2. (LAB2) Application Layer and Local LAN

Introduction: In this lab, we will explore various components in a subnet and illustrate small-scale enterprise network devices. The works include:

- [NetW1-part1] (Experiment 1) Implement a network application with hybrid mode, focus on the first phase client-server paradigm
- [NetW2] (Experiment 2) Create a network with several hosts, web HTTP server, DNS server
- [NetW1-part2] (Experiment 3) Sniffing and analyzing packets of the protocols HTTP, DNS, TCP and UDP.
- [NetW2] (Optional Experiment 4) Add router default gateway to the subnet.

Prerequisites: Before this lab, you must equip yourself with the following outcomes from the previous lab (Lab1):

- [NetW1] Complete perform the baseline client.py and server.py with all provided resources and guided tutorial.
- [NetW2] Complete the subnet with DHCP to allocate IP addresses
- [NetW2] Complete the HTTP server with web browser load.

  Check the right click on GNS3 link between 2 devices and successfully open packet sniff with Wire shark (check missing software only)

Outcomes: After finishing these mandatory tasks, student can:

- [NetW2] Setup a private network for a small-size organization, maybe up to 200 PCs..
- [NetW2] Understand the data format and working mechanism of protocols in application layer and transport layer: HTTP, DNS, TCP and UDP.
- [NetW1] Implement a network application with various (peer) nodes and exchange information

2.1. NetW1 (part 1): Programming Network applications

2.1.1. Experiment 1: implement netapp in the first phase of client-server paradigm

Prerequisites: With a test bed having a set of tool server.py client.py and a demo tutorial how to run these processes on different hosts.

The system design: The hybrid NetApp system involves two entities: the tracker (acted as server role) and the node (acted as peer). The hybrid NetApp operation has purely two, and separable, phase paradigms:

Phase1: client server paradigm

Step 1: send_info (peer_ip, peer_port)

Step 2: add_list (peer_ip, peer_port)

Step 3: get_list ()

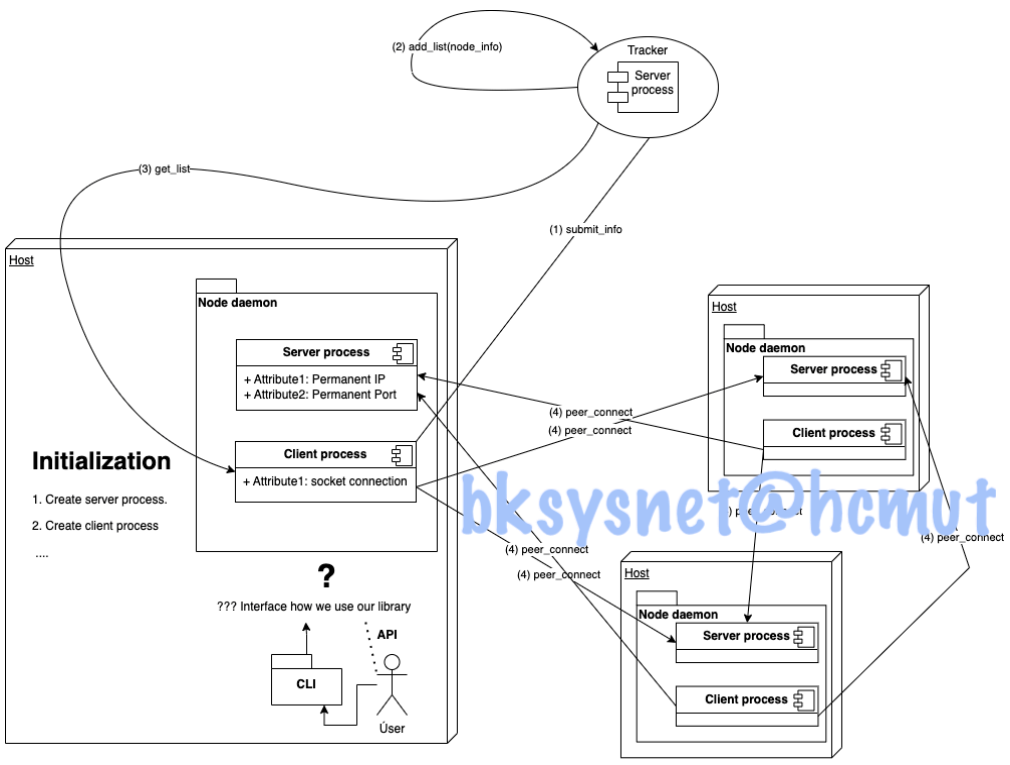
Phase 2: peer-to-peer paradigm

Step 4: peer_connect(IP_x, Port_x)

Step 5: peer_transfer(data)

The implementation requirements: implement the client-server paradigm with the following steps: step 1 and step 2. After step 2, the tracker (server) prints out the list of peer IP and peer port that it received and added to the peer list.

