Name: Simonas Mulevicius
College: Homerton College
User Identifier: sm2354

Computer Science Tripos Part II Project Proposal

QUIC offloading using NetFPGA smart NIC

22nd October 2020

Project Originator: Dr Andrew W. Moore

Project Supervisor: Dr Andrew W. Moore

Signature:

Director of Studies: Dr John Fawcett

Signature:

Overseers: Dr Robert Mullins and Dr Marcelo Fiore

Signatures:

Introduction and Description of the Work

QUIC is a new transport layer protocol built on top of UDP. QUIC has numerous advantages compared to TCP: QUIC establishes connections faster, it aims to prevent ossification of network standards by encrypting network headers and QUIC's multiplexed connections don't suffer from the head-of-line blocking. Some implementations of QUIC provide a mechanism to fall back to TCP if QUIC is not supported by either of the communicating parties or an intermediate network infrastructure (e.g. firewalls). This detection mechanism is done in parallel, so QUIC should be no worse than TCP. Despite these advantages, for historical reasons, TCP has better hardware support than QUIC. For example, QUIC requires 350% more CPU cycles than TCP with TLS [1]. The reason for such a clear difference is that in the last years TCP/IP network stack was considered to be the backbone protocol of the Internet. As a result, multiple middleboxes and hardware accelerators were created to support and enhance TCP.

In contrast, QUIC doesn't have widespread hardware support as it is a new and developing standard. Consequently, there is a risk that because of this lack of support, QUIC may not be adopted. To mitigate this issue, I am planning to do partial QUIC offloading with smart NICs on the receiver's side. In other words, the goal of this project is to delegate some CPU intensive but primitive tasks to hardware, leaving more complex control operations to CPU.

There is an increasing workload pressure for CPUs as network throughput keeps increasing. Processors need to perform expensive context switch operations to handle incoming packets. For instance, CPUs may perform cryptographic operations on packets, and they usually reply with acknowledgements. Contention for CPUs can be relieved by delegating specific instructions to NICs (network interface cards).

Studies show that after encryption packet reordering is another bottleneck of QUIC[2]. My core goal is to create packet reordering logic in hardware using NetFPGA-SUME smart NIC. However, by design QUIC encrypts packet numbers so this project would require me to turn off encryption of QUIC. In case it turns out to be impossible, I am planning to offload packet pacing functionality to Net-FPGA SUME board.

If time allows, I am planning to implement additional offloading functions in hardware. For example, potential extension goal is to implement QUIC packet segmentation in hardware. Another additional task could be to offload QUIC encryption and/or decryption functions to hardware.

I am planning to provide some semi-random stimulus to the test logic. I would have to test that offloaded logic correctly deals with both correct and malformed QUIC packets. As a result, I should randomise payload of the potential packets to have representative inputs. Moreover, the random stimulus can be tested in both the application layer and in hardware (NetFPGA NIC). Application layer testing would require an end-to-end testing environment of two machines which would be using QUIC for communication. Random message contents or random lengths could be exchanged to ensure that my offloading logic does not introduce new errors. Hardware-level testing could be performed in isolation analysing the behaviour of FPGAs. This project is concerned with partial offloading so my offloading logic shouldn't make any complicated control decisions (e.g. CPU would have to deal with the complicated task of removing unsuitable QUIC packets). Hence, packet reordering should not change the size of incoming data. As a result, I could add additional supervising logic to ensure that this property is preserved.

Starting Point

This project will use insights from other researchers who investigated the possibility of QUIC offloading [2] and who actually offloaded cryptographic functions [7]. Moreover, a similar part II project was done in the past with a different transport layer protocol (TCP) [4]. My initial project will be built on top of the NetFPGA SUME template for building network interface cards (NICs) [6].

To simplify FPGA debugging process, I am planning to make use of numerous network debugging tools. First of all, qvis can be used to represent logs of QUIC graphically. Moreover, the NetFPGA SUME board uses Xilinx software which includes Vivado simulator. Finally, it seems that packet manipulation tool Scapy could be used for testing as well.

