NeSTiNg - Network Simulator for Time-Sensitive Networking (TSN)

Tsung-Ying Kuo

National Taiwan University

Spring 2021

Outline

- **Installation & Setting Environment**
- Usage
- Running Simulation
- Analysis
- Limitation about Nesting

Installation & Setting Environment

- **OMNeT++ Installation**
  - Refer to omnetpp official document
  - https://doc.omnetpp.org/omnetpp/InstallGuide.pdf
- **NeSTiNg Installation & Setting Environment**
  - Refer to README of producer's gitlab
  - Command Line Installation
  - https://gitlab.com/ipvs/nesting

Outline

- Installation & Setting Environment
- **Usage**
  - **Overview**
  - NED Files
  - INI Files
  - XML Format Files
- Running Simulation
- Analysis
- Limitation about Nesting

Required Files

- NED File
  - ➢ Define network topology
    - Host, Switch, Link…
- INI File
  - ➢ Modify arguments of the hosts and switches defined in NED file
    - Address, Processing Delay, Buffer Capacity…
  - ➢ Set actions for hosts and switches
    - Gate Control, Data Flow of Host…
- XML Format File
  - ➢ Routing Database for switches
  - ➢ Scheduling setting for hosts and gates of switch

Outline

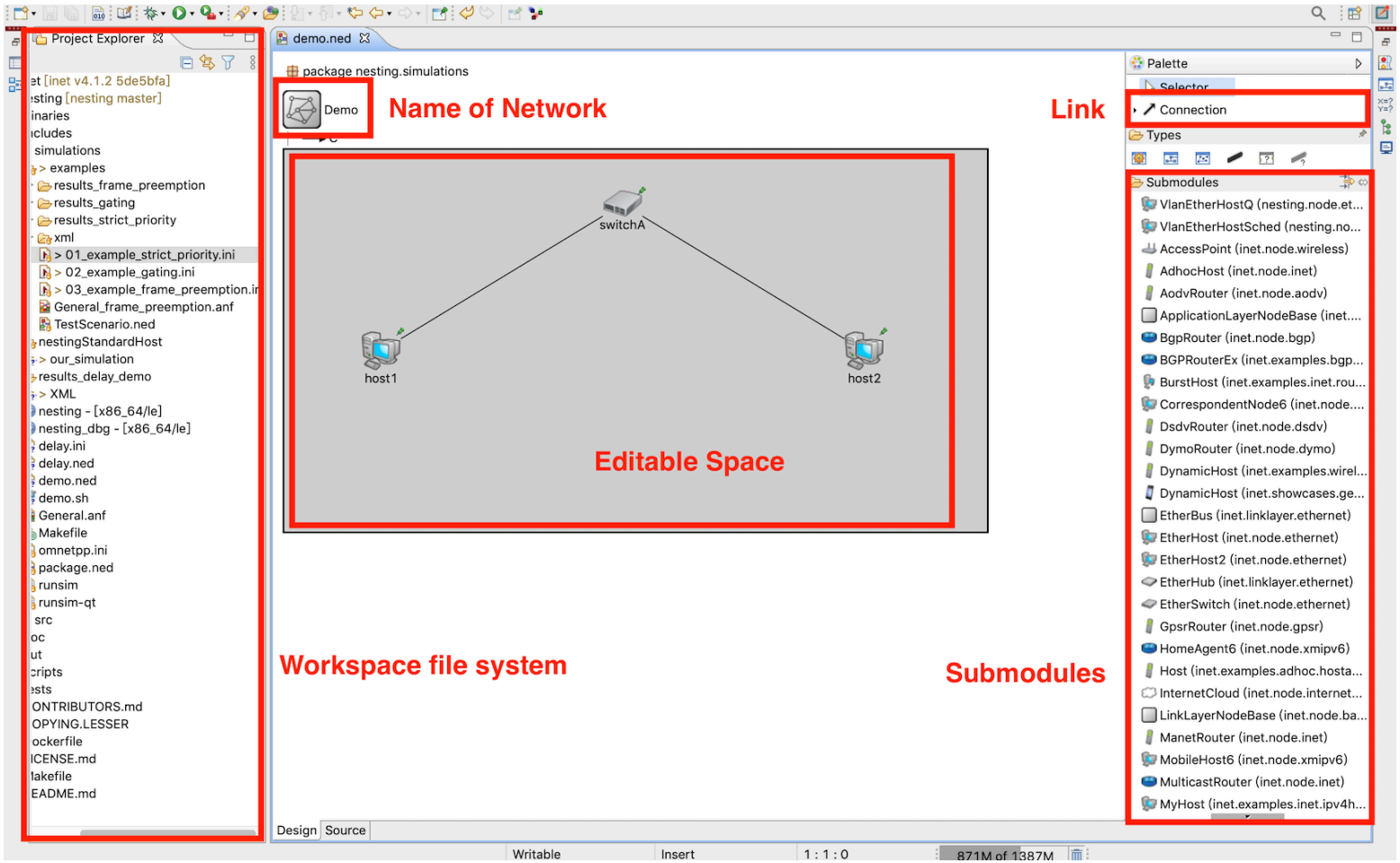
- Installation & Setting Environment
- **Usage**
  - Overview
  - **NED Files**
  - INI Files
  - XML Format Files
- Running Simulation
- Analysis
- Limitation about Nesting

GUI(1/2)

- Add a NED file
  - Right click the folder in workspace file tree
  - Select **NEW** → **Network Description File (NED)**
- GUI
  - Select Submodule or Connection from the right side of window
  - Click on the editable area to add a submodule
  - Click two submodules in editable area to add a Connection

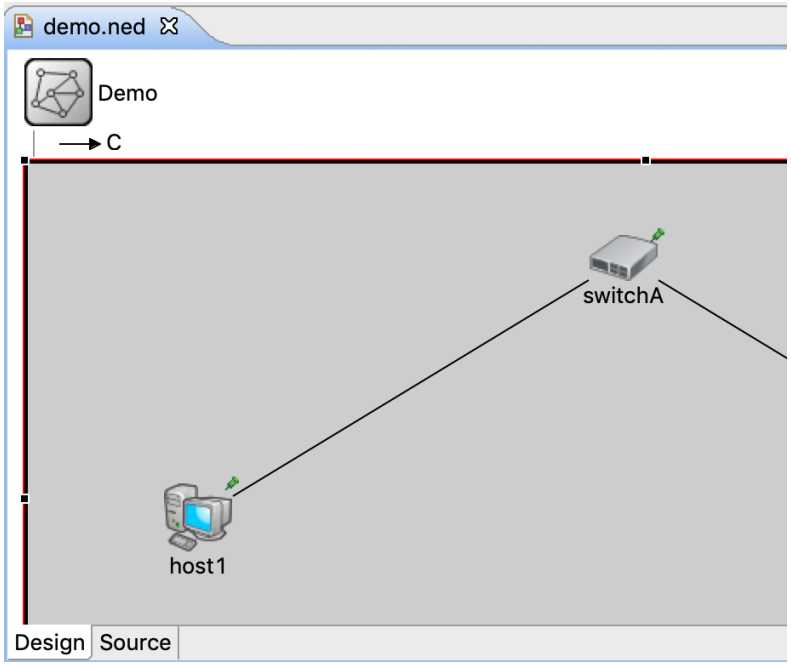
GUI(2/2)

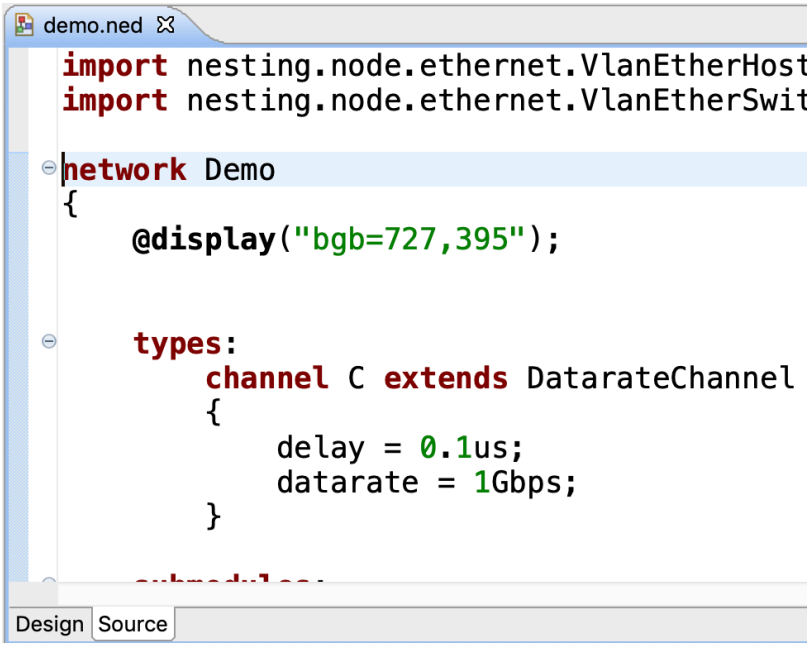
- GUI


Project Explorer
demo.ned
package nesting.simulations
Demo
Name of Network
Link
Palette
Connection
Types
Submodules
switchA
host1
host2
Editable Space
Workspace file system
Submodules
Design
Source
Writable
Insert
1 : 1 : 0