Students need to develop the function of send_info() at client program and add_list() at server program.



To implement the sending and receiving data, we use the function send() and recv() as in socket-objects section of socket[1].

Tip box: The two helper for sending and receiving data in Python

Sending Data

```
sendData = "text".encode("utf-8")
client_socket.send(sendData)
```

[1]https://docs.python.org/3/library/socket.html

Receiving Data

```
receiveData = client_socket.recv(1024)
data = receiveData.decode("utf-8") # convert bytes to string
# if we need print out
print(f"Received: {data}")
```

2.2. NetW2: Network infrastructure - Create Local LAN or a single subnet

2.2.1. Experiment 2: Create a local subnet with DHCP server, HTTP server and DNS server.

Step 1: Create a network as in the schematic in Figure: the schematic of a local subnet.

- Create hosts and servers by using the following VM template (check the e-learning portal for the VM image template resources)
  - NetApp-Server (use to create DHCP server and HTTP server)
  - NetApp-Host
  - NetApp-Server DNS (specify for DNS server)

Step 2: Configure the network system

In this work, we based on the previous work of **Lab 1** - NetW1: Protocol analyzer using Wireshark packet sniffer software, and changed the subnet to 192.168.**13**.x to make the difference.

- The HTTP server and DHCP server are configured as static IP. The network window is open by following steps:

(Taskbar placed at Screen Bottom) Control Panel > Network > Network Interface DHCP configuration. Set static IP

(1) HTTP Server 192.168.**13**.5
(2) DHCP Server 192.168.**13**.10
(3) DNS Server 192.168.**13**.**254**

2.2.1.1. Server step 1: Setup The DHCP server to performs DHCP service

- DHCP server need eth0 is configured as static IP
  - Right click on the desktop and choose SystemTool > Control Panel. Click on Network button
  - Choose **no** in use DHCP broadcast and set static IP:
    - IP address **192.168.13.10**
    - Network mask **255.255.255.0**
    - Broadcast **192.168.13.255**
    - Gateway **192.168.13.1** (it needs to allow PC sent packet to out of the subnet 192.168.13.x)
    - Name Server: **192.168.13.254**

Read IP address and verify the correct address to ensure the previous static configuration has been done

```
$ ifconfig
```

The IP must be 192.168.13.10

Run DHCP server

```
$ sudo udhcpd -f udhcpd.conf
```

The content of udhcpd.conf already existed on DHCP-Server Virtual Machine harddisk. The main configuration in the file are

```
start 192.168.13.100
end 192.168.13.200
interface eth0
option subnet 255.255.255.0
option router 192.168.13.1
option dns     192.168.13.254
```

#Notice: we need to check all lines of the configuration file carefully not copycat from the previous lab config. The required configuration has changed and updated to another level of configuration.

2.2.1.2. Server step 2:Setup The HTTP server to performs web service

- HTTP server need eth0 is configured as static IP
  - Right click on the desktop and choose SystemTool > Control Panel. Click on Network button
  - Choose **no** in use DHCP broadcast and set static IP:
    - IP address **192.168.13.5**
    - Network mask **255.255.255.0**

- Broadcast **192.168.13.255**
- Gateway **192.168.13.1** (it needs to allow PC sent packet to out of the subnet 192.168.13.x)
- Name Server: **192.168.13.254**

Read IP address and verify the correct address to ensure the previous static configuration has been done

```
$ ifconfig
```

The IP must be 192.168.13.5

Run DHCP server

```
$ sudo lighttpd-angel -Df lighttpd.conf
```

#Notice: lighttpd or lighttp-angel can be used exchangeable, self-reading the command manual page for more details

