

1 Evaluation

This section focusses on the comparison testing which was carried out between the C and Java implementations of basic middleboxes. Evaluation of the results will then be discussed and further improvements to the Java solutions will be considered.

mention
machine
specs

1.1 Packet Generating

Packet generating is the act of creating packets with random payloads to be sent to certain MAC addresses on the network. This can either be done via the use of specialised hardware or using software. They are used for load testing of packet processing applications to test the amount of data which applications can process per second. This can reveal whether limitations on a system is software or hardware based.

Talk about pktgen module in kernel - does dpdk version use this

Pktgen is open source software tool, maintained by Intel, which aims to generate packets using the DPDK framework. It can generate up to 10Gbits of data per second with 64 byte frame size, and send the data in the form of packets across a compatible network interface card/controller. It has a number of benefits which include:

- Real time packet configuration and port control
- Real time metrics on packets sent and received
- Handles UDP, TCP, ARP and more packet headers
- Can be commanded via a Lua ?? script

how does pktgen work and why we used it

Pktgen has limitations though, since 10Gbps transmitting is only possible with 64 byte frames. Frames of greater size up to 1024 bytes can be transmitted although total bytes per seconds is reduced, therefore limiting the testing ability.

possibly
run more
ports on
more than
1 core?

1.2 Initial Testing

The initial testing of applications was carried out on a local Mac OS X machine running Ubuntu 14.04 LTS 64-bit on a VirtualBox virtual machine. Although this set-up didn't provide the ability to load test on very high speeds (anything above 1Gbit/s), it allowed for basic testing to check that the application was running as expected. Load testing of speeds up to roughly 700Mbit/s were also possible which have a basic testing platform without the need to move code to servers.

which
ones?

ref this
and
ubuntu

1.2.1 Set-up

Testing could be carried out using the 2 available 1Gbit NICs of the machine via a bridged network from the host to guest machine which severely reduced transmission speed. This allowed an ethernet cable to be looped back and connected between the ports, meaning anything transmitted via 1 port was guaranteed to be received by the other port.

Pktgen and the custom application were booted up simultaneously running in parallel. Careful memory allocation, port addressing and processor core assignment had to be carried out to stop shared resources impacting the overall performance of either application. This allowed Pktgen to send packets and the application to receive and process them.

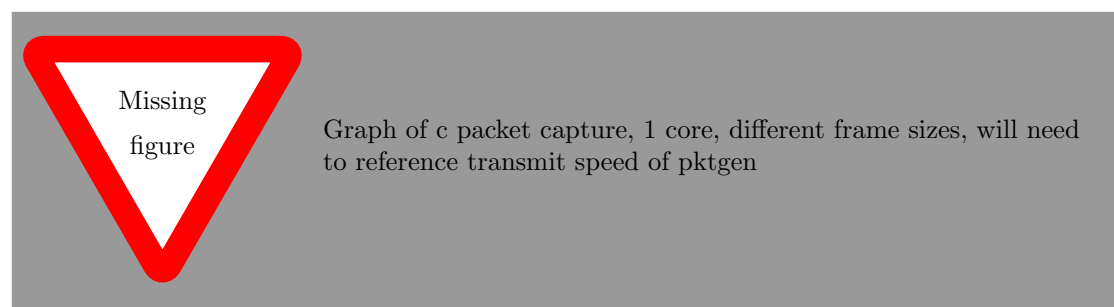
1.2.2 Methods

1.2.3 Results

1.3 Further Testing

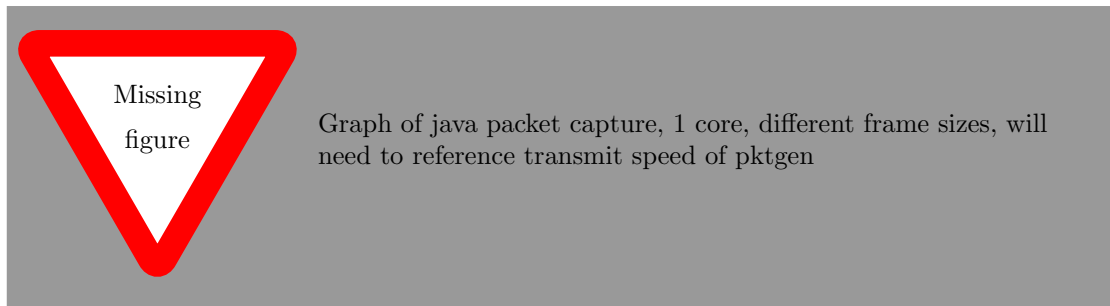
In order to fully understand the capabilities of the middleboxes, testing was carried out on Imperial College's Large Scale Distributed Systems (LSDS) testbed. Although this system consists of numerous machines, tests were carried out using just 2. The first machine was used to host the middlebox application and received the packets. The other machine was used as the client and ran the pktgen software allowing it to generate packets at up to 10Gbps in order to take advantage of the machines network interface controllers.

The first test on the LSDS testbed involved comparing the C and Java implementations of a packet capturing application which simply received the packets and freed them straight away without forwarding them. The C implementation is used to give the optimal readings possible from this and further tests, since very limited processing is carried out between receiving and dropping the packet. The figure below shows the performance of the application at varying packet sizes.



The same testing was carried out with an identical packet capture algorithm which was instead implemented in Java using the DPDKJava framework described previously. These results are shown below.

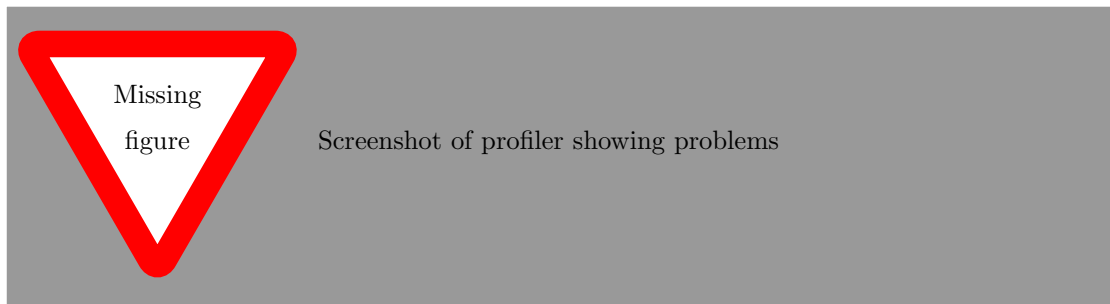
what does
the results
show



The results above are very low to what was expected for such a simple application.

expand on this

To check where the problems lied within the code, a Java profiler (JProfiler) was used to check multiple parts of the code including memory usage, cpu usage and the number of method calls and the average time per method call. This provided invaluable analysis of where the problems were. As can be seen by figure ??, the major problem lied with???



1.3.1 Set-up

1.3.2 Methods

1.3.3 Results

1.4 Software Design

Mention somewhere about the limitations of pktgen

1.4.1 Portability

1.5 Possible Improvement