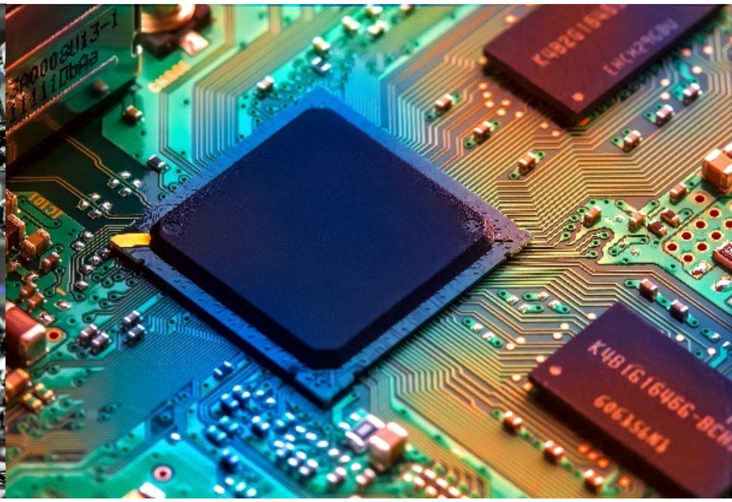
Optimizing Networking Approaches using P4

Debopam Sanyal (Advisor: Professor Sibin Mohan)

Undergraduate, Computer Engineering



ILLINOIS

Electrical & Computer Engineering

GRAINGER COLLEGE OF ENGINEERING

Introduction

- Networking; forwarding packets is the key.
- Optimization is very important (Real-Time Systems).
- Software-defined networks.
- Implemented 3 different packet forwarding algorithms.
- P4 vs OpenFlow; NetFPGA SUME board.

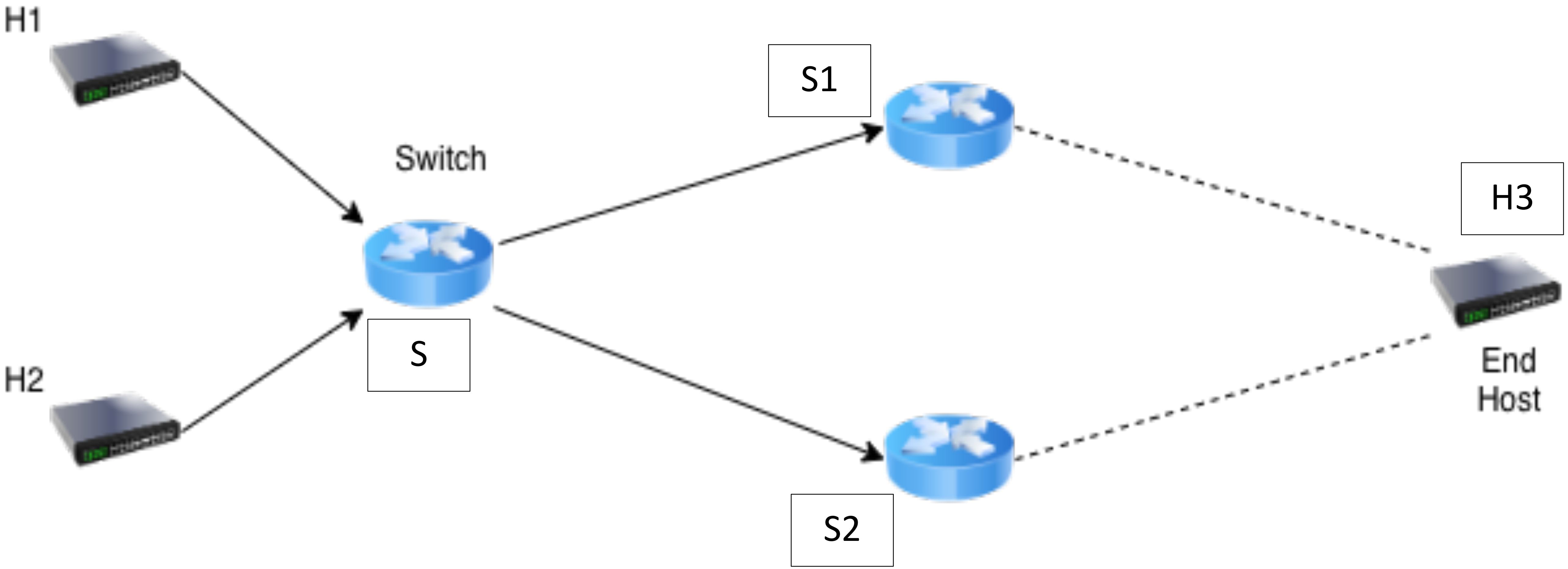
Background

- Switches are prevalent in network connections.
- Emergence of SDNs and its benefits.
- Rise of OpenFlow as a research tool.
- Its major drawback: protocol-dependent.
- P4 is developed; its main goals.
- Link failure recovery and congestion notification.
- Fast failover mechanism.

