Improving TCP Performance over Mobile Ad Hoc Networks by Exploiting Cross-Layer Information Awareness

Xin Yu

Department Of Computer Science

New York University, New York

Presented By : Adwait Belsare

Outline

- **Introduction**
- Background
- Mobility, TCP and ELFN
- Early Packet Loss Notification and Best-Effort Ack Delivery
- Performance Evaluation
- Related Work
- Conclusions
- Comments

Introduction

- TCP performance degrades as a result of packet losses due to route failures in mobile ad hoc networks.
- TCP treats this as congestion and invokes congestion control mechanisms resulting in reduction of throughput.
- ELFN : A transport layer mechanism to address problems caused by mobility.

Early Link Failure Notification (ELFN)

- One of the promising approaches to provide link failure feedback to TCP.
- On detecting link failure, a node will notify the TCP sender about the failure and the packet that encountered the failure.
- On receiving notification, TCP freezes its retransmission timer and periodically sends a probing packet until it receives an ACK.
- TCP then restores its transmission timer and continues as normal.

- TCP times out due to data packets as well as ACK losses.
- This paper proposes to make routing protocols aware of lost data packets and ACKs and help reduce TCP timeouts for mobility induced losses.
- Two mechanisms: Early Packet Loss Notification (ELPN) and Best Effort ACK delivery (BEAD) are proposed.

- Introduction
- **Background**
- Mobility, TCP and ELFN
- Early Packet Loss Notification and Best-Effort Ack Delivery
- Performance Evaluation
- Related Work
- Conclusions
- Comments

Background

- Dynamic Source Routing (DSR).
- Distributed Cache Algorithm
  - An algorithm to address cache staleness issue.
  - When a node detects a link failure, the algorithm proactively notifies all reachable nodes that have cached that link.
  - Each node maintains a cache table.
  - The table stores routes as well as information necessary for cache updates.
  - The algorithm uses local information kept by each node to achieve distributed cache updating.

Simulation Environment

- *ns*-2 network simulator.
- The network interface uses 802.11 DCF MAC protocol.
- The mobility model is *random waypoint model* in a rectangular model.
- Field configurations : 1500m x 1000m with 50 nodes and 2200m x 600m with 100 nodes.
- TCP Reno with packet size of 1460 bytes.
- Used FTP flows.

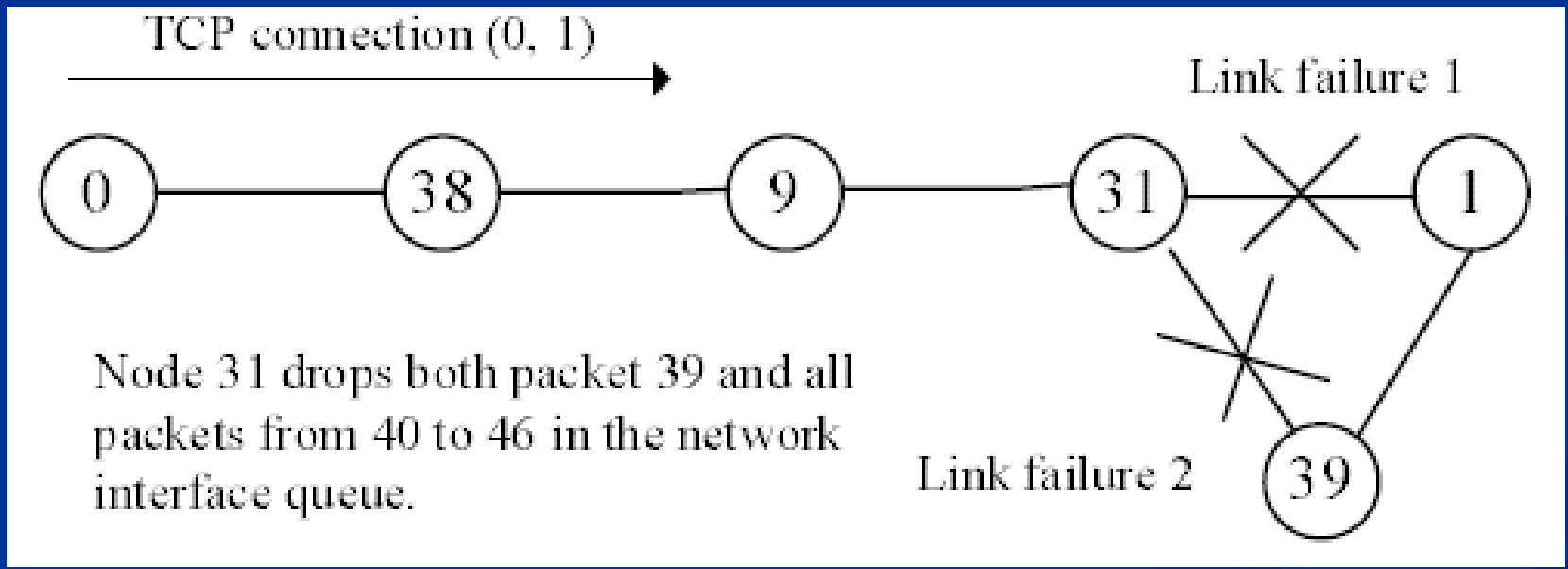
- Introduction
- Background
- **Mobility, TCP and ELFN**
- Early Packet Loss Notification and Best-Effort Ack Delivery
- Performance Evaluation
- Related Work
- Conclusions
- Comments

Mobility, TCP and ELFN

- Explore three issues

1. How to set RTO and cwnd after congestion control mechanisms are restored.
2. Whether to freeze TCP upon route failures or upon packet losses.
3. The network layer is unaware of lost data packets and ACKs.

How to set RTO and cwnd after congestion control mechanisms are restored

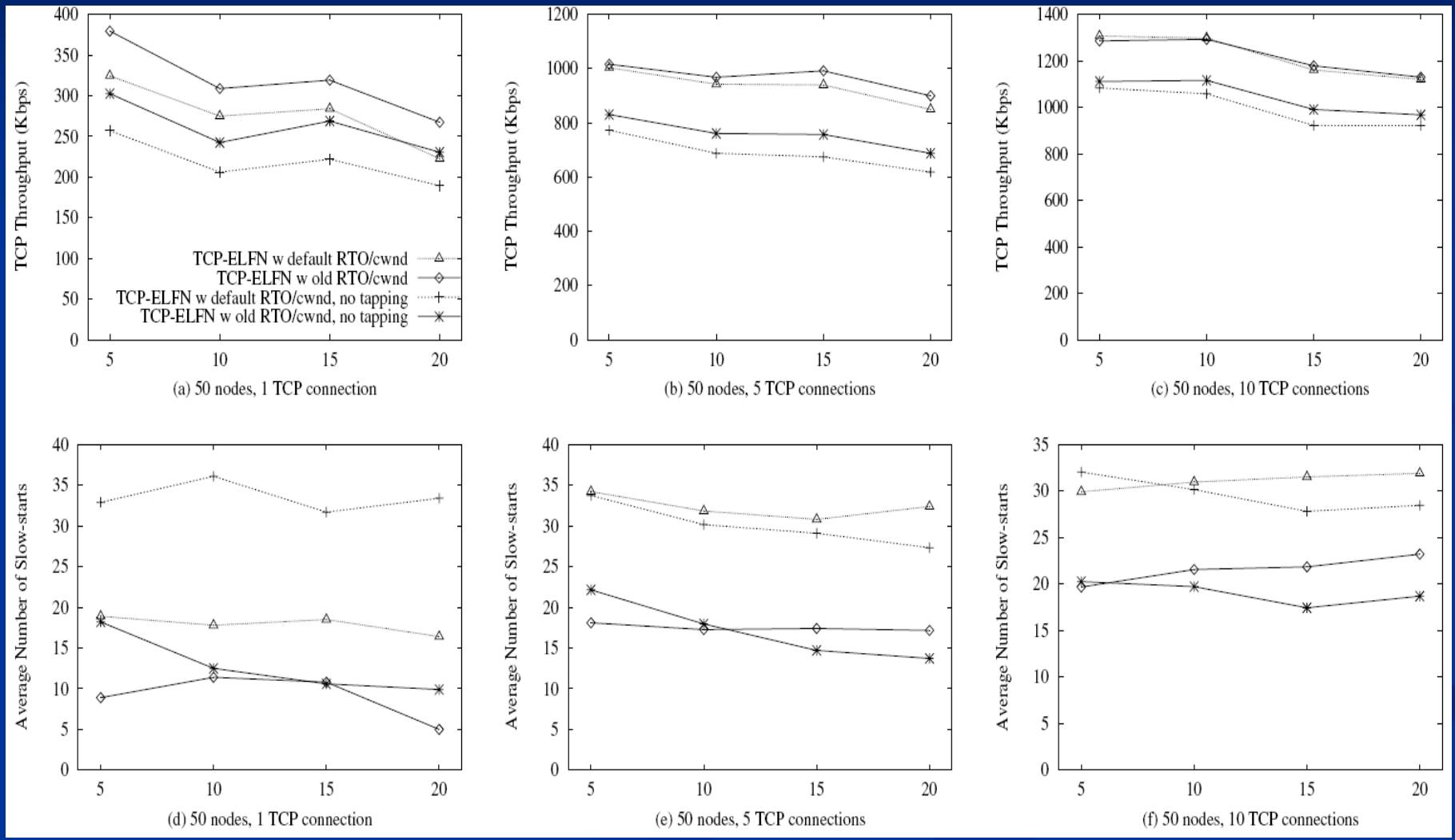


- Previous work concluded that using default values of RTO and cwnd does not impact throughput .
- Author argues that using default values degrades throughput as small cwnd causes TCP to go in idle state.
- Author concludes keeping TCP state the same as it was frozen as TCP relies on RTO to recover from idle state.

