Network Application with Multiple Clients and Concurrent Server {January 8, 2013}

Course Project (progc)

Points and Due Dates

Proposal 2 points Due: 6 p.m. Thursday, February 14, 2013

Each project group must submit a typed project proposal. The proposal includes team members and an explanation of the specific application that you propose to implement and support on MININET.

Design Report 43 points Due: 4 p.m. Monday, March 4, 2013

Program teams must deliver a hard copy typed design report to my office. Pseudo-code is **unacceptable** for your design report. Professional technical prose is expected. This report will receive a letter grade and a point grade based on a grading rubric that includes all the standard criteria of a professional technical report (i.e., grammar, writing style, typos/misspellings, clarity and content will **ALL** be considered).

Final Project 90 points Due: 11:59 p.m. Monday, April 15, 2013

Team Demos scheduled between April 16 – April 19 and April 24 – April 26, 2013

Each project must schedule a one hour time slot for a LIVE demonstration of their project during this period. Bring to your demo: a short users guide that lists the commands and explains the parameters used; a hard copy of your program which must conform to standard commenting expectations; the original marked-up copy of your design report, a short addendum that explains any significant changes and improvements made to your design; and a **README** document that itemizes those aspects of the project that are working correctly and those which are not fully working as of the demo.

Introduction

The networks course project is a three-member team assignment to design and implement a three layer network model which permits two or more application client processes to communicate over the Internet with a concurrent application server. The expectation is that these programs will be written in C or C++ on CCC Linux hosts and will use TCP/IP to exchange messages. This assignment is intended to expose students to at least three aspects of computer networks - the client/server paradigm, key issues in the design of the data link layer, and working with a real network protocol (TCP/IP).

Once Program 1 has been completed, students can submit their preferences for a team partner. Note – **each** partner needs to send an email indicating their preference! I will assign team membership for those students who have not submitted a partner preference by **February 5th**. Students who not do complete Program 1 successfully **MUST** schedule a meeting with me before being assigned to a project team.

Each team must submit a typed project proposal (one page max). Each project team must then schedule a half hour **required team design meeting** to review project design issues such that this meeting is completed by **February 22nd**.

The following description outlines the project but **deliberately** does not specify all the details. This provides student teams with important choices in the functionality of the model implemented and puts some of the design decision in the hands of the students. Throughout the following description it is important to separate the abstraction of a three layer model, henceforth referred to as MININET, from the details of the UNIX socket interface to TCP/IP.

The MININET protocol consists of three layers - application, data link, and physical layer. The application layer consists of an application layer protocol between a client/server pair of processes and an interface to the lower layer. Application layer communication is in the form of request and response messages. Clients send requests to a concurrent server and receive responses to these requests from the server. The data link layer provides an ***emulated*** frame-level communication between client and server processes on separate CCC computers. The data link layer operates over an **emulated** unreliable channel subject to lost and garbled frames. The physical layer provides the illusion of a two-way communication channel on which the data link layer can send and receive arbitrary byte streams. The physical layer is emulated using TCP.

MININET Application Layer

The application layer implements an interactive request/response protocol and provides an interface to the data link layer. An application layer client reads commands from standard input, converts them into server requests, sends the request messages to the server, and prints the response messages returned by the server. The concurrent server listens to establish a TCP connection with each client, interprets the received client requests and reacts to client requests by sending back appropriate responses.

Each project team MUST propose, design and implement their specific application layer. Your design report includes an explanation of the functionality of both the client and the server and a precise, detailed specification of the application layer protocol. You must specify valid commands that the client accepts from input, the meaning of the command request to the server, and the server responses. Your application **MUST** include at least **six** distinct client commands. Your command set must involve at least one large transfer in both directions (e.g. a 1 Mbyte jpeg image). These transfers should be of sufficient size to test your sliding window mechanism. Your application specification needs at least a primitive login command that verifies that a client belongs to the set of authorized users of the service provided by the application server.

Messages are sent between peer application layer entities. To simplify the overall abstraction of this assignment, it is useful to absorb a small network layer with the application layer. While there are no size limitations on message objects, to simplify the grading of many course projects, assume that messages are broken up and encapsulated into network layer packets which have a maximum size of **192 bytes**. These network packets include encapsulated message 'chunks' and any overhead bytes specified in your design report. Note, the interface between the application layer and this small network layer must be able to message objects that are large files. The network layer (as the interface to the data link layer) sends and receives packets. Hence, the critical MININET abstraction is that both the client and server network layers interface with their respective data link layers ONLY via two mechanisms ***dl_send*** and ***dl_recv***. Exactly what these communication mechanisms are and the specific entity types involved in these communications are important design decisions.

If the server encounters an application-level error while processing a request, it returns an appropriate error message. The application client process then prints out appropriate error explanations to the user.

Choices made with respect to concurrency control among the emulated layers are also key design decisions for this project. It is essential that each project team discuss and decide upon their approach to these MININET design choices PRIOR TO writing the design report.

MININET Data Link Layer

The data link layer accepts packets passed via ***dl_send***, places a packet into a data link frame, and sends the frame to the physical layer for transmission. The data link layer communicates with its peer server process over a two-way communications channel accessed through UNIX file descriptions (see the physical layer).

