CMPE 257: Wireless and Mobile Networking

Spring 2005
E2E Protocols (point-to-point)
E2E Protocols (multipoint)
Announcements
Today

E2E protocols (cont’d).
Finish point-to-point.
Reliable multipoint.
E2E Approaches

Strict E2E versus E2E with intermediate node involvement.

E2E with intermediate node involvement:

Explicit notifications.
Explicit Notification Schemes

General Philosophy

Approximate ideal TCP behavior.

Ideally, TCP sender should simply retransmit a packet lost due to transmission errors without taking any congestion control actions.

A node determines whether packets are lost due to errors and informs sender using an “explicit notification”.

Sender, on receiving the notification, does not reduce congestion window, but retransmits lost packet.
Explicit Notification Schemes

Motivated by Explicit Congestion Notification (ECN) proposals [Floyd94].

Variations proposed in literature differ in:
Who sends explicit notification.
How they decide to send explicit notification.
What sender does on receiving notification.
Explicit Loss Notification

[Balakrishnan98]

MH is the TCP sender.
Wireless link first on path from sender to receiver.
Base station keeps track of holes in packet sequence.
When a dupack is received from the receiver, BS compares the dupack sequence number with recorded holes.

If there is a match, an ELN bit is set in dupack.
When sender receives dupack with ELN set, it retransmits packet, but does not reduce congestion window.
Explicit Loss Notification

[Biaz99thesis]

Adapts ELN proposed in [Balakrishnan98] for the case when MH is receiver.

Caches TCP sequence numbers at base station, similar to Snoop. But does not cache data packets, unlike Snoop.

Duplicate acks are tagged with ELN bit before being forwarded to sender if sequence number for the lost packet is cached at BS.

Sender takes appropriate action on receiving ELN.
Sequence numbers cached at base station

TCP sender 36 37 37

Dupack with ELN

TCP receiver 39 38 37

ELN [Biaz99thesis]
Explicit Bad State Notification
[Bakshi97]

MH is TCP receiver.
BS attempts to deliver packets to MH using link layer retransmission scheme.
If packet cannot be delivered using small number of retransmissions, BS sends a Explicit Bad State Notification (EBSN) message to TCP sender.
When TCP sender receives EBSN, it resets RTO.
  Timeout delayed when wireless channel in bad state.
Partial Ack Protocols

[Cobb95][Biaz97]

Send two types of acknowledgements. Partial ACK informs sender that a packet was received by an intermediate host (typically, base station). Normal TCP cumulative ACK needed by sender for reliability purposes.
Partial Ack Protocols

When packet for which partial ack is received detected to be lost, sender does not reduce its congestion window.

Loss assumed to be due to wireless errors.
Base station may or may not locally buffer and retransmit lost packets.
Strict E2E Schemes
Receiver-Based Scheme

[Biaz98Asset]

MH is TCP receiver. Receiver uses heuristics to “guess” cause of packet loss. When receiver believes that packet loss is due to errors, it sends notification to sender. TCP sender, on receiving notification, retransmits lost packet, without reducing congestion window.
Heuristics

Receiver uses inter-arrival time between consecutively received packets to guess cause of packet loss.

On determining a packet loss as being due to errors, the receiver may:
  - Tag corresponding dupacks with an ELN bit, or
  - Send an explicit notification to sender.
Receiver-Based Scheme

Packet loss due to congestion:

Congestion loss
Receiver-Based Scheme

Packet loss due to transmission error:

- FH
- BS
- MH

12 11 10

2 T

Error loss
Sender-Based Discrimination Scheme
Sender-Based Discrimination
[Biaz98ic3n,Biaz99techrep]

Sender can attempt to determine cause of a packet loss.
If packet loss determined to be due to errors, do not reduce congestion window.
Sender can only use statistics based on round-trip times, window sizes, and loss pattern.