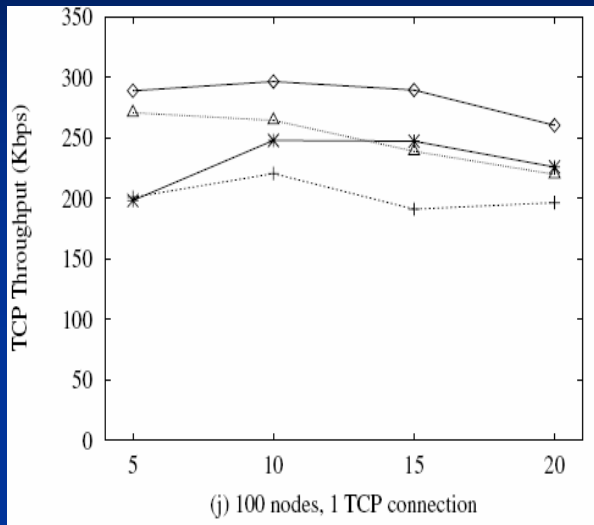
Evaluation

- Evaluated TCP performance for DSR under promiscuous and non promiscuous mode.
- Optimization of DSR by disabling network interface's filtering function.

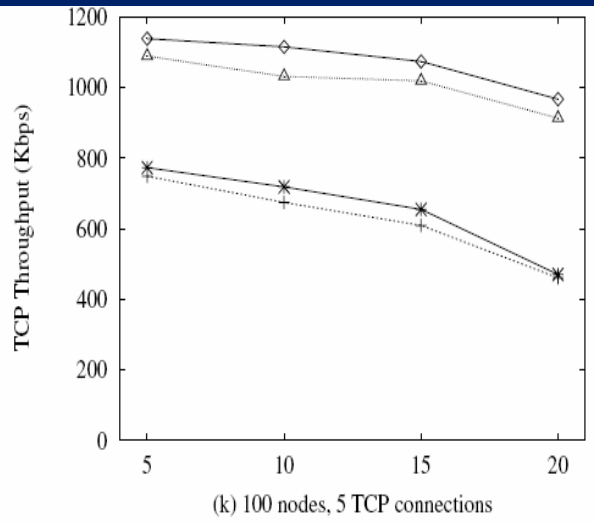
Evaluation


TCP Throughput (Kbps)
TCP-ELFN w default RTO/cwnd
TCP-ELFN w old RTO/cwnd
TCP-ELFN w default RTO/cwnd, no tapping
TCP-ELFN w old RTO/cwnd, no tapping
(a) 50 nodes, 1 TCP connection
(b) 50 nodes, 5 TCP connections
(c) 50 nodes, 10 TCP connections
Average Number of Slow-starts
(d) 50 nodes, 1 TCP connection
(e) 50 nodes, 5 TCP connections
(f) 50 nodes, 10 TCP connections

