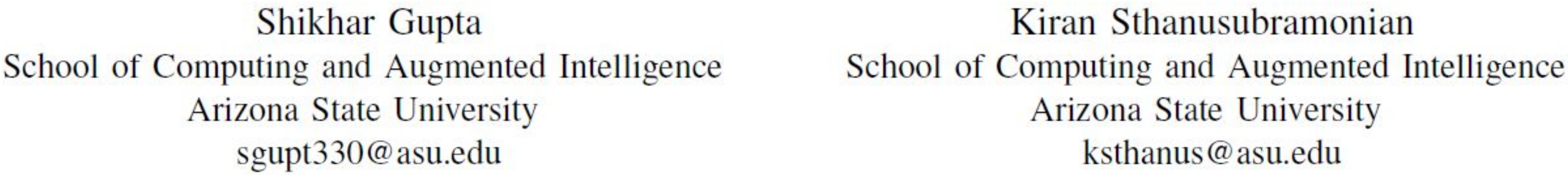
CSE 534 SLN 98070, Fall 2022

Advanced Computer Networks

Dynamic Routing using P4 Switches and the FABRIC Measurement Framework

Authors:



Shikhar Gupta
School of Computing and Augmented Intelligence
Arizona State University
sgupt330@asu.edu

Kiran Sthanusubramonian
School of Computing and Augmented Intelligence
Arizona State University
ksthanus@asu.edu

Problem Statement

- Setting up a Dynamic Routing Experiment for simple network topologies on the **FABRIC Testbed**.
- Use P4 to create a programmable switch backbone.
- Create a dynamic routing algorithm which leverages:
  - **FABRIC Measurement Framework Library (MFLib):** collect node-level metrics,
  - **Aggregate the metrics** at the Controller to find an **optimal routing path** between two hosts on the network,
  - **P4 Runtime:** dynamically configure the optimal routing path on the switch topology.

Related Works

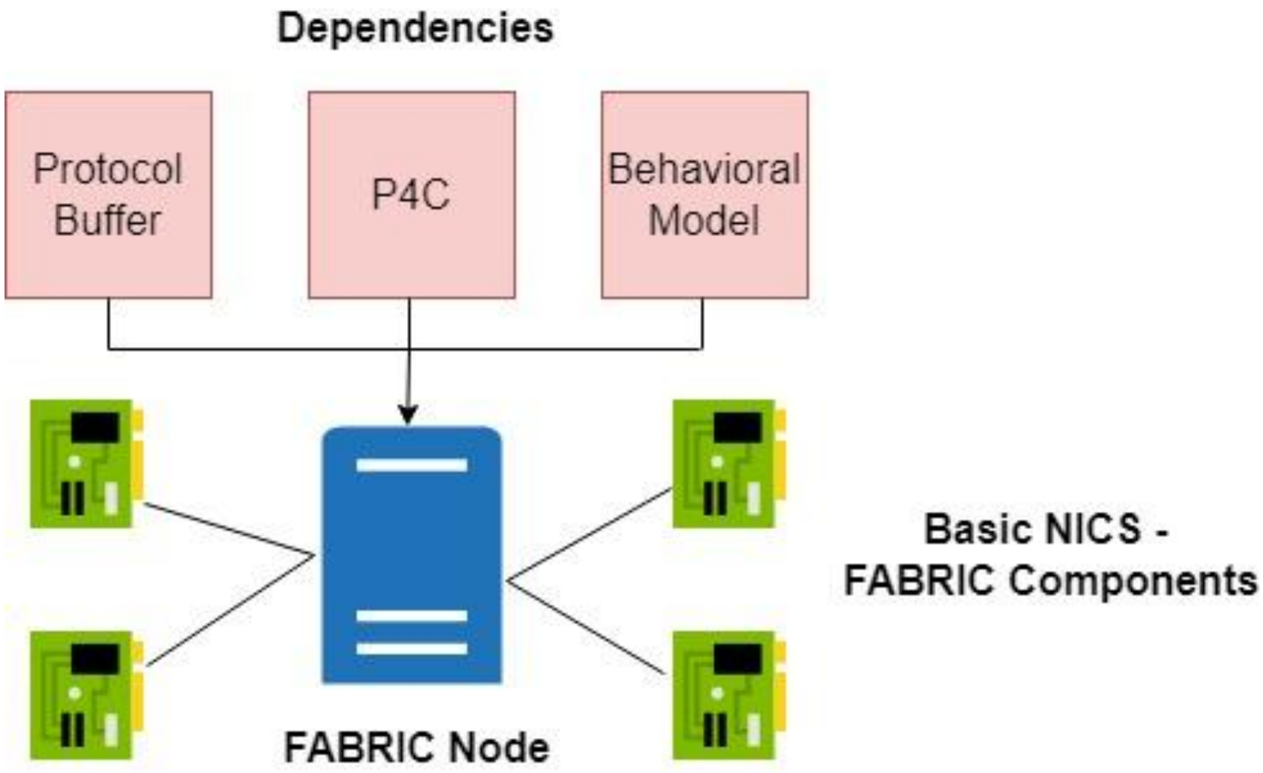
- **P4 in general:**
  - DDoS Attack Prevention
  - Load balancing
  - Caching on the Data Plane
- **P4 for dynamic routing solutions:**
  - RouteScout: Hybrid System for BGP Routing
- **The In-band Network Telemetry (INT) Specification:**
  - Very popular implementation of P4
  - Efficient network monitoring systems using INT developed

