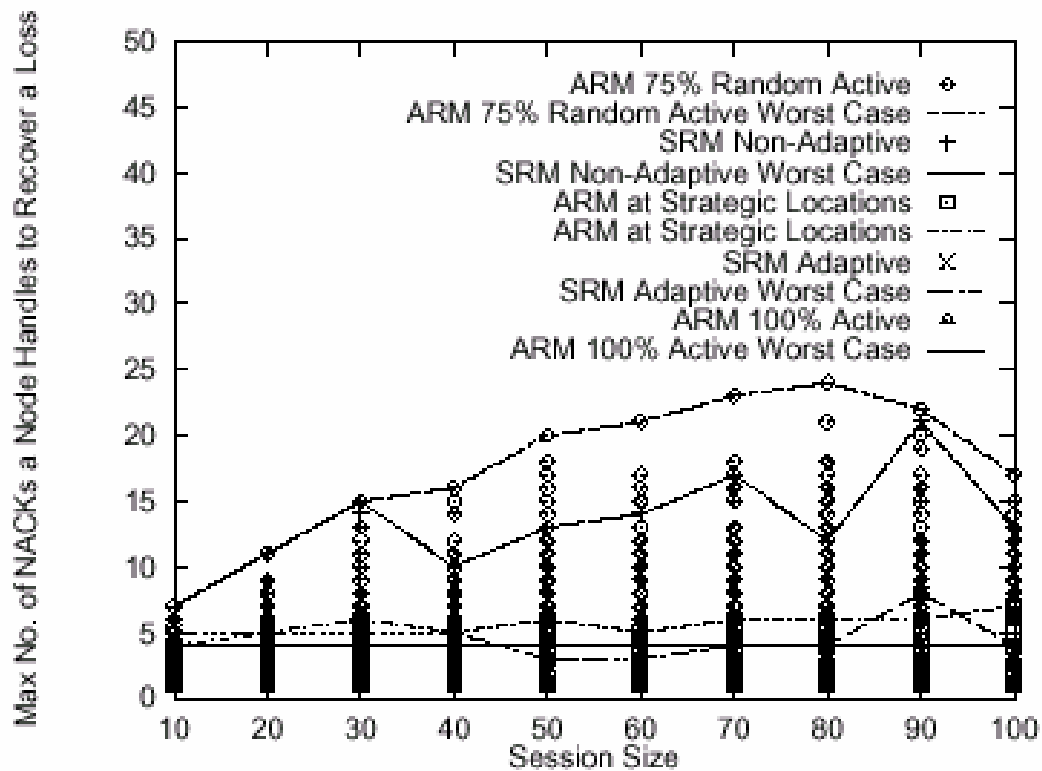


Fig. 3. NACK implosion control (loss near source, 1000 nodes, degree 4). ARM caches repairs, but not fresh data packets.

