Selective-TCP for Wired/Wireless Networks

Rajashree Paul
rpaul2@cs.sfu.ca

Communication Networks Laboratory
Roadmap

- Motivation
- Background and related work
- Selective-TCP
  - overview
  - implementation
- Simulation results
- TCP packet control
- Performance evaluation: comparison of Selective-TCP and TCP packet control
- Conclusions
Motivation

- TCP is a transport protocol extensively used over the Internet: 95% of IP traffic in 2002
- TCP performance degrades in wireless networks and thus, also over mixed wired-wireless networks (cellular networks)
- Main reason for TCP’s poor performance in wireless networks is TCP’s inability to distinguish losses due to wireless link errors from losses due to congestion
Roadmap

- Motivation
- Background and related work
- Selective-TCP
  - overview
  - implementation
- Simulation results
- TCP packet control
- Comparison of Selective-TCP with TCP packet control
- Conclusions
Background

Three approaches:

- end-to-end: TCP Reno, TCP NewReno
- split connection: I-TCP, M-TCP
- link layer level: Snoop, TCP packet-control

End-to-end schemes are promising:
- significant performance gain without help from intermediate nodes: though not as effective as link layer based approaches in handling wireless losses

Related work

- Improving TCP performance in wired/wireless network by detecting the type of loss:
  - End-to-end:

Related work

- Improving TCP performance over wired-wireless network by detecting the type of loss:

- Link layer level:


Roadmap

- Motivation
- Background and related work
- Selective-TCP
  - overview
  - implementation
- Simulation results
- TCP packet control
- Performance evaluation: comparison of Selective-TCP and TCP packet control
- Conclusions
Selective-TCP algorithm: overview

- An end-to-end solution
- Implemented in ns version 2.27
- Type of loss detected at the receiver
- Corrective measures:
  - **wireless loss:** Selective Negative Acknowledgement (SNACK) sent to the sender
  - **congestion loss:** sender’s congestion window size set according to the bandwidth measured at the receiver
  - \[ \text{bandwidth} = \frac{(\text{no. of lost packets} \times \text{size of packets in bits})}{(\text{inter-arrival time between last in-sequence packet received and most recent packet} \times 1,000)} \text{ kbps} \]


Loss detection scheme

- Detects the type of loss based on the packet inter-arrival times at the receiver
- Assumes the following:
  - wireless link is the bottleneck
  - sender performs bulk data transfer
  - on the connection path, only the last link is wireless
- Given the assumptions, accuracy of detection is high

Loss detection scheme

- Wireless link is the bottleneck: packets will queue at the base station

- Thus, for a single packet loss:
  - If \( T < \text{inter-arrival time} \leq 2T \) \{ wireless loss \}
  - Else \{ congestion loss \}
TCP-SNACK

- An option of satellite communication protocol stack-transport protocol (SCPS-TP)
- Widely used for satellite links
- Negative: receiver informs sender about the segments not received
- Selective: can send information for multiple lost segments:
  - good for long delay networks
- When a sender receives SNACK:
  - aggressively retransmits packets indicated as lost, preventing unnecessary retransmission time-outs

Selective-TCP: implementation

- Extensions made in:
  - tcp-newreno.cc
  - tcp-newreno.h
  - tcp-sink.cc
  - tcp.h
- Location: ns-allinone-2.27/ns-2.27/tcp

- Scenarios generated:
  - TCL script files
if (out-of-order packet received) {
    // check type of loss
    if (wireless loss) {
        if (snack_delay = 0)
            send SNACK
        else
            do nothing
    } else {
        // congestion loss
        1) set congestion_count = congestion_count + 1
        2) set congestion_info = current bandwidth measured at the TCP receiver
        if (congestion_count = k) {
            1) send congestion_info to the TCP sender
            2) reset congestion_count
        } else
            send ACK // as in the case of standard TCP sink
    }
} else // in-sequence packet received
    send ACK (same as standard TCP-sink)
Selective-TCP: module at receiver

- tcp-sink.cc:

```c
void SnackSink::recv(Packet* pkt, Handler*)
{
    // code inserted by Rajashree
    prev_pkt_ts = present_pkt_ts;
    present_pkt_ts = Scheduler::instance().clock();
    tmin = present_pkt_ts - prev_pkt_ts;
    // T_min is the minimum packet inter-arrival time seen so far
    if (T_min > tmin)
        T_min = tmin;

    ................
}
```
Pseudo-code of Selective-TCP at the TCP sender

```java
if (SNACK received) {
  1) retransmit lost packet(s) as indicated in SNACK
  2) reset retransmission timer
}
else if (congestion_info ≠ 0) {
  // set size of congestion window as the bandwidth measured at receiver
  1) set cwnd_ = congestion_info/base_rtt
  // cwnd_ denotes congestion window size and
  // base_rtt is the initial round trip time
  2) reset congestion_info
}
else //standard ACK received
  do as standard TCP NewReno sender
```
Selective-TCP: module at sender

