Application Aware Switching with Zodiac FX

Markus Moskopp
Chair of Communication Networks
Department of Electrical and Computer Engineering
Technical University of Munich
Munich, Germany
m.moskopp@tum.de

Mayukh Das
Chair of Communication Networks
Department of Electrical and Computer Engineering
Technical University of Munich
Munich, Germany
mayukh.das@tum.de

***Abstract*—The work deals with finding out the performance improvement of the low cost Zodiac FX switch with optimized firmware that is sufficient for a specific switching task. As it is assumed that the processing time of the switch is directly depended to the code running in it, the optimization process should improve the throughput. The firmware version v0.85 of the open source ZodiacFX firmware was optimized for this task and the performance was tested.**

***Index Terms*—ZodiacFX, Software-Defined Networking, OpenFlow**

I. Introduction

Software-Defined Networking (SDN) is an architecture that aims to make networks agile and flexible. SDN architectures decouple network control and forwarding functions, enabling network control to become directly programmable and the underlying infrastructure to be abstracted from applications and network services. The centralized SDN controller directs the switches to deliver network services wherever they're needed, regardless of the specific connections between a server and devices. A typical representation of SDN architecture comprises three layers: the application layer, the control layer and the infrastructure layer. The control layer represents the centralized SDN controller software that acts as the brain of the software-defined network. This controller resides on a server and manages policies and the flow of traffic throughout the network [1].

Some of the frequently used controllers are Pox [2], Ryu SDN Framework [3], and Open Daylight [4]. The application layer, not surprisingly, contains the typical network applications. The infrastructure layer is made up of the physical switches in the network - Northbound Networks' ZodiacFX switch [5] in our case. These three layers communicate using respective northbound and southbound application programming interfaces (API). For example, applications talk to the controller through its northbound interface, while the controller and switches communicate using southbound interfaces, such as OpenFlow - although other protocols exist [1]. OpenFlow is used to communicate to the switches on the network which then process the data going through their ports using what are called "flows" [6]. These flows can be generated by the programmed application in the controller.

The low cost SDN switch Zodiac FX [5] and the open-source SDN protocol OpenFlow enable researchers and developers of Software-Defined Networking in testing their applications. Although simulation environments, such as mininet [7] exist, they also have some limitations that can be overcome by real hardware.

II. Problem at hand

The Zodiac FX hardware switch by Northbound Networks is considered as the smallest software defined network test bed available. It includes many features, some of which are [8]

- 4 x 10/100 Fast Ethernet ports with integrated magnetics,
- Command line interface accessible via USB virtual serial port,
- Support for OpenFlow 1.0, 1.3 & 1.4, 512 entry software flow table,
- 802.1q VLAN support for 64 groups from 4096 IDs

However, Zodiac FX processes all data and control packets in the software running on the main processor (Atmel SAM4E8CA [5]). Hence, the processing time of the switch directly depends on the code running on it. The firmware is complex as it supports extensive features which are not always used for specific switching applications. For example, if we have a controller that uses simple in one port and out the other using OpenFlow 1.0, many aspects of the firmware such as support for OF 1.3 and 1.4 become useless for this particular use case. The goal of the project was to optimize the firmware of the switch for a particular use case and compare the change in throughput of the data plane traffic. This test becomes fairly possible as the Zodiac FX firmware is open source and fully available at GitHub so that anyone can create contribute or create an own custom version [9].

III. Design & Setup

This section describes the design decisions we made and finally explains the setup we chose for our validation measurements. The relevant parts, as the SDN controller and the modified firmwares are presented below.

A. Controller

For our setup we use a Ryu controller application. Ryu supports various protocols for managing network devices, such as OpenFlow (1.0, 1.2, 1.3, 1.4, 1.5) and provides a simple API to register custom handler functions for OpenFlow events. Ryu also provides built-in example code for a simple L2-Learning

Switch that we used as a base for our own implementation [3]. In contrast to an L2 Hub that floods all ports with any frames received, an L2 Switch must efficiently route unicast Ethernet frames to the specific physical port that is attached to the device with the destination data-layer (MAC) address. When a switch is first started, it does not know the location of every device it is connected to, so some flooding must occur until frames are received from every device that will be addressed. We choose to work with OpenFlow version 1.0 for compliance to the custom firmware described in section III-B.

