Progress Report 2 (GSOC ‘16)

The project now has been completely moved to this repository. Each scenario now has its own status report showing the success/failure of each test and their respective description. Comments regarding the current progress and suggestions can be made here.

Pending Scenarios

Path MTU Discovery

For this scenario, until now these are the conclusions to which I have arrived at -

- FreeBSD **TUN** device does not support **TSO**, which probably is the reason that the receiver is not generating **ICMP (needfrag)** message when an overloaded segment is sent.
- The test can be successful if I use remote mode with two FreeBSD machines. I will be setting up another VM for this purpose and will then be reporting about the further progress. Leaving this for later since now there are more important scenarios to consider.

Scenarios which require review

Socket Shutdown

All the tests and results related to this scenario are available here. All the tests for this scenario have failed. These are some of the possible reasons which I have drawn regarding their expected and observed behavior -

- After calling `shutdown(4, SHUT_RD)`, `read()` should not be able to run. This is indeed the observed behavior. However, `read()` still returns `0`, although I think `-1` should be the appropriate return value. This is the log during the test runtime -

```
tests/bsd/tcp/shutdown/shutdown-rd.pkt:18: runtime error in read
call: Expected result -1 but got 0
```

- After calling `shutdown(4, SHUT_RD)`, if client writes some data to the sender, the TCP stack should respond with a **RST**. However, FreeBSD in this case responds with a delayed **ACK**. This however is also the case with Linux, as it is mentioned here and is justified by the similar test.
  This is the log during the test runtime.

  ```
  script packet:  0.360000 R 1:1(0)
  actual packet:  0.367084 . 1:1(0) ack 1001 win 1008
  ```

Handling of incoming ICMP packets

All the tests and status for this scenario are available here. Currently I have successfully tested **19 out of 56** ICMP types. For the implementation of the remaining types, I will have to look into the source code for packetdrill (`icmp_packet.c`) and add corresponding missing ICMP types. I am also currently looking into finding a way of directly sending ICMP packets to the **TUN** device and studying responses, which will ease the entire testing process and I will finally make changes in the code for all the successful types.

Current problems

I am currently not able to figure out what property is being asserted by the following code snippet -

```
assert tcpi_reordering == 3
```

I think this is pointing to RFC4737 but am not sure. It would be very helpful if you could tell whether it is the appropriate reference. Once this gets figured out, I will be able to correctly write tests for all the remaining scenarios.

Timeline (Slight modification)

Start	End	Task
10 June	15 July	All the remaining scenarios will be done.
16 July	31 July	Attempt at completing the additional scenarios mentioned in proposal.
1 Aug	11 Aug	Attempt at patching packetdrill by adding a new mode of testing in which remote host will not need an instance of packetdrill running.
12 Aug	14 Aug	Code review
15 Aug		End of coding (soft)
23 Aug		End of coding (hard)

The idea in which the new patch for packetdrill will be developed was mentioned in the proposal, but I will be focussing on this from 1 Aug when all the additional scenarios mentioned in the proposal will be successfully completed.
I will also be trying to give a try into the other patch which I mentioned in the proposal for supporting multiple concurrent connections during this period itself.