Hardware Offloading for Connection Tracking

AMANDA NEE ANDREAS KINNUNEN EMILIA BLIDBORG
JOHAN EDMAN RAMAN SALIH
VIGNESH PURUSHOTHAM SRINIVAS
anee | kinnu | blidborg | jedma | ramans | vips @kth.se

March 29, 2022

1 Background

In recent years Network Function Virtualization (NFV) has developed as a type of network architecture that is employed to allow for better flexibility and scaling. Examples of network functions that commonly are virtualized are network security systems, such as firewalls and IDS, accelerators, and load balancers. However, as network bandwidths continue to increase, the central processing units (CPUs), that are generally doing most of the heavy lifting in packet processing, start to become bottlenecks - and processing them at line rate has become a difficult task in most network systems.

A common way to reduce the load on general computing units, such as the CPU, is to implement the functionality directly in custom silicon. Specific tasks such as compression, cryptography, and longest prefix matching (LPM) can then be offloaded from the CPU and performed in the specialized hardware. The main benefit of this is that the CPU does not have to be involved to such a large extent, if at all. In many cases, it often leads to computations to be performed at a higher rate as well.

One virtualized network function of particular interest is load balancing. A load balancer has the task of spreading incoming network packets to multiple destination servers so that they are all, to the best extent possible, evenly divided. In stateful load balancing, the system must keep track of each connection established to assure that packets are forwarded correctly and that flows remain coherent. Connection tracking has proved to be an expensive task [1, 2, 3] with the risk of being bound by the CPUs processing speed at high network speeds. Network interfaces with the support to perform such features have recently become available, but it remains to see how effective they are at offloading the CPU.

2 Problem statement

The project aims to investigate the performance gain available from offloading some of the flows coming through a NFV load balancer to the underlying network interface controller (NIC). First, a stateful load balancer will be implemented in FastClick [4], and secondly, it will be extended with the functionality to offload flows to the underlying NIC. Finally, the performance difference between the two different approaches will be compared, to understand what there is to gain by offloading traffic flows to hardware.

3 Problem

Connection tracking is an essential functionality that many network services and applications rely on to classify and maintain coherent packet flows. While the CPU installed on the load balancer are flexible and can handle such tasks, the computational resources required for the ever increasing network speeds and amount of individual packets do not scale well in terms of load.

4 Hypothesis

Offloading connection tracking to hardware will be a more effective way to reduce CPU load whilst maintaining, or increasing, performance (i.e. throughput, latency) in stateful load balancers.

5 Goals

- Develop test suit for load balancer. Including traffic generator, basic forwarder and scripts to run unsupervised tests and gather the resulting data.
- Benchmark latency and throughput of the load balancer and its classifier based on CPU core count, hash table size and number of flows.
- Implement hardware offloading and move connection tracking from the software load balancer to the hardware.
- Examine the latency and throughput based on CPU core count, hash table size and number of flows.
- Develop a technique to classify flows as elephant, or mice flows [5], and offload them accordingly.
- Develop and benchmark techniques to optimize hardware offloading, with the goal of improving, and optimizing, connection tracking performance.

6 Measurable objectives

The difference in performance between a load balancer working purely in FastClick, and a load balancer that utilizes the NICs capabilities for offloading traffic classification.

7 Evaluation techniques

The collection of the measurements will be done on three different implementations of the load balancer, the purely software based one, the naive hardware, and optimized hardware. The measurements and data needed is the following:

- The difference in latency and throughput when changing the number of processing cores.
- Drop rate when changing traffic rate.
- Drop rate and throughput when changing the packet size.
- Drop rate and throughput when changing the unique number of flows.

8 Deliverables and deliveries

- The project website
- The project plan
- A load balancer in FastClick
- A traffic generator in FastClick
- A basic forwarder in FastClick
- Automated testing scripts
- Midterm Progress Report
- Hardware offloading of classification to the NIC
- Performance differential evaluation
- Video presentation
- Final report
- Final presentation
- Lessons learned document

9 Approach

The project will be conducted on remote servers running Linux. We will make shell scripts for automating environment setup and testing, and C++ for programming and extending of program functionality. GNUPlot will be used for plots and diagrams, Git will be used for version control, and Overleaf for documents with some exceptions that will be handled in Google Sheets.

