CSE 222
Graduate Networking

Winter 2001

Lecture 15: Adaptive and Reliable Multicast

Geoffrey M. Voelker

Topic Overview

Three lectures discussing multicast routing:

1. Multicast as an IP layer service
   - Seminal paper by Deering
2. Multicast++
   - Protocol Independent Multicast (PIM)
   - Single-source multicast
3. Adaptive and Reliable Multicast
   - Tough problem, creative solutions

Overview

- Receiver-driven Layered Multicast (RLM)
  - How do you have sources adapt their rates to multiple heterogeneous link bandwidths?
- Scalable Reliable Multicast (SRM)
  - How do you recover from losses when sending to multiple receivers?
- Unicast
  - Sender does all of the hard work
- Multicast
  - Receivers do all of the hard work
  - Only way to make it scale to the Internet

RLM: The Problem

- You want to do streaming video on the Internet
- And life is looking good because you now have multicast routing and datagram delivery
  - It's ok if it's only best-effort because you don't want retransmissions in video, and your codec can tolerate some losses anyway
- But now you have to decide at what rate you should stream the video
  - Receivers are going to be listening over paths with different bottleneck bandwidths (physical, congestion)
- Whose rate should you target?

Rate Adaption

- One approach is to send at a fixed rate
  - Lowest common denominator satisfies everyone, but the worst link determines the performance of everyone
  - And a higher rate will swamp the slower links
  - What about congestion?
- Another is to adapt the sender
  - Solves the congestion problem, but not heterogeneity
- And a third is to adapt to congestion and heterogeneity
  - Sender transmits at highest rate
  - Flow degrades as it goes to more constrained links
  - Everyone receives the best quality that their link permits

Layered Compression and Transmission

- Use a layered approach to adapt
  - Compress stream into multiple layers
  - Successive layers successfully refine the stream quality
  - Sender transmits all layers
  - Layers are only forwarded down paths that can support them
- Also works if stream is transmitted at multiple rates (simulcast)
  - Not as network-efficient as layered encoding

Receiver-Driven

- Two immediate questions
  - How do you decide what the path capacity is?
    - » Physical and congestion
  - Who does the deciding, and who degrades the flow?
    - » Senders, routers, receivers?
- RLM uses a receiver-driven approach
  - Receivers determine the path bottleneck capacity
    - » Using drops, as with unicast senders
  - Receivers decide how the flow gets degraded
    - » Receivers decide which layers to listen to
  - No changes needed to the network (routers)
    - » Just need IP multicast

RLM Approach

- Senders transmit a stream in multiple layers
  - Each layer is sent to a separate multicast group addr
- Receivers listen to as many layers/groups as the path back to the sender can support
  - Level of subscription
  - Multicast tree automatically extended when needed
  - Routers implicitly configured to add/drop a layer by whether a receiver on link joins/leaves the multicast group for the layer
- Adaption
  - If receiver detects congestion, drops a layer
  - If receiver detects capacity, add a layer
  - But how do you detect capacity? How does unicast do it?

Capacity Inference

- Receivers probe link capacity by performing join-experiments
  - Join to the group for the next layer
  - If congestion, back off
  - If not, probe higher until congestion is induced
  - Repeat periodically to adapt to changes in network behavior
- Join-experiments are overhead
  - Do them infrequently when they are likely to fail
  - Frequently when likely to succeed
  - Use a join-timer with exponential backoff for each layer

Experiments and Congestion

- How do you correlate congestion with a join-experiment?
  - Need to make sure that the join caused the congestion
  - Need to know how long a join takes to setup and be noticed by the receiver (detection-time)
  - Congestion after detection-time implies experiment was successful
  - Congestion before detection-time implies experiment failed
- Backoff when experiment fails
  - Update timer according to time interval between start of experiment and the onset of congestion
  - Detection-time estimated adaptively (c.f. RTT)

Scaling RLM

- What happens if join-experiments are done by receivers independently?
  - Frequency of experiments increases with number of receivers
  - Does not scale well
- Want
  - Join-experiment rate to be independent of group size
  - Without decreasing rate at which receivers learn from experiments
- Solution: Shared learning
  - Receivers of a layer learn from experiments to join that layer
  - C.f. membership reports in multicast routing

