Evaluation of SDN Controller and Its Impact on Information-Centric Networking (ICN)

Overview, Use Cases and Performance Evaluation

Karim Md Monjurul

27-12-2018

School of Computer Science, Beijing Institute of Technology

Table of contents

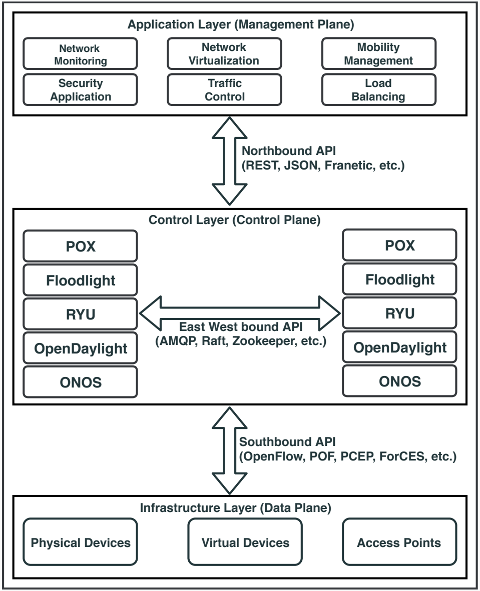
Introduction

What is an SDN Controller

An SDN controller

- is known to be the **"brain"** of the network in a software-defined networking environment
- relays information to the switches/routers through **Southbound API (SBI)** and the applications logic through **Northbound API (NBI)**
- can be a single application that manages **network/traffic/packet flows** to enable intelligent and automated networking
- also known as **Network Operating System (NOS)** that go beyond managing flow control and does multiple operations of the existing network

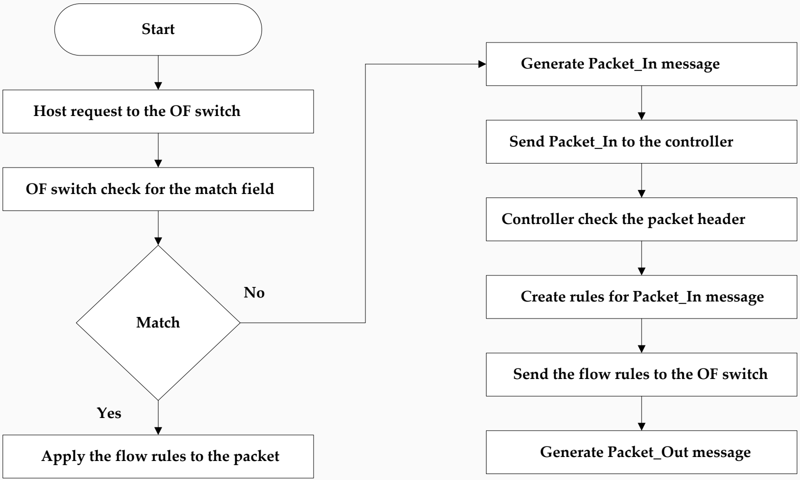
General Overview of SDN Architecture



Visual Representation of an SDN Controller

www.slideshare.net/PetervanderVoort1/sdn-beyond-the-obvious-29656744

Communication Flows between Controller and Switch



Topology discovery in Software Defined Network” IEEE Communication Survey and Tutorial 2017 Vol 19

Programming Languages

Programming Languages of SDN Controller (C and C++)

- NOX was written using almost **32,000** lines of C++ codes
- C++ based controllers performs better in the **low-level** environment
- Better synchronization with Faster Packet Processing Data Planes like **DPDK (Data Plane Development Kit)** and **Netmap (framework for fast packet I/O)**
- C++ was used to build the core module of a number of controllers like **Ethane, NOX, Rosemary, OpenMUL, DCFabric, Onix**

Programming Languages of SDN Controller (Java)