NED Language(1/5)

- NED Language
  - ➢ Switch from “Design” to “Source” at the bottom of editable area





NED Language(2/5)

- Submodules
  - ➢ Set the coordinate of submodule by **@display** parameter
  - ➢ Specify **@ethg[n]** to make the number of gates of switches equal to n

```
submodules:
    host1: VlanEtherHostQ {
        @display("p=100,208");
    }
    host2: VlanEtherHostSched {
        @display("p=600,208");
    }
    switchA: VlanEtherSwitchPreemptable {
        parameters:
            @display("p=350,55");
        gates:
            ethg[2];
    }
```

NED Language(3/5)

- Channel Type
  - Define your own channel type
  - Use the key word **extends** to use predefined channel with different parameters
    - e.g. DatarateChannel
    - Specify link delay and data rate for the channel

```
types:
    channel C extends DatarateChannel
    {
        delay = 0.1us;
        datarate = 1Gbps;
    }
```

NED Language(4/5)

- Connection
  - Connect two device by following syntax
    - **device1.ethg <—> [Channel Name] <—> device2.ethg**
    - e.g. connect ethg[0] and ethg[1] of switchA to host1 and host2 respectively

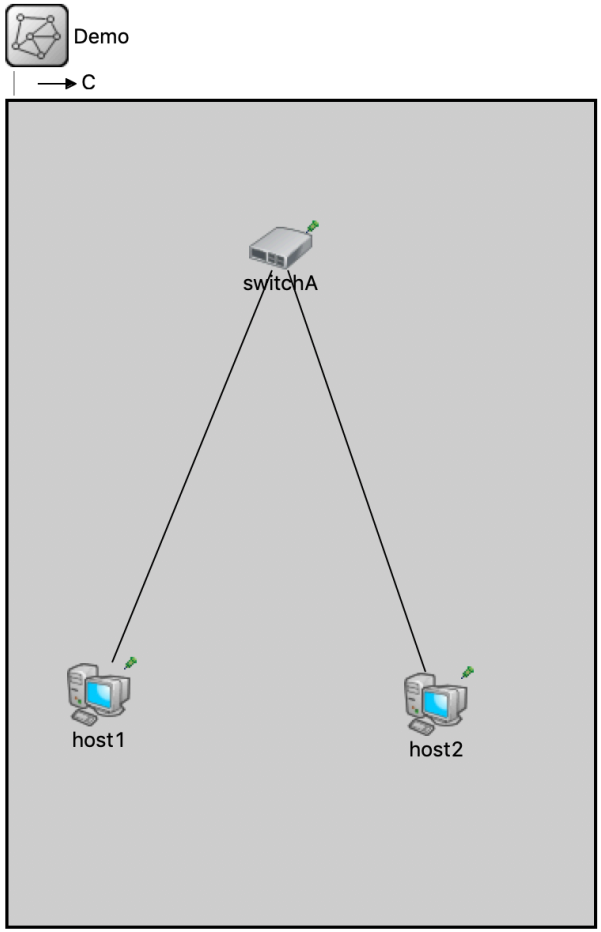
```
connections:
    switchA.ethg[1] <--> C <--> host2.ethg;
    switchA.ethg[0] <--> C <--> host1.ethg;
```

NED Language(5/5)

- Sample Code

```
package nesting.simulations;
import ned.DatarateChannel;
import nesting.application.ethernet.VlanEtherTrafGenSched;
import nesting.node.ethernet.VlanEtherHostQ;
import nesting.node.ethernet.VlanEtherHostSched;
import nesting.node.ethernet.VlanEtherSwitchPreemptable;

network Demo
{
    @display("bgb=727,395");
    types:
        channel C extends DatarateChannel
        {
            delay = 0.1us;
            datarate = 1Gbps;
        }
    submodules:
        host1: VlanEtherHostQ {
            @display("p=100,208");
        }
        host2: VlanEtherHostSched {
            @display("p=600,208");
        }
        switchA: VlanEtherSwitchPreemptable {
            parameters:
                @display("p=350,55");
            gates:
                ethg[2];
        }
    connections:
        switchA.ethg[1] <--> C <--> host2.ethg;
        switchA.ethg[0] <--> C <--> host1.ethg;
}
```



Commonly Used Submodules

- vlanEtherHostQ
  - Simple Host with the capability to create and receive vlan tagged packet
- vlanEtherHostSched
  - Simple Host with the capability to send out frames according to a given schedule
  - Being able to send multiple data flow
- vlanEtherSwitchPreemptable
  - A switch that support frame preemption

Outline

- Installation & Setting Environment
- **Usage**
  - Overview
  - NED Files
  - **INI Files**
  - XML Format Files
- Running Simulation
- Analysis
- Limitation about Nesting

Basic

- Add a INI file
  - Right click the folder in workspace file tree
  - Select **NEW** → **Initialization File (ini)**
- Several basic arguments
  - **network**
    - Specify the name of network topology
  - **sim-time-limit**
    - The time limit of simulation time
  - **result-dir**
    - The directory where the result of analysis been saved

```
[General]
network = Delay

result-dir = results_delay_demo
sim-time-limit = 240us
```

MAC address

- MAC address
  - Set the Mac address of hosts
    - A hex string of 12 digits

```
# MAC Addresses of hosts
**.host1.eth.address = "00-00-00-00-00-01"
**.host2.eth.address = "00-00-00-00-00-02"
```

Including XML(1/2)

- What can XML file do?
  - ➢ Set the filtering database of switch
    - The forwarding policy inside the switch
    - Ex. Forward packets with destination MAC “00-00-00-00-00-02” to ethg[2]
  - ➢ Set the gate control of switch ports
    - Ex. With a cycle period of 400 us, open only 7th gate in the first 200us
  - ➢ Create flow for vlanEtherHostSched
    - Ex. Send a packet with 1200 bytes to host2 every 100us
- Syntax
  - ➢ xmldoc(xml_file_path, entry_name_in_xml_file)

Including XML(2/2)

- Filtering Database

```
**.filteringDatabase.database = xmldoc("xml/rout.xml", "/filteringDatabases/")
```

- Gate Controller

```
**.switchA.eth[8].queue.gateController.initialSchedule = \
xmldoc("xml/sched.xml", "/schedules/switch[@name='switchA']/port[@id='8']/schedule")
```

- Flow Creation

```
**.host1.trafGenSchedApp.initialSchedule = xmldoc("xml/sched.xml")
```

Flow Creation(1/2)

- Create flow for vlanEtherHostQ
  - Destination Address
    - MAC address of destination host
  - Packet Length
    - Ex. 100Bytes, 1200Bytes…
  - Interval
    - The time nterval between sending two packets
    - Ex. 0.1s, 100us…
  - PCP (Priority Code Point)
    - From 0 to 7
    - Pass through the gates of the same number in switch port
  - Start Time
    - The time of the first packet been sent

Flow Creation(2/2)

- For example
  - Arguments
    - Destination Address : 00-00-00-00-00-02
    - Packet Length : 100 bytes
    - Interval : every 12 us
    - PCP : 7
    - Start time : 0us

```
# host1 schedule
**.host1.trafGenApp.destAddress = "00-00-00-00-00-02"
**.host1.trafGenApp.packetLength = 100Byte
**.host1.trafGenApp.sendInterval = 12us
**.host1.trafGenApp.pcp = 0
**.host1.trafGenApp.startTime = 0us
```

Traffic Shaping(1/2)

- Time Aware Shaping
  - Reference : [wiki-TSN-TAS]
  - Make the maximum delay predictable for high-priority packets
  - Implemented by gate control in XML file (Which will be mentioned later)

```
**.switch*.eth[*].queue.tsAlgorithms[0].typename = "StrictPriority"
**.switch*.eth[*].queue.tsAlgorithms[1].typename = "StrictPriority"
**.switch*.eth[*].queue.tsAlgorithms[2].typename = "StrictPriority"
**.switch*.eth[*].queue.tsAlgorithms[3].typename = "StrictPriority"
**.switch*.eth[*].queue.tsAlgorithms[4].typename = "StrictPriority"
**.switch*.eth[*].queue.tsAlgorithms[5].typename = "StrictPriority"
**.switch*.eth[*].queue.tsAlgorithms[6].typename = "StrictPriority"
**.switch*.eth[*].queue.tsAlgorithms[7].typename = "StrictPriority"
```

Traffic Shaping(2/2)

