

# ***Loop-free routing using a dense label set in wireless networks***

Marc Mosko and J.J. Garcia-Luna-Aceves  
U.C. Santa Cruz  
Jack Baskin School of Engineering  
<http://www.cse.ucsc.edu/research/ccrg>

ICDCS 2004, Tokyo Japan, Mar 24 - 26

# Outline

---

- Loop-free routing in ad hoc networks
- Previous protocols (very short review)
- Split Label Routing (SLR)
- Split-label Routing Protocol (SRP)
- Performance
- Conclusion

# Loop-free routing in ad hoc nets

---

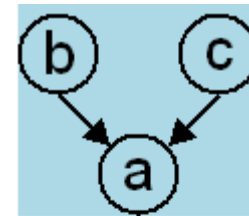
## Loop-free on-demand routing

- Ad hoc on-demand routing
  - Mobile nodes with radio links
  - Only maintain information for “in-use” routes
- Loop free at all times, no temporary loops
- Want to keep overhead low, even with high mobility
- Main problem is how to **re-label** graph due to errors or mobility.
- For distance vector, problem boils down to maintaining a topological order of node labels.

# Terminology

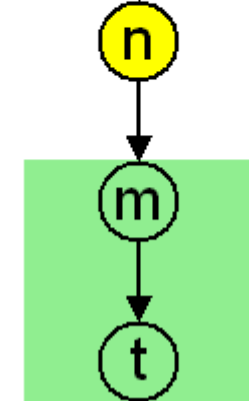
- **Predecessors**

- For some destination,  $t$ , those nodes that use  $n$  to reach  $t$ .
- Example: nodes  $a, b, c$ .



- **Successors**

- For some destination,  $t$ , those nodes between  $n$  and  $t$ .
- Example: node  $m$ .



# Previous loop-free protocols

---

- DSR
  - Source routing.
    - Data packet source specifies complete route.
  - When there is an error
    - Node must drop the packet and send route error.
    - DSR does not “re-label”, must start new route discovery at the source (route request).
  - Not loop-free when salvaging routes.
  - GloMoSim and Qualnet simulations show poor DSR performance with mobility.

# Previous loop-free protocols (2)

---

- GB/LMR/TORA
  - Use “arbitrary” (loosely related to distance) node labels to maintain topological order.
  - Iterative graph algorithms.
  - Re-labeling is global.
    - GB/LMR: can be unstable.
    - TORA: Node that loses last downstream link sets new “reference level” to be global maximum.

# Previous loop-free protocols (3)

---

- AODV
  - Distance vector.
  - Node labels = (seqno, distance).
  - Loop-free at all times.
  - Potential for localized re-labels based on “local recovery RREQ”.
  - Loop-free feasibility test overly restrictive
    - To re-label, node detecting error increases sequence number.
    - Increased sequence number invalidates all predecessors as potential successors. Prevents loops.
    - However, may also invalidate many potential loop-free successors.

## Previous loop-free protocols (4)

---

- Label Distance Routing (LDR)\*
  - Similar to AODV, but....
    - Same node label = (seqno, distance).
    - Node with error does not increase sequence number.
    - Allows any node that satisfies feasible distance criteria [DUAL] to reply to route request.
    - Improved localized re-labels compared to AODV.
    - Uses sequence number to re-label *successor path* if RREQ travels over out-of-order nodes.