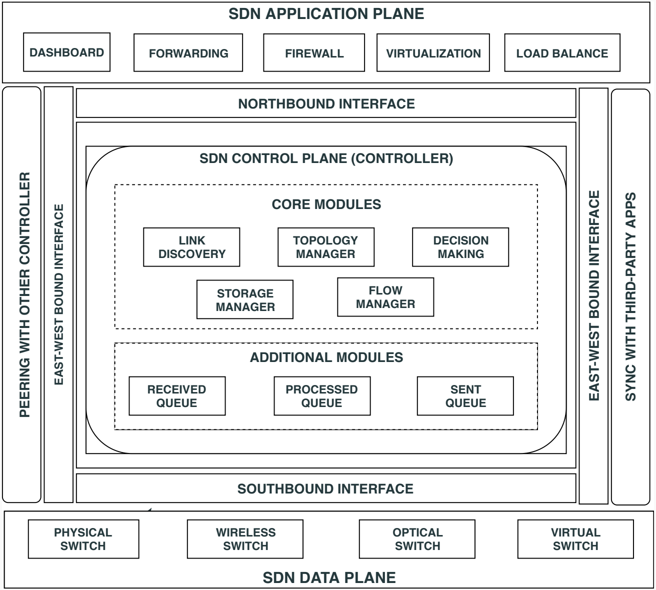
- Java-based controllers are ahead of the competition when it comes to **Multithreading** and **Parallelism**
- **Automatic Memory Management** and **Platform Independency** are two primary factors behind the selection of Java-based industrial-ready controllers
- Two of the most widely adopted controllers developed in Java. **ONOS** has been widely utilized in **Wide Area Networks** whereas **OpenDaylight** is more suitable for **Data Centers** and **Optical Network**

Programming Languages of SDN Controller (Python and Oth

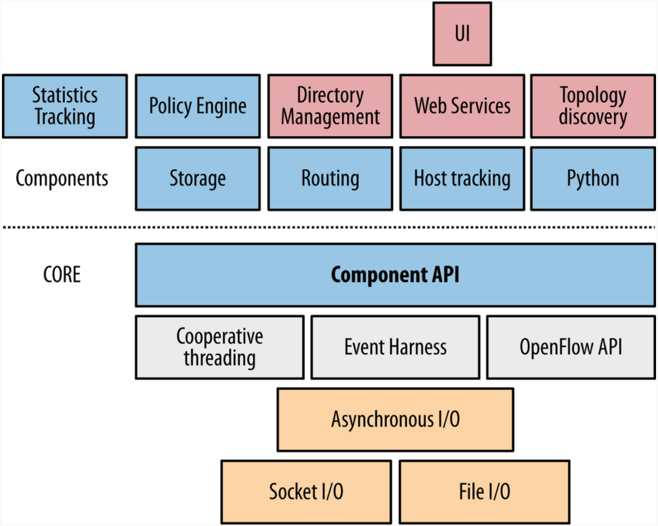
- Python-based Controller offers faster Compilation and Debugging
- Offers Simplified Scripting and Stitching together other pieces of code
- Extensive range of other programming languages used to develop SDN Controllers. Example: **JavaScript, Ruby, Haskell, Go and Erlang**

