Software Defined Networking: Distributed systems and Trust computation

Guided by: Ms. Vaishnavi Moorthy
(Assistant Professor (O.G.), SRMIST

Naman Arora-RA1511003010235
Nikhil Gupta-RA1511003010245

Abstract

The internet, since the advent of ARPANET, has come along a very long way. It has undoubtedly changed millions of lives and even now is in its infancy. Software Defined Networking (SDN) is presented as a paradigm shift in this regard. It strives to standardize the networking on all levels. This is an initiative to redesign the current networking stack and compartmentalize into three main planes, the data plane, the control plane and the management plane, respectively moving from bottom up. We, here, have put an effort to augment the idea of SDN to a more distributed framework. Using cleverly designed topologies like Spine leaf, we demonstrate the interconnection of controllers using relay system designed from bottom up as the first phase. The second phase, on other hand, acknowledges the need to secure such translations and we try to mitigate Denial of Service (DoS) attacks on the control plane.

Introduction

1) Currently SDN, as defined by OpenFlow, does not stipulate inter-controller communication.
2) SDN, thus, is limited to single controller and non-scalable networks.
3) Huge trend-setters in industry rely on topologies like Mesh Networks.
4) Introduction of relay communication between controllers.
5) Facilitation of broadcast like capabilities is proposed.
6) Security threats to control planes in form of DoS attacks are explored.
7) Mitigation of pre-specified Denial of Service (DoS) attacks on control plane.

Innovative Idea of Project

1) The current industry standard relies on Mesh topology within controllers.
2) Incorporation of Spine Leaf topology in distributed SDN networks here is done through relays.
3) Relays are custom built to facilitate rapid, scalable and flexible inter controller communication.
4) In-built features like duplex connection between controllers and relay, broadcast capabilities, leveraging multi-threaded design to eliminate bottlenecks and relay nesting.
5) This, furthermore, reduces network overhead as a whole in an initiative to conserve energy.

Purpose of Project

1) This is an effort to take on the challenge of connecting the controllers in most effective and efficient way.
2) We strive to make the network as distributed as possible without losing on the issue of scalability, security and efficiency in practical prospects.
3) We present an idea that is open for all community to scrutinize and augment.

Scope of the Project

1) The project aims to deliver an interconnection standard between controllers in an OpenFlow compliant distributed network environment.
2) It strives to deliver this in most efficient way possible.
3) Certain targets include minimized latency, maximized throughput, minimized energy consumption, etc.

Limitations of Current System

1) Less scalability factor.
2) More energy consumption.
3) More than required redundancy
4) Greater expense for physical cables.
5) More vulnerable area to secure.
6) $(n-1)^n$ connections within controllers due to Mesh topology.

Proposed System

1) We propose a relay to act as a bridge between the controllers in a distributed system.
2) The relays can be sub-relayed as per geographical requirements.
3) Controllers use relay as proxy to broadcast flow query in the network.
4) A duplex connection between each controller and relay facilitates simultaneous broadcast and reply.
5) Bottlenecks are eliminated using frequent multi threaded constructs with appropriate synchronization algorithms for maintaining data integrity.
6) A UDP server, as child process, listens for controllers’ first initiative for a short time then converts to TCP server listing for those particular connections.
7) We use python/java as controller language to interface with dynamic shared libraries coded in pure C, while relay engine remaining in pure C.

Software Requirements

1) Mininet:- for topology creation, switch and host emulation.
2) Onos:- OpenFlow compliant controller framework.
3) Ryu:- A second controller framework to demonstrate flexibility.
4) GNU GCC:- Compiler tool chain.
5) LD:- Linker
6) Docker:- Containerization tool.
7) Java:- OpenJDK or Oracle JDK for Onos support.
8) Python:- The python programming language for Ryu.
9) Tmux:- Terminal multiplexer for usage in docker shells.
10) Vim:- (Vi Improved) editor of choice
11) Linux:- Code environment (Specific choice- Arch Linux vanilla).

References

1) Diego Kreutz et al., “Software-Defined Networking: A Comprehensive Survey” in Proceedings of the IEEE, vol 103.
2) Kannan Govindarajan et al., “A literature review on software-defined networking (SDN) research topics, challenges and solutions” in Advanced Computing (ICoAC), 2013 Fifth International Conference.
3) Abubakar Siddique Muqaddas et al., “nter-controller traffic in ONOS clusters for SDN networks” in Communications (ICC), 2016 IEEE International Conference.
4) Radware DefenseFlow Security Operations Model, Whitepaper.