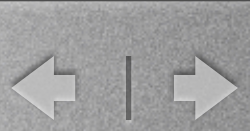



CS 415: Operating Systems Practicum

Project 4- Implementing UDP

Oliver Kennedy
okennedy@cs.cornell.edu

Goals

- Learn about...
  - Network interface multiplexing.
  - The UDP part of UDP/IP
- Build a simple one-hop ghetto-UDP implementation.


The network card

- The underlying interface gives you a way to...
  - identify an endpoint (network_address_t).
  - sending data to a particular host.
    - network_send_pkt()
  - being alerted when new data arrives.
    - network_initialize(handler);
- All defined in network.h


What's missing?

- We're multiplexing the processor, we also need to multiplex the network card.
- The network card sends and receives messages.
  - Create multiple virtual network cards.
  - Call them "ports".
  - A port can send messages to another port.
    - ... on the same machine or another


What goes into a send?

- Sending program calls minimsg_send()
- The sender tags the data with a destination and a return address and inserts it into the network.
- The recipient receives the data and stores it for consumption by a waiting thread.
- The recipient thread calls minimsg_receive()


Anatomy of a packet

- Header
  - Who sent it (address, port)
  - Who is it meant for (port, address?)
  - Message Type
  - Body Size
- Body


Anatomy of a packet

- typedef struct minimsg_hdr minimsg_hdr_t;
- struct minimsg_hdr {
  - network_address_t src_addr, dst_addr;
  - short src_port, dst_port;
  - int msg_type, msg_len;
- }


Anatomy of a port

- Local Port (dropbox)
  - Stores a queue of incoming messages.
  - Producer/Consumer semaphore.
- Remote Port (pointer)
  - A ‘pointer’ to a (host address,port#)
- Both are identified by port #.


The Basics

- minimsg.c/h
  - minimsg_initialize()
  - miniport_local_create()
  - miniport_remote_create()
  - miniport_destroy()
  - minimsg_send()
  - minimsg_receive()


minimsg_send()

- Check for error cases (invalid params)
- Create the header
- For a remote port
  - network_send_pkt()
- For a local port
  - Bypass the network.


minimsg_receive()

- Check for error cases
- Block until the local port has something in its queue
  - Semaphores!
- Extract the header, body, and length
- Create a new remote port from the return address
  - Application responsible for cleaning this up
- (be sure to free() the packet contents if necessary)


But wait...

- What happens when we send from a local port to another local port?
- Do we need to allocate a remote port for the return address?
- Can't we just return the local port?
  - (hint: no)


Gotchas

- Synchronization
  - Interrupt handler should not need to obtain locks.
  - Be careful which synchronization method you use.
- Semaphores
  - V() should NOT block!


Other notes

- Local ports all need a #.
  - Use hard-coded ports provided by the app.
- Test with network[1-6].c
- Message sizes/Body sizes are capped!


Good Luck