Software Defined Networking: Distributed systems and Trust computation

What are Software Defined Networks?

- In a software-defined network, a network engineer or administrator can shape traffic from a centralized control console.
- The centralized SDN controller directs the switches to deliver network services wherever they're needed.
- A typical representation of SDN architecture comprises three layers: the application layer, the control layer and the infrastructure layer.

SDN Architecture

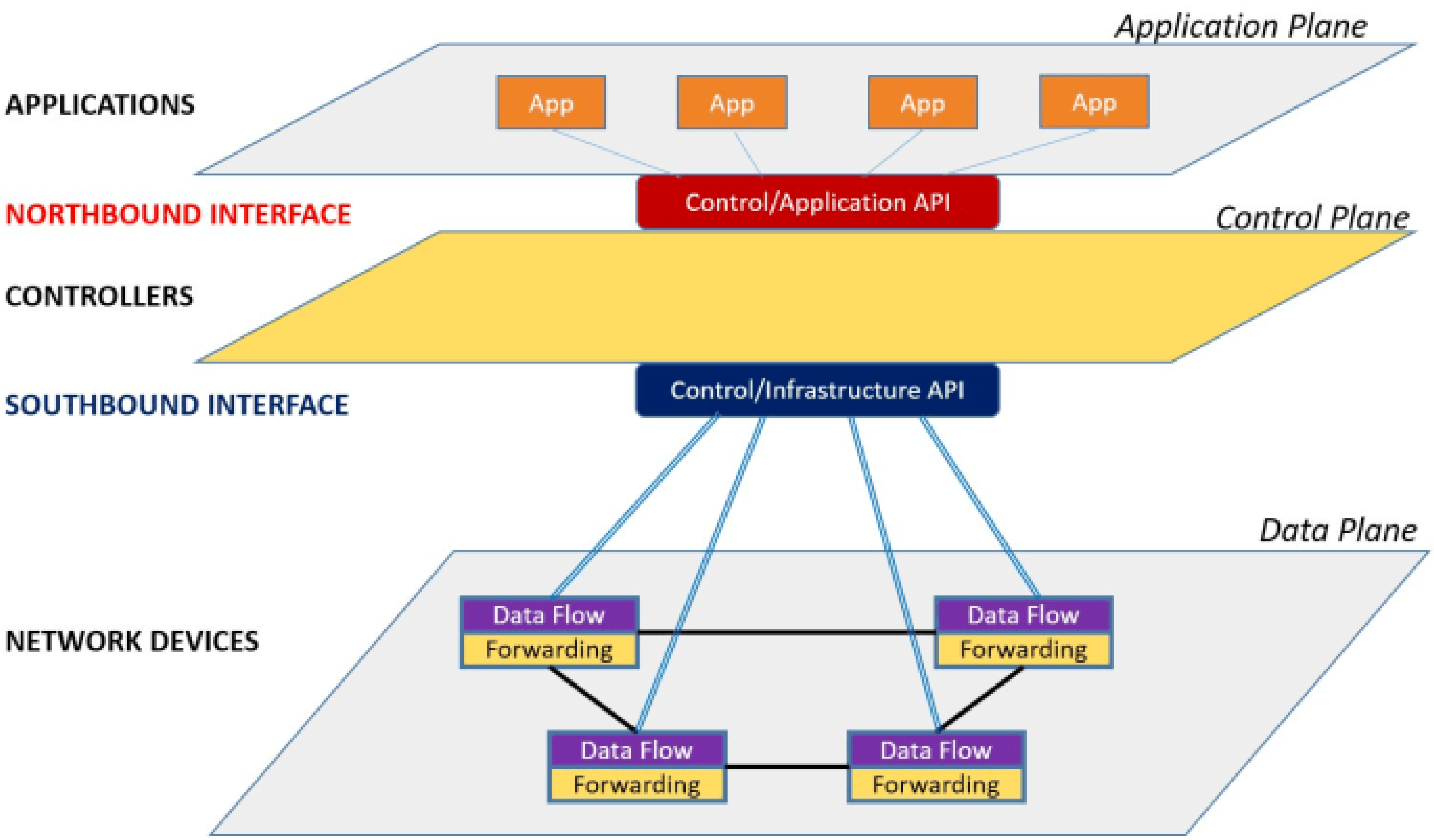


Figure 1 - Software-Defined Networking – A high level architecture

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

- 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.
- Using cleverly designed hybrid topologies, we demonstrate inter-controller connection in a distributed environment in first phase.
- The second phase acknowledges the need to secure such translations and we try to mitigate Denial of Service (DoS) attacks on the control plane.