- tcp-newreno.cc:

```cpp
void NewRenoTcpAgent::recv(Packet *pkt, Handler*)
{

...............  // Sending Snack
    if (tcph->snack_option())
        processSnack(pkt);

    else if (tcph->congestion_info()) // Setting congestion window size
    {
        seq_num = tcph->seqno();
        cwnd_ = tcph->congestion_info()/base_rtt;
        output(t_seqno_++,0);
        Packet::free(pkt);
    }

...............  // Setting congestion window size

}
```
Select packet reordering and increased goodput

- In case of wireless loss, the receiver sends SNACK
  - sender retransmits the lost packets immediately without waiting for retransmission timer to expire:
    - no duplicate ACKs sent
    - congestion control not invoked
  - SNACK is selective
    - helps in presence of packet reordering
  - result: increased bandwidth utilization and higher goodput
Selective-TCP algorithm: description

- In case of congestion loss:
  - sender’s congestion window size is set according to the bandwidth measured at receiver
  - TCP’s AIMD algorithm is prevented from setting congestion window lower than necessary
  - result: increased bandwidth utilization and higher goodput
Roadmap

- Motivation
- Background and related work
- Selective-TCP
  - overview
  - implementation
- Simulation results
- TCP packet control
- Performance evaluation: comparison of Selective-TCP and TCP packet control
- Conclusions
Simulation scenario: network topology

- Wired link: bandwidth 2–4 Mb, propagation delay 1 ms
- Wireless link: bandwidth 1 Mb, propagation delay 5 ms

TCP source:
sending rate = 2 mbps

UDP source:
sending rate = 512 Kbps
Simulation scenario: error model

- Two-state Markov model for modeling burst errors over wireless link

\[
\pi = \begin{bmatrix} p & 1-p \\ 1-q & q \end{bmatrix}
\]

and

\[
\varepsilon = \frac{1-p}{2-p-q}
\]

Simulations:

good state: 0 packet loss
bad state: 1 packet loss

p = 0.9913 and q = 0.8509

Simulation results:

In the presence of congested link:

[Graphs showing simulation results for Goodput and Size of congestion window for Selective-TCP and TCP NewReno.]
Simulation results:

In the presence of congested link:
Simulation results:

In the absence of congested link:

![Graphs showing simulation results for Goodput (maximum sequence number) and Size of congestion window over simulation time.](image)
Simulation results:

In the absence of congested link:
Roadmap

- Motivation
- Background and related work
- Selective-TCP
  - overview
  - implementation
- Simulation results
- TCP packet control
- Performance evaluation: comparison of Selective-TCP and TCP packet control
- Conclusions
TCP packet control: overview

- A link layer based approach to improve TCP performance in wired/wireless networks:
  - modifications made in the intermediate node (base station): implemented in ns version 2.26
  - addresses two problems specific to wireless networks
    - delay variation (spurious fast retransmit)
    - sudden large delays (spurious fast timeout)
  - hides wireless losses from TCP sender (fixed host)
  - filtering done locally at base station
  - improves goodput up to 30% with 1% packet loss, compared with TCP-Reno

TCP packet control: implementation

- Data and ACK filters:
  - ll-wz.cc
  - in ns-allinone-2.26/ns-2.26/mac
Packet control algorithm: data filter

- **Data filter**: filters data segments, sent from fixed host

  ```
  if (new or unacknowledged data segment)
      forward to receiver (mobile host)
  else // acknowledged data segment
      drop the segment
  ```
Packet control algorithm: ACK filter

- **ACK filter**: filters ACKs, sent from mobile host

  ```
  if (old ACK received)
      drop the ACK
  else if (new ACK received) {
      i) update last_received_ACK
      ii) reset number of DUP_ACKs to 0
      iii) forward the ACK to fixed host
  }
  else //duplicate ACK received {
      i) update number of DUP_ACKs
      ii) drop or forward the duplicate ACKs depending on user-defined DUPACK_threshold
  }
  ```
TCP-packet control: data filter

void LLWz::myRecv(Packet *p, Handler *h, bool bDelayedPacket)

if(enableAckControl_ != 0)
{
    //Data Filter
    if(tcph->seqno() <= m_iLastWirelessAck)
    {
        //we drop every second retransmitted packet.
        if(recordRetransPacket(tcph->seqno()) % 2 == 1)
        {
            m_iNumOfWiredDataPacketDropped++;
            Packet::free(p);
            return;
        }
    }
}
Roadmap

- Motivation
- Background and related work
- Selective-TCP
  - overview
  - implementation
- Simulation results
- TCP packet control
- Performance evaluation: comparison of Selective-TCP and TCP packet control
- Conclusions
Simulation results:

In the presence of congested link:

![Graphs showing simulation results for Selective-TCP, TCP NewReno, and TCP packet control.](image)

- Selective-TCP: Blue line
- TCP NewReno: Green line
- TCP packet control: Red line
Simulation results:

In the absence of congested link:
Roadmap

- Motivation
- Background and related work
- Selective-TCP
  - overview
  - implementation
- Simulation results
- TCP packet control
- Performance evaluation: comparison of Selective-TCP and TCP packet control
- Conclusions
Conclusions

Selective-TCP:
- an end-to-end approach
- distinguishes wireless losses from congestion losses
- takes corrective action depending on the type of loss
- improves goodput up to 45% in mixed wired/wireless networks in presence of 5% burst errors, compared with TCP NewReno
References


References


References


