CO515: Advances in Computer Networks: Selected Topics – Lab11
SNMP Lab for Network Monitoring with Mininet

E/19/446: Wijerathna I.M.K.D.I.

27/06/2024

Activity 1: Setting up Mininet Environment

Since, configuring SNMP in each host which are inside a Mininet network required each host to be attached to a VM or a container, I used Mininet-Containernet platform.

Official documentation: https://containernet.github.io/

Using Mininet with Containernet, allows us to create a nested docker environment where we can run Mininet topologies so that we can use official and latest OS images for Mininet hosts.

The following is my lab environment setup along with the Mininet topology.

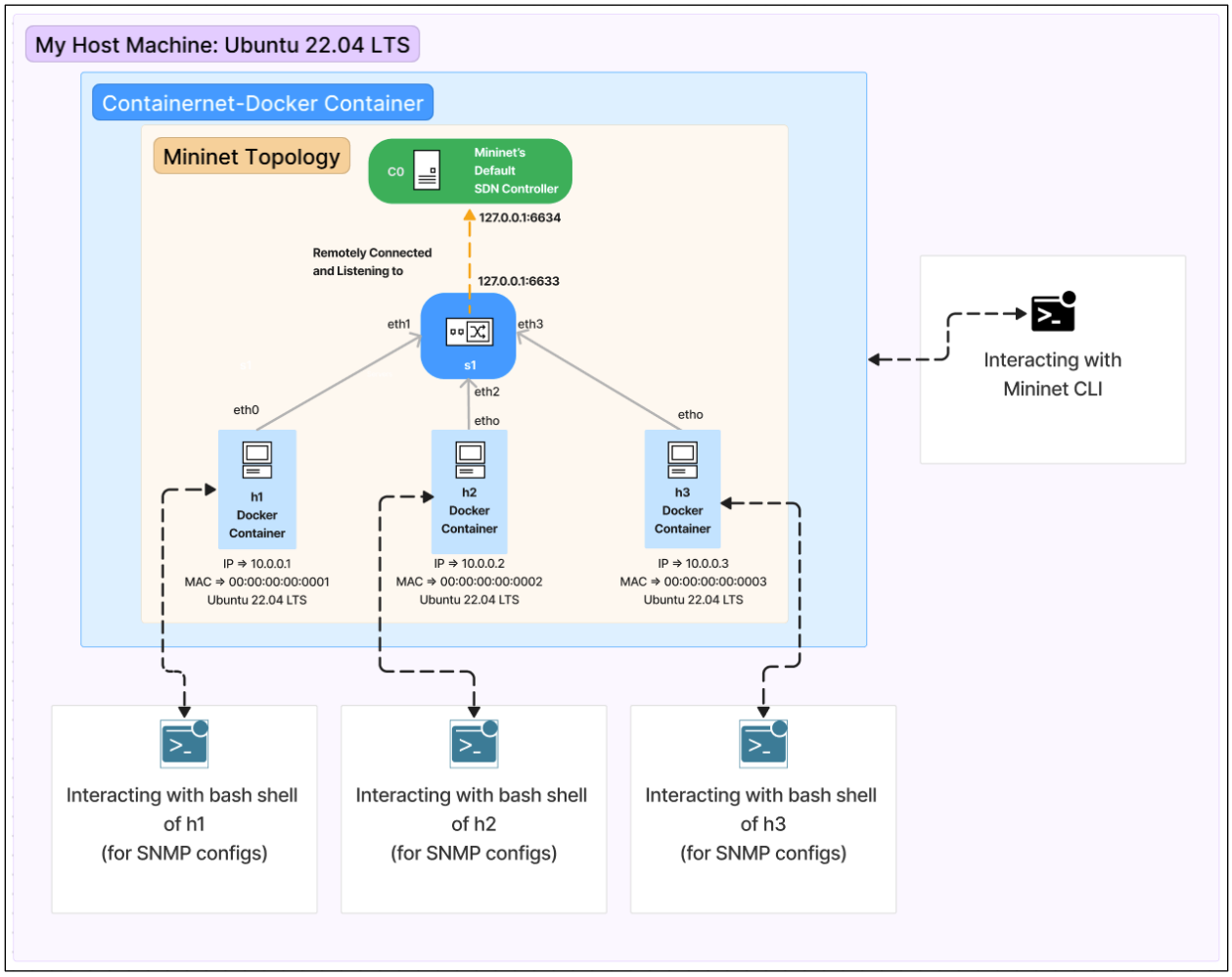

Figure 1: Lab Environment Setup(for Activity 1&2)
[CLI interactions for the hosts were performed using "docker exec -it" commands]

The python code used for defining the Mininet topology: (my-topo.py)

```
from mininet.net import Containernet # for Conatinerizing Mininet hosts
from mininet.node import Docker,Controller,OVSSwitch
from mininet.link import TCLink
from mininet.log import setLogLevel, info
from mininet.cli import CLI  # Import CLI for Containernet


def setup_network():
    net = Containernet(controller=Controller, switch=OVSSwitch) # using defualt controller and
OpenFlow switch

    print("*** Adding controller")
    net.addController('c0')

    info('*** Adding docker containers\n')
    h1 = net.addDocker('h1', ip='10.0.0.1', dimage="ubuntu:latest", mac='00:00:00:00:00:01',
shell='/bin/bash')
    h2 = net.addDocker('h2', ip='10.0.0.2', dimage="ubuntu:latest", mac='00:00:00:00:00:02',
shell='/bin/bash')
    h3 = net.addDocker('h3', ip='10.0.0.3', dimage="ubuntu:latest", mac='00:00:00:00:00:03',
shell='/bin/bash')

    info('*** Adding switch\n')
    s1 = net.addSwitch('s1')

    info('*** Creating links\n')
    net.addLink(h1, s1, cls=TCLink)
    net.addLink(h2, s1, cls=TCLink)
    net.addLink(h3, s1, cls=TCLink)

    # Install necessary packages and configure IP addresses
    for h, ip in zip([h1, h2, h3], ['10.0.0.1', '10.0.0.2', '10.0.0.3']):
        h.cmd('apt-get update')
        h.cmd('apt-get install -y iputils-ping net-tools snmpd snmp vim')

        # Set IP address using ifconfig command
        h.cmd(f'ifconfig {h}-eth0 {ip} netmask 255.255.255.0')

        # Start SNMP service
        h.cmd('service snmpd start')

    info('*** Starting network\n')
    net.start()

    info('*** Running CLI\n')
    CLI(net)  # Use CLI(net) to enter the CLI

    info('*** Stopping network\n')
    net.stop()

if __name__ == '__main__':
    setLogLevel('info')
    setup_network()
```

Starting the Mininet topology and verifying the network setup by pinging between hosts.

```
root@54a1ec28ca17:/containernet/examples# sudo vim my-topo.py
root@54a1ec28ca17:/containernet/examples# python3 ./my-topo.py
*** Error setting resource limits. Mininet's performance may be affected.
*** Adding controller
*** Adding docker containers
h1: kwargs {'ip': '10.0.0.1', 'mac': '00:00:00:00:00:01', 'shell': '/bin/bash'}
h1: update resources {}
h2: kwargs {'ip': '10.0.0.2', 'mac': '00:00:00:00:00:02', 'shell': '/bin/bash'}
h2: update resources {}
h3: kwargs {'ip': '10.0.0.3', 'mac': '00:00:00:00:00:03', 'shell': '/bin/bash'}
h3: update resources {}
*** Adding switch
*** Creating links
*** Starting network
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Running CLI
*** Starting CLI:
containernet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
containernet>
```

Figure 2: Running and Successfully pingall the network

After the above verification of the network connectivity, it was observed that all the four docker containers are running as expected as show below.

```
kalindu@kalindu-Inspiron-13-5310:~/Lab11/workspace/containernet$ docker ps
CONTAINER ID   IMAGE                      COMMAND                  CREATED          STATUS          PORTS     NAMES
6fec7f33550b   ubuntu:latest              "/bin/bash"              14 seconds ago   Up 14 seconds             mn.h3
b5e0bc243fae   ubuntu:latest              "/bin/bash"              15 seconds ago   Up 14 seconds             mn.h2
42c1a7f17ce5   ubuntu:latest              "/bin/bash"              15 seconds ago   Up 15 seconds             mn.h1
54a1ec28ca17   containernet/containernet  "util/docker/entrypo…"   6 hours ago      Up 6 hours                containernet
```

Figure 3: Docker containers overview after starting the network

Thereafter, I started configuring the SNMP agents in each of the three hosts in the following manner.