NACK bandwidth consumption

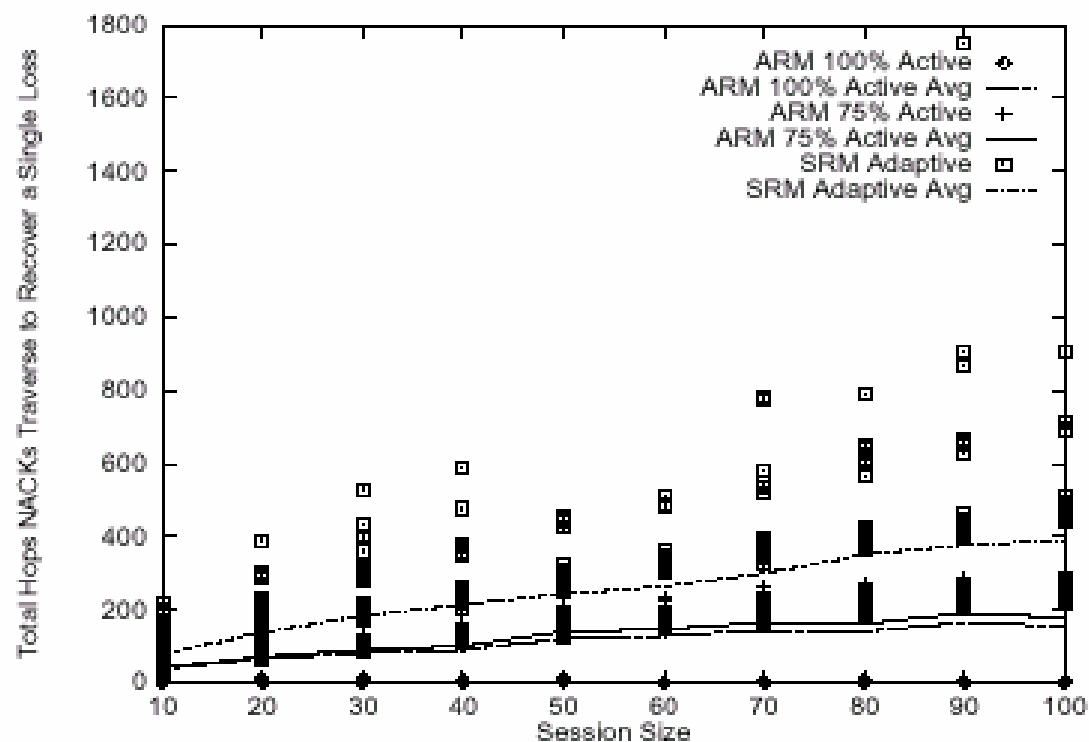


Fig. 4. NACK bandwidth consumption (loss near source, 1000 nodes, degree 4). ARM caches repairs, but not fresh data packets. SRM uses adaptive algorithm.

