Real Time and Embedded Systems: Programming Assignment

Dr. Colin Perkins

25th February 2009

1 Introduction

Many Internet service providers (ISPs) are starting to offer Internet Protocol Television (IPTV) as a value-added service. The ISP provides a set-top box to the consumer, who connects it to their television and uses it in exactly the same way they use any other set-top box; the only difference being that it connects to their broadband connection, rather than a dedicated TV aerial or cable connection.

Internally, these IPTV set-top boxes run a cut-down version of a general purpose operating system, such as Linux, on a relatively low performance processor with limited memory, augmented by a hardware video decoder chip. The operating system is loaded from flash memory, and the device boots directly into the IPTV application on start-up. When the consumer is watching TV, programmes are streamed to the set-top box in real time, over the broadband connection, using RTP as the application-layer protocol. The data rate varies from 4Mbps for low-quality standard definition channels, up to 19.2Mbps for high-definition channels, with a range of intermediate rates possible depending on programme content and the compression options chosen by the ISP. The lower-rate programmes can be watched on most ADSL connections, while the higher-rate programmes require fibre-to-the-home connections. The IPTV system can stream video using either UDP or TCP at the transport layer. UDP is generally preferred, since it provides no features that can disrupt the media timing, but TCP is sometimes used instead, despite its use of retransmissions to achieve reliability, since it can be tunnelled over an HTTP connection for ease of firewall traversal.

The aim of this exercise is to explore the performance of UDP and TCP streaming over local area networks and the Internet, to evaluate how much timing disruption is caused by the use of TCP compared to UDP.

2 Performance Evaluation: UDP

Begin by writing a test program, `udp-sender`, to simulate an IPTV sender running RTP over UDP. When started with a hostname and a desired transmission rate (in Mbps) as command line parameters, this program should send 1000 byte UDP packets to port 5004 of the host specified on the command line, with spacing chosen to achieve the desired transmission rate. Each packet should contain 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 (in a real system, they'd contain the streaming video data). In addition to sending data, this program should also listen for packets on UDP port 5005. When it receives a packet on this port, which will have the format described below, it should retrieve and display the statistics on the fraction of packets lost and the interarrival jitter contained within the packet.

Write a second program, `udp-listener`, that listens to the packets being sent by the `udp-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, and the interarrival time jitter of the packets. The interarrival time jitter is calculated as follows. If Si is the transmission timestamp from packet i, and Ri is the time of arrival for packet i, in the same units, then for two packets i and j, the difference in the relative transit time, D(i,j), may be expressed as

```
D(i,j) = (Rj - Sj) - (Ri - Si)
```

The interarrival time jitter is then calculated continuously as each data packet i is received, using the value of D for that packet and the previous packet i-1 in order of arrival (not necessarily in sequence), according to the formula

```
J(i) = J(i-1) + (|D(i-1,i)| - J(i-1))/16
```

Whenever a reception report is issued, the current value of J is sampled and reported as the interarrival time jitter (assume that `J(0) = 0` i.e. the jitter of the first packet is zero). In addition, 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, and the current value of the interarrival jitter. 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.

Run the sender and listener programs, sending data between two hosts in the lab, for a range of data rates representing IPTV traffic. If you are able, also run tests between a lab machine and a host outside the University. Using the data captured by the listener program, plot timing graphs – similar to that shown in lecture 15 – of send time versus arrival time for the system, over a period of a couple of minutes, for several transmission rates. Also, plot the packet loss rate and the interarrival jitter as provided in the reception quality reports. 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.

Discuss your results, and the data presented in your various timing graphs. Do the Linux systems you have tested have sufficiently accurate timing to be suitable for use as an IPTV set-top box? What, if any, problems are observed in the timing of the data? 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. Experiment with the system to determine what is the maximum rate the hosts and network can support while maintaining reasonable timing behaviour. Document your experiments, and discuss how you might improve the performance of the system

3 Performance Evaluation: TCP

Write two more programs, `tcp-sender` and `tcp-listener`, to perform the same functions as described in Section 2, but this time running over a TCP connection. Note that the `tcp-sender` may be rate limited by the TCP congestion control algorithm, so you should measure the rate at which you're able to send, and log an error if this is lower than the desired rate.

Run these TCP-based sender and listener programs, sending data between two hosts in the lab, using a range of data rates representing IPTV traffic. If you are able, also run tests between a lab machine and a host outside the University. Measure and discuss the performance, plotting timing graphs and any other measurements you consider important. Discuss how your results differ between the TCP and UDP based systems.

Finally, write a program (`tcp-wget`) to download a file using HTTP, monitoring the timing of the download as it does.[1] This program should plot interarrival times of data (as measured by the time between successive `read()` requests completing) and throughput of the download over time. Measure and discuss the performance of this program when downloading large files. Compare these results to those using the `tcp-sender` and `tcp-listener` programs, and discuss any implications you see for wide-area IPTV streaming.

4 Submission

This assignment is worth 15% of the mark for this course: 5% for the code, and 10% for the performance evaluation *and discussion*. You should submit a single written report outlining the design and implementation of your code, describing the experiments you undertook, then presenting and discussing your results, and their implications for IPTV systems. This report should include printouts of your code in an appendix. Note that clarity of writing and presentation, and quality of written argument, will be explicitly marked (as part of the 10% marks for performance evaluation and discussion).

Completed assignments must be submitted by 1:00pm on Monday, 16th March 2009 via the locked box outside the Teaching Office. You must include a pink declaration of authorship form with your submission. Late submissions will be awarded zero marks unless accompanied by a valid special circumstances form.

[1] This part of the programming assignment is similar to the exercise given as part of the NS3 course, and the notes for that course – available on Moodle – might be useful to refresh your memory of HTTP operation.