The edited "/etc/snmp/snmpd.conf" file on each host.(The configuration was exactly the same for all three hosts)

```
###############################################################################
# snmpd.conf
###############################################################################
# SECTION: System Information Setup

sysLocation    Sitting on the Dock of the Bay
sysContact     Me <me@example.org>
```

```
# sysservices: The proper value for the sysServices object | arguments:  sysservices_number
sysServices    72


###############################################################################
# SECTION: Agent Operating Mode
master  agentx
# agentaddress: The IP address and port number that the agent will listen on.
agentaddress   udp:161,udp6:[::1]:161



###############################################################################
# SECTION: Access Control Setup
#  system + hrSystem groups only
view   all  included   .1.3.6.1.2.1.1
view   all  included   .1.3.6.1.2.1.25.1


# rocommunity: a SNMPv1/SNMPv2c read-only access community name
#   arguments:  community    [default|hostname|network/bits] [oid | -V view]
# Read-only access to everyone to the systemonly view
rocommunity  public default -V all
rocommunity6 public default -V all

# rouser: a SNMPv3 read-only access username
#    arguments: username [noauth|auth|priv [OID | -V VIEW [CONTEXT]]]
rouser authPrivUser authpriv -V systemonly

# include a all *.conf files in a directory
includeDir /etc/snmp/snmpd.conf.d
```

```
kalindu@kalindu-Inspiron-13-5310:~/Lab11/workspace/containernet$ docker exec -it mn.h1 /bin/bash
root@h1:/# vim /etc/snmp/snmpd.conf
root@h1:~# netstat -tuln | grep 161
root@h1:~# /etc/init.d/snmpd start
 * Starting SNMP Services snmpd
root@h1:~# netstat -tuln | grep 161
udp        0      0 0.0.0.0:161             0.0.0.0:*
udp6       0      0 :::161                  :::*
root@h1:~#
```

Figure 4: Editing the snmpd.conf and restarting the service

As it can be observed in the snapshot above, snmp was successfully running on Mininet-host 'h1' and listening to all udp connections. Other 2 hosts were configured similarly to setup SNMP.

Activity 2: Querying SNMP Data

Exercise: Query SNMP data for system uptime, interface statistics, and other relevant metrics. (Using snmpwalk and snmpget commands)

For this task, I logged into the bash shell of Mininet-host 'h2' and queried SNMP data from h1 and h3.

1) *Initial Testing for SNMP connection:*

For h1:

```
root@h2:/# snmpwalk -v2c -c public 10.0.0.1 .1.3.6.1.2.1.25.1
iso.3.6.1.2.1.25.1.1.0 = Timeticks: (2822600) 7:50:26.00
iso.3.6.1.2.1.25.1.2.0 = Hex-STRING: 07 E8 07 03 0B 33 00 00 2B 00 00
iso.3.6.1.2.1.25.1.3.0 = INTEGER: 393216
iso.3.6.1.2.1.25.1.4.0 = STRING: "BOOT_IMAGE=/boot/vmlinuz-6.5.0-41-generic root=UUID=729b4fc1-0ac4-4b12-89a6-3840e84e8401 ro quiet splash v
t.handoff=7
"
iso.3.6.1.2.1.25.1.5.0 = Gauge32: 0
iso.3.6.1.2.1.25.1.6.0 = Gauge32: 4
iso.3.6.1.2.1.25.1.7.0 = INTEGER: 0
```

For h3:

```
root@h2:/# snmpwalk -v2c -c public 10.0.0.3
iso.3.6.1.2.1.1.1.0 = STRING: "Linux h3 6.5.0-41-generic #41~22.04.2-Ubuntu SMP PREEMPT_DYNAMIC Mon Jun  3 11:32:55 UTC 2 x86_64"
iso.3.6.1.2.1.1.2.0 = OID: iso.3.6.1.4.1.8072.3.2.10
iso.3.6.1.2.1.1.3.0 = Timeticks: (6628) 0:01:06.28
iso.3.6.1.2.1.1.4.0 = STRING: "Me <me@example.org>"
iso.3.6.1.2.1.1.5.0 = STRING: "h3"
iso.3.6.1.2.1.1.6.0 = STRING: "Sitting on the Dock of the Bay"
iso.3.6.1.2.1.1.7.0 = INTEGER: 72
iso.3.6.1.2.1.1.8.0 = Timeticks: (0) 0:00:00.00
iso.3.6.1.2.1.1.9.1.2.1 = OID: iso.3.6.1.6.3.10.3.1.1
iso.3.6.1.2.1.1.9.1.2.2 = OID: iso.3.6.1.6.3.11.3.1.1
iso.3.6.1.2.1.1.9.1.2.3 = OID: iso.3.6.1.6.3.15.2.1.1
iso.3.6.1.2.1.1.9.1.2.4 = OID: iso.3.6.1.6.3.1
iso.3.6.1.2.1.1.9.1.2.5 = OID: iso.3.6.1.6.3.16.2.2.1
iso.3.6.1.2.1.1.9.1.2.6 = OID: iso.3.6.1.2.1.49
iso.3.6.1.2.1.1.9.1.2.7 = OID: iso.3.6.1.2.1.50
iso.3.6.1.2.1.1.9.1.2.8 = OID: iso.3.6.1.2.1.4
iso.3.6.1.2.1.1.9.1.2.9 = OID: iso.3.6.1.6.3.13.3.1.3
```

2) *System Uptime [OID= 1.3.6.1.2.1.1.3.0]:*

This OID returns the time (in hundredths of a second) since the network management portion of the system was last re-initialized.

For h1:

```
root@h2:/# snmpget -v2c -c public 10.0.0.1 1.3.6.1.2.1.1.3.0
iso.3.6.1.2.1.1.3.0 = Timeticks: (76632) 0:12:46.32
```

For h3:

```
root@h2:/# snmpget -v2c -c public 10.0.0.3 1.3.6.1.2.1.1.3.0
iso.3.6.1.2.1.1.3.0 = Timeticks: (14819) 0:02:28.19
```

3) *System Description [OID= 1.3.6.1.2.1.1.1.0]:*

This OID provides a textual description of the entity.

For h1:

```
root@h2:/# snmpget -v2c -c public 10.0.0.1 1.3.6.1.2.1.1.1.0
iso.3.6.1.2.1.1.1.0 = STRING: "Linux h1 6.5.0-41-generic #41~22.04.2-Ubuntu SMP PREEMPT_DYNAMIC Mon Jun  3 11:32:55 UTC 2 x86_64"
```

For h3:

```
root@h2:/# snmpget -v2c -c public 10.0.0.3 1.3.6.1.2.1.1.1.0
iso.3.6.1.2.1.1.1.0 = STRING: "Linux h3 6.5.0-41-generic #41~22.04.2-Ubuntu SMP PREEMPT_DYNAMIC Mon Jun  3 11:32:55 UTC 2 x86_64"
```

4) *System Name [OID=1.3.6.1.2.1.1.5.0]:*

This OID returns the name of the device.

For h1:

```
root@h2:/# snmpget -v2c -c public 10.0.0.1 1.3.6.1.2.1.1.5.0
iso.3.6.1.2.1.1.5.0 = STRING: "h1"
```

For h3:

```
root@h2:/# snmpget -v2c -c public 10.0.0.3 1.3.6.1.2.1.1.5.0
iso.3.6.1.2.1.1.5.0 = STRING: "h3"
```

5) *List all interfaces [OID=1.3.6.1.2.1.2.2.1.2]:*

For h1:

```
root@h2:/# snmpwalk -v2c -c public 10.0.0.1 1.3.6.1.2.1.2.2.1.2
iso.3.6.1.2.1.2.2.1.2.1 = STRING: "lo"
iso.3.6.1.2.1.2.2.1.2.86 = STRING: "h1-eth0"
iso.3.6.1.2.1.2.2.1.2.110 = STRING: "eth0"
```

For h3:

```
root@h2:/# snmpwalk -v2c -c public 10.0.0.3 1.3.6.1.2.1.2.2.1.2
iso.3.6.1.2.1.2.2.1.2.1 = STRING: "lo"
iso.3.6.1.2.1.2.2.1.2.90 = STRING: "h3-eth0"
iso.3.6.1.2.1.2.2.1.2.114 = STRING: "eth0"
```

6) *CPU Load [OID=1.3.6.1.4.1.2021.10.1.3]:*

For h1:

```
root@h2:/# snmpwalk -v2c -c public 10.0.0.1 1.3.6.1.4.1.2021.10.1.3
iso.3.6.1.4.1.2021.10.1.3.1 = STRING: "0.50"
iso.3.6.1.4.1.2021.10.1.3.2 = STRING: "0.32"
iso.3.6.1.4.1.2021.10.1.3.3 = STRING: "0.25"
```

For h3:

```
root@h2:/# snmpwalk -v2c -c public 10.0.0.3 1.3.6.1.4.1.2021.10.1.3
iso.3.6.1.4.1.2021.10.1.3.1 = STRING: "0.50"
iso.3.6.1.4.1.2021.10.1.3.2 = STRING: "0.47"
iso.3.6.1.4.1.2021.10.1.3.3 = STRING: "0.35"
```

