NS4: Enabling programmable data plane simulation

Meister Rados

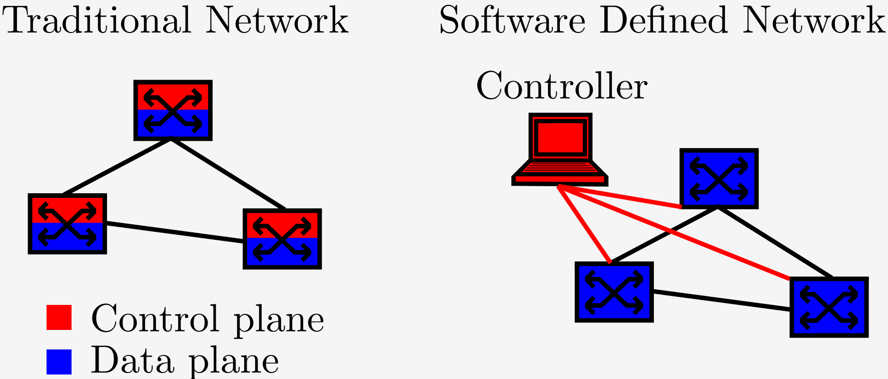
Institute for Telematics

July 18, 2018

Overview

SDN → Programmable Data Plane → P4 → NS4

- Centralized control view
- Separation of data plane and control plane
- Software defined network applications

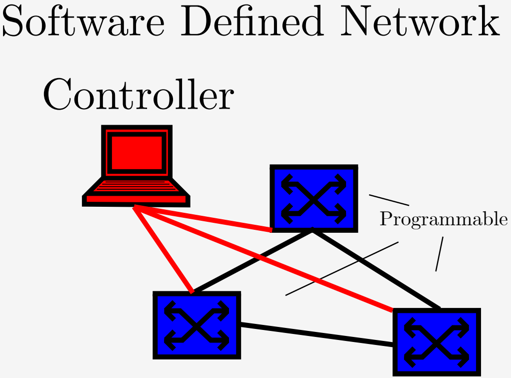


🟥 Control plane
🟦 Data plane

Overview

SDN → Programmable Data Plane → P4 → NS4

- Forwarding idea within SDN
- Networking devices can be programmed and become protocol independent

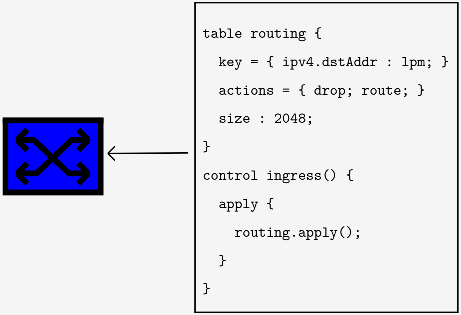


Overview

SDN → Programmable Data Plane → **P4** → NS4

- Domain-Specific language
- Used to describe how networking devices process arriving packets
- Target and protocol independence

P4 enabled network device



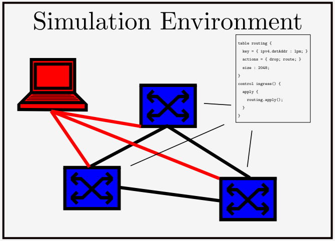
```
table routing {
  key = { ipv4.dstAddr : lpm; }
  actions = { drop; route; }
  size : 2048;
}
control ingress() {
  apply {
    routing.apply();
  }
}
```

