Thanh Bui
Downing College
ttb29

Computer Science Tripos - Part II Project

An accelerated, network-assisted TCP recovery

May 17, 2019

Proforma

Name:	**Thanh Bui**
College:	**Downing College**
Project Title:	**An accelerated, network-assisted TCP retransmit**
Examination:	**Computer Science Tripos – Part II, June 2019**
Word Count:	**1587**[1]
Project Originator:	Dr Noa Zilberman
Supervisor:	Dr Noa Zilberman

Original Aims of the Project

This project investigates the usage of a programmable switch to assist the Transmission Control Protocol fast retransmit

Work Completed

All that has been completed appears in this dissertation.

Special Difficulties

Learning how to incorporate encapulated postscript into a LaTeX document on both Ubuntu Linux and OS X.

[1] This word count was computed by `detex ttb29_dissertation.tex | tr -cd '0-9A-Za-z \n' | wc -w`

Declaration

I, Thanh Bui of Downing College, being a candidate for Part II of the Computer Science Tripos, hereby declare that this dissertation and the work described in it are my own work, unaided except as may be specified below, and that the dissertation does not contain material that has already been used to any substantial extent for a comparable purpose.

SIGNED DATE

Contents

List of figures **5**

1 Introduction **7**
1.1 Motivation 7
1.1.1 Background 7
1.2 Problem Outline 8
1.3 Related Work 8

2 Preparation **10**
2.1 Transmission Control Protocol 10
2.1.1 Test 10
2.2 Software Used 10
2.2.1 Programming Languages 10

3 Implementation **11**
3.1 The SimpleSumeSwitch Architecture 11
3.2 The Cache Queue 11

4 Evaluation **12**

5 Conclusion **13**
5.1 Results 13
5.2 Lessons Learnt 13
5.3 Future Work 13

Bibliography **14**

List of figures

1.1 The standard convention of TCP handling. 7
1.2 The proposed TCP handling. 8

Acknowledgements

This document owes much to an earlier version written by Simon Moore [**?**]. His help, encouragement and advice was greatly appreciated.

Chapter 1

Introduction

1.1 Motivation

1.1.1 Background

Currently, when there is packet loss, signaled by duplicate ACKs (DUP ACKs), the DUP ACKs will traverse all the way back to the host (**Figure 1.1**). The host receives the DUP ACKs, then resends the packet. The goal of my project is to implement a fast TCP recovery mechanism, which aims to reduce the response time to DUP ACKs and reduce changes to the congestion window by retransmitting packets from within the network (e.g., from the switch), instead of sending the DUP ACKs back to the host (**Figure 1.2**). The implementation will be based on KVS concepts, where the key is the flow Id and sequence number, and the value is the packet.

For this project, I will be using the P4 programming language [1]. It is a language designed to allow the programming of packet forwarding planes. Besides, unlike general purpose languages such as C or Python, P4 is domain-specific with a number of constructs optimized around network data forwarding, hence is well suited to such a network application.

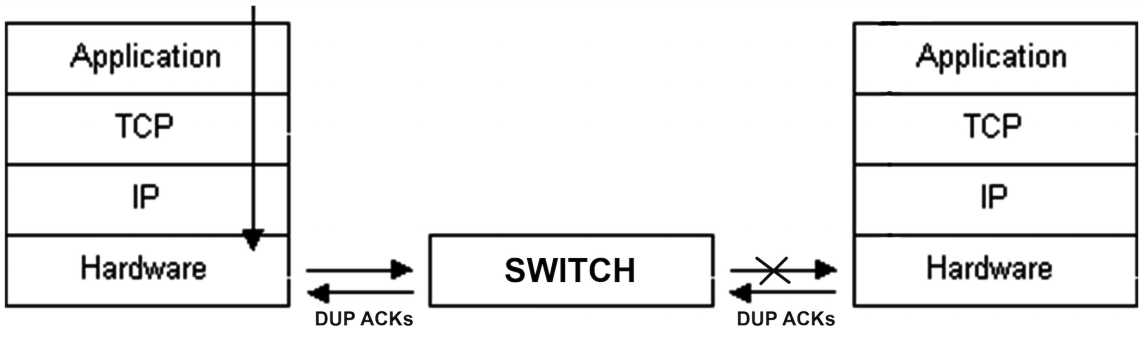

Figure 1.1: The standard convention of TCP handling.