7) *Swap Memory*

a. *Total Swap memory [OID=1.3.6.1.4.1.2021.4.3.0]*

For h1:

```
root@h2:/# snmpget -v2c -c public 10.0.0.1 1.3.6.1.4.1.2021.4.3.0
iso.3.6.1.4.1.2021.4.3.0 = INTEGER: 2097148
```

For h3:

```
root@h2:/# snmpget -v2c -c public 10.0.0.3 1.3.6.1.4.1.2021.4.3.0
iso.3.6.1.4.1.2021.4.3.0 = INTEGER: 2097148
```

b. *Available Swap memory [OID=1.3.6.1.4.1.2021.4.4.0]*

For h1:

```
root@h2:/# snmpget -v2c -c public 10.0.0.1 1.3.6.1.4.1.2021.4.4.0
iso.3.6.1.4.1.2021.4.4.0 = INTEGER: 1739516
```

For h3:

```
root@h2:/# snmpget -v2c -c public 10.0.0.3 1.3.6.1.4.1.2021.4.4.0
iso.3.6.1.4.1.2021.4.4.0 = INTEGER: 1739516
```

8) *RAM*

a. *Total RAM [OID=1.3.6.1.4.1.2021.4.5.0]*

For h1:

```
root@h2:/# snmpget -v2c -c public 10.0.0.1 1.3.6.1.4.1.2021.4.5.0
iso.3.6.1.4.1.2021.4.5.0 = INTEGER: 7859944
```

For h3:

```
root@h2:/# snmpget -v2c -c public 10.0.0.3 1.3.6.1.4.1.2021.4.5.0
iso.3.6.1.4.1.2021.4.5.0 = INTEGER: 7859944
```

b. *Available RAM [OID=1.3.6.1.4.1.2021.4.6.0]*

For h1:

```
root@h2:/# snmpget -v2c -c public 10.0.0.1 1.3.6.1.4.1.2021.4.6.0
iso.3.6.1.4.1.2021.4.6.0 = INTEGER: 370264
```

For h3:

```
root@h2:/# snmpget -v2c -c public 10.0.0.3 1.3.6.1.4.1.2021.4.6.0
iso.3.6.1.4.1.2021.4.6.0 = INTEGER: 367376
```

Activity 3: Setting Up an SNMP Manager (Nagios/Zabbix)

For this part, I configured a Zabbix SNMP manager on h2 host by following this official documentation: https://www.zabbix.com/download?zabbix=7.0&os_distribution=ubuntu&os_version=22.04&components=server_frontend_agent&db=mysql&ws=apache

The topology is similar to Figure-1. However, slight changes were done in "my-topo.py" (Mininet topology), and "/etc/csnmp/snmpd.conf" files.

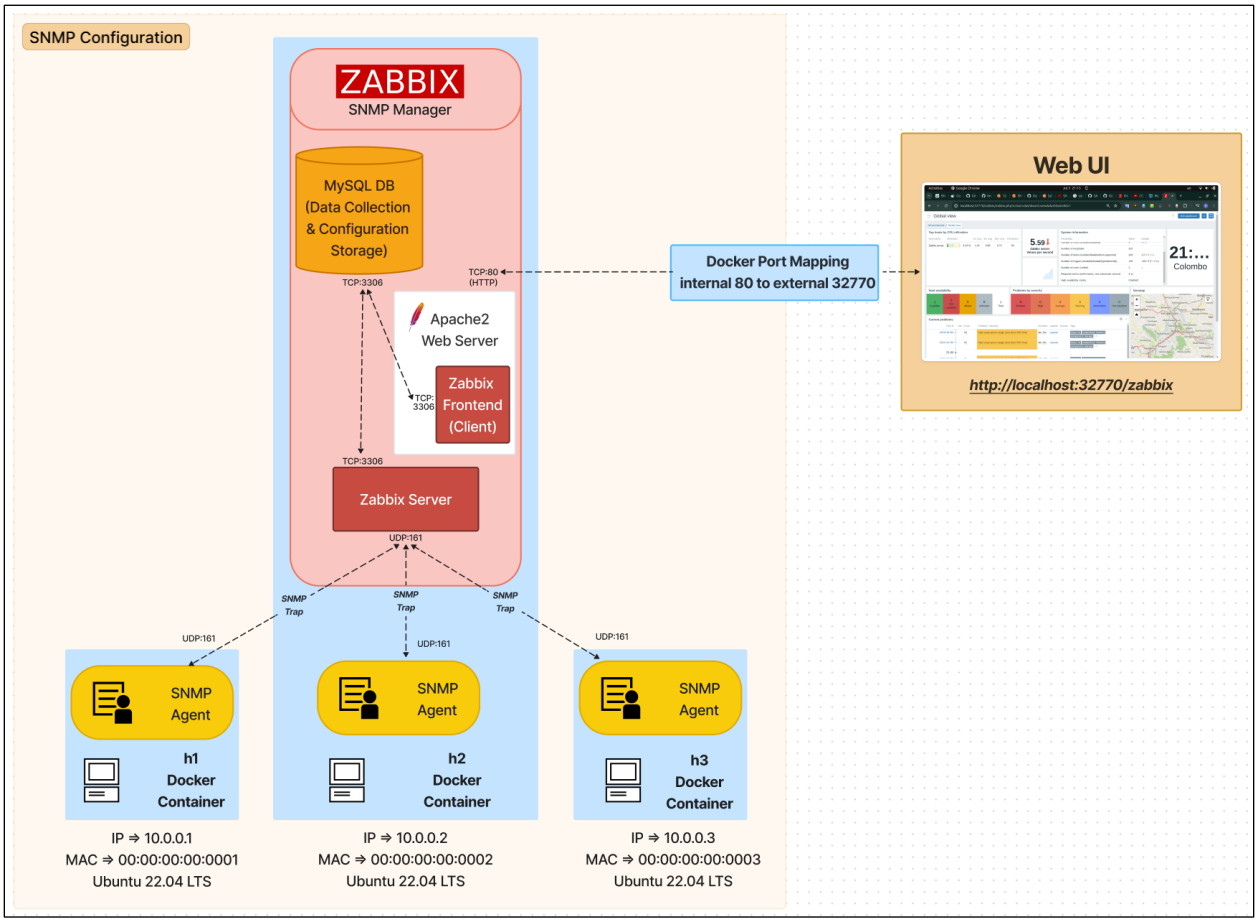

Figure 5: SNMP Configuration with Zabbix Manager

Following is the complete Zabbix SNMP Manager Configuration I used, along with the Mininet-Containernet Network Topology