Unless network provides more information (example: explicit loss notification)
Heuristics for Congestion Avoidance

Define condition C as a function of congestion window size and observed RTTs.
Condition C evaluated for new RTT. If \( C == \text{True} \), reduce congestion window.
Heuristics for Congestion Avoidance: Some proposals

TCP Vegas [Brakmo94]

expected throughput $ ET = \frac{W(i)}{RTT_{\text{min}}}$

actual throughput $ AT = \frac{W(i)}{RTT(i)}$

Condition $C = (ET - AT > \beta)$
Sender-Based Heuristics

Record latest value evaluated for condition C. When a packet loss is detected:

If last evaluation of C is TRUE, assume packet loss due to congestion.

Else assume packet loss due to transmission errors.

If packet loss determined to be due to errors, do not reduce congestion window.
Sender-Based Heuristics: Disadvantage

Does not work quite well enough!!

Reason

Not much correlation between observed short-term statistics, and onset of congestion.
Sender-Based Heuristics: Advantages

Only sender needs to be modified.

Needs further investigation to develop better heuristics.

Investigate longer-term heuristics.
Reliable Point2Point Transport Layer: Outline

TCP/IP basics.
Impact of transmission errors on TCP performance.
Approaches to improve TCP performance on wireless networks.
Classification.
TCP on cellular.
TCP on MANETs.
TCP in Mobile Ad Hoc Networks
Issues

Route changes due to mobility.
Frequent route changes may cause OOO delivery.

Wireless transmission errors.
Problem compounded due to multiple hops.

MAC
MAC protocol can impact TCP performance.
Throughput over Multi-Hop Wireless Paths [Gerla99]

When contention-based MAC protocol is used, connections over multiple hops are at a disadvantage compared to shorter connections.

They have to contend for wireless access at each hop.

Delay or drop probability increases with number of hops.
Analysis of TCP Performance over MANETs [Holland99]

Impact of mobility.
Simulation study.
Performance metric: throughput.
Baseline: ideal (expected) throughput.
Upper bound.
Static network.
Throughput versus Hops

TCP throughput over 2 Mbps 802.11 MAC, fixed, linear MANET.
Expected Throughput

\[
\text{expected throughput} = \frac{\sum_{i=1}^{\infty} ti \times Ti}{\sum_{i=1}^{\infty} ti}
\]

Ti is measured throughput for i hops using static linear chain topology. 
\(ti\) time duration of TCP connection containing i hops.
Throughput versus speed

Average Throughput (Over 50 runs)

Expected

Actual

Speed (m/s)
Throughput versus Speed

But not always...

20 m/s
30 m/s

Actual throughput

Mobility pattern #
Impact of Mobility

TCP Throughput

Actual throughput vs. Ideal throughput (Kbps) at 2 m/s and 10 m/s.

CMPE 257 Spring 2005
Impact of Mobility

Actual throughput

Ideal throughput

20 m/s

30 m/s
Why Throughput Degrades

mobility causes link breakage, resulting in route failure

Route is repaired

TCP sender starts sending packets again

TCP data and acks en route discarded

No throughput despite route repair

No throughput
Why Throughput Degrades?

- Mobility causes link breakage, resulting in route failure.
- TCP sender times out. Backs off timer.
- Route is repaired.
- TCP sender resumes sending.

TCP data and acks en route discarded.

No throughput despite route repair.

Larger route repair delays especially harmful.
Why Throughput Improves?
Low Speed Scenario

1.5 second route failure

Route from A to D is broken for ~1.5 second.
When TCP sender times out after 1 second, route still broken.
TCP times out after another 2 seconds, and only then resumes.
Why Throughput Improves?

Higher Speed Scenario

Route from A to D is broken for ~ 0.75 second.

Before TCP sender times (after 1 second), route is repaired.
Why Throughput Improves?

**General Principle**

TCP timeout interval somewhat independent of speed.

Network state at higher speed, when timeout occurs, may be more favorable than at lower speed.

Network state:
- Link/route status.
- Route caches.
- Congestion.
How to Improve Throughput

Network feedback. Inform TCP of route failure explicitly. Let TCP know when route is repaired.