Current Situation

1) Current SDN industry standard controller implementations do not inherently support distributed inter-controller communication.
2) Some of them who do, use infeasible and expensive mesh topologies.
3) SDN, thus, is limited to single controller and non-scalable networks.
4) Inter-controller communication of different types is also a problem.

Limitations of Current System

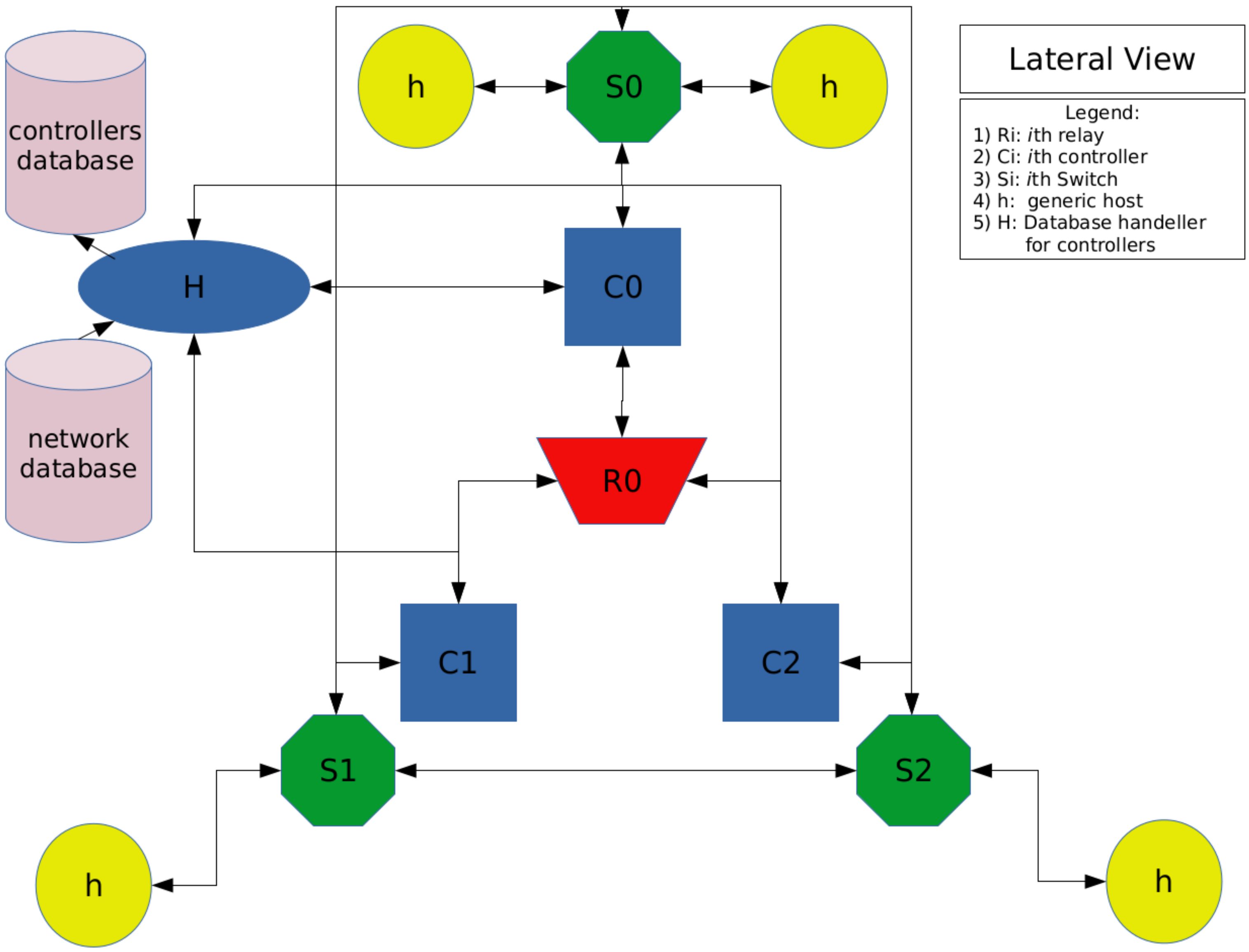
1) Less scalability factor.
2) More energy consumption.
3) Greater expense for physical cables.
4) More vulnerable area to secure.
5) $(n-1)^n$ connections within controllers due to Mesh topology.

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.
6) A TCP server listening for particular connections at controller handler, relay and controller as well for the duplex connection.
7) Python/java has been used as controller implementation language to interface with dynamic shared libraries coded in pure C, while relay engine remaining in pure C.