10 Tasks

Before any actual project work can be undertaken, the initial task will be to establish a plan for the project. The project itself can then be divided into a set of different phases.

The initial phase will consist of setting up, configuring, and becoming familiar with the environment that will be used. That is, configure the two servers that will be implemented as the load balancer and traffic generator in this project. The Device Under Testing (DUT) and traffic Generator (TG), as we call them will be set up and verified of their functionality. Ultimately, the latter step will be automated.

After basic functionality has been configured and verified, the step that follows is to install the software stack that will be used. Namely, Data Plane Development Kit (DPDK) and FastClick on the respective host. This ends the initial phase.

Once the initial phase is done, development can begin. This will include three programs, a basic forwarder, a traffic generator, and a load balancer. The software will be developed with the help of tests to verify functionality. The goal for the traffic generator will be to achieve high throughput and ideally to saturate the 100 Gbps link. For the forwarder, the goal is to be able to return traffic at same rate as is sent to it. Additionally the load balancer should have easily changeable parameters to make automated testing possible. This will enable different configurations and their performance to be easily measured.

Furthermore, once the implementations are done the hardware offloading features that exists within the hardware will be taken into account.

In conjunction with these tasks, scripts for gathering, analyzing, and plotting statistics will be considered; so that different tweaks easily can be measured, compared, and validated against the research project goal.

11 Method

All measurements and evaluations will be performed by adapting an empirical research method and done in a reproducible way. Data will be collected to support or reject the given hypothesis.

12 Gantt diagram and milestone chart (time schedule)

The tasks presented for the project have been assigned periods during the 18 weeks the project will run. This is visualized by a Gantt diagram presented in Appendix A.

13 Risk analysis

Some avenues for project failure include:

- **Hardware:** Setup and remote access to the hardware. Additionally, sharing and accessing the same test bed among teammates can pose some risks of slowing down progress.
- **Traffic Generator:** Designing and implementing the traffic generator that operates at 100 Gbps is a challenging task and might take more time than it should.
- Unforeseen bugs and implementation challenges taking up much more time than there is available.
- Hardware off-loading may never see an optimal configuration point for all type of flows.

14 Unique contribution of the team members

All members know a little bit of everything but some are more familiar with some software than others, for instance, FastClick and DPDK. The tasks will be divided based on familiarity or interest. Everyone will be programming and contributing to each task to some extent.

15 Dissemination of results and expected impact

There will be two types of deliverables, course-specific and more public. The course-specific are listed under section 8, The deliverables encompass the project plan, reports, presentations, and a website and source code. The public deliverables are code and a report. The code will be submitted as a pull request to the FastClick project.

References

[1] Daniel E Eisenbud et al. “Maglev: A fast and reliable software network load balancer”. In: *13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16)*. 2016, pp. 523–535.

[2] Rohan Gandhi et al. “Duet: cloud scale load balancing with hardware and software”. In: *SIGCOMM*. 2014.

[3] Anuj Kalia et al. “Raising the Bar for Using GPUs in Software Packet Processing”. In: *Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation*. 12th USENIX Symposium on Networked Systems Design and Implementation. usenix, 2015. ISBN: 978-1-931971-218. URL: https://www.usenix.org/system/files/conference/nsdi15/nsdi15-paper-kalia.pdf.

[4] Cyril Soldani Barbette Tom and Mathy Laurent. “Fast userspace packet processing”. In: *2015 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*. IEEE. 2015, pp. 5–16. DOI: 10.1109/ANCS.2015.7110116.

[5] Mohammad Al-fares et al. “Hedera: Dynamic flow scheduling for data center networks”. In: *In Proc. of Networked Systems Design and Implementation (NSDI) Symposium*. 2010.

