CSE123 Discussion 1

Project 1 Overview
Chenyu
Thursday, 3:30 – 5:30PM, CSE B240A
Agenda

• Project setup
• Get the project skeleton code from Github:
  • https://classroom.github.com/a/nz_MBYn1
• Project 1 introduction and overview
Project overview

• Grading is done on Linux machine
• Make sure your program compiles and runs on Linux
• Logging into ieng6, contact ACMS: acms-help.ucsd.edu if you have trouble accessing lab machines
• Adding SSH key to GitHub(optional)
• https://help.github.com/articles/adding-a-new-ssh-key-to-your-github-account/
Get project skeleton code

• Signup for a GitHub account if you don’t have one
• Go to https://classroom.github.com/a/nz_MBYn1, accept the invitation, and follow instructions
• Once created your own repo, project skeleton code will be available at https://github.com/ucsd-cse123-fa17
• ‘git clone your_project1_url’ in the terminal
• ‘git add file_to_commit’, ‘git commit –m ‘commit message’”, ‘git push’, etc to make commit to your repo
• For tutorials on git: https://try.github.io/levels/1/challenges/1
• Or use GitHub’s GUI app (if you absolutely hate using git in terminal)
Code instructions

• Put names, PID, email in README.MD

• YOU MUST COMPILE AND RUN YOUR CODE ON IENG6 MACHINES!!!

• Only C/C++ code will be accepted

• Developing and testing on those machines to avoid last minute compatibility agonies
Project 1 – Sliding Window

• Ensure reliable communication on packet switched networks where packets can get
  • Dropped
  • Delayed
  • Corrupted
Framing

- Break up large messages into smaller pieces called **frames** with additional metadata
- e.g. we want to send message “helloworld” from sender with id:0 to receiver with id:1 and the frame payload size is 5. Then we need to break down the message and send it by 2 frames, frame 1 contains “hello”, frame 2 contains “world”
SWP Components

• Flow control
  • Sender and receiver buffer
    • Sender limits number of in flight messages relative to buffer size
    • Let’s say the buffer size is 4 and the number of messages is 8. We can only send 4 messages at a time.
    • See sendWindowNotFull on page 111-112 of P&D
  • Frame ordering
    • SWP ensures that frames are delivered in order to the receiver application
SWP Components

• Reliable communication
  • Detecting missing frames
  • Detecting corrupted frames
    • “hello” can be corrupted to become “hell”
    • CRC computation is used to detect frame corruption
      • CRC will be covered in lecture
• Acknowledgement from receiver to sender
• Retransmission in case of loss from sender to receiver (timeout)
File overview

- **main.c**: responsible for handling command line options and initializing data structure

- **common.h**: houses commonly used data structures among various source files

- **communicate.c**: Takes care of transporting messages between the sender and receiver threads. Don’t modify

- **input.c**: Responsible for handling messages inputted by the user (e.g. msg 0 0 0 hello world). Don’t modify.
File overview

- **util.c**: Contains utility functions, namely, all of those for the provided linked list implementation. **common.h**: houses commonly used data structures among various source files.

- The majority of your work should be here
  - **sender.c**: Contains the skeleton code for the sender threads
  - **receiver.c**: Contains the skeleton code for the receiver threads.
Skeleton code – common.h

• Contains struct and constant definitions that are across different system components
• Cmd structs are populated by input thread
• Receiver struct contains state for the receiver -- Add SWP receiver state
• Sender struct contains state for the sender -- Add SWP sender state
• Frame struct contains data -- modify to contain the header and CRC footer Don’t forget to modify FRAME_PAYLOAD to be 48! You can use at most 16 bytes for the combined header and footer. A good place to start is to read P&D chapter 2 up through section 2.5.
Skeleton code – util.h, util.c

• Linked list implementation functions operate on head node don’t modify
• Comparing two struct timeval objects
• Functions to serialize a frame to a char[64] and back to a frame -- implement these! Suggestion: Write frame printing function and test that composition of conversion functions is equal to the original input, i.e. f (f^-1(x)) = x and f^-1(f(y)) = y.
Skeleton code – sender.c, sender.h

• Majority of code base for sender thread
• Perform 4 major steps in loop
  • Receive and process commands from input thread
  • Transmit / buffer messages
  • Process incoming acknowledgements
  • Retransmit timed out frames
Skeleton code – sender.c, sender.h

• Need to update or fill in the following:
• init_sender() -- fill in with Sender sender’s initialization
• sender_get_next_expiring_timeval() -- return timeval* for next buffered message to expire
• handle_incoming_acks() -- update SWP sender state
• handle_input_cmds -- split long input messages in to multiple frames
• handle_timedout_frames -- retransmit timed out frames
Skeleton code – receiver.c, receiver.h

• Implements receiver thread
• Receive and acknowledge incoming message from senders
Skeleton code – receiver.c, receiver.h

• Need to update the following: handle_incoming_messages()
• We recommend following the steps suggested in the TODO
General Advice

• Following the steps outlined in the project description in order will help you.
• Trying to implement the whole thing in one go is likely fraught with peril.
• Get started early. This project is often both labor and details-oriented.
• Be aware that the code in P & D is incomplete. Copying it verbatim without understanding SWP is unwise.
Submission instructions

• Prepare a design document explaining
  • Important data structures
  • Frame structure
  • Algorithms used
  • Anything you want to highlight to the graders
  • Be sure to proofread it before submitting Ask yourself, could someone else understand what I did by reading this report?

• Put name, PID and email in README
Next time

• Troubleshooting SWP Sequence number wrap around
• Acks that are not in the sender’s window
• Handling timeouts in the presence of struct timeval overflow in tv_usec
• Other common pitfalls
• CRC codes and modulo 2 arithmetic SWP and CRC will be covered in lecture
Reading

• Textbook, chapter 2.5