
GlobalLogic
A Hitachi Group Company
EDUCATION
Network tooling
in Operating System
Andrii Beregovenko

Agenda:

- Configuring network of the operating system
- Debugging network configuration state
- Software for automated configuration of the network

Configuring operating system

Configuring address

The Interface IP configuration is related to the Layer 3 (OSI model) configuration. On this level the IP networking related configuration is being made.

Here are list of necessary parameters in order to make IP networks work:

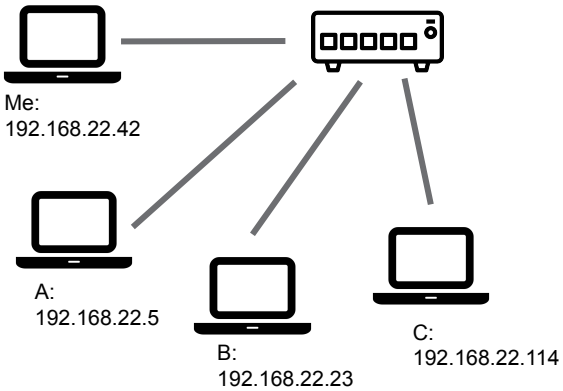
- Interface IP address
- Network address
- Network mask
- Network broadcast address

Each physical or virtual network interface has to have a separate configuration, depends on to which network it is connected.

There are common agreements and well defined standard approach in networking, such as:

- There are defined IP network classes (RFC917)
- There are definite IP networks for use as a private network, and they are not routable in the global network (RFC1918)
- There are defined relation between network, mask and broadcast
- Some values can be calculated out of other

Example of configuration:
Let's define a host IP address as 192.168.22.42
But we have friends and we are in the **same physical network**, so IP network to be configured. So we need network which fit em all and myself.
Thus, let's define network address 192.168.22.0
To fully define a network we have to specify network mask, so that the size of network will be known.
First let's define last IP address in range: 192.168.22.255
So, computed mask will be 255.255.255.0
The last mandatory parameter is a broadcast address.
By specification the broadcast address is the last address of the network 192.168.22.255



Fortunately configuration can be done simpler and by having IP/mask pair everything else can be calculated automatically using bit operations:

```
Address:   192.168.22.42        11000000.10101000.00010110 .00101010
Netmask:   255.255.255.0        11111111.11111111.11111111 .00000000
```

Calculated parameters will be:

```
Network:   192.168.22.0/24      11000000.10101000.00010110 .00000000
Broadcast: 192.168.22.255       11000000.10101000.00010110 .11111111
HostMin:   192.168.22.1         11000000.10101000.00010110 .00000001
HostMax:   192.168.22.254       11000000.10101000.00010110 .11111110
```

The two hosts are belonging to the same network if their IP/mask pair produces the same Network and Broadcast addresses

In Linux there is a software to configure network interfaces IP parameters: **iproute2**. It contains a particular tool called ip.
Here is an example how to use it:

- Add an ip address to the interface:
  - `ip address add ip/mask dev eth0`
- Remove ip address from interface:
  - `ip address del ip/mask dev eth0`
- Show network interface ip configuration
  - `ip address show dev eth0`

This type of interface configuration is called **static** or **manual**.

Dynamic IP configuration

In most of the cases when personal devices are used it is inconvenient to manually specify a network parameters for the user. In many cases you might not even have such information provided.

To solve this problem there is a Dynamic Host Configuration Protocol, a.k.a. DHCP. This special protocol allows your device (yet unconfigured) to discover correct network parameters and configure own network interface without user involvement.

Usually the default software of the device operating system have all necessary tools and does DHCP configuration for you.

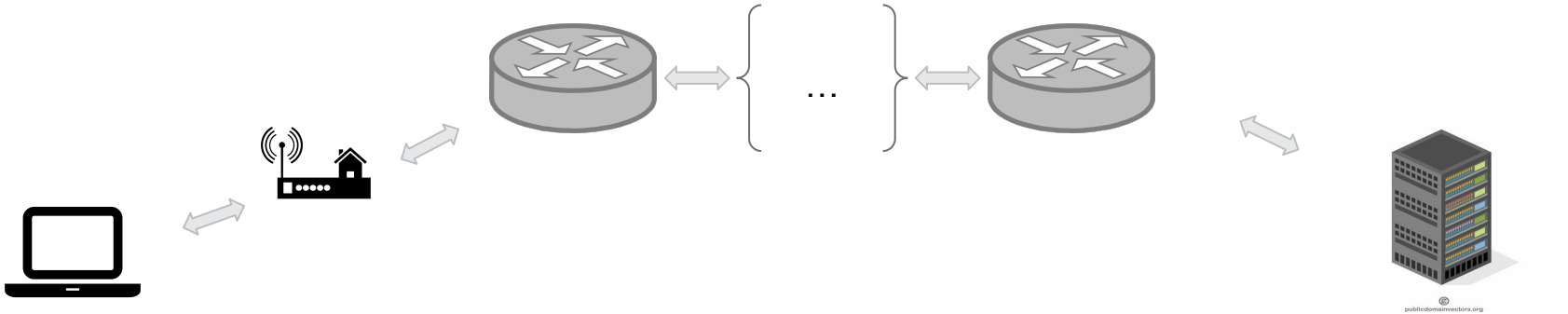
Typically DHCP server sends you a following information