The controller has the following additional functionality compared to Ryu's example code. For Packet In events that include a packet that originates from the IP address *10.0.3.7* and targets the IP address *10.0.3.9*, a new flow is installed on the switch that matches with the source and destination MAC address and will also match every packet that will be sent out during a measurement. Then, 100 "dummy" flows are installed, yet this happens only once. These dummy flows are installed with a lower priority to not influence the resulting logical behaviour of the switch. The main purpose of adding these flows is to check whether adding them changes the switch's throughput in specific use cases. Also, the way the Zodiac FX switch processes flows and packets was considered (see section III-B3).

Since the controller always has to match our custom firmwares as descibed in section III-B, we decided to use a this simple switch OpenFlow controller application. For more advanced use cases and scenarios, this controller will not work.

B. Firmware

The following sections describe the different modifications to the firmware (version 0.85) of the Zodiac FX switch.

1) Custom Firmware A: First, the firmware was modified to drop support for OpenFlow version 1.3 and VLAN actions/matching and only work with OpenFlow version 1.0 from now on. This removes some load from the switch, as many OF 1.3 only features require several counters that need to be updated on every package that the switch processes. Furthermore, the whole http server that provided an easy to use frontend for inexperienced users, was removed. These first modifications of the code reduce the size of the resulting firmware from 199kB (v0.85) to 116kB which is roughly 60% of the original binary and only fills around 23% of the processor's flash memory [10].

2) Custom Firmware B: The next modification removes some more features from the OpenFlow protocol to further simplify and adapt the firmware to the measurement setup (see section C below). In this next iteration of code adjustments, all code sections that build port stats, table stats and flow stats were removed. This reduces the overhead of maintaining all the statistics structures and counters. Additionally, flow timeouts (soft and hard timeouts) were completely disabled. Those timeouts are not needed in this small switch for the setup described in section III-C.

3) Custom Firmware C: The last alteration of the code is small, yet has a big impact on the way the switch matches the packets to a flow in the flow table. This changes from matching on the flow with the highest priority to the first flow that matches the packet attributes. The function that matches the packet to a flow does then not have to go through all flows, if the matching flow is not the last flow in the flow table. In an example setup, the flow that matches most/all packets in the switch is the first one in the flow table which contains a lot other flows. This setup best shows the impact of this modification, since the original firmware will unnecessarily check all flows in the table, while the modified one only must check the first flow in the table.

C. Setup

The following describes how the setup for measuring the throughput of the Zodiac FX switch is built. Starting from preparing the firmware and finishing with the hardware setup. Also, the most error prone steps in the whole process are mentioned.

1) Preparing the Firmware: Since the Zodiac FX switch validates a checksum of uploaded firmware before actually running it, the built firmware binary must be prepared with such a checksum after building it from the source code. For simplicity, the built binary can be flashed on the switch as described in section III-C2. After the file is uploaded, the switch outputs an error message on the serial connection that the firmware verification failed. Subsequently, the hidden CLI command *get crc* can be invoked which outputs the required checksum that must be appended to the firmware file with a hex editor [9].

2) Flashing the Firmware: The firmware of the Zodiac FX switch can be updated in multiple ways, yet only one way is working with all (custom) firmware versions. Therefore, a Windows PC with the tool "extraputty" is needed as it supports the XMODEM protocol over the serial connection[1] [11]. Additionally, the firmware can be resetted as described in the Zodiac FX documentation. Also, when connecting to the virtual serial port of the switch via "extraputty", the "update" command is available from the root menu of the switch's CLI to initiate a firmware upload via XMODEM [9]. For actually starting the upload of the binary firmware file, the XMODEM option in extraputty and then the file to be uploaded must be selected. However, for the original firmware, if the upload is not started within six seconds, the firmware update will abort. In custom firmware versions, this time can be increased by modifying the respective function in code.