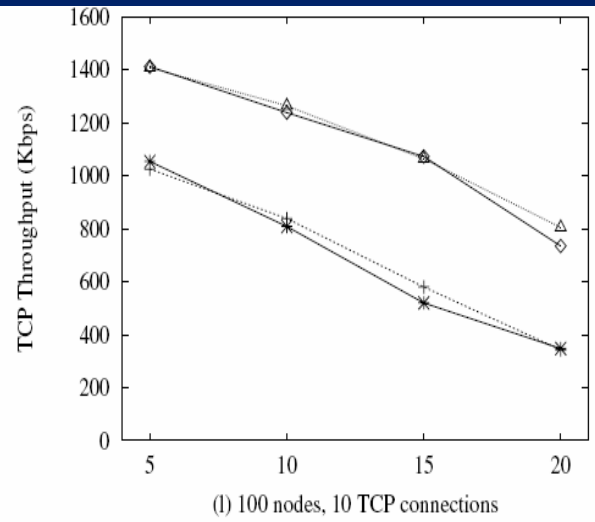
Evaluation



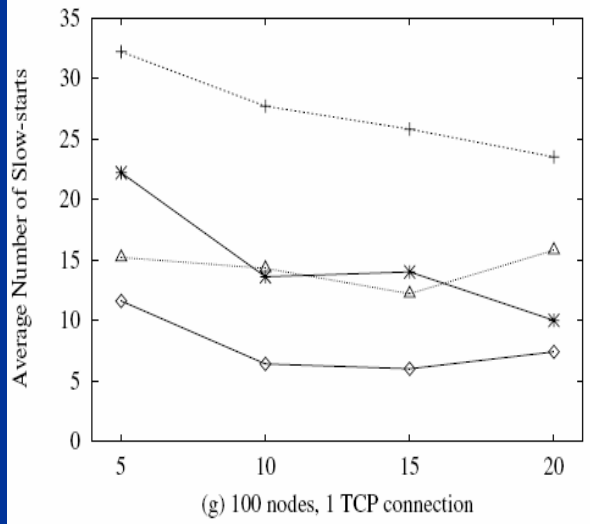
(j) 100 nodes, 1 TCP connection



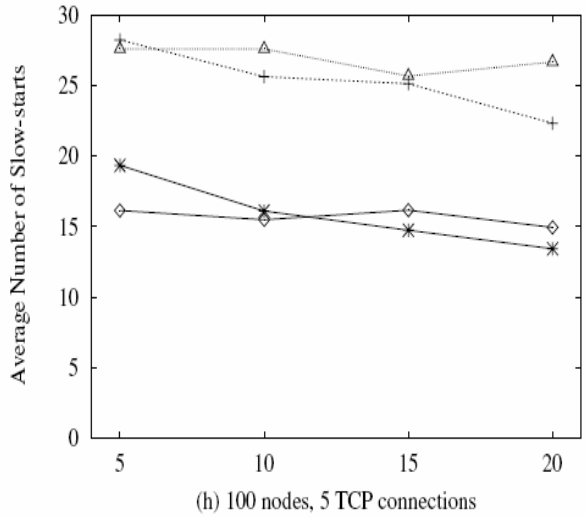
(k) 100 nodes, 5 TCP connections



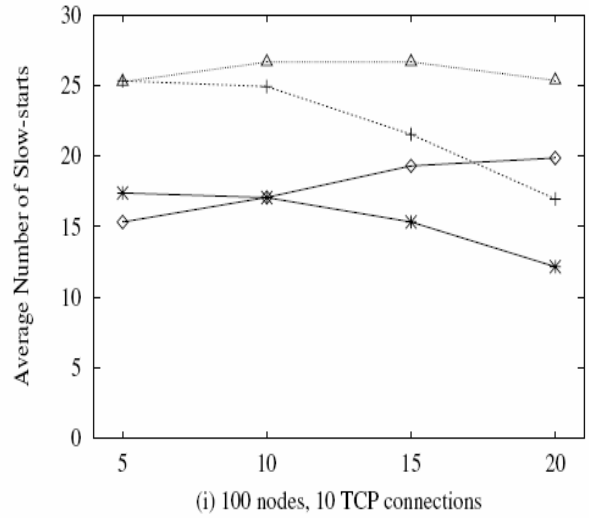
(l) 100 nodes, 10 TCP connections



(g) 100 nodes, 1 TCP connection

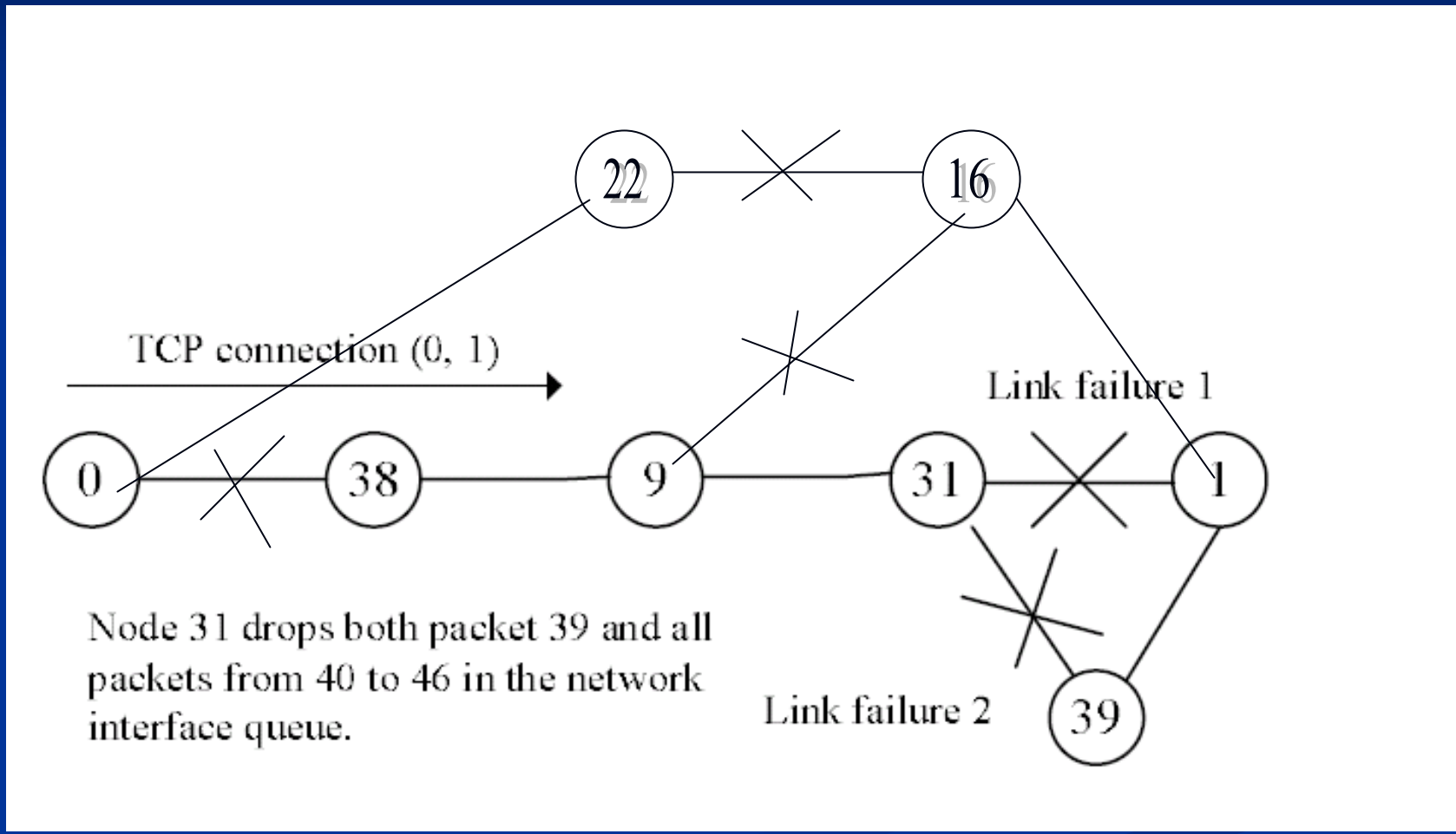


(h) 100 nodes, 5 TCP connections



(i) 100 nodes, 10 TCP connections

When should TCP be frozen?



TCP should be frozen on route failures.

Unaware of Lost packets and ACKs

- Upon route failures, a routing protocol silently drops all the packets with the same next hop in the network interface queue.
- TCP unaware of these losses times out.
- Therefore, it is necessary to let TCP know about lost packets which can be done by intermediate nodes.
- Same is true for ACKs.

- Introduction
- Background
- Mobility, TCP and ELFN
- **Early Packet Loss Notification and Best-Effort ACK Delivery**
- Performance Evaluation
- Related Work
- Conclusions
- Comments

Overview

- **IDEA**: The intermediate nodes notify TCP senders about lost data packets and retransmit ACKs for lost ACKs by extensively using cached routes.
- Three types of packets are considered:

1. Data packets.
2. ACKs
3. Packet loss notifications.

Operation of the mechanisms

- Data packets or ACKs are dropped and this is the first time they encounter a link failure.
- Data packets are dropped after being salvaged by intermediate node.
- ACKs are dropped after being salvaged by an intermediate node.
- When forwarding a notification about lost ACK, the node attempts to retransmit an ACK with highest sequence number.
- Notification packet dropped.

At the node detecting link failure

- If the node is TCP sender, it sends an ICMP message to TCP including the sequence number of the packet.
- If the node is intermediate node, it piggybacks on the LINK ERROR information about the lost packets that have the same source node.
- For lost packets that have different source nodes the node sends notification to each node.

At the node receiving a notification

- If the node is a TCP sender, it sends ICMP message to TCP for each sequence number.
- If the node is TCP receiver and no ACK was sent, it sends ACK with highest sequence number among lost ACKs if it has cached route to the sender.
- The node is an intermediate node.

Variables:
iph: IP header; *tcph*: TCP header; *srh*: source route header;
p: the current packet; *new_p*: a new packet;
deliver_to_dest: whether to send an ACK to the TCP sender;

```
if has_conn_info then
  for all entry e ∈ conn_list do
    if e.src = net_id then
      p.iph.saddr ⇐ e.src
      p.iph.sport ⇐ e.sport
      p.iph.daddr ⇐ e.dst
      p.iph.dport ⇐ e.dport
      if e.ptype = TCP then
        for all seq_no ∈ e do
          new_p ⇐ p.copy()
          new_p.tcph.seqno ⇐ seq_no
          sendICMPtoTCP(new_p)
        end for
      end if
      if e.ptype = ACK and e.ack_sent = FALSE then
        new_p ⇐ p.copy()
        new_p.tcph.seqno ⇐ max(seq_no ∈ e)
        new_p.srh.has_conn_info ⇐ FALSE
        new_p.src ⇐ net_id
        new_p.dest ⇐ p.iph.daddr
        if findRoute(new_p.dest, new_p.route) then
          sendOutPacketWithRoute(new_p)
        end if
      end if
    else
      deliver_to_dest ⇐ FALSE
      new_p ⇐ p.copy()
      if e.ptype = ACK and e.ack_sent = FALSE then
        if findRoute(e.dst, new_p.route) then
          new_p.dest ⇐ e.dst
          deliver_to_dest ⇐ TRUE
        else if findRoute(e.src, new_p.route) then
          new_p.dest ⇐ e.src
        end if
      else if e.ptype = TCP then
        if findRoute(e.src, new_p.route) then
          new_p.dest ⇐ e.src
        end if
      end if
      if new_p.route ≠ ∅ then
        new_p.src ⇐ net_id
        if e.ptype = ACK and deliver_to_dest = TRUE then
          new_p.tcph.seqno ⇐ max(seq_no ∈ e)
          new_p.iph.saddr ⇐ e.src
          new_p.iph.sport ⇐ e.sport
          new_p.iph.daddr ⇐ e.dst
          new_p.iph.dport ⇐ e.dport
          new_p.srh.has_conn_info ⇐ FALSE
          sendOutPacketWithRoute(new_p)
        else if num_route_error = 1 then
          new_p.srh.conn_list[0] ⇐ e
          new_p.srh.has_conn_info ⇐ TRUE
          sendOutPacketWithRoute(new_p)
        end if
      end if
    end if
  end for
end if
```