- Credit Base Shapingn
  - Reference : [wiki-TSN-CBS]
  - Provide fair scheduling for low-priority packets and smooth out the traffic to eliminate congestion
  - The idle slope should be specified

```
**.switch*.eth[*].queue.tsAlgorithms[0].typename = "CreditBasedShaper"
**.switch*.eth[*].queue.tsAlgorithms[1].typename = "CreditBasedShaper"
**.switch*.eth[*].queue.tsAlgorithms[2].typename = "CreditBasedShaper"
**.switch*.eth[*].queue.tsAlgorithms[3].typename = "CreditBasedShaper"
**.switch*.eth[*].queue.tsAlgorithms[4].typename = "CreditBasedShaper"
**.switch*.eth[*].queue.tsAlgorithms[5].typename = "CreditBasedShaper"
**.switch*.eth[*].queue.tsAlgorithms[6].typename = "CreditBasedShaper"
**.switch*.eth[*].queue.tsAlgorithms[7].typename = "CreditBasedShaper"
```

```
**.switchA.eth[*].queue.tsAlgorithms[0].idleSlopeFactor = 0.05
```

Frame Preemption(1/2)

- Frame Preemption
  - Reference : [wiki : TSN- frame-preemption]
  - Two types of MAC frame for switch egress port
    - Preemptable MAC (pMAC)
    - Express MAC (eMAC)
  - Express frames can interrupt transmission of preemptable frames

Frame Preemption(2/2)

- Example
  - ➢ The configuration to enable frame preemption of port 3 of switchA
    - Frames who will go through queues[7] are express frames
    - Frames in queue[0~6] are preemptable frames

```
**.switchA.eth[3].queue.queues[0].expressQueue = true
**.switchA.eth[3].queue.queues[1].expressQueue = true
**.switchA.eth[3].queue.queues[2].expressQueue = true
**.switchA.eth[3].queue.queues[3].expressQueue = true
**.switchA.eth[3].queue.queues[4].expressQueue = true
**.switchA.eth[3].queue.queues[5].expressQueue = true
**.switchA.eth[3].queue.queues[6].expressQueue = true
**.switchA.eth[3].queue.queues[7].expressQueue = false

**.switchA.eth[3].mac.enablePreemptingFrames = true
```

Outline

- Installation & Setting Environment
- **Usage**
  - Overview
  - NED Files
  - INI Files
  - **XML Format Files**
- Running Simulation
- Analysis
- Limitation about Nesting

rout.xml

- Filtering Database of switches
  - Set the forwarding policy inside switch
  - For Example
    - Configuration for switch “switchA”
    - Address of “host1” : “00-00-00-00-00-01”
    - Address of “host2” : “00-00-00-00-00-02”
    - Forward packets to host1 to switchA’s port[0]
    - Forward packets to host2 to switchA’s port[1]

```
<filteringDatabases>
    <filteringDatabase id="switchA">
        <forward>
            <!-- Forward packets addressed to host1 to port[0] -->
            <individualAddress macAddress="00-00-00-00-00-01" port="0" />
            <!-- Forward packets addressed to host2 to port[1] -->
            <individualAddress macAddress="00-00-00-00-00-02" port="1" />
        </forward>
    </filteringDatabase>
</filteringDatabases>
```

sched.xml(1/2)

- Gate Control
  - Set cycle period
  - Use an 8 bits vector to control the opening and closing of gate
  - Example
    - Configuration of switch “switchA”
    - 400 us each cycle
    - Open only 8th gate in the first 200 us
    - Open 1st ~ 7th gates in the remaining 200 us

```
<switch name="switchA">
    <port id="3">
        <schedule cycleTime="400us">
            <entry>
                <length>200us</length>
                <bitvector>10000000</bitvector>
            </entry>
            <entry>
                <length>200us</length>
                <bitvector>01111111</bitvector>
            </entry>
        </schedule>
    </port>
</switch>
```

sched.xml(2/2)

- Create flow for vlanEtherHostSched
  - Similar to the flow creation of vlanEtherHostQ
  - For Example
    - Configuration for host “host1”
    - Destination MAC Address : “00-00-00-00-00-04”
    - Queue (PCP) : 7
    - Start at 10us, sending 354 Bytes every 400 us

```
<host name="host1">
    <cycle>400us</cycle>
    <entry>
        <start>10us</start>
        <queue>7</queue>
        <dest>00:00:00:00:00:04</dest>
        <size>354B</size>
        <flowId>1</flowId>
    </entry>
</host>
```

Outline

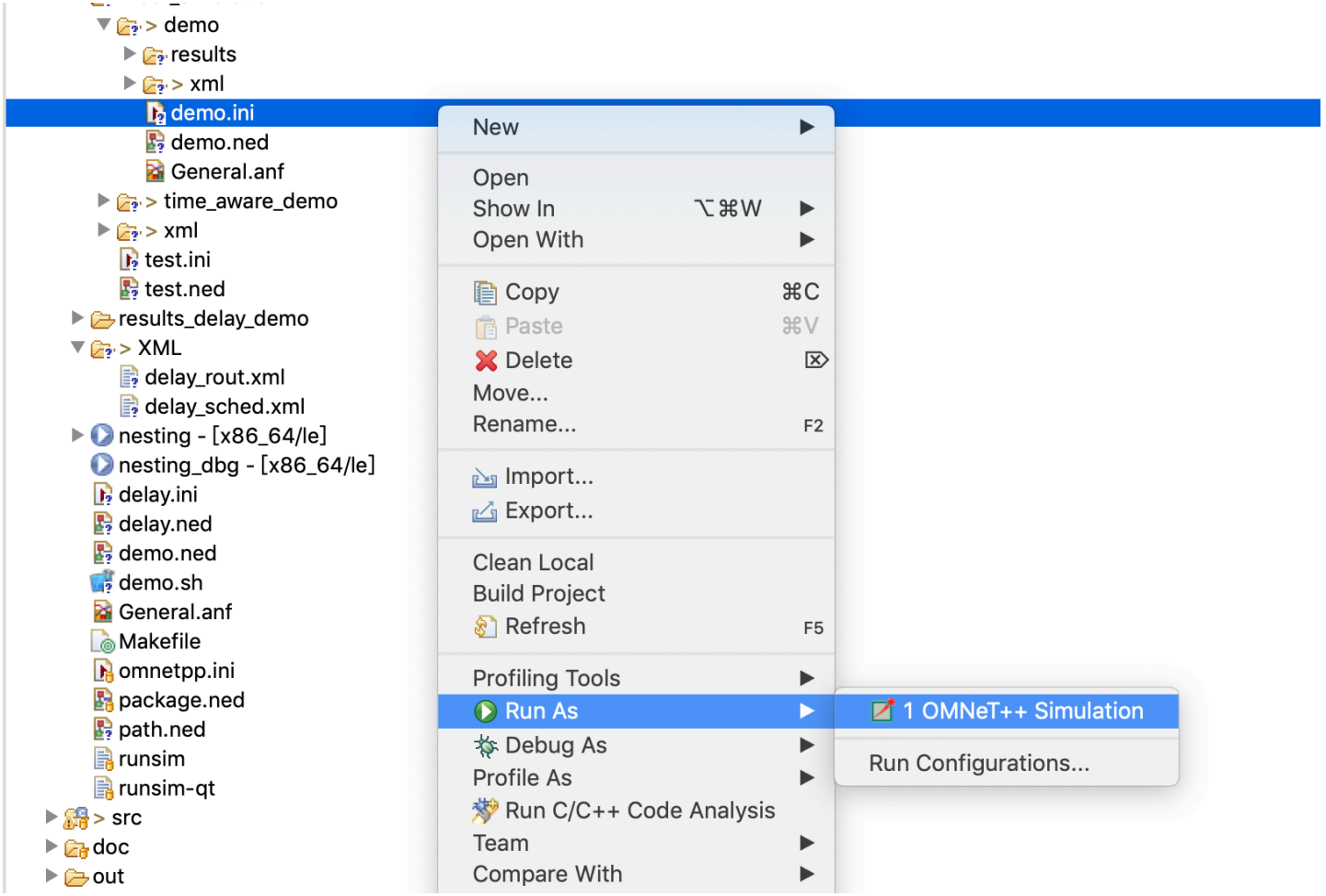
- Installation & Setting Environment
- Usage
- **Running Simulation**
- Analysis
- Limitation about Nesting

GUI Simulation(1/3)

- Procedure
  - Right click on ini file in workspace file system from the left side of window
  - Run As → OMNeT++ Simulation
  - A new window will appear if no error
    - Click RUN in the tool bar to run the simulation automatically step by step
    - Click STEP to show the next step of whole network
    - Click FAST to run faster
    - Click FASTEST to see the result immediately

