SDN based Deterministic Time-Sensitive Networking

Master's Thesis Project-1 || Mid Semester 2024

Presented By:

Pallav Kumar(2023CSM1010)

Pursuing M.Tech. in Computer Science and Engineering

Supervisor:

Dr. Abhinandan S P

Overview

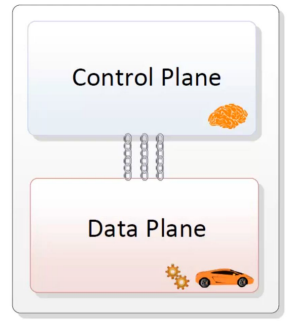
- Introduction
- Software Defined Networking (SDN)
  - What is SDN?
  - Layers Of SDN
  - OpenFlow
  - NFV
- Literature Review
  - Motivation
  - Methodology
  - Bounds on E2E latency and Jitter
  - Implementation and Testing
- Future work

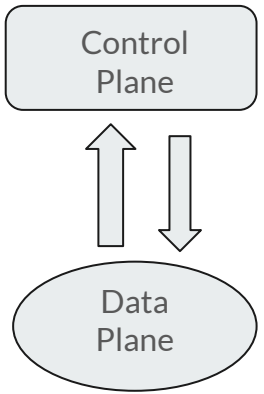
Introduction

- Deterministic Networking
  - Provide guaranteed performance.
  - Guaranteed Quality of Service(QoS).
  - Reduced latency and Jitter.
- Commonly used in modern applications
  - Industrial Automation
  - Autonomous Vehicles
  - Telemedicine

What is SDN?

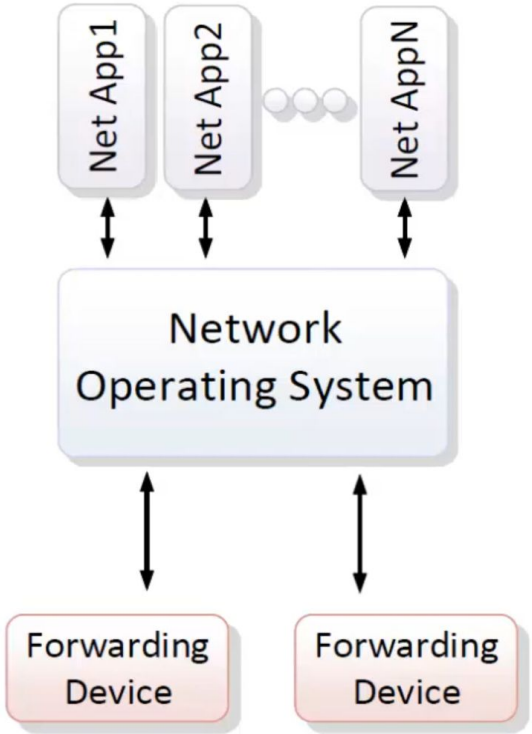
- In traditional network, the control and the data plane are tightly integrated into same device.
- Scaling a traditional network requires adding more devices and manually configuring each one, which can be time-consuming and error-prone.
- In the Software Defined Networking (SDN) architecture, the control and data planes are decoupled, network intelligence and state are logically centralized, and the underlying network infrastructure is abstracted from the applications.*
- In other words in the SDN paradigm, not all the processing happens inside the same device.





* Software-Defined Networking: The New Norm for Networks ONF White Paper, April 13, 2012

Layers of SDN



Network Applications

Application Interfaces
- Java API
- Northbound (e.g. RESTConf)

SDN Controller/Control Plane
- Topology Service
- Inventory Service
- Statistics Service
- Host Tracking

SouthBound Interface
- OpenFlow
- OVSDB
- NETCONF
- SNMP

Forwarding Devices/
Data Plane

Image Source: https://www.youtube.com/watch?v=DiChnu_PAzA&t=516s

OpenFlow

- OpenFlow is a communication protocol which enables the network controller to interact with the data plane devices such as switches, routers etc.
- This gives control plane a way to control the behaviour of switches in the network dynamically and programmatically.
- OpenFlow facilitates centralized network management and automation, enhancing network flexibility and scalability.
- OpenFlow allows dynamic, real-time updates to network policies.
- OpenFlow does not equal SDN. It is a flavor, or a subset of SDN.

OpenFlow (Contd.)