Components and Use Cases of SDN Controllers

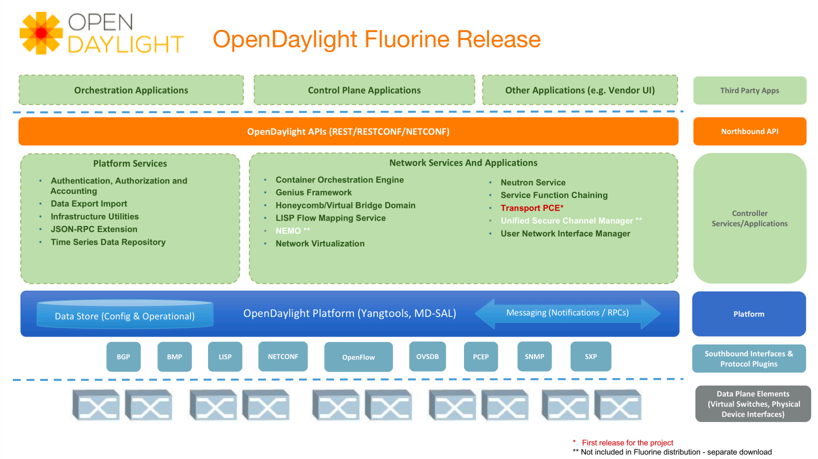
Core Components of an SDN Controller



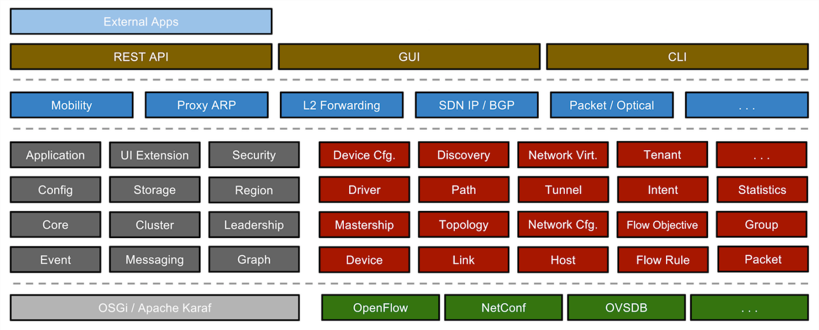
NOX Architecture



OpenDaylight Components



ONOS System Components



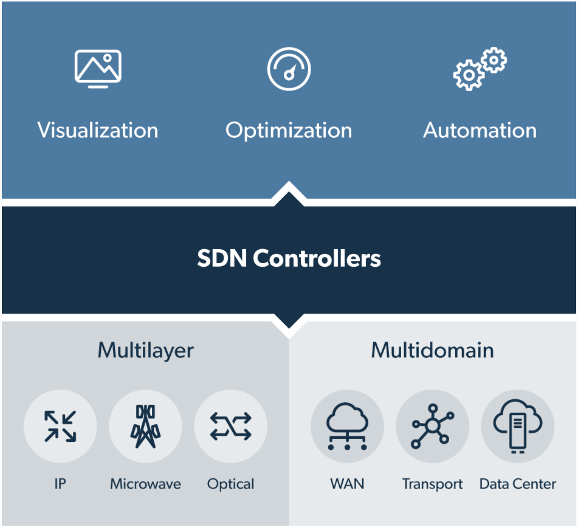
Feature comparison of Different Controllers

	POX	Ryu	Trema	FloodLight	OpenDaylight
Interfaces	SB (OpenFlow)	SB (OpenFlow) +SB Management (OVSDB JSON)	SB (OpenFlow)	SB (OpenFlow) NB (Java & REST)	SB (OpenFlow & Others SB Protocols) NB (REST & Java RPC)
Virtualization	Mininet & Open vSwitch	Mininet & Open vSwitch	Built-in Emulation Virtual Tool	Mininet & Open vSwitch	Mininet & Open vSwitch
GUI	Yes	Yes (Initial Phase)	No	Web UI (Using REST)	Yes
REST API	No	Yes (For SB Interface only)	No	Yes	Yes
Productivity	Medium	Medium	High	Medium	Medium
Open Source	Yes	Yes	Yes	Yes	Yes
Documentation	Poor	Medium	Medium	Good	Medium
Language Support	Python	Python-Specific + Message Passing Reference	C/Ruby	Java + Any language that uses REST	Java
Modularity	Medium	Medium	Medium	High	High
Platform Support	Linux, Mac OS, and Windows	Most Supported on Linux	Linux Only	Linux, Mac & Windows	Linux
TLS Support	Yes	Yes	Yes	Yes	Yes
Age	1 year	1 year	2 years	2 years	2 Month
OpenFlow Support	OF v1.0	OF v1.0 v2.0 v3.0 & Nicira Extensions	OF v1.0	OF v1.0	OF v1.0
OpenStack Networking (Quantum)	NO	Strong	Weak	Medium	Medium

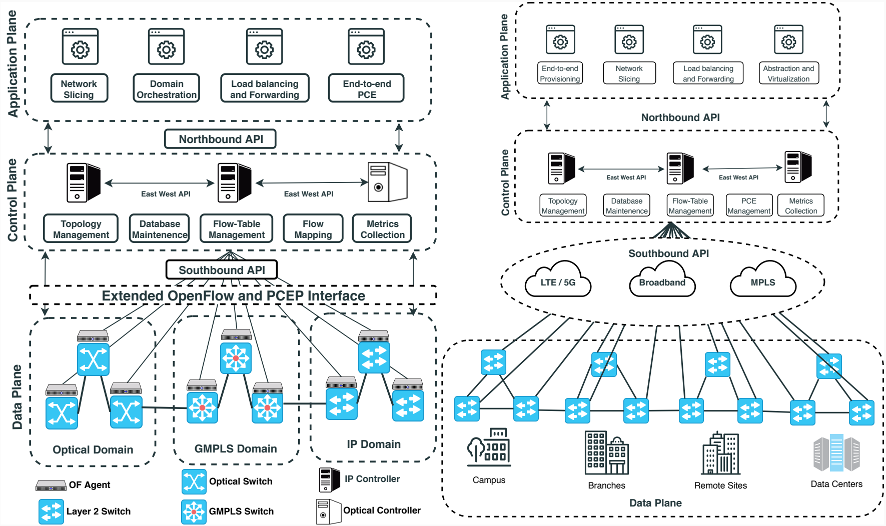
Application comparison of Different Controllers