3) Test Bed: For setting up the test bed, two computers, a Zodiac FX switch, three RJ-45 patch cables and one USB type A to USB micro type B cable are needed. One computer, that will run the Ryu application, needs two network interfaces. The first is configured to the static IP address *10.0.1.99* and connected to the switch's controller port, the second is configured to the static IP address *10.0.3.7* and connected to one of the free switch ports. The second computer also needs a network interface which is configured to the static IP address

[1]We tested different UNIX tools for flashing the firmware via XMODEM, however, the update always resulted in corrupted firmware.

10.0.3.9 and connected to another free switch port. A sketch of this setup can be seen in Fig. 1.

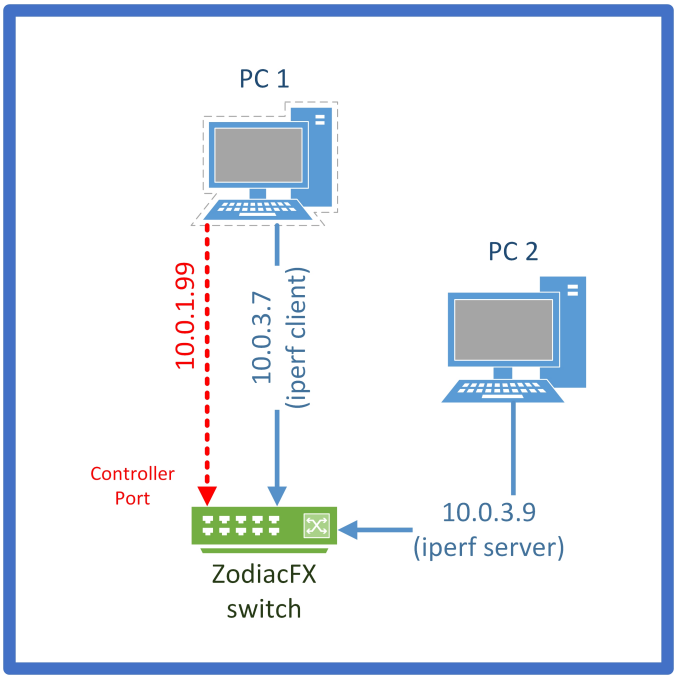

Fig. 1. Zodiac FX measurement setup

Before any measurement can take place, the switch must be configured and initialized correctly. The configuration can be done via the switch's command line interface. Particularly the configuration option "failstate" must be set to "safe". Otherwise, the switch will clear all flows when losing connection to the controller. For the initialization, all previous flows have to be cleared via the CLI and a measurement must be done to install the correct flows on the switch. This also can be verified in the CLI of the switch. The CLI must be disconnected before the measurements, as a busy CLI can affect the test results. The switch is then ready for the measurements.

The throughput is measured with iperf3 which "is a tool for active measurements of the maximum achievable bandwidth on IP networks" [12]. In this setup, we measure the bandwidth through the switch for 10 seconds. One computer serves as an iperf3 server on the interface with IP address *10.0.3.9* and specifies the port *5201*. The other computer connects to the server through the interface with IP address *10.0.3.7* and starts a TCP bandwith measurement[2].

IV. Results

All custom firmware versions described in section III-B as well as the original Zodiac FX firmware in version 0.85 were tested on the test bed described in section III-C3. The size of the respective firmwares is shown in table I.

Since the HTTP server of the original firmware makes up a big part, the modified firmwares that removed that feature are much smaller. Also, from version A to version B of the modified firmwares, the removal of all statistic related structures, reduce the binary size even more.

TABLE I
Firmware Results

Firmware Version	Size (kiloBytes)
0.85	199
Custom A	116
Custom B	108
Custom C	107

Finally, the throughput was measured six times for each firmware and the mean value over those measurements was taken. The results are displayed in table II.