P4

- *De facto* method for data plane programmability.
- Extension to Software Defined Networking (SDN).
- Help to realize OpenFlow 2.0
  - **Tackle Protocol Ossification**
  - **Hardware Independence.**
- Advantages:
  - **Active Networking** - increase in compute and storage power
  - Critical in building **robust & secure** networks in real-time!

P4 - FABRIC Implementation

- FABRIC Nodes are converted to ad-hoc P4 Switches. How?

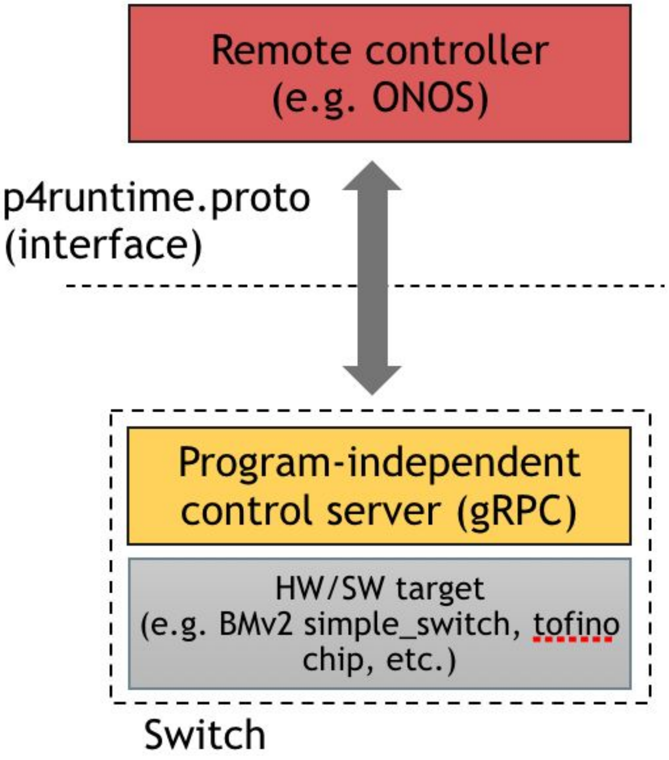


P4 - Our Usecase

- A simple Routing P4 Program is compiled and deployed on each switch on the network. Target: **simple-switch**
- IPv4 based Routing Table:
  - **Key:** Destination IPv4 Address,
  - **Parameters:** next_hop MAC Address, Egress Port #
- P4 Match-Action Table commands of interest:
  - **table_add:** adds a new table entry
  - **table_modify:** modifies a table entry of an existing handle
  - **table_delete:** deletes a table entry of an existing handle