Ratio of “strategic” nodes

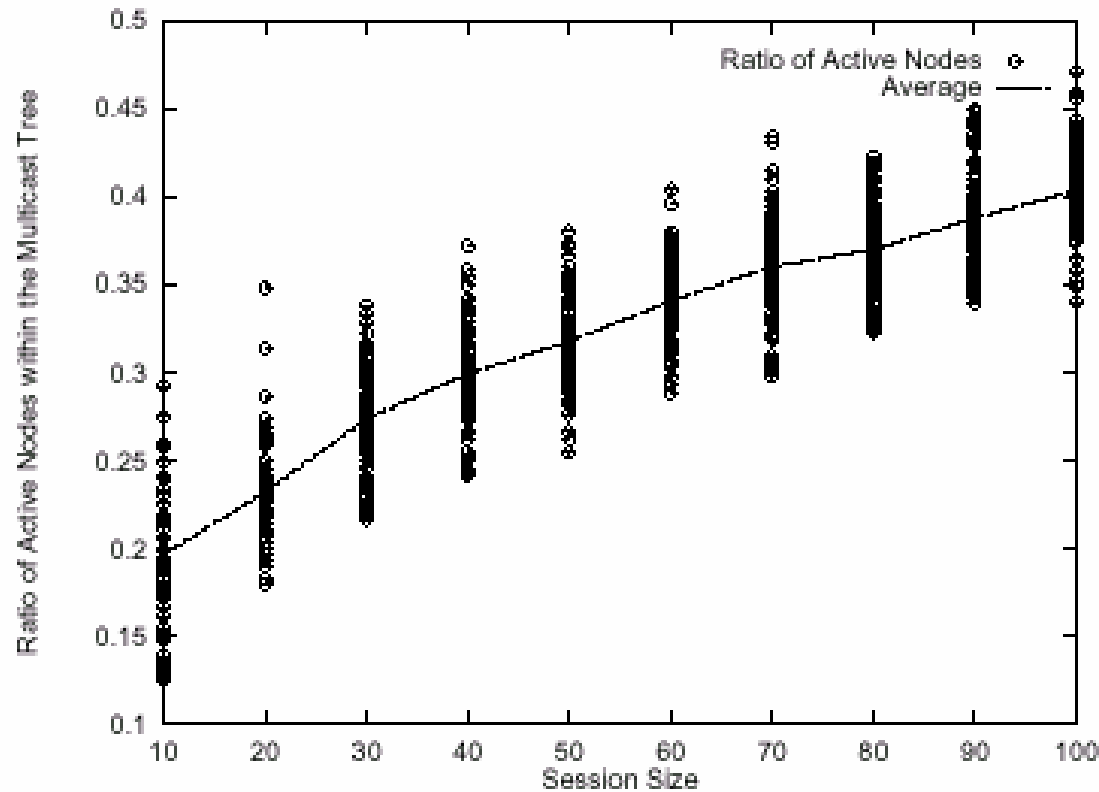


Fig. 5. Ratio of “strategic” nodes to number of non-leaf nodes in multicast tree (1000 nodes, degree 4).

NACK BW consumption with strategic active nodes

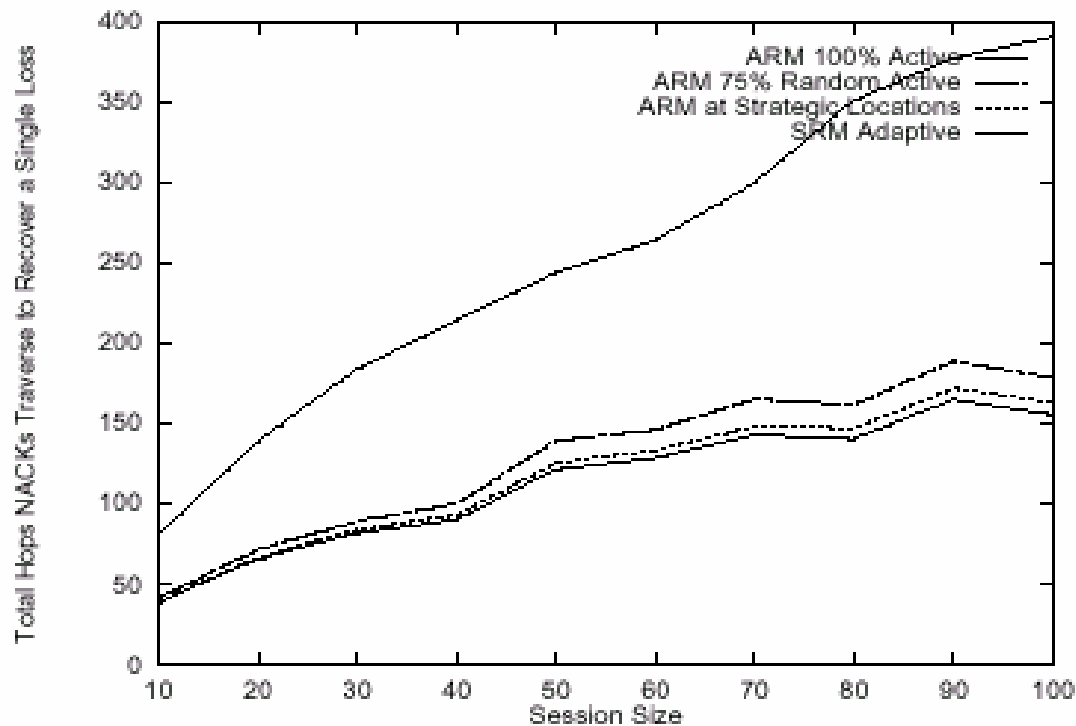


Fig. 6. NACK bandwidth consumption with strategically placed active nodes (loss near source, 1000 nodes, degree 4). ARM caches repairs, but not fresh data packets. SRM uses adaptive algorithm.