Shared Learning

- Receivers multicast experiment notifications to the group for the layer
  - Receivers interested in layer watch for congestion
    - » One experiment serves multiple receivers
  - Receivers decide based on failed experiments only
    - » An experiment can succeed for some receivers but fail for others
  - Q: Should this really be done to the entire group?
- Problem: Experiments for different layers can overlap
  - Again the problem of correlating congestion with experiments
- Solution: Prioritize according to layers
  - Only start experiments when none in progress…
  - Or for lower layers while higher layer experiments in progress
  - Enables newer receivers to quickly adapt to level of subscription

Protocol State

- RLM requires a modicum of state
  - Join-timers for each layer, min/max intervals
  - Detection-time mean, deviation
- And magic numbers
  - Join-timer backoff and relaxation constants
  - Detection-time scaling terms, filter constants
- Indicative of adaptive algorithms
- Challenge is determining appropriate magic numbers
  - Simulation
  - Experimentation

Evaluation

How do you evaluate solutions to this kind of problem?

- Paper presents a couple of metrics and a number of simulations across various topologies
- Metrics
  1. Worst-case loss rate over varying time scales
  2. Throughput as determined by time until system converges
- Topologies
  1. Single link: Delay scalability
  2. Multiple receivers: Session size scalability
  3. Multiple sets of receivers: Heterogeneity
  4. Multiple senders and receivers: Overlayed independent sessions

Results

- Latency
  - Loss rates increase as delays increase (expected)
  - Short-term loss rates high for all delays
  - Medium- and long-term loss rates reasonable for delays < 1s
- Session
  - Loss rate independent of session size
  - Logarithmic dependence of convergence time, session size
- Heterogeneity
  - Slightly higher than homogeneous experiment
- Superposition
  - Full utilization (although slightly unfair), loss consistent with other experiments, slight dependence on session size

Discussion

- Interaction with router queueing
  - I guess the real question is how does FQ play with multicast? If a FQ router sees all each tuple of { source, group } as a flow, then simply layering the signal via different multicast groups is an easy way to grab more than your fair-share. But how would you get around this? Looking at tuples of { source, * } or { *, group} don't seem to be any better.
- Experimental evaluation
  - It also works very well to hide the fact that the loss rate percentage is increasing linearly with the number of sessions in the Superposition case. The last data point experiences double the packet drop rate of the previous data point... And then the just end the graph at 35 sessions. Why not go up to 100 like the other graphs?

Discussion (2)

- Vulnerability
  - A rogue receiver could just request level after level and cause unwanted congestion that the other receives have no power to change.
  - Another obvious problem of this work is that it requires that all the users should cooperate.
- Congestion friendly
  - Yet others are the interaction of RLM with other bandwidth-adaptive protocols like TCP
- Practical layering
  - Another question is how does one divide its source onto layers, what are the guidelines to separation (i.e T3/LAN vs Modem), because it seems that how you split source onto layers depends on the actual bandwidth not hypothetical bandwidth of the connection

Scalable Reliable Multicast

- Problem: IP multicast is best-effort
  - How do you get all the data reliably to all receivers?
  - Efficiently?
- Long paper
  - And still many open issues
  - A complex, rich problem domain…
- Note that this is not the only approach to the problem
  - XTP, RBP, RMP, MTP, Trans/Total, LBRM, RMTP, ISIS…

wb

Used to motivate SRM framework

- Group members
  - Named using globally unique persistent Source ID
  - A member can be a sender or receiver
- Whiteboard separated into pages
  - Any member can draw on or create a page
  - Named using globally unique persistent Page ID
    - » Source ID + local sequence number
- Operations
  - Members perform a sequence of drawops on pages
    - » Primarily idempotent
  - Timestamped, sequenced relative to sender
    - » Each stream independent of others

SRM Approach

- Sender multicasts data to the group
  - Data can be lost anywhere in network
- As with RLM, reliability is receiver-driven
  - Receivers responsible for detecting loss
  - Receivers responsible for requesting retransmissions
  - Any group member can respond to any rexmit with a repair
  - Repair requests and rexmits are multicast to whole group
  - One rexmit can repair many losses
- Randomization to prevent implosion
  - Receivers wait for a random time before requesting a repair
  - Wait time depends on distance from source
  - Hosts closer to failure more likely to timeout