- your IP address
- your network mask
- the default router ip address
- the list of DNS server to convert internet addresses into IP addresses
- additional IP routes
- miscellaneous configurations related to general network aspects

Configuring IP routing

The previously done configuration is good enough if the only computer we would like to contact, belongs to our network.

If we would like to reach external hosts, which belongs to other networks, for instance in Internet, we have to configure IP routing in our computer



The routing configuration can be done using the tool ip, like in example

- Add new route to a single host
  - `ip route add ip/32 via router_ip`
  - Example: `ip route add 8.8.8.8/32 via 192.168.22.1`
- Remove any route from local routing table
  - `ip route del ip/mask via router_ip`
- Show current routing table
  - `ip route show`
- A special scenario, when we want to send all the traffic through the router(except of the local network traffic)
  - `ip route add default via router_ip`
    or
  - `ip route add 0.0.0.0/0 via router_ip`

The routing table has a very important rule

- The bigger network is, than less priority it will have while network stack searching in routing table how to reach desired IP address

The biggest network is 0.0.0.0 with mask 0.0.0.0, which literally means that any existing IP address will belong to this network.

The smallest network is the network which contains only one host ip/255.255.255.255, e.g. 192.168.22.42/255.255.255.255 is a single-host network

Configuring hostname

In the networks for simplicity of use there was introduced a hostnames.

Every server might have a name associated with it. However, this is not a mandatory, and can easily operate with no given hostname, using just IP.

The hostname is a name of the host, while FQDN (fully qualified domain name) contain both hostname and domain name, separated by “.” sign.

For example:

- the server hostname is: vega
- the network domain is: uac.com
- thus the fqdn of the server is: vega.uac.com

Locally on the server/computer such information can be stored in files:

- /etc/hostname - hostname
- /etc/domainname - the domain name where server belongs to
- /etc/hosts - contains static information about translation hostname or fqdn into ip addresses

The process of finding IP address by hostname or FQDN is called name resolution

The file /etc/hosts might contain information about any FQDN to ip resolution, not necessarily local ones.

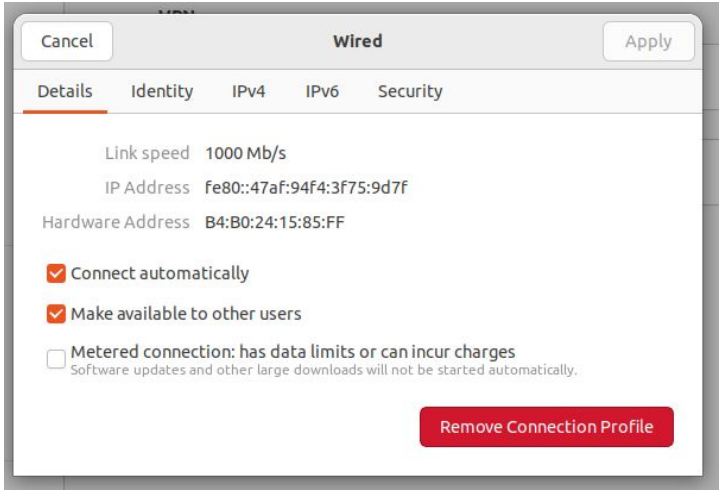
In the Internet there is a special service called DNS which helps to resolve any domain name.

System network helper services

On typical GNU/Linux based machine there is a software which suppose to help you with automatic configuration of the network interfaces and subsystems, as well as preserving this configuration across reboot, so that you do not need to reconfigure it again after reboot.

On a consumer grade computer in most of the cases such software is a NetworkManager. It provides you a graphical UI as well as TUI (text user interface), so that the network configuration can be configured using simple interfaces. Advanced(really advanced, for instance interface MTU) options usually needs to be configured from the command line using a console tool called nmcli.