\* JJ. Garcia-Luna-Aceves, C. Perkins, M. Mosko, PODC 2003

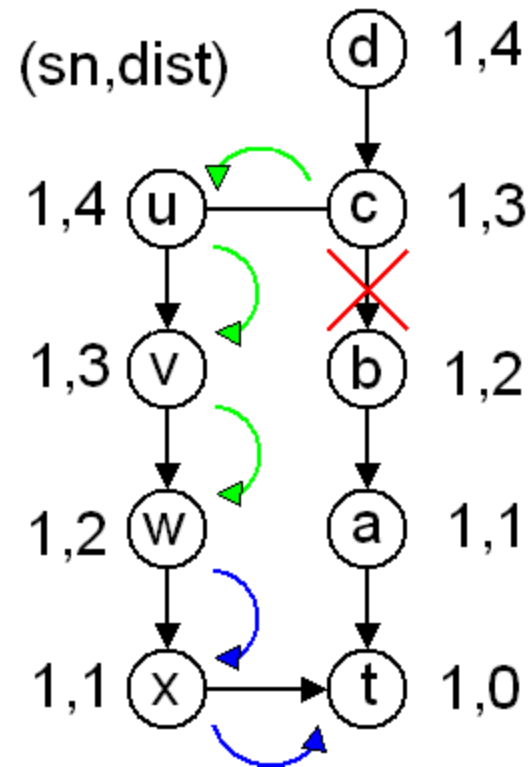
# Re-labeling in LDR

---

- LDR uses integer node labels
  - If RREQ travels over out-of-order node, relay must set “reset-required” bit to get higher sequence number.
  - May cause RREQ to travel further than needed solely because of label set, not loop-freedom.

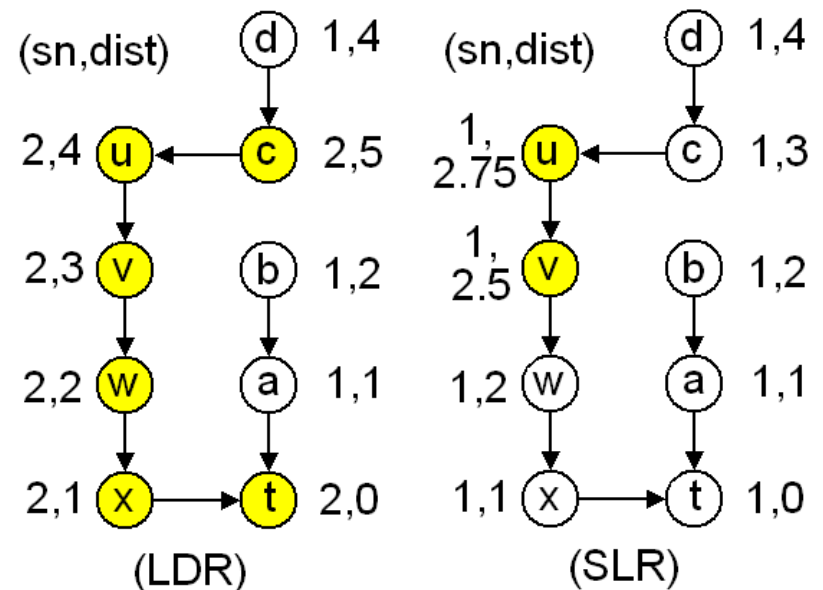
# LDR Example

- Link  $c-b$  breaks
- RREQ must travel to  $w$  for loop-free path based on existing labels.
- Integer labels cannot put  $u$  and  $v$  in-order between 2 and 3.
- LDR must get seq. number reset from  $t$ .
  - Node  $w$  unicasts RREQ to node  $t$ .



# Idea behind SLR

- LDR resets whole path (t,x,w,v,u,c) with higher seq. number.
- SLR “splices” in new labels for (u,v). Does minimum re-labeling. Observations:
  - Only needs seq. number if overflow available “splits”.
  - **A node may independently decrease label as long as greater than maximum successor.**



# Split Label Routing (SLR)

---

- Use a dense label set instead of integers
  - SLR is general idea of using dense label set in on-demand ad hoc routing protocol.
  - Use arbitrary node labels, not distance labels.
    - Still use distance as metric to choose between loop-free paths.
  - Let the label set  $\mathcal{L}$  have a strict partial order  $<$ .
  - Dense:  $\forall x < y \in \mathcal{L}, \exists z \in \mathcal{L}$  s.t.  $x < z < y$ .
  - Examples
    - Lexicographic strings
    - Subsets of the real numbers
      - Proper fractions:  $m/n$  where  $m < n$  are integers.

