Iowa State University
Department of Electrical and Computer Engineering
Cpr E 489: Computer Networking and Data Communications
Lab Experiment #9
TCP Congestion Control

Objective

To write two programs to implement the TCP Tahoe congestion control mechanism, one program for the Sender, and another for the Receiver.

Pre-Lab

Go over how to read files into a program and write data into a file using C. Familiarize yourself with the three major techniques that are used in TCP Tahoe: Slow Start, Congestion Avoidance, and Fast Retransmit.

Lab Expectations

Work through the lab and let the TA know if you have any questions. After the lab, write up a lab report with your partner. Be sure to

1) summarize what you learned in a few paragraphs
2) include your answer to the exercise
3) specify the effort levels of each group member (totaling to 100%)

Your lab report is due one week from the assign date. Be sure to submit your well-commented code with your lab report for grading. Also, please demonstrate your program to the TA at 2048 Coover Hall during any one of the following three time slots:

- 04/24 Tuesday 4:10 – 6:00 PM
- 04/25 Wednesday 1:10 – 3:00 PM
- 04/27 Friday 1:10 – 3:00 PM

Problem Description

In this lab experiment you are required to design and develop two programs to implement the TCP Tahoe congestion control mechanism: one program for the Sender and the other for the Receiver. You will implement the three major techniques that are used in TCP Tahoe: Slow Start, Congestion Avoidance, and Fast Retransmit.

To simulate network congestion, please use the provided `AddCongestion.c` routine to introduce errors to the transmitted packets. For CRC generation and error detection, please use the provided `ccitt16.o` object file.

Sender Program

Write a program to implement the Sender, which shall

- Establish a TCP connection to the Receiver and send packets over that connection.
- Read the provided `input.txt` file into the memory as an array of *char* and send the content to the Receiver.
- Form each packet according to the following format:

Sequence Number	Data	CRC
2 bytes	2 bytes	2 bytes

- **Sequence Number**
  - Byte index, starting from 1000.
- **Data**
  - Two characters (sent from Sender to Receiver)
  - No data is sent from Receiver to Sender

- **CRC**
    - CRC generated for this entire packet, including Sequence Number and Data fields.

- Apply the `AddCongestion.c` routine to the entire packet. It has two arguments:
    - a null terminated version of the packet above
    - a bit error rate (BER), between 0.00001 and 1

  This step introduces random errors to the packet (with the provided BER), which simulates congestion in the network.
- Implement an RTO timer, with RTO = 3 seconds.
- Implement the congestion window, with MSS = 2 bytes. For this lab, assume that the advertised receiver window (rwnd) is always larger than the congestion window (cwnd).
- Implement the TCP Tahoe congestion control mechanism:
    - Slow Start: start with cwnd = 1 (MSS), and increase it by one when a non-duplicate ACK is received.
    - Congestion Avoidance (with the initial ssthresh = 16 MSS): increase cwnd by one only after the sender has received cwnd non-duplicate ACKs.
    - Fast Retransmit: retransmit the packet when three duplicate ACKs are received.
- Record cwnd after each RTT.

Receiver Program

Write a program to implement the Receiver, which shall

- Accept connections from the Sender.
- Accept data packets from the Sender.
- Run the CRC check:
    - If the packet is received error free, then **delay for 1 second** and sends back an ACK with the following format:

Acknowledgment Number
2 bytes

    - If the packet is received in error, do nothing.
    - ACK packets are not corrupted.

Procedure

- Write the Sender and Receiver programs as described above.
- Transmit the provided `input.txt` file from the Sender to the Receiver and record cwnd after each RTT.
- Plot the *congestion window size* vs. *RTT* curve with the X-axis ranging from 0 to 40 RTTs.
- Submit your figure and code to the TA for grading.

Exercises

1) Run your program with each of the following BER values: 0.0001, 0.001, 0.005, 0.01, and plot the *congestion window size* vs. *RTT* curves for each BER value.

Compiling

An object file, `ccitt16.o`, is provided that will generate and check CRC for you. In order to use the "`.o`" file, you first need to include `ccitt16.h` with your file (e.g., `sender.c`):

```
#include "ccitt16.h"
```

Now, compile your file with the **`-c`** option to generate a "`.o`" file:

```
gcc -c sender.c
```

Finally, compile both "`.o`" files together to create an executable:

```
gcc -o sender sender.o ccitt16.o
```

Usage

The function provided by `ccitt16.o` has the following prototype:

```
short int calculate_CCITT16(unsigned char cData[], unsigned int iLen,
      unsigned int iAction);
```

`iAction` is defined as either `GENERATE_CRC` or `CHECK_CRC` in the `ccitt16.h` header file:

`#define GENERATE_CRC 1`

- Returns the checksum of `cData[]` with length `iLen` as a short `int`

`#define CHECK_CRC 2`

- Uses the last two bytes of `cData[]` as CRC check bits to check `cData[]`; returns either 0 or 1:
    - `#define CRC_CHECK_SUCCESSFUL 0`
    - `#define CRC_CHECK_FAILURE 1`