As far as network simulation is concerned, it seems that TLEM network emulator [8] could be used for integration testing. TLEM has already been used in the paper which compared different implementations of QUIC [2]. Hence, I assume that I could also make use of this emulator.

Essential Tasks to be completed

The core goal can be split into multiple sub-tasks:

- First of all, I will need to construct an initial hardware component

which could retrieve the relevant information (such as connection ID) from QUIC headers. According to [3], all the QUIC packets, including packet numbers, are encrypted. Hence, I would need to find a mechanism which would allow me to turn off encryption. If that doesn't work, then I will have to create my own simplified version of unencrypted QUIC protocol. For this reason, I can't tell which implementation of QUIC I will be using during the development time.

- Later on, relevant packet information will be used to group and place packets into the buffers. Each connection should have its own dedicated buffer. It turns out that there is no buffer memory manager, so I would have to create one myself. Moreover, I anticipate that I would have to implement an additional layer which would enable communications between the current QUIC implementations and my underlying offloading hardware. This task would require special attention when dealing with concurrency. Hardware analogues 'of test-and-set' instructions or spin-locks might be useful here.
- Additionally, I would need to create a timestamp generator and/or some 'garbage collector' so that I could remove lost or outdated packets. I found that a former part II student observed abnormal behaviour of Enthernet's TX and RX clocks in his Part II dissertation [4]. My current knowledge allows me to assume that this issue is irrelevant as I would be using a separate system clock.

Extended goals

If time allows, I should implement the following tasks:

- offload QUIC packet segmentation to hardware
- offload QUIC encryption and/or decryption to hardware

Success Criteria

To demonstrate the success of my project, I would have to test my implementation of packet reordering mechanism. To do this, I will have to write unit tests which would show that my logic is capable of dealing with different (e.g. non-standard or corrupt) packets. If it turned out that it was impossible to overcome QUIC encryption, but I implemented and tested packet pacing logic, then my project would still be considered a success.

In addition to numerous unit tests, I could also use interoperability tests to ensure that offloaded core functions of QUIC still work. Unfortunately, I am planning to use QUIC in a non-standard way (without encryption), so interoperability tests would not be relevant.

A further success criterion is to obtain quantitative performance metrics of my implementation. I should compare two network stacks: one which uses QUIC and another which uses QUIC with hardware offloading. It will be sufficient to measure delay, throughput and the number of CPU cycles consumed by each stack when sending data between two connected machines. Also, associated costs of offloading (used FPGA area and processing time of packets) should be included in the final report. I expect that this experiment would demonstrate that my implementation could reduce the number of required CPU cycles to process QUIC packets.

However, it is highly likely that clocks of different machines will be skewed. Even the smallest difference in time could change performance metrics. To mitigate this issue, I could change the message path – all the packets would have to go from sender to receiver and then back again to the sender. Alternatively, I would have to measure an upper boundary for the clock skew, and I would have to take it into account when reporting my findings.

Timetable and Milestones

In order to be able to track my progress, I am planning to split my work into slots of 14 days. It seems reasonable to start each work item on Saturday and finish it after two weeks on Friday because various deadlines will be on Fridays.

Slot 1 - *24th October – 6th November*

Tasks

- Create a testing environment where two processes on different Virtual Machines would exchange QUIC packets.
- Finish tutorial (provided by https://netfpga.org/) about NetFPGA SUME boards.
- Obtain required special hardware and set up a development environment.
- Read papers about offloading techniques used for TCP and QUIC.

- Find a mechanism to turn off QUIC encryption or find QUIC implementation which doesn't have it. If that turns out to be impossible, then I would have to propose minimal requirements to implement my own version of QUIC.

Milestone

- Produce and send a report to the project supervisor about the key functionality of NetFPGA SUME boards, TCP and QUIC offloading and propose a strategy to get rid of QUIC encryption. *Due 6th November.*

Course work deadlines

- Unit of assessment practical work deadline – 27th October.

Slot 2 - *7th November – 20th November*

Tasks

- Remove the encryption from QUIC packets. This can be done by either using QUIC an implementation which allows that, adjusting encryption module or creating a minimal version of my own QUIC implementation which doesn't have encryption.
- Create hardware component which would retrieve packet number.
- Write unit tests for this hardware component.
- (Extra) Study documentation of Vivado simulator and packet manipulation tool Scapy.