Optional only (do as homework)

This is an optional if you want to master the configuration, you can add the module into lighttpd.conf to make it log the HTTP request. To enable accesslog module, we can add the following configuration line into the tail of the configuration file lighttpd.conf
(skip this optional step if it make you a complicated problem, ignoring this configuration is fine)

```
server.modules += (
    “mod_accesslog”,
)

accesslog.filename=”/var/log/lighttpd.log”
```

then the file access log will be append to the tail of log file /var/log/lighthttpd.log

2.2.1.3. Server step 3: Setup the DNS server

- DNS server need eth0 is configured as static IP
  - Right click on the desktop and choose SystemTool > Control Panel. Click on Network button
  - Choose **no** in use DHCP broadcast and set static IP:
    - IP address **192.168.13.254**
    - Network mask **255.255.255.0**
    - Broadcast **192.168.13.255**
    - Gateway **192.168.13.1** (it needs to allow PC sent packet to out of the subnet 192.168.13.x)
    - Name Server: **192.168.13.254**

Read IP address and verify the correct address to ensure the previous static configuration has been done

```
$ ifconfig
```

The IP must be 192.168.13.254

Run DNS service

```
$ sudo /usr/local/sbin/named -4 -g -c /usr/local/etc/bind/named.conf
```

#Comment: named or bind is only the program implementing the DNS service server process. All the configuration is at /usr/local/etc/bind and the name.conf you can read and investigate by yourself later since its very popular service and well documented. For this stage, we use it in a simple way to perform computer lab course theory illustration.