# Independent re-labeling

---

- Independent
  - Any feasible (loop-free) subset of connected nodes with existing labels may re-label themselves within a RREQ/RREP computation.
- Definitions, per destination:
  - Let  $S_{\max}^i$  be the maximum successor label in-use at  $i$ .
  - Let node  $i$  have label  $L_i$
  - Based on the reverse-route of a RREQ, let  $M_i$  be the minimum reverse-route label. This is a strict upper bound for node  $i$ 's new label.
  - Let  $L_{rrep}$  be the successor label in a RREP. This is a strict lower bound for new label.

# Maintaining Order

---

Based on a RREP for a RREQ, node  $i$  may independently choose new label  $G_i$  that satisfies four conditions:

$$G_i \leq L_i$$

New label maintains existing predecessor ordering.

$$G_i < M_i$$

New label is feasible along RREQ reverse path.

$$L_{rrep} < G_i$$

New label is in-order with regard to successor sending RREP.

$$S_{\max}^i < G_i$$

New label is in-order with regard to existing successors (may need to drop some existing successors).

# Existence and satisfiability

---

- We show that a simultaneous solution exists to 1<sup>st</sup> 3 inequalities.
  - Proofs depend on label set being dense.
- May need to drop some or all existing successors to satisfy 4<sup>th</sup> inequality.
- Algorithm to choose  $G$  depends on label set.

$$G_i \leq L_i$$

$$G_i < M_i$$

$$L_{rrep} < G_i$$

$$S_{\max}^i < G_i$$

# Split-label Routing Protocol (SRP)

---

- An implementation of SLR w/ specific label set.
- We use proper fractions
  - Mediant:  $p/q < (p+r)/(q+s) < r/s$
  - Add ideals:
    - 0/1 (least element)
    - 1/1 (greatest element)
  - Does not use any floating point
  - Constant size

# SRP Label Set

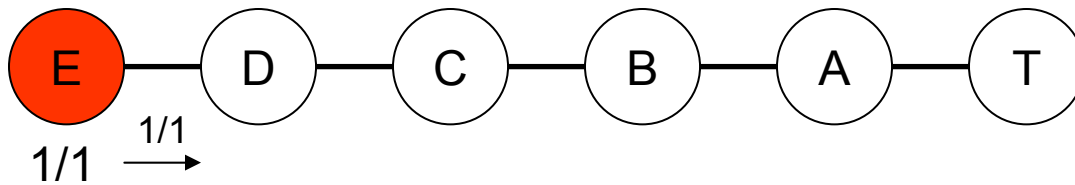
---

- Sequence number and proper fraction
  - If fraction does not overflow (e.g. 16-bit or 32-bit numerator & denominator), seq. number not used.
  - $\mathcal{O}_a = (SN_a, F_a)$ , where  $F_a = m_a/n_a$
  - Strict partial order:
    - $F_a < F_b$  iff  $m_a n_b < m_b n_a$
    - $\mathcal{O}_a < \mathcal{O}_b$  iff
      - $SN_a > SN_b$  or
      - $(SN_a = SN_b) \wedge (F_a < F_b)$

# Labeling Example

---

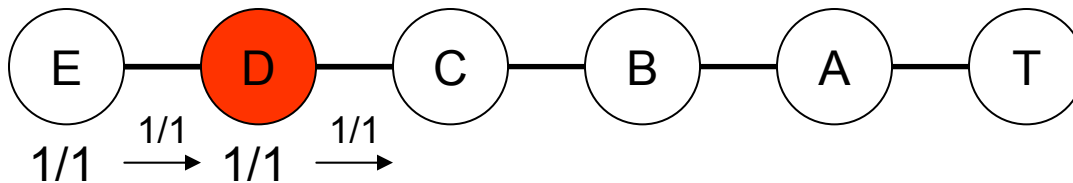
- *E* issues RREQ. Has no label, use uses maximum 1/1 label.