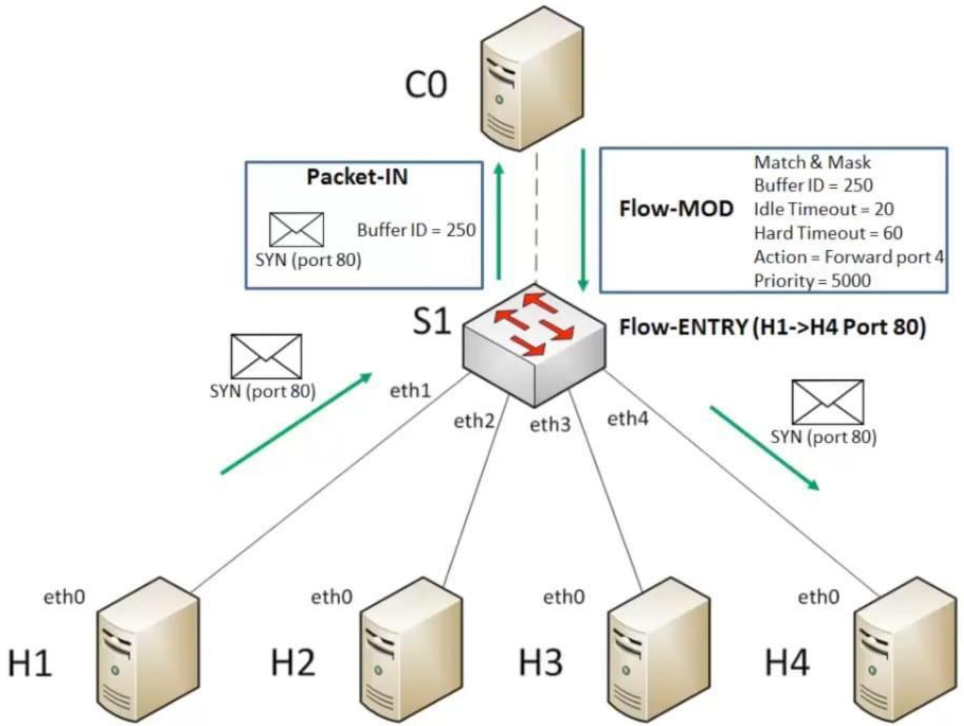


Image Source: https://www.youtube.com/watch?v=I25Ukkmk6Sk&t=2s

Network Function Virtualization (NFV)

- Network Function Virtualization (NFV) is replacement of hardware networking components with virtual (software) networking functions.
- These virtual networking functions can be run on different virtual machines.
- The virtual machines require hypervisor to run networking software and processes such as routing and load balancing.
- A hypervisor or software-defined networking controller allows network engineers to program all of the different segments of the virtual network, and even automate the provisioning of the network.

