Network Programming

Dr. Thaier Hayajneh

Computer Engineering Department

Advanced UDP Sockets

Outline

- Advanced UDP Sockets (Chapter 22)
  - ➤ UDP or TCP? (section 22.4)
  - ➤ Adding Reliability to a UDP Application (section 22.5)
  - ➤ Concurrent UDP Servers (section 22.7)

UDP or TCP ?

- UDP must be used for broadcast or multicast
  - ➤ Error control?
  - ➤ Reliable multicast protocols
- UDP can be used for simple request-reply applications
  - ➤ Acknowledgments, timeouts, and retransmission?
  - ➤ Flow control?
- TCP for bulk data transfer
  - ➤ An exception is TFTP (Trivial File Transfer Protocol)

Adding Reliability to a UDP Application 1/2

Need to add 2 features

- *Sequence numbers so client can verify that a reply is for the appropriate request*
  - ➤ Client adds a sequence number to each request and server echo number back to client in reply
- *Timeout and retransmissions to handle datagrams that are discarded*
  - ➤ Send a request and wait for N seconds
  - ➤ If no response, retransmit and wait another N seconds
  - ➤ Repeat for a number of times and then application gives up
  - ➤ A linear retransmission timer

Adding Reliability to a UDP Application

• *Timeout and retransmissions to handle datagrams that are discarded*
  - ➤ RTT can vary from LAN to WAN
  - ➤ RTT between a client and server can change rapidly
  - ➤ Need a timeout and retransmission algorithm, that takes into account actual RTT

Concurrent UDP Servers

• Most UDP servers are iterative
  - ➤ Wait for client request, read request, process request, send back reply
  - ➤ How about if processing of client request takes along time → need for concurrency
• Simple to fork with TCP
  - • every client connection is unique
  - • TCP socket pair is unique for every connection
• What about UDP?

Concurrent UDP Servers

• Two different types of UDP servers
  - ➤ *Simple UDP server*
    - ✓ Server reads client request
    - ✓ Fork a child to handle the request
    - ✓ Request and socket address structure containing the client's protocol address passed to child in its memory image from fork
    - ✓ Child sends reply directly to client

Concurrent UDP Servers

• Two different types of UDP servers
  - ➤ *More involved UDP server*
    - ✓ Exchanges multiple datagrams with the client
    - ✓ Client only knows the server's well-known port number
    - ✓ Client sends first datagram of its request to well-known port number
    - ✓ How can the server distinguish between subsequent datagrams from that client and new requests?

Concurrent UDP Servers

- Two different types of UDP servers

  - *More involved UDP server*

    - ✓How can the server distinguish between subsequent datagrams from that client and new requests?

      - ❑Server creates a new socket for each client

      - ❑Binds an ephemeral port to that socket

      - ❑Use that socket for all its replies

      - ❑Client must look at port number of the server's first reply and send subsequent datagrams for this request to that port

9