# Labeling Example

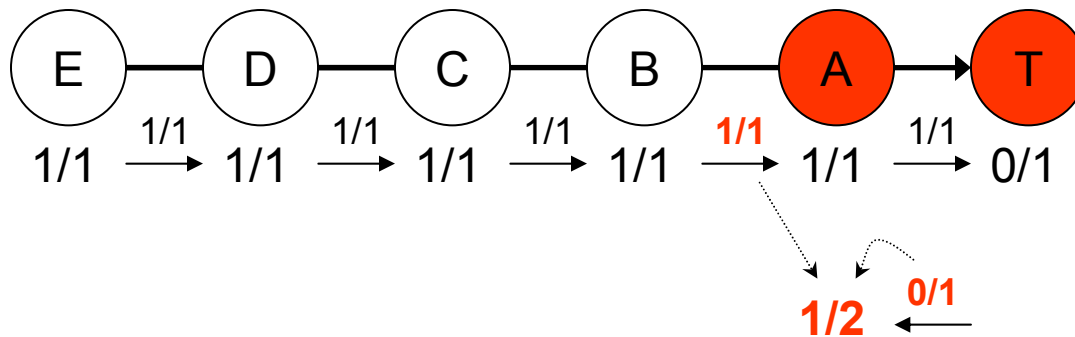
---

- *E* issues RREQ. Has no label, use uses maximum 1/1 label.
- Relay nodes are unlabeled and keep the “minimum” label 1/1.



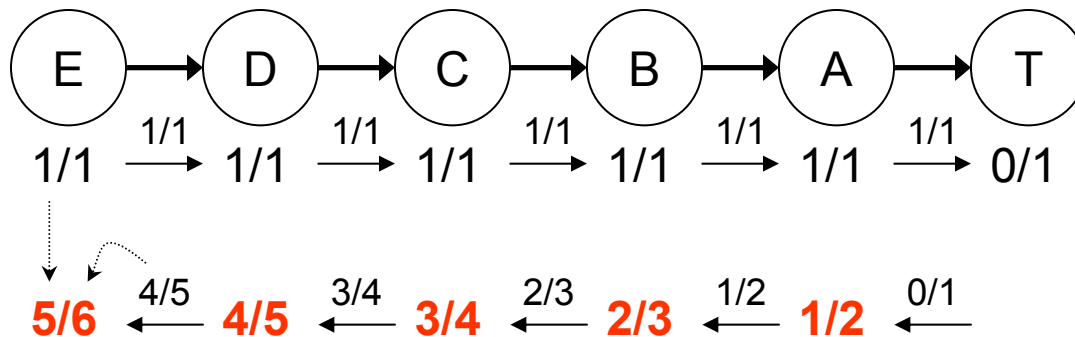
# Labeling Example

- *E* issues RREQ. Has no label, use uses maximum 1/1 label.
- Relay nodes are unlabeled and keep the “minimum” label 1/1.
- **Node T responds with 0/1 and A takes mediant of 0/1 and predecessors' 1/1 = 1/2 label.**



# Labeling Example

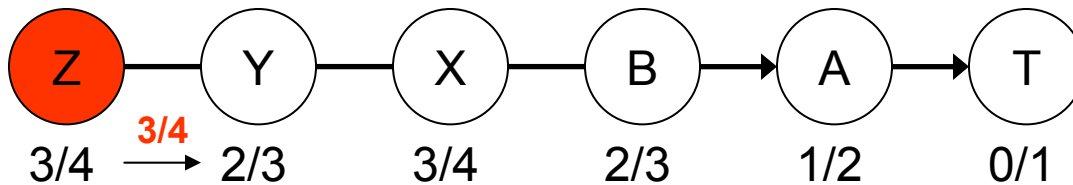
- *E* issues RREQ. Has no label, use uses maximum 1/1 label.
- Relay nodes are unlabeled and keep the “minimum” label 1/1.
- Node *T* responds with 0/1 and node *A* takes mediant of 0/1 and predecessors’ 1/1 = 1/2 label.
- **Final labeling: 5/6 → 4/5 → 3/4 → 2/3 → 1/2 → 0/1**