Probing. Explicit notification.

Reduce repeated TCP timeouts and backoff.
Explicit Link Failure Notification. Piggyback notification onto DSR’s route failure message to sender. TCP responds by disabling congestion control until route is fixed.

Disable retransmission timers.
When ACK is received, TCP restores state and resumes normal operation.
Performance Improvement

Without network feedback

Actual throughput

Ideal throughput

2 m/s speed

With feedback
Performance Improvement

Without network feedback

With feedback

Ideal throughput
30 m/s speed
Performance with Explicit Notification

![Graph showing throughput as a fraction of ideal vs. mean speed (m/s)]

- **Base TCP**
- **With explicit notification**

Throughput as a fraction of ideal vs. mean speed (m/s)
Issues: Network Feedback

Network knows best (why packets are lost).

- Network feedback beneficial.
  - Need to modify transport & network layer to receive/send feedback

Need mechanisms for information exchange between layers.
Impact of Caching

Route caching has been suggested as a mechanism to reduce route discovery overhead (e.g., DSR).

Each node may cache one or more routes to given destination.

When route from S to D detected as broken, node S may:

- Use another cached route from local cache, or
- Obtain a new route using cached route at another node.
To Cache or Not to Cache

Actual throughput (as fraction of expected throughput)

Average speed (m/s)
Why Performance Degrades With Caching

When a route is broken, route discovery returns cached route from local cache or from nearby node.

Cached routes may also be broken.

- timeout due to route failure
- timeout, cached route is broken
- timeout, second cached route also broken
To Cache or Not to Cache

Caching can result in faster route "repair".

But, faster does not necessarily mean correct.

If incorrect repairs occur often enough, caching performs poorly.

Need mechanisms for determining when cached routes are stale.
Caching and TCP Performance

Caching can reduce overhead of route discovery even if cache accuracy is not very high.

But if cache accuracy is not high enough, gains in routing overhead may be offset by loss of TCP performance due to multiple timeouts.
Window Size After Route Repair

When route breaks: may be too optimistic or may be too conservative. **Better be conservative** than overly optimistic.

Reset window to small value after route repair. TCP needs to be aware of route repair (Route Failure and Route Re-establishment Notifications). Impact low on paths with small delay-bw product.
RTO After Route Repair

If new route longer, RTO may be too small, leading to timeouts.

New RTO = function of old RTO, old route length, and new route length.

   Example: new RTO = old RTO * new route length / old route length
   Not evaluated yet.
TCP Over Different Routing Protocols [Dyer2001]

Impact of routing algorithm on TCP performance.

Metrics: connect time, throughput and overhead.

On-demand routing.

AODV and DSR.

ADV: adaptive on-demand with proactive updates.

Sender-based heuristic to improve TCP’s performance.
Fixed-RTO

TCP does not exponentially backoff the RTO. Uses sender-based heuristic to distinguish between congestion and “route failure” losses.

Route failure assumed if 2 consecutive timeouts. Unack’d packet retransmitted.
No RTO backoff in the second (and +) timeout. RTO remains fixed until retransmission is ack’d.
Improving TCP under OOO Delivery [Wang02]
Out-of-Order Packet Delivery

Route changes may result in out-of-order (OOO) delivery.

Significantly OOO delivery confuses TCP, triggering congestion control.

Potential solutions:

Avoid OOO delivery by ordering packets before delivering to IP layer

Turn off fast retransmit.

Can result in poor performance in presence of congestion
TCP DOOR

Detect and respond to out-of-order (OOO) packets.

Differentiate between OOO and congestion losses.

OOO delivery caused by:

- Retransmissions.
- Route changes.
Detecting OOO

OOO delivery can happen in either direction.
Sender detects OOO (duplicate) ACKs.
Receiver detects OOO data packets.
ACKs

Sequence number of packet being ACKed: monotonically increasing.

Why? ACKs are not re-transmitted.

