1 Overview

In this assignment, we will learn to develop data plane programs using the P4 [9] language. The P4 language is a domain-specific language (DSL) designed for packet processing. Programming Assignment 2 contains *two* parts – parts A and B, with deadlines on Week 10, and Reading Week, respectively. Your task is to implement *three* P4 programs, following the outlined requirements.

> **Important!** Programming Assignment 2 constitutes **18%** of your final grades. You are allowed to work either individually or in a group of two. If you intend to work in a group, your group should be registered using the form at `https://forms.gle/YT7cxmVWELgfBKKo9` before the end of Week 7, i.e., 7 Oct 2023 at 2359 Hrs.

To help you gauge the P4 language, we will need to first pick up the basics of P4 programming by going through some examples of the P4 tutorials [4] (more on this in Section 3). These will also be discussed in the tutorial sessions starting from Week 7. Notwithstanding, you will be tested on P4 fundamentals in Quiz 3 on Week 8.

2 Environment Setup

2.1 P4 Development VM

To work on the assignment, we will be using pre-built virtual machines (VM) with the necessary dependencies installed. The provided VMs are built using adapted scripts from [11].

We only support x86_64 (a.k.a. AMD64) and AArch64 (a.k.a. ARM64) processors. The corresponding software dependencies and steps are described in the following subsections. We recommend having at least 30 GB of free disk space for the purpose of this assignment.

2.1.1 x86_64 (Intel/ AMD processors)

For x86_64-based machines, we support VirtualBox [3] as the hypervisor regardless of the operating system, i.e., Linux, Windows, or macOS.

> **Note:** While VMware Workstation Pro/Player or Parallels may be used as alternative hypervisors for running the P4 development VM, the teaching team might not offer support for issues that arise due to their usage.

VirtualBox Installation: As there is abundant support from the Internet community on VirtualBox [3], you can Google for its documentation for the installation steps, or even ask ChatGPT [10]!

Downloading the VM: Download the VM (∼4 GB) from: `https://www.comp.nus.edu.sg/~khooixz/cs5229/p4-dev-x86_64-build-220823.zip`. Then, compute the SHA256 checksum of the downloaded ZIP to verify the file integrity:

`cd52775f23ab77a15f8c5b547330abb42fbe653e4483f89a8540b86418cef13f`. Finally, extract the ZIP file to get the `.ova` file.

Importing the VM: Bring up VirtualBox, go to "File", click on "Import appliance", and select the extracted VM. The import process should take several minutes. Finally, boot up the VM and you should be able to log in to the VM with the username/password: `p4`/`p4`.

2.1.2 AArch64 (Apple Silicon SoCs – M1/ M2)

For AArch64-based machines, currently, we *only* support UTM [7] using the QEMU [6] backend as the hypervisor on macOS.

> **Note:** The setup is only tested with macOS 13+. VMWare Fusion is not supported. Other hypervisors (e.g., Parallels) or earlier macOS versions or Windows on ARM or Linux ARM may work, but the teaching team might not offer support for issues that arise due to their usage.

UTM and QEMU Installation: We will be using UTM with QEMU as the backend. First, download UTM from [7]. Then, install QEMU [6] by following the instructions at [1]. You will need Homebrew [2] on your Mac to install QEMU.

Downloading the VM: Download the VM (∼4 GB) from: `https://www.comp.nus.edu.sg/~satis/cs5229/p4-dev-aarch64-build-220823.zip`. Then, compute the SHA256 checksum of the downloaded ZIP to verify the file integrity: `e302f8d533b6645cc3b0c7d09e10315868d6ecf3025f94431fbf88313591df30`. Finally, extract the ZIP file to get the `.utm` file.

Importing the VM: Bring up UTM, go to "File", and "Open" the extracted VM image. Finally, boot up the VM and you should be able to log in to the VM with the username/password: `p4`/`p4`.

2.2 IDE/ Code Editor for P4

We have pre-installed Visual Studio Code alongside the P4 Language Server extension from Christophe Beaulieu. This is the recommended toolset for this assignment.

2.3 Disabling IPv6s on the VM

To minimize "interference" from unrelated packets during testing, you are asked to disable IPv6 on your VM. First, open the `/etc/sysctl.conf` file with `sudo` privileges using your favorite text editor, e.g., `nano` or `vim`. Add the following lines at the end of the file.

```
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
```

Then, execute the command `sudo sysctl -p`. You can then verify that IPv6 is disabled by running the following command: `ip a | grep inet6`. Finally, reboot the VM. In the event that you still see IPv6 addresses associated with the primary network interface – `enp0s1`, you can safely disregard it.

3 Prelude: P4 Tutorial

As an introduction to the P4 programming language, we will be using the publicly available P4 Tutorial [4] developed by the P4 Language Consortium. At the home directory of the `p4` user, you should be able to find the `tutorials/` folder.

You are asked to go through *four* exercises from the P4 tutorials, namely `basic`, `calc`, `mri`, and `firewall`, following the order in which they have been mentioned. These four exercises should equip you with sufficient knowledge to work on the assignment. While the solutions for these exercises are provided, please attempt the exercises first before referring to the solutions. For the remaining exercises, we highly recommend you take a look at them.

Note: We will also discuss P4-related concepts in the tutorial sessions starting from Week 7. That said, you are encouraged to take a look at the P4 Tutorials beforehand to get a better understanding prior to the tutorial sessions.

