NASA Homework 0

林靖昀

Network Administration

1. Short Answer

- **P1**

  From layer 5 to 1:

  - **Application layer:**

    Provides an interface and protocols/services for applications to use.

    Example:

    A web browser and a web server communicating with the HTTP protocol.

  - **Transport layer:**

    Segments the data through two main protocols, TCP / UDP, provides multiplexing through ports and reliable data transfer(TCP).

    Example:

    HTTP using TCP's service to ensure a connection.

  - **Network layer:**

    Packs segments(or UDP datagrams) into IP datagrams, then routes the data to the destination.

    Example:

    TCP using IP to route to a destination.

  - **Data link layer:**

    Frames the data, and defines protocols on how data is sent from device to device.

    Example:

    Using ARP to discover the MAC address associated with the IP address.

  - **Physical layer:**

    In charge of turning digital data into a format that can be sent over different physical medias.

    Example:

    Data being translated into radio wave signals to be transferred over wifi.

  Reference:

  Computer Networking A Top-Down Approach

- **P2**
  i. VLANs are the logical separation of a physical network, allowing us to create virtual LANs. It abstracts out the physical layout of the network and allows us to group and design different virtual networks regardless of the underlying network layout.
  ii. **Switch:**
  Forwards data between the devices connected directly to it, enabling communication between those devices. A switch operates at layer 2 in the TCP/IP model. It provides direct link access with MAC addresses.
  Router:
  Forwards data between the networks connected to it, providing routing between different networks. With different protocols, routers route data over an optimal path over the internet. A router operates at layer 3 in the TCP/IP model. It provides inter-network communication while implementing routing.
  iii. **Broadcast storm:**
  A Broadcast storm is the accumulation of broadcast messages on a networks bandwidth,commonly caused by infinitely looping broadcast messages (from a switching loop.)
  Prevention:
    - Getting rid of switching loops via link aggregation or other techniques.
    - Segmenting the broadcast domain physically or logically (with VLANs).

  Switching loop:
  A switching loop happens when there exists more than 1 path between 2 switches, thus forming a loop, allowing data to loop indefinitely.
  Prevention:
    - Using Link aggregation.
    - Using the spanning tree protocol to build loop free logical networks.

  Broadcast storms and switching loops are similar in that they both stem from infinitely looping data occupying the network, broadcast storms are different from switching loops as they are caused specifically by broadcast messages.

Reference:
https://en.wikipedia.org/wiki/VLAN
https://en.wikipedia.org/wiki/Network_switch
https://en.wikipedia.org/wiki/Router_(computing)
https://en.wikipedia.org/wiki/Broadcast_storm
https://en.wikipedia.org/wiki/Switching_loop

- **P3**
  i. Because there are not enough IPv4 addresses.
  ii. Considering the fact that IPv6 contains 340,282,366,920,938,463,463,374,607,431,768,211,456 addresses (128 bit address space), giving: “3,911,873,538,269,506,102 addresses per square meter of the surface of the planet Earth”, it is highly unlikely that we would need to change protocols because of insufficient addresses. That being said, this does not mean that there won't be changes to our internet protocol stack, the architecture of the internet might change in the future bringing new protocols.
  iii. The main difference is the address space size, other changes where also made in IPv6, like replacing TTL with hop limit, and defining a fixed header size. In IPv6, only the sender can perform fragmentation, while in IPv4, nodes along the route can fragment the data if needed.
  iv. Since a the internet was initially implemented on IPv4, a lot of old hardware and software are not built to support IPv6, transitioning between the two cannot be done instantly and needs to be slowly phased in.

Reference:
https://www.ciscopress.com/articles/article.asp?p=2803866&seqNum=3
https://en.wikipedia.org/wiki/IPv6
https://en.wikipedia.org/wiki/IPv6_packet
https://en.wikipedia.org/wiki/IPv4
https://www.geeksforgeeks.org/differences-between-ipv4-and-ipv6/#difference-between-ipv4-and-ipv6

- **P4**
  - **UDP**
    UDP is connectionless protocol, it works by providing "best effort" service: sending data without establishing a connection before hand, allowing you to send data without needing to care if the other end is even available or not.
  - **TCP**
    TCP works by establishing a connection first, then sending data over, TCP provides reliable data transfer, deals with errors and packet loses, and provides congestion control.

  TCP and UDP both provide multiplexing through ports. TCP provide more services as listed above, while UDP provides bare bones best effort service.
  UDP is used when high latency cannot be tolerated, and reliable data transfer is not important (video streaming, online gaming, etc.)
  TCP is used when transmission errors cannot be tolerated, and latency is not that important (SMTP)
  Reference:
  Computer Networking A Top-Down Approach

- **P5**
  EFK is a software stack consisting of Elasticsearch, Fluentd, and Kibana. Fluentd collects and unifies log data to Elasticsearch, which is a search engine. Kibana is a data visualization frontend dashboard for Elasticsearch.
  EFK allows system admins to aggregate and process logging data of many distributed systems while also being scalable. This is suitable for our department, since we have many services running on multiple nodes, each having their own logs and data. The main disadvantages of EFK are the overhead it brings, and the large amount of storage that is required to store the logs.
  Reference:
  https://platform9.com/blog/logging-monitoring-of-kubernetes-applications-requirements-recommended-toolset/

- **P6**
  The multiplexing at layer 4 is the differentiation of different services on the same host, ports are used to multiplex and de-multiplex in this case.
  Generally in networking, multiplexing is resource sharing between multiple hosts, 3 common types of multiplexing are:
  i. Time-division multiplexing
  ii. Frequency-division multiplexing
  iii. Statistical multiplexing