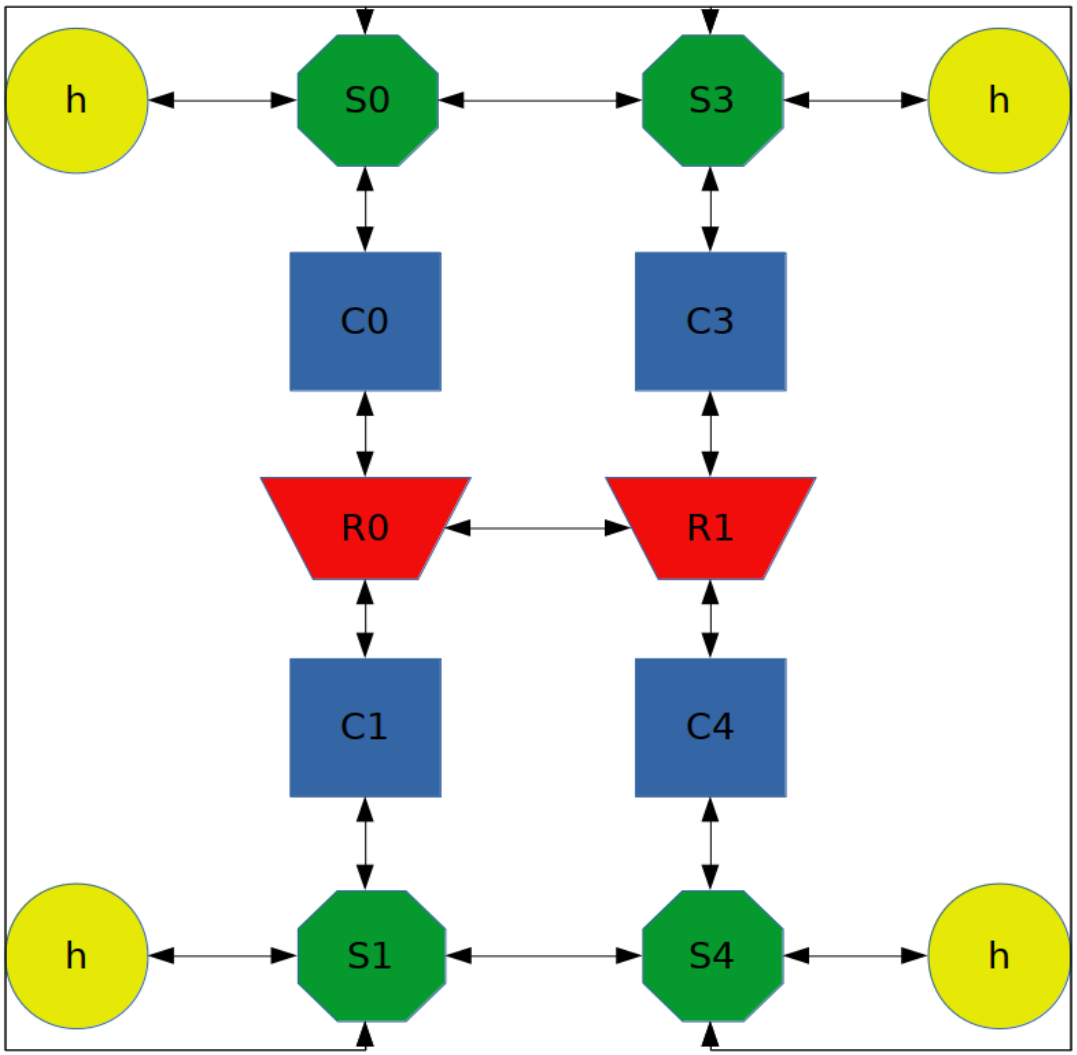
Review 1

Designed Architecture



Lateral View

Legend:
1) Ri: *i*th relay
2) Ci: *i*th controller
3) Si: *i*th Switch
4) h: generic host
5) H: Database handeller for controllers



Cross-Sectional View (Observer’s left)