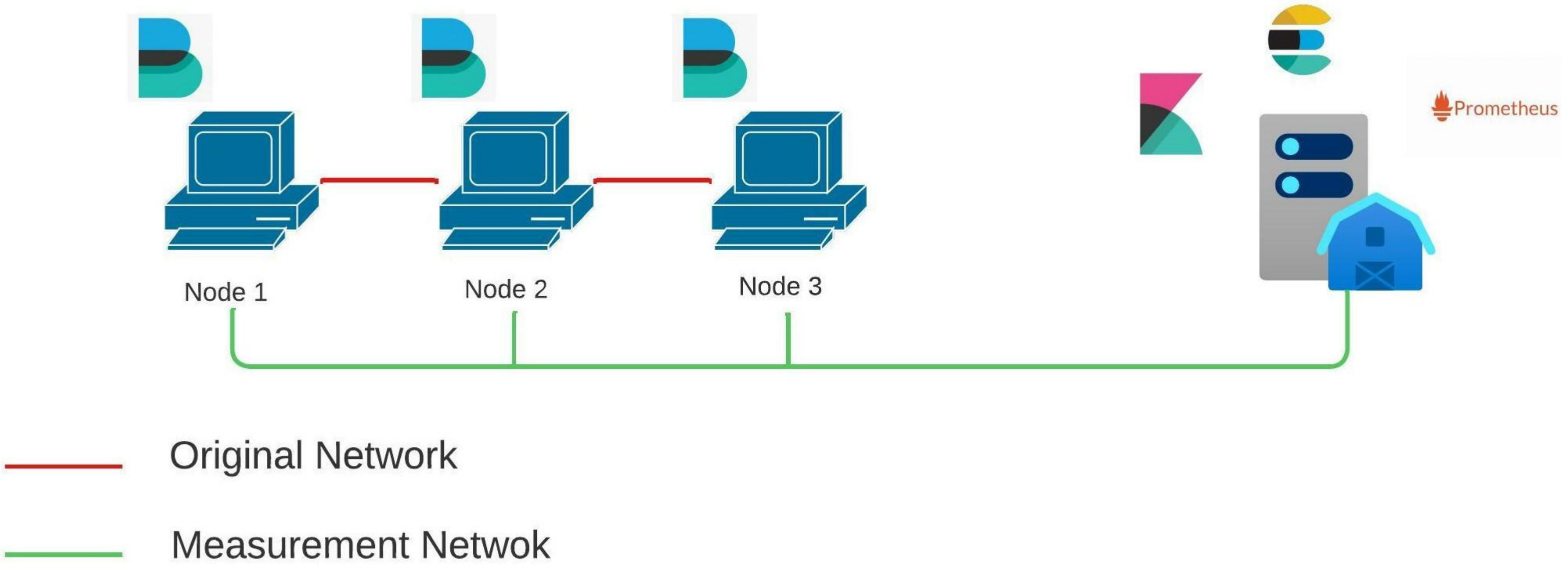
P4Runtime

- Control Plane Specification for P4
- Two popular workflows:
  - Set Forward Pipeline Information
  - Get Forward Pipeline Information



FABRIC Measurement Framework Library

1. **Prometheus:** Metrics Collection
2. **Elasticsearch:** Stores the Metrics
3. **Kibana & Grafana:** Dashboards and Visualizations



Additional Tools used

- **Scapy:** Python module for Traffic Generation & Packet Sniffing
- **Prometheus Client:** Used for writing customized queries to extract real-time metrics gathered on the Measurement Node