Your data link layer must include the ability to handle arbitrary data streams and to emulate an error-prone channel. Thus, the data link layer must include considerations for framing, data stuffing, buffering, retransmissions, checksums, and some form of flow control. Retransmissions will require a timer mechanism to detect transmission errors.

You must implement a sliding window protocol (see the Tanenbaum handout for details). Whether **Go Back N** or **Selective Repeat** is used is a team design choice. However, you must plan on using a sender sliding window size of at least **four** frames. Conceptually, the data link layer buffers the data frames internally and allows the application/network layer to continue processing. If the sender window becomes full of un-acknowledged frames, the data link layer must block (and disable the application/network layer entity) until frame buffer space becomes available.

In the ***dl_recv*** functionality, a new data frame may arrive at the data link layer entity before the network layer actually calls ***dl_recv***. Because frames must be processed when they arrive (otherwise you might ignore an acknowledgement frame), received data frames will have to queued. When ***dl_recv*** is called, it must check for the presence of data in the received queue. If none is present, it will have to suspend itself until an appropriate event occurs. As in Tanenbaum's discussion, the data link layer entities must respond to timeout, frame_arrival and frame_error events.

MININET Physical Layer

The actual communication between the client and the server takes place using Linux sockets and TCP. The physical layer of MININET handles the details of establishing a TCP connection and sending real TCP packets over the WPI campus network. Assume the maximum size of the frame payload is **100 bytes**. That is, the data link layer hands off frames to the physical layer for transmission. To simplify matters, you may make reasonable assumptions on the maximum amount of data stuffing needed. The design of the full frame is a team choice but it must include “artificial” framing bytes at each end of the frame sent.

The physical layer is responsible for inducing errors in the transmission. Your design must include an adjustable error rate input as a command line argument for both the client and the server. The error rate is used to ***randomly*** insert a single bit error into the frame checksum field before a frame is sent by the client or the server.

Test Data

Since program teams are specifying the application layer, it is also the team's responsibility to provide interesting and meaningful test data to show that your MININET works correctly. If the final project turned in does not work completely, then provide mechanisms (even if they have to be jury-rigged) to demonstrate the maximum working functionality of MININET modules during the final demo. All routines MUST be assigned a single, primary author in the program internal documentation.

Output Logs and Monitoring

To observe the performance of the sliding window protocol and to check whether your implementation is working properly, you need to collect statistics. For debugging reasons, you should design a statistics gathering monitor on both the clients and the server machines.

The best form of monitor is one that records both ongoing and final statistics. When you demonstrate your working project, it is advantageous to have the client and server monitors outputting information to separate screen windows. The following are some suggested statistics that should be maintained for both clients and the server. The important idea is that the totals that you provide on the clients and server should be such that one can add and subtract totals to show that your sliding window scheme works in the face of errors. (You should include any additional statistics that are germane to the specific application that your MININET is supporting):

1. the total number of data frames sent
2. the total number of retransmissions sent
3. the total number of acknowledgments sent
4. the total number of data frames received correctly
5. the total number of acknowledgments received correctly
6. the total number of data frames received with errors
7. the total number of acknowledgements received with errors
8. the total number of duplicate frames received
9. the number of times the data link layer blocks due to a full window.

While debugging you may wish to show the size of frames sent and received. It is wise to have a verbose enable/disable switch such that you can turn on and off debug messages inside your programs. For performance analysis, you should measure the time required to satisfy individual client requests.

Comments and Suggestions

There are several aspects of this project which require specialized systems programming knowledge - the timer mechanism, communication and signaling mechanisms, using processes or threads to implement in conjunction with ***dl_send*** and ***dl_recv***, and the LINUX socket interface. When making design decisions, it is valid to propose a *less-than-elegant* solution due to your time constraints. However, your design report needs to explain and defend such choices.

Note, this document outlines the project without taking up the issues of suggested strategies to build your MININET. Consider developing your network by gradually adding complexity to the problem. For example, build and test an errorless data link layer first. Start with a simple stop-and-wait protocol and subsequently switch to the more difficult sliding window scheme.

Design Report

The primary purpose of the required team design meeting is to go over the expectations and key components that will impact the grading of your design report.

Teams need to meet **quickly** after they are assignment and prepare for and promptly schedule the required team design meeting.

Your design report will be graded based on basic technical writing standards including: grammar, organization, formal presentation style, typos and clarity of writing. The three key points to focus on in your design report are: a detailed specification of your proposed application layer; a thorough explanation of the mechanisms to be used when communicating between the layers; and a discussion about the concurrency control mechanisms that you will employ (e.g., semaphores, shared memory, pipes, processes and/or threads). Your design proposal must identify your specific choices. Warning: do not get too fancy with both your application choice and your design. Avoid basing your design on systems programming mechanisms that you have not used previously.

Your design report must include: the detailed specification of the application protocol, a diagram that explains the components and interfaces in your design, an allocation of work between team members, a milestones schedule, and a bibliography. The design paper should be approximately 12-24 typed pages not including pictures and diagrams.