3.1 Quiz 3 on P4 Fundamentals

As a checkpoint, Quiz 3 will be covering P4 fundamentals, as well as the topics related to P4 discussed in the tutorial sessions on Weeks 7 and 8.

4 Assignment Description

Programming Assignment 2 is split into two parts – parts A (8%) and B (10%). You will need to clone the repository from `https://github.com/NUS-CIR/cs5229-2023-p4` (available after 1 Oct 2023) to get the assignment package. More details on the assignment can also be found in the `README` of the respective folders.

4.1 Part A: Baby Steps in P4 (8%)

After picking up the basics in P4 in Section 3, we will now get our hands dirty with two simple exercises to put your P4 knowledge into practice. The two exercises can be found under the `part_a/` folder.

Part A Submission Deadline: 23 October 2023 (Monday) at 2359 Hrs.

4.1.1 A-1: Fake Ping (4%)

We will implement a P4 program that responds to ICMP echo requests from a host to any *permitted* destination. The topology given contains only one host and one switch. The P4 program shall ignore any other packets except ICMP. You are expected to define (some of) the headers and implement the parser/deparser and the body of the P4 program. Read the `README` carefully, and refer to the `TODO` sections outlined in the skeleton P4 program included.

Submission: Upload only `fake_ping.p4` to Canvas, under the folder "Programming Assignment 2 - Part A - Fake Ping".

4.1.2 A-2: Secret Message Exchange (4%)

The "hidden drop" or dead drop [8], refers to a location where secret messages, information, or items are left for someone to retrieve. In this exercise, we will be using the switch to play the role of a dead drop facility. The topology given comprises two hosts and one switch. One of the hosts will be responsible for depositing "secret messages" on the switch, while the other will be retrieving them. The dead drop facility can only be accessed through a "secret" UDP port. You are expected to define (some of) the headers and implement the parser/deparser and the body of the P4 program. We will be using the `register` extern as stateful memory to maintain the stored "secret messages". Read the `README` carefully, and refer to the `TODO` sections outlined in the skeleton P4 program included.

Submission: Upload only `secret.p4` to Canvas, under the folder "Programming Assignment 2 - Part A - Secret".

4.2 Part B: Network Monitoring with Sketches (10%)

We will now take a look at a practical application of P4 – network monitoring. The exercise can be found under the `part_b/` folder.

> **Part B Submission Deadline:** 20 November 2023 (Monday) at 2359 Hrs.

In Lecture 4, we discussed the concepts, and techniques for network monitoring. One of which, is the use of sketches, a probabilistic data structure with sub-linear memory requirements, for packet counting to perform network monitoring. By now, you should be familiar with sketches after Programming Assignment 1.

You are asked to make use of a sketch data structure that identifies heavy hitters (given a specific threshold) in a given stream of packets. You will be using the `register` extern as the stateful memory to implement the data structure, and then you should mirror a copy of the detected heavy flow to an external collector to report them. The collector should not receive any duplicated reports. At the same time, you will also identify and drop potentially malicious sources (e.g., DNS amplification attacks). Make sure to read the `README` carefully, and refer to the `TODO` sections outlined in the skeleton P4 program included.

Submission: Upload only `sketch.p4` to Canvas, under the folder "Programming Assignment 2 - Part B - Sketch".

5 Grading Policy

Unfortunately, code that does not compile is *graded harshly*, i.e., 0%. If you want partial credit on code that does not compile, comment it out and make sure your file compiles! Your code will be assessed through automated test scripts with our pre-defined test cases using the packet test framework (PTF) [5].

You are also reminded to strictly follow the stated submission instructions, e.g., file naming, and files that are to be submitted. There will be a 1% penalty for each submission that does not follow the stated submission instructions. For group submissions, only the *registered* leader shall submit to Canvas.

Reminder: While you can discuss with your classmates and refer to the internet, we will not condone plagiarism. You are expected to adhere to the NUS student code of conduct. Stern action will be taken for any form of plagiarism detected.

5.1 Late Penalty

Late submissions will be penalized 20% per day, i.e., submissions 5 days after the deadline will not be graded, and marked as 0%.

6 Getting Help

If you face any issues or have any questions related to the assignment, please raise them on `Piazza` and make sure to *tag* your post under the topic `programming2`. In case you are unable to access Piazza, please reach out to TA Xin Zhe via email.

Warning! ChatGPT is known to produce obsolete and problematic answers for P4. Thus, we advise you to take ChatGPT responses related to P4 with a grain of salt.

References

[1] Download QEMU. `https://www.qemu.org/download/#macos`.

[2] Homebrew — The Missing Package Manager for macOS (or Linux). `https://brew.sh/`.

[3] Oracle VM VirtualBox. `https://www.virtualbox.org/`.

[4] P4 Tutorial. `https://github.com/p4lang/tutorials` [commit: 01ed2c4].

[5] PTF Packet Testing Framework. `https://github.com/p4lang/ptf`.

[6] QEMU: A generic and open source machine emulator and virtualizer. https://www.qemu.org/.

[7] UTM — Virtual machines for Mac. https://mac.getutm.app/.

[8] Hacker lexicon: What is a dead drop? https://www.wired.com/story/what-is-dead-drop/, 2019.

[9] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, et al. P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review*, 44(3):87–95, 2014.

[10] ChatGPT. Install VirtualBox: Steps. https://chat.openai.com/share/e93c02c2-db16-4096-b251-af273efc5797, 2023.

[11] Andy Fingerhut. P4 Guide. https://github.com/jafingerhut/p4-guide [commit: 5042ab7].