Thesis Title

APPROVED BY

SUPERVISING COMMITTEE:

Jacob Abraham, Supervisor

Erwin Schrödinger

Albert Einstein

Charles Townes

Arthur Schawlow

Thesis Title

by

Erick Carvajal Barboza, B.S.

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2016

Dedicated to my parents.

Acknowledgments

I wish to thank to CRUSA Foundation, The University of Costa Rica and the Costa Rican Science and Technology Ministry for sponsor my Master Studies at the University of Texas at Austin.

Thesis Title

Erick Carvajal Barboza, M.Sc.
The University of Texas at Austin, 2016

Supervisor: Jacob Abraham

Here comes the abstract, not more than 350 words.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Software Defined Networks

The growth that the Internet has reached is causing several changes on the networking industry. The Internet components are reaching a limit, and thus slowing down the innovation and progress on the area. Nowadays, there is no practical method for the researchers to experiment with new network protocols in a realistic setting that will grant confidence for their widespread deployment, as a result, a lot of new ideas pass untried and untested [7].

To overcome this barrier, a joint effort of Stanford University, the University of California at Berkeley, and a number of other universities founded the Global Environment for Network Innovation (GENI) program in 2000 [6].

A very important outcome of the GENI initiative is the Software Defined Networking (SDN) which main goal is to evolve the vertical network model that is currently used into a more horizontal model. In the SDN model, the control plane is separated from the data plane, so that thee control plane is able to run in software on standard servers rather than in the router itself, making the switches simple forwarding devices and allowing the control logic to be logically centralized in the controller [4]. Figure 1.1 shows a simplified view of the SDN architecture, it is important to mention that even when the controller is logically centralized, that does not means it is physically centralized, actually, in order to have the expected levels of performance, reliability and scalability is necessary to have a physically distributed control panel.

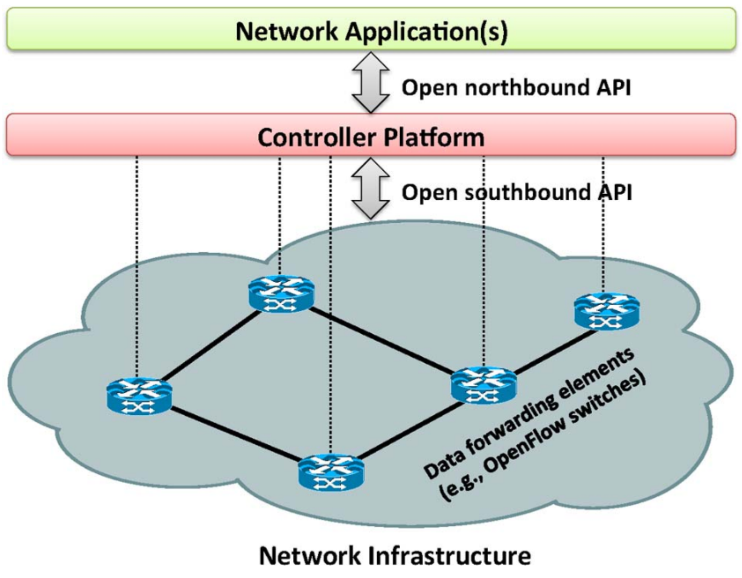

Figure 1.1: Simplified View of the SDN architecture. Taken from [4]

This new model of network will allow a number of new capabilities like a rapid introduction new network functions, given that the changes will be applied at software level, rather than hardware or firmware [6].

Since the goal is to apply new features to the switches it is necessary that these switches allow to be programmed. One way to do it is persuade the equipment vendors to provide and open platform on their switches that allow the researchers can experiment their new protocols, this alternative is very unlikely because vendors will not want to open up the interfaces that they have spend years developing.

Another alternative is to use existing open software platforms, but these alternatives do not have the required performance nor port density. A more promising approach is OpenFlow, which exploits the common set of functions that run in the flow-tables of many switches from different vendors, which keeps the vendors necessity for closed platforms [7].

OpenFlow provides an open protocol that allows to program the flow-tables on routers of different vendors. With this protocol, the network administrator can partition the network traffic and allow the researchers to control their own flows and their processing. This allows the research to try new routing protocols, security models and addressing schemes, all this on the same network that has an isolated traffic that is being processed with the regular schemes to avoid affectation of the network [7].

An OpenFlow switch consists of one or more flow-tables, a group-table and a secure channel to an external controller. Each of the flow-tables contains a set of flow entries, each entry consists of matching fields, counters and instructions to apply to the packet. The matching occurs in a pipeline manner through all the tables, the entries on the tables are ordered by priority, so that the first matching entry will be used. Once the match is found the actions associated are executed. If no match is found then the packet may be forwarded to the controller or dropped, depending on the switch configuration.

The entries on the flow-tables can also point to a group, which specifies an additional processing, this action groups are stored on the group-table, each of this groups contains a list of action buckets with specific semantics that depend on the group type.