Milestone

- Remove encryption and implement simple hardware module which extracts QUIC packet numbers.
- (Extra) Create a short report about Vivado simulator and Scapy. *Due 20th November.*

Course work deadlines

- Unit of assessment practical work deadline – 10th November.

Slot 3 - *21st November – 4th December*

Tasks

- Finish removal of encryption if there are any unexpected difficulties.

- Check if it is possible to use Xilinx software licences outside the Cambridge network.
- Set up a working environment so that it could be accessed remotely.

Milestone

- Have a prepared remote working environment. Moreover, at this point, I should have removed encryption from QUIC and implemented simple hardware module which extracts QUIC packet numbers.
  Due 4th December.

Course work deadlines

- Unit of assessment report submission deadline – 3rd December.

Slot 4 - *5th December – 18th December*

Tasks

- Implement Verilog block which would generate timestamps.
- Write Verilog code which would store QUIC packets in registers.
- Implement unit tests which would ensure the correctness of these blocks.
- Read about the concurrent-safe hardware operations available in Net FPGA SUME boards.

Milestone

- Have a working Verilog code which would order QUIC packets.
  Due 18th December.

Slot 5 - *19th December – 1st January*

Tasks

- Start working on the integration of QUIC offloading with actual QUIC protocol.
- Continue developing previous Verilog items if I encounter any unexpected difficulties.

Milestone

- Have a working Verilog code which would order QUIC packets. Also, at this point, I should have successfully integrated QUIC offloading with QUIC protocol. Also, I should send an update of my progress to the supervisor.
  Due 1st January.

Slot 6 - *2nd January – 15th January*

Tasks

- Catch up with any unexpected difficulties.

Milestone

- Send an update of my progress to the supervisor.
  Due 15th January.

Slot 7 - *16th January – 29th January*

Tasks

- Create end-to-end integration testing environment.
- Create simulation environment which would delay and reorder packets according to the specified parameters.
- Start measuring offloading performance.
- If time allows, then start working on extensions.
- Continue measuring offloading performance.

Milestone

- Measure throughput and latency of offloaded QUIC protocol using different size packets and network conditions.
- Measure CPU usage of the offloaded QUIC protocol.
- Measure consumed FPGA area and processing time of packets.
  Due 29th January.

Slot 8 - *30th January – 12th February*

Tasks

- Start working on the progress report and its presentation.

Milestone

- Send progress report before the deadline.
- Finish progress report presentation.
  Due 12th February.

Deadlines

- Progress Report Deadline – 5th February, 12 noon.
- Progress Report Presentations – 11th, 12th, 15th or 16th February, 2:00 PM.

Course work deadlines

- Unit of assessment assignment deadline – 1st February.

Slot 9 - *13th February – 26th February*

Tasks

- If time allows, continue working on extensions.
- Otherwise, perform more evaluation tests.

Milestone

- Send progress update to the project supervisor.
  Due 26th February.

Course work deadlines

- Unit of assessment assignment deadline – 15th February.

Slot 10 - *27st February – 12th March*

Tasks

- If time allows, write unit tests for extensions.
- Otherwise, finish evaluation testing.

Milestone

- Send conclusions of evaluation to the project supervisor.
  Due 12th March.

Course work deadlines

- Unit of assessment report deadline – 12 March.

Slot 11 - *13th March – 26th March*

Tasks

- Write a chapter about introduction and preparation.

Milestone

- Send introductory chapters to the project supervisor.
  Due 26th March.

Slot 12 - *27th March – 9th April*

Tasks

- Adjust previous chapters to the comments from the supervisor.
- Write a chapter about implementation.

Milestone

- Send implementation chapters to the project supervisor.
  Due 9th April.

Slot 13 - *10th April – 23rd April*

Tasks

- Adjust previous chapters to the comments from the supervisor.
- Write evaluation chapter.

Milestone

- Send draft version of dissertation to the project supervisor and Director of Studies.
  Due 23rd April.

Slot 14 - *24th April – 7th May*

Tasks

- Adjust dissertation to the comments from the supervisor and Director of Studies.