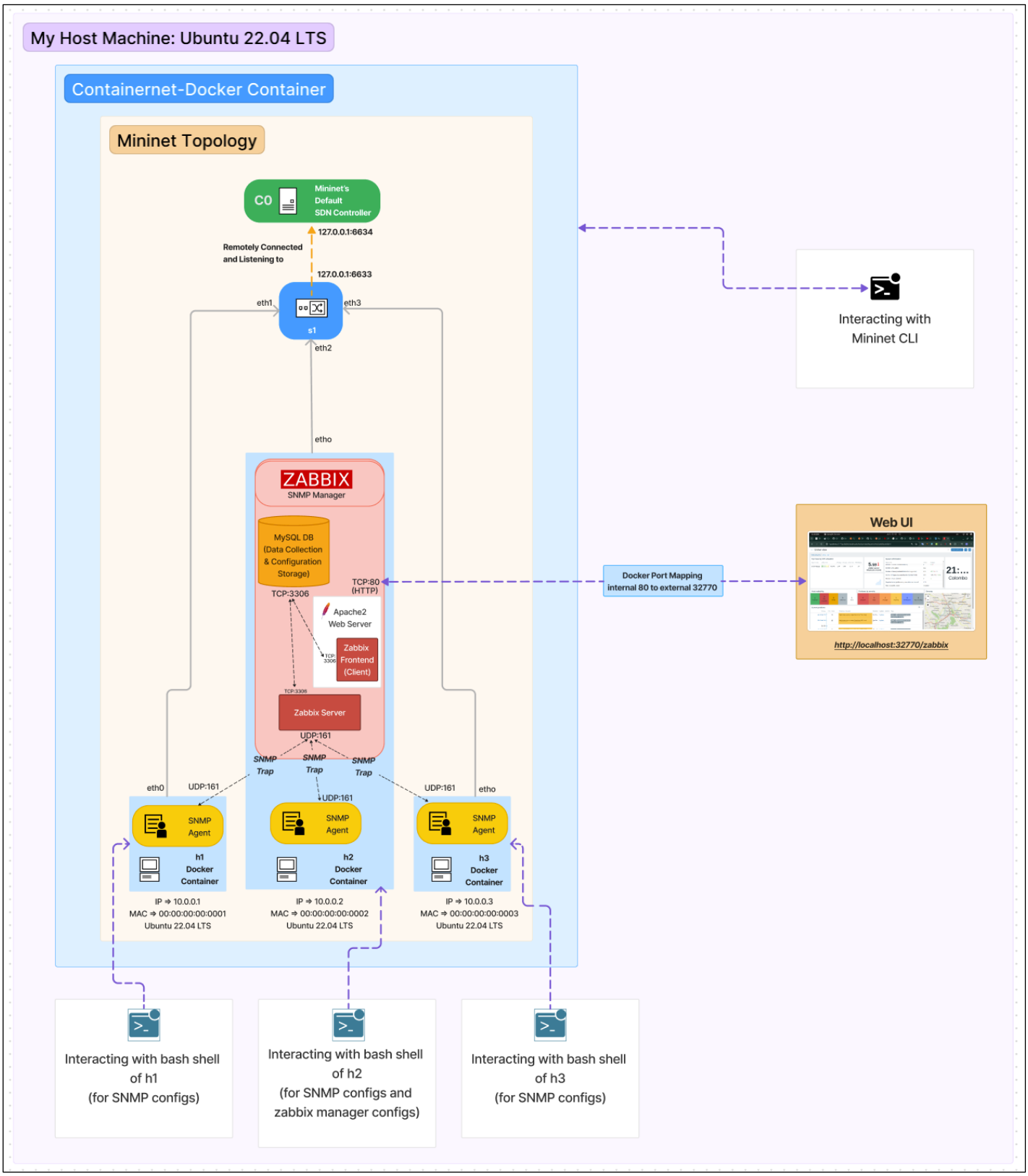

Figure 6: Complete Lab Environment Setup(for Activity 3&4) along with Zabbix Manager and Mininet-Contaiernet

- *Even though the Mininet topology is the same as in Activity 1 & 2, the ports regrading Zabbix services needed to be exposed in the new setup for h2. Therefore, the following change has been done in the Mininet Topology file: my-topo.py*

```
    # old config
    # h2 = net.addDocker('h2', ip='10.0.0.2', dimage="ubuntu:latest", mac='00:00:00:00:00:02', shell='/bin/bash')

    # new config
    h2 = net.addDocker( # exposing zabbix server ports
        'h2',
        ip='10.0.0.2',
        dimage="ubuntu:latest",
        mac='00:00:00:00:00:02',
        ports=[10051, 80],
        port_bindings={10051: 10051, 32770: 80},
        shell='/bin/bash'
    )
```

(full code for new topology has been attached in the Appendix)

- *snmpd.conf for all three hosts(h1,h2,h3) has been modified in order to connect with Zabbix SNMP manager through SNMP traps. The following is the new snmpd.conf used for all three hosts.*

```
###############################################################################
# snmpd.conf
###############################################################################
# SECTION: System Information Setup
# syslocation: The [typically physical] location of the system.
#   arguments:  location_string
sysLocation    Kandy-Home-Office
sysContact     me@home.org

###############################################################################
# SECTION: Agent Operating Mode
#   This section defines how the agent will operate when it is running.
# agentaddress: The IP address and port number that the agent will listen on.
#   arguments: [transport:]port[@interface/address],...
agentaddress   udp:161,udp6:[::1]:161

###############################################################################
# SECTION: Access Control Setup
#   This section defines who is allowed to talk to your runningsnmp agent.

# rocommunity: a SNMPv1/SNMPv2c read-only access community name
#   arguments:  community [default|hostname|network/bits] [oid | -V view]
rocommunity  public
```

- *Then, after following the Zabbix official documentation all services were configured and obtained the following Web page.*

Figure 7: Starting Zabbix Server, Zabbix Agent and Apache2

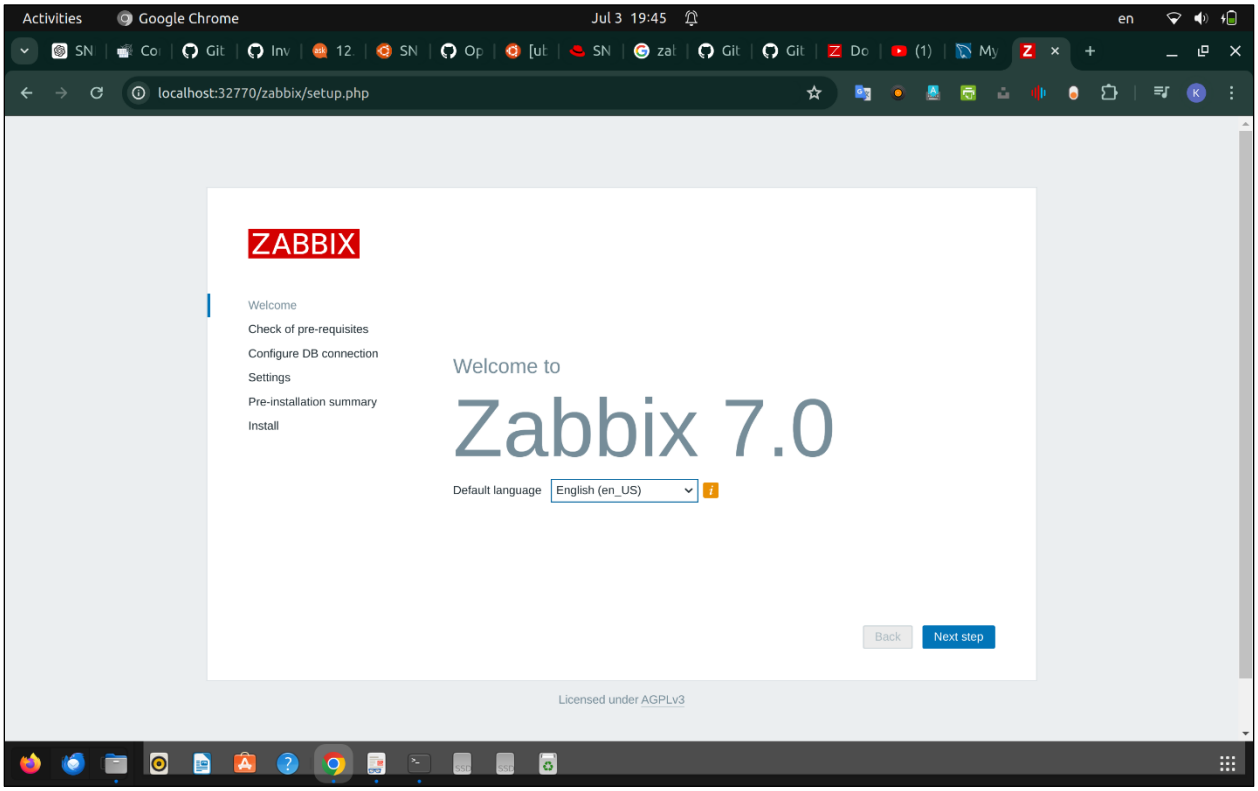

Figure 8: The Obtained Zabbix Setup Page

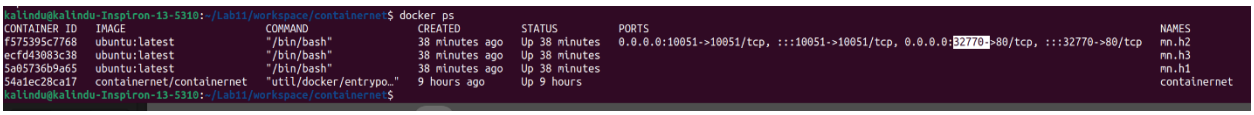

Figure 9: All docker containers successfully running while host h2 exposing necessary port for Zabbix services

The full Configuration steps and commands I followed are documented in the following repository:
https://github.com/KalinduWijerathna/CO515-Advanced-Networking-Labs-2024/

- *After Completing the Setup through the web UI the following Completion page was obtained.*

