5. Project 3. To the Internet

Important

Please read the following instructions carefully:

1. This project is to be completed by each group individually.
2. This project contains 12 tasks. 8 of them are optional (10 points + 6 points).
3. The task score does not reflect the implementation difficulty but rather the recommended level.
4. Suggested workload is 4~8 FULL days. Manage your time.
5. Submit your code through Blackboard. The submission due date is **Dec. 8, 2024**.
6. Each group needs to submit the code once and only once. Immediately after TAs' checking. The submission is performed by one of the group members.
7. Tasks with "Optional" tag are optional tasks. Their due date is the end of the semester.
8. Unless otherwise mentioned, the instructor is responsible for grading optional tasks. Contact the instructor to check if you have finished one or more of them.
9. *Task 4, 5, 7, 8, and 9* are graded by TAs.
10. Tasks are graded according to their hierarchy. The hierarchy of this project is: *Task 1* < *Task 7* < *Task 10*, *Task 2* < *Task 8*, and *Task 3* < *Task 9*. A full score of one task automatically guarantees the full score of non-overdue preceding tasks (left side of "<").
11. Free to use any tools for debugging, but only the provided toolkit is permitted in performance assessment.
12. During the performance assessment, any task can only be attempted up to 5 times.

5.1. Overview

The first two projects built a local area network using audio signals. Allow me humbly name the network Aethernet. Aether- and Ether- share the same meaning, and the term A(coustic) aligns well with its physical medium. The goal of this project is to interconnect the Aethernet network with operational networks, and this inevitably relies on the IP (Internet Protocol, RFC 791).

IP plays a key role in the Internet, serving as the foundation for interconnecting networks. As depicted in Figure 5.1, network nodes with IP support can be categorized into two types. The router nodes, equipped with multiple network interfaces, are responsible for packet forwarding and interconnecting networks. The remaining network nodes, excluding routers, are termed as host nodes.

The following tasks will implement IP among Aethernet nodes. The optional tasks include implementing an Aethernet driver, which will provides a universal network interface to the operating system and applications.

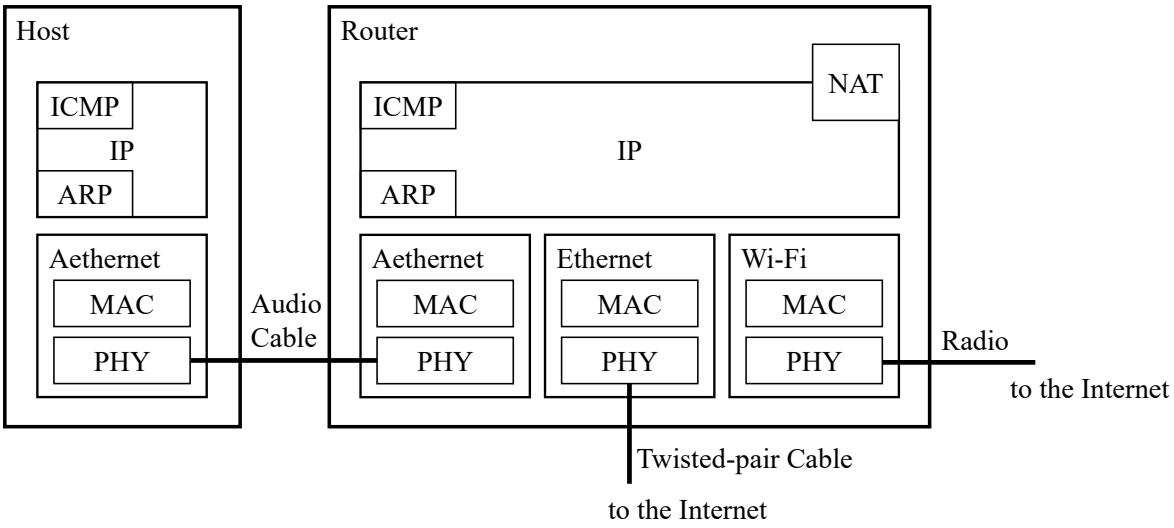

Figure 5.1. Project 3 Overview.