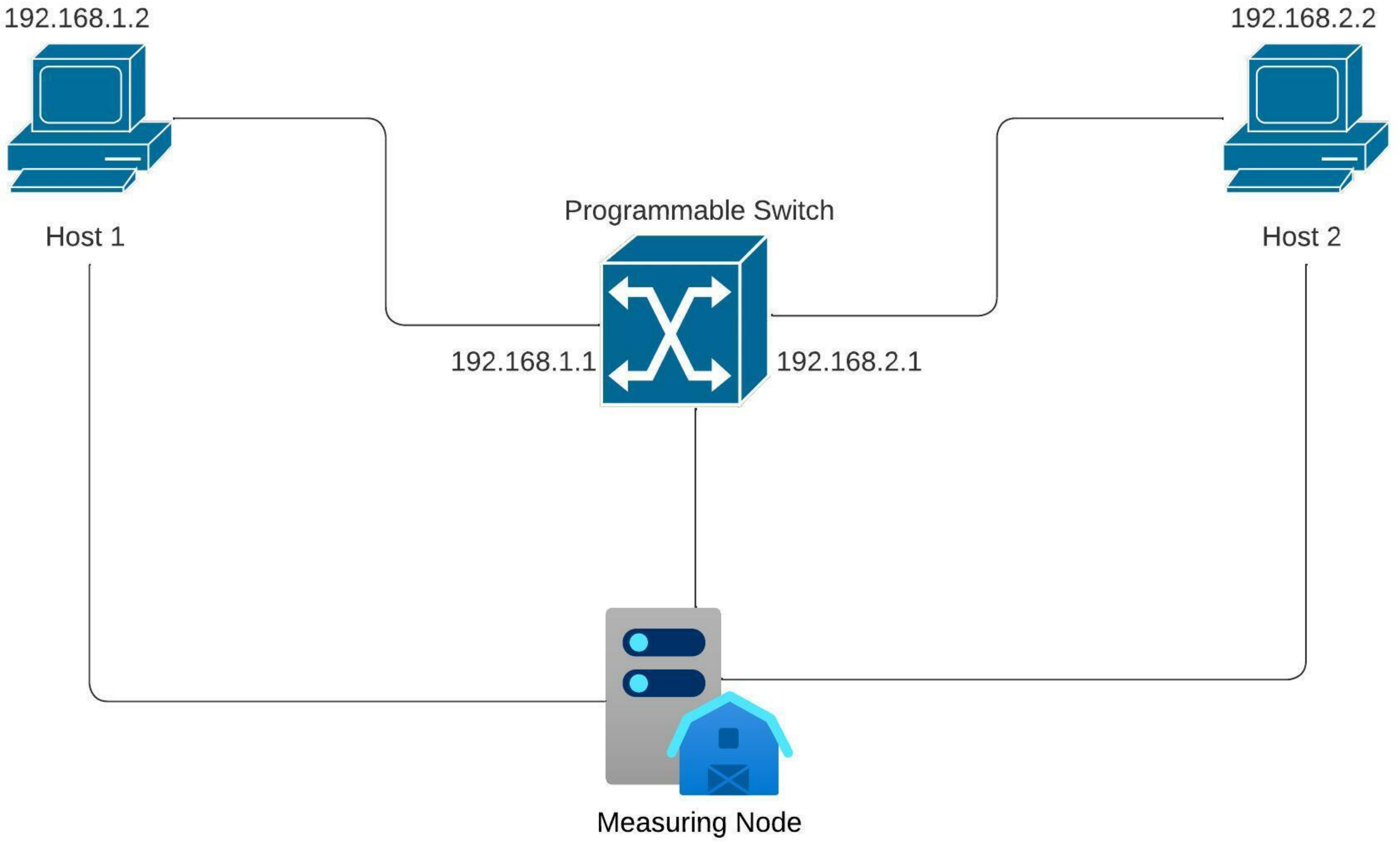
Dynamic Routing Algorithm

1. Incoming packet transmission request between two hosts to the controller.
2. Controller sets the optimal routing path among multiple available routes between the hosts:
   a. **Query & Aggregate** the **available bandwidth / queueing buffer size** on each egress ethernet interface present on the routes
   b. **Optimal Route:** Path with Maximum Available Bandwidth / Minimum Queuing Buffer Size
3. Configure the Optimal Route on the P4 Switches using P4Runtime