  The wifi in our department building likely uses FDM (or some variant of it), since wifi is transmitted over radio waves.
  Reference:
  Computer Networking A Top-Down Approach

2. Command Line Utilities

- **P1**

i. traceroutev results:

```
traceroute speed.ntu.edu.tw
traceroute to speed.ntu.edu.tw (140.112.5.178), 30 hops max, 60 byte packets
 1  10.200.200.200 (10.200.200.200)  21.048 ms  21.002 ms  20.989 ms
 2  ip4-126.vpn.ntu.edu.tw (140.112.4.126)  21.188 ms  21.223 ms  21.239 ms
 3  140.112.5.178 (140.112.5.178)  21.268 ms !X  21.299 ms !X  21.313 ms !X
```

ii. ping results:

```
ping speed.ntu.edu.tw
PING speed.ntu.edu.tw (140.112.5.178) 56(84) bytes of data.
64 bytes from 140.112.5.178: icmp_seq=1 ttl=62 time=23.4 ms
64 bytes from 140.112.5.178: icmp_seq=2 ttl=62 time=17.9 ms
^C
--- speed.ntu.edu.tw ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 17.852/20.642/23.432/2.790 ms
```

nslookup results:

```
nslookup speed.ntu.edu.tw
;; Got recursion not available from 140.112.254.4, trying next server
;; Got recursion not available from 140.112.2.2
Server:         140.112.2.2
Address:        140.112.2.2#53

Name:   speed.ntu.edu.tw
Address: 140.112.5.178
```

iii. From the reserved ip address ranges for private networks:

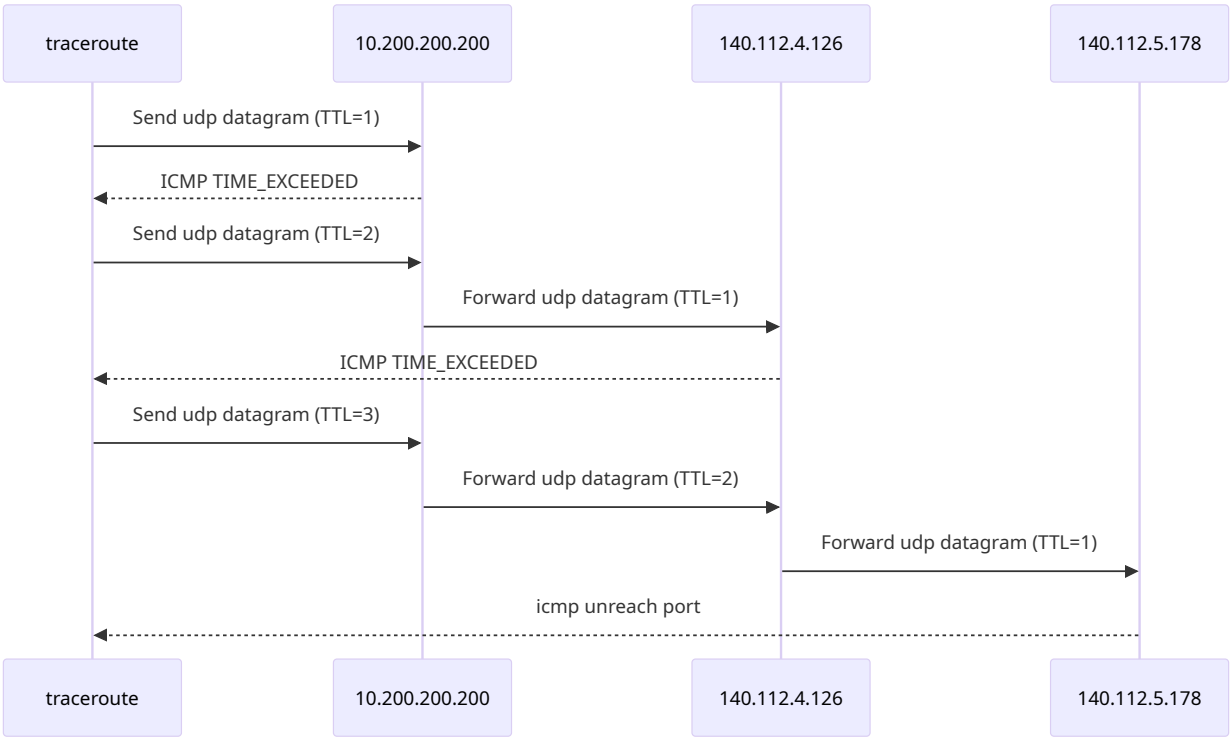
10.200.200.200 belongs to a private network.

140.112.4.126 and 140.112.5.178 belongs to the public network.

iv. traceroute works by sending probes with incrementing TTL, at each TTL, by default traceroute will send 3 probes, the 3 time values are the round trip time of each probe, if we change the amount of probes we send per TTL, the amount of time values we get will also change.