Naming in SRM

- SRM assumes many of the naming semantics as wb
  - Data has a globally unique, persistent name
  - Data space is partitioned into “pages”
    - » Pages are also globally unique with persistent names
  - Sources have globally unique, persistent names
    - » To resume same role from previous session

Application Level Framing

- SRM uses the ALF model
  - SRM is a framework that relies upon application-level info
- Name space
  - Data is named using application data units (ADUs)
  - Easy for application to handle data out-of-order
  - ADUs are global names, e.g., anyone can respond to request
- Channel management, action priorities
  - Same channel used simultaneously for transmitting new data, session messages, recovering from losses (from all sources)
  - Application can schedule these actions

Session Messages

- Receivers multicast periodic session messages
  - Report active sequence number state for all active sources
  - Detect loss of the last packet in a sequence
  - Sender monitor status of receivers
  - Members determine participants in session
- Scalability
  - Need to partition name space according to application
  - Members only report state of page being “viewed”
- Timing
  - Used to estimate one-way distances between nodes
  - Distances used by timers in the repair algorithm

Loss Recovery

- Loss recovery provides the reliable delivery
- Upon detecting a loss, receivers
  - Wait a random time and then multicast a repair request
  - These requests suppress other receivers with losses
    - » They backoff exponentially
- Upon receiving a repair request, members
  - Wait a random time and then multicast a repair response
  - These responses suppress other members that can repair
- Timeout values for request and repair timers
  - Are a function of the member’s estimated distance to source
  - Hosts closer to failure more likely to timeout

Deterministic Suppression

- SRM relies upon two forms of request suppression to prevent control message implosion
- Deterministic suppression relies upon member distances from sources (chain topology)
  - When members are all at different distances from the source, members closest to the failure send request and repair
  - These deterministically suppress all other requests and repairs (in simplified conditions)
  - Losses far away can be repaired faster than in unicast (!)
    - » Because repairs are initiated close to failure with fast timeouts

Probabilistic Suppression

- Probabilistic suppression relies upon randomization when members are all at the same distance (star topo)
  - All members detect loss at same time, randomly set timers
  - Number of requests and repairs depends on timer constants
  - Increasing the timing range
    - » Reduces expected number of requests
    - » But increases expected delay until the first request is sent
- Tree topologies rely upon both deterministic and probabilistic suppression to prevent implosion
  - Deterministic suppresses requests/repairs further down tree
  - Probabilistic suppresses requests/repairs at same depth

Simulation Overview

- There are many variables and parameters
  - Network topology
  - Set of session members, fraction of sources
  - Request and repair timer parameters
- Some high-level results
  - Fixed timer params work well in dense sessions
    - » Less well for sparse sessions
  - Chains
    - » C1 needs to be relatively high so that members down the chain do not timeout early simply due to propagation delay
  - Stars
    - » Small C2 values work well for dense sessions (delay and dups)

Adaptive Timer Algorithms

- Because performance is sensitive to topology, session memberships, and loss patterns, want timer algorithms to be adaptive
  - Function of past history of algorithms
  - Adaptive algorithms lower repairs (4), delay (2*RTT)

Local Recovery

- SRM's recovery algorithm is global – a key problem
  - Requests and repairs are multicast to entire group
  - Does not scale well (shown in later papers by McCanne)
- Idea: Constrain recovery to loss neighborhood
  - Set of members who are experiencing same set of losses
  - Local loss: # members experiencing loss << # in session
- Approaches
  - Administrative scope (within same admin domain)
  - Separate multicast groups
  - TTL score
- Aside: Applicable to RLM?

Discussion

- Scalability
  - I have doubts about the scalability of the framework as a whole. For instance, can the request/repair really scale?
- Processing overhead
  - Canceling a request because another host has the data, calculating the shortest distance between each host, modifying timers to "de-synchronize" host actions, and so on make SRM more complicated than the minimal framework it was intended to be.
- Out-of-order delivery
  - Unless wb is saving its state for all of its previous drawing operations, then it's going to be highly unhappy when it gets packets out of order, because it will have to redo all of the draw operations after that point.
  - Coincidentally, wb does save all state…

For Next Time...

- Wireless papers
  - TCP performance over wireless links
  - Routing in ad-hoc networks