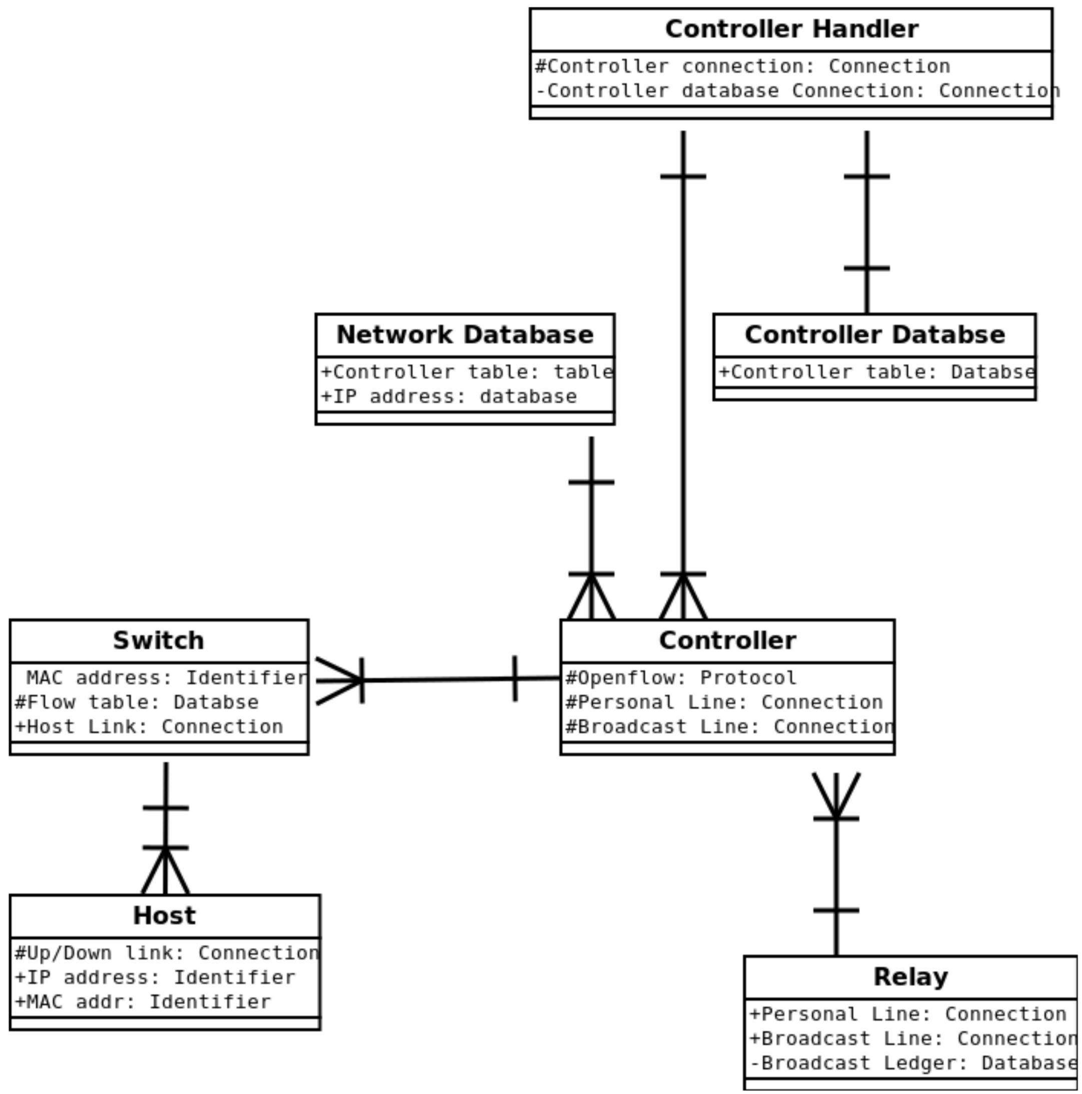
Legend:

1) Ri: *i*th relay
2) Ci: *i*th controller
3) Si: *i*th Switch
4) h: generic host
5) H: Database handeller
   for controllers

How the Architecture works?

1) Database docker image starts
2) Controller handler start connecting to ‘controllers’ database.
3) Controllers boot sending information to controller handler to be registered.
4) Mininet fetches the Controller Database
5) User Input for the required number of hosts and switches.
6) Random pairing for switches and hosts using the libsodium library.
7) Random number of hosts connected to the switches.
8) Topology is described as a 3d array, where the entries in database are separate tables with each controller has a separate table with its IP as the name for each table.
9) Class A Ips are automatically assigned by the Mininet.
10) Static Ips assigned to each controller on connection.
11) Standalone controller connection through relay.
12) Network database accessed by controller upon the connection.

ER Diagram


Controller Handler
#Controller connection: Connection
-Controller database Connection: Connection
Network Database
+Controller table: table
+IP address: database
Controller Databse
+Controller table: Databse
Switch
MAC address: Identifier
#Flow table: Databse
+Host Link: Connection
Controller
#Openflow: Protocol
#Personal Line: Connection
#Broadcast Line: Connection
Host
#Up/Down link: Connection
+IP address: Identifier
+MAC addr: Identifier
Relay
+Personal Line: Connection
+Broadcast Line: Connection
-Broadcast Ledger: Database

Thank You!

Prepared and Presented by:
Naman Arora
RA1511003010235
Nikhil Gupta
RA1511003010245