Overview

SDN → Programmable Data Plane → P4 → NS4

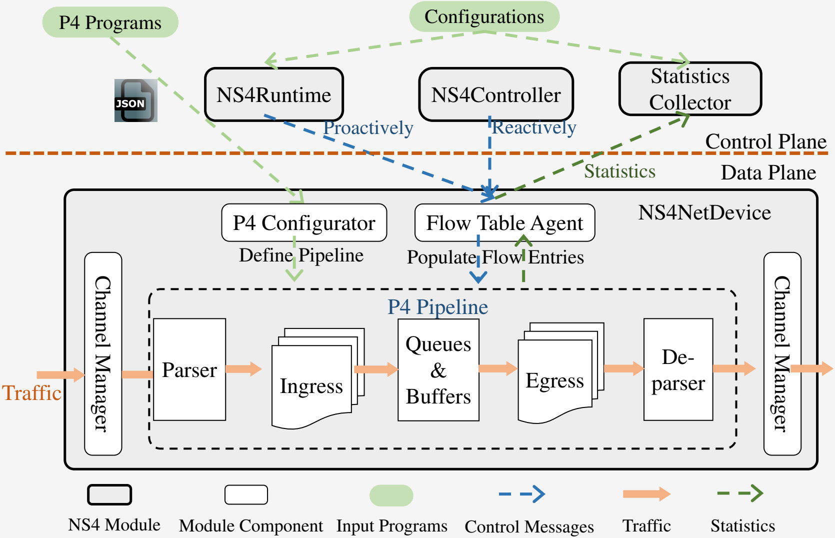
- NS4 is a simulator with support for multiple P4 enabled devices
- Operators can validate P4 protocol correctness and evaluate their efficiency
- Benefit of advantages of P4
- No library dependent simulator calls, leading to denser source code

Simulator

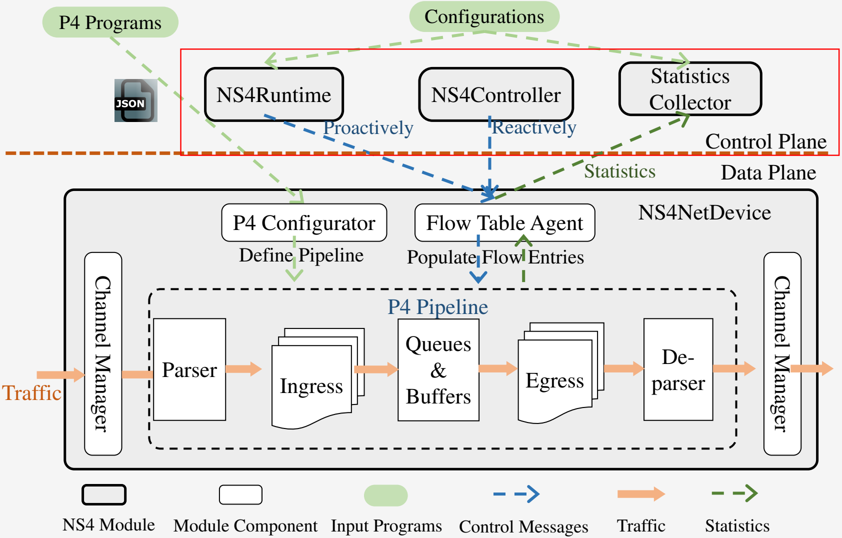


```
table routing {
  key = { ipv4.dstAddr : lpm; }
  actions = { drop; route; }
  size : 2048;
}
control ingress() {
  apply {
    routing.apply();
  }
}
```