The OpenFlow channel is the interfaces that connects the switches to a controller, through this interface, the controller configures and manages the switch [10].

1.2 Bounded Model Checking

Model Checking refers to a set of algorithms that verify the properties of state transitions systems using a search of their associated state transition graph. The properties to be verified are expressed using temporal logic, this kind of expressions allow, for example, to assert a property which is not true in the present but may eventually become true in the future [2].

The model checking implementations started at the 1980s [3], but these used explicit representations of state transition graphs and, with the state explosion problem where the number of states is growing exponentially, the use of these techniques is no longer possible. This techniques were used for designs with less than a million states, making them unsuitable for industrial usage.

In the 1990s techniques that used symbolic state space exploration appeared. These techniques explore the state space through the use of Binary

Decision Diagrams (BDD) [1]. The BDDs hold the characteristic functions of sets of states, allowing the computation of transitions among sets of states rather than individual sets. However, even though these techniques increased the order of magnitude in the size of the designs that could be verified using model checking, it was still insufficient for the industrial designs [2].

A technique called bounded model checker emerged on the 2000s. This method is applicable to safety (Bad things will not happen) and liveness (Good things will eventually happen) properties. The verification of a safety property involves checking if a set of states is reachable, and the verification of liveness properties involves detecting loops in a system. This method requires little by-hand manipulation from the users, while the BDD based verification often requires manual variable ordering, or other handmade abstractions. The robustness and capacity of this method makes it a very suitable one for industrial applications.

The main disadvantage of this method is that it lacks completeness because the model is checked until a certain number of transitions have occurred, indicating errors where there are none or hiding errors that require a lot of running time. That is the reason why the main use of this tool is to find bugs, rather than proving correctness [2].

Appendices

Appendix A

Lerma's Appendix

The source LaTeX file for this document is no longer quoted in its entirety in the output document. A LaTeX file can include its own source by using the command `\verbatiminput{\jobname}`.

Appendix B

My Appendix #2

B.1 The First Section

This is the first section. This is the second appendix.

B.2 The Second Section

This is the second section of the second appendix.

B.2.1 The First Subsection of the Second Section

This is the first subsection of the second section of the second appendix.

B.2.2 The Second Subsection of the Second Section

This is the second subsection of the second section of the second appendix.

B.2.2.1 The First Subsubsection of the Second Subsection of the Second Section

This is the first subsubsection of the second subsection of the second section of the second appendix.

B.2.2.2 The Second Subsubsection of the Second Subsection of the Second Section

This is the second subsubsection of the second subsection of the second section of the second appendix.

Appendix C

My Appendix #3

C.1 The First Section

This is the first section. This is the third appendix.

C.2 The Second Section

This is the second section of the third appendix.

Bibliography

[1] Jerry R Burch, Edmund M Clarke, Kenneth L McMillan, David L Dill, and Lain-Jinn Hwang. Symbolic model checking: 1020 states and beyond. *Information and computation*, 98(2):142–170, 1992.

[2] Edmund Clarke, Armin Biere, Richard Raimi, and Yunshan Zhu. Bounded model checking using satisfiability solving. *Formal methods in system design*, 19(1):7–34, 2001.

[3] Edmund M. Clarke, E. Allen Emerson, and A. Prasad Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 8(2):244–263, 1986.

[4] Diego Kreutz, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015.

[5] Daniel Kroening and Michael Tautschnig. Cbmc–c bounded model checker. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 389–391. Springer, 2014.

[6] Chung-Sheng Li and Wanjiun Liao. Software defined networks. *IEEE Communications Magazine*, 51(2):113–114, 2013.

[7] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[8] Hyungbae Park, Sejun Song, Baek-Young Choi, and Henry Zhu. Software-defined networking (sdn) control message classification, verification, and optimization system. In *Software Reliability Engineering Workshops (ISSREW), 2015 IEEE International Symposium on*, pages 25–28. IEEE, 2015.

[9] Vladislav Podymov and Uliana Popesko. Uppaal-based software-defined network verification. In *Tools & Methods of Program Analysis (TMPA), 2013*, pages 9–14. IEEE, 2013.

[10] OpenFlow Switch Specification. Openflow switch specification version 1.1.0, 2011.

Vita

Erick Mauricio Carvajal Barboza was born in San José, Costa Rica on May 11th 1992, son of Danilo Carvajal Campos and Patricia Barboza Cascante and brother of Andrés Carvajal Barboza. He graduated with honors from The University of Costa Rica as an Electrical Engineer in 2014.

During his time as a student, he worked at Intel Corporation in Costa Rica for a year and a half performing structural design for graphic processors. After he graduated, he worked at The University of Costa Rica as an adjunct professor for two semesters and one summer session. There he lectured Signals and Systems and Computers Architecture.

He started his Masters Degree at the University of Texas at Austin in August 2015.

Permanent address: 5200 N Lamar Blvd
Austin, Texas 78751

This thesis was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.