Later results might not always be larger, for example, if a later node has a better path back to us, the round trip time might add up to be smaller.

v. Sequence diagram:



Reference:

traceroute man page

- **P2**

  i. ping sends ICMP ECHO_REQUESTs.

  ping results:

```
•● ➜ ping 140.112.91.2
PING 140.112.91.2 (140.112.91.2) 56(84) bytes of data.
^C
--- 140.112.91.2 ping statistics ---
14 packets transmitted, 0 received, 100% packet loss, time 13164ms
```

  ii. nmap results:

```
•● ➜ nmap -sn 140.112.91.2
Starting Nmap 7.95 ( https://nmap.org ) at 2025-01-28 14:05 CST
Nmap scan report for nasa2023team02.csie.ntu.edu.tw (140.112.91.2)
Host is up (0.020s latency).
Nmap done: 1 IP address (1 host up) scanned in 4.06 seconds
```

  We see that ping is unable to get a response while nmap with the `-sn` option can, from nmaps man page:

  > The default host discovery done with -sn consists of an ICMP echo request, TCP SYN to port 443, TCP ACK to port 80, and an ICMP timestamp request by default. When executed by an unprivileged user, only SYN packets are sent (using a connect call) to ports 80 and 443 on the target.

  Using the `-vvv` option we see:

```
•● ➜ nmap -sn -vvv 140.112.91.2
Starting Nmap 7.95 ( https://nmap.org ) at 2025-01-28 14:19 CST
Initiating Ping Scan at 14:19
Scanning 140.112.91.2 [2 ports]
Completed Ping Scan at 14:19, 0.02s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 14:19
Completed Parallel DNS resolution of 1 host. at 14:19, 0.00s elapsed
DNS resolution of 1 IPs took 0.00s. Mode: Async [#: 1, OK: 1, NX: 0, DR:
 0, SF: 0, TR: 1, CN: 0]
Nmap scan report for nasa2023team02.csie.ntu.edu.tw (140.112.91.2)
Host is up, received conn-refused (0.023s latency).
Nmap done: 1 IP address (1 host up) scanned in 0.03 seconds
```

  We received a SYN-ACK, meaning the TCP SYN sent to port 80 / 443 gave us a response.

  iii. Scan reults:

```
•● ✖ nmap -p80 -sV 140.112.91.2
Starting Nmap 7.95 ( https://nmap.org ) at 2025-01-28 14:23 CST
Nmap scan report for nasa2023team02.csie.ntu.edu.tw (140.112.91.2)
Host is up (0.015s latency).

PORT   STATE SERVICE VERSION
80/tcp open  http    nginx 1.26.2

Service detection performed. Please report any incorrect results at http
s://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.45 seconds
```

  Service: http

  Version: nginx 1.26.2

  http is a request-response protocol used to transmit data over the web.

  Command: `nmap -p80 -sV 140.112.91.2`

  iv. POST response:

```
•● ✖ curl http://140.112.91.2/ -d "hi"
Great, meet me at somewhere between port 48000 and 49000%
```

  nmap scan results:

```
Starting Nmap 7.95 ( https://nmap.org ) at 2025-01-28 14:30 CST
Nmap scan report for nasa2023team02.csie.ntu.edu.tw (140.112.91.2)
Host is up (0.052s latency).
Not shown: 1000 closed tcp ports (conn-refused)
PORT      STATE SERVICE
48763/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 2.67 seconds
```

  netcat response:

```
•● ➜ nc 140.112.91.2 48763
welcome to nasa!!! (screenshot me as proof for your answer)%
```

Reference:

ping, nmap, and curl man pages.

- **P3**

  i. nslookup results:

```
•● ➔ nslookup Bocchi-Tracker.csie.ntu.edu.tw
Server:         192.168.50.1
Address:        192.168.50.1#53

Non-authoritative answer:
Name:   Bocchi-Tracker.csie.ntu.edu.tw
Address: 140.112.30.131
```

  IP: 140.112.30.131

  ii. nslookup results:

```
•● ➔ nslookup 140.112.30.131
131.30.112.140.in-addr.arpa     name = Starry.csie.ntu.edu.tw.

Authoritative answers can be found from:
```

  Domain name: Starry.csie.ntu.edu.tw

  iii. nslookup results:

```
•● ➔ nslookup -q=txt Starry.csie.ntu.edu.tw
Server:         192.168.50.1
Address:        192.168.50.1#53

Non-authoritative answer:
Starry.csie.ntu.edu.tw  text = "Your guitar is in the box"

Authoritative answers can be found from:
csie.ntu.edu.tw nameserver = ntuns.ntu.edu.tw.
csie.ntu.edu.tw nameserver = csman.csie.ntu.edu.tw.
csie.ntu.edu.tw nameserver = csman2.csie.ntu.edu.tw.
csman.csie.ntu.edu.tw   internet address = 140.112.30.13
csman2.csie.ntu.edu.tw  internet address = 140.112.30.14
```

  TXT: "Your guitar is in the box"

  iv. dig results:

```
•● ➔ dig "Bocchi.csie.ntu.edu.tw"

; <<>> DiG 9.20.4 <<>> Bocchi.csie.ntu.edu.tw
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 5209
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;Bocchi.csie.ntu.edu.tw.                IN      A

;; ANSWER SECTION:
Bocchi.csie.ntu.edu.tw. 600     IN      CNAME   Gu1tArHer0.csie.ntu.edu.
tw.

;; AUTHORITY SECTION:
csie.ntu.edu.tw.        3600    IN      SOA     csman.csie.ntu.edu.tw. t
a221.csie.ntu.edu.tw. 2020091805 3600 3600 604800 3600

;; Query time: 1510 msec
;; SERVER: 192.168.50.1#53(192.168.50.1) (UDP)
;; WHEN: Tue Jan 28 14:42:06 CST 2025
;; MSG SIZE  rcvd: 124
```