Applicability	OpenDaylight	ONOS	Ryu	Trema
Documentation	Good	Medium	Poor	Poor
Management interfaces	SB (OpenFlow) NB (REST, JAVA RPC)		SB (OpenFlow, Management via OVSDB and JSON) NB (REST)	SB (OpenFlow)
Routing	Yes	Yes	Yes	Yes
Traffic Engineering	Yes	Partial	Partial	Partial
Service Insertion/Chaining	Yes	Partial	Partial	No
Load Balancing	Yes	Partial	Partial	No
Network Monitoring	Yes	Yes	Yes	Partial
Modularity	High	Medium	Medium	Medium
TLS Support	Yes	Yes	Yes	Yes
Openstack Networking	Medium	Medium	High	Week
Open Source	Yes	Yes	Yes	Yes
GUI	Yes	Yes	Yes with RES via ryu.app.gui_topology.gui_topology	No

SDN Controllers in Different Sectors



SDN Controller in Optical and Wide Area Network



Performance Evaluation of SDN Controller

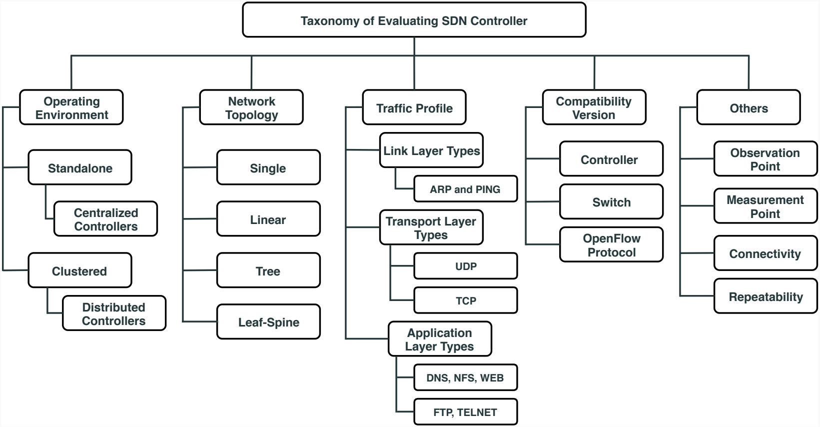
Why Evaluating the Performance of a Controller

- To **Maximize the Performance** with **Available Physical Resources**
- To Evaluate **Controller-Switch Communication** Efficiency
- To Understand the Impact of **Topology**
- To Measure the **Reliability** of **Trustability** of Controller

Metrics to be Considered Evaluating Controllers

- **Throughput**
- **Latency**
- CPU and Memory Utilization
- Round Trip Time
- And Many More

Taxonomy of Evaluating a Controller



Lists of Performance Testing Tool

- CBench
- PktBlaster
- OFNet
- Others: WCbench, OFCBenchmark, OFCProbe, HCProbe.

What is CBench

- Cbench emulates a configurable number of **OpenFlow switches** that all communicate with a single OpenFlow controller
- Each emulated switch sends a configurable number of **new flow messages** to the Controllers
- Waits for the appropriate **flow setup responses** and records the difference in time between **request and response**
- It supports two modes of operation: **Latency and Throughput**

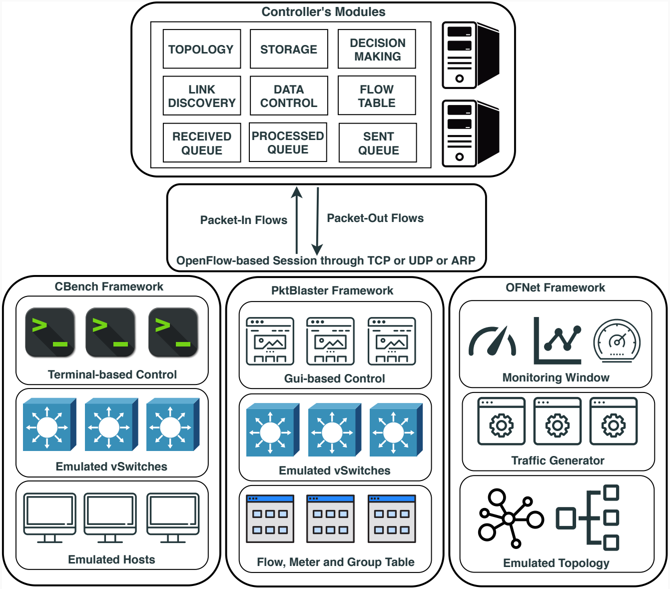
Key Features of PktBlaster

- Real world Network Emulation for SDN
- **Flow-mod** and **Packet-out** based Performance Benchmarking
- Supports both OpenFlow 1.0 and 1.3
- User-friendly **GUI**
- Comprehensive Test Results, Analysis and Comparison

