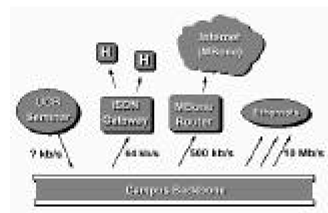
Receiver-driven Layered Multicast

S. McCanne, V. Jacobsen and M. Vetterli
University of Calif, Berkeley and
Lawrence Berkeley National Laboratory

SIGCOMM Conference, 1996

The Problem

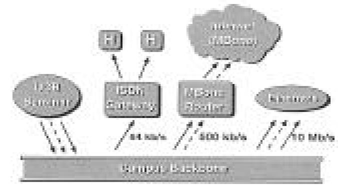
- Want to send to many recipients
  - → Multicast
- One bandwidth for all is sub-optimal
  - Min? Max?



Approaches

- Adjust sender rate to network capacity
  - Not well-defined for multicast network
  - Does not scale well if receiver gets feedback
- Layer server output so receiver can have gracefully degraded quality

The Layered Approach



- Router will drop packets upon congestion
- Receiver receives only requested channels
- No explicit signal to sender needed
- This work's contribution
  - Explicit exploration of second approach
  - *Receiver-driven Layered Multicast* (RLM)

Outline

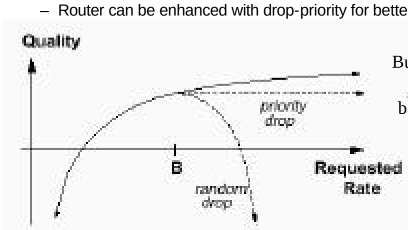
- Introduction
- ***RLM***
- Evaluation
- Conclusion

Network Model for RLM

- Works with IP Multicast
- Assume
  - Best effort (packets may be out of order, lost or arbitrarily delayed)
  - Multicast (traffic flows only along links with downstream recipients)
  - Group oriented communication (senders do not know of receivers and receivers can come and go)
- Receivers may specify different senders
  - Known as a *session*

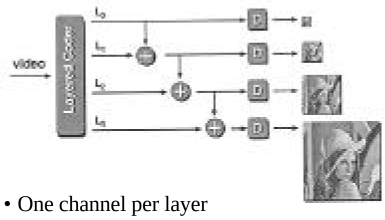
RLM Sessions

- Each session composed of layers, with one layer per group
- Layers can be separate (ie- each layer is higher quality) or additive (add all to get maximum quality)
  - Additive is more efficient
  - Router can be enhanced with drop-priority for better quality



But rewards high bandwidth users!

Layered Video Stream



- One channel per layer
- Layers are additive
- Adding more channels gives better quality
- Adding more channels requires more bandwidth

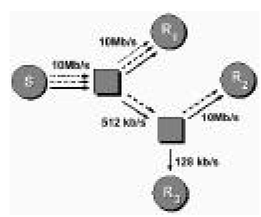
Groupwork

- Consider MPEG video
- Consider voice-quality audio
- Devise layering scheme
  - As many layers as you want
- Explain

The RLM Protocol

- Abstraction
  - on congestion, drop a layer
  - on spare capacity, add a layer

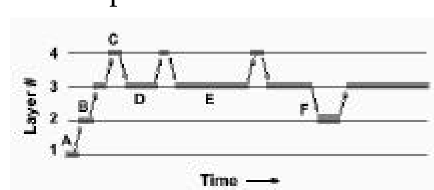
→ Similar to bandwidth probing in TCP



Adding and Dropping Layers

- Drop layer when packet loss
- Add does not have counter-part signal
- Need to try adding at well-chosen times
  - Called *join experiment*
- If join experiment fails
  - Drop layer, since causing congestion
- If join experiment succeeds
  - One step closer to operating level
- But join experiments can cause congestion
  - Only want to try when might succeed

Join Experiments



- Short timers when layer not problematic
- Increase timer length exponentially when above layer has congestion
- How to know join experiment has succeeded?
  - *Detection time*

Detection Time

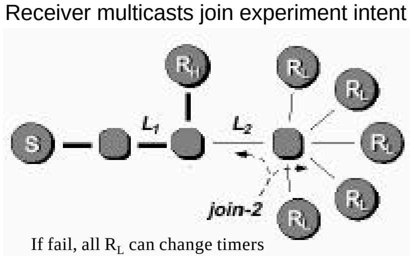
- Hard to measure exactly
  - (How to estimate?)
- Start conservatively (ie – large)
- Increase as needed with failed joins
  - When congestion detected after join, updated detection time to start of join experiment to detection

Scaling RLM

- As number of receivers increase, cost of join experiments increases
  - does not scale well
- Join experiments of others can interfere
  - Example, R1 tries join at 2 while R2 tries join at 4
    - Both might decide experiment fails
- Partial solution: reduce frequency of join experiments with group size
  - But can take too long to converge to operating level
- Solution
  - Shared learning

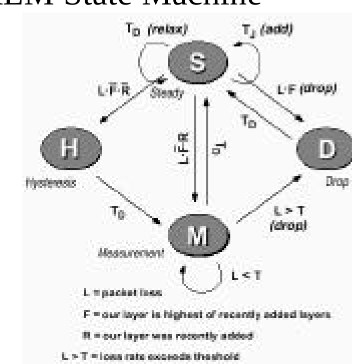
Shared Learning

- Receiver multicasts join experiment intent



If fail, all R_L can change timers
Upper layer join will repress join experiment
Same or lower layer can all try
(Note priority drop will interfere … why?)