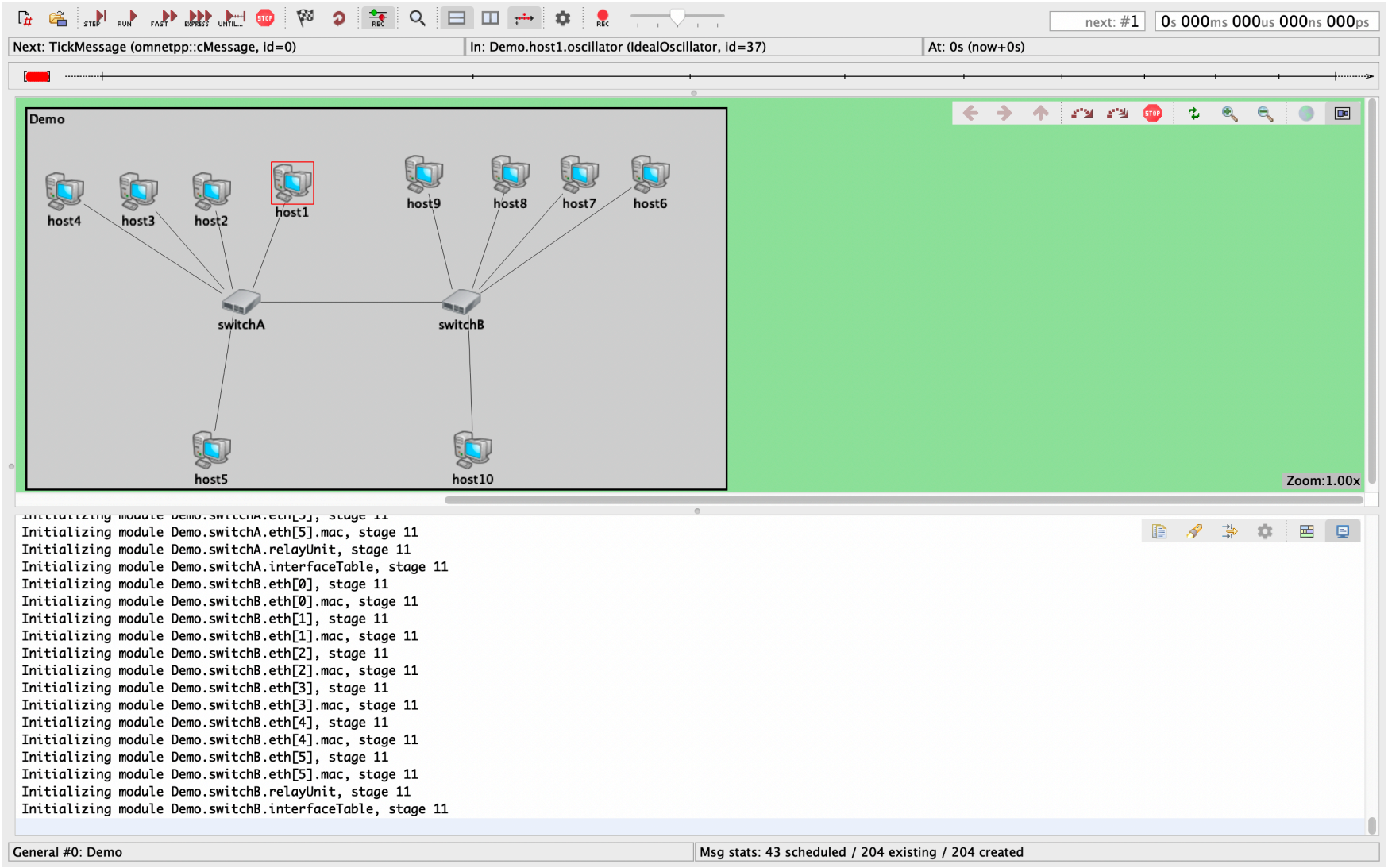
GUI Simulation(2/3)

- Running Simulation



GUI Simulation(3/3)

- Exemplary Simulation window



CLI Simulation

- Procedure
  - Place ini and ned file in directory : [workspace]/nesting/simulations/
  - Change working directory into [workspace]/nesting/simulations/
  - Use the command : ./runsim [ini_file_name] to run simulation
  - Exemplary Simulation

```
➜  simulations git:(master) ✗ ./runsim delay.ini
OMNeT++ Discrete Event Simulation  (C) 1992-2019 Andras Varga, OpenSim Ltd.
Version: 5.6.2, build: 200518-aa79d0918f, edition: Academic Public License -- NOT FOR COMMERCIAL USE
See the license for distribution terms and warranty disclaimer

Setting up Cmdenv...

Loading NED files from .:  10
Loading NED files from ../src:  37
Loading NED files from ../../inet/src:  726

Preparing for running configuration General, run #0...
Assigned runID=General-0-20210513-18:13:56-15661
Setting up network "Delay"...
Initializing...

Running simulation...
** Event #0   t=0   Elapsed: 4.8e-05s (0m 00s)  0% completed  (0% total)
     Speed:     ev/sec=0   simsec/sec=0   ev/simsec=0
     Messages:  created: 39   present: 39   in FES: 9
** Event #9633   t=0.00024   Elapsed: 0.019147s (0m 00s)  100% completed  (100% total)
     Speed:     ev/sec=504346   simsec/sec=0.0125654   ev/simsec=4.01375e+07
     Messages:  created: 1362   present: 263   in FES: 11

<!> Simulation time limit reached -- at t=0.00024s, event #9633

Calling finish() at end of Run #0...

End.
```

Result Files

- Lay out of result file
  - ➢ Recall the argument result-dir in ini file
  - ➢ After simulation, 3 result files will be generated and placed in result-dir
    - vec : vector output, e.g. a list of end to end delay of all packets
    - vci : the index file for vec file
    - sca : scalar output, e.g. total number of packets had been transferred
- Export results as csv format
  - ➢ Use the scavetool command
  - ➢ scavetool x [result_file_path] -o [output_csv_path]

Outline

- Installation & Setting Environment
- Usage
- Running Simulation
- **Analysis**
- Limitation about Nesting

Analysis

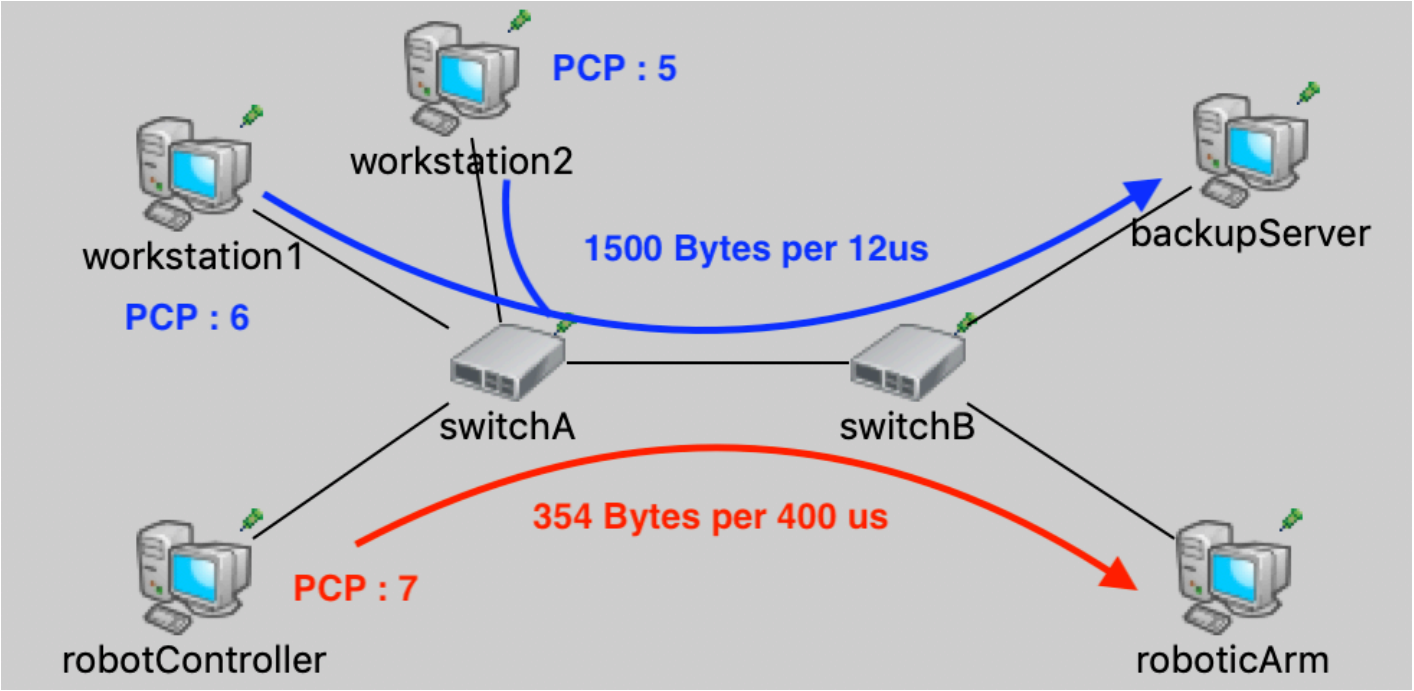
- Delay
  - Focus on the end-to-end delay of messages
  - Recorded in the entry “endToEndDelay:vector” of the output .vec file
- Visualize
  - Use omnet++ GUI
    - Open .vec file in omnet++
    - Right click on the entry “endToEndDelay:vector” and check “plot”
    - Or just double click the entry
  - Use python or any language that can make chart
    - Export .vec file as an csv file
    - Extract the fields of “vectime” and “vecvalye” of “endToEndDelay:vector”

Outline

- Installation & Setting Environment
- Usage
- Running Simulation
- **Analysis**
  - **Time Aware Shaper**
  - Credit Based Shaper
  - Partition and Delay
  - Routing Path and Delay
- Limitation about Nesting

Time Aware Shaper(1/5)