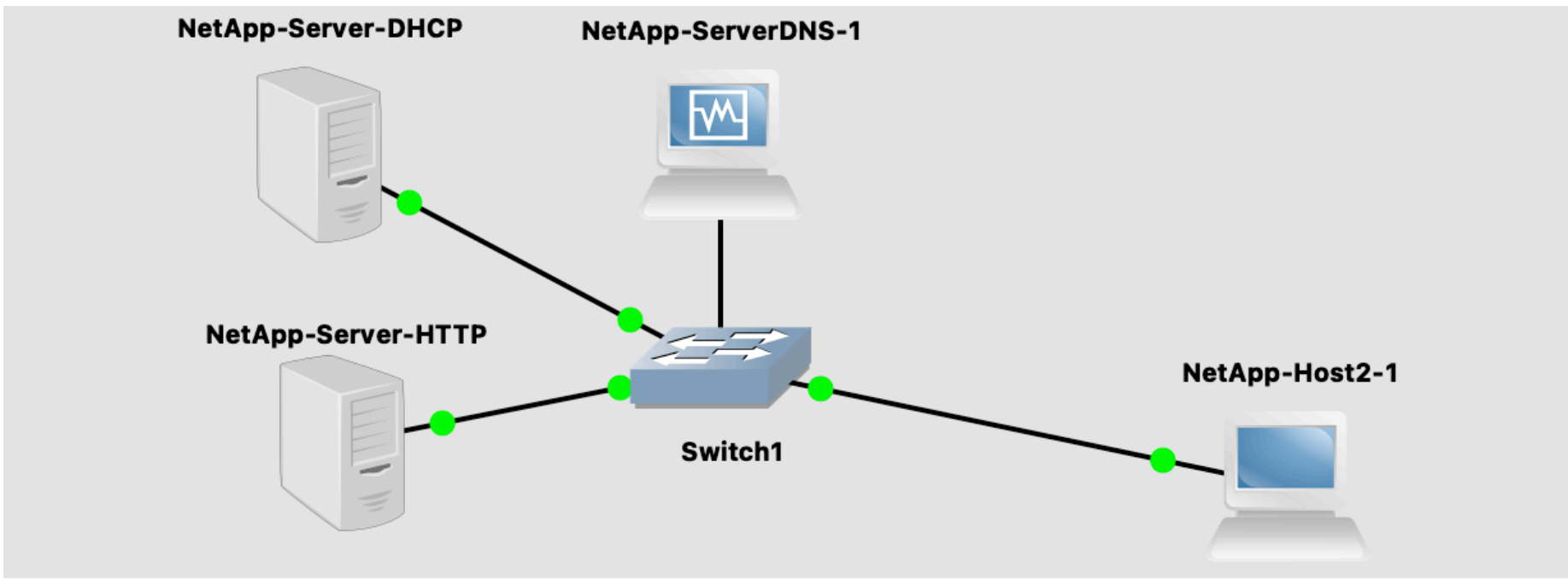


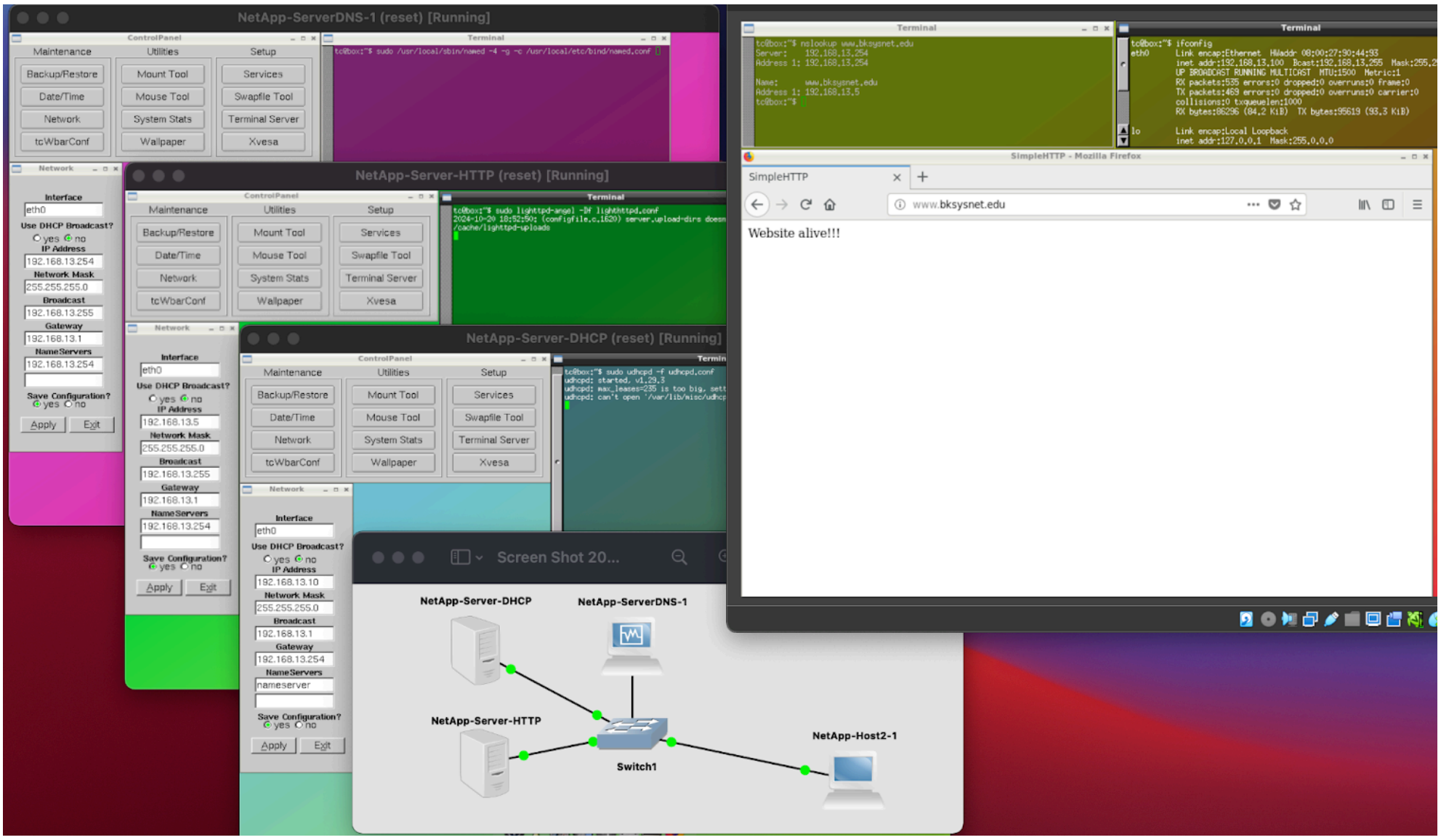
Figure 2.1 The schematic of local subnet