5.2. Task 0: (0 point) Sending and Receiving IP Datagram

Subsequent project tasks will frequently involve packet construction, sending, receiving, and parsing. These operations are usually done with the socket interface. The textbook introduces basic socket programming. Please first practice sending and receiving packets between two nodes over a regular network using socket.

In addition to IP, sockets also support TCP/UDP protocols. Sending and receiving IP packets of arbitrary content requires explicitly specifying the `raw socket` type. Due to some subtle issues of raw socket in, e.g., winsock, and java, we also recommend using a more universal but slightly more cumbersome low-level alternative: libpcap. In Figure 5.2, libpcap can capture and manipulate the PHY layer payload at will.

For guidance on utilizing libpcap, refer to the API documentation and code examples (C, java, npcap). For debugging packet encapsulation and network protocols, Wireshark is an essential tool. It can capture and analyze the network traffic of the specified network interface.

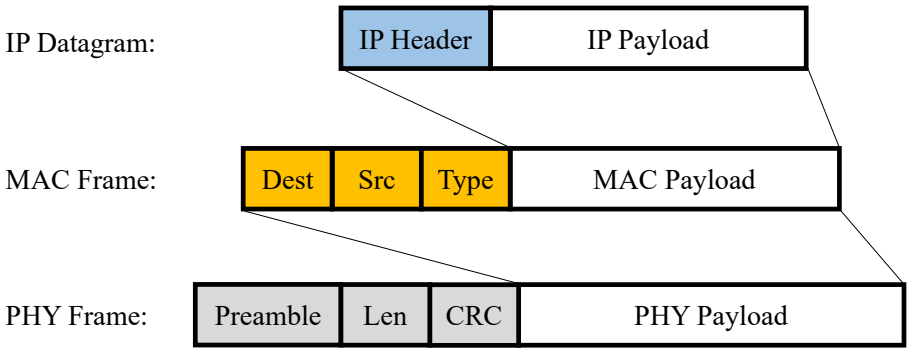

Figure 5.2. Example: IP Layer Packet Structure.

Tip

- libpcap uses Ethernet as the reference to encapsulate the Layer-2 frame. However, it cannot enforce the actual network device to perform transmission as such. To use a customized Aethernet frame, you can replace the Ethernet header with the Aethernet

header. But when analyzing packets with Wireshark, you have to revert the Ethernet header back because Wireshark's built-in parsing templates do not include user-defined frame structures.

- On Windows, the libpcap implementation is provided by Npcap. When installing Npcap (or Wireshark), please select the "WinPcap Compatible Mode."

Performance Assessment

The group provides two devices: `NODE1` and `NODE2`. They are connected through the campus LAN. Use libpcap or socket to send an IPv4 packet from `NODE1` to `NODE2`, and capture the packet on `NODE2` using Wireshark. Please complete this part on your own.

5.3. Task 1: (3 points) ICMP Echo

ICMP (Internet Control Message Protocol, RFC 792) is an network control and diagnostic protocol. ICMP is encapsulated within IP datagram, but it is generally considered as a part of IP. All IP hosts have to implement ICMP (RFC 1122), and routers are required to support ICMP Echo.

ICMP Echo and Echo Reply messages are commonly used for testing network reachability and latency. An IP node receiving an ICMP Echo message must respond with an Echo Reply message. The Echo Reply message swaps the source and destination addresses of the Echo message while keeping the payload and ID (Identifier) fields unchanged. The ID field is utilized to pair Echo and Echo Reply messages. The built-in `ping` tool in operating systems is implemented based on ICMP Echo.

This task and subsequent tasks will implement and utilize ICMP to validate the implementation of the IP protocol. The network topology for this project is shown in Figure 5.3. While the shown network addresses are recommended for a smoother assessment, they should adhere to the actual DHCP assignments.

In Figure 5.3, `NODE1` and `NODE2` should be connected by audio cables. Different network interfaces of `NODE2` are distinguished by the numbers with a "_" prefix. `NODE3` can be a smartphone. `NODE4` should be a fully functional PC connected to the campus LAN. It could be a laptop, a single-board computer, or a server that allows remote login. TAs will provide a laptop as `NODE4` during the performance assessment.