Key Features of OFNet

- Function as a **Network Emulator**, **Debugging Framework** and **Controller Testing** Tool
- Tests can be done through **Customized Topology**
- Features In-built **Traffic Generator**
- Have Additional Metrics other than Latency and Throughput. For Example: **Flow Generations Rate, Flow Failure Rate, vSwitch CPU utilization and Average RTT**

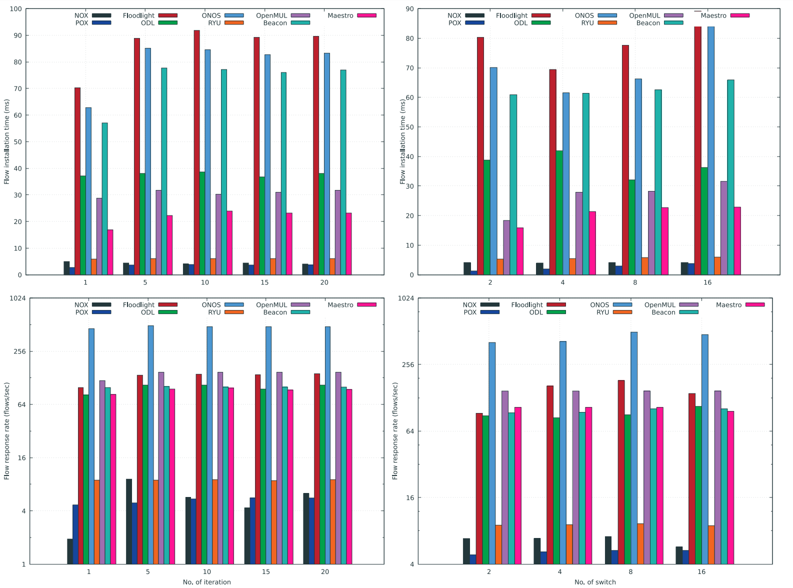
Architecture of Benchmarking Tool



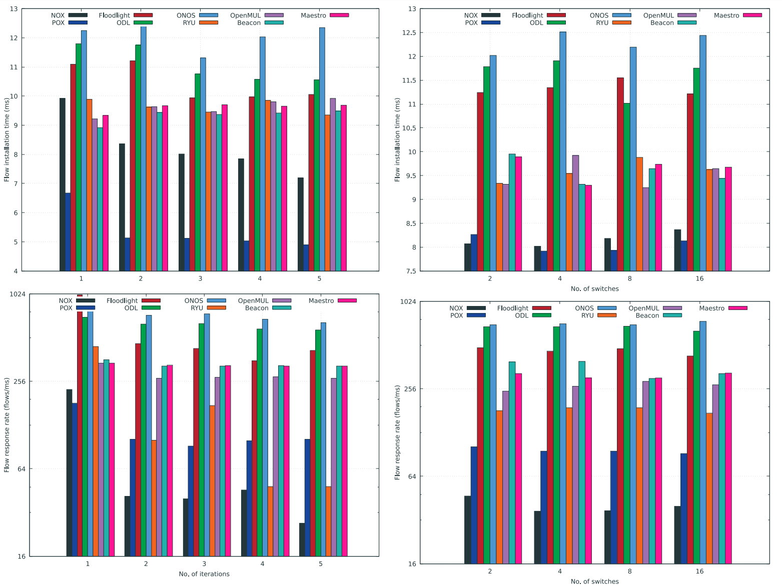
Benchmark Parameters

Tool	Parameter Values	
CBench	Number of Switch	2, 4, 8, 16
	Number of Test Loops	20
	Test Duration	300 sec
	MAC Addresses per Switch (Hosts)	64
	Delay between Test Intervals	2 sec
PktBlaster	Number of Switch	2, 4, 8, 16
	Test Duration	300 sec
	Number of Iterations	5
	Traffic Profile	TCP
	Ports per Switch (Hosts)	64
	Flow Counts per Table	65536 (Default)
	Packet Length	64 bytes
OFNet	Number of Hosts	20
	Number of Switchs	7
	Desired Traffic Rate	100 flow/sec
	Flow measured by	Packet-out & Flow-Mod
	Total Test Duration	300 sec

