Timer Reconsideration for RTCP

Henning Schulzrinne
Columbia University, New York
schulzrinne@cs.columbia.edu

Jonathan Rosenberg
Bell Laboratories and Columbia University
jdrosen@bell-labs.com

Overview

- Problem Definition
- Algorithm Descriptions
- Step Join Performance
- Steady State Behavior
- Fairness
- Synchronization
- BYE Storms
- Polling

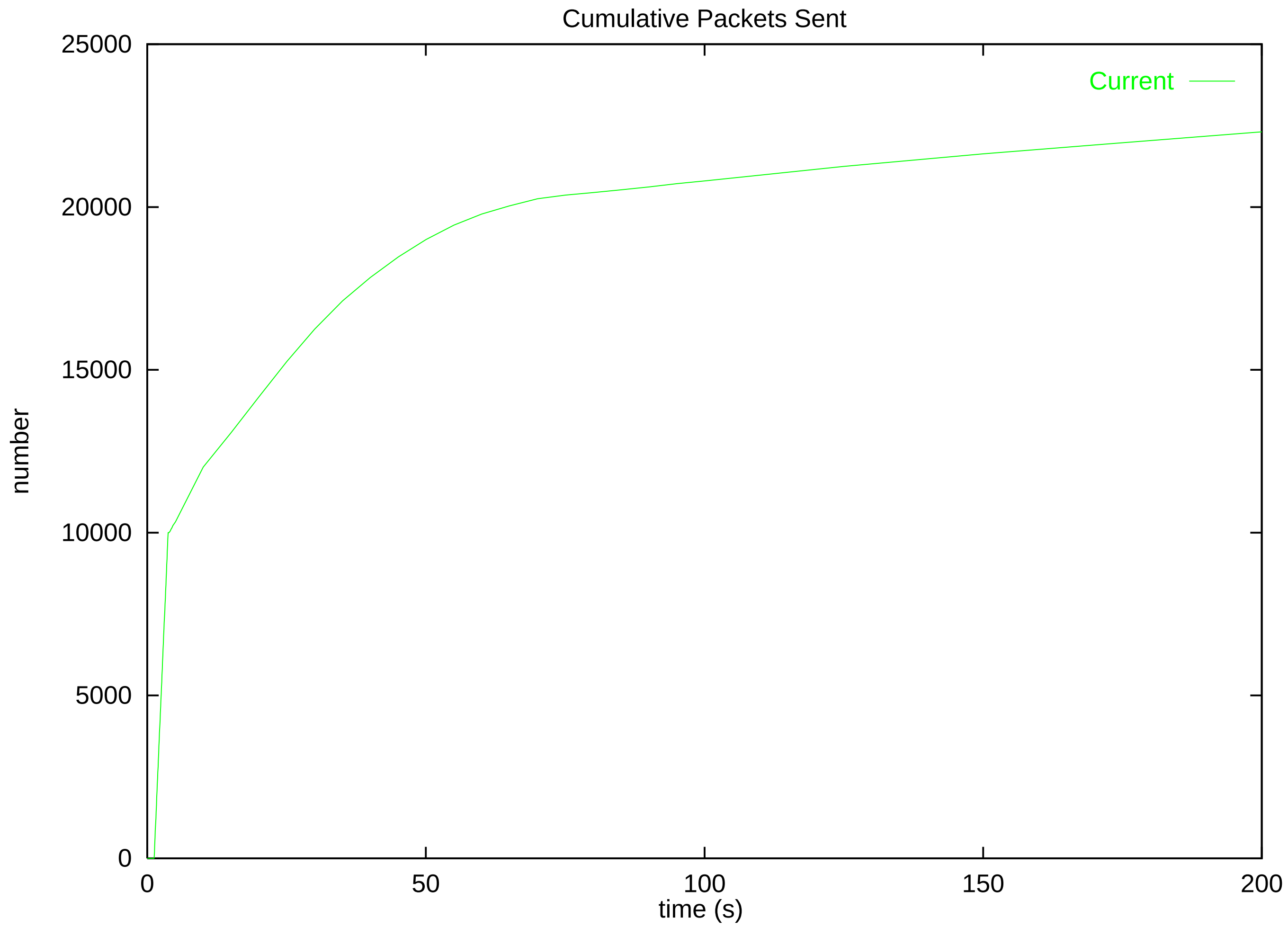
Problem Definition

Step Joins

- Simultaneous Join of many users
  - Cable Networks ➠ “Friends at 8pm”
  - Automatic joins in response to sdr announcements
- All users start with minimum interval ($T_{min} = 2.5s$ for first packet)
- Randomization is added, all packets scheduled between $1/2, 3/2$ of T_{min} ➠ 1.25..3.75s
- Timers expire, all users send no matter what has happened since timer set at $t = 0$
  - Congestion (exacerbated by slow access links)
  - Losses ➠ don’t learn about these users ➠ group size estimates are still low ➠ more congestion

Simulation Example

- Network delay uniform between 300 and 600 ms
- $10,000$ users all join at $t = 0$
- Network provides full, instant connectivity
- All users connected via dialup lines, 28.8 downstream, ∞ upstream
- Access link buffers 50 kBytes
- RTCP packets 1 kB
- All receivers, no senders