Conditional Forwarding and Experiments 1&2

- Forward packets based on a condition. Per-packet analysis.
- Three main function of switches in my experiments.
- Experiment 1: Use H1’s path for forwarding H2 packets until H1 packets arrive after a time window *t*. H1’s path is more optimal. H1 has a higher priority than H2.
- Experiment 2: Priority doesn’t matter if the more critical flow is given preference. That is if H2 has a higher packet rate than H1, H2 gets to use the optimal path (S1).

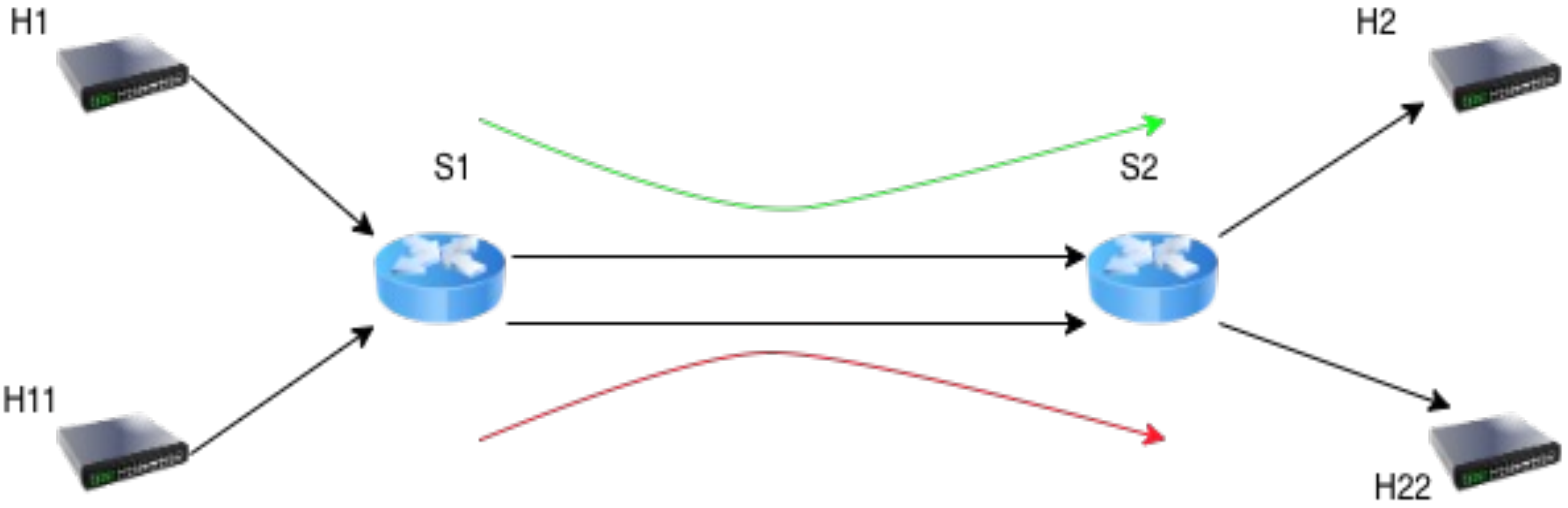
Topology for Experiments 1&2



Experiment 3

- Explicit Congestion notification at end host.
- One link between S1 and S2 used by both flows.
- If flow rate in the link becomes more than threshold *r*, S2 (connected to the end host) notifies S1.
- Threshold *r* is significantly less than link capacity *R*.
- S1 keeps reducing incoming packet rate at *k* packets/sec^2.
- Does this until traffic rate in the link becomes $\leq r$.

