CS4514 Computer Networks **B06**

Programming Assignment 4 **60 points**

October 20, 2006

Hurricane Katrina Concurrent Application Servers using Sliding Windows in the Data Link Layer Due Friday, December 15, 2006 at Team Demo Presentation

This assignment consists of two components: a design paper and an implementation that runs one of three application layers on an emulated three layer stack similar to assignment 2. This program is to be completed in three person teams. Each team must reserve a one-hour live demonstration of your project in an available slot on December 13, 14 or 15. If any team member **must leave town** by December 15, then the team demo should be scheduled for December 13 or Decmber 14. Unless an exemption is granted, all members of the team need to be present for the demo.

The general assignment is to build a **concurrent server** that handles requests from two or more clients. **[*Unix socket calls are used with TCP to emulate a physical layer for transmitting between clients and a concurrent server*].** Both the clients and the server will have a small application layer protocol that defines the interaction between clients and the server.

When a new client connects to the concurrent server, the server will ***create*** a child to handle all interactions with that client. Whether the server children are ***forked as processes*** or generated using ***threads*** is a team design decision. However, forked processes will then require implementation of shared memory mechanisms to handle concurrent accesses to the database structures required in the design of your application.

Application Choices

Assume FEMA has "roughly" outlined three specific applications that it has advertised for contract bids and that your team must submit a design proposal for one of the three applications outlined below. The intent of this assignment is to implement "limited" versions of one of the three applications layers on top of the three network layers specified below. The three application choices are: **Disaster Victim Location Database, Medical Examiner Identification Database and Evacuee Bulletin Board,.**

1. Disaster Victim Location Database

Note: Program 1 and Program 2 represent preliminary concepts related to this application.

FEMA wants to provide a concurrent server that maintains a Location database that acts as a repository of current information about disaster victims. The objective is to keep information in the server database that can be used to respond to client queries from **authorized** disaster victims or their families and to provide for remote entries to the Location database by **FEMA-authorized** individuals. The design team must decide what information to retain about the current location of an individual and include the possibility the person has been positively identified as being dead. The information retained could include a photo sent such as in program 2.

One of the key aspects of this application is that the repository might only hold partial information on an individual (e.g., only the first name of a child). Your application should include one query that deals with this problem.

2. Medical Examiner Identification Database

FEMA sees the purpose of this database is to serve as a repository for information regarding bodies collected during a disaster. One objective is for authorized clients to input information that might include photographs, fingerprints, dental records and other information such as tattoos that would facilitate identification of a body. The database needs to include information on where the body was discovered.

Another function to be supported is for a qualified medical examiner to input an official declaration that a specific body has been positively identified.

Similar and related to the first application, the design should provide for client queries from authorized family members to inquire whether a relative has been positively identified.

Note: Program 2 can be used as a mechanism to transfer data into the database.

One key question for the design of this application is how to provide information about positive identification that guarantees privacy and only authorized access.

3. Evacuee Bulletin Board

Seeing families scattered by a disaster such as a hurricane or flood and the fact that families will be forced to stay in "temporary" shelters such as high school gyms, FEMA wants to provide a Bulletin Board mechanism such that various groups such as extended families or other groups such as those seeking information about jobs or housing can communicate via a Bulletin Board server.

This Bulletin Board server needs to provide emails to be received, to be grouped by topic, to be read by evacuee clients.

Generic Application Layer Requirements

Given the time frame of this course, your design proposal for one of the three choices needs to support only a few primitive operations that correspond to client/server interactions. However, **the minimum requirement is four operations such that there is at least one long operation in each direction that will easily test that your sliding window mechanism works properly.**

The Network Layer

While the ***network layer*** is quite similar to the network layer in program 2, the **exact** network layer interface to your application layer will be specific to the application supported and needs to be defined in your design report. The network layer receives "messages" from the application layer and creates packets consisting of: 200 bytes of message payload, two bytes for packet sequence number, and any other control

bytes deemed necessary by your team's proposed design. The ***network layer*** sends packets to the ***data link layer*** and waits for packets received from the ***peer network layer*** through the ***data link layer.*** Received packets can be data packets or ACK packets or piggybacked packets. The ***network layer*** is responsible for taking packet payloads out of received data frames and constructing "messages' to be sent up to the application layer.

Initially, the ***client network layer*** calls the physical layer to establish a connection with the ***server network layer***. Once a connection has been established, the client network layer begins receiving 200 byte "chunks" of message from the ***client application layer*** and depositing each 200 byte chunk into a packet payload.

