ECE 697J Fall 2003: Lab 1 – TCP/IP Flow Identification

Due: 10/30/03

The goal of this assignment is to program a TCP/IP flow identification algorithm. The program should read a trace of packets and tell the following statistics at the end of the run:

- Number of flows observed. That is the number of unique 5-tuples consisting of IP source and destination, layer 4 protocol number, and source and destination ports.
- Number of properly terminated TCP flows.
- Maximum number of active flows at any given point.
- 5-tuple of flow with most data transferred.

For handling packets, we use the PacketBench environment (see Ramaswamy, R., Wolf, T.: "PacketBench: A Tool for Workload Characterization of Network Processing," in *Proc. of 6th IEEE Annual Workshop on Workload Characterization (WWC-6)*, Austin, TX, October 2003. http://www.ecs.umass.edu/ece/wolf/courses/ECE697J/papers/wwc2003.pdf). PacketBench provides a simple interface for handling packets that does not require any understanding on how to process packet trace files.

The lab can be implemented on any UNIX, Linux, or Cygwin system (e.g., your ECS account). The detailed instructions are as follows:

1. Download the lab code and data from:
   http://www.ecs.umass.edu/ece/wolf/courses/ECE697J/labs/ece697j-lab1.tar.gz
   and save the file into a new directory.
2. Extract the files:
   ```
   gunzip ece697j-lab1.tar.gz
   tar xvf ece697j-lab1.tar
   ```
   This should give you a number of files including two trace files (`AIX-1058228090-1.tsh` AND `OSU-1058584197.tsh` (10MB!)).
3. Compile all files by running: `make`
4. Edit the `Makefile` if necessary. The VERBOSE switch can be toggled by uncommenting/commenting "-DVERBOSE". To recompile run:
   ```
   make clean
   make
   ```
5. To run PacketBench, use the command line syntax as follows :
   ```
   bench -N {tracefile name} {dump file} {drop file}
   ```
   Example :
   ```
   ./bench -N ./AIX-1058228090-1.tsh dump.raw drop.raw
   ```
   Dump file is the file in which packets get written to after normal processing.
   Drop file is the file in which packets which don't get processed (i.e. packets that are dropped) are written to.
   The output should look like:

   ```
   bench.c:291(main),Processing packet...
   Packet handler received IP packet
   bench.c:33(write_packet_to_tsh_file),Writing packet to dump file
   bench.c:277(main), Warning : TSH trace file used. Packet payload is
   invalid
   [...]
   bench.c:291(main),Processing packet...
   Packet handler received IP packet
   bench.c:33(write_packet_to_tsh_file),Writing packet to dump file
   ```

```
Total packets processed : 1981
Packets dumped : 1981
Packets dropped : 0
```

Note that the warnings are ok (due to the nature of the trace files that we use).

6. The packet handling code is in the file `flowid.c`. Your code should go into `flowid.c` and `flowid.h` in the appropriate, marked places. At present, the packet handling code simply prints a confirmation message when it receives a packet.
7. Two trace files are provided. Both are in binary time sequenced header (TSH) format (44 bytes per packet).
    - `AIX-1058228090-1.tsh` ~ 2,000 packets
    - `OSU-1058584197.tsh` ~ 248,000 packets
8. There is a hash function provided in the `flowid.c` file that might come in handy.

You should turn in a brief (around 2-4 pages) report that contains the following:

- A diagram showing the fields of and IP and a TCP header.
- A diagram of a TCP state machine that you use to determine the state of a connections.
- The results from your measurement (see above) for both sample traces.
- A printout from your `flowid.c` and `flowid.h` file (does not count towards page count).

There will be a help session on 10/28/03.