lgorithm 1: At the Node Receiving a Packet Loss Notification

Examples

- EPLN

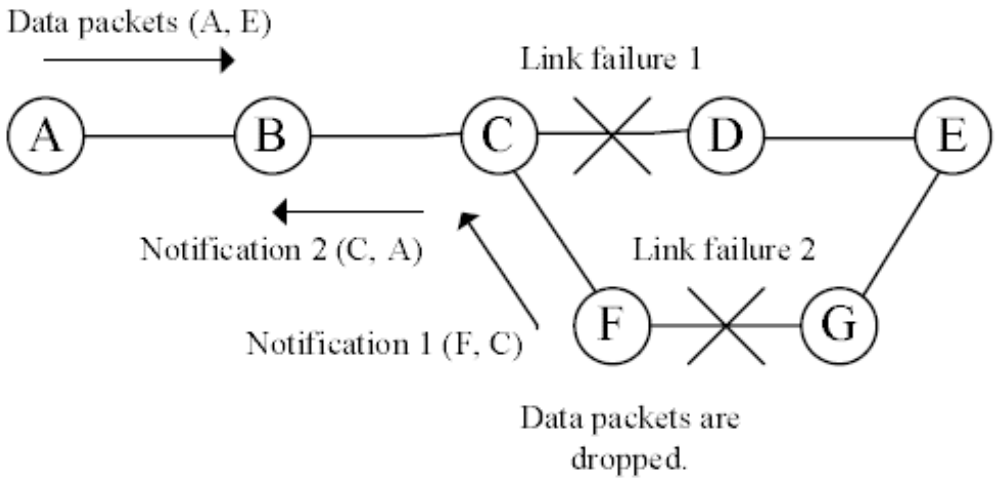


Figure 2: An Example of Early Packet Loss Notification

Examples

- BEAD

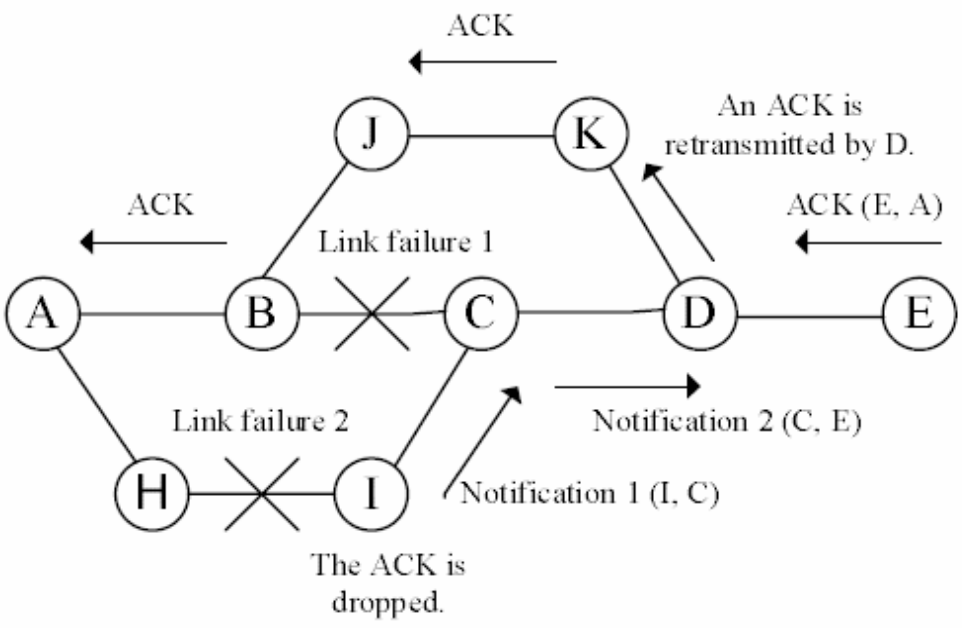


Figure 3: An Example of Best-Effort ACK Delivery

Cross Layer Interactions

- They happen at TCP sender.
- Since TCP is frozen upon route failures, the network layer will send ICMP message to TCP even if the packet encountering the link failure is salvaged.
- The ICMP message sent to TCP sender also contains information indicating whether a packet is lost.

TCP-Reno *recv*():
Variables:
tcph: TCP header; *icmp*: ICMP header;
p: the current packet;
thaw_timer: the thaw timer in ELFN;
t_thaw: the probing interval;
thaw_seqno: the sequence number of the probing packet;
lost_pkt: an array recording the lost packets that have not been retransmitted;
num_lost_pkt: the number of packets in *lost_pkt*;
tcp_melt: whether TCP's state is restored or not;

```
if p.ptype = ICMP then
    if not (p.tcph.seqno > highest_ack_ and
      p.tcph.seqno <= highest_ack_+window()) then
        free(p)
        return
    end if
    if not frozen() then
        freeze()
    end if
    if thaw_timer.status() = TIMER_IDLE then
        thaw_timer.resched(t_thaw)
        thaw_seqno ⇐ p.tcph.seqno
        if p.icmp.pkt_lost = TRUE then
            output(thaw_seqno)
        end if
    else if p.tcph.seqno < thaw_seqno then
        thaw_seqno ⇐ p.tcph.seqno
        if p.icmp.pkt_lost = TRUE then
            output(thaw_seqno)
        end if
    else if p.tcph.seqno = thaw_seqno then
        if p.icmp.pkt_lost = TRUE then
            output(thaw_seqno)
        end if
    else if p.tcph.seqno > thaw_seqno then
        if p.icmp.pkt_lost = TRUE and
          p.tcph.seqno ∉ lost_pkt then
            lost_pkt[num_lost_pkt] ⇐ p.tcph.seqno
            num_lost_pkt ⇐ num_lost_pkt + 1
        end if
        free(p)
    end if
else if frozen() then
    melt()
end if
if tcp_melt = TRUE then
    if num_lost_pkt ≠ 0 then
        for all seqno ∈ lost_pkt do
            if seqno > last_ack_ then
                output(seqno)
            end if
        end for
        num_lost_pkt ⇐ 0
        tcp_melt ⇐ FALSE
    end if
end if
{/*followed by the operation executed by TCP when it receives
an ACK.*/}
```

Algorithm 2: At a TCP Sender

- Introduction
- Background
- Mobility, TCP and ELFN
- Early Packet Loss Notification and Best-Effort ACK Delivery
- **Performance Evaluation**
- Related Work
- Conclusions
- Comments

Performance Evaluation

- Compared TCP performance enhanced with three combination of mechanisms at transport layer and the network layer:

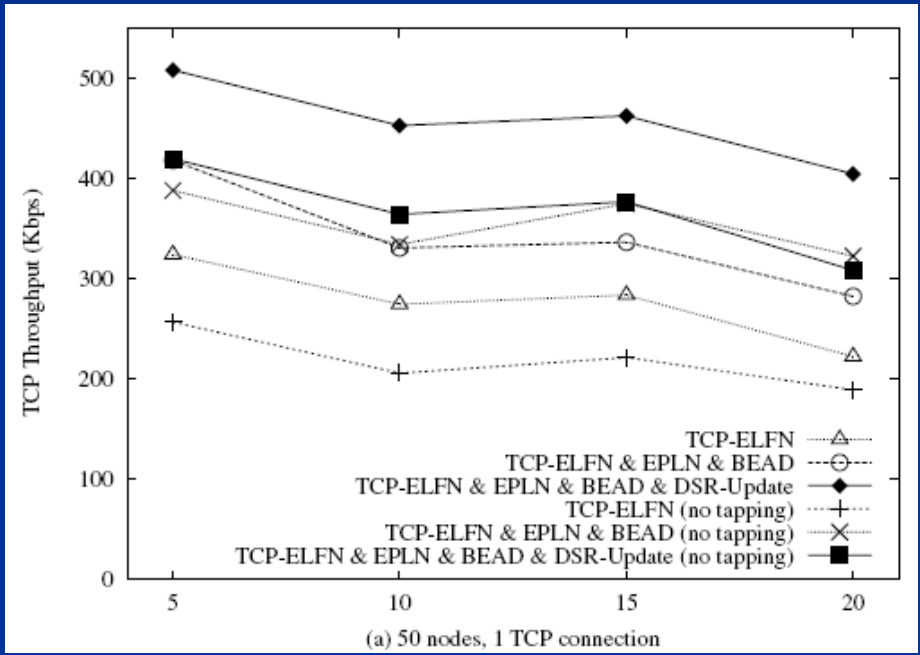
1. TCP-ELFN with default RTO 6s and cwnd 2; and DSR.
2. TCP-ELFN with RTO and cwnd set to the values computed before TCP was frozen, and DSR with EPLN, BEAD.
3. TCP-ELFN with RTO and cwnd set to the values computed before TCP was frozen, and DSR with EPLN, BEAD and DSR update.