# Re-labeling Example

---

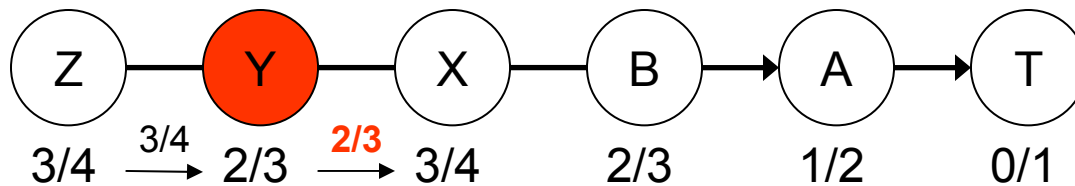
- Z issues RREQ for T with its current  $3/4$  label.



# Re-labeling Example

---

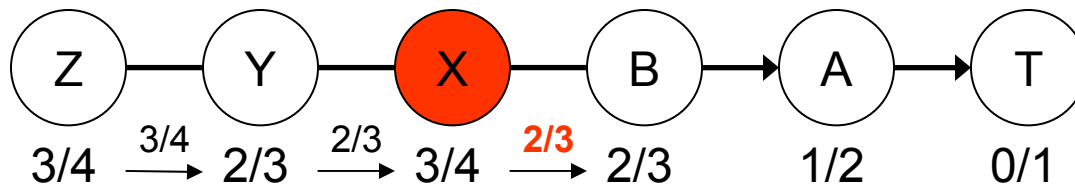
- Z issues RREQ for T with its current  $3/4$  label.
- **Y relays request with new minimum  $2/3$ .**



# Re-labeling Example

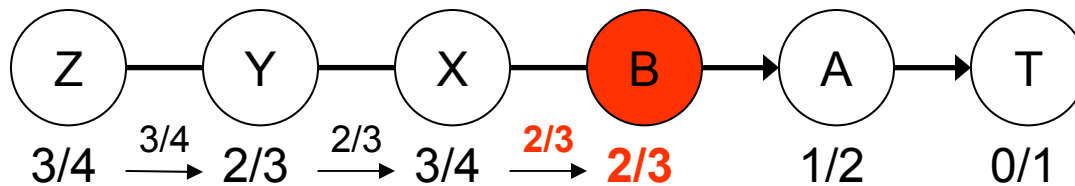
---

- Z issues RREQ for T with its current  $3/4$  label.
- Y relays request with new minimum  $2/3$ .
- **When X relays, leaves min at  $2/3$ .**



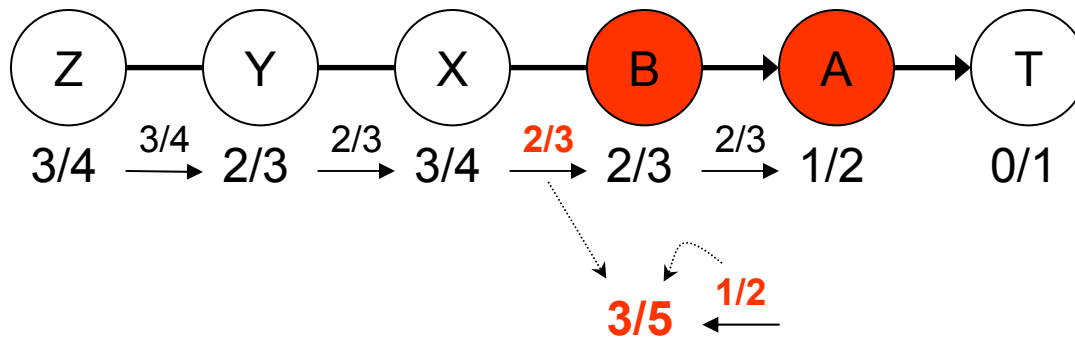
# Re-labeling Example

- Z issues RREQ for T with its current  $3/4$  label.
- Y relays request with new minimum  $2/3$ .
- When X relays, leaves min at  $2/3$ .
- **B cannot reply because its  $2/3$  label not less than requested minimum  $2/3$ .**