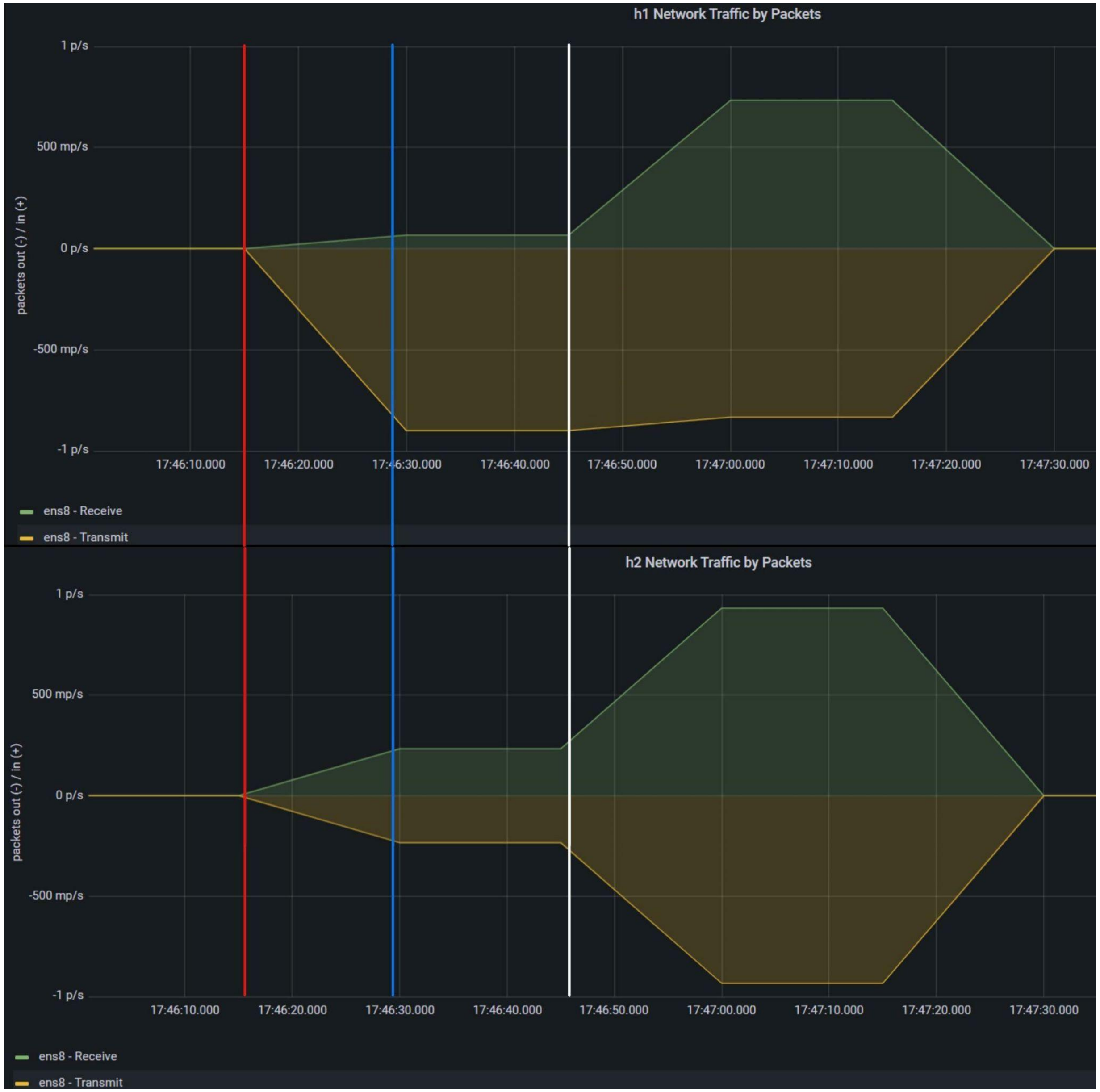
Dynamic Routing Algorithm - Assumptions

1. The available routing paths between any two given hosts are **known beforehand.**
2. Limited number of available routing paths between the two hosts.
3. Complete information about the ingress and egress ports of the network devices available to the Controller.