  CNAME: Gu1tArHer0.csie.ntu.edu.tw

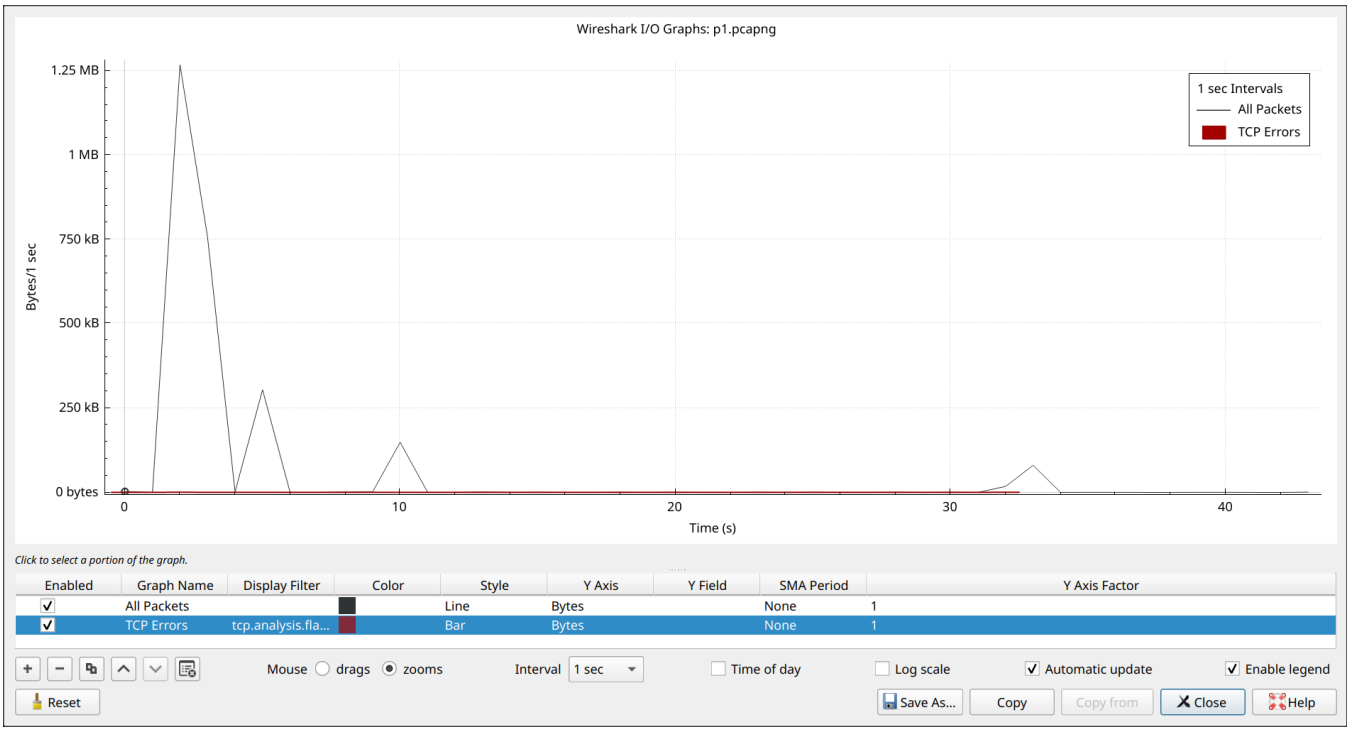
Reference:

nslookup & dig man pages

https://en.wikipedia.org/wiki/List_of_DNS_record_types

3. Basic Wireshark

- **P1**
  i. Port 3000
  ii. We see multiple ACKS from port 3000 after SYN was sent to it, we also see that port 3000 receives HTTP requests, and sends back HTTP responses.
  iii. I/O throughput graph:



  iv. Highest transmission speed: 1.268MB/s
  Time: 2.837 (second)
  v. 29, search with filter with 'http.request.method == "GET"'.
  vi. There is a POST request to "/dashboard/invoices/create"
  Data in customer ID field:33393538646339652d373132662d343337372d383565392d666563346236613634343261

Reference:
https://www.wireshark.org/docs/wsug_html_chunked/ChStatIOGraphs.html

- **P2**
  i. a. Go to Edit>Preferences.
  b. In the protocols drop down menu on the left, select TLS.
  c. Edit RSA keys list.
  d. Set IP to 127.0.0.1, port 443, and select private key file.
  ii. Packet number 32.
  a. Go to File>Export Objects>HTTP.
  b. Save image.
  Image:

Reference:
https://my.f5.com/manage/s/article/K19310681
https://osqa-ask.wireshark.org/questions/35123/fastest-way-to-display-a-png-file/

4. Cryptography

- **P1**
  i. **Flag:** NASA_HW0{1_10V3_r54}
  ii. Process:
    a. Generate key
    b. Receive encrypted
    c. Decrypt with CRT:

$$d_p = d \mod (p-1)$$
$$d_q = d \mod (q-1)$$
$$q_{inv} = q^{-1} \mod p$$
$$m_1 = c^{d_p} \mod p$$
$$m_2 = c^{d_q} \mod q$$
$$h = (q_{inv}(m_1 - m_2)) \mod p$$
$$m = m_2 + h \times q$$

    d. Convert decrypted int into bytes then into text.