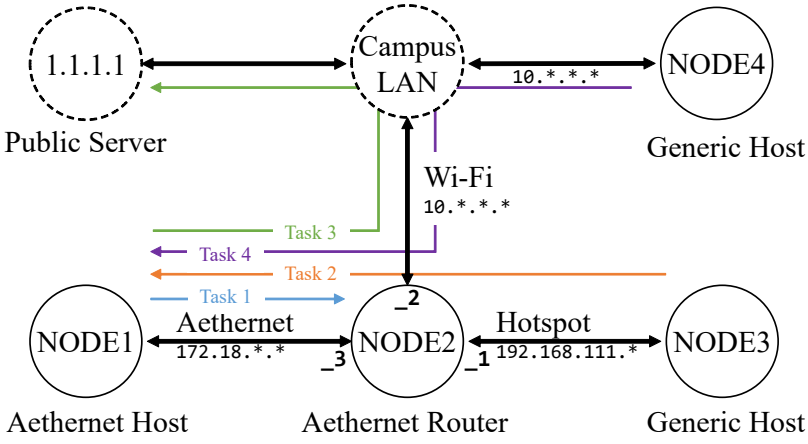

Figure 5.3. Connecting Aethernet Nodes to the Internet.

This task will enable IP packets sent by `NODE1` to reach `NODE2`:

- Based on the outcome of Project 2, assign a subnet range and LAN IP addresses to the Aethernet programs of `NODE1` and `NODE2`.
- Bind the IP and Aethernet MAC addresses and share this binding information between `NODE1` and `NODE2`. That is, implementing ARP is not required except for Task 10.
- Incorporate the tools introduced in Task 0 into the Aethernet program to send and receive IP packets through the audio interface.
- Enable ICMP Echo between `NODE1` and `NODE2`.

Tip

- When conducting ICMP tests, the firewall of the host should permit ICMP traffic.

Performance Assessment

The group provides `NODE1` and `NODE2`, and connects them with the toolkit according to Figure 4.2.

In the Aethernet program of `NODE1`, enter the following content:

```
#NODE1
(Aethernet program) ping NODE2_3.IP -n 10
```

-n (Windows) or -c (Linux) specifies the number of ICMP Echos.

The output format should resemble typical `ping` tools and should include the IP addresses of the ICMP peers and the Round-Trip Time (RTT) measured by ICMP Echo and Echo Reply. TAs record the average of the three minimum RTTs as RTT_12.

The RTT between `NODE1` and `NODE2` must be within 300 ms:

RTT_12	Percentage Earned
<200 ms	100%
<300 ms	80%
>300 ms	0%

5.4. Task 2: (4 points) Router

A router is a multi-port node in IP networks, with each port assigned an IP address. Routers are responsible for forwarding IP datagrams that are not destined for them. The forwarding decision is based on the headers (and payload) of the IP datagram and the router's routing table. The routing table is determined by routing protocols or static configuration.

This task will enhance the network functionalities of `NODE2`, providing it with simple static routing capabilities. Through `NODE2`, IP datagrams from `NODE1` and `NODE3` should be able to reach each other:

- Based on Task 1, enable the Wi-Fi hotspot on `NODE2` and allow `NODE3` to connect to it as a Wi-Fi client.
- (Optional) Disconnect `NODE2` from the campus LAN.

- Implement proper forwarding logic on `NODE2`:
    - If the destination IP address of the datagram belongs to the directly connected networks, forward it to the corresponding port.
    - Otherwise, discard it.
- Enable ICMP Echo from `NODE3` to `NODE1`.

Tip

- When using a smartphone as `NODE3`, use ping apps.

Performance Assessment

The group provides `NODE1` and `NODE2`, and connects them with the toolkit according to Figure 4.2. The group provides `NODE3`, and connects it to the Wi-Fi hotspot of `NODE2`.

