1 Introduction

1.1 Motivation

As modern computing techniques advance, people are trying to find more generic solutions to problems which have been solved by native applications in the past. A main area of focus has been network middleboxes which are developed to process network packets. Common examples of middleboxes are firewalls, network address translators (NATs) and load balancers, all of which inspect or transform network packets in the middle of a connection between a public and private network. In the past, middleboxes have been purpose built hardware and software solutions which perform one task, but do this task extremely well. However, looking forward for network architecture, constant advancements such as 4G and the start of development for 5G for mobile communication has given rise to an ever increasing internet usage. This increase in data traffic provides a problem to the purpose built solutions which don't scale well to new technologies and increase demand. In recent years, people have been developing a number of programmable middleboxes which allow these generic solutions to be used on a wide scale basis.

As middleboxes are mainly used for networking purposes, they are required to process network packets at line rate (i.e. at speeds which allow packets to be processed as they are received) which generally is 10Gbps (gigabits per second), but speeds can reach 40Gbps and even 100Gbps. Performance increases for networking hardware have vastly outshone those of the software, opening up a new area of research to find new techniques to overcome this hurdle. This requires the application to retrieve the packet from the network line, inspect and transform the packet in the desired way and then insert the packet back onto the network line, all within a time period sufficient enough to not cause a backlog or dropped packets. High performance implementations of such applications are available, but are written in native languages, predominately in C/C++. However, more and more high performance computing projects are been developed in Java and have succeeded in performing at similar speeds to C/C++ applications. These high performance projects typically utilise a distributed system, so it makes sense to have the middleboxes written in Java as well for easy scaling.

The main challenge is actually getting the I/O system for the Java application to run at line rate speeds, due to challenges with how the JVM (Java Virtual Machine) interacts with memory, computer's kernel and the network interface controller/card (NIC). Once this challenge has been overcome, there are no reasons why programmable middleboxes written in none native languages such as Java can't exist within networking systems.

1.2 Contributions

This project contributions can be broadly split up into 5 categories:

- Research into existing Java technologies which can be exploited to dramatically increase throughput of data between a native language and one which runs on a virtual machine. Statistical analysis of these technologies and an evaluation of the suitability for the project.
- A new Java based fast packet processing framework named DPDK-Java. This is based upon an existing native framework but abstracts low level complications into an easy to use Java framework allowing quick implementation and deployment of applications.

- Multiple middlebox implementations using the fore-mentioned DPDK-Java framework focussing on the ease of use.
- Comparison testing between existing native middlebox implementations and the new Java based middleboxes. This focusses on analysing bottlenecks within the applications and suggested improvements
- Abstracting away key components and techniques of fast packet processing to give rise to more generic implementations using different languages.

1.3 Potential Applications

standard high volume (shv) servers - aim to deliver services much more rapidly and give network which is scaliable and flexible. aim for software to be run on shv servers which can be scaled and are flexible. should work on VM

relate back to quick boot up vm servers on cloud

1.4 Report Structure

Section **??** provides background information for a number of components which are part of the networking model and are useful to gain a full understanding of implementation details. The Java Virtual Machine and the Java Native Interface are explained in detail in sections **??** and **??** respectively and are needed to fully understand the techniques discussed. Finally, related works and applications are mentioned while briefly touching on performance testing which is used throughput the report.

As DPDK was chosen for the framework to be utilised, section **??** looks further into the implementation, most of which will be abstracted into the Java framework later on. Key concepts surrounding what enhances the performance over standard frameworks allow for techniques to be used later.

Section **??** primarily focusses on testing certain techniques and outlines the general design consideration needed to implement the new framework. The actual implementation is discussed in section **??**, which also touches on the middlebox applications as well.

The new framework performance is fully compared to that of native applications in section **??**. It looks through analysis of the applications performance in terms of memory and CPU usage and improves upon the initial implementation in a number of key ways to aim for matched performance.

Finally, a conclusion is drawn on the effective of the new framework in section **??** and touches upon future works surrounding this project. An abstraction of key concepts is drawn from the full report in the later stages to allow for generic implementations regardless of language choice.