Oliver Black

Software IPv6 Router in Rust

Computer Science Tripos – Part II

Selwyn College

April 24, 2019

Declaration of Originality

I, Oliver Black of Selwyn 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 Oliver Black

Date April 24, 2019

Proforma

Name:	**Oliver Black**
College:	**Selwyn College**
Project Title:	**Software IPv6 Router in Rust**
Examination:	**Computer Science Tripos – Part II, July 2001**
Word Count:	**"FIX ME& footnote"** [1]
Project Originator:	Oliver Black & Dr Richard Watts
Supervisor:	Andrew Moore

Original Aims of the Project

The IPv6 standard contains a large number of complex requirements, complicating understanding. I aim to design and implement a simple IPv6 router using Rust[1] that behaves as specified in the IPv6 RFCs[3]. This router should implement the minimum functionality required by the relevant standards, yet still be functional, minimal, & stable. Rust is a new programming language that aims to be as fast as C while maintaining memory safety, I wanted to understand how practical it was to develop in.

Work Completed

Despite having initial difficulties setting up my test environment using Mininet[2], due to its lack of support for IPv6, I successfully implemented a functioning IPv6 router in rust that met almost all of my core requirements. Both the router itself, and the test bench, are available for public use. TODO mention speedup/code coverage/size/RFC coverage/throughput AND fill rest in when main body done. mention how ambitious original claims were

Special Difficulties

None. TODO fill in

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

Contents

List of Figures

Acknowledgements

Many thanks to:

- My supervisor Andrew Moore for his helpful advice.
- My Director of Studies Dr Richard Watts for his guidance.
- Friends & family for proofreading.

Chapter 1

Introduction

Slowly but surely the internet is making progress towards IPv6, and how do pages of Requests for Comments (RFCs) translate into real world network components. The aim of this project was to develop an IPv6 Router in Rust that explores the functionality of IPv6, and how different parts of the various standards fit together. The project has been a success, I have produced a functioning router and accompanying test suite.

Due to the popularity of the Internet, there are now not enough IPv4 addresses to go around. IPv6 is the incoming internet addressing standard that solves numerous issues with IPv4. Primarily it increases the number of addresses, however it also fixes many flaws in the IPv4 design, and standardises common non-standard practices. For example, the Time To Live in IPv4 was defined partly in terms of seconds left to live[5], but in practice was decremented by 1 every hop between nodes. In IPv6 the field is accurately renamed to Hop Limit, and is now defined in terms of hops between nodes (as opposed to seconds). Many subtle decisions like this have gone into the IPv6 standard, with an aim to making an internet that works well, rather than one that just works.

Rust[1] is an up and coming modern low level programming language. It aims to match the performance of C++ without sacrificing memory safety, and avoiding garbage collection. It does this through zero-cost high level abstractions such as *ownership* and *lifetimes*. For example, if you pass a struct to a function, that function then owns that struct, with it being inaccessible after the function returns. It is possible for functions to borrow values instead, using "`&`", similar to passing by reference. I chose Rust for my project as it can be easier to debug than C or C++, but mainly because I was interesting in learning Rust.

Mininet[2] is an open source virtual network simulator that was developed at Stanford and until 2016 was used in the Part 1B Computer Networking course, it is written in Python. It creates lightweight virtual networks by making use of Linux's *networked namespaces*, allowing processes to share a kernel, yet be behind different network interfaces. This made it the ideal candidate to build my router and test suite on top of.

Routers are the backbone of the internet, at the most simple level many of them have a *control plane* that deals with addressing, and a *forwarding plane* that deals with sending packets. There are many open source routers out there, but almost all of them have lots of IPv4 code. This makes it difficult to isolate and understand how the IPv6 part actually works. Starting from scratch allows you to avoid having to deal with IPv4 at all. Using the IPv6 standard as a framework, combined with some knowledge about the internals of routers, it is possible to develop and IPv6 router that is stable, small, simple, & fast.

Chapter 2

Preparation

Before starting the implementation lots of research

2.1 Exploration

2.2 Design

2.3 Test Plan

This chapter is empty!

Examples Mininet: Ipv4 - see below Simple Router: https://github.com/mininet/mininet/wiki/Simple-Router

Design RFCs Router knowledge - diagram

Test bench Test criteria (appendix) - spreadsheet More detail in implementation on mininet

Chapter 3

Implementation

Chapter 4

Evaluation

4.1 Printing and binding

Use a "duplex" laser printer that can print on both sides to print two copies of your dissertation. Then bind them, for example using the comb binder in the Computer Laboratory Library.

