Alexandria University
Faculty of Engineering
Computer and Systems Engineering Dept.
Fourth Year
Spring 2012

Wireless Networks and Mobile Computing
Assigned:30th April 2012
Due:19th May 2012

Split TCP in Ad hoc Networks

Description

In this project, you will implement Split TCP protocol for wireless ad hoc networks to improve the performance of TCP in these networks, using Java in Windows environment.

Details

The major responsibilities of TCP in an active session are to:

- provide reliable in-order transport of data: to not allow losses of data.
- control congestions in the networks: to not allow degradation of the network performance.
- control a packet flow between the transmitter and the receiver: to not exceed the receiver's capacity.

In general, we distinguish between the following operational phases in TCP:

- slow-start phase (also known as exponential start);
- congestion avoidance phase;
- congestion control phase;
- fast retransmit phase;
- fast recovery.

TCP Tahoe congestion control

Every transmission starts with connection setup and followed by slow start phase:

- the sender starts the session with a congestion window set to maximum segment size (MSS):
  - it sends MSS bytes of data;
  - starts retransmission timeout (RTO) and waits for acknowledgement packet (ACK).
- if ACK is received in RTO the congestion window is doubled and two MSSs of data are sent;
- the congestions window is doubled with every ACK until it reaches slow-start threshold;

The slow-start phase is followed by congestion avoidance phase:

- when the slow-start threshold is reached, the congestion window grows linearly (AI);

- if the ACKs are received before timers (RTOs) expire:
    - the congestion window grows until the receiver window advertised in connection setup;
    - (lastByteSent - lastByteAcked) <= rcvWin to NOT allow overflow!

If the ACK is not received in RTO, TCP assumes the packets is lost (congestion):

- TCP enters the congestion control phase performing the following:
    - reduces the slow-start threshold to 1=2 of current CW (MD);
    - resets the congestion window to one MSS;
    - activates the slow-start algorithm and resets the timeout.

Why does not TCP work well in ad-hoc networks?

One of the main reasons for the poor performance of TCP in ad hoc networks is the misinterpretation of packet loss: TCP attributes packet loss to congestion while in ad hoc networks, high packet loss percentage occurs due to link failures (as a result of mobility).

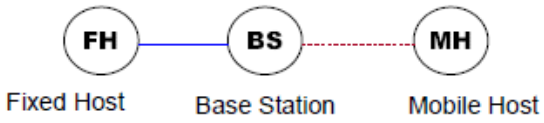
Split-TCP provides the solution by splitting the TCP functionalities into two aims:

- congestion control;
- end-to-end reliability.

End-to-end TCP connection is broken into one connection on the wired part of the route and one over wireless part of the route.

A single TCP connection split into two TCP connections:

- FH-MH = FH-BS + BS-MH
- "Acks" are intercepted and managed at proxy node (BS):



Generally, the protocol works as follows:

- Fixed Host(FH) sends a packet to Base Station(BS) specifying that it should be destined to Mobile Host (MH).
- Once the packet makes it to the base station, it has traversed the previous “zone” and thus, we avoid having to go all the way back to the source if the packet needs to be retransmitted.
- BS buffers the packet, acknowledges its receipt to the FH by sending a local acknowledgement (LACK), and takes over the responsibility of delivering the packet to the MH.
- When the packet reaches the MH, it sends two acknowledgments:

- a local acknowledgement (LACK) to the base station (BS).
- an end-to-end acknowledgement (ACK) to the fixed host (FH). The fixed host does not clear the packet from its buffer until it has received the ACK from the MH.

- So, in conclusion, the fixed host should receive two ACKs: a local one from the BS and an end-to-end one from the MH.

Specifications

You are required to implement mobile TCP services over UDP/IP protocol using Java.

1) Fixed Host Sender (FH)

- Implement a standard TCP sender with stop-and-wait reliability and Tahoe congestion control.
- FH has **two** transmission windows: the congestion window which depends on the ACKs received from the BS, and the end-to-end window which depends on the ACKs received from the MH destination.
- FH does not remove a packet from its buffer except after it receives an end-to-end ACK from the MH destination.

2) Base Station/Proxy Node (BS)

- It buffers the sent packet,
- sends acknowledgement to the fixed host before it sends the packet to the mobile host
- and takes over the responsibility of delivering the packets to the mobile host.

3) Mobile Receiver (MH)

- Implement a standard TCP receiver(sink) which when receives a packet, sends:
  - a local acknowledgement to **BS**.
  - an end-to-end acknowledgement to **FH**.
- End-to-end ACKs should be infrequent (one ACK for every 100 packets that are received by the destination).

4) Packet Loss Simulation

- your implementation should provide two types of packet loss errors:
  - loss due to congestion (implemented in the send() method of the FH)
  - loss due to wireless link failure (implemented in the send() method in the BS)
- For both packet loss types, your implementation should allow the user to set a packet loss probability (PLP) which ranges from 0 to 1. For PLP=0 your implementation should behave with no packet loss. For non-zero values of PLP, you should simulate packet loss (by dropping datagrams given as parameters to the send() method) with the corresponding probability - i.e. a datagram given as a parameter to the send() method is transmitted only 100*(1 - PLP)% of the time.

5) **Results and Statistics Graphs**

- Experiments: a series of transfers of large files (e.g.: 4 MBytes, or higher). Your test runs should be performed with at least the following PLP values: 1%, 5%, 10% and 30% for each type of packet loss error. For each PLP value, you should report the average of at least 5 (consecutive) runs.
- Compare Split TCP with simple TCP by measuring the cumulative throughput which is defined as the ratio of the total number of data bytes transported to the destination over the duration of the session and plot a graph showing the results.
- Plot a graph showing the changes in the congestion window size over the duration of the session in case of Split TCP and simple TCP.

Notes

- Develop this assignment using Java programming language.
- You should deliver your source code.
- You should deliver a report explaining your design, implementation, results and statistics graphs.
- Project is due 19th May 2012.

Grading Policies

- No Late submission is allowed.
- You should work in groups of 4 students.
- Delivering a copy will be severely penalized for both parties, so delivering nothing is so much better than delivering a copy.

References

[1] Split TCP for Mobile Ad Hoc Networks: http://www.cs.ucr.edu/~krish/splittcp.pdf