For DUPACKs, add 1-byte to count DUPACKs.

ADSN: ACK duplication sequence number.
TCP header option.
Each DUPACK carries different ADSN.
OOO Data Packets

At receiver.
Why comparing sequence numbers doesn’t work?
  Retransmissions: higher sequence #’s can arrive earlier.
  Out-of-sequence event.
Use extra sequence number: incremented with every data packet, including retransmissions.
  2-byte TCP packet sequence number (TPSN) as TCP option.
  Or timestamp.
Sender needs to be notified.
OOO Response

At sender.

2 types of response:

- Temporarily disable congestion control for fixed time interval $T_1$.
- If in congestion avoidance mode in the last $T_2$ time interval, go back to prior state.
Evaluation

Simulation environment:

ns-2 + CMU extensions.

Mobility: random way-point.

Workload: single TCP between fixed S and R with and without congestion.
Results

Significant goodput improvement (~50%) when 2 response mechanisms used.

Sender versus receiver detection.
  Seem to perform the same.
  Correlation between OOO ACKs and data.

Response mechanisms.
  Both in place show better performance.
DSR Caching

With DSR caching enabled, lower performance improvements. Claim TCP performance was better than when caching was off. Why?
Reliable Multicast in MANETs
Motivation

Group communication services as important application class for key MANET scenarios: special operations, emergency response, etc.

   E.g., teleconferencing and data dissemination.

Multicast: efficient way of delivering many-to-many data.
Challenges

Achieve reliability in the presence of constant mobility, route changes, transmission errors over multi-hop wireless routes.

MANETs are very sensitive to load and congestion.

- Contention-based MACs.
- Hidden terminal problems.

Not much has been done...
Reliable Broadcast

[Pagani1997]

Special case: all nodes are receivers.
Reliable broadcast: all nodes deliver same set of messages to application.
“Exactly-once” semantics.
Assumes underlying clustering algorithm.
Primitives

\[ \text{rbcast (msg, group-id)} \]

Caller blocks until message received by clusterhead.

\[ \text{deliver (msg)} \]

Notification of message reception by all other nodes.
Entities
Protocol

Sender unicasts message to its clusterhead. Clusterhead broadcasts message within cluster and waits for ACKs. Nodes reply with ACK. Gateways forward message to directly connected clusters (through clusterheads). Gateway delays its ACK until it gets ACK from corresponding clusterheads.
Protocol (Cont’d)

If same message received over multiple paths: clusterhead selects one; piggybacks the sender id in the message and broadcasts. Non-selected gateway receives the broadcast and records it’s the leaf for that message. Prevents loops, allows multi-path, reduces duplicates. Reverse path is recorded: ACKs flow back.
Failures and Mobility

Timeouts and retransmissions.
Recovery by clusterheads & gateways.
Summary

2 phases:

Diffusion: message diffused from source to destinations.

Gathering: source collects ACKs.

On-demand forwarding tree rooted at source. If “virtual” links break, flooding is used. Clusterheads buffer messages until ACK comes back.

Reliability guaranteed as long as “liveness” property holds.

Topology is stable enough.
Issues

Larger networks?
  High delays.
  Target smaller groups (10->100???).

What happens when clusterheads and gateways fail/move?
  Satisfying “liveness” is tricky.

Heavy duty protocol.
  State at nodes.
  ACKs for reliability.
Anonymous Gossip
[Chandra2001]

Approach:
Use of underlying “best-effort” multicast routing mechanism.
Repair losses through gossip-based propagation.
Gossip propagation is well-known!
Examples?
Gossip Round

Node A randomly chooses node B. A and B exchange information on messages they have and don’t have. A and B exchange missing messages.
Anonymity

No need for membership information.

gossip-message and gossip-reply.

Node selects random neighbor and sends gossip-message.

If node is not group member, selects neighbor and forwards; if member, decides to accept or forward.

If accepts, unicasts gossip-reply back.
Locality

Choosing closer-by or farther members.