NFV Architecture

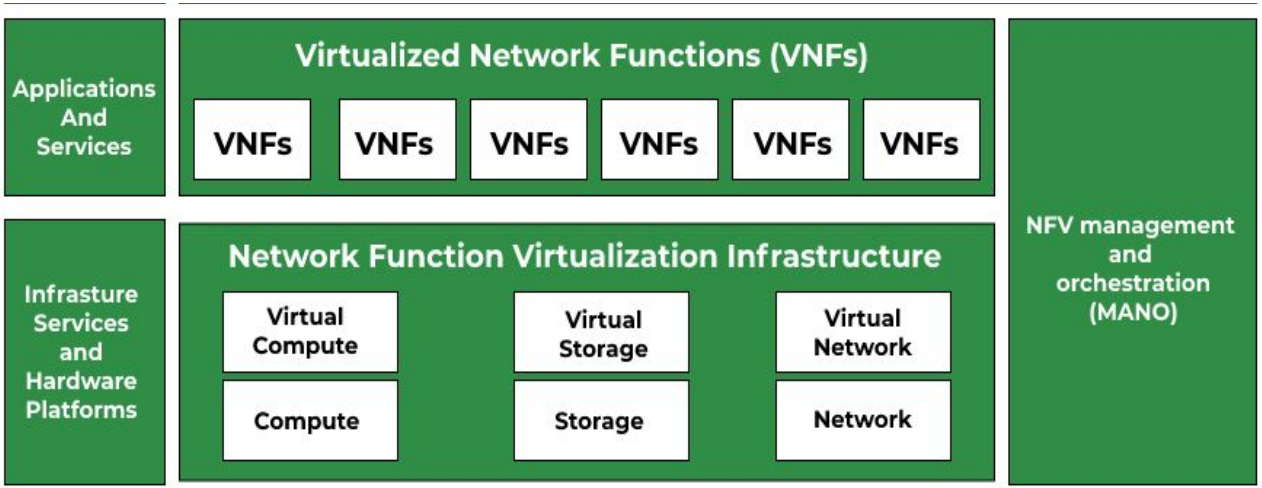


Image source:https://www.geeksforgeeks.org/network-functions-virtualization/

NFV vs. SDN

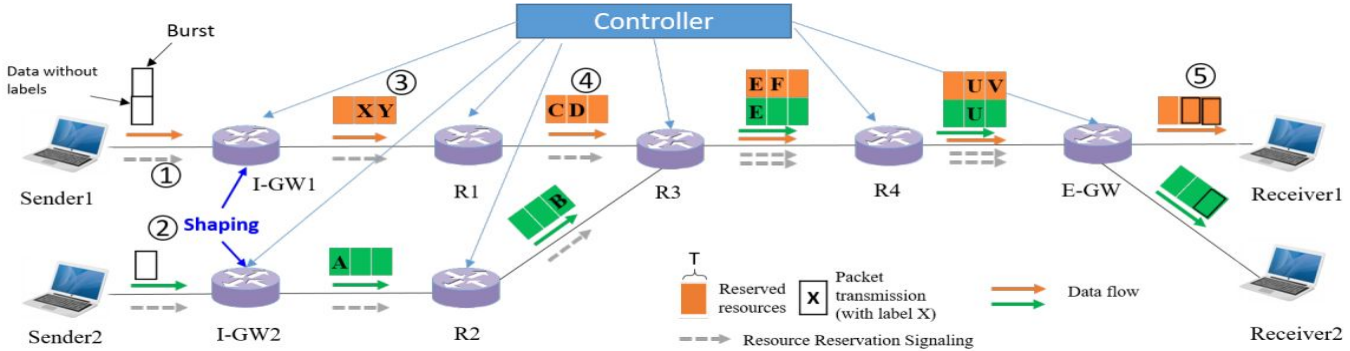
- NFV and SDN are not dependent on each other, but they do have similarities.
- Both rely on virtualization and use network abstraction, but how they separate functions and abstract resources is different.
- While NFV separates networking services from dedicated hardware appliances, software-defined networking, or SDN, separates the network control functions such as routing, policy definition and applications from network forwarding functions.
- A software-defined network can sit on top of either a virtual network or a physical network, but a virtual network does not require SDN to operate.

Literature Review

Title: Towards Large-Scale Deterministic IP Networks[1]

Author: Bingyang Liu, Shoushou Ren, Chuang Wang, Vincent Angilella, Paolo Medagliani, Sebastien Martin, Jeremie Leguay

Contribution: Presented a realistic implementation of a Large-Scale Deterministic Network (LDNs) to provide E2E latency and bounded jitter in large-scale IP networks.



Literature Review (Contd.)

Motivation

- Traditional IP networks work on a best-effort delivery model, which does not guarantee when data will arrive.
  - Latency upper bound is nearly 980µs.
  - Latency lower bound is 19µs.
  - Resulting jitter of 961µs.
- Scaling deterministic networks to a large scale.
- Integration with existing IP infrastructure.

Methodology

- The LDN architecture, designed to bring deterministic performance to large scale.
- It is scalable unlike TSN.
- By carefully controlling how and when data is transmitted through the network, LDN ensures low and consistent latency, as well as minimal jitter—key factors for real-time communication.
- The three core components are **fine-grained traffic shaping**, **asynchronous cyclic queuing** and **label-based forwarding**, which together make this solution feasible and effective.

Fine-Grained Traffic Shaping at Ingress Nodes

- Traffic bursts management.
- Gate-Controlled queues manage how traffic is released into the network.
- Granularity of traffic shaping.

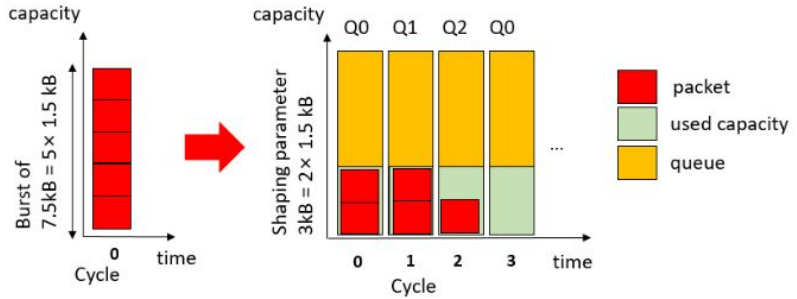


Fig. 2. Example of shaping of a burst of size $b_f = 7.5$ kB with a shaping parameter $b'_f = 3$ kB/cycle, 3 kB used on each cycle and shaping delay $3T$.

Asynchronous Cyclic Queuing and Forwarding

- LDN divides time into fixed cycles, and each cycle is responsible for processing traffic, ensuring that packets are scheduled and transmitted predictably.
- Unlike other methods, LDN operates asynchronously.
- Three FIFO queues per node, used in a round-robin fashion.
- This round-robin method ensures that all packets are sent in an orderly, predictable manner.

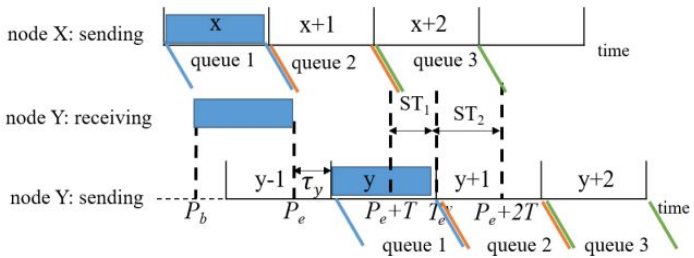


Fig. 3. Cyclic queue scheduling on core LDN nodes.