NS4 Architecture: Overview



NS4 Architecture: Overview



P4 Programs

Configurations

NS4Runtime

NS4Controller

Statistics Collector

Proactively

Reactively

Control Plane

Statistics

Data Plane

P4 Configurator

Flow Table Agent

NS4NetDevice

Define Pipeline

Populate Flow Entries

P4 Pipeline

Channel Manager

Parser

Ingress

Queues & Buffers

Egress

De-parser

Channel Manager

Traffic

NS4 Module

Module Component

Input Programs

Control Messages

Traffic

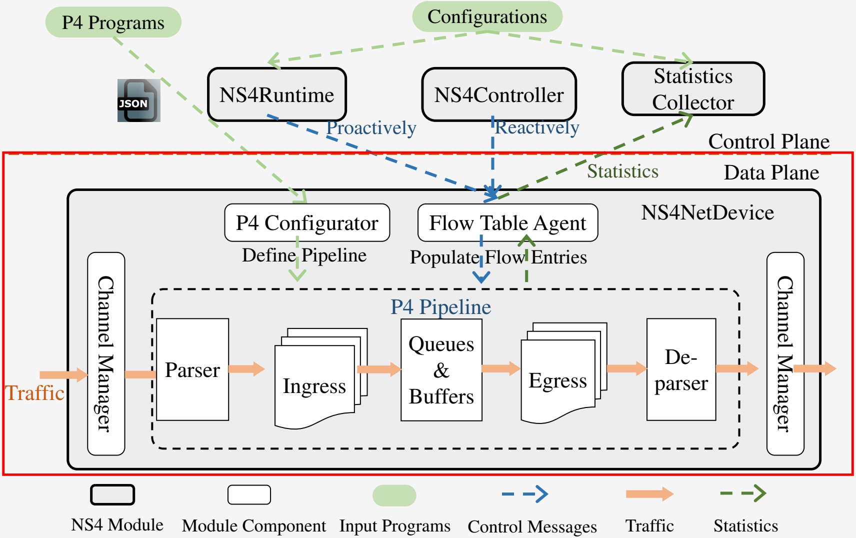
Statistics

Control Plane

Components

- NS4 Runtime
  - Proactively generates flow table entries from user configurations
  - Every line of the configuration is translated into flow table entries
  - So called “discrete population”
  - Can modify the flow table entries at runtime
- NS4 Controller
  - Designed to be optional
  - Can populate flow table entries reactively
  - Trivial entries that can be derived from network topology
  - So called “automatic population”
- Statistics Collector
  - Creates discrete events for gathering statistical information
  - Can set and reset counters for specific values in the flow tables

NS4 Architecture: Overview



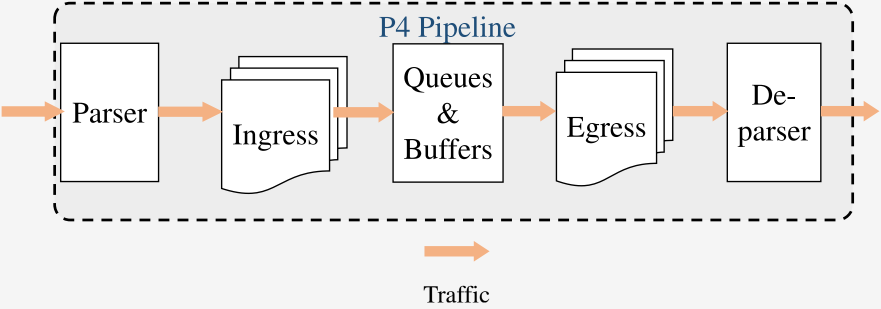
Data Plane of NS4

Module: NS4 Net Device
Components:

- P4 Configurator
  - Takes P4 programs and configures pipelines in NS4 Net Device
  - Line-by-line translation from P4 to discrete events
- Flow Table Agent
  - Populates flow table entries according to NS4Runtime and NS4Controller
  - Offers statistical information to the statistics collector
- Channel Manager
  - Eases communication between multiple net devices
  - Downward compatibility with traditional devices given

P4 Pipeline within NS4 Net Device

- P4 described ingress- and egress pipeline
- Encapsulated by a parser and deparser
- Queuing system and scheduler in between



Queuing System and Scheduler

- Queuing system decouples ingress pipeline from egress pipeline
- Multiple queues per NS4 Net Device, which one a packet passes through is determined by its metadata
- Scheduler dequeues packets by priority and passes them to the egress pipeline
- Allowing QoS and other features
- In case of equal priorities: Round-Robin

Workflow

Steps to conduct a simulation with NS4

1. Describe programmable data plane behavior using P4 language.
2. p4c compiles the P4 programs and passes them to the P4 configurator
3. Set up the control plane: configure flow table operations, determine statistics gathering
4. Install applications and define network topology
5. Trigger simulation