In `NODE3`'s system terminal or ping app, enter the following (or equivalent) content:

```
# NODE3
ping NODE1.IP -n 10
```

TAs record the average of the three minimum RTTs as RTT_13.

RTT_13 must be within 350 ms:

RTT_13	Percentage Earned
<250 ms	100%
<350 ms	80%
>350 ms	0%

5.5. Task 3: (3 points) NAT

To enable certain network functions, routers may need to modify the content of the IP header and the payload. NAT (Network Address Translator, RFC 1631) utilizes TCP/UDP ports to reuse public IPv4 addresses and is prevalent on today's Internet.

This task will implement an ICMP-compatible NAT on `NODE2` to allow ICMP Echos from `NODE1` to reach the Cloudflare public network server 1.1.1.1:

- Based on Task 2, connect `NODE2` to the campus LAN, and verify the reachability from `NODE2` to 1.1.1.1 (typical RTT is 70 ms).
- Implement NAT on `NODE2`. While ICMP does not have port fields like TCP/UDP, its ID field can serve the same purpose. Specifically, `NODE2` should record the IDs of ICMP Echo messages. Upon receiving the ICMP Echo Reply messages, use their IDs to determine the forwarding actions:
    - If the destination IP address of the datagram belongs to the directly connected networks, forward it to the corresponding port.
    - If the destination IP address of the datagram is `NODE2_2.IP`, modify the destination IP address according to the NAT table, and

forward it to the corresponding port.

    - If the destination IP address of the datagram does not belong to the directly connected networks, modify its source IP address to `NODE2_2.IP` and forward it to the default router connected to port `NODE2_2`.

        - If the datagram is an ICMP Echo message, add its ICMP ID and the source IP address to the NAT table.

- Analyze ICMP Echo messages from `NODE3` to 1.1.1.1.
- Enable ICMP Echo from `NODE1` to 1.1.1.1.

Performance Assessment

The group provides `NODE1` and `NODE2`, and connects them with the toolkit according to Figure 4.2. The group provides `NODE3`, and connects it to the Wi-Fi hotspot of `NODE2`.

In `NODE3`'s system terminal or `ping` app, enter the following (or equivalent) content:

```
# NODE3
ping 1.1.1.1 -n 1000
```

then, in the Aethernet program of `NODE1`, enter the following content:

```
# NODE1
(Aethernet program) ping 1.1.1.1 -n 10
```

TAs record the average of the three minimum RTTs as RTT_3S and RTT_1S, correspondingly.

RTT_1S - RTT_3S must be within 300 ms:

RTT_1S-RTT_3S	Percentage Earned
<200 ms	100%
<300 ms	80%
>300 ms	0%

At the same time, RTT_3S must be within 300 ms:

RTT_3S	Percentage Earned
<300 ms	100%
>300 ms	0%

5.6. Task 4: (Optional, 1 point) NAT Traversal

The traversal problem is to allow a local IP address to be addressed from external networks. This requires new forwarding rules in NAT:

- Based on Task 3, connect `NODE4` to the campus LAN and verify the reachability from `NODE4` to `NODE2_2.IP`.

- One possible approach for ICMP traversal is to use the payload field of ICMP Echo to index the local IP address. By default, the payload is filled with random or ordered chars, but it can be explicitly specified using `$ ping -p` (Linux only) or other ping-like tools, e.g., nping `$ nping --data`.
- Enable ICMP Echo from `NODE4` to `NODE3`.
- Enable ICMP Echo from `NODE4` to `NODE1`.

Tip

- ICMP Echo messages from `NODE4` can trigger replies from `NODE2_2`. Disable ICMP reply on `NODE2` in this task.

Performance Assessment

The group provides `NODE1` and `NODE2`, and connects them with the toolkit according to Figure 4.2. The group provides `NODE3`, and connects it to the Wi-Fi hotspot of `NODE2`.

TAs provide a Linux laptop as `NODE4`, and connect it to the campus LAN.

The group provides `[group-specified options]` for `ping`.

In `NODE4`'s system terminal, enter the following content to address `NODE3`:

```
# NODE4
ping NODE2_2.IP -n 1000 [group-specified options]
```

TAs record the average of the three minimum RTTs as RTT_43.