RLM State Machine



T_d – drop timer
T_j – join timer

Outline

- Introduction
- RLM
- ***Evaluation***
- Conclusion

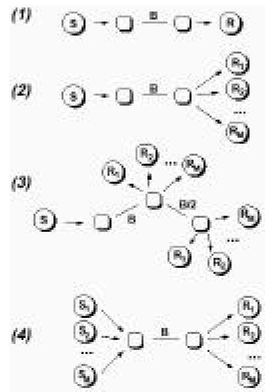
Evaluation

- Simulate in NS
  - Want to evaluate scalability
- Model video as CBR source at each layer
  - Have extra variance for some ‘think’ time, less than 1 frame delay
  - (But video often bursty! Future work)

Parameters

- Bandwidth: 1.5 Mbps
- Layers: 6, each 32 x 2^m kbps (m = 0 … 5)
- Start time: random (30-120) seconds
- Queue management :DropTail
- Queue Size (20 packets)
- Packet size (1 Kbyte)
- Latency (varies)
- Topology (next slide)

opologies



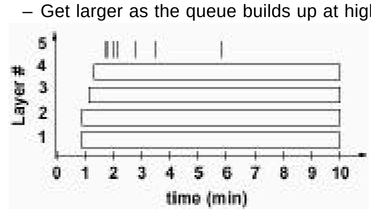
1 – explore latency
2 – explore scalbility
3 – heterogenous with two sets
4 – large number of independent sessions

Performance Metrics

- Worse-case lost rate over varying time intervals
  - Short-term: how bad transient congestion is
  - Long-term: how often congestion occurs
- Throughput as percent of available
  - But will always be 100% eventually
    - No random, bursty background traffic
  - So, look at time to reach optimal
- Note, neither alone is ok
  - Could have low loss, low throughput
  - High loss, high throughput
  - → Need to look at both

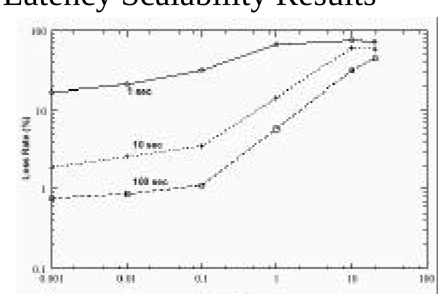
Latency Scalability Results

- Topology 1, delay 10 ms
- Converge to optimal in about 30 seconds
- Join experiments less than 1 second
  - Get larger as the queue builds up at higher levels



Next, vary delay 1ms to 20s and compute loss

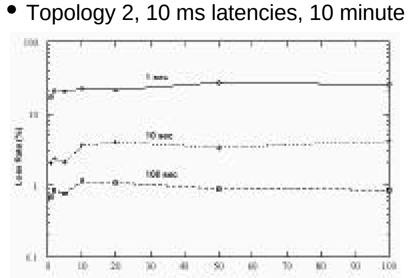
Latency Scalability Results



Window size averaged over 1, 10 and 100 secs

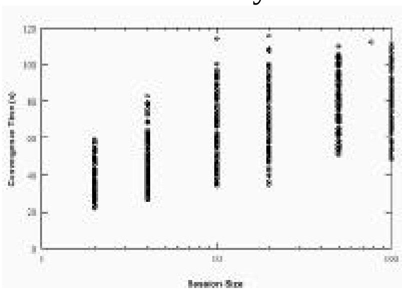
Session Scalability Results: Loss

- Topology 2, 10 ms latencies, 10 minute run



Independent of session size
Long term around 1%

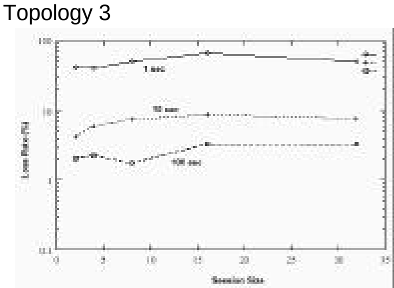
Session Scalability Results: Loss



Linear trend suggests logarithmic convergence
(sharing is helping more)

Bandwidth Heterogeneity Results

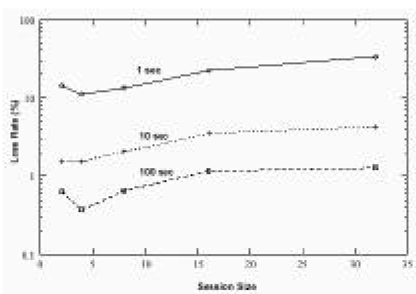
- Topology 3



Bit higher than homogenous
Small session matters more because of collisions

Many Sessions Results

- Topology 4, bottleneck bwidth and queue scaled



And converged to 1, but very unfair early on

Network Dependencies

- Requires receiver cooperation
  - If receiver application crashes, host still subscribed
- Group maintenance critical
  - Router must handle join and leaves quickly
- Network allocation may be unfair
  - Should be 'good' level for all that share link
  - TCP has same problem
- AQM (RED +) may help
  - decrease time to detect failed session experiment

The Application

- Build compression format knowing network constraints
  - Not vice-versa
- Have a real working application
  - Integrated in *vic*
- RLM component is not in 'fast-path' since changes slower
  - Done in TCL

"Future" Work

- Compression scheme that can more finely compress layers
  - Adapt compression to receivers
  - For example, if all high and one low then can compress in two levels
- RLM with other traffic (TCP)
- RLM combination with SRM

Summary

- Multicast
- Receiver-based performance
- Layered video

- All been done before, but first complete system with performance

Conclusions

Evaluation of Science?

- Category of Paper
- Science Evaluation (1-10)?
- Space devoted to Experiments?