- Purpose
  - Try to compare the end to end delay in the following situations
    - There's no shaper and frame preemption in the network
    - Apply Time Aware Shaper
    - Frame Preemption
- Scenario (Reproduce the example in Nesting)



Time Aware Shaper(2/5)

- Scenario (TAS)
  - Setup gate control on both switches
  - Take 400us as the cycle, open the 7-th gate in the first 200us, close it in the last 200us
    - This create a exclusive time slot for the frame with PCP equals to 7

```
<schedule cycleTime="400us">
    <entry>
        <length>200us</length>
        <bitvector>01111111</bitvector>
    </entry>
    <entry>
        <length>200us</length>
        <bitvector>10000000</bitvector>
    </entry>
</schedule>
```

Time Aware Shaper(3/5)

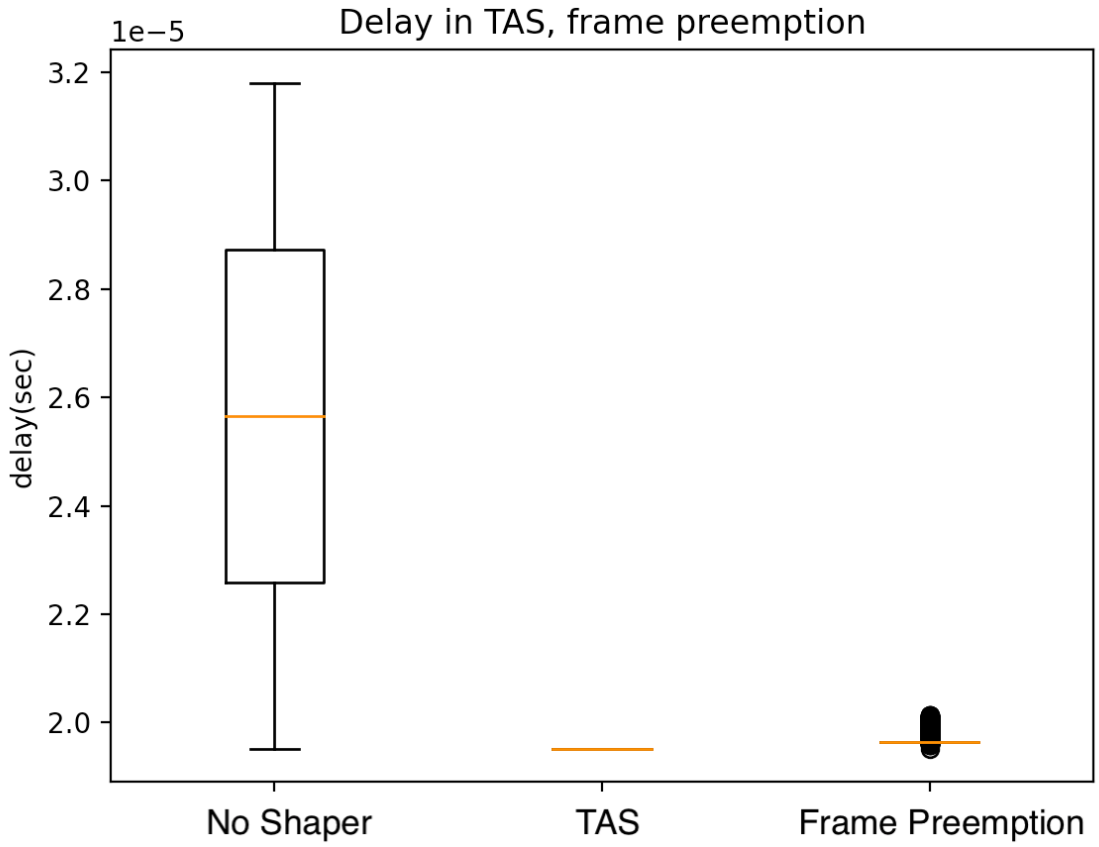
- Scenario (Frame Preemption)
  - Enable frame preemption on 3-th port in switchA (the port that connect to switchB)
    - Frames whose PCP equal to 7 are express frames
    - Frames with PCP 0~6 are preemptable frames

```
**.switch*.eth[*].queue.queues[0].expressQueue = false
**.switch*.eth[*].queue.queues[1].expressQueue = false
**.switch*.eth[*].queue.queues[2].expressQueue = false
**.switch*.eth[*].queue.queues[3].expressQueue = false
**.switch*.eth[*].queue.queues[4].expressQueue = false
**.switch*.eth[*].queue.queues[5].expressQueue = false
**.switch*.eth[*].queue.queues[6].expressQueue = false
**.switch*.eth[*].queue.queues[7].expressQueue = true

**.switchA.eth[3].mac.enablePreemptingFrames = true
```

Time Aware Shaper(4/5)

- Result
  - The delay keep low when using TAS to create exclusive time slot
  - With Frame Preemption, the delay suffers from jitter and a slightly increased end-to-end-delay caused by blocked links due to other frames in transmission.



Time Aware Shaper(5/5)

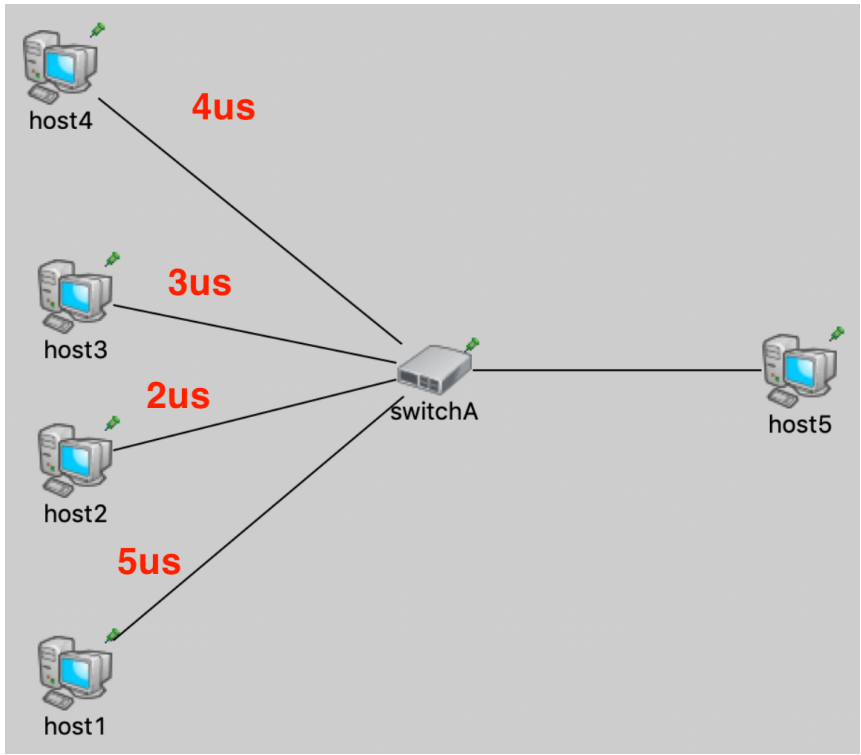
- Reference
  - NeSTiNg: Simulating IEEE Time-sensitive Networking (TSN) in OMNeT++, Jonathan Falk,∗ David Hellmanns,∗ Ben Carabelli,∗ Naresh Nayak,∗ Frank Dürr,∗ Stephan Kehrer,‡ Kurt Rothermel∗

Outline

- Installation & Setting Environment
- Usage
- Running Simulation
- **Analysis**
  - Time Aware Shaper
  - **Credit Based Shaper**
  - Partition and Delay
  - Routing Path and Delay
- Limitation about Nesting

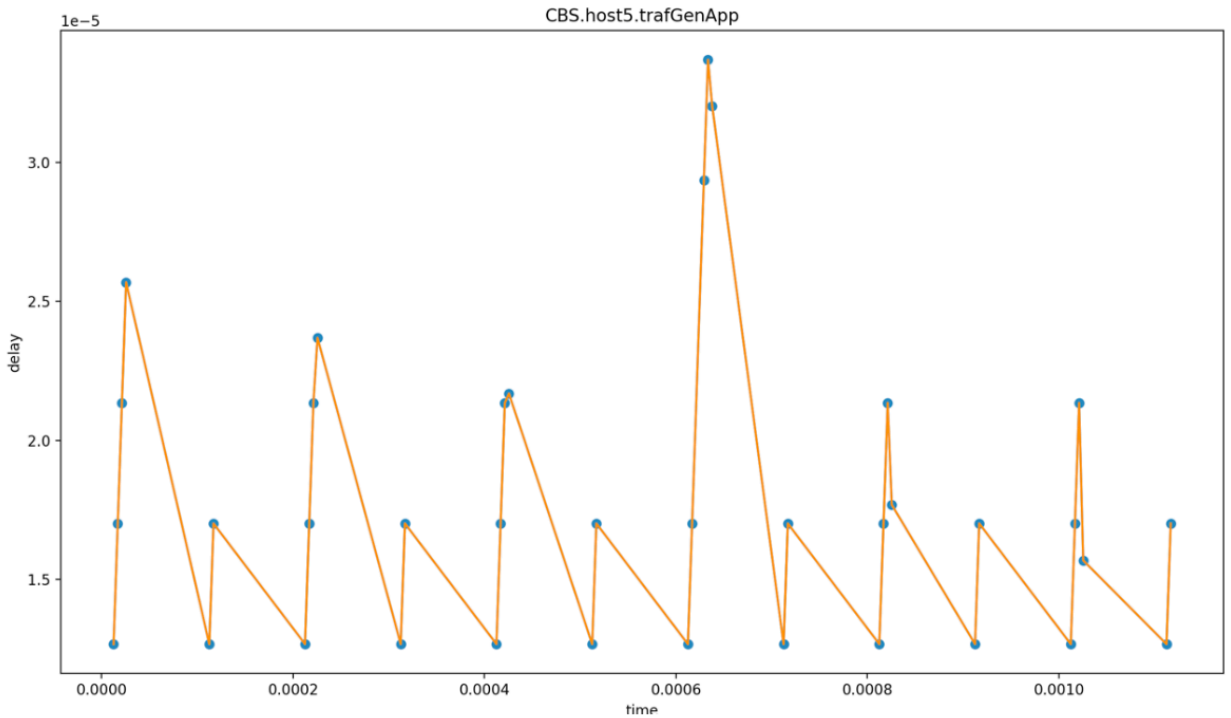
Credit Based Shaper(1/5)