Topology for Experiment 3



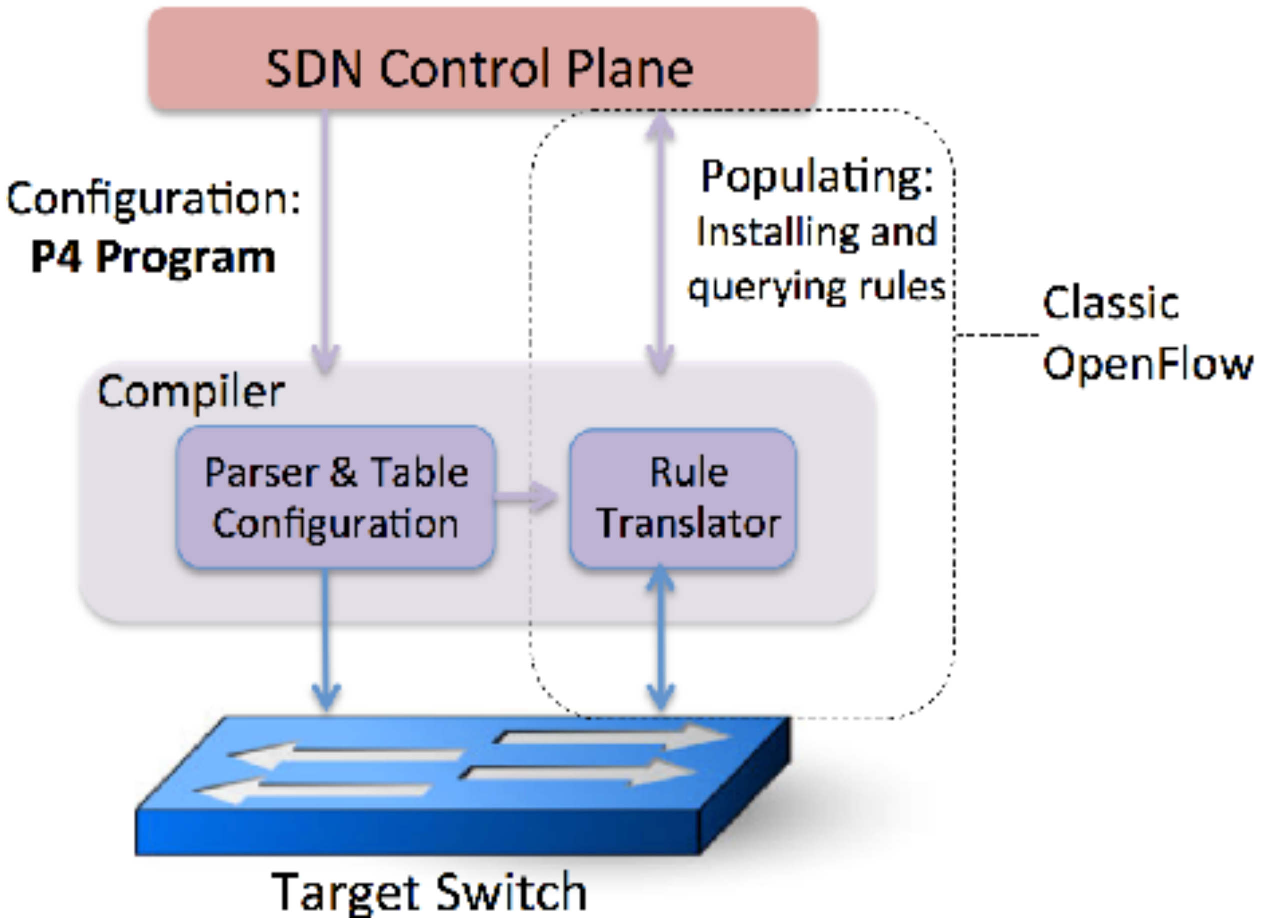
Results

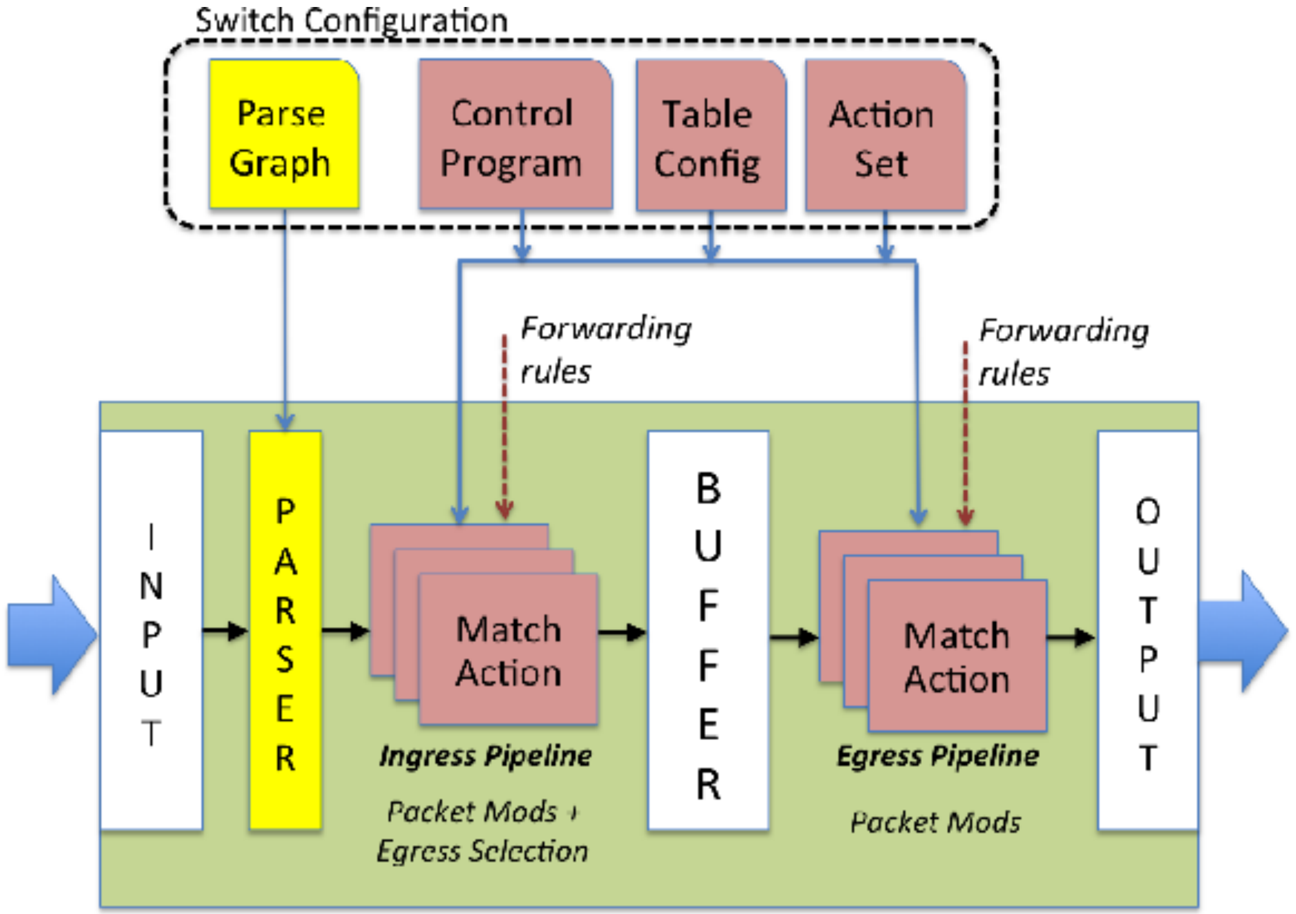
Experiment #	Components	Parameters	Benefits	Result
1	H1, H2, H3, S, S1, S2	t = 5 sec, 10 sec	Optimizes packet transmission in the network	Success
2	H1, H2, H3, S, S1, S2	No parameters	Provides end-to-end deadline guarantees	Success
3	H1, H11, H2, H22, S1, S2	R = 12 packets/sec, r = 10 packets/sec, k = 0.5 packets/sec^2	Ensures network connectivity	Success

Implementation in P4

- Define packet processing rules via P4 program.
- Based on an architecture description. I used V1 model architecture.
- Contains parser, MAP(has tables and algorithms), deparser.
- Ingress and egress ports responsible for forwarding.
- Used python to generate and receive packets at hosts. Also used to alter packet generation rates.

P4 Architecture Diagrams





Testing on Mininet and NetFPGA

- Compiled programs using P4 compiler. Tested on Mininet.
- Determined correctness by checking log files of switches.
- Checked the NetFPGA board for faults (Acceptance Test).
- Used Vivado and SDNet licenses to use the toolchain.
- Python files (test data) to verify correctness.
- P4-SDNet compiler creates HDL useful for debugging.
- Ran the experiments on SDNet simulation.

Implementation steps and Future work

- Test programs on the NetFPGA-SUME hardware.
- Modify recovery rule in Experiment 3. New link (H11-H22).