2.2.1.4. Set DHCP client on the NodeApp-Host

- Each NodeApp-Host has one NIC and is named eth0
- PCs need eth0 is configured as DHCP mode to automatically obtain the IP address:
  - Right click on the desktop and choose SystemTool > Control Panel. Click on Network button
  - Choose **yes** in use DHCP broadcast
  - Click Apply to commit the settings contents
- The result of IP obtaining can be verified through the command to display the network interface info. There are some alternative commands depending on the current OS on the VM.

```
$ ifconfig
```

#Notice: for lab 2, if your system got problem in IP allocation, you can set static IP for NetApp-Host as 192.168.13.200
with gateway 192.168.13.1 and name server 192.168.13.254
Finally, these hosts are connected through an Ethernet Switch.



Step 3: Verify the successful HTTP service and DNS

1) Verify the IP address is obtained (by using “ifconfig” or “ip addr” command)
2) Verify that all the hosts are connected together (by using ping and successfully ping between all pairs of hosts)
3) Verify that the browser successfully access the webpage on HTTP server

Verify that the name service work by open Terminal Window on NetApp-Host and type nslookup www.bksysnet.edu Verify that the web browser can open the webpage at the address www.bksysnet.edu
The troubleshoot: in the case of a failed name service, we can verify the HTTP service only by performing the lab1 IP address access. **Verification of HTTP service**: Open the web browser and open the website at the IP address http://<sever-IP-address>/ Ex: http://192.168.13.5 **Verification of DNS service**: nslookup www.bksysnet.net

2.3. [NetW1 - part 2] Sniffing and analyzing packets of the protocols HTTP, DNS, TCP and UDP.

2.3.1. Experiment 3 Sniffing and analyzing packets

Using the wireshark to perform the packet sniffing.

2.3.1.1. Sniffing 1: packet DNS

Step 1: open sniffing packet tool on the network link of the NetApp-Host.
Step 2: use nslookup to create a naming query. For the details of nslookup refer to the appendix nslookup. On a NetApp-Host open a terminal window and execute the following command

```
nslookup www.bksysnet.edu
```

Step 3: Open wireshark capture window and locate DNS packets: In the packet list pane, look for packets labeled as “DNS.” You can click on these packets to view their details.

Answer the following question
Question DNS-1: Check UDP details

- In the packet details, look for the "User Datagram Protocol" section. Here, you'll see the source and destination ports (typically source port is random, and destination is 53)
- Observe the "Length" field to understand the size of the UDP packet

Question DNS-2: Examine DNS queries

- Select a DNS query packet and expand the details in the packet details pane. Look for section labels "Domain Name" and "Transaction ID" (used to match queries and responses).

Question DNS-3: Examine DNS response codes

- Find corresponding DNS response packets with the same "Transaction ID"
- Within the DNS response packets, look for the "Flags" field.

Question DNS-4: Check for additional records:

- If applicable, analyze any additional records that may provide more context (e.g., NS records, MX records)

Question DNS-5: Identify DNS over UDP Characteristics

- Note that DNS queries and responses are typically small. Analyze how many packets were generated for a single query and the response time

2.3.1.2. Sniffing 2: packet UDP

Use the above DNS packet and answer the following questions
Question UDP-1: Examine packet details

- Check the sections for the Ethernet II, Internet Protocol (IP), and User Datagram Protocol (UDP).
- Take note of: Source port, Destination Port and Length

Question UDP-2: Analyze payload

- Examine the payload of the UDP packet. This may contain application-specific data, then expand to the UDP section to see any relevant information carried in the payload (for this example of DNS).

Question UDP-3: Analyze throughput and performance

- Look for patterns, such as consistent packet sizes or timing to assess the throughput of the UDP stream

Question UDP-4: Identify the packet loss

- Check for any noticeable gaps in the sequence of packets, which could indicate packet loss.