TABLE II
Throughput Results

Firmware Version	Measurements (Mbit/s)						Mean (Mbit/s)
	1	*2*	*3*	*4*	*5*	*6*	
0.85	25.6	24.2	23.4	22.6	18.5	24.2	23.1
Custom A	25.0	22.9	22.9	23.4	23.1	22.3	23.3
Custom B	24.5	25.0	25.6	25.1	25.3	23.1	24.8
Custom C	92.4	92.1	92.2	92.5	92.0	92.2	92.2

The first two iterations of firmware changes did not bring any significant change in throughput. Therefore, the parts of the Zodiac FX firmware we targeted in those iterations do not influence the switch's throughput significantly in this setup. In the third modified firmware, we directly targeted some flaws in the original firmware. From the original to this modified firmware, the throughput increases by almost 300% which showcases the impact of adapting the switch to the specific application.

V. Conclusion

The sole conclusion of the project was that with firmware optimization we can indeed achieve better performance, yet it is highly application dependent. We also support this with the results we got for different firmware modifications. This idea could also be extended to other applications. However, modifying the firmware of the SDN switch is not practical for testing different applications. In those cases, some light modifications like shown in section III-B1 and III-B2 can be applied. Finally, working with the Zodiac FX switch is fairly easy as it provides a simple command line interface, web interface and freely available source code. The firmware is however complex, not well constructed and contains not much high level documentation. Although there is some guidance available online, getting started with the whole firmware modifications and updating is not an easy task, since the process has some quirks that need to be learned.

[2]The exact commands are, for the server: *iperf3 -s -B 10.0.3.9 -p 5201*, and for the client: *iperf -c -B 10.0.3.7 -p 5201*

REFERENCES

[1] Bruno Nunes Astuto, Marc Mendona, Xuan Nam Nguyen, Katia Obraczka, Thierry Turletti. A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks. Communications Surveys and Tutorials, IEEE Communications Society, Institute of Electrical and Electronics Engineers, 2014, 16 (3), pp.1617 - 1634.

[2] "noxrepo/pox", GitHub, 2019. [Online]. Available: https://github.com/noxrepo/pox. [Accessed: 25- Jul- 2019].

[3] "Ryu SDN Framework", Osrg.github.io, 2019. [Online]. Available: https://osrg.github.io/ryu/. [Accessed: 25- Jul- 2019].

[4] "OpenDaylight", OpenDaylight, 2019. [Online]. Available: https://www.opendaylight.org/. [Accessed: 25- Jul- 2019].

[5] N. Networks, "Zodiac FX", Northbound Networks, 2019. [Online]. Available: https://northboundnetworks.com/products/zodiac-fx. [Accessed: 25- Jul- 2019].

[6] McKeown, Nick & Anderson, Tom & Balakrishnan, Hari & Parulkar, Guru & L. Peterson, Larry & Rexford, Jennifer & Shenker, Scott & Turner, Jonathan. (2008). OpenFlow: Enabling innovation in campus networks. Computer Communication Review. 38. 69-74. 10.1145/1355734.1355746.

[7] Lantz, Bob & Heller, Brandon & McKeown, Nick. (2010). A Network in a Laptop: Rapid Prototyping for Software-Defined Networks. Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks. 19. 10.1145/1868447.1868466.

[8] Wang, Song & Gomez Chavez, Karina & Sithamparanathan, Kandeepan & Zanna, Paul. (2018). The smallest software defined network testbed in the world: Performance and security. 1-2. 10.1109/NOMS.2018.8406116.

[9] "NorthboundNetworks/ZodiacFX", GitHub, 2019. [Online]. Available: https://github.com/NorthboundNetworks/ZodiacFX. [Accessed: 25- Jul- 2019].

[10] "ATSAM4E8C - 32-bit SAM Microcontrollers", Microchip.com, 2019. [Online]. Available: https://www.microchip.com/wwwproducts/en/ATSAM4E8C. [Accessed: 25- Jul- 2019].

[11] "ExtraPuTTY — Fork of PuTTY", Extraputty.com, 2019. [Online]. Available: http://www.extraputty.com/. [Accessed: 25- Jul- 2019].

[12] V. GUEANT, "iPerf - The TCP, UDP and SCTP network bandwidth measurement tool", Iperf.fr, 2019. [Online]. Available: https://iperf.fr/. [Accessed: 25- Jul- 2019].