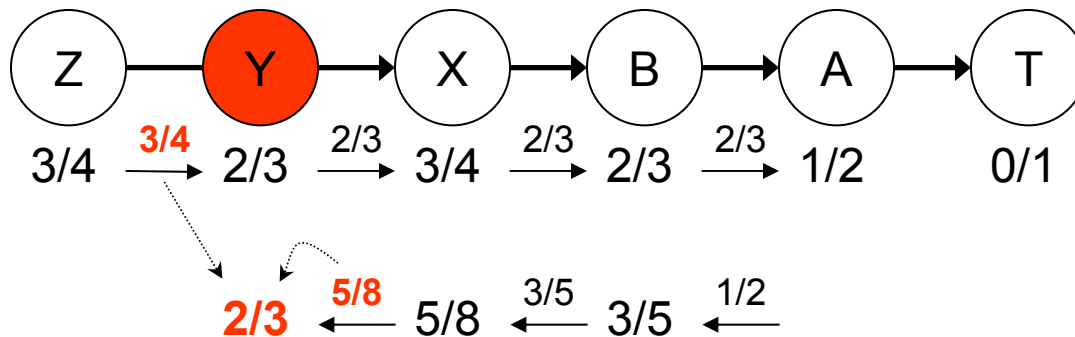
# Re-labeling Example

- Z issues RREQ for T with its current  $3/4$  label.
- Y relays request with new minimum  $2/3$ .
- When X relays, leaves min at  $2/3$ .
- B cannot reply because its  $2/3$  label not less than requested minimum  $2/3$ .
- **A replies and B takes new label from mediant.**



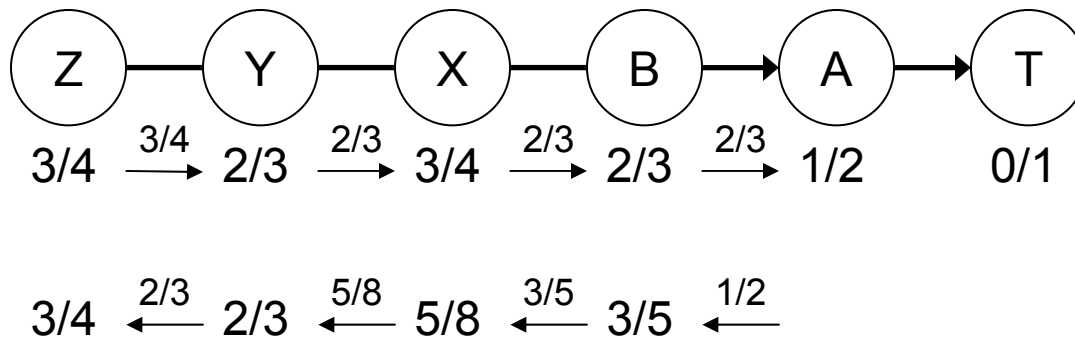
# Re-labeling Example

- Z issues RREQ for T with its current  $3/4$  label.
- Y relays request with new minimum  $2/3$ .
- When X relays, leaves min at  $2/3$ .
- B cannot reply because its  $2/3$  label not less than requested minimum  $2/3$ .
- A replies and B takes new label from mediant.
- **Y keeps  $2/3$  because  $5/8 < 2/3 < 3/4$ .**



# Re-labeling Example

- Z issues RREQ for T with its current  $3/4$  label.
- Y relays request with new minimum  $2/3$ .
- When X relays, leaves min at  $2/3$ .
- B cannot reply because its  $2/3$  label not less than requested minimum  $2/3$ .
- A replies and B takes new label from mediant.
- Y keeps  $2/3$  because  $5/8 < 2/3 < 3/4$ .
- **Final labeling:  $3/4 \rightarrow 2/3 \rightarrow 5/8 \rightarrow 3/5 \rightarrow 1/2 \rightarrow 0/1$**



# Overflow

---

- 32-bit integer proper fractions may overflow after large number of mediant operations (without using reductions).
- Include a sequence number for LDR-style path reset in those cases.
- Like OSPF, use open-ended SN space. Could be based on real-time clock, no practical overflow problems.