Here is an example of typical configuration screen of the NetworkManager graphical user interface, where you can change various parameters of each network interface using mouse



Also here is some example commands for NetworkManager console tool

Add new connection:

```
nmcli con add type ethernet con-name "static-ip" ifname eth0 \
 ipv4.method manual ipv4.addresses 192.168.2.150/24 gw4 192.168.2.1
```

See various networking information

```
nmcli connection show
nmcli device
```

Another basic software which helps with a network subsystem configuration is ifup. Mostly Debian based, it is available on almost any GNU/Linux based system nowadays. It is good for stationary computer which never have their configuration changed

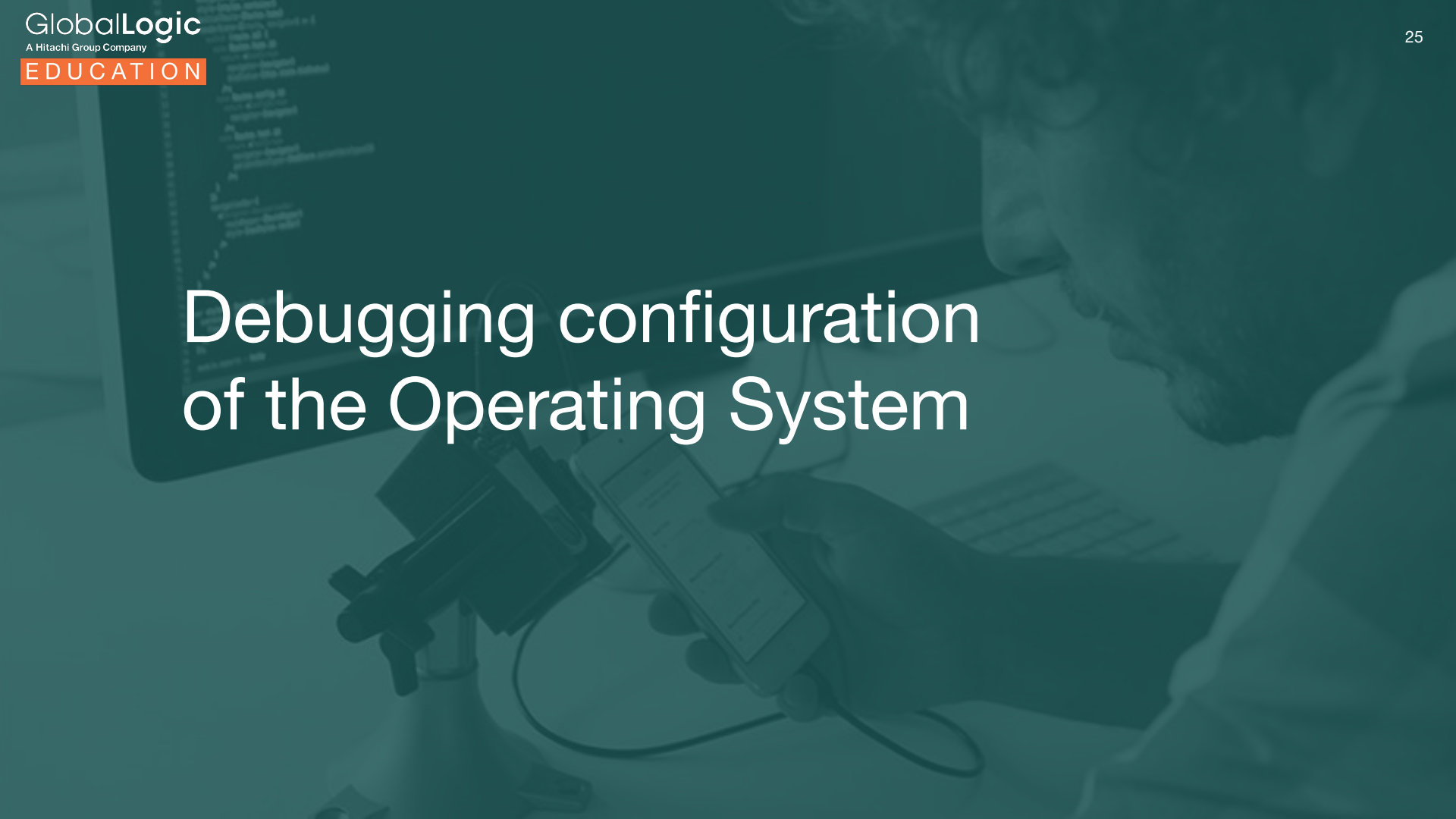
A typical config file might look like this:

file: `/etc/network/interfaces`

```
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
    address 192.168.0.42/24
    gateway 192.168.0.1
    dns-nameservers 8.8.8.8
```