Data Link Layer

The data link layer receives packets from the network layer, creates frames, and sends frames to the physical layer for transmission. The data link layer also receives transmitted frames from the physical layer, extracts the payload, reassembles packets, and forwards packets to the network layer.

The intent is to reuse as much as possible from your implementation of program 2 except that the **PAR protocol is to be replaced by a Go Back N sliding window mechanism.**

Frame Format

All frames need a frame-type byte to distinguish data and ACK frames. All data frames must have two bytes for the sequence number, two bytes for error-detection, and one end-of-packet byte. The data link layer sends data frames that contain from 1 to 128 bytes of payload (encapsulated data from the network layer packet). ACK frames consist of **zero** bytes of payload, a two-byte sequence number, and a two-byte error detection field.

In this assignment you are to implement the **Go Back N** sliding window protocol at the data link layer. Your design may add other "overhead" bytes to the frame structure deemed necessary to implement **Go Back N**. As in program 2, your design needs to include two error-detection bytes. Your design will need to include sequence numbers in the frames and a mechanism for handling ACK's**.** The minimum frame size is your choice, and it is your design decision whether to *piggyback* ACK's or to send separate ACK frames.

Those groups seeking an A grade for the project and BS/MS students must implement a sending and receiving window size of four frames. A sliding window of size 1 is recommended for ALL groups that are running out of time at the end of B term. The data link layer continues to receive packets from the network layer until its sliding window is full of unACK'ed frames. This requires **multiple timers** on both the client and server side.

The Client Process Flow

Each Client will call the ***physical layer*** to establish a connection to the concurrent server. The ***data link layer*** then gets packets from the ***network layer***, puts together frames and gives them to the ***physical layer*** to send. The ***data link layer*** flow depends on events coming from the ***network layer***, the current availability within the sliding window, and events coming from the ***physical layer***. Your application design needs to provide a clean mechanism for the client to notify the server that it is done and to cleanly close the client's connection to the server.

The Server Child Process Flow

This assignment requires implementation of a concurrent server that supports concurrent conversations with multiple clients.

Each ***data link layer server child process*** waits for frames from the **server physical layer** and passes packets up to the ***server network layer***. Similar to the client side, the flow of the ***server data link layer*** depends on whether there is traffic from the Server to be sent back to the Client (either frames with packets or ACK frames).

The Physical Layer

The ***physical layer*** for this assignment is the same as in program 2.

The ***physical layer*** uses Unix sockets to send the constructed frames as actual TCP messages between the clients and the concurrent server. When the ***client physical layer*** first establishes a connection to the concurrent server, it **must** send one TCP message to the server child process or thread to identify itself.

Frame Error Simulation

Since real TCP guarantees no errors at the emulated physical layer, your program must inject artificial transmission errors into your physical layer.

Force a **transmission error** in every 8^{th} data frame sent by flipping any single bit in the error-detection bytes prior to transmission of the frame. Force a **transmission error** in every 6^{th} ACK frame sent by using the same flipping mechanism. (i.e., data frames 8, 16, 24, … sent will be perceived as in error by the receiver and ACK frames 6, 12, 18, … sent will be perceived as error by the receiver.) Note: since both the clients and the server can all send data frames and ACK frames each of these counts that trigger transmission errors must be kept relative to whichever node is sending data or ACK frames. When the ***client or server data link layer*** times out due to either type of transmission error, it resends the data frame with the correct error-detection byte.

Assignment Hints

- **[Design]** Plan your design in a modular fashion such that if everything is **not** totally working, you can turn in and demo some type of output that shows **exactly** what is working. Relaxing the sliding window scheme is one option when your project is in trouble.

- **[Documentation]** Your commented program **must** have a special section to explain the details of your specific design decisions. **Remember: This is a team project and all routines must specify the author as part of the documentation!! Team members may not receive the same grade on an assignment due to uneven workload.**

- **[DEBUG]** Include print statements in the various layers while debugging. You should consider some type of verbose debugging flag that can be turned on and off.

- **[Performance Timing]** You **must** measure and print out the total execution time of the complete emulated transfer per client.

- **port numbers** - Your clients should have unique port numbers and the clients should treat the server port number like a "well-known" port number.

What to turn in for Assignment 4

The README file **must indicate how much of project is working at the time of the demo. It also needs to indicate the non-working parts and some indication of why specific modules are not working correctly. This README information is critical for teams to receive partial-credit for for non-operational versions of program 4.**

See the Design Proposal Report for more detailed information about what to turn in at the project demonstration.