- Purpose
  - Try to observe the change of end-to-end delay when setting different idleslope for CBS.
- Scenario
  - Host1~4 send 500 bytes to Host5 in different cycle time periods
  - Datarate is 1Gbps for every links
  - Credit Based Shaper in switchA



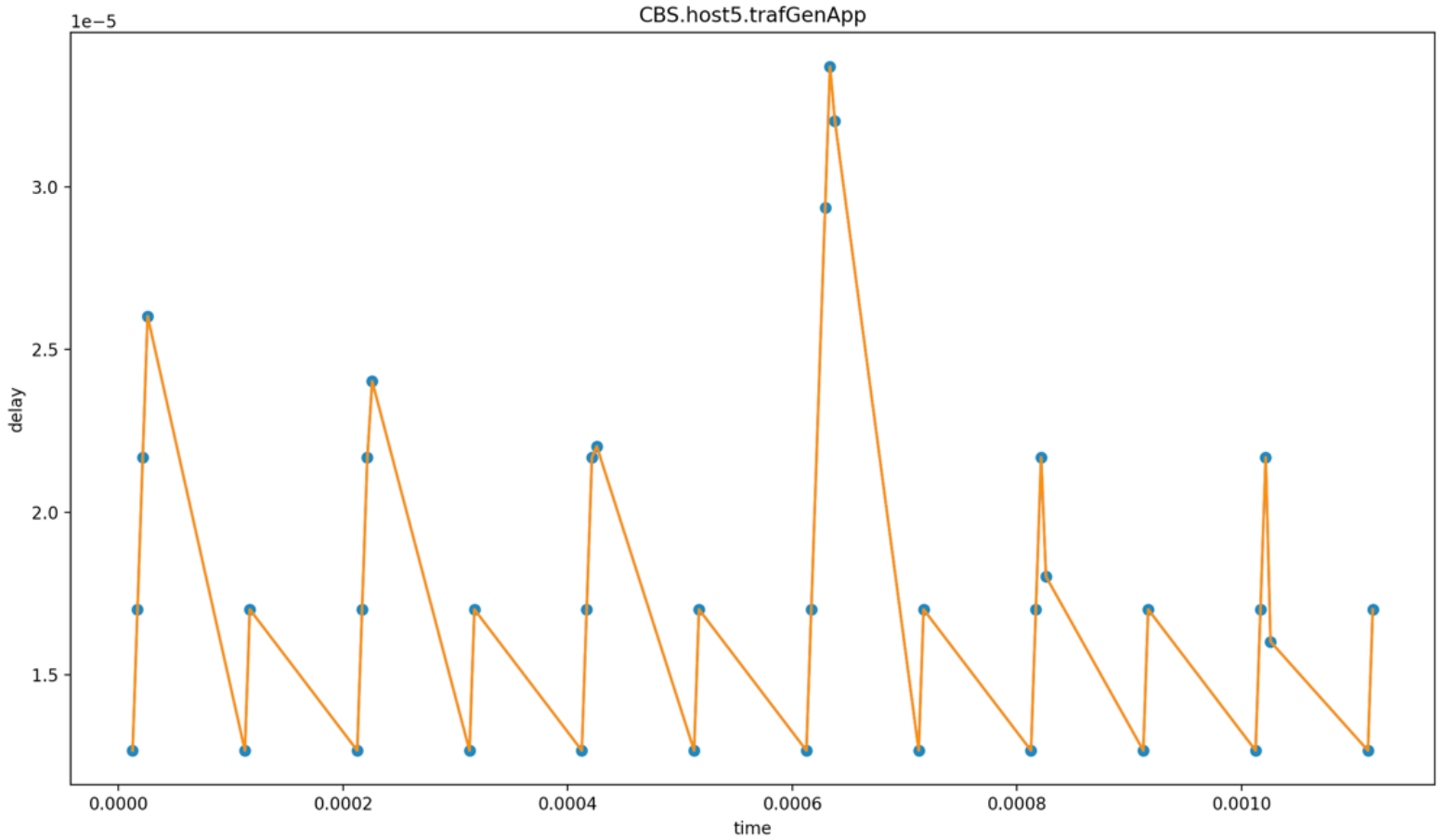
Credit Based Shaper(2/5)

- Result
  - ➢ Smaller idleslope will lead to longer interval for output port opening.
    - The smaller the idleslope, the smoother the delay growth curve
    - There's no serious congestion in this scenario, so an idleslope that is too low may lead to higher maximum end-to-end delay.
  - ➢ Below is the delay curve when there's no shaper



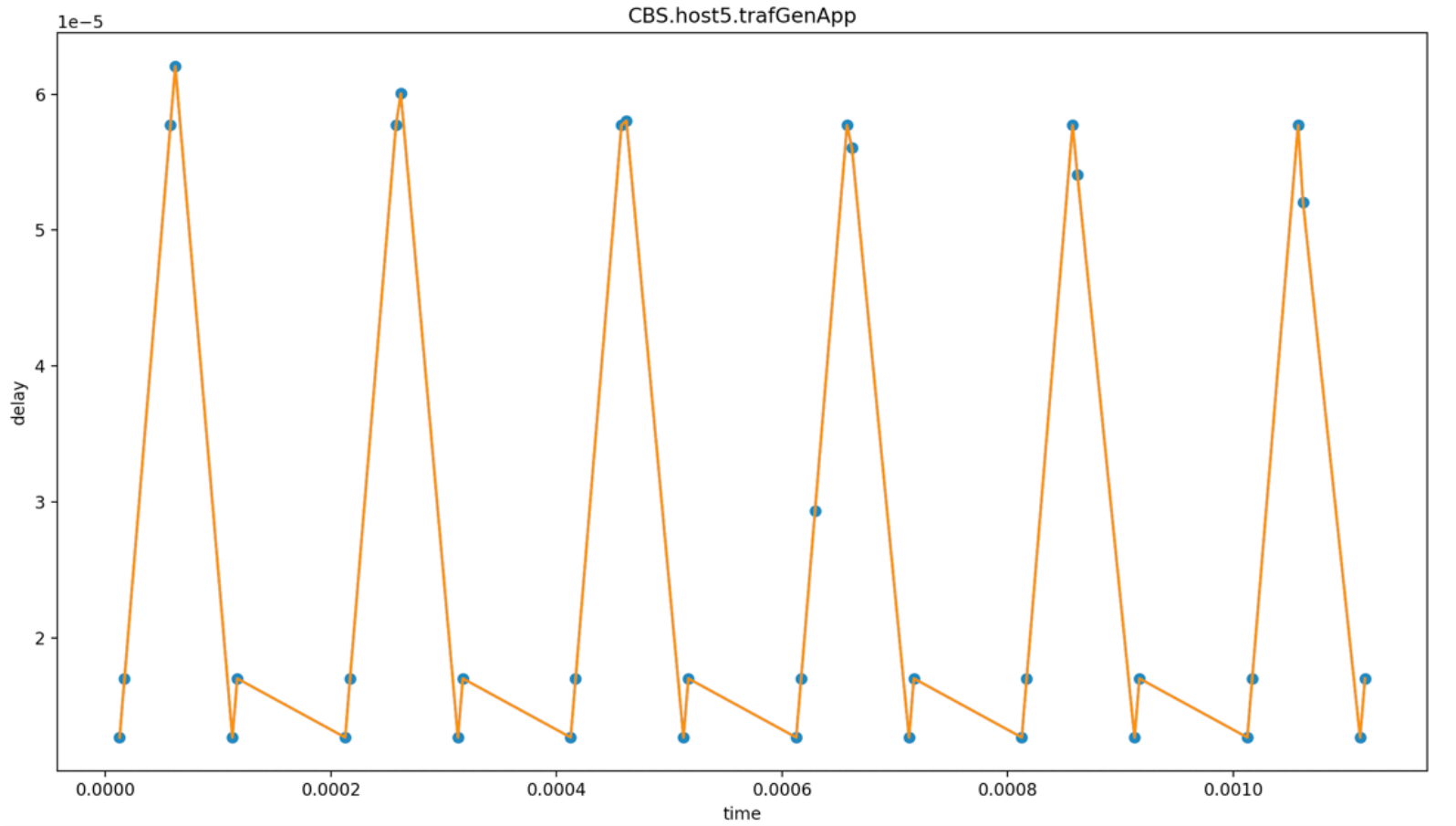
Credit Based Shaper(3/5)