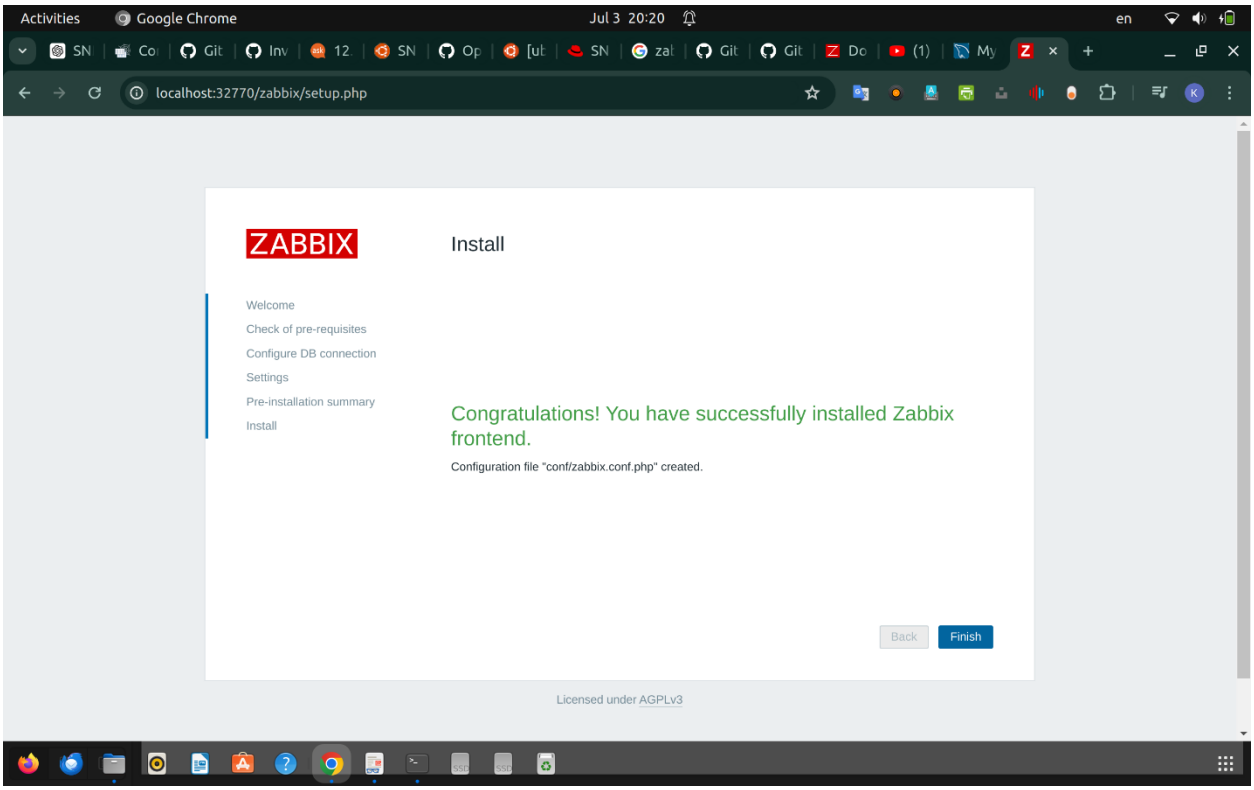

Figure 10: Zabbix Installation completion page

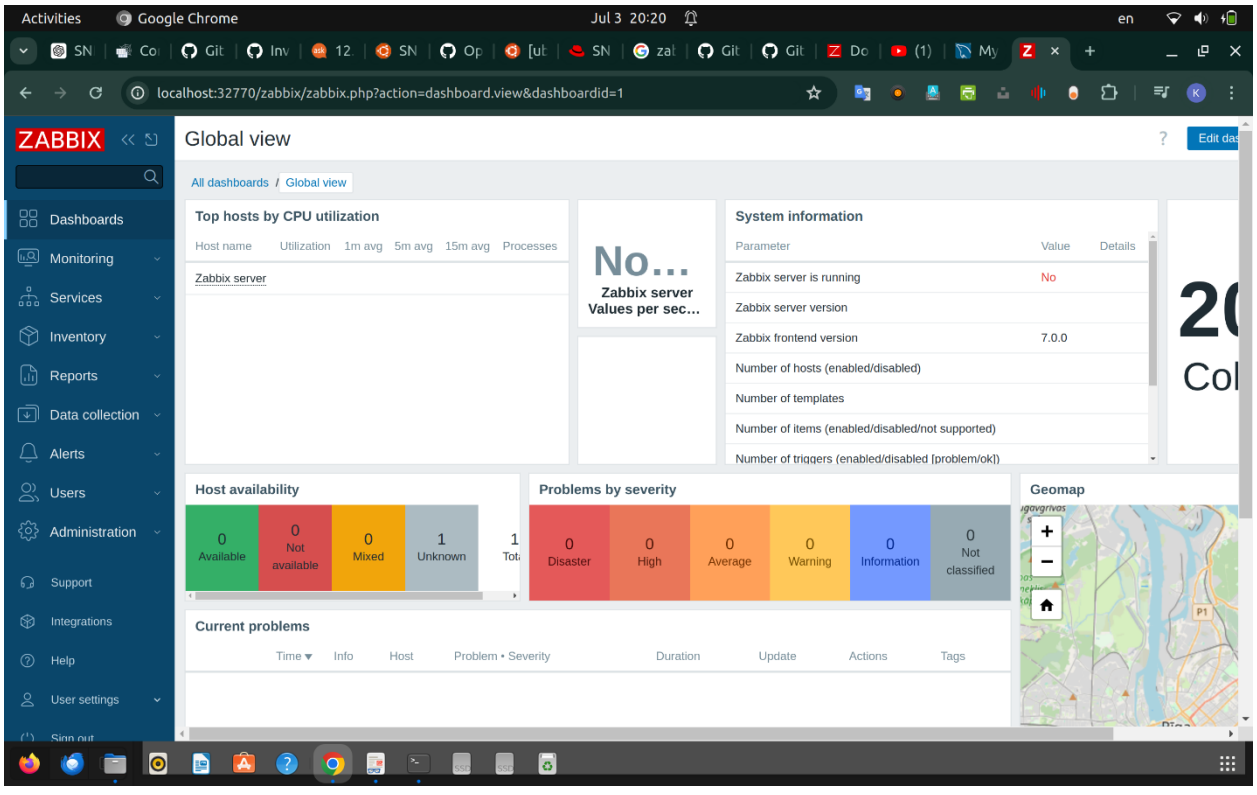

Figure 11: Initial Dashboard before connecting SNMP agents and SNMP traps

- *Adding the Mininet hosts as monitored devices in the SNMP manager*

  i) Setting up snmpd.conf in all three hosts using the above-mentioned configuration code

```
root@h1:/# vim /etc/snmp/snmpd.conf
root@h1:/# /etc/init.d/snmpd restart
 * Restarting SNMP Services snmpd                                                    [ OK ]
root@h1:/# netstat -tulpn
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
udp        0      0 127.0.0.1:161           0.0.0.0:*                           -
udp6       0      0 ::1:161                 :::*                                -
root@h1:/#
```

Figure 12: Setting up SNMP agents and verifying that the service is running for host h1

The configuration was exactly the same for h2,h3 hosts as well.

ii) Adding the SNMP agents as monitored devices

- *The three hosts were added with SNMP traps*
- *Problems were set up as well to notify, and inform warnings and information and other severe system data and important events*

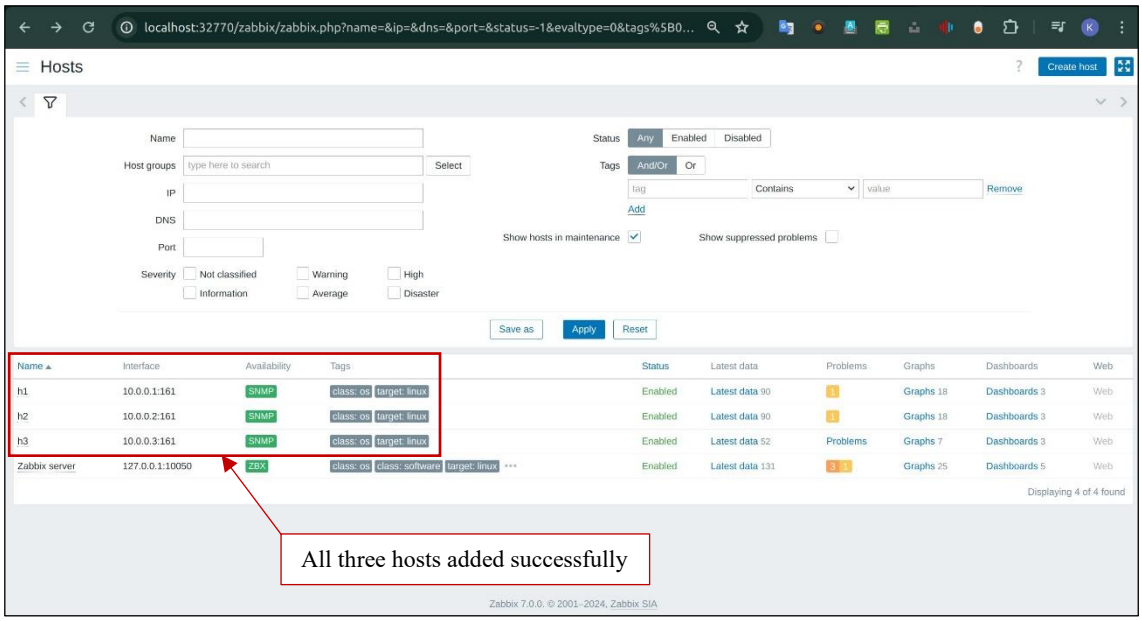

Figure 13: All three hosts being added to the Zabbix Server

Activity 4: Monitoring Network Performance

Simulate network traffic and monitor the impact on performance metrics.