4.2 Further information

See the Unix Tools notes at

`http://www.cl.cam.ac.uk/teaching/current-1/UnixTools/materials.html`

Chapter 5

Conclusion

I hope that this rough guide to writing a dissertation is LaTeX has been helpful and saved you time.

Bibliography

[1] Rust, a modern low level programming language, https://www.rust-lang.org/

[2] Mininet, a network virtualisation library in Python, http://mininet.org/

[3] Internet Protocol, Version 6 (IPv6) Specification, RFC 8200, July 2017

[4] IP Version 6 Addressing Architecture, RFC 4291, February 2006

[5] INTERNET PROTOCOL DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION, RFC 791, September 1981

[6] Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification, RFC 4443, March 2006

Appendix A

Latex source

A.1 diss.tex

A.2 proposal.tex

Appendix B

Makefile

B.1 makefile

B.2 refs.bib

Appendix C

Project Proposal

Part II Project Proposal

Software IPv6 Router in Rust

2018-10-11

Oliver Black (olb22) - *Selwyn College*

Overseers - *Jean Bacon / Ross Anderson / Amanda Prorok*
Supervisor - *Andrew Moore*
Director of Studies - *Richard Watts*

Originator - *Oliver Black / Richard Watts*

Introduction & Description of Work

IPv6 is the next generation internet addressing standard, it solves numerous problems with IPv4, such as the shortage of IPv4 addresses. It does more than just increase the number of addresses though, many advanced features not in IPv4 are added with IPv6, full details of the current IPv6 standard can be found in IETF RFCs [1.1]. Mininet [1.2] is a virtual network simulator (supporting IPv6) that was developed at Stanford and until 2016 was used in the 1B Computer Networking Course [1.3] (the year before I attended it).

This project will make use of Mininet to write a software implementation of an IPv6 router in Rust [1.4]. This will begin with gaining an understanding of how Mininet works, then doing research into the IPv6 specification, and finally developing a router and a test bench. Initially that router will implement a minimum amount of IPv6 functionality (as per the IPv6 standards), moving on to more advanced functionality if time permits.

Resource Declaration

No special resources will be required. Work will be undertaken on a personal laptop, with git used for source control, and github for cloud backup. This will enable work to be seamlessly continued on an MCS machine if the laptop is taken out of action.

Starting Point

I took the 2017 1B Computer Networking course, so have a good overview of what a router is meant to do. I have spent around 1 hour fiddling with Mininet, and reading up on the 2016 Computer Networking course, to check what I want to do is feasible. An implementation of an extremely simple router already exists as an optional extension of that course, and I have looked at it briefly [3.1]. I have attended a Rust talk at the CL, and have done my own brief research into the programming language. I have used continuous integration testing methodologies during my summer internship at Solarflare.

Substance & Structure of the Project

The aim of this project is to write a software IPv6 router that complies with the IPv6 RFC standard [4.1]. It'll begin by complying with all the 'must's mentioned in the main IPv6 RFC standard, with other aspects of the standard being extensions, see success criteria for more details.

The first stage of the project will be research and refining requirements, this will be two fold. On a practical level, understanding how the Simple Router [4.2] project works and interfaces with Mininet, also gaining a working knowledge of Rust. On a non-practical level, researching IPv6 and gaining a deeper understanding of the requirements listed in success criteria. From these requirements a detailed plan will be written, including the router's system architecture.

The next stage of the project will be development and testing. A test bench will be created as the project proceeds, with tests being added for each new requirement that is implemented. The test bench will take the form of scripts that set up a Mininet environment, and then have IPv6 nodes sending packets to each other and the router. All the tests will be run every time a feature is added, the result being a rudimentary form of manual continuous integration.

The development itself will primarily be in Rust[4.3], with bits of C and Python. Rust is a safe modern low level language, and will be an interesting learning experience, the project can always be completed entirely in C if there are insufficient libraries available, or the learning curve is too steep. Interfacing with Mininet at a high level (for the test bench) will be done in Python, as that is the language the API is written in. Exploratory work will be done in C due to preexisting code bases, such as the Simple Router example, being in C.

The final stage will be the verification of the test bench, with additional end-to-end tests being performed. This will be followed by an evaluation to demonstrate the implementation has been successful, and then by the writing of the dissertation.

Success Criteria

To have a software IPv6 router and accompanying test bench running on top of Mininet. The router will comply with all of the core requirements of an IPv6 router, and the test bench will test each of these requirements.

The core requirements are divided into two parts: basic and advanced. The basic requirements are an implementation of a control and data plane in software that can forward packets to the correct unicast addresses, with static address allocation. The advanced requirements are an implementation of ICMPv6 (i.e. proper error handling), and handling anycast and multicast addresses.