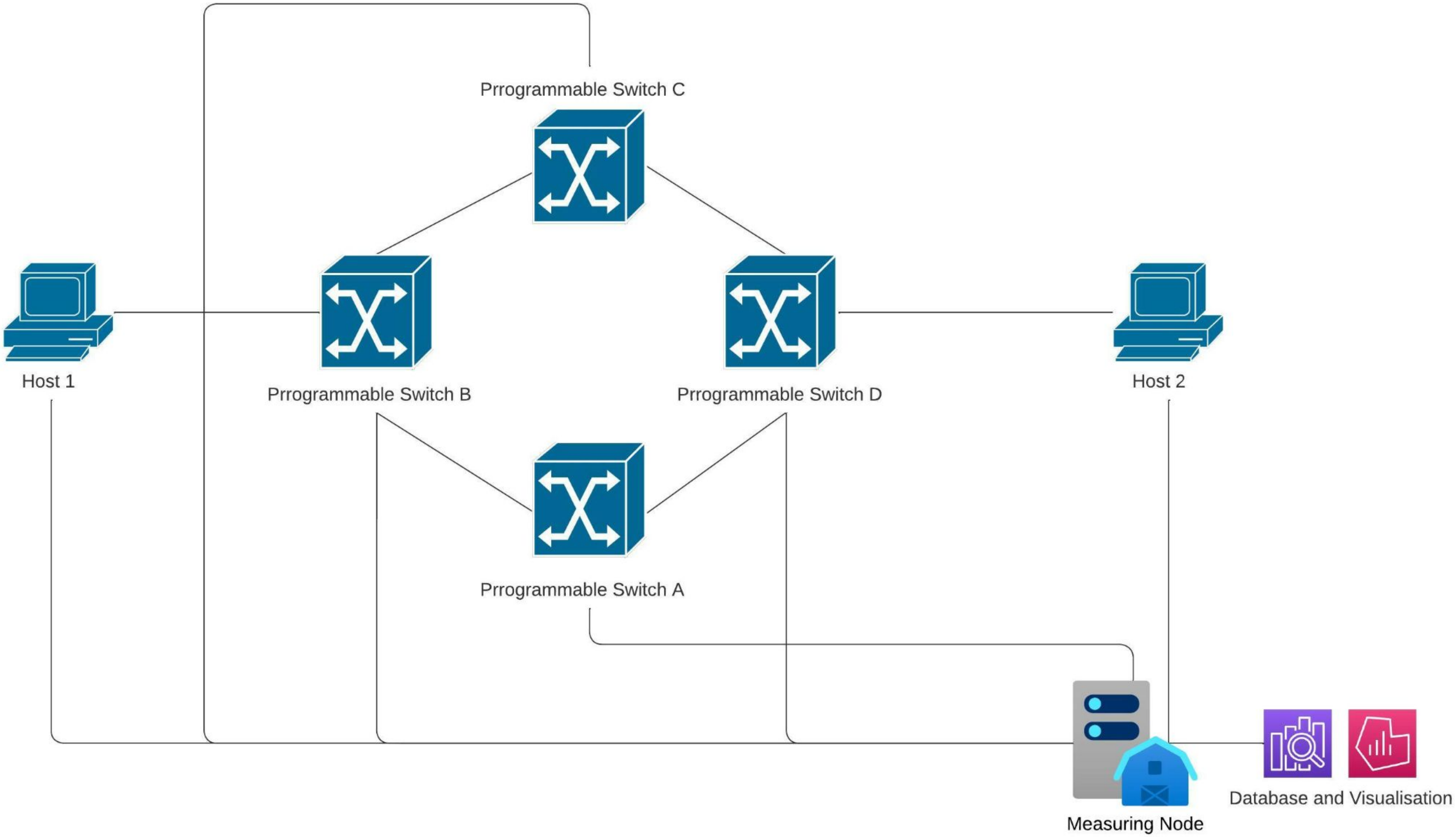
Experiment #1



Experiment #1 - Results



Experiment #2



Experiment #2 - Expected Results

- Compare the Round-trip Time (RTT) between the hosts:
    - without the Dynamic Routing algorithm - use pre-configured route / pick any arbitrary route,
    - with the Dynamic Routing algorithm.
- Visualize the slice-level Throughput using MFLib & Grafana.

Challenges Faced

1. Setting up MFLib:
    - Minimal codebase documentation
    - Required to change few minor implementations on the codebase
    - Dependency issues with Python versioning
    - We had to find the required credentials used for accessing Prometheus Client running on the Measurement Node
2. Insufficient resources on how to deploy P4 Programs / P4Runtime / modifying match-action table entries, etc.

Remaining Tasks

- Implement the Optimal Route Finding algorithm & conduct necessary experiments:
  - The Prometheus queries are ready!
- Implementing P4Runtime on the FABRIC Testbed:
  - An alternative approach using FABRIC Library (FABLib) Commands has been successfully implemented!

Potential Improvements

- Use Explicit Congestion Notification (ECN) to detect congestion on one path:
    - With ECN, we can reconfigure to an optimal routing path only when necessary.
    - ECN can be implemented using P4.
    - Minimizes path configuration requests.
- Repeatedly aggregate and cache the metrics on the Controller to make faster route reconfigurations.

Our Contributions

- Dockerize the entire P4 & P4Runtime custom implementation on FABRIC.
- Open-source our implementation for setting up customized topologies with setup of MFLib.