Performance

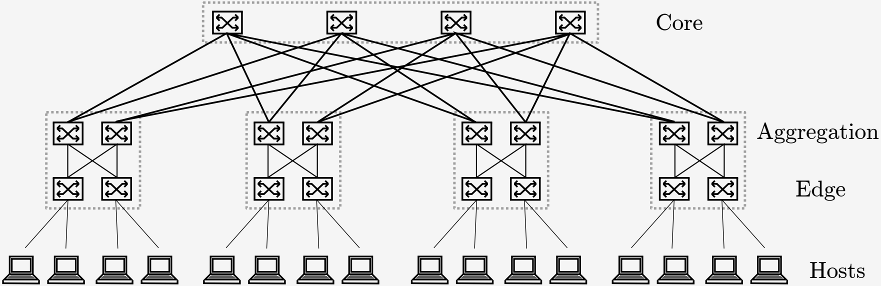
3 Aspects will be regarded:

1. Effectiveness, by simulating a case study: Silk Road
2. Efficiency, by simulating large scale networks
3. Code size in comparison to ns-3

Simulation Setup

Demonstration of effectiveness and efficiency of NS4

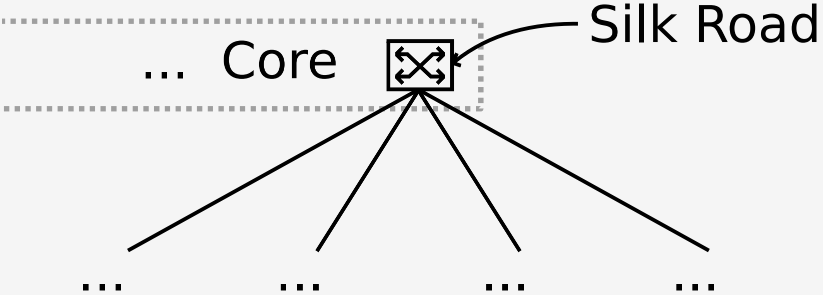
- Topology: Fat-Tree Network, line denotes depicts bandwidth
- Sender & receiver chosen randomly, but all hosts covered at least once
- Every flow has a size of 1MB
- Typical use case: data centers



1. Demonstrating effectiveness

Simulating Silk Road with NS4

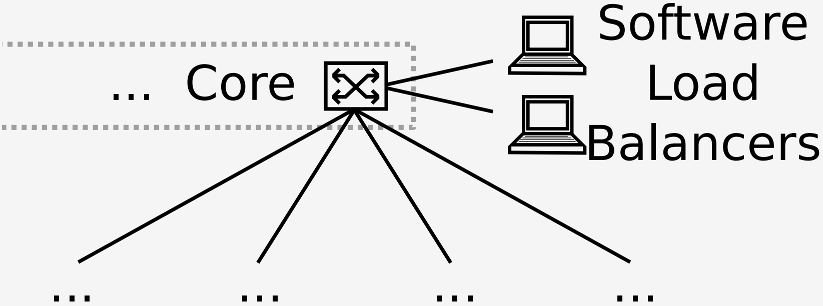
- Silk Road: an load balancing function implemented in P4
- Switches themselves decide how to forward incoming traffic
- Could not be simulated without NS4



SLB

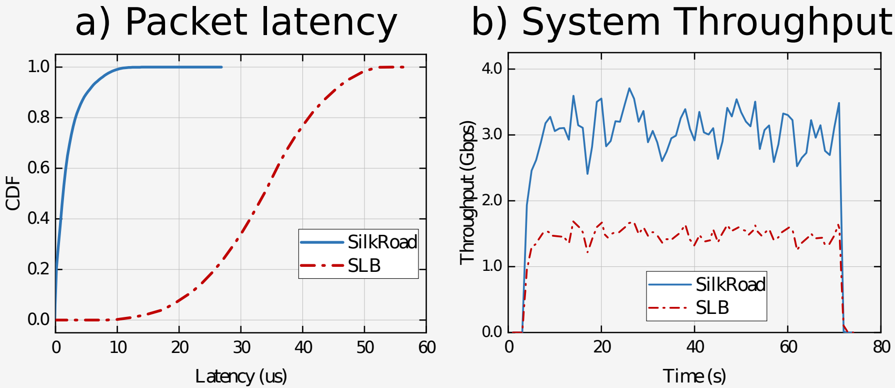
Software load balancing

- Load balancing done in software by dedicated load balancing servers
- Bringing high cost for dedicated devices
- Higher forwarding latencies due to redirection of packets
- Simulation with SLBs is a typical use case of ns-3