Testing is required for all of these, and will meet the success criteria if there is a unit test for each implemented bit of functionality and the relevant requirements listed in the specification, and if the router passes all these tests. This includes tests for edge cases, for example when TTL is 1.

The extension requirements are implementing IPv6 extension headers (both those included in the main IPv6 RFC, and those with their own RFC), implementing SLAAC & DHCPv6 (stateless and stateful), compatibility features with IPv4 addresses, and checking addresses comply with IPv6. Other stretch goals include optimisation, these will be tested through load testing and comparisons with non-optimised versions. Additionally any optional minor features encompassed by the core requirements are extension requirements. A further potential extension would be to pull the software router out of mininet, and get it running on a switch, testing it with real machines.

All of these requirements are based on the relevant RFCs[1.1]. Where applicable, the core requirements only include aspects of the RFC prefaced with ‘must’, whereas the extension requirements include all functionality specified.

Plan of Work

Note: Throughout, any work on extension goals can be replaced with work from a previous 2 week slot, making the plan more flexible and responsive to unexpected changes.

20th October

Start of project - Time since last entry/special events
<> - Work that is completed/stopped on this date
Kick off, begin research - Work that is begun on this date
Milestones: - List of milestones expected by this date

3rd November

2 weeks
Research completed.
Start implementation of core requirements.
Milestones:
List of core and extension requirements mapped out in detail from IPv6 standards and documentation, including system architecture.
Simple Router implementation and interface with Mininet understood.
Basic Rust concepts understood.

17th November

2 weeks
Basic routing framework completed.
Start working on the advanced core requirements.
Milestones:
Router software that can perform basic packet forwarding.
Test bench that can perform basic tests for packet forwarding.

1st December

2 weeks - End of Michaelmas term
Core criteria all met.
End-to-end testing begins, more comprehensive test cases to be added to test bench, small problems fixed as testing continues, big problems documented and listed. Implementation evaluated based on success criteria.
Milestones
Router software that meets all of the core success criteria, with accompanying test bench.

15th December

1 week - 1 week holiday
List of big problems completed.
Big problems fixed in order of severity.
Milestones:
List of remaining identified problems with core functionality.

29th December

2 weeks
All major issues with core functionality resolved.
Begin work on extension goals.
Milestones:
Stable router with comprehensive test bench covering all core functionality.

12th January

2 weeks - Start of Lent Term
Extension work suspended.
Begin evaluation and writing dissertation.
Milestones:
List of implemented extension functionality, with accompanying router software and test benches.

26th January

2 weeks - 1st February: Progress Report Deadline
Evaluation completed.
Continue writing dissertation, write a progress report for presentation.
Milestones:
Evaluation and test report.
(midway) Progress report completed and submitted

9th February

2 weeks
Draft dissertation completed, dissertation work suspended.
Resume work on extension goals.
Milestones:
Completed draft dissertation.

23rd February

2 weeks
Extension work suspended.
Resume work on dissertation - get friends to read.

Milestones:
List of implemented extension functionality, with accompanying router software and test benches.

9th March

2 weeks
<>
Based on feedback received begin work on weaknesses in dissertation.
Milestones:
Improved dissertation based on feedback received.
List of areas of weakness to be worked on.

23rd March

2 weeks - End of Lent Term
Areas of weakness resolved/explained.
Exam revision.
Milestones:
Dissertation submitted to overseers and supervisors for review

6th April

2 weeks
<>
Exam revision.

20th April

2 weeks - Start of Easter Term
<>
Alter dissertation based on comments from from overseers and supervisor
Milestones:
Completed Dissertation

4th May

2 weeks
<>
Exam revision

17th May

2 weeks - 17th May: Dissertation Deadline
<>
Dissertation reread and then submitted.
Milestones:
Submitted dissertation and source code.

Links

1.1: IPv6 https://tools.ietf.org/html/rfc8200
Addressing: https://tools.ietf.org/html/rfc4291
ICMPv6: https://tools.ietf.org/html/rfc4443
DHCPv6: https://tools.ietf.org/html/rfc3315
SLAAC: https://tools.ietf.org/html/rfc4862
Authentication Header: https://tools.ietf.org/html/rfc4302
Encapsulating security payload: https://tools.ietf.org/html/rfc4303
1.2: http://mininet.org/
1.3: https://www.cl.cam.ac.uk/teaching/1617/CompNet/handson/
1.4: https://www.rust-lang.org/en-US/
3.1: https://github.com/mininet/mininet/wiki/Simple-Router
4.1: 1.1
4.2: 3.1
4.3: 1.4