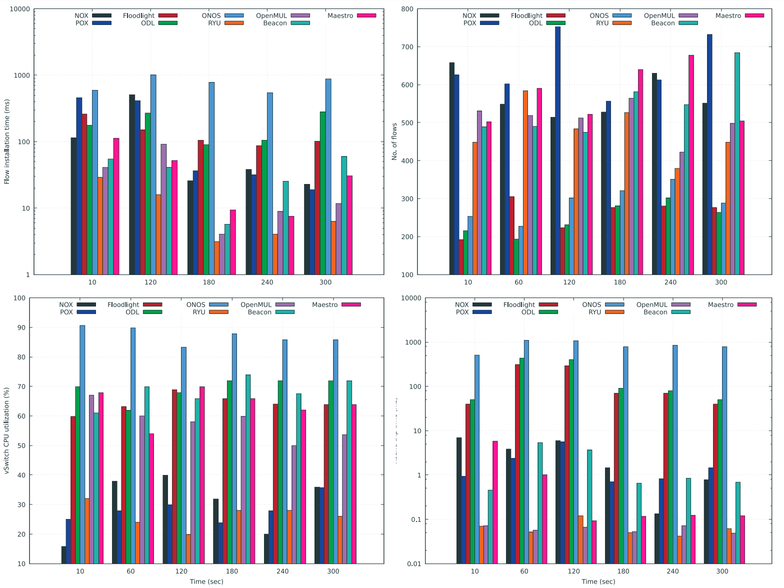
Performance Comparison using CBench



Performance Comparison using PktBlaster



Performance Comparison using OFNet



SDN Controller in an ICN Scenario

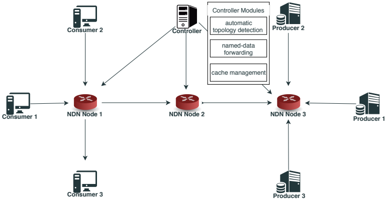
Why We Need to Combine SDN and ICN

- **Automated** and **Intelligent** Content Delivery
- Content-based Mobility Support in **5G** and **Vehicular** Network
- In-network Caching based on **Content Popularity**
- Content-based **Traffic Engineering**

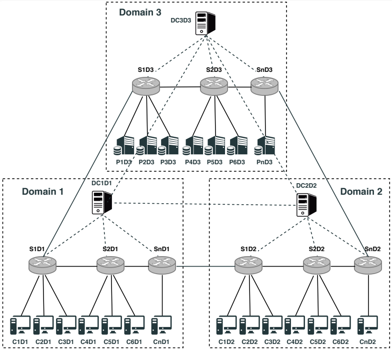
Types of SDN-ICN Architecture

- Centralized Architecture
- Distributed Architecture
- Clean-State Architecture
- Overlay and Underlay Architecture

SDN-ICN Architecture



SDN-ICN Architecture with Centralized Controller



SDN-ICN Architecture with Distributed Controllers

Controllers’ Role in SDN-ICN Architecture

- Topology Discovery and Statistics Collection
- Name-based Content Forwarding
- Content Discovery and Caching

Intended Research Findings from SDN-ICN Implementation

- Improve Caching Scheme
- Controller-to-Controller Communication through Contents
- Improved Content Security

Tools We Intend to Use

- Testbed using ndnSIM
- Controller App
- ICN Node App

Conclusion

Google had big problems Regarding High financial cost Managing their Data Centers

- Hardware and software upgrade
- Over provisioning (fault tolerant)
- Manage large backup traffic
- Time to manage individual switch
- A lot of men power to manage the infrastructure

Final Words-2

What are the Problems They were having

- Delay caused by rebuilding connections after Link Failure
- Slow to rebuild the routing tables after Link Failure
- Difficult to Predict what the New Network may perform

How They Solve these Problems

- Built their hardware and wrote their own software for their internal data centers
- Surprised the industries when Google announced SDN was possible in production

How did they do it?

B4: Experience with a Globally-Deployed Software Defined WAN

Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, Jonathan Zolla, Urs Hölzle, Stephen Stuart and Amin Vahdat
Google, Inc.
b4-sigcomm@google.com

Questions?