Computer Networking I

Switch assignment

Martin Sehnoutka

November 1, 2015

Contents

1. Messages format

2. Software architecture

2.1 Server side - switch

```
main :: IO()
main = do
  controlChannel <- atomically newTChan
  readChannel <- atomically newTChan
  let channels = (controlChannel, readChannel)
  socket <- listenOn $ PortNumber 4242
  putStrLn $ Tg.control ++ "Starting switch on port: " ++ show 4242
  forkIO $ switch channels []
  acceptLoop socket channels
```

2.2 Client side

3. Installation on Lintula machines

First step is to add ghc [1] and cabal [2] into your path variable:

```
$ PATH=$PATH:/usr/local/lang/haskell/ghc-7.10.2/bin
```

Then it is necessary to update cabal package database. These packages will be installed into ~/.cabal directory, thus it does not need root privileges.

```
$ cabal update
```

Now it is possible to install both switch and client using cabal installers.

[1] Glasgow Haskell Compiler

[2] Common Architecture for Building Applications and Libraries

```
$ cd <SWITCH_DIR>/switch
$ cabal sandbox init
$ cabal install -j
$ cd <SWITCH_DIR>/client
$ cabal sandbox init
$ cabal install -j
```

4. Testing

I wrote little script for testing purposes. This script runs switch and then in loop starts 8 clients. Output of each process is stored in different file: switch.log, client0.log, client1.log, etc.. By exploring these file I can compare messages that arrived to each client and switch.

For example I can explore messages that arrived into switch with destination address 3 and compare them with client 3 output:

```
$ cat switch.log | grep --regexp='(Unicast) [0-9]->3' | tail --lines=6
[client]  18:16:52 (Unicast) 5->3: uksvykjm
[client]  18:16:58 (Unicast) 1->3: kgxqmenyqw
[client]  18:17:03 (Unicast) 2->3: vntppmmkjudrwag
[client]  18:17:05 (Unicast) 0->3: uuonx
[client]  18:17:05 (Unicast) 7->3: turvwhxhf
[client]  18:17:05 (Unicast) 6->3: plexvthznugwllv
$ cat client3.log | grep --regexp='(For me)' | tail --lines=6
(For me) 1->3: kgxqmenyqw
(For me) 2->3: vntppmmkjudrwag
(For me) 0->3: uuonx
(For me) 7->3: turvwhxhf
(For me) 6->3: plexvthznugwllv
(For me) 7->3: nuwzqpmkko
```

As you can see, messages are sent and received in different order, but this is expected behavior. I can do the same for broadcast messages:

```
$ cat switch.log | grep Broadcast | head --lines=5
[client]  18:16:45 (Broadcast) 7->8: mdowboqvzxjg
[client]  18:16:46 (Broadcast) 3->10: ywspkekbq
[client]  18:16:46 (Broadcast) 4->8: ofxcfrixmtqps
[client]  18:16:46 (Broadcast) 6->8: xqgtta
[client]  18:16:46 (Broadcast) 5->8: zfhkjya
$ cat client7.log | grep Broadcast | head --lines=5
(Broadcast) 3->10: ywspkekbq
(Broadcast) 4->8: ofxcfrixmtqps
(Broadcast) 6->8: xqgtta
(Broadcast) 5->8: zfhkjya
(Broadcast) 5->9: bclpwbfpshnuum
```

In this case I can even see, that switch does not send messages back to their source address because very first message is from client 7, but is not present in client's log file.