Evaluation

Silk Road compared with SLB implementation



a) Packet latency

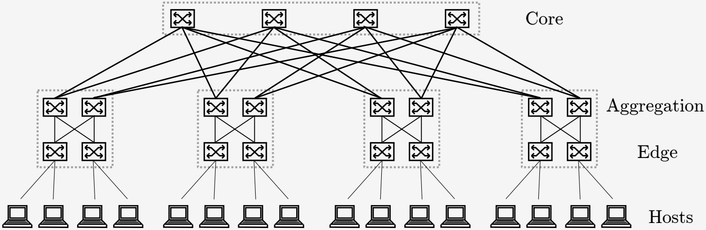
b) System Throughput

Executing hardware: Dell PowerEdge R370 (2x Intel Xeon-2620, 64GB RAM)

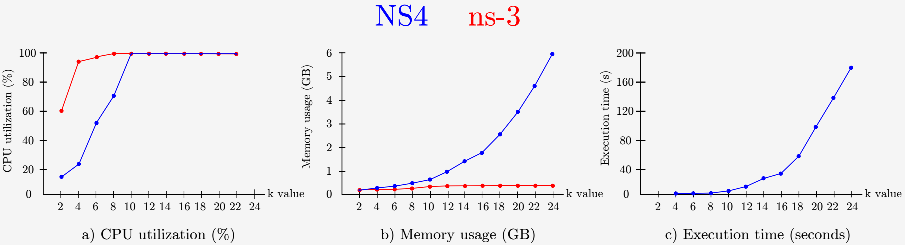
2. Demonstrating efficiency

Simulating large scale networks with NS4

- Topology: Fat-Tree Network
- Value of k will be increased from 4 to 24
- Measuring metrics: CPU usage, RAM utilization and execution time
- Simulation time: 100s



Hardware utilization



a) CPU utilization (%)

b) Memory usage (GB)

c) Execution time (seconds)

- Cubic memory usage due to automatic population
- Can be reduced if operators choose not to generate all flow entries
- Execution time of ns-3 omitted
    - Is at hour level

3. Lines of code

Same functionality implemented in NS4 and ns-3

Features	ns-3	NS4	Percent smaller
L2 / L3 Switch	598	165	72.408%
with ACL	803	252	68.617%
with ACL & NAT	1038	494	52.408%
with ACL & NAT &SC & SG	1219	637	47.744%

ACL: Access control list
NAT: Network Address translation
SG: Source Guard
SC: Storm Control

Conclusion

- Programmable data plane with multiple devices can now be simulated
- Simulation setup much easier compared to ns-3
- Direct migration of simulated behavior to real-world devices possible
- Less error prone code writing
- Performance improved significantly

Thank you for your attention

Thank you for your attention

Any questions?

Backup-Slides: History of NS4

Backup-Slides: History of NS4

Open source, event-driven network simulators

- ns-1
  - evolved in 1989 from REAL Network Simulator
  - terminal based
  - C++ and Tcl
  - Support for asic TCP, routing and scheduling algorithms
- ns-2
  - OTcl instead of Tcl
  - Support for sensor-networks, UDP, Multicast, wireless, satellites, ...
  - Implementation now object oriented
- ns-3
  - Python besides OTcl and adherence to C++ style patterns
  - Improved incorporation with other open-source network simulators
  - Support for multiple interfaces per node, sockets, customization of statistics output without restructuring of the simulation core

Related Work

Predecessors of NS4

- ns-1 (1989)
- ns-2 (1995)
- ns-3 (2011)

Other network simulators (excerpt)

- OPNet
  - GUI, but no P4 support
- PFPSim
  - Supports only one P4 enabled device

Simulation and emulation

Comparison / Differences

- Purpose
- Interactivity
- Reproducibility
- Runtime speed
- Precision