Milestones

- Send final version of dissertation to the supervisor and Director of Studies.
- Submit dissertation before the deadline. *Due 7th May.*

Slot 15 - *8th May – 14th May*

Tasks

- Ensure that the dissertation is submitted a few days before the deadline.

Milestone

- Have a submitted dissertation.
  Due 14th May.

Deadlines

- Dissertation Deadline – 14th May, 12 noon.
- Source Code Deadline – 14th May, 5:00 PM.

Resources Declaration

This project requires specific hardware resources:

- **Two NetFPGA SUME boards and all the required licences**
  Contact person: Dr Andrew W. Moore (andrew.moore@cl.cam.ac.uk)
  These smart NICs have programmable FPGAs which would allow me to implement packet reordering logic in hardware. Moreover, these boards have high throughput (1-10 Gbps), and there is a large NetFPGA community of developers. According to [5], 'Current NetFPGA work is licensed under LGPL 2.1'.
- **Two pre-build NetFPGA Cube systems**
  Contact person: Dr Andrew W. Moore (andrew.moore@cl.cam.ac.uk)
  These integrated computer systems will substantially simplify the development process as they combine NetFPGA SUME boards with motherboards and additional hardware components. Moreover, these cube systems will reduce the risk of physical damage.
- **Connecting Ethernet cables**
  Contact person: Dr Andrew W. Moore (andrew.moore@cl.cam.ac.uk)

I am planning to use my own personal computer because of convenience and familiarity. This is a dual-boot system with both Windows 10 Pro and Ubuntu 18.04. My computer has Intel(R) Core(TM) i7-8550U CPU whose working frequency is 1.80GHz. 8 GB amount of RAM is dedicated for Windows partition, and there are 7.6 GB available RAM dedicated for Ubuntu partition.

I accept full responsibility for this machine, and I have made contingency plans to protect myself against hardware and/or software failure. For example, I am planning to use Git for version control and upload my daily progress to a remote repository on GitHub. Moreover, each week I will be backing up my files to Google Drive and a separate hard disk which would be used only for this purpose. I will buy a second computer if my current machine stops working.

In case my special hardware stops working properly, I could always transition to work in the Computer Laboratory. There are similar older generation NetFPGA boards ('NetFPGA 1G') whose network throughput is smaller (1Gbps instead of 10Gbps).

External Information Sources

[1] Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan Iyengar, Jeff Bailey, Jeremy Dorfman, Jim Roskind, Joanna Kulik, Patrik Westin, Raman Tenneti, Robbie Shade, Ryan Hamilton, Victor Vasiliev, Wan-Teh Chang, and Zhongyi Shi. 2017. The QUIC Transport Protocol: Design and Internet-Scale Deployment. In Special Interest Group on Data Communication (SIGCOMM). ACM.

[2] Xiangrui Yang, Lars Eggert, Steve Uhlig, Zhigang Sun, Gianni Antichi. 2020. Making QUIC Quicker With NIC Offload
Figure 3 from https://dl.acm.org/doi/pdf/10.1145/3405796.3405827

[3] https://tools.ietf.org/id/draft-ietf-quic-manageability-04.html

[4] Eliot Lim Part II dissertation "Performant TCP bytestreams on FPGAs" https://www.cl.cam.ac.uk/teaching/projects/archive/2020/el462-dissertation.pdf, page 3.

[5] Noa Zilberman, Yury Audzevich, G. Adam Covington and Andrew W. Moore,
"NetFPGA SUME: Toward 100 Gbps as Research Commodity," IEEE Micro, vol.34, no.5, pp.32-41, Sept.-Oct. 2014, doi: 10.1109/MM.2014.61

[6] https://github.com/NetFPGA/NetFPGA-SUME-public/wiki/NetFPGA-SUME-Reference-NIC

[7] Offloading QUIC - An Implementation Guide
Manasi Deval, Gregory Bowers,

IETF 104, Prague, March 2019
https://www.ietf.org/proceedings/104/slides/slides-104-quic-offloading-quic-00

[8] https://www.researchgate.net/publication/306925556_Very_high_speed_link_emulation_with_TLEM