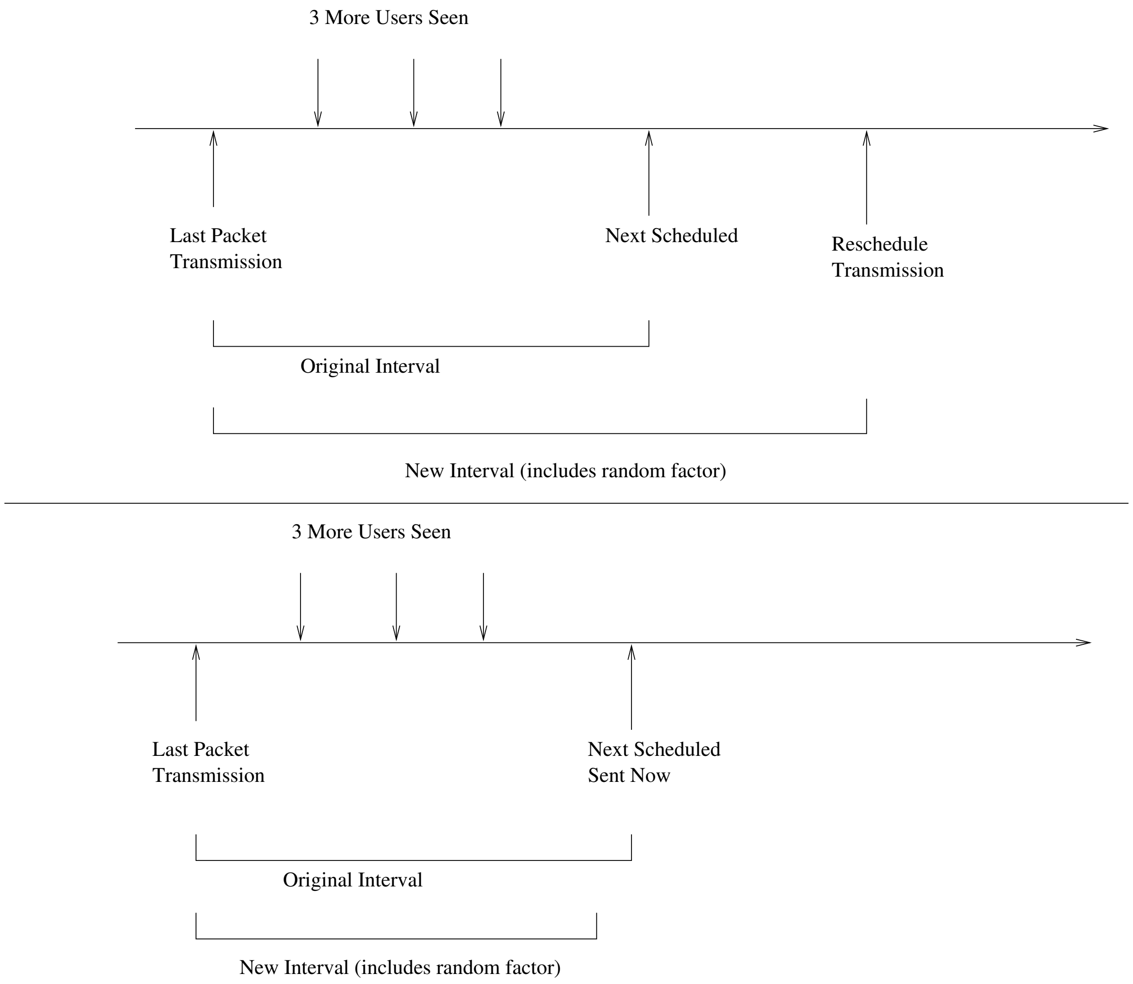

Cumulative Packets Sent
Current
number
25000
20000
15000
10000
5000
0
0
50
100
150
200
time (s)

Solutions

Timer Reconsideration

- Basic idea ➠ Before sending a packet, check if group size changed
- If changed, recompute interval and randomization factor
- Add new interval to last send time
- If result is less than current time, send
- Else, reschedule for time just computed
- Intuition ➠ for step joins, rapid arrival of first few packets pushes off transmission of remaining packets into future

Reconsideration Algorithm

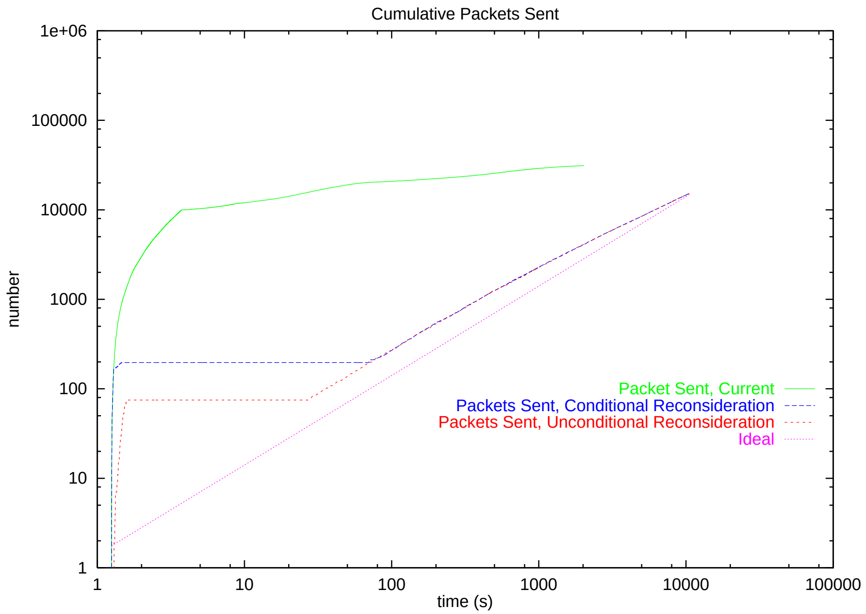


Reconsideration Flavors

- conditional
  - Exactly as described above - recomputing timer is conditional on a change in group size
- unconditional
  - Always recompute interval (including random factor) even if group size has not changed
  - Add new interval to previous packet transmission time
  - If result is before current time, send, else reschedule for future time
  - Observation ➠ do not need to wait for packet arrivals to change group size estimate; this is needed for conditional
  - Observation ➠ When converged, causes an increase in average interval by 21%