# Simulations

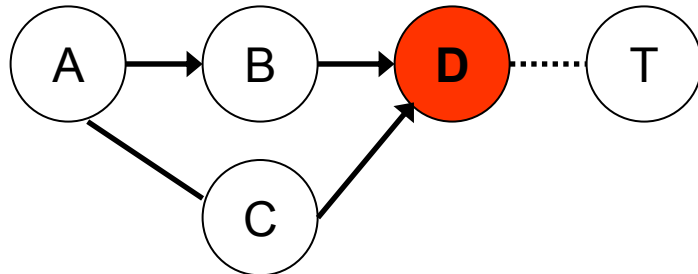
---

- GloMoSim simulator
- 802.11 MAC, 2 Mbps
- 100-nodes, random-waypoint 0 – 20 m/s
- 30 flows at 4 pps each, 60s mean length
- CBR traffic, 512-byte packets
- Ran 10 simulations per pause time, average results

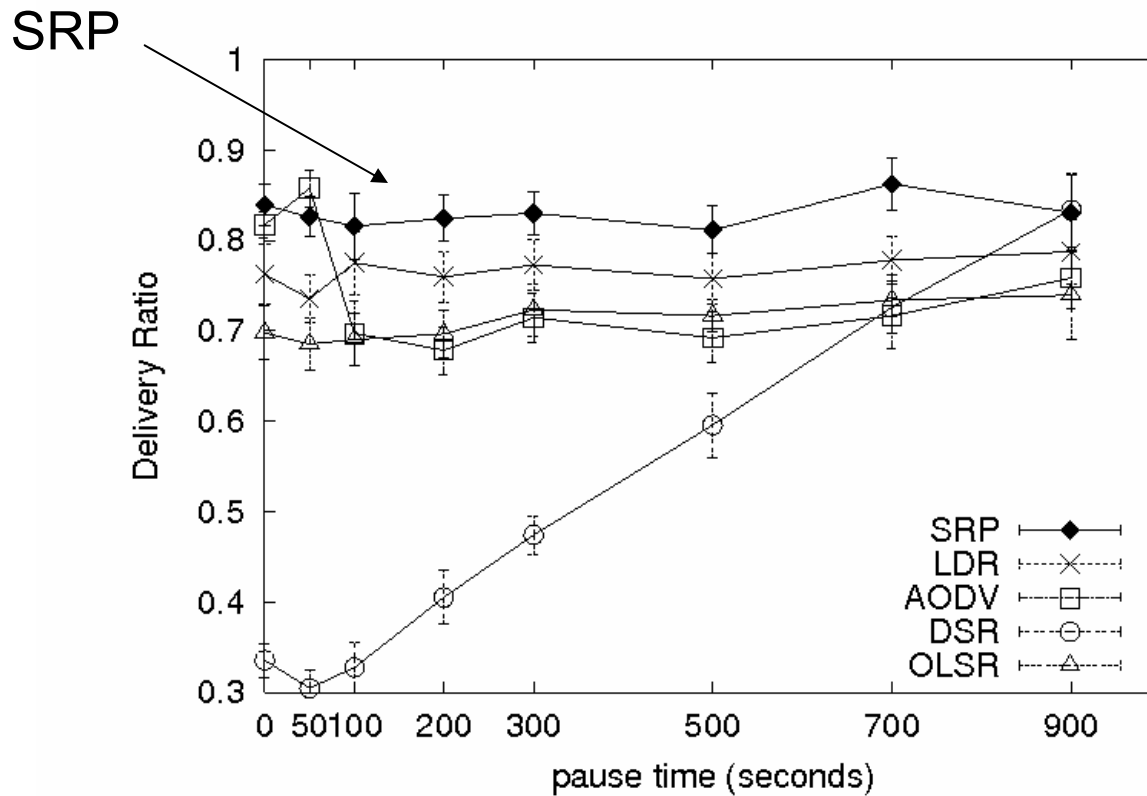
# Optimizations

---

- Use packet cache
  - Use link-layer drop notification
  - RREQ & resend packets dropped at link-layer
- Avoid false positives
  - Lie about ordering to force a packet to travel several hops before getting a response.