Question UDP-5: Examine flow

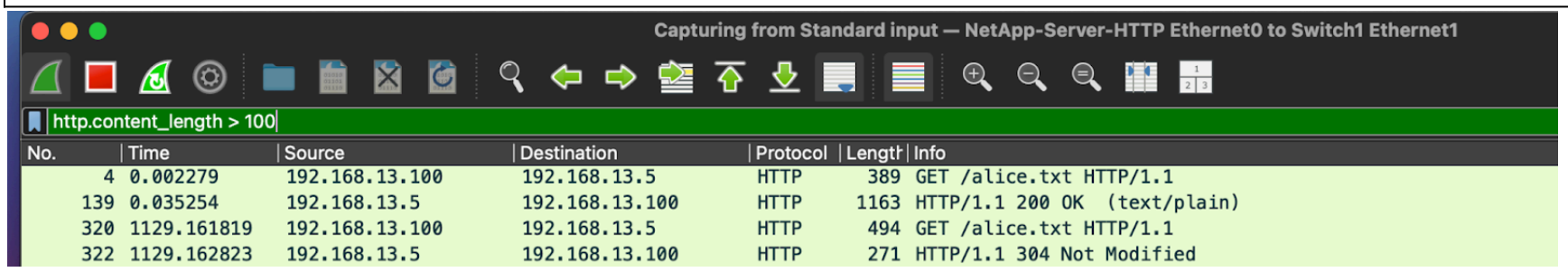
- Use **Statistics** > **Protocol Hierarchy** to see the breakdown of protocol

2.3.1.3. Sniffing 3: packet HTTP

Step 1: open sniffing packet tool on the network link of the NetApp-Server-HTTP.
Step 2: open web browser on the NetApp-Host and open the link http://www.bksysnet.edu/alice.txt
Step 3: Open wireshark capture window and locate HTTP packets: in the packet list pane, look for packets labeled as "HTTP." These will be your requests and responses.

Optional step: Filter for Long Packets to specifically find long HTTP packets, you can apply a display filter based on packet size. For example

```
http.content_length > 100
```



Use the capture HTTP package to alice.txt and answer the following questions.
Question HTTP-1: Examine HTTP responses: In the packet details pane, expand the HTTP section to review key fields:

- **Request Method**: GET, POST, etc
- **Status Code**: Indicates if the request was successful (e.g., 200 OK).
- **Content-Length**: Size of the response body.

Question HTTP-2: Analyze payload:

- Within the HTTP packet details, you can further inspect the payload: expand the "Data" section to see the actual content of the response.

Question HTTP-3: Check for compression

- Look for headers such as Content-type. If it says gzip or deflate, the content is compressed. If it says text/plain then its plain text.

Question HTTP-4: Check additional headers

- Review other HTTP headers (like Cache-Control, Expires, or ETag) that can provide.

Step 4: Refresh the web browser on NetApp-Host to examine conditional-GET
Question HTTP-5: Identify Conditional GET Requests: look for headers that indicate a conditional request:

- If-Modified-Since: This header tells the server to return the resource only if it has been modified since the specified date.
- If-None-Match: This header checks against the resource's ETag (a unique identifier for a specific version of the resource).

Question HTTP-6: Examine the Server Response: Find the corresponding HTTP response packet. Then, check the status code:

- 304 Not Modified: Indicates the resource hasn't changed, and the cached version can be used.
- 200 OK: Indicates the resource has changed, and the server provides the updated content.

Question HTTP-7: Check for Cache Behavior:

- Analyze how caching headers (Cache-Control, Expires) influence the conditional GET requests. This can help you understand the caching strategy employed in the server and the browser.

Question HTTP-8: Evaluate the response time

- Note the timestamps for both the request and response packets to evaluate the latency of the conditional GET request.
- Compare the times for a normal GET request versus a conditional GET request.

Question HTTP-9: Monitor bandwidth usage

- Observe the amount of data transferred in response to conditional GET requests versus regular GET requests.