Why is this an issue?

Choose near members with higher probability.

Need to keep extra state: nearest-member.

Extra overhead to maintain this information.

Dependence on the underlying routing protocol!
Cached Gossip

Gossip with known members. member-cache keeps information on known members.

Updated when messages are received (data, gossip-reply, RREQ, etc.).

Node uses AG with probability $p_{anon}$; otherwise cached gossip.
Performance Evaluation

Simulations using GloMoSim.
M-AODV.
Single source.
Random way-point.
Parameters: range, number of nodes, and maximum node speed.
Results

Improves packet delivery ratio.
Overhead?
Delay?
Comparison with flooding?
**CALM [Tang 2002]**

Like AG, e2e approach.

Initial study comparing SRM to UDP showed SRM performs badly in MANETs.

Why?

- SRM is heavy duty.
- No congestion control!
Impact of Congestion Control on Reliability?

CALM uses “rate-based” CC.

Data is sent at application rate.

If congestion (NACK), sender clocks sending rate based on receivers experiencing congestion.

Congested receivers kept in receiver-list.
Source multicasts next data packet to selected receiver in receiver-list.
Source explicitly requests problematic receiver to ACK.
If ACK received before time-out, receiver removed from receiver-list.
When receiver-list empty, source reverts back to nominal application sending rate.
Feedback is unicast to the source.

Generated after N consecutive packets are missing.
Evaluation

Simulations using QualNet. Comparison between CALM, SRM, and (multicast) UDP. ODMRP. Metrics: packet delivery, overhead, delay, goodput.
Results

CALM outperforms SRM.
UDP performs surprisingly well, except under high traffic loads.

Proves need for congestion control.
SRM’s main problem is extra load caused by its control packets.
Congestion control helps but still need reliability.
RALM [Tang03]

Reliability + congestion control.
RALM Overview

Source initially sends at application’s nominal rate.
If NACK, source enters “loss recovery”.
  Receivers experiencing losses added to “receiver list”.
  Source selects receiver from “receiver list” to reliably transmit: “feedback receiver”.
  Source multicast(s) lost packet(s) to feedback receiver.
    Feedback receiver unicasts ACKs/NACKs.
    Only feedback receiver sends feedback.
RALM Overview (cont’d…)

During loss recovery:

Source retransmits lost packets one at a time: stop-and-wait. Why?

When receiver list is empty, source goes back to application sending rate.
Results

From paper...
ReACT [Rajendran04]

Source-based + local recovery.

Receiver-based loss differentiation.
  Distinguish between “global congestion” and local losses.
  Use cross-layer information.
  Multicast receiver samples its MAC queue.
  Intermediate nodes also report if their MAC queue grows above certain threshold.

Local recovery: uses non-congested nodes.
ReACT: Source-based control

Rate-based CC.
Initial rate establishment and congestion control modes.
At initial rate establishment, source decides the initial sending rate.

- Probes directly connected members.
- Uses inverse of longest measured RTT to set source’s initial rate.
- Rate periodically updated based on feedback from directly connected receivers.
ReACT: Source-based control (cont’d…)

Congestion control mode:
Similar to RALM.
ReACT: Receiver-based recovery

Detect losses that can be recovered locally. Receivers need to know about other local group members.

For local recovery, node chooses non-congested member with highest receive rate, lowest hop count and latest timestamp.

Source route to recovery node.
Route, hop count and congestion information are updated as packet is forwarded.
ReACT: Receiver-based recovery

NACK with missing sequence numbers sent to recovery node if losses are local. Otherwise, NACK is forwarded to source.

Losses are global if all paths to recovery nodes are congested or node’s local queue is getting full.
Evaluation

Simulations.
Comparison against RALM.
Metrics: reliability, (reliable) throughput, overhead.
Effect of congestion and mobility.
ReACT achieves better reliability and goodput with lower overhead.

Local recovery prevents source from reducing rate unnecessarily.

Local recovery also lowers overhead.