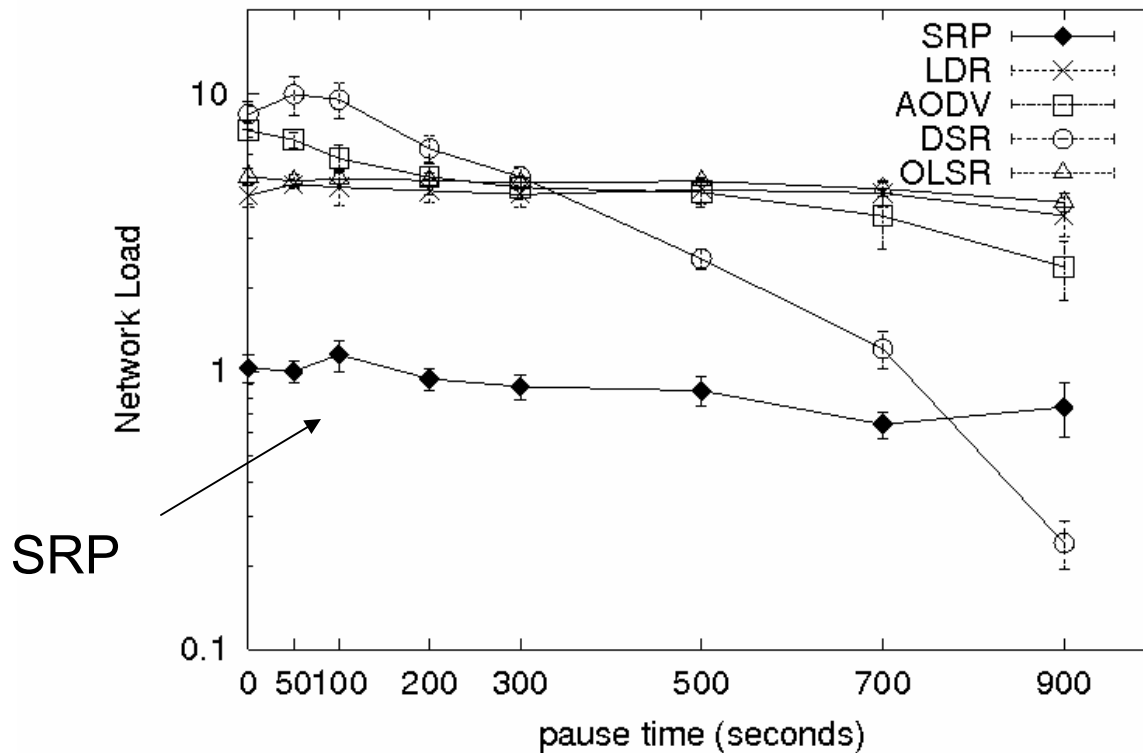


# Performance: Delivery Ratio



100-node simulation, 30-flows, 120 packets per second

# Performance: Control overhead



100-node simulation, 30-flows, 120 packets per second

## Future work

---

- Fraction reductions
- Do other label sets make sense?
- Implement under NS2
- Implement multi-path version
- Try non-minhop path choices
- Try our GloMoSim code under Linux
  - We currently have a Linux/Glomosim wrapper.

## Conclusion

---

- New class of on-demand routing protocols using dense label set.
- Results in minimum re-labeling.
- Re-labeling done independently based on information in RREQ/RREP packets.
- Significant reduction in control overhead.