A Gantt Chart

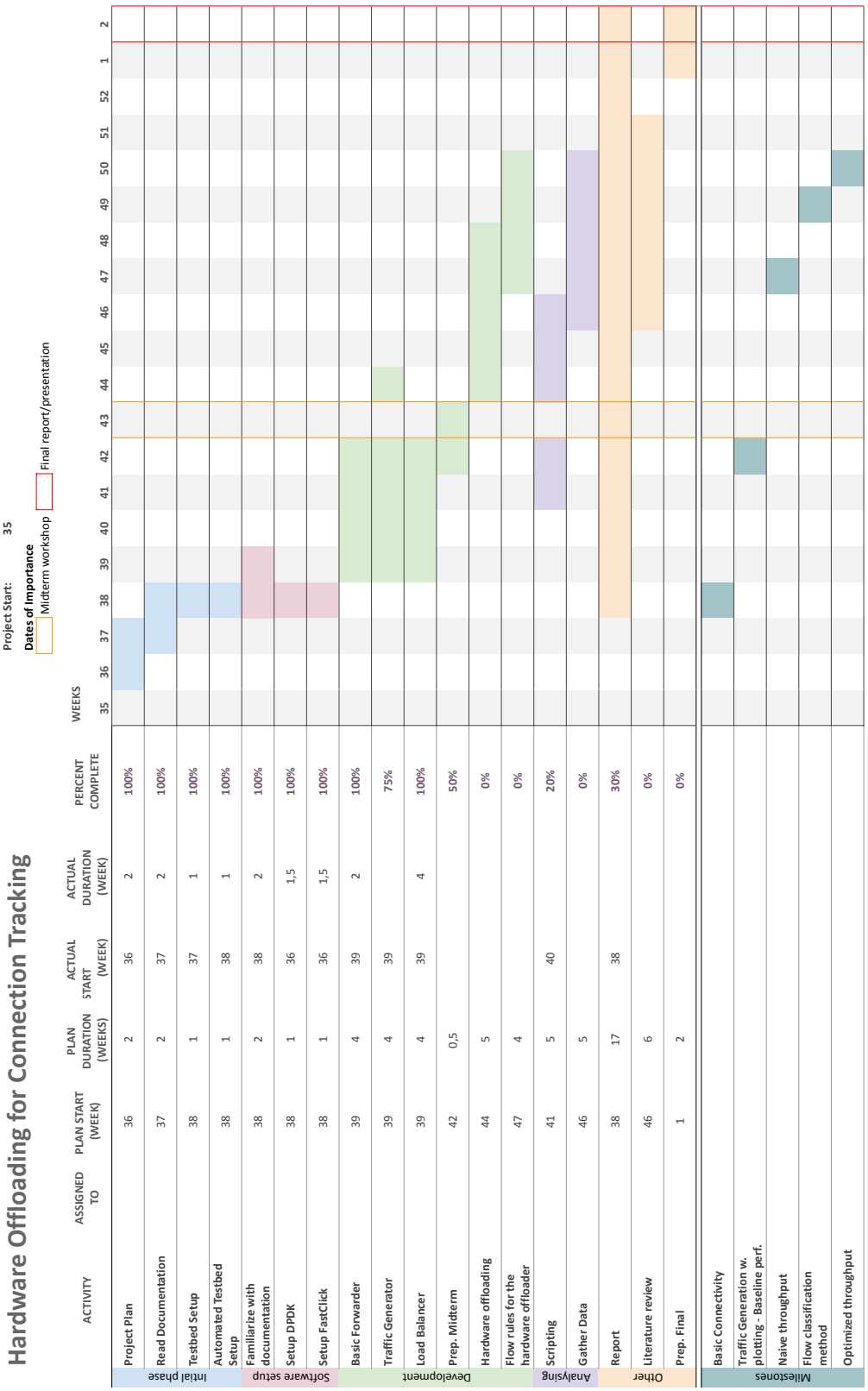

Figure 1: Gantt Chart

B Acronyms

CPU central processing unit

DPDK Data Plane Development Kit

DUT Device Under Testing

LPM longest prefix matching

NFV Network Function Virtualization

NIC network interface controller

TG traffic Generator