- *The following is the overview of the Zabbix Dashboard*

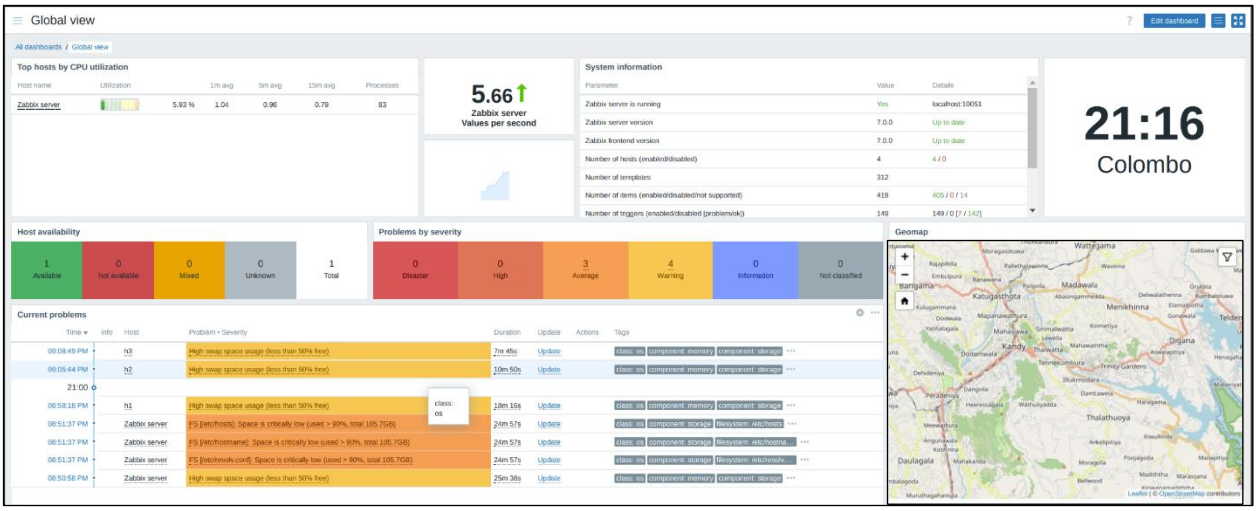

Figure 14: Zabbix Dashboard after Connecting SNMP agents

- *Simulating Network traffic in the Mininet network (using pingall)*

```
containernet> pingall
*** Ping: testing ping reachability
h1 -> h3 h2
h3 -> h1 h2
h2 -> h1 h3
*** Results: 0% dropped (6/6 received)
containernet>
```

This was continued with repeating 'pingall' and separate pings for selected hosts. The following are the Zabbix dashboards for each hosts network performance.

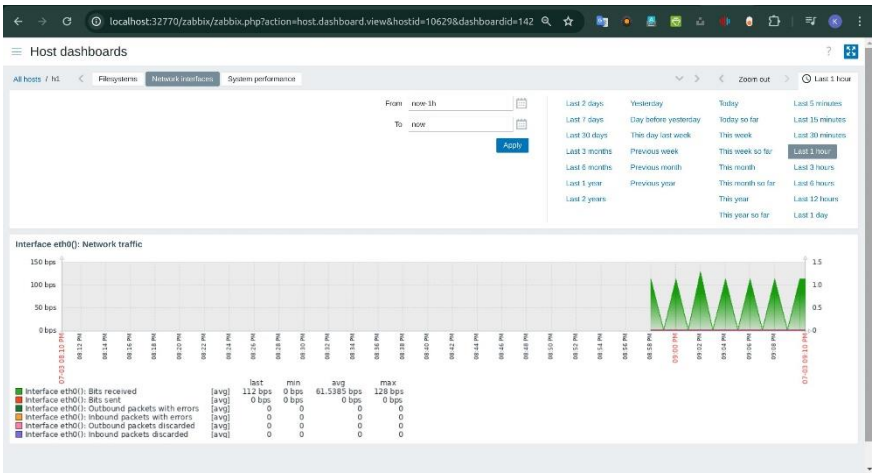

Figure 15: Network Performance Metrics for h1

Figure 16: Network Performance Metrics for h2

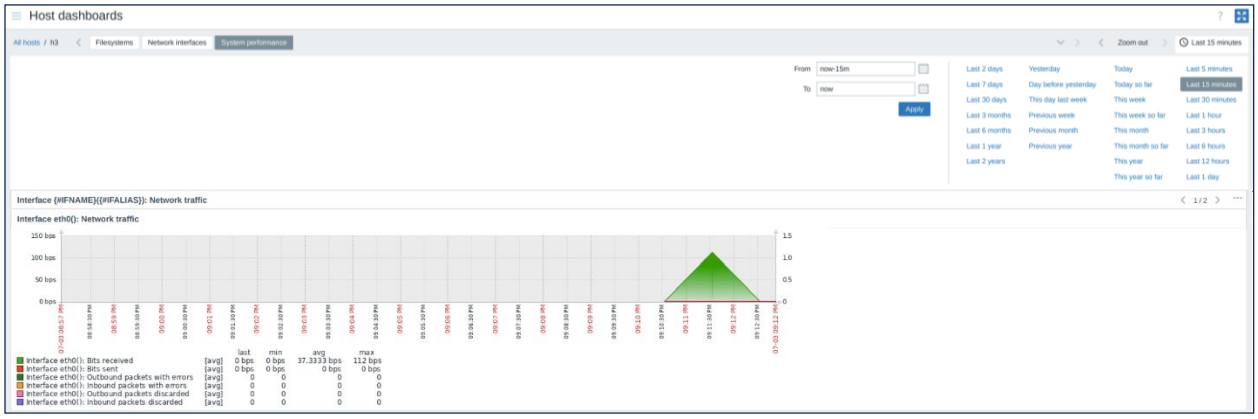

Figure 17: Network Performance Metrics for h3

By Observing these three graphs, it can be noted that h1, h3 both don't seem to have abnormal spikes, or errors in traffic flow. However, h2 show an unusual spike in packets-sent traffic, which suggest a possible increase of delay and a slight variation of jitter.

Following pages contain full dashboards (Overall System info) of Zabbix server, h1, h2, h3 as received by Zabbix server.