Python code:

```python
from pwn import *
from Crypto.PublicKey import RSA

key = RSA.generate(4096)
target = remote("140.112.91.1", 48763)

print(target.recvuntil("n: "))
target.sendline(str(key.n))

print(target.recvuntil("e: "))
target.sendline(str(key.e))

print(target.recvline())
print(target.recvuntil(": "))
secret = int(target.recvall()[:-1])
secret = int(secret)

dp = key.d % (key.p - 1)
dq = key.d % (key.q - 1)
qinv = pow(key.q, -1, key.p)
m1 = pow(secret, dp, key.p)
m2 = pow(secret, dq, key.q)

h = (qinv*(m1 - m2)) % key.p
m = m2 + h * key.q

m = m.to_bytes(m.bit_length() // 8 + 1).decode()
print(m)
```

Reference:

https://en.wikipedia.org/wiki/RSA_(cryptosystem)
https://guyinatuxedo.github.io/02-intro_tooling/pwntools/index.html
https://pycryptodome.readthedocs.io/en/latest/src/public_key/rsa.html

5. 為什麼簽不了憑證？？？

- **P1**
  A subnet is a logical division of an IP network, usually represented by partitioning the ip address into a subnet id and a host id.
  Reference:
  https://en.wikipedia.org/wiki/Subnet
- **P2**
  Gateways are software or hardware that support multiple protocols in order to connect different networks together, they can operate at all 7 layers of the OSI model.
  Reference:
  https://en.wikipedia.org/wiki/Gateway_(telecommunications)
- **P3**
  X -> WAN -> A1
  Reference:
  https://en.wikipedia.org/wiki/Default_gateway
- **P4**
  If machine A has a route specification that points to X through A1:
  A1 -> WAN -> X
  Otherwise:
  A2 -> LAN -> B2 -> WAN -> X
  Reference:
  https://en.wikipedia.org/wiki/Default_gateway
- **P5**
  Stateful firewalls monitor packets by tracking the state of incoming network connections, while stateless firewalls monitor each package individually.
  Stateful firewalls are more likely to block a TCP ACK without a SYN.
  Reference:
  https://en.wikipedia.org/wiki/Stateful_firewall
  https://www.checkpoint.com/cyber-hub/network-security/what-is-firewall/what-is-a-stateless-firewall/
- **P6**
  It seems that during the TCP handshake in step 2, X sends SYN which arrives at A1, A sends (SYN, ACK), which goes through the default gateway, since B is a stateful firewall, it blocks A's out going (SYN, ACK), as it hasn't seen an ACK beforehand. This means X never receives the (SYN, ACK) no matter how many times it tries, and thus cannot establish a TCP connection.
- **P7**
  i. System shutdown: 5 ~ 15 minutes depending on services and active connections.
  ii. Certificate Update: 30 ~ 60 minutes, assuming we are talking about domain validation certificates, which are usually automatically validated.
  iii. System restart and testing: 1 ~ 2 hours, to ensure all services are functional.
  iv. Grace period/overflow time: 1 ~ 2 hours, For when something goes wrong and we need more time to identify, fix or reschedule another system maintenance.
  In total 6 hours.
- **P8**
  Remove A's A1 interface, and have it accept all data from its default gateway.
  Add a route entry to A's route table that directly connects A1 to X through the WAN.
- **P9**
  Yes :3

System Administration

6. btw I use arch

- **P0**

  My installation of Arch was done in a proxmox VM, pre-installation steps:

  i. Set boot mode to OVMF (UEFI) in VM creation configuration.

  ii. Enter firmware settings upon first boot.

  iii. Disable secure boot.

  Arch installation steps:

  i. Set the console keyboard layout and font

  a. Use default keymap layout (US).

  b. `setfont ter-132b` to increase font size.

  ii. Verify the boot mode: `cat /sys/firmware/efi/fw_platform_size` .

  iii. Verify internet connection: `ip link` .

  iv. Update system clock.

  a. `timedatectl` shows that timezone is set to UTC +0000

  b. Set time zone: `timedatectl set-timezone Asia/Taipei`

  v. Partion disk.

  a. `fdisk /dev/sda` .

  b. Create partition table `g` .

  c. Create partitions: `n` , default, default, `+Size` .

  d. Change partition types: `t` , type.

  e. Check changes: `p` .

```
Command (m for help): p
Disk /dev/sda: 22 GiB, 23622320128 bytes, 46137344 sectors
Disk model: QEMU HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 47F46C69-40A7-4081-96FA-6BE62C65EC1A

Device        Start      End  Sectors  Size Type
/dev/sda1      2048 31459327 31457280   15G Linux filesystem
/dev/sda2  31459328 41945087 10485760    5G Linux filesystem
/dev/sda3  41945088 44042239  2097152    1G EFI System
/dev/sda4  44042240 46135295  2093056 1022M Linux swap

Command (m for help):
```

  f. Write changes: `w` .

  vi. Format partitions:

  Root partition: `mkfs.ext4 /dev/sda1` .

  Home partition: `mkfs.ext4 /dev/sda2` .

  ESP: `mkfs.fat -F 32 /dev/sda3` .

  Swap partition: `mkswap /dev/sda4` .

  vii. Mount partitions:

  Root partition: `mount /dev/sda1 /mnt` .

  Home partition: `mount --mkdir /dev/sda2 /mnt/home` .

  ESP: `mount --mkdir /dev/sda3 /mnt/boot` .