- hybrid
  - Start off with unconditional
  - After receiving a packet from another user, switch to conditional
  - Result ➠ Same initial performance as unconditional, and same steady state performance as current RTCP algorithm
  - More complicated

Performance

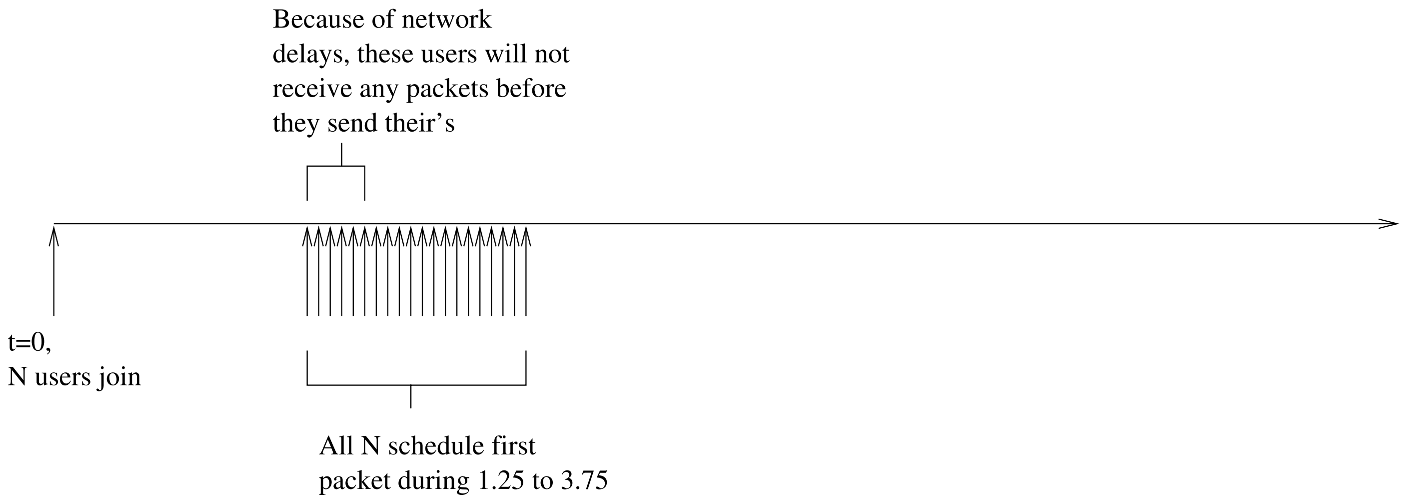


- 10,000 step join, same network model as previous
- Current algorithm ➠ 10,000 packets in 2.5 s
- Conditional ➠ 197 packets in 210 ms
- Unconditional ➠ 75 packets over 327 ms
- Conditional and unconditional stop sending after initial spike

