Real-Time and Embedded Systems: Programming Assignment

Dr. Colin Perkins

21st February 2007

1 Background

Many companies are now marketing voice-over-IP (VoIP) phones. These look like a traditional desk telephone, but connect to an Ethernet LAN instead of a phone line. They can make calls to other voice-over-IP devices using RTP over UDP/IP as a network protocol (discussed in lecture 16), or can use a central gateway server to make calls to the traditional circuit switched telephone network.

Internally, VoIP phones run a cut-down version of a general-purpose operating system, such as Linux, on a low speed processor with limited memory. Software is loaded from flash memory, and the phone boots directly into the telephony application after configuring its network interface using DHCP. When a call is initiated from the user interface, the phone will negotiate a connection using the Session Initiation Protocol, then start transmitting and receiving real-time speech data. The G.729 speech compression algorithm is often used, and produces an 80 bit frame of compressed speech every 10 ms. Running the speech compression algorithm can take up to 30% of the system processor power, decompression takes 20%. The speech data is transmitted in RTP packets sent over UDP/IP every 20 ms. The RTP packets comprise two frames of speech data and an RTP header, for a total of 32 bytes (an average rate of 12.5 kbps). In addition to RTP packets containing the speech data, reception quality reports are sent every 5 seconds. Received speech packets are decompressed and played to the user as they arrive.

More advanced systems provide video in addition to voice. These video phones work in a conceptually similar manner to VoIP phones, except that they can generate hundreds, possibly thousands, of RTP packets per second, since there is more data to transmit.

2 Voice-over-IP System Design

The first part of this assignment is an exercise in real-time system design. Considering the VoIP system outlined in Section 1, identify the periodic and aperiodic tasks comprising the system, and list any known parameters of the periodic tasks. Describe any dependencies between these tasks.

The VoIP system is to be realised on a platform that supports pthreads with POSIX extensions for real-time priority scheduling. Explain what scheduling algorithm you would use to implement the VoIP system. Discuss under what constraints it is guaranteed to be possible to schedule the system, and how the constraints on schedulability determine the parameters of the tasks in the system.

Outline a high-level design for the system in C-like pseudo-code. You should illustrate the behaviour of the system, the manner in which the schedule is implemented (including task priorities), and the way network operations for sending/receiving speech data are performed. Your pseudo-code should be sufficiently detailed to show how the POSIX API calls relating to threading, scheduling, and networking are used, and how the system timing is maintained, but does not need to show details of the implementation of the speech compression, network protocols, or user interface.

3 Real-Time Behaviour of Linux

Linux has a history of problems due to system call latency where, in some circumstances, tasks may be delayed for tens of milliseconds when the operating system blocks during access to hardware devices. It is possible that this may disrupt the operation of a voice over IP system if implemented using Linux, since the inter-packet spacing is of the same order as the reported system call latency. The aim of the second part of the assignment is to test the real-time behaviour of Linux, to see if these concerns are justified.

Write a test program, `sender`, to simulate a voice-over-IP sender. When started with the IP address of another host as a command line parameter, this program will perform two functions:

- Every 20ms, it should send a datagram packet to UDP port 5004 of the host specified on the command line. These packets must contain 32 bytes of data, to simulate a voice-over-IP system using the G.729 speech compression algorithm. The contents of the 32 bytes should include a sequence number that increases by one with each packet sent, and a timestamp indicating the time at which the packet was sent. For the purposes of this test, the values of the other data bytes are unimportant.
- It will listen for packets on UDP port 5005. When it receives a packet on this port, which will have the format described below, it should should retrieve and display the statistic on the fraction of packets lost.

Write another program, `listener`, that listens to the packets being sent by the `sender` program and generates reception quality reports:

- For each packet it receives on UDP port 5004, the `listener` program should record the value of the sequence number and timestamp contained in that packet, and also the arrival time of the packet. The sequence numbers should be used to calculate the number of packets lost in transit. The transmission and reception timestamps should be stored in a file for later analysis.
- Every five seconds, the `listener` should send a datagram packet (a "Reception Quality Report") back to UDP port 5005 of the host running the `sender` program (the IP address of the sender host should be specified on the command line). This datagram should be 70 bytes in length, and should report the fraction of the packets sent on port 5004 that have been lost since the last Reception Quality Report was sent. For the purpose of this test, the remainder of the contents are unimportant.

The two test programs should be written in C using the sockets API for UDP/IP communication, and should run on Linux. The code you develop does not need to have a "pretty" user interface. You should submit a printout of the source code to the two programs, along with any design notes you have made.

Run the `sender` and `listener` programs, sending data from one host in the lab to another host in the lab. Using the data captured by the listener program, plot a timing graph – similar to that shown in lecture 15 – of send time versus arrival time for the system, over a period of approximately one minute. Repeat on both a lightly loaded host and on a heavily loaded host (for example, when the receiver machine is idle, and when it is compiling a large program). You should use a program such as gnuplot or Microsoft Excel to plot the graphs, producing plots to illustrate any interesting aspects of the timing behaviour of the system.

Comment on the data presented in your two timing graphs. Do the Linux systems you tested have sufficiently accurate timing to be suitable for use in a voice-over-IP system? How often does your system miss its deadlines? You may need to zoom in on a small part of the graph, or instrument your code to plot additional statistics, to observe the timing properties.

If packets were to be sent at a higher rate (smaller inter-packet spacing) would these systems still be usable? For example, consider the behaviour of a system sending high quality video with several thousand packets per second, instead of audio with 50 packets per second. Experiment with the system to determine what is the minimum inter-packet spacing the hosts can support while maintaining reasonable timing behaviour. Document your experiments, and discuss how you might improve the performance of the system.

4 Submission

This assignment is worth 15% of the mark for this module: 5% for the high-level VoIP system design and pseudo-code (Section 2), 10% for the design and implementation of the `sender` and `listener` programs, the timing graphs, and the associated discussion of how system performance changes with varying packet rates (Section 3).

Completed assignments must be submitted by 5:00pm on Friday, 16th March 2007 via the locked box outside the Teaching Office. Submissions must be in an unsealed A4 envelope with your name, the name of the course, and the assignment number clearly written on the front. You must include your pink declaration of authorship form in the envelope. Please note that failure to provide an envelope may result in other students seeing your mark. Any late submission will be awarded zero marks unless accompanied by a valid special circumstances form.