  Swap partition: `swapon /dev/sda4` .

  viii. Install essential packages:

  `pacstrap -K /mnt base linux linux-firmware networkmanager vim man-db man-pages texinfo` .

  ix. Generate Fstab: `genfstab -U /mnt >> /mnt/etc/fstab` .

  x. Chroot: `arch-chroot /mnt` .

xi. Time:
    a. Set timezone: `ln -sf /usr/share/zoneinfo/Asia/Taipei /etc/localtime`.
    b. generate `/etc/adjtime`: `hwclock --systohc`.
xii. Localization:
    a. Uncomment `en_US.UTF-8 UTF-8` in `/etc/locale.gen` and run `locale-gen`.
    b. Create `locale.conf` with `LANG=en_US.UTF-8` inside.
    c. Use default us keymap.
xiii. Create and change hostname file: `/etc/hostname`.
xiv. Set root password: `passwd`.
xv. Install GRUB:
    a. Install packages: `pacman -S grub efibootmgr`
    b. Grub-install: `grub-install --target=x86_64-efi --efi-directory=/boot --bootloader-id=GRUB`
    c. Generate main configuration file: `grub-mkconfig -o /boot/grub/grub.cfg`
xvi. Reboot:
    a. `exit` chroot.
    b. Unmount: `umount -R /mnt`
    c. `reboot`
xvii. Network setup:
    Start `NetworkManager`: `systemctl enable NetworkManager`, `systemctl start NetworkManager`.
    Add dns server: `resolvectl dns <interface> <dns-server>`.
    Start `systemd-resolved`: `systemctl enable systemd-resolved`, `systemctl start systemd-resolved`.
xviii. Change font:
    a. Install `terminus-font`: `pacman -S terminus-font`.
    b. Edit `/etc/vconsole.conf`: Add `FONT=ter-132b`
    c. Reload: `systemctl restart systemd-vconsole-setup`.
xix. Add user: `useradd -m nasa`, and set password: `passwd nasa`

Reference:

https://wiki.archlinux.org/title/Installation_guide
https://wiki.archlinux.org/title/System_time
https://wiki.archlinux.org/title/Fdisk
https://man.archlinux.org/man/vconsole.conf.5.en
https://wiki.archlinux.org/title/Linux_console#Fonts
https://wiki.archlinux.org/title/Users_and_groups#Example_adding_a_user

- **P1**

  Hostname can be changed via: `hostnamectl set-hostname <hostname>`

```
[root@arch ~]# cat /etc/hostname
b12902116
```

- **P2**

```
[root@arch ~]# lsblk -f
NAME   FSTYPE  FSVER           LABEL       UUID                                 FSAVAIL FSUSE% MOUNTPOINTS
sda
├─sda1 ext4    1.0                         de37132a-362f-4326-b994-85ca00e368ea   11.9G    14% /
├─sda2 ext4    1.0                         8b18944b-686d-4d8f-933e-1cf91acde0cf    4.6G     0% /home
├─sda3 vfat    FAT32                       49D4-F05D                             863.6M    15% /boot
└─sda4 swap    1                           48ca91ed-1400-4359-9f4e-0901a309187f                [SWAP]
sr0    iso9660 Joliet Extensio ARCH_202501 2025-01-01-08-45-10-00
```

```
[root@arch ~]# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda      8:0    0   22G  0 disk
├─sda1   8:1    0   15G  0 part /
├─sda2   8:2    0    5G  0 part /home
├─sda3   8:3    0    1G  0 part /boot
└─sda4   8:4    0 1022M  0 part [SWAP]
sr0     11:0    1  1.1G  0 rom
```

Reference:

lsblk man page.

- **P3**

```
[root@arch ~]# cat /etc/os-release
NAME="Arch Linux"
PRETTY_NAME="Arch Linux"
ID=arch
BUILD_ID=rolling
ANSI_COLOR="38;2;23;147;209"
HOME_URL="https://archlinux.org/"
DOCUMENTATION_URL="https://wiki.archlinux.org/"
SUPPORT_URL="https://bbs.archlinux.org/"
BUG_REPORT_URL="https://gitlab.archlinux.org/groups/archlinux/-/issues"
PRIVACY_POLICY_URL="https://terms.archlinux.org/docs/privacy-policy/"
LOGO=archlinux-logo
```

```
[root@arch ~]# uname -sr
Linux 6.12.10-arch1-1
```

Reference:

uname man page.

7. Flag hunting

- **P1**
  i. With `echo $HISTFILE` :

```
nasa@nasa:~$ echo $HISTFILE
/home/nasa/kickstart.nvim/.git/logs/refs/remotes/origin/HEAD
```

  Path: `/home/nasa/kickstart.nvim/.git/logs/refs/remotes/origin/HEAD`
  ii. With `export HISTSIZE=<number>`
  iii. With `export HISTFILESIZE=<number>`
  iv. Default history file:

```
nasa@nasa:~$ cat .bash_history
./gen_flag --line 104 --out new_history_file
exit
```

  In line 104 of new history file (using `less` and go to line x):

```
echo NASA{y0UF1nd+heCoRr3tFL4G}
```

  Flag: NASA{y0UF1nd+heCoRr3tFL4G}
  Reference:
  https://datawookie.dev/blog/2023/04/configuring-bash-history/

- **P2**
  From `./treasure` :

```
nasa@nasa:~$ ./treasure
Searching for treasures....(It might take some time)

j
You found a really big treasure chest!
But there are many garbage around...
What you are looking for is in the smallest file, and at the start of line 418.
 That's all I know for now...
Good luck!
```

  In `treasure-chest/` , with `sort -Sl` :

```
-rw-rw-r-- 1 nasa nasa 2023063 Feb  1 07:00 flag-068
-rw-rw-r-- 1 nasa nasa 2019035 Feb  1 07:03 flag-708
-rw-rw-r-- 1 nasa nasa 2018028 Feb  1 07:04 flag-900
-rw-rw-r-- 1 nasa nasa 2017021 Feb  1 07:04 flag-795
-rw-rw-r-- 1 nasa nasa 2014000 Feb  1 07:04 flag-815
-rw-rw-r-- 1 nasa nasa 1102665 Feb  1 07:04 flag-962
nasa@nasa:~/treasure-chest$
```