In another `NODE4`'s system terminal, enter the following content to address `NODE1`:

```
# NODE4
ping NODE2_2.IP -n 10 [group-specified options]
```

TAs record the average of the three minimum RTTs as RTT_41.

RTT_41 - RTT_43 must be within 300 ms:

RTT_41 - RTT_43	Percentage Earned
<200 ms	100%
<300 ms	80%
>300 ms	0%

At the same time, RTT_41 must be larger than RTT_43:

RTT_41 - RTT_43	Percentage Earned
>20 ms	100%
<20 ms	0%

5.7. Task 5: (Optional, 1 point) IP Fragmentation

The MTU (Maximum Transmission Unit) size of Aethernet is likely to be inconsistent with (smaller than) that of other networks. IP supports packet fragmentation and reassembling at the destination. This task uses longer ICMP messages to verify this feature.

Tip

- Some virtual gateways (for virtual machines) do not support fragmented ICMP messages. Use a physical machine to avoid this issue.

Performance Assessment

The group provides `NODE1` and `NODE2`, and connects them with the toolkit according to Figure 4.2. The group provides `NODE3`, and connects it to the Wi-Fi hotspot of `NODE2`.

TAs provide a Linux laptop as `NODE4`, and connect it to the campus LAN.

The group provides `[group-specified options]` for `ping`.

In `NODE4`'s system terminal, enter the following content to address `NODE1`:

```
# NODE4
ping NODE2_2.IP -n 10 -l 200 [group-specified options]
```

-l specifies the size of the ICMP Echo payload (in bytes). 200 is an estimated value. The group can explicitly inform TAs about their Aethernet MTU size. TAs should choose a payload size that is at least 3 times the informed MTU size. This size should also be smaller than the Ethernet MTU of 1500 bytes. This is because if the ICMP Echo from `NODE4` is already fragmented before reaching `NODE2`, it will complicate the NAT implementation.

TAs do not have to record RTT. They use Wireshark on `NODE4` to check whether the ICMP Echo Reply messages are fragmented and correctly reassembled.

5.8. Task 6: (Optional, 0 point) Virtual Network Device

Computers use network devices for network transmissions. As depicted on the left side of Figure 5.4, typical network devices like Ethernet devices implement PHY and a part of MAC in hardware (network cards). They provide software interfaces to the operating system through their device drivers. The operating system aggregates network traffic of different network devices in the TCP/IP protocol stack, and then provides packet sending and receiving interfaces to applications through socket.