2.3.1.4. Sniffing 4: packet TCP

Use the above DNS packet and answer the following questions

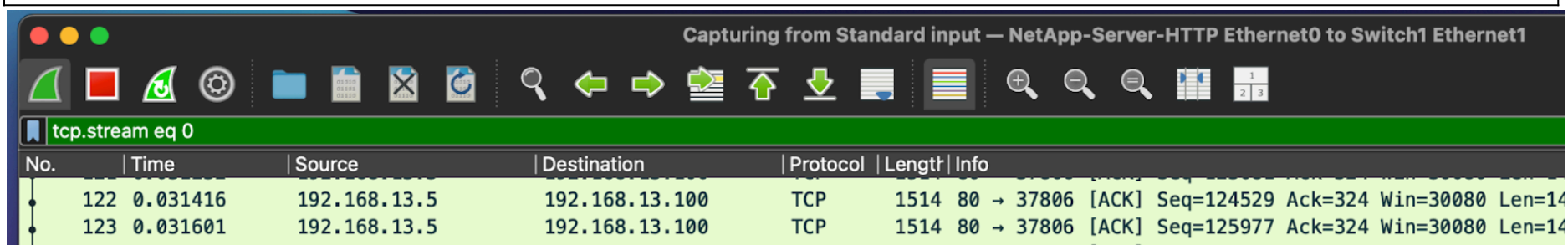
Question TCP-1: Identify fragmented packets

- Look for the "TCP" section. Check the section "Sequence Number" and "Acknowledgment Number"
- The resemble up to 103 TCP segment

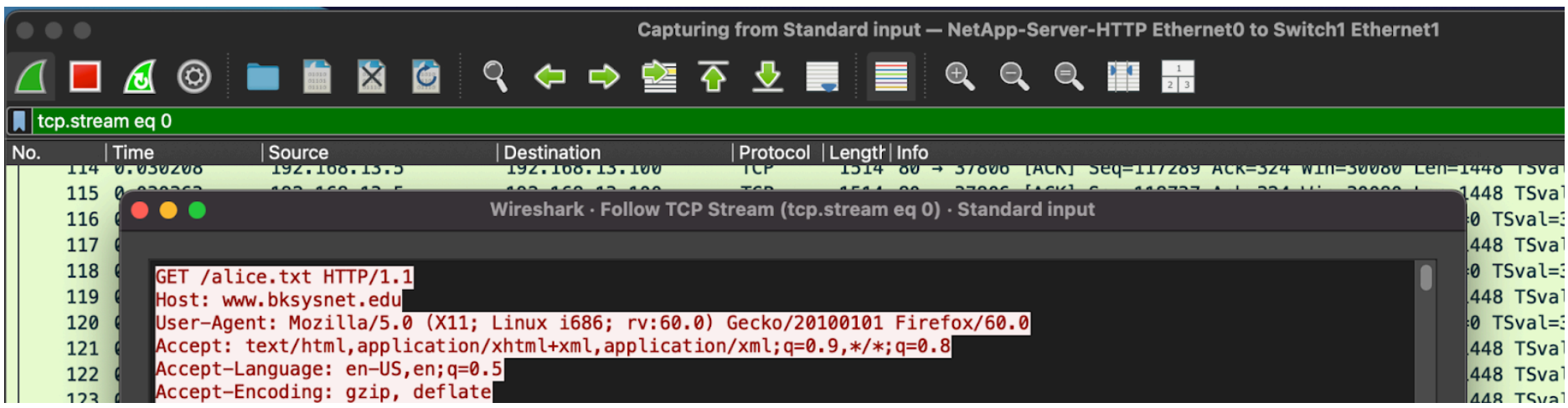
Question TCP-2: Check for TCP Segmentation

- You can filter the packets using a specific TCP connection:

tcp.stream eq [stream_number]



- Use the "Follow TCP Stream" feature (right-click on a TCP packet) to view the entire conversation



Question TCP-3: Inspect TCP Flags

- In each fragmented TCP packet, check for flags such as **SYN**, **ACK**, and **FIN**. Ensure that the connection establishment and termination processes are properly observed.

Question TCP-6: Check for Retransmissions

- Look for retransmitted packets, which may indicate issues with packet loss due to fragmentation

Question TCP-7: Evaluate Performance:

- Check the round-trip time (RTT) for the packets to identify any delays caused by fragmentation

Question TCP-8: Analyze Throughput:

- Calculate the throughput by measuring the amount of data sent over time

2.3.2. [NetW2] (Optional) Add router gateway to subnet

In this section, due to licensing restrictions of the router binaries, we present in this section an illustration employing Cisco Packet Tracer to utilize Cisco routers officially. The configuration steps are applicable on other platforms with custom router binary images.