Effectiveness of ARM scoped retransmissions

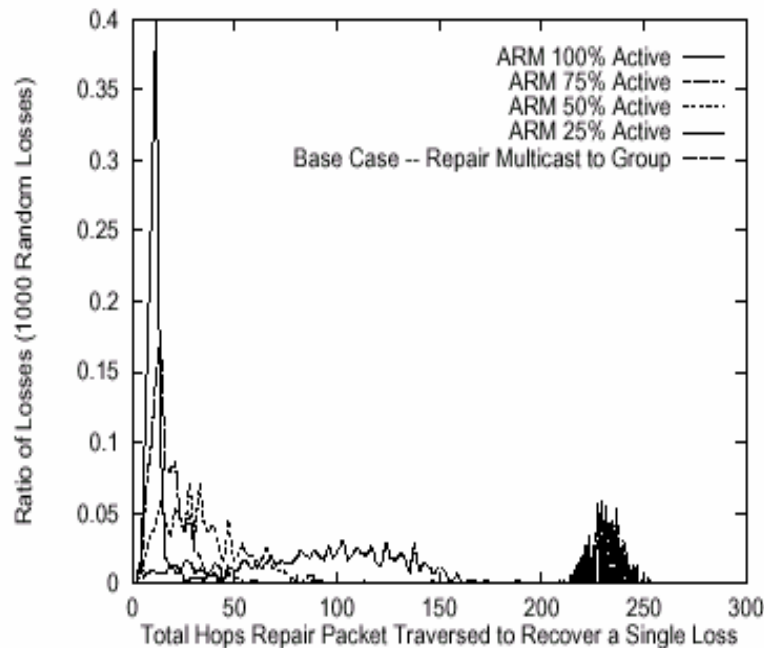


Fig. 7. Effectiveness of ARM scoped retransmissions (random loss, 1000 nodes, group size 100, degree 4). Active nodes scope retransmission and cache repair packets, but do not cache fresh data packets.

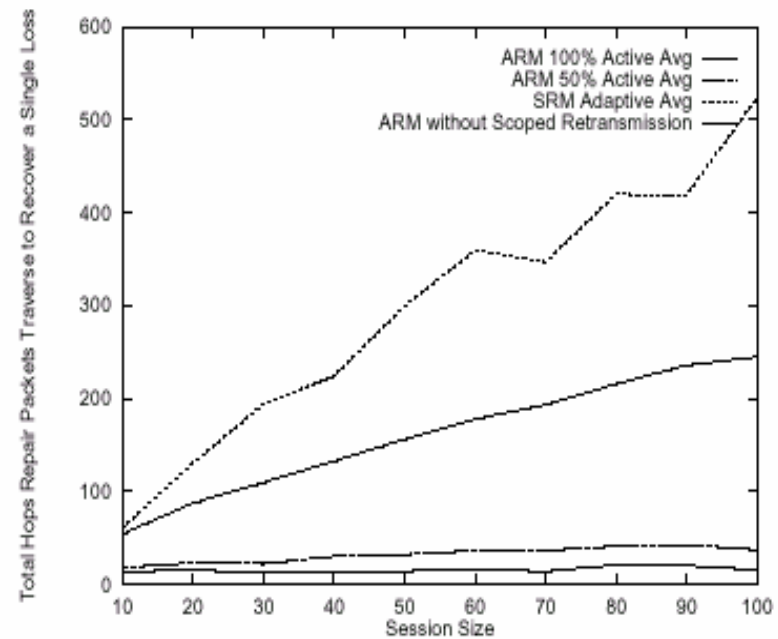


Fig. 8. Hops traversed by repair packets to recover a single loss (random loss, 1000 nodes, degree 4). ARM caches repairs, but not fresh data packets. SRM uses adaptive algorithm, 40th round.