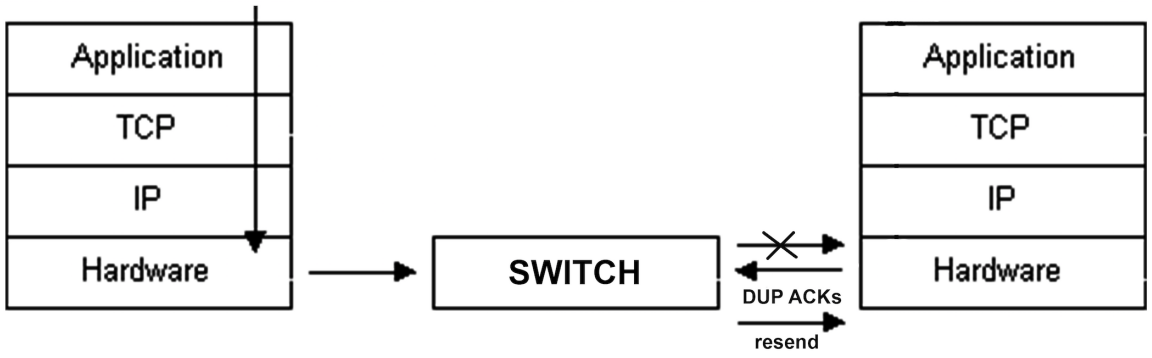

Figure 1.2: The proposed TCP handling.

1.2 Problem Outline

1.3 Related Work

Test

Chapter 2

Preparation

In this chapter, I discuss the Transmission Control Protocol in more dteail. This is followed by a discussion of sftware deliverables, formal requirements, existing libraries and tools levarage, workflow and starting point.

2.1 Transmission Control Protocol

2.1.1 Test

2.2 Software Used

2.2.1 Programming Languages

I used **P4** and **Python** for this project.

In this project, I mainly used the **P4** programming language [1]. It is a language designed to allow the programming of packet forwarding planes. Besides, unlike general purpose languages such as C or Python, P4 is domain-specific with a number of constructs optimized around network data forwarding, hence is well suited to such a network application.

Python:

I also made use of the `make` build automation tool to automate project builds, tests and benchmarks.

2.2.2 P4-NetFPGA Platform

a

2.2.3 The NetFPGA SUME board

2.2.4 Development Environment

Git: I used `git` for the

2.3 Starting Point

This project uses the knowledge about TCP introduced in the Part IB *Computer Networking* course.

During the development of this project, I made use of the materials covered in the following Part II and Part III course:

- *Principle of Communications* — Test
- *High Performance Networking* — ;
- *LaTeX and Matlab* — typesetting the project proposal and dissertation.

All code was, using the . Apart from

In terms of , I had little prior experience with P4 Programming Language and the P4-NetFPGA framework. I had significant prior experience in Python and Git.

2.4 Requirements Analysis

Empty for now

Chapter 3

Implementation

3.1 The SimpleSumeSwitch Architecture

This project will work mainly with a NetFPGA SUME board [2], using P4 programming language. I will be using the P4-NetFPGA workflow, which provides infrastructure to compile P4 programs to NetFPGA [3]. Apart from that, everything else will be built from scratch.

I have no prior experience with either NetFPGA or P4, but this will be mitigated through self-learning in which I will make use of the online tutorials, Google's resources and the P4 community documentation, as well as the experience of my supervisors.

3.2 The Cache Queue

The relevant Tripos courses that can serve as a starting point for this project are primarily: *Computer Networking*, *Principles of Communications* and *ECAD and Architecture Practical Classes*. Since the courses are introductory, I will also consult Part III's *High Performance Networking* course. I also plan to bridge any knowledge gap through extensive personal reading as well as help from my project supervisor.

Chapter 4

Evaluation

Tests

Chapter 5

Conclusion

5.1 Results

5.2 Lessons Learnt

5.3 Future Work

Bibliography

[1] "The P4 Language Consortium," https://p4.org/.

[2] N. Zilberman, Y. Audzevich, G. A. Covington, and A. W. Moore, "NetFPGA SUME: Toward 100 Gbps as research commodity," *IEEE micro*, vol. 34, no. 5, pp. 32–41, 2014.

[3] "NetFPGA/P4-NetFPGA-public," https://github.com/NetFPGA/P4-NetFPGA-public/wiki.