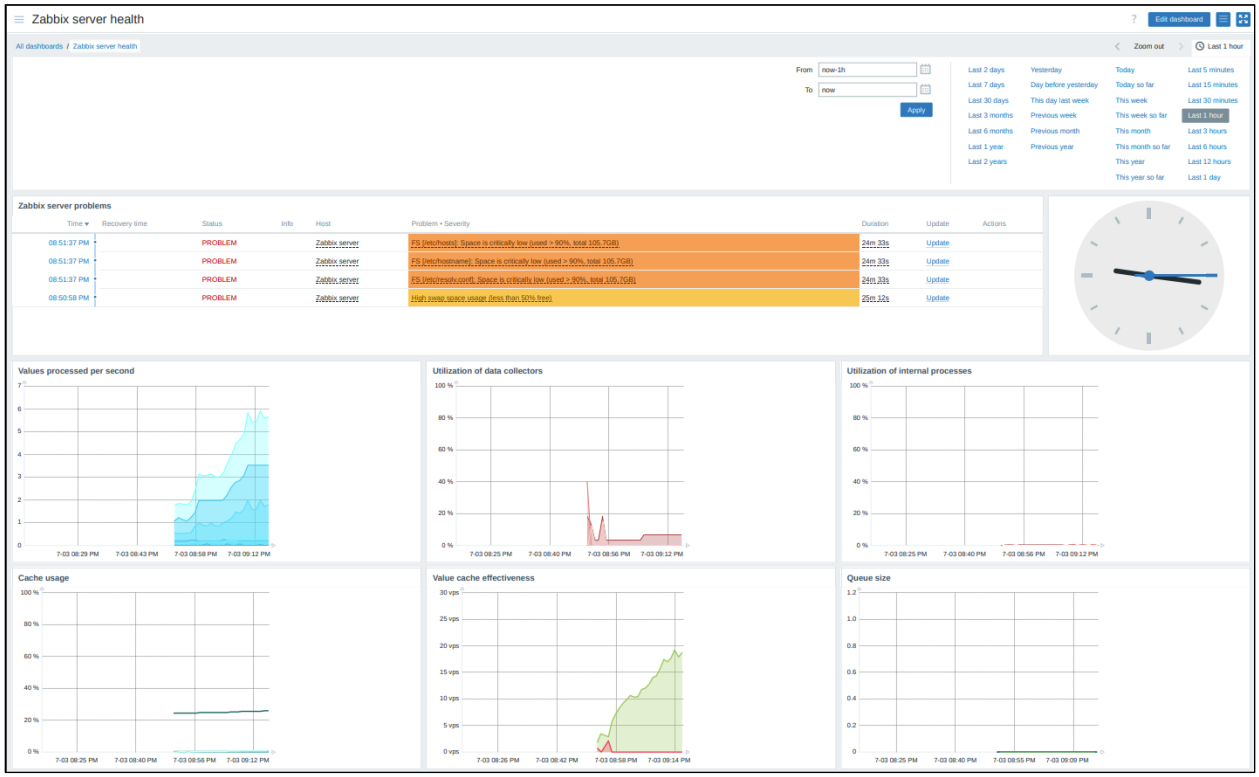

Figure 18: Zabbix Server Health Monitoring

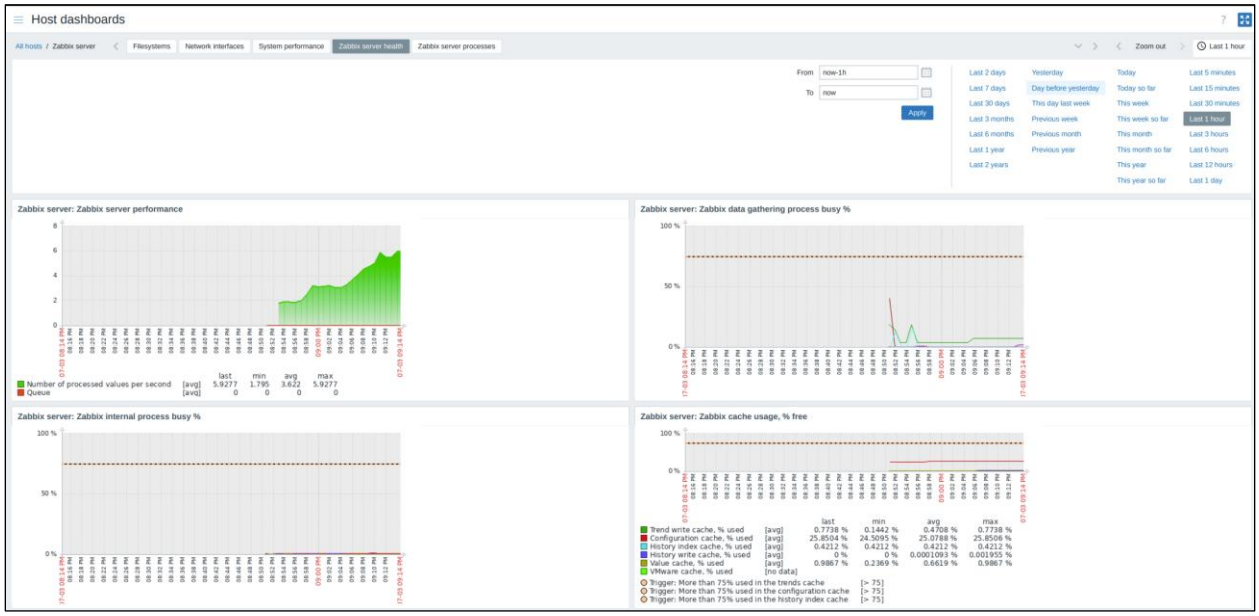

Figure 19: Zabbix Server Performance Monitoring

Mininet Host – h1

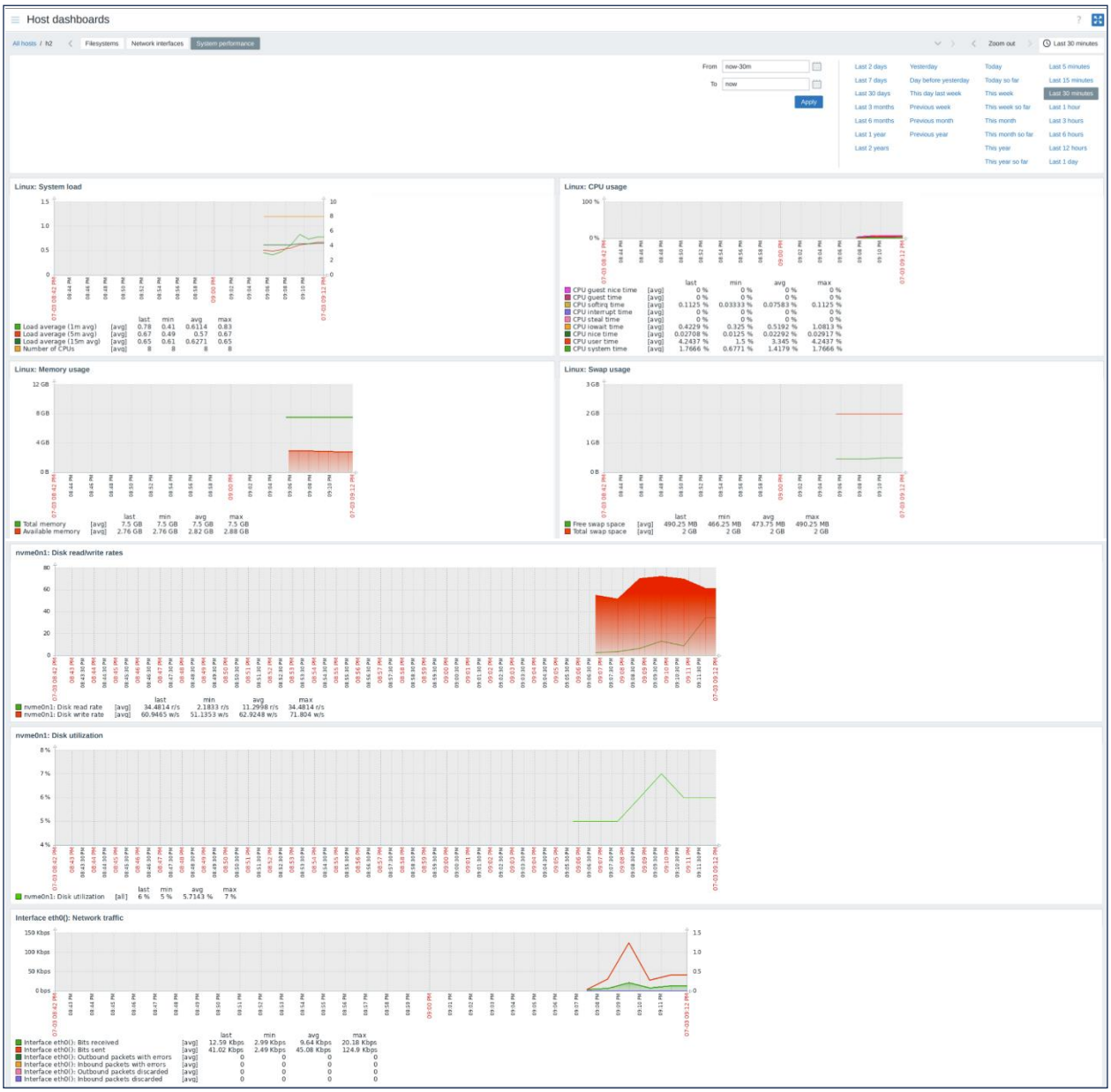

Figure 20: Mininet host - h1: full dashboard

Mininet Host – h2

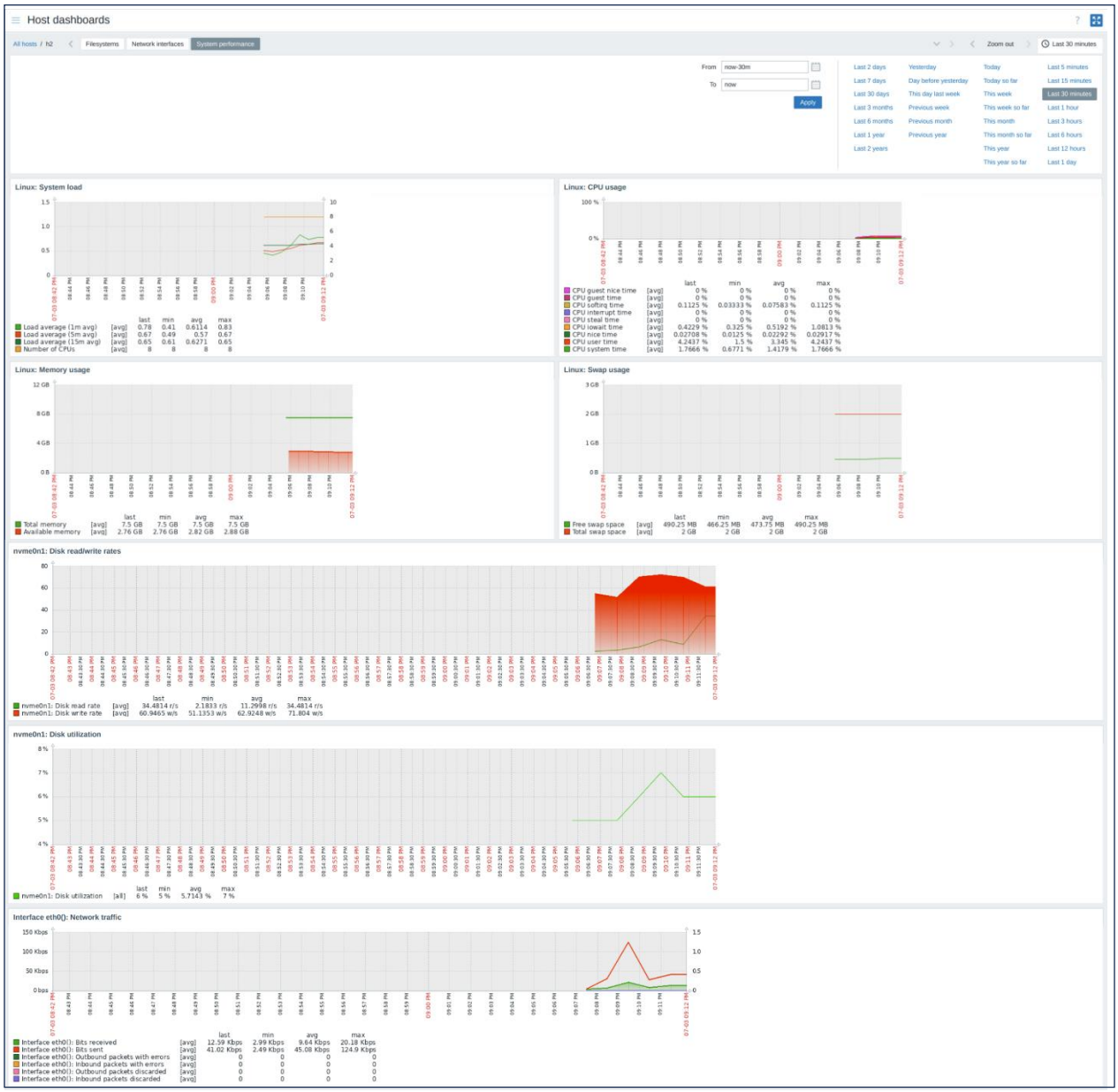

Figure 21: Mininet host - h2: full dashboard

Mininet Host – h3

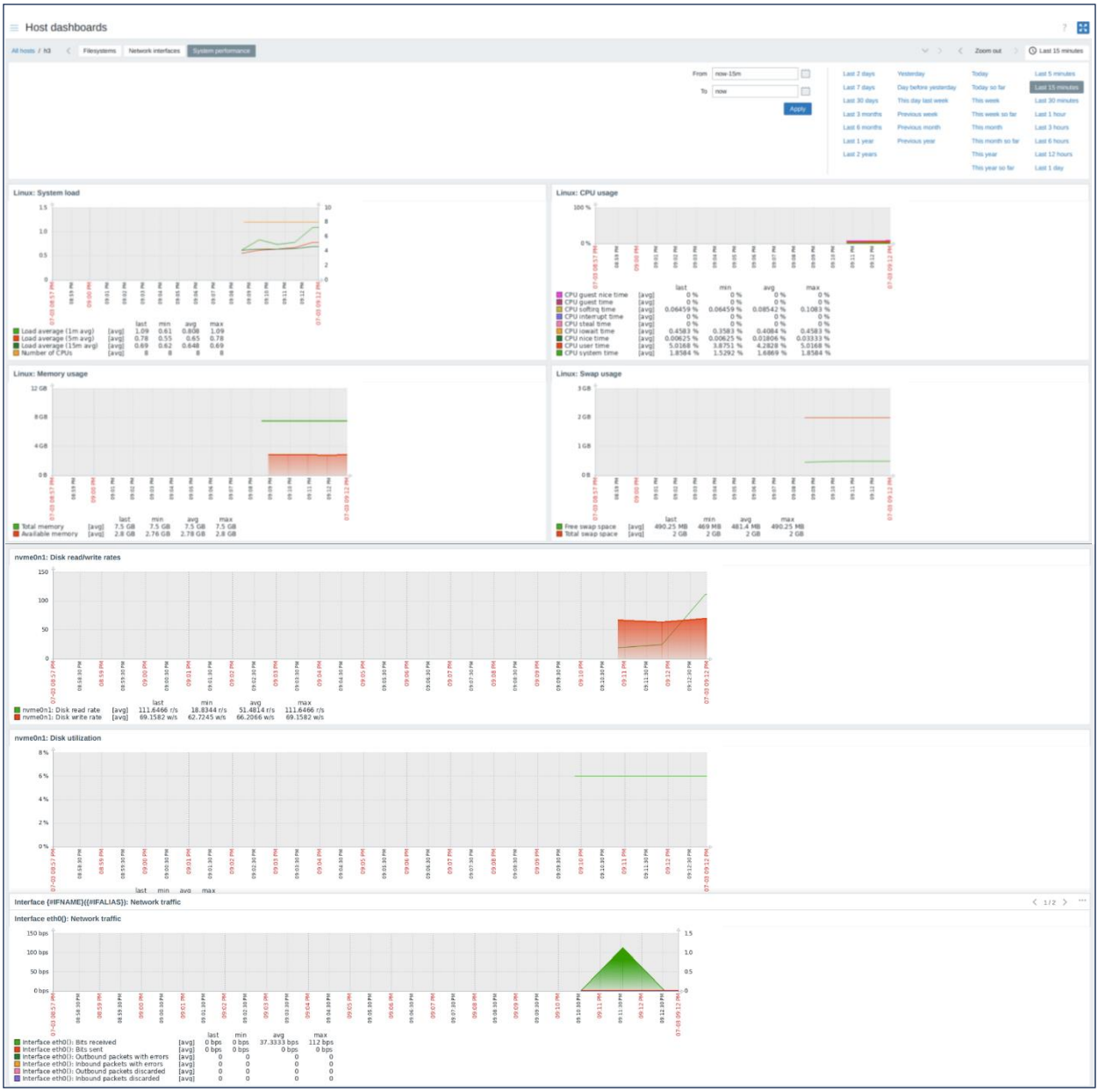

Figure 22: Mininet host - h3: full dashboard

Appendix

Python Code for Mininet-Containernet Topology with Zabbix SNMP Manager

(All the source codes along with config commands are documented in the following repository: https://github.com/KalinduWijerathna/CO515-Advanced-Networking-Labs-2024/)

```
from mininet.net import Containernet
from mininet.node import Docker,Controller,OVSSwitch
from mininet.link import TCLink
from mininet.log import setLogLevel, info
from mininet.cli import CLI  # Import CLI for Containernet


def setup_network():
    net = Containernet(controller=Controller, switch=OVSSwitch)

    print("*** Adding controller")
    net.addController('c0')

    info('*** Adding docker containers\n')
    h1 = net.addDocker('h1', ip='10.0.0.1', dimage="ubuntu:latest",
mac='00:00:00:00:00:01', shell='/bin/bash')
    h3 = net.addDocker('h3', ip='10.0.0.3', dimage="ubuntu:latest",