In this section, we add a router gateway as in Figure 2.2 The schematic of local subnet with gateway router

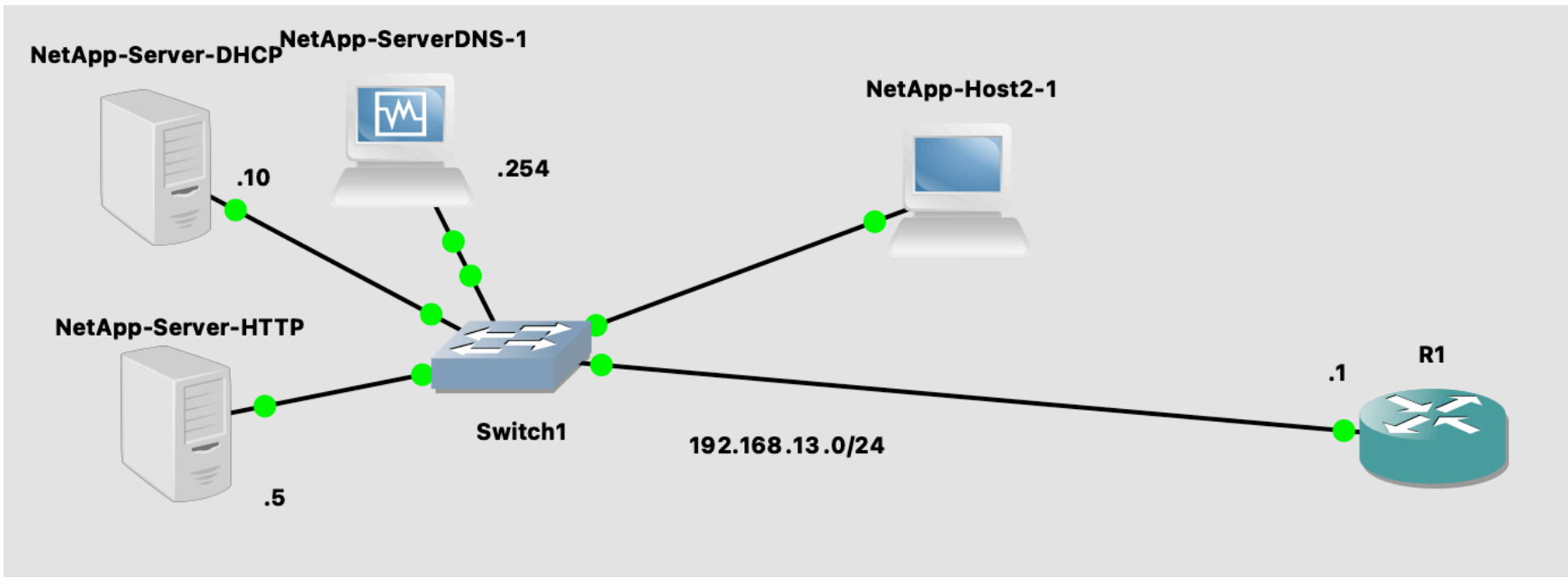


Figure 2.2 The schematic of local subnet with gateway router

Set up the IP address for the gateway router. The interface name need to be obtained from your device settings by listing “show interfaces”

```
R1#en
R1#show interfaces
……..
……..
R1#configure terminal
R1(config)#interface Ethernet 0/0
R1(config-if)#ip address 192.168.13.1 255.255.255.0
R1(config-if)#no shutdown
R1(config-if)#exit
R1#wr
Warning: Attempting to overwrite an NVRAM configuration previously written
by a different version of the system image.
Overwrite the previous NVRAM configuration?[confirm]
*Mar  1 00:03:42.592: %SYS-5-CONFIG_I: Configured from console by console
[confirm]
```

After that, all the host in subnet can set the default gateway to the router interface IP 192.168.13.1

Appendix Name server lookup

`nslookup` (Name Server Lookup) is a command-line tool used for querying the Domain Name System (DNS) to obtain domain name or IP address mapping, or other DNS records.

Basic commands

- Query a domain name

```
nslookup example.com
```

- Query an IP address:

```
nslookup 1.1.1.1
```

- Specifying a DNS server by adding the server address

```
nslookup example.com 8.8.8.8
```

- Getting different record types, you can query different type of DNS record such as NX (mail exchange), TXT (text) or NS (Name Server) (not support on NetApp-Host, student can try on physical PC)

```
nslookup -query=mx example.com
nslookup -query=txt example.com
nslookup -query=ns example.com
```