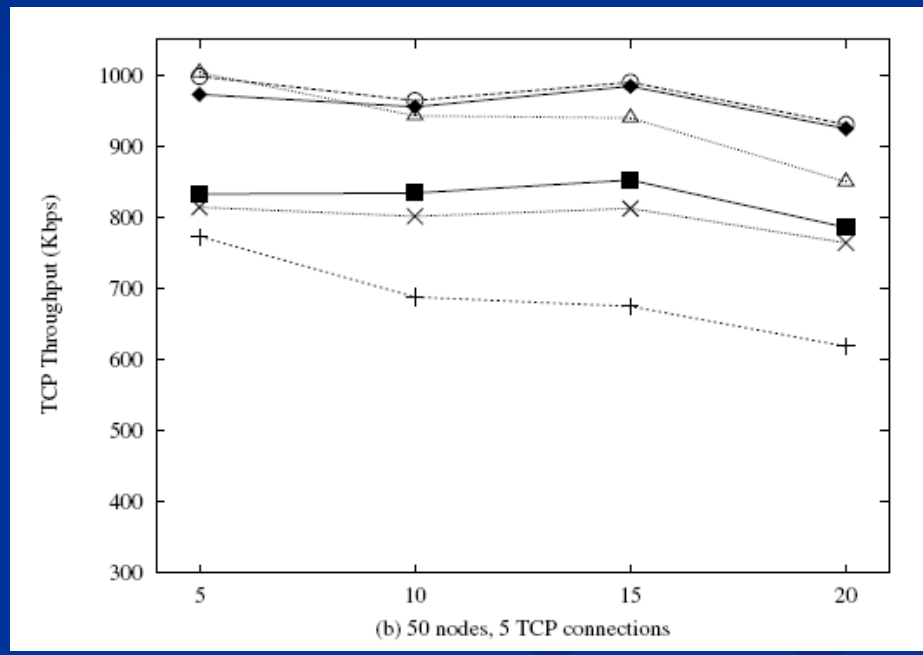
Performance Evaluation

- Traffic load with 1,5 and 10 TCP connections scenarios.
- Node speeds used were 5,10,15 and 20 m/s.
- Metrics:

1. TCP throughput
2. Average number of slow starts.
3. Packet Overhead.

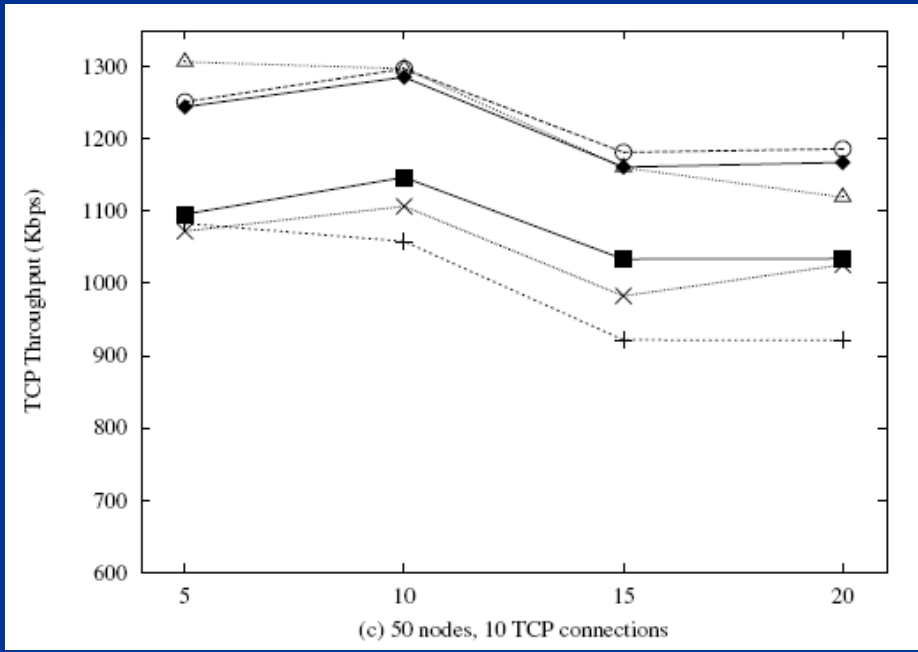
TCP throughput

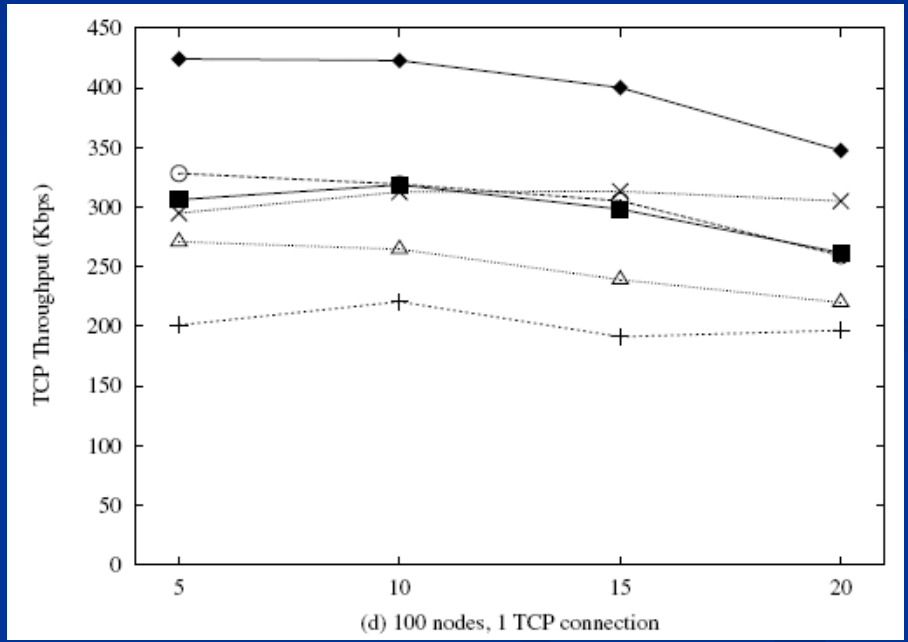




TCP with EPLN and BEAD with DSR update performs almost similar to TCP with EPLN and BEAD using path caches under promiscuous mode.