- Result
  - Idleslope = 0.5



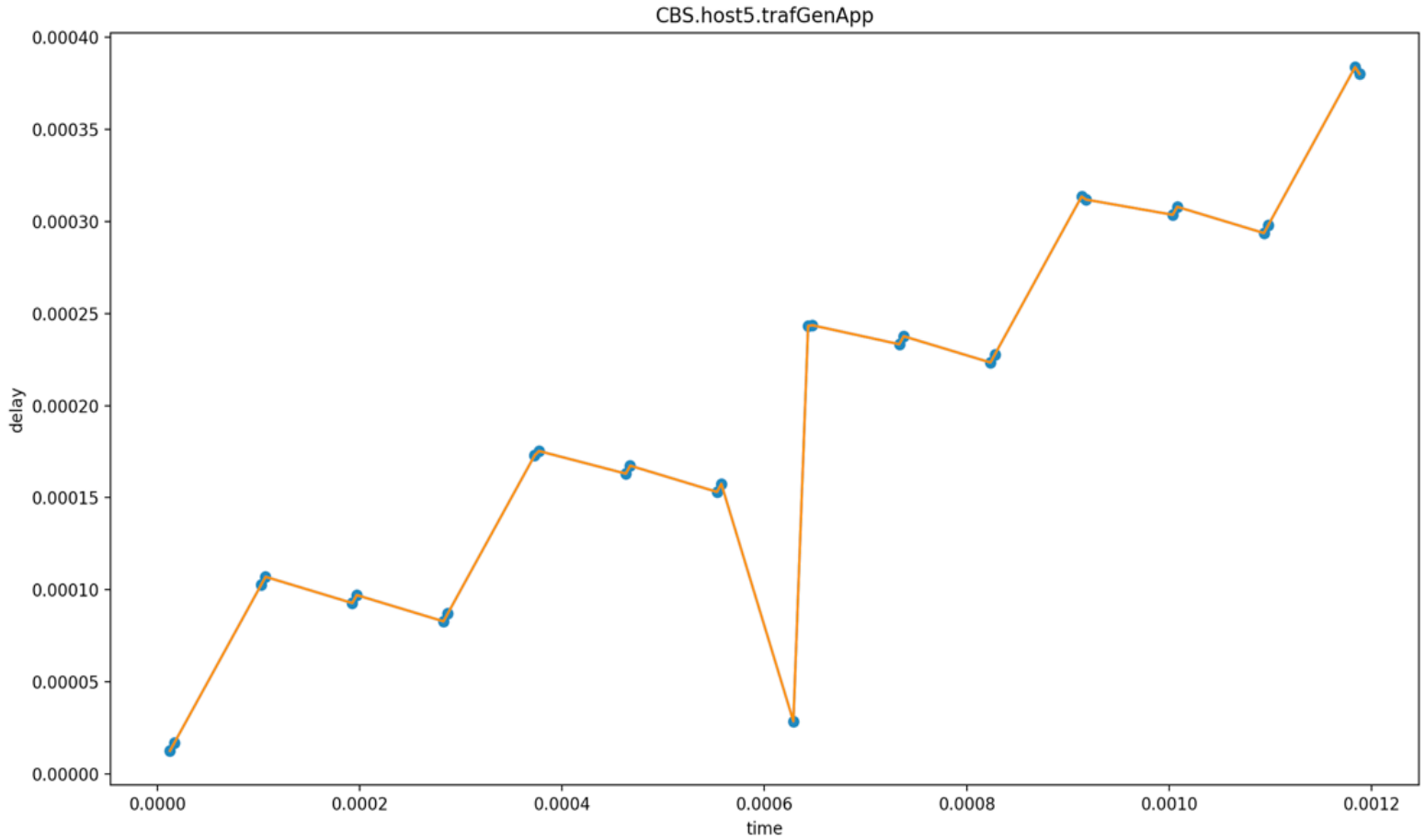
Credit Based Shaper(4/5)

- Result
  - Idleslope = 0.1



Credit Based Shaper(5/5)

- Result
  - Idleslope = 0.05

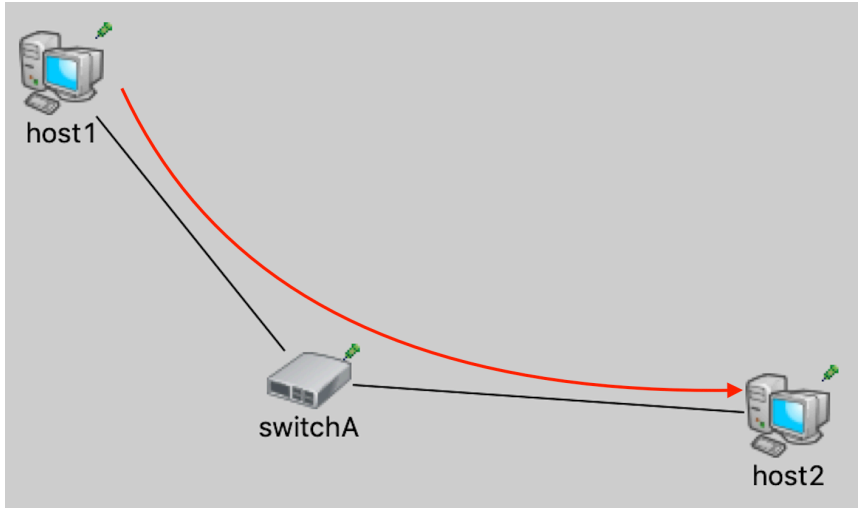


Outline

- Installation & Setting Environment
- Usage
- Running Simulation
- **Analysis**
  - Time Aware Shaper
  - Credit Based Shaper
  - **Partition and Delay**
  - Routing Path and Delay
- Limitation about Nesting

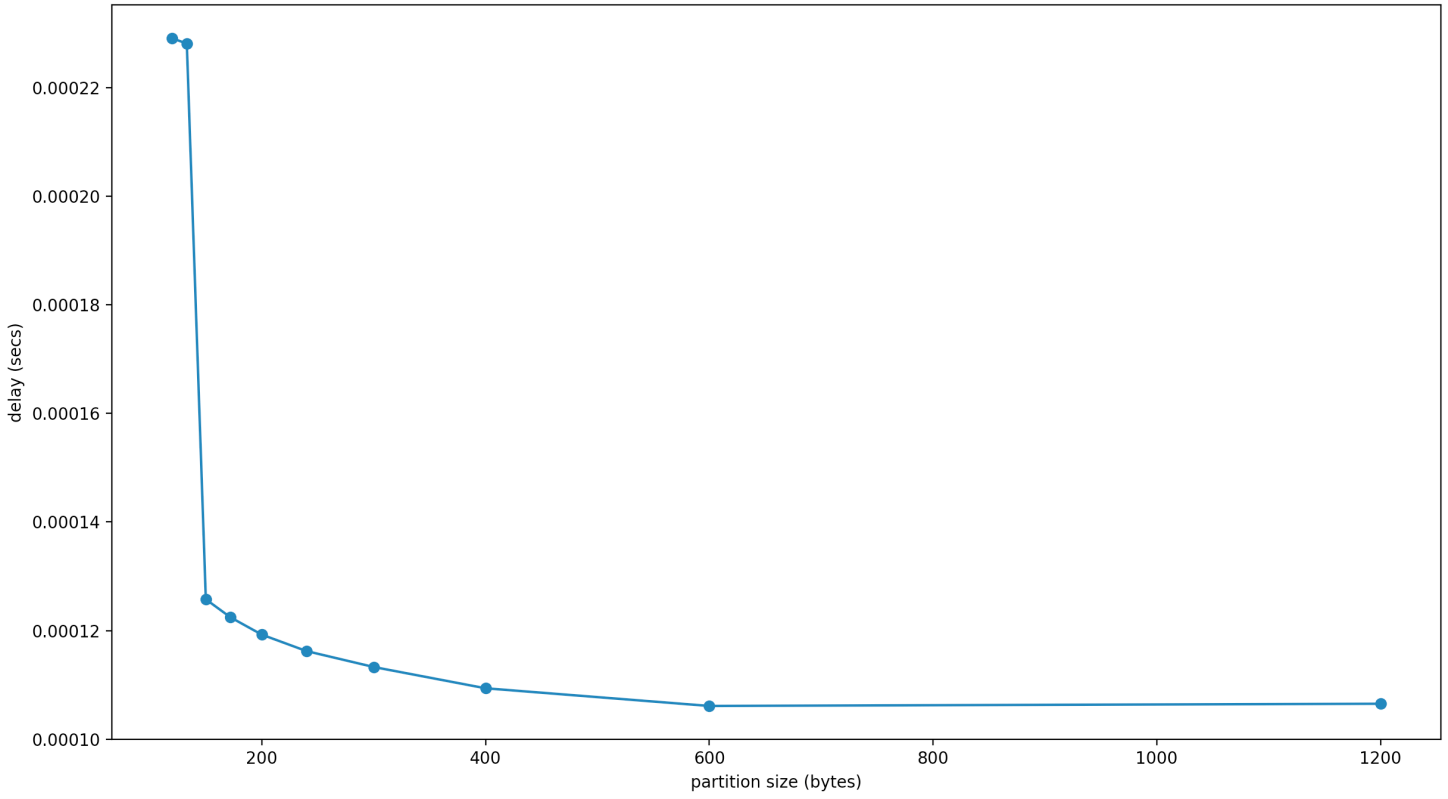
Partition and Delay(1/2)