  In line 418 in `treasure-chest/flag-962` (using `less` and go to line x):

```
NASA{EZ_TrEa$Ur3_HunT!}TVcbbpVRPzo90Q4SXu1HxICFaZro3tdWEV81vM{TQp}Yy3C4XXIUpsoU}Yt3mA{nf7geKclNXdr4h62RKRHxnj25bCzulgVMjS57YuJuF4B1xM4TXIvD9dYzI7Nnz2LqPEpDfnz6k
S1BsgtR7NH20XfSB}9T9UsTFQVQcD6CV13{kCWr1qe64OTDJIqGX}Xy}my50yHoiDyUzgkrYSq9MeAlay85fCtlDKrG46mXUVwvTAoAN3KnweVIOpIcbntQ_LV9RJGHk1Sg_d0phPLQmlF3DXdIDAJLXdIi3r3eX
WHVnKveoYeL51WKGnMrWtcoA}8}rna871McOJJhp14qd}sWx7unwGKjGJv5ICbOSK6wpOQ_R{Bb{_vz{MsrjbD_PO2Brmiv8iJezXpnXcyZ3XQ9csaercpxWBUuYap}Rlon{{lM5Sh2ljcqqN4WJV5Fx{49fZR18
kjWsNXedQZH2LlXdlB3EFK6NR2UN}8CsmVMR1nb9LE5o1JTej7kRDrrGwszpuYPJhwVuKLaiXw8Hc}ziQXlJ8PkBmX6XrWgDeTmcXqFc2GWkC3KuG2mKYol8nyHVj9kAIo8a3GXbfiuENjtsfC739acgFWK9Y8YS
BI5XrDUPgmZOp1mSMdEJXvEy9XiQHlXm6SECuFuM_vwumGvA62rKxr4g4ogJII0}2Ui7eIDtWlAxW_Q77uJfPKNOfOX5Fp}lueyYa5dhdcol3ofnT}qDK3o3Yb2wX1K0ZxW3knyJf}{rydFddn01sVcv2zJktpy1
Ggc}owNUtMhU902ljcsMGiUlSwGz{9lFxpFS3gVV}YsEWpjmMZPgnXXR6xc4bqIr9ej7HKZFZ}GprYP1BVkJ3TZxvLssg1IdV9rfF8Qr7Z0pBKX3zTSyOhZ94Qly94LNbgUEsUlkCmP4Rs15fIpcj21y2oDnXCKD
LZB5VEi55k3EXf7_coxhGyFDOeYw0ZlOgQ}SAsTPX7Slg}
```

  Flag: NASA{EZ_TrEa$Ur3_HunT!}
  Reference:
  sort man page

- **P3**

  With `./boss > tmp & pkill -P $!` :

```
nasa@nasa:~$ ./boss > tmp & pkill -P $!
[1] 2211
nasa@nasa:~$ cat tmp
If you can kill all my subprocesses within 3 seconds, I will show you my secret!
Well done!
NASA{m0dERn_Pr0B1em$_reQU1r3_m0dERn_SOluT10N5}
[1]+  Done                    ./boss > tmp
```

  Flag: NASA{m0dERn_Pr0B1em$_reQU1r3_m0dERn_SOluT10N5}

  Reference:

  pkill man page

  https://unix.stackexchange.com/questions/30370/how-to-get-the-pid-of-the-last-executed-command-in-shell-script

- **P4**

  We can use two `grep` s to get the passcode:

```
nasa@nasa:~$ strings chal | grep 486 | grep re02
ioewe3h486hu5tnjdsre029y814mmq
```

```
nasa@nasa:~$ ./chal ioewe3h486hu5tnjdsre029y814mmq
Here is the flag: NASA2025{n4ndeharuh1ka93yatt4n0}
```

  Flag: NASA2025{n4ndeharuh1ka93yatt4n0}

- **P5**

  Find and check `tmux` config file:

```
nasa@nasa:~$ find . -name ".tmux.conf"
./.tmux.conf
```

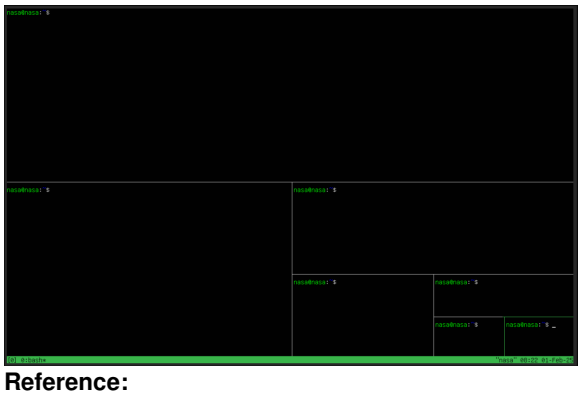
```
nasa@nasa:~$ cat .tmux.conf
unbind C-b
set-option -g prefix C-a
bind-key C-a send-prefix
```

  We see that the prefix has been set to Ctrl-a.