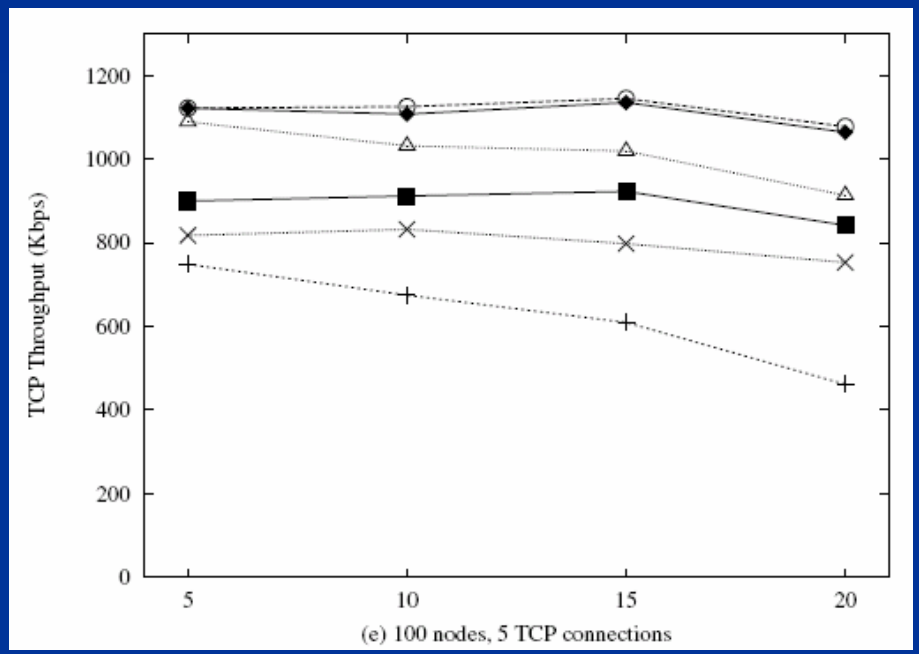
TCP throughput

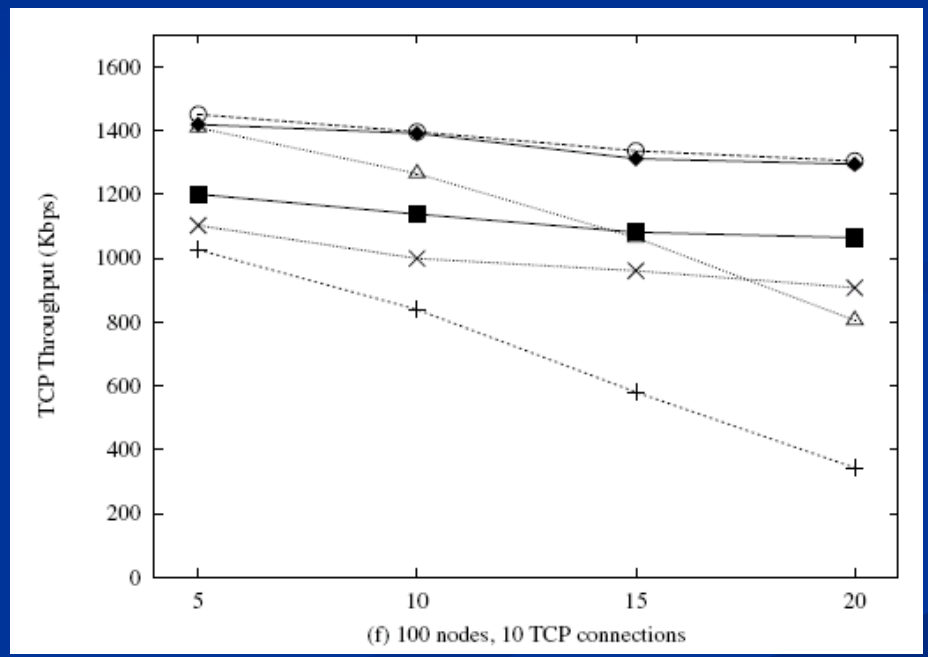




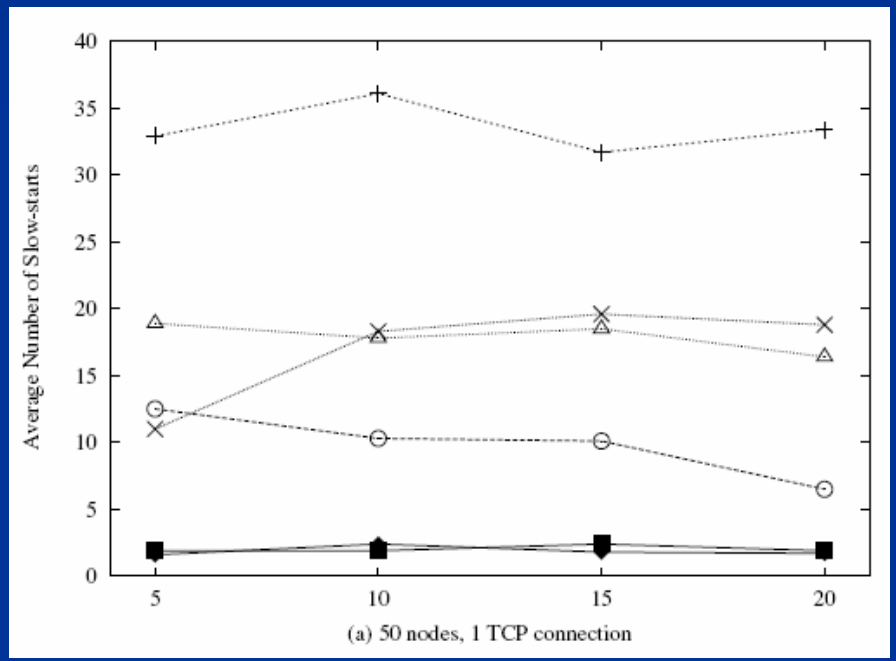
Claim: TCP with EPLN and BEAD with DSR update always outperforms the other two mechanisms under non promiscuous mode. (NOT TRUE)