- Definition
  - Transmit a fixed amount of data in fixed time, but the data is divided into different partition size.
  - Try to observe the relationship between the partition size and the maximum end to end delay.
- Scenario
  - Host1 sends 3600 bytes to Host2 in 18 us
  - The maximum packet length is 1500 bytes
  - Datarate is 1Gbps for both links



Partition and Delay(2/2)

- Result
  - As can be seen from following figure, the smaller the partition size, the higher the maximum delay.
  - This maybe because when the partition is smaller, the more packets have to be sent in the same time period, which create more congestion.

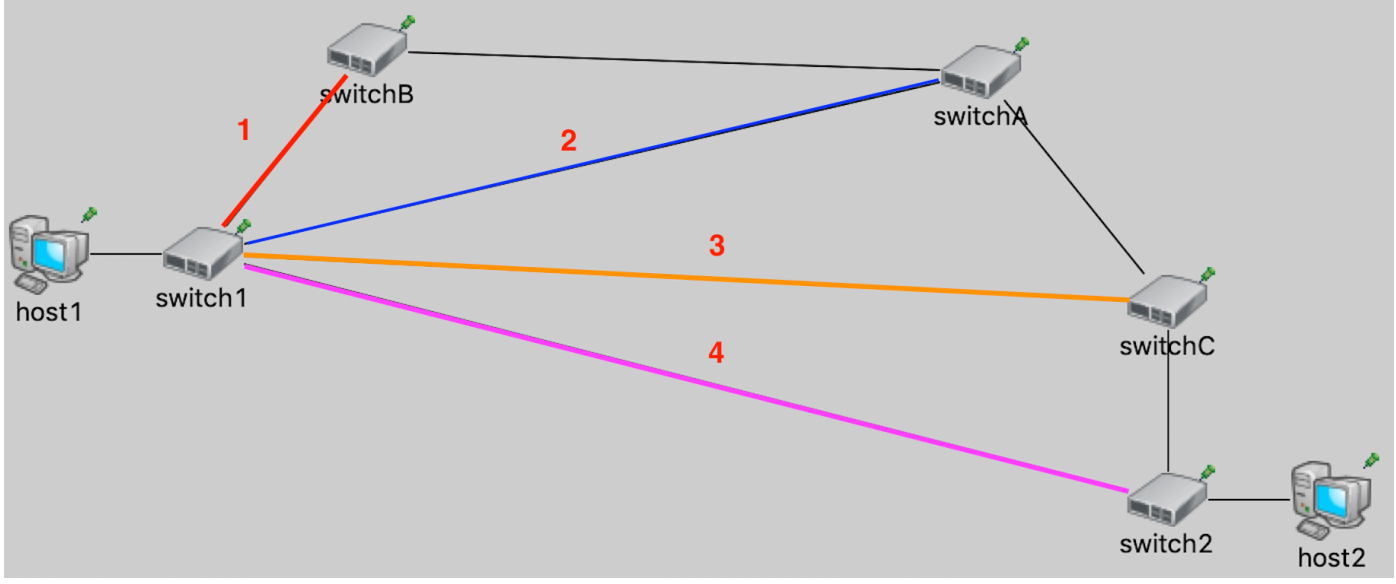


Outline

- Installation & Setting Environment
- Usage
- Running Simulation
- **Analysis**
  - Time Aware Shaper
  - Credit Based Shaper
  - Partition and Delay
  - **Routing Path and Delay**
- Limitation about Nesting

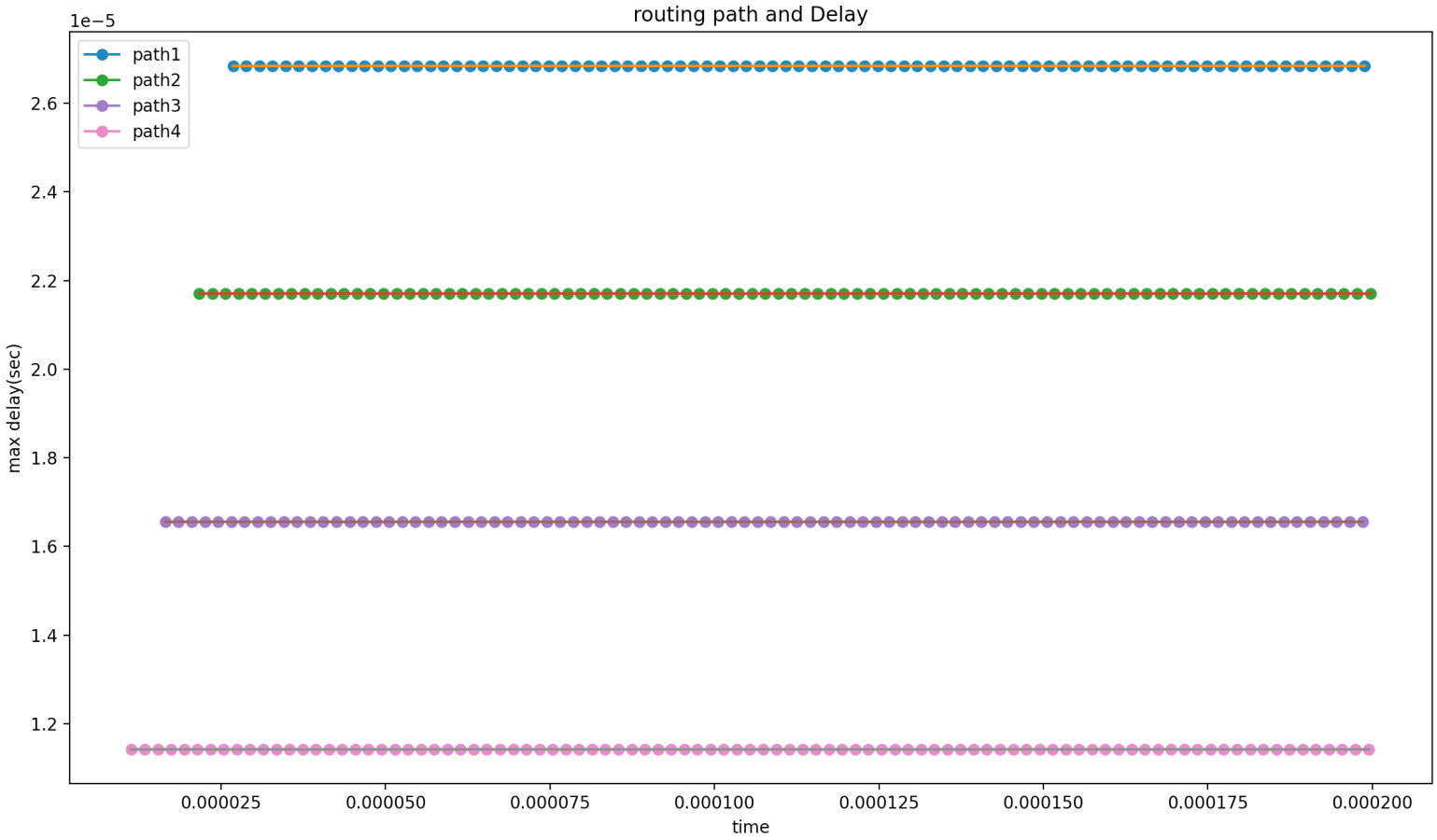
Routing Path and Delay(1/2)

- Definition
  - Transmit a fixed amount of data in fixed time, but go through different paths
  - Try to observe the relationship between the routing path and the maximum end to end delay.
- Scenario
  - Host1 send 100 bytes to Host2 every 2 ns
  - Datarate is 1Gbps for every links



Routing Path and Delay(2/2)

- Result
  - As can be seen from following figure, the more switches the packet pass, the higher the maximum end-to-end delay.



Outline

- ❑ Installation & Setting Environment
- ❑ Usage
- ❑ Running Simulation
- ❑ Analysis
- ❑ **Limitation about Nesting**

Routing Algorithm

- Filtering Database
  - Decide the output port only by the destination address of packets
  - Can't be used in software-defined network
    - In following scenario, flow1 and flow2 can not exist simultaneously