Label-Based Packet Forwarding

- Each packet is tagged with a label that indicates which cycle it should be transmitted in.
- The label simplifies the forwarding process because each node just needs to look at the label to decide when to send the packet.
- Each node in the network maintains a mapping table that tells it how to convert an incoming label into an outgoing label.
- This table is “quasi-static,” meaning it doesn’t change often, which reduces the complexity of network management while still allowing for flexibility.

Bounds on E2E latency and jitter

- The worst E2E queuing delay happens when a packet, sent at the first hop at T_b^1 , is forwarded, at the last hop, at T_e^h.
  - $$D_{\text{worst}} = T + \sum_{i=2}^{h} (\tau_i + T)$$
- In the best case, a packet, sent at the first hop at time Te1 ,is sent out at the last hop at time Tbh .
  - $$D_{\text{best}} = \sum_{i=2}^{h-1} (T + \tau_i) + \tau_h$$
- Jitter = $D_{worst} - D_{best} = 2T$
- $(h - 2)T \leq$ E2E queuing delay $\leq (2h - 1)T$.

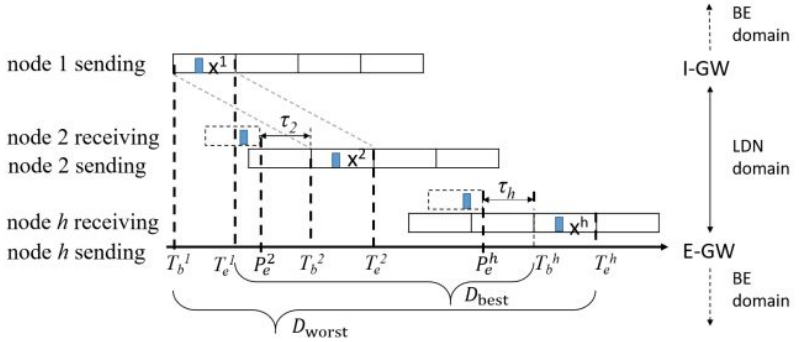


Fig. 4. End-to-end latency and jitter analysis.

Implementation and Testing

- PoC implementation on a Huawei router.
  - T = 10μs
  - Worst case E2E queuing delay is 67.046μs.
  - The best case delay is 49.886μs.
  - Largest jitter experienced 67.046 − 49.886 = 17.16μs
- Simulations conducted using OMNeT++, a network simulator, on a real-world ISP topology (AS 1239 of Sprint).
- Compared the performance of LDN with traditional OSPF routing.

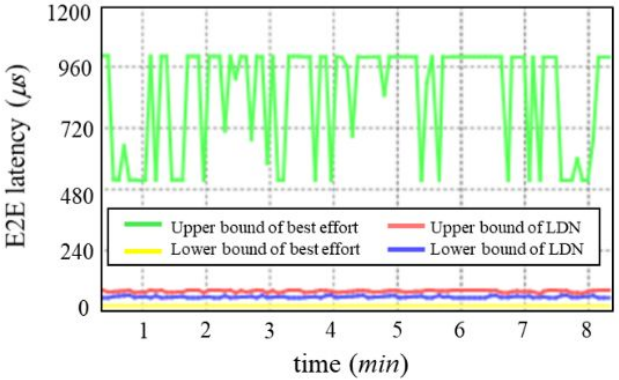


Fig. 6. Test results over the PoC implementation.

References

1. Liu, B., Ren, S., Wang, C., Angilella, V., Medagliani, P., Martin, S., & Leguay, J. (2021, June). Towards large-scale deterministic IP networks. In *2021 IFIP Networking Conference (IFIP Networking)* (pp. 1-9). IEEE.
2. https://www.youtube.com/watch?v=l25Ukkmk6Sk&t=2s
3. https://www.youtube.com/watch?v=DiChnu_PAzA&t=516s
4. https://youtube.com/playlist?list=PLpherdrLyny8YN4M24iRJBMCXkLcGbmhY&feature=shared&authuser=0
5. https://www.vmware.com/topics/network-functions-virtualization-nfv
6. https://www.redhat.com/en/topics/virtualization/what-is-nfv
7. https://www.geeksforgeeks.org/network-functions-virtualization/

Future Work

- Implement the paper “Towards Large-Scale Deterministic IP Networks”.
- In the implementation I will,
    - Create a mininet network with specified topology.
        - Currently working with RocketFuel topology
    - Randomly create a set of m Ingress and n Egress node.
    - Then try to build network as discussed in the paper.

Thank You