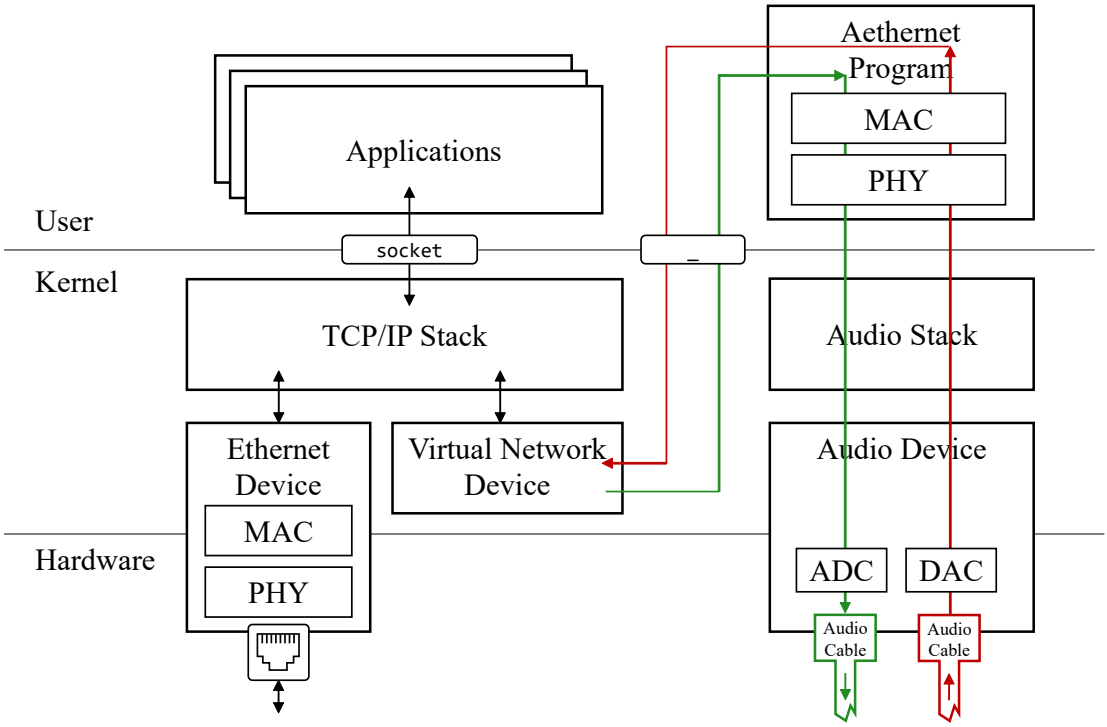

Figure 5.4. Aethernet Network Device.

Aethernet's PHY and MAC programs have frame reception and transmission functionalities over audio devices, but these functions can only be invoked within the Aethernet program. This means that Aethernet cannot be recognized and used by other programs. To this end, this task will create a standard network device for Aethernet to integrate it into operating systems.

As shown on the right side of Figure 5.4, a strategy is to make use of the virtual network device. A virtual network device does not include hardware; it is simply a software driver that acts as a standard network device driver. Its output can be directed to user-space programs, and its input can come from user-space programs. The virtual network device driver can be used for debugging purposes, but its most impactful applications is in various network tunnels. Specifically, packets passing through the virtual network device can be directed to user-space programs, e.g., VPN (Virtual Private Network), for encapsulation and sending through other suitable channels, such as a whitelisted network server, Minecraft [SP23], and, of course, sound waves.

These are relatively easy-to-use virtual network device drivers:

- Microsoft loopback adapter: This driver is a proprietary virtual network device driver in Windows. Its traffic can be accessed and manipulated through libpcap.
- TAP and TUN: They are both virtual network device drivers. TUN operates at Layer 3, while TAP operates at Layer 2. TAP is compatible with TUN, and TUN cannot be used to produce arbitrary Layer 2 frames. Linux comes with tuntap drivers, which can be accessible in user space through device files. tap-windows is a TAP driver for Windows. Its usage can be referenced from OpenVPN or the ping example by Thomas Watteyne. wintun is a TUN driver for Windows.

Redirecting the traffic of these virtual drivers to/from the Aethernet program essentially provides the operating system with a standard network interface, the Aethernet interface, for sending and receiving over audio signals.

Tip

- Deselect unnecessary protocol suites in the virtual network device to reduce background traffic.

Performance Assessment

The group provides two devices: `NODE1` and `NODE2`. They are connected through the campus LAN. Use socket and the crafted Aethernet Interface to send an IPv4 packet from `NODE1` to `NODE2`, and capture the packet on `NODE2` using Wireshark. Please complete this part on your own.

5.9. Task 7: (Optional, 1 point) ICMP Echo

Use Aethernet interface to complete Task 1.

Performance Assessment

Assessment criteria and procedures are the same as in Task 1, except:

In the system terminal of `NODE1`, enter the following content:

```
# NODE1
ping NODE2_3.IP -n 10
```

5.10. Task 8: (Optional, 1 point) Router

Use Aethernet interface to complete Task 2.

Performance Assessment

Assessment criteria and procedures are the same as in Task 2. Additionally, TAs use Wireshark to capture the Aethernet Interface to check if it receives ICMP Echo messages from `NODE3`.

5.11. Task 9: (Optional, 1 point) NAT

Use Aethernet interface to complete Task 3.

