

CS 415

Operating Systems Practicum

Project 4: Reliable Networking

Due: April 19, 23:59

Oliver Kennedy

okennedy@cs

So far...

- You can send messages from one computer to another.
 - But those messages can get lost.
- Can we do better?
 - We'll build a simplified TCP.
 - Add sequence numbers and acks.

Overview

- There are multiple approaches.
 - Extend miniports to support error cases.
 - Have messages sent to one miniport and processed by a background thread.
 - Make one miniport a control socket.
- Your interface consists of five functions defined in minisocket.c/.h

The functions

- minisocket_server_create()
 - Waits for a connection to start
- minisocket_client_create()
 - Starts a connection with a listener
- minisocket_send()/minisocket_receive()
 - Send/receive data
- minisocket_close()
 - Finish a connection

Sending Packets

- How do you detect transmission errors?
 - Acknowledge each packet
 - Assign each packet a sequence number
- If a packet is dropped, you should re-send it.
- If a packet drops enough times, throw an error.
- Use timers + semaphores for timeouts.

Sending Packets (cont)

- Attempt to send 7 times.
 - Start the timeout at 100 ms.
 - Double the timeout after each attempt.
(resetting after a success)
- Send should block until the packet is acknowledged.
- If a packet is too big, break it up.

Creating a Socket

- `minisocket_server_create()`
 - Block until a create message arrives.
 - Send a create acknowledgement...
 - ... and as with any data packet, wait for an ack.

Creating a Socket

- minisocket_client_create()
 - Send a create command...
 - ... while waiting for a create ack
 - Acknowledge the acknowledgment.

Buffering Data

- As data is received, it should be saved.
- When you read, you should save any unread data in a packet.
 - Use a ring buffer.
 - Use a queue, “shortening” the packet at the head of the queue.
 - Use the miniport queue and a temporary buffer.

Port for each Socket

- If you use this approach, you will need to extend miniports to handle error cases.
 - Packet receipt on an inactive port.
 - Per-packet preprocessor to acknowledge packet receipt.

De-multiplexing

- Yet another background thread.
 - Call `miniport_receive()` repeatedly.
 - Figure out what kind of message it is.
 - Figure out what minisocket it's going to.
 - Make sure the packet isn't a duplicate.
 - Deal with it (or send back an error).

Things to keep in mind

- Don't forget to destroy the remote port returned by `receive()` (unless you need it).
- Make sure `deregister_alarm()` is working.
- ALL your minisockets code must be threadsafe. All of it should be doable with semaphores/locks.
- You may want to create a `minisocket_initialize()`.
- There are other ways of doing this project.

Good Luck