Program Assignment 5

CMSC 417 Spring 2012

Apr 22

1 Deadline

May 08, 2012.

2 Introduction

The goal of this project is to implement unreliable rate-controlled message delivery to neighbors. This project extends the previous project by providing rate-controlled message transmission. We'll add a "sendmsg" command that will inject a message into the network when doing so will not exceed a rate limit. The rate-control requirement ensures that messages from a host do not swamp its neighbors or the network. Because the service runs over UDP (over IP), without any retransmission, it remains unreliable.

The rate control is the so-called "leaky bucket" algorithm. Water (packets) pours into a leaky bucket (queue). When the bucket has water, it leaks steadily (sends at a fixed rate). If the incoming water flow is so fast that it fills the bucket (queue is filled), water spills out (packets are dropped).

3 Team Project

Please continue with the same team that you had for Project 3, unless you have strong reasons not to.

4 Requirements

1. "sendmsg" command

    - This command takes two input parameters.
      << **sendmsg dst-address msg** >>

  - dst-address is the neighbor's logical address, the 32 bit source address field in the packet header of received "hello" message.
  - msg is a message to send. You may assume that msg contains no space in it.
- Limits the rate of sending to 10 packets (of maximum size 1000 bytes each) per second to each neighbor
- Uses a per-neighbor output queue of maximum size 10 packets (irrespective of packet size) to limit the sending rate.
- When sendmsg is invoked, it does the following in sequence:
  - Look up the IP address and UDP port of the target neighbor in the neighbor table.
  - Check if last packet was sent 0.1 seconds ago or earlier:
    - If so, send the message immediately using UDP sendto
    - If not, put the message in the output queue
    - If the queue is full, drop packet
  - Ensures that the queue, if not empty, is drained by one packet every 0.1 seconds. That is, your program can send unicast messages to each neighbor at most every 0.1 seconds.
- **Don't send messages to the multicast address!**

2. Implementation

- Requirements of project 4.
  - Periodic "hello" message.
    - You need not implement input arguments such as "TIME-X", "TIME-Y" and "PORT".
    - Default input arguments are "25sec", "100sec" and "5050", just as in project 4.
  - "print neighbor table" command.
- Implement the sendmsg functionality
  - Handle queue overflow by printing "ERROR:NOBUFF". Do not block the sendmsg command waiting for the queue to drain.
  - Handle unknown or dead neighbor by printing "ERROR:NOROUTE"
- Print messages received on the unicast socket (with version 1, protocol 1 in packet header).

- Ensure that your program does not leak memory.
- Your executable must be named “three” (for the testing scripts to operate correctly).