mac='00:00:00:00:00:03', shell='/bin/bash')
    # old config
    # h2 = net.addDocker('h2', ip='10.0.0.2', dimage="ubuntu:latest",
mac='00:00:00:00:00:02', shell='/bin/bash')

    # new config
    h2 = net.addDocker( # exposing zabbix server ports
        'h2',
        ip='10.0.0.2',
        dimage="ubuntu:latest",
        mac='00:00:00:00:00:02',
        ports=[10051, 80],
        port_bindings={10051: 10051, 32770: 80},
        shell='/bin/bash'
    )


    info('*** Adding switch\n')
    s1 = net.addSwitch('s1')
```

```
    info('*** Creating links\n')
    net.addLink(h1, s1, cls=TCLink)
    net.addLink(h2, s1, cls=TCLink)
    net.addLink(h3, s1, cls=TCLink)

    # Install necessary packages and configure IP addresses
    for h, ip in zip([h1, h2, h3], ['10.0.0.1', '10.0.0.2', '10.0.0.3']):
        h.cmd('apt-get update')
        h.cmd('apt-get install -y iputils-ping net-tools snmpd snmp vim')

        # Set IP address using ifconfig command
        h.cmd(f'ifconfig {h}-eth0 {ip} netmask 255.255.255.0')

        # Start SNMP service
        h.cmd('service snmpd start')


    info('*** Starting network\n')
    net.start()

    info('*** Running CLI\n')
    CLI(net)  # Use CLI(net) to enter the CLI

    info('*** Stopping network\n')
    net.stop()

if __name__ == '__main__':
    setLogLevel('info')
    setup_network()
```