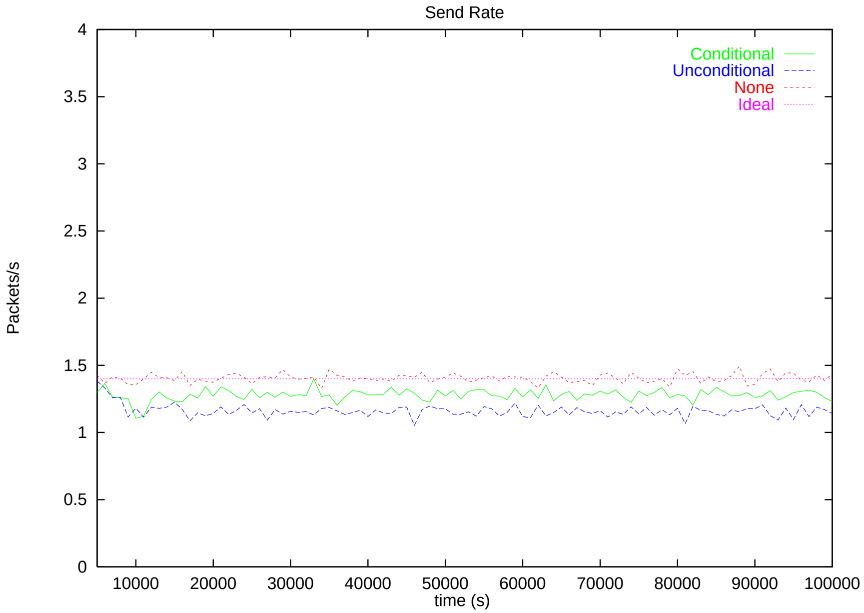
- Why ➠ left hand side of reconsideration window exceeds current time ➠ all packets rescheduled for future
- Gives time for network buffers to clear
- Conditional and unconditional send at 10% session bandwidth after initial spike, gradual reduction to 5%
- Note: In all cases, initial spike still depends on size of step join
  - unconditional:
$$P \approx \frac{ND^2}{8\alpha^2 T_{min}^2} \tag{1}$$
  - conditional
$$P \approx \frac{ND}{2\alpha T_{min}} \tag{2}$$
  - current
$$P = N \tag{3}$$

  $\alpha = .5$, $D =$ net delay, $N =$ no. users

Unconditional Performance

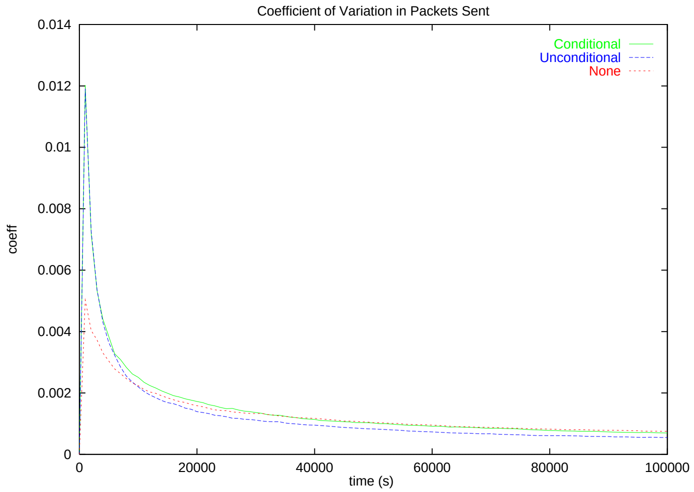


Steady State Behavior



- Average of 10, 000 members, slight oscillations in membership
- Ideal packet rate = 1.4 per second ➠ current algorithm
- Unconditional packet rate = 1.149 packets per second
- Conditional packet rate is halfway between ideal and unconditional

Fairness and Synchronization



- Coefficient of variation (standard dev. over mean) of packets sent over time
- All three are extremely fair
- Synchronization ➠ unconditional causes small reduction in variance of interval, but still more than enough

BYE Storms

- Can have simultaneous leaves ➠ flood of BYE messages
- cannot use reconsideration; application terminates immediately
- Solutions
  1. Don't send BYE if you never sent any other RTCP
  2. Don't send BYE if group size is larger than threshold
  3. Don't send BYE if you've seen more than some number of BYE's in the last T seconds (say, link bandwidth)
- Pros and Cons
  - No. 1 is only beneficial, BYE is useless if no other packet sent
  - No. 2 works OK, no control on rate of BYE's
  - No. 3 works pretty well; can control rate of BYE's

Polling

- RTCP multicasting not appropriate in all cases
- May want central poller
- Several unresolved issues
  - Authentication and Security (denial of service attacks)
  - New RTCP packet format
  - Precise polling algorithms
- Should have text for RFC 1890 appendix next meeting

Proposal

- Add: unconditional algorithm to RFC 1889
- Add: “no RTCP, no BYE” to RFC 1889
- Add: don’t send BYE if more than a threshold of BYE’s have already been seen over last T seconds