Performance Assessment

Assessment criteria and procedures are the same as in Task 3, except:

In the system terminal of `NODE1`, enter the following content:

```
# NODE1
ping 1.1.1.1 -n 10
```

5.12. Task 10: (Optional, 1 point) ARP

ARP (Address Resolution Protocol, RFC 826) allow auto-binding of IP addresses and MAC addresses. Use Aethernet interface to complete Task 1 without writing static ARP entries.

Performance Assessment

Assessment criteria and procedures are the same as in Task 1, except:

In the system terminal of `NODE1`, enter the following content:

```
# NODE1
ping ANY_IP_OF_THEIR_SUBNET -n 10
```

Use Wireshark to monitor ARP frames flowing in `NODE1`'s Aethernet Interface.

5.13. Task 11: (Optional, * points) Star

Every group has independently crafted a network known as Aethernet. While these networks may not share many similarities beyond the name, their IP-compatibility allows them to be interconnected. This task is designed to celebrate this exciting outcome. Please refer to Figure 5.5 to establish connections among groups. Configure the routing table and NAT on `NODE5` to allow all `NODE1*` to reach each other.

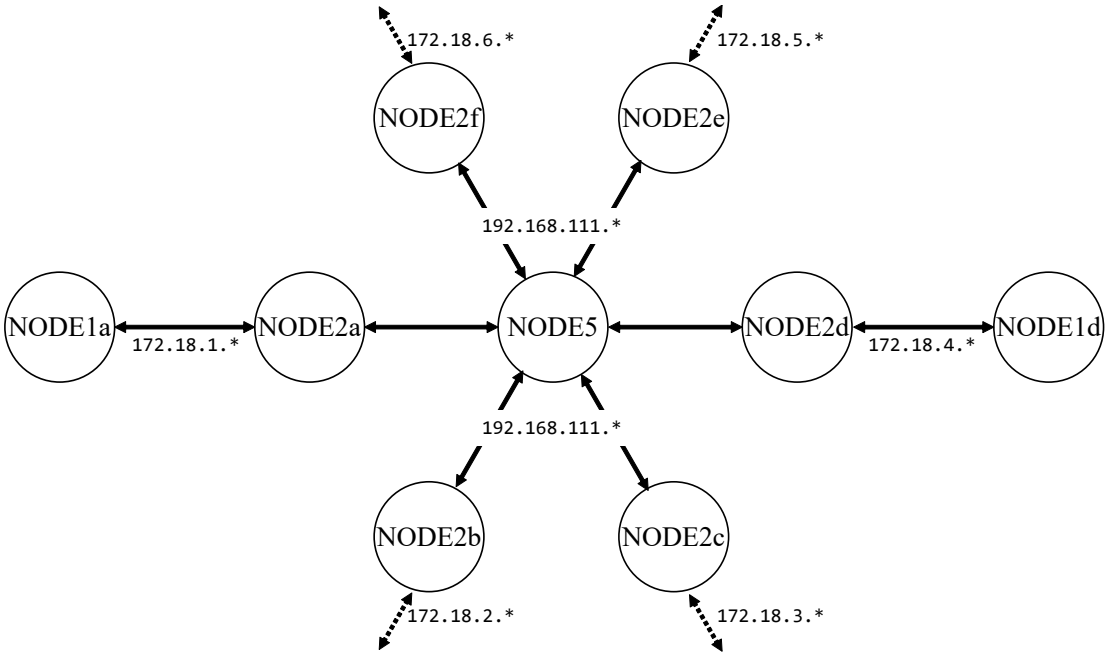

Figure 5.5. A Network of Aethernet Networks.

Performance Assessment

Each group provides two devices: `NODE1*` and `NODE2*`, and connects them with Aethernet. One group provides `NODE5`. All groups connect their `NODE2*` devices to the hotspot of `NODE5`.

In the system terminal of `NODE1*`, enter the following content without using `[group-specified options]`:

```
# NODE1x
ping NODE1y.IP -n 1000
```

where x and y are the IDs of the participating groups.

The earned rewards are 0.5*N, where N is the number of groups that can be interconnected.

References

[SP23] Telepath: A Minecraft-based Covert Communication System https://doi.org/10.1109/SP46215.2023.10179335