Effect of caching repairs packets in ARM

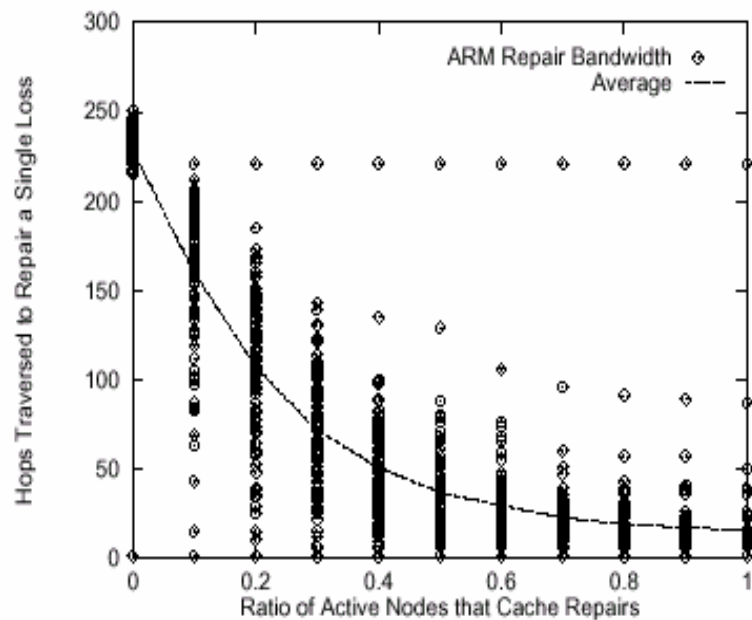


Fig. 9. ARM tradeoff for caching repairs and hops traversed by repair packets (random loss, 1000 nodes, group size 100, degree 4). All nodes active. No nodes cache fresh data packets. Nodes that cache repair packets are picked randomly. Source always caches repairs.

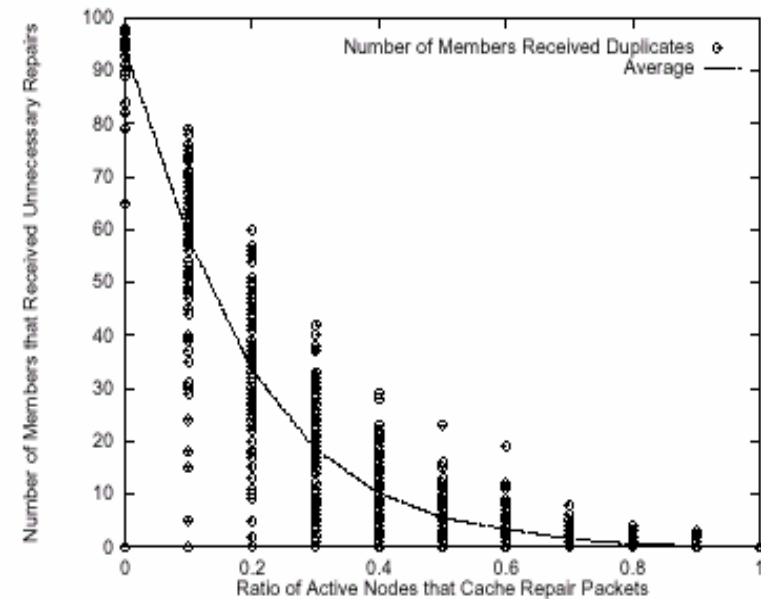


Fig. 10. Effect of caching repairs in ARM and number of receivers that receive unnecessary repairs (random loss, 1000 nodes, group size 100, degree 4). All nodes active. No nodes cache fresh data packets. Nodes that cache repair packets are picked randomly. Source always caches repairs.



Conclusions

- Network based processing and storage can enhance the performance and scalability of reliable multicast
- ARM can reduce both NACKs and repair traffic and help distribute the retransmission load by caching multicast data
- ARM is robust with respect to dynamic change in group membership
- Simulation results show that ARM enhance recovery latency, implosion control, and repair bandwidth
- Much smaller set of active nodes placed at strategic location can give same benefit



Can ARM be the really good solution for reliable multicast?

- Router processing cost – router itself can turn into bottleneck
 - In addition to multicast session, it may decrease performance of unicast traffic)
- Deployment Issue – money matters
 - Do not need to add active functionality to every router, however 50% is still too much.



Questions???

- Can we integrate some of FEC scheme into this solution?
 - IP level FEC at the router for real time multicast applications