  After entering `tmux` , we can make the layout with the following series of commands:

  i. `Ctrl-a "` (Split horizontally)
  ii. `Ctrl-a %` (Split vertically)
  iii. `Ctrl-a "` (Split horizontally)
  iv. `Ctrl-a %` (Split vertically)
  v. `Ctrl-a "` (Split horizontally)
  vi. `Ctrl-a %` (Split vertically)

  Result:



  Reference:

  https://wiki.archlinux.org/title/Tmux

8. NASA 國的大危機

- **P1**

`Dockerfile`：

```
nasa@nasa-hw0-pickle:~/mystic-cup$ cat Dockerfile
FROM python:3.9-slim
RUN apt-get update && apt-get install -y \
    build-essential \
    libssl-dev \
    net-tools \
    iproute2 \
    tcpdump \
    tshark \
    nano \
    curl \
    wget \
    vim \
    less \
    procps \
    lsof \
    iputils-ping \
    && rm -rf /var/lib/apt/lists/*

RUN mkdir -p /usr/libexec/run

COPY usr/libexec/run/dist/transfer /usr/libexec/run/transfer
COPY usr/libexec/run/run.sh /usr/libexec/run/run.sh

RUN chmod +x /usr/libexec/run/transfer
RUN chmod +x /usr/libexec/run/run.sh

CMD ["/usr/libexec/run/run.sh"]
```

`FROM python:3.9-slim` : Specify the base image that we are building.

`RUN apt-get update && apt-get install -y ... && rm -rf /var/lib/apt/lists/*` : Update package manager and install packages, the `rm` is to remove files created by `apt-get update` , this is done to reduce the layer size.

`RUN mdkir -p /usr/libexec/run` : create directory.

`COPY usr/libexec/run/dist/transfer /usr/libexec/run/transfer` : Copies file into the filesystem.

`COPY usr/libexec/run/run.sh /usr/libexec/run/run.sh` : Copies file into the filesystem.

`RUN chmod +x /usr/libexec/run/transfer` : Change the newly copied file into an executable.

`RUN chmod +x /usr/libexec/run/run.sh` : Change the newly copied file into an executable.

`CMD ["/usr/libexec/run/run.sh"]` : Sets the script `run.sh` to run when launching the build image.

Reference:

https://docs.docker.com/reference/dockerfile/

https://docs.docker.com/get-started/docker-concepts/building-images/understanding-image-layers/

https://askubuntu.com/questions/179955/var-lib-apt-lists-is-huge

https://linux.die.net/man/8/apt-get

https://opensource.com/article/20/5/optimize-container-builds

- **P2**

  `docker images` :

```
nasa@nasa-hw0-pickle:~/mystic-cup$ docker images
REPOSITORY     TAG       IMAGE ID       CREATED       SIZE
my-magic-cup   latest    3cddc4add21e   4 weeks ago   727MB
```

  `docker run <ID>` :

```
nasa@nasa-hw0-pickle:~/mystic-cup$ docker run 3cddc4add21e
System routine check: FAILED. Exiting...
```

  Check `run.sh` :

```
nasa@nasa-hw0-pickle:~/mystic-cup$ cat usr/libexec/run/run.sh
#!/bin/bash

if [ "$MAGIC_SPELL" = "hahahaiLoveNASA" ]; then
    echo "System routine check: OK. Service started..."

    echo "Starting sending secret message..."
    /usr/libexec/run/transfer &
    tail -f /dev/null
else
    echo "System routine check: FAILED. Exiting..."
    exit 1
fi
```

  Set environmental variable with `-e VAR=value` :

```
nasa@nasa-hw0-pickle:~/mystic-cup$ docker run -e MAGIC_SPELL="hahahaiLoveNASA" 3cddc4add21e
System routine check: OK. Service started...
Starting sending secret message...
```

  Reference:

  docker man page & docker --help

  https://docs.docker.com/compose/how-tos/environment-variables/set-environment-variables/

- **P3**

  Rerun image in background with `-d` flag:

```
nasa@nasa-hw0-pickle:~/mystic-cup$ docker run -e MAGIC_SPELL="hahahaiLoveNASA" -d 3cddc4add21e
ab2be26d023d3b8166afdc1deefc7437975634d9d76caa225a0ad516546fe959
```

  Get container id:

```
nasa@nasa-hw0-pickle:~/mystic-cup$ docker ps -q
2c4abc7196a8
```

  Enter container with `docker exec -it <ID> /bin/bash` :

```
nasa@nasa-hw0-pickle:~/mystic-cup$ docker exec -it 2c4abc7196a8 /bin/bash
root@2c4abc7196a8:/#
```

  Use `tcpdump -i any -nn -A` to sniff packets and show packet contents:

```
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on any, link-type LINUX_SLL2 (Linux cooked v2), snapshot length 262144 bytes
10:50:34.536751 lo    In  IP 127.0.0.1.5000 > 127.0.0.1.6000: UDP, length 40
E..D..@.@.3.....p.0.Cflag[I'll send our killer on 3948/02/22]
10:50:39.538231 lo    In  IP 127.0.0.1.5000 > 127.0.0.1.6000: UDP, length 40
E..D
2@.@.2u.....p.0.Cflag[I'll send our killer on 3948/02/22]
10:50:44.539637 lo    In  IP 127.0.0.1.5000 > 127.0.0.1.6000: UDP, length 40
E..D..@.@.1.....p.0.Cflag[I'll send our killer on 3948/02/22]
```

  Flag:

  flag[I'll send our killer on 3948/02/22]

  Reference:

  docker man page & docker --help

  https://docs.docker.com/reference/cli/docker/container/exec/

  tcpdump man page

  https://opensource.com/article/18/10/introduction-tcpdump