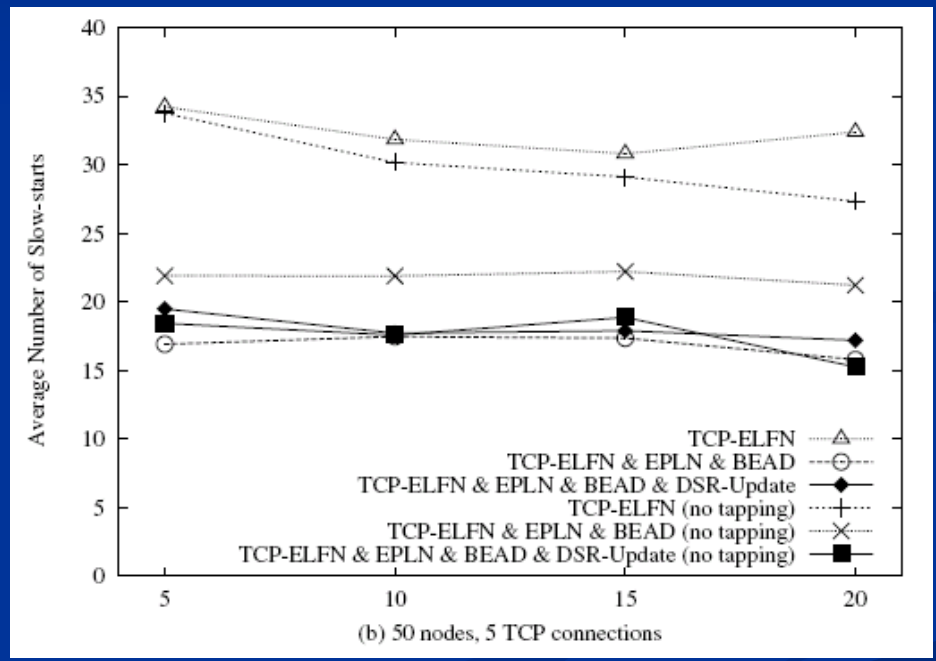
TCP throughput



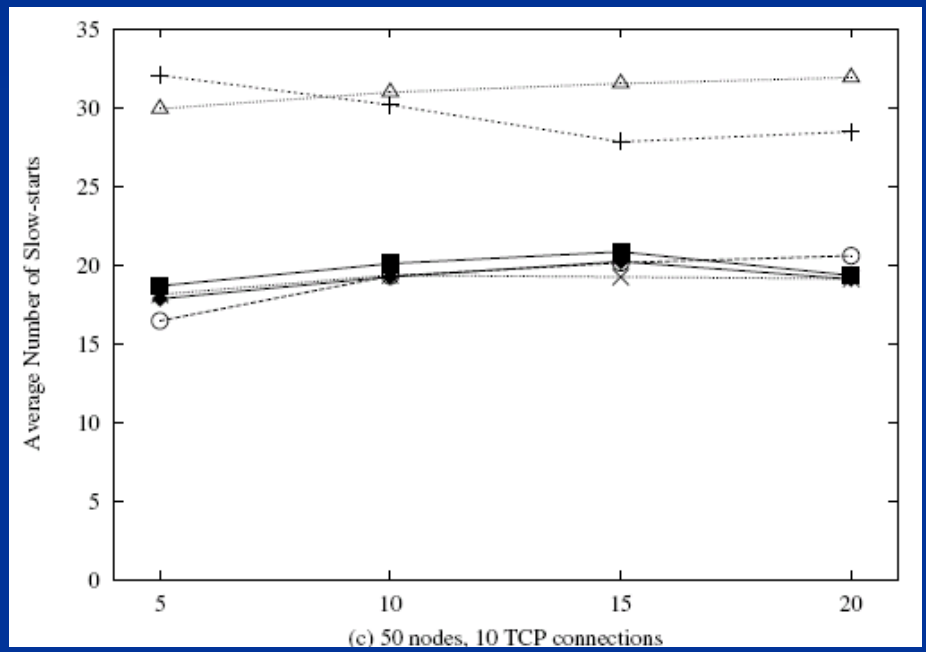


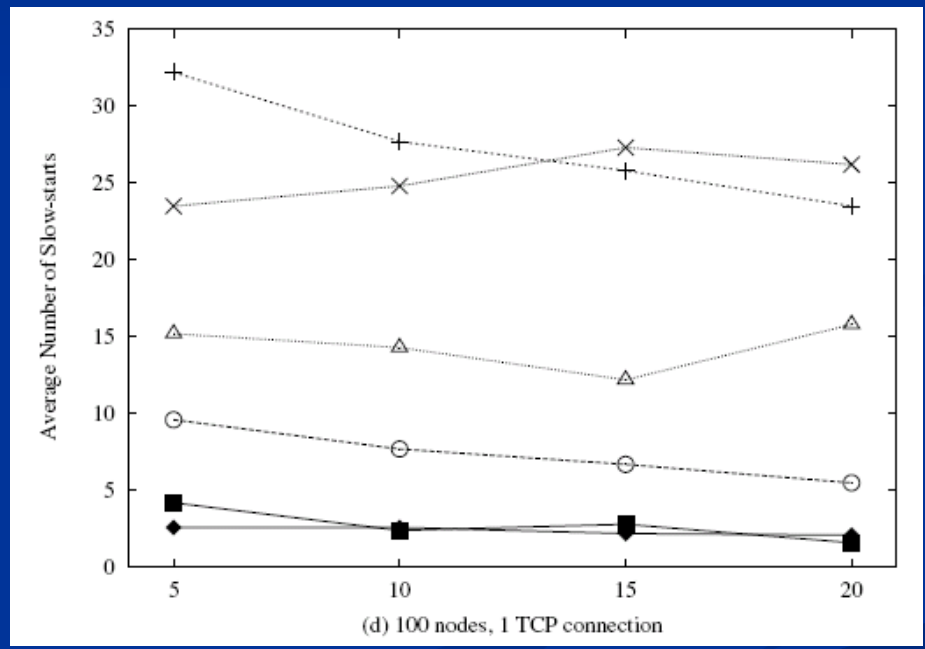
Average number of slow starts



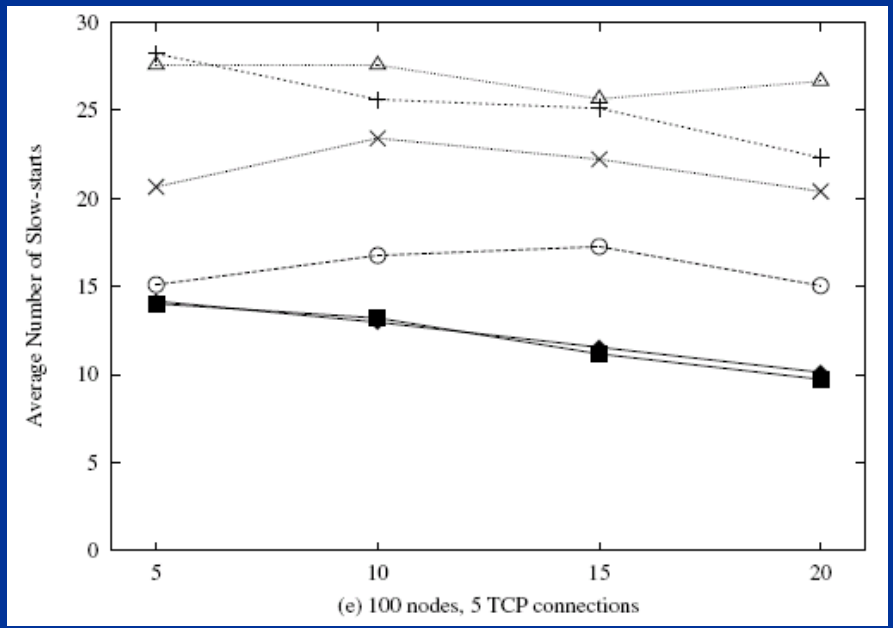


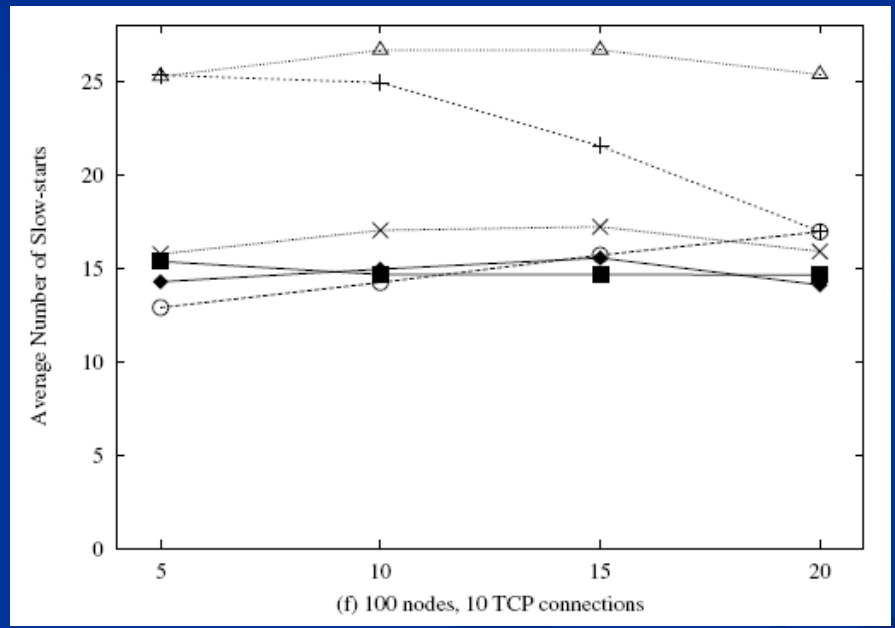
Average number of slow starts



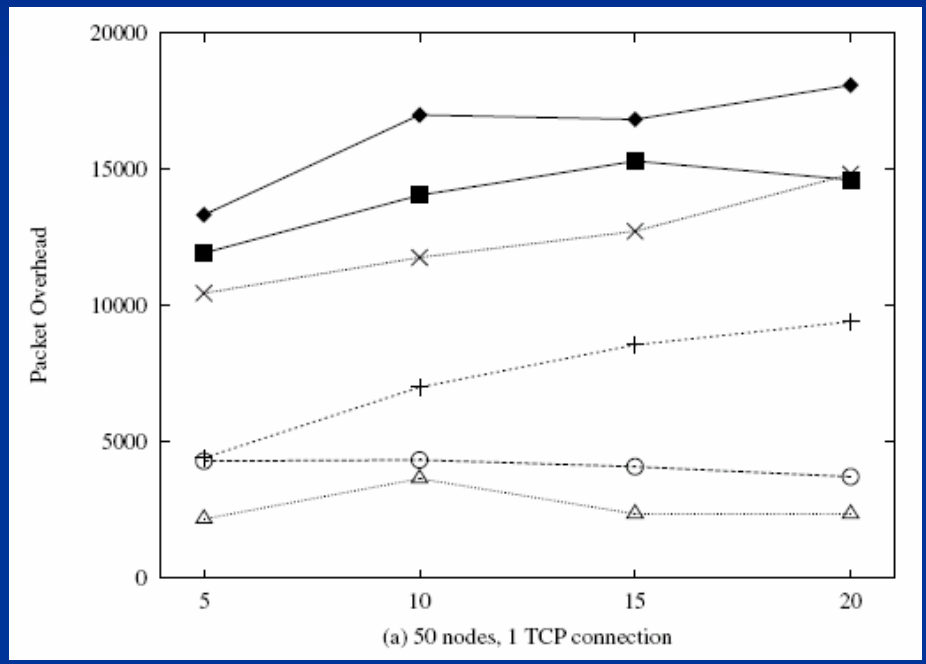


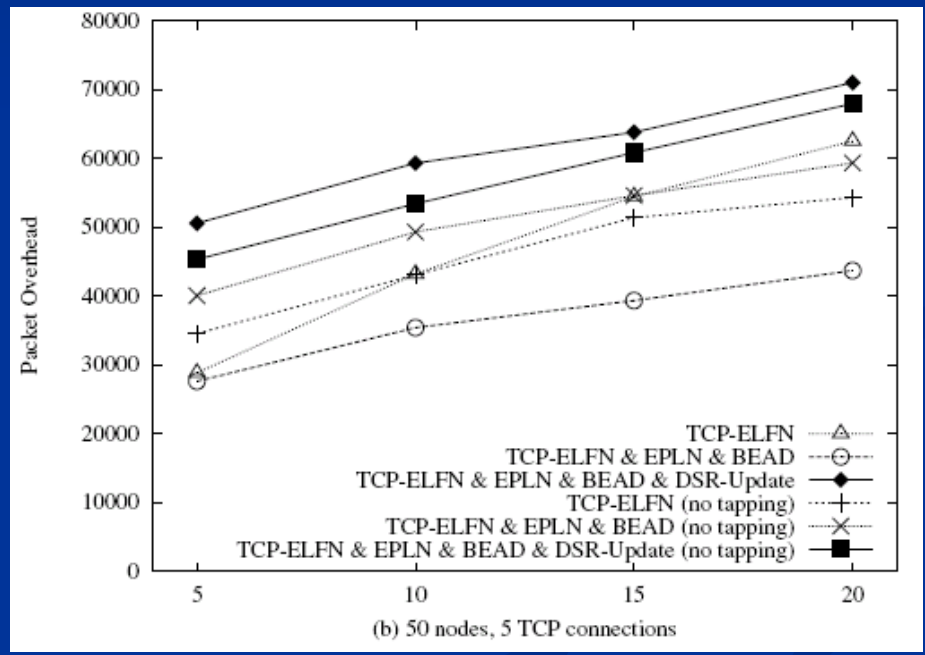
Average number of slow starts





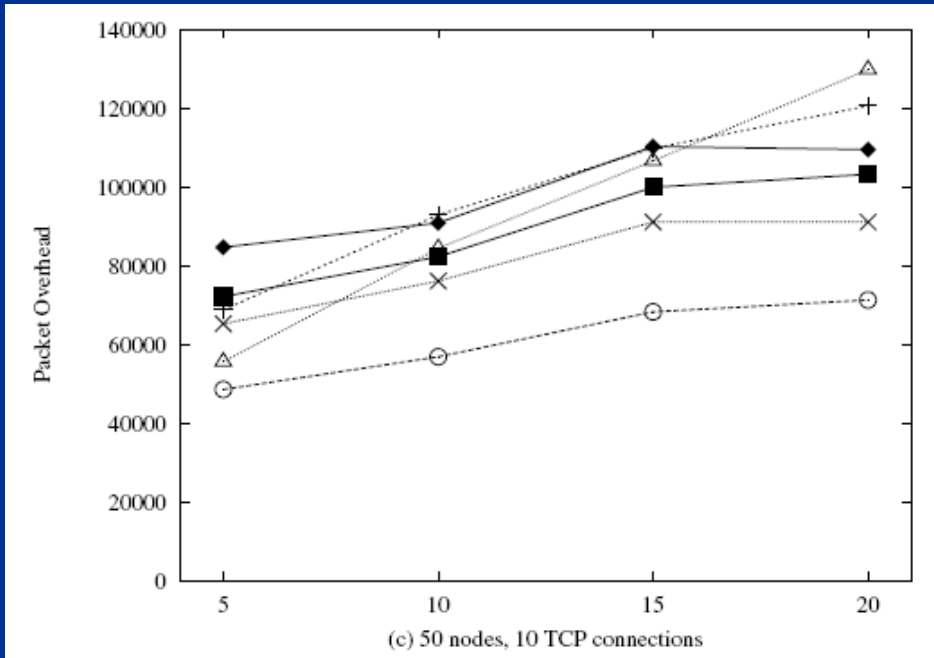
Packet Overhead

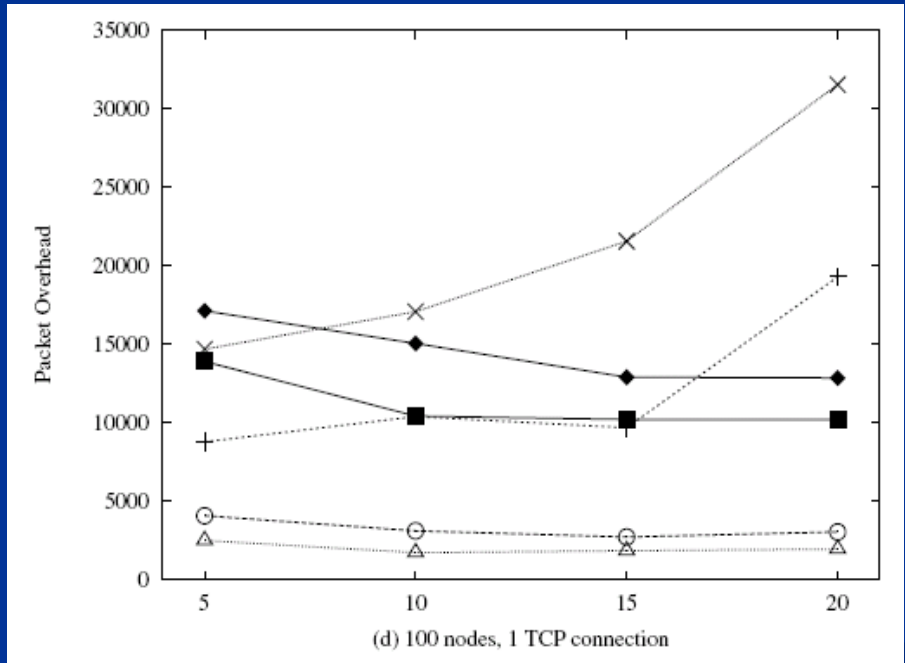




High packet overheads for TCP with EPLN, BEAD with DSR update.

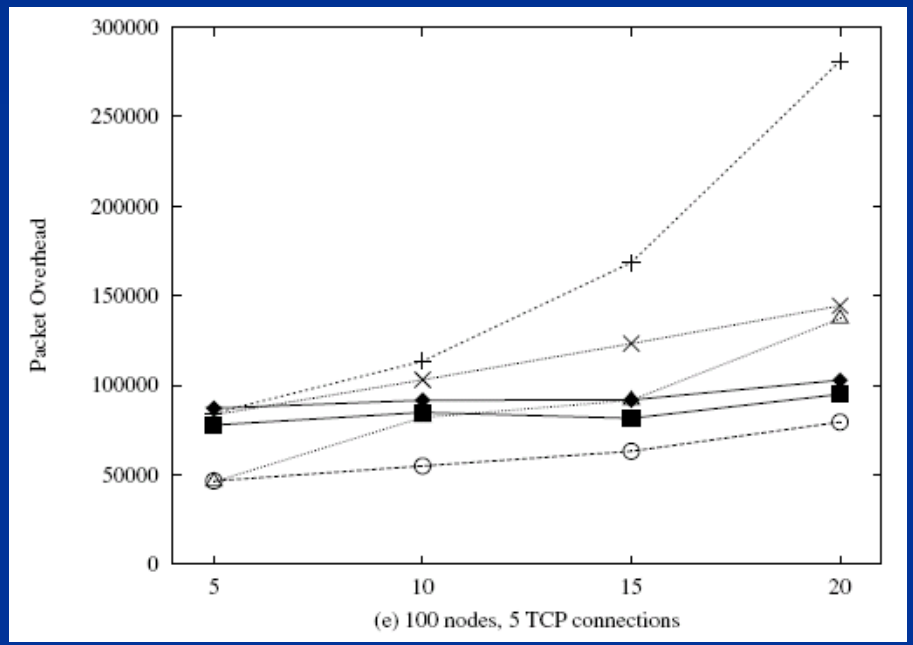
Packet Overhead

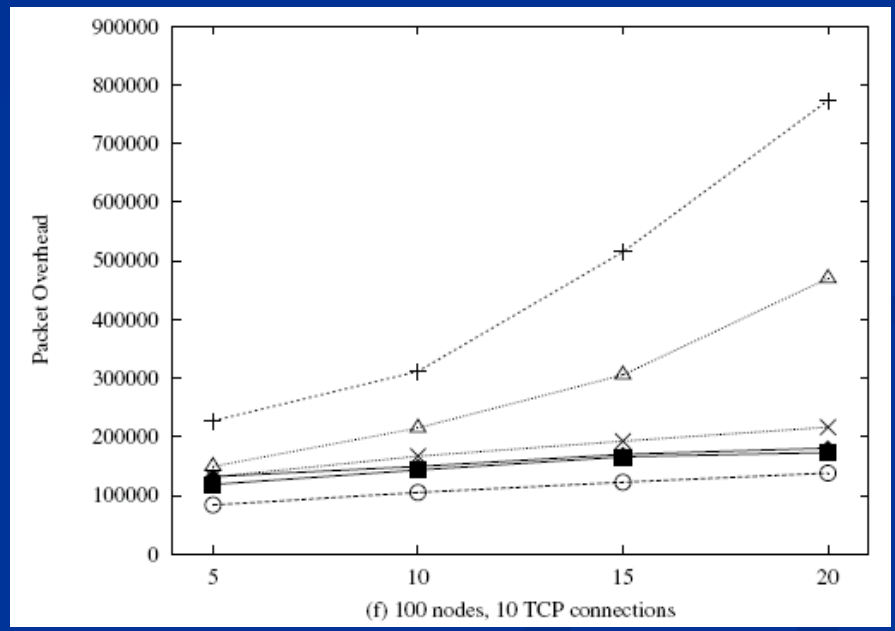




The overhead of TCP ELFN decreases under promiscuous mode as DSR uses secondary cache which helps reduce route discoveries.

Packet Overhead





Under promiscuous mode TCP with EPLN and BEAD performs best

- Introduction
- Background
- Mobility, TCP and ELFN
- Early Packet Loss Notification and Best-Effort ACK Delivery
- Performance Evaluation
- **Related Work**
- Conclusions
- Comments

Related Work

- Most of the previous work focused on transport layer mechanisms. The major approach was to provide link failure feedback to TCP.

  For e. g ELFN.

- Ignored the important fact that route failures are do not imply that packets are lost.

- Introduction
- Background
- Mobility, TCP and ELFN
- Early Packet Loss Notification and Best-Effort ACK Delivery
- Performance Evaluation
- Related Work
- **Conclusions**
- Comments

Conclusions

- Cross layer information awareness is the key to making TCP efficient in the presence of mobility.
- As a result TCP reacts quickly to lost packets and is unaware of lost ACKs.
- It is important to make route caches adapt fast to topology changes because the validity of cached routes affects TCP performance.

- Introduction
- Background
- Mobility, TCP and ELFN
- Early Packet Loss Notification and Best-Effort ACK Delivery
- Performance Evaluation
- Related Work
- Conclusions
- **Comments**

Comments

- The two feedback mechanisms are applicable to any routing protocol as they address general problems that occur at network layer.
- The cache route updates need to be synchronized in order to achieve proposed solution which is not simple in real networks.
- A node takes no action if it does not have cached route to the destination even in case of dropped packets.
- The simulations presented do not confirm the claims made by the author.

Thank You