The modern rising star of the cloud based computers is netplan, which provides you a very flexible way of the network subsystem configuration using YAML-based config files. Very suitable for the environment where network infrastructure is highly dynamic, or node-based. Allows for simple generative configuration, where YAMLs are being distributed to the virtual computational nodes, so they can talk to each other. Typical config file looks like following(`/etc/netplan/ethernet.yaml`):

```
network:
  version: 2
  renderer: networkd
  ethernets:
    eth0:
     dhcp4: no
     addresses: [192.168.0.42/24]
     gateway4: 192.168.0.1
     nameservers:
       addresses: [8.8.8.8,8.8.4.4]
```



Debugging configuration of the Operating System

The diagnostic of the host IP configuration can be done by using ip tool

- `ip address <command>` - will show
  - the interface ip address configuration
  - the physical interface state state: UP, DOWN, NO-CARRIER, etc
- `ip route <command>` - will show
  - current routing table
  - known directly connected networks
  - in particular command ip route get <ADDRESS> will show where the packet will go if would be sent to that <ADDRESS>

To diagnose remote hosts availability there are two pretty much standard tools:

- `ping`
- `traceroute`

The ping tool will use a special ICMP protocol to communicate with remote host and will return you a time to the host it took to travel
Example of use: `ping 8.8.8.8`

The traceroute tool will show you the actual path how packets are traveling to destination through intermediate routers.
Example of use: `traceroute 8.8.8.8`

Additional very useful tool is netstat

It allows to see much of the different information related to network and communication but in many different contexts.

Such as:

- show currently active tcp services
- show currently established connections
- show which application uses connection
- show hosts in both hostnames or IP addresses
- and much more

Debugging of the hostnames resolution can be done using two pretty much standard tools:

- host - tries to resolve given hostname and show all IP addresses associated with this name (yes, there might be multiple IP addresses associated with single FQDN); This tool does not showup a lot of information how resolution had happened, so good for quick check.
- dig - more advanced tool compatring to host. Allows to precisely specify how name resolution should go, which DNS servers to use; provides a lot of debug information. Good for debugging complicated issues with DNS system.

In addition to that both ping and traceroute will try to perform forward and reverse(resolve hostname by ip) names resolution when it is applicable.

Example

Interfaces configuration example

```
jet@krk1-ldl-p91456:/storage/vizio/platform$ ip address show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN group default qlen 1000
    link/ether b4:45:06:3d:b6:f2 brd ff:ff:ff:ff:ff:ff
    altname enp0s31f6
3: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether cc:15:31:2d:08:84 brd ff:ff:ff:ff:ff:ff
    altname wlp0s20f3
    inet 10.11.12.115/24 brd 10.11.12.255 scope global dynamic noprefixroute wlan0
       valid_lft 56518sec preferred_lft 56518sec
    inet6 fe80::8e94:d35f:bac:567b/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
6: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:42:2f:fd:c0 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
    inet6 fe80::42:42ff:fe2f:fdc0/64 scope link
       valid_lft forever preferred_lft forever
```

iproute software examples

Show the list of routes on a host

```
jet@krk1-ldl-p91456:~$ ip ro sh
default via 10.11.12.1 dev wlan0 proto dhcp metric 600
10.0.3.0/24 dev lxcbr0 proto kernel scope link src 10.0.3.1
10.11.12.0/24 dev wlan0 proto kernel scope link src 10.11.12.115 metric 600
169.254.0.0/16 dev lxcbr0 scope link metric 1000
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1 linkdown
jet@krk1-ldl-p91456:~$ ;1~
```

Get a route which would be chosen for selected destination

```
jet@nuc1:~$ ip route get 8.8.8.8
8.8.8.8 via 172.26.136.65 dev eno1 src 172.26.136.72 uid 1001
    cache
jet@nuc1:~$
```

Examples of netstat use

See list of service on the server, limiting them to IPv4 only

```
jet@nuc1:~$ sudo netstat -plnt4
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:3389            0.0.0.0:*               LISTEN      1005/sshd: /usr/sbi
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      1005/sshd: /usr/sbi
tcp        0      0 10.13.15.1:53           0.0.0.0:*               LISTEN      960/dnsmasq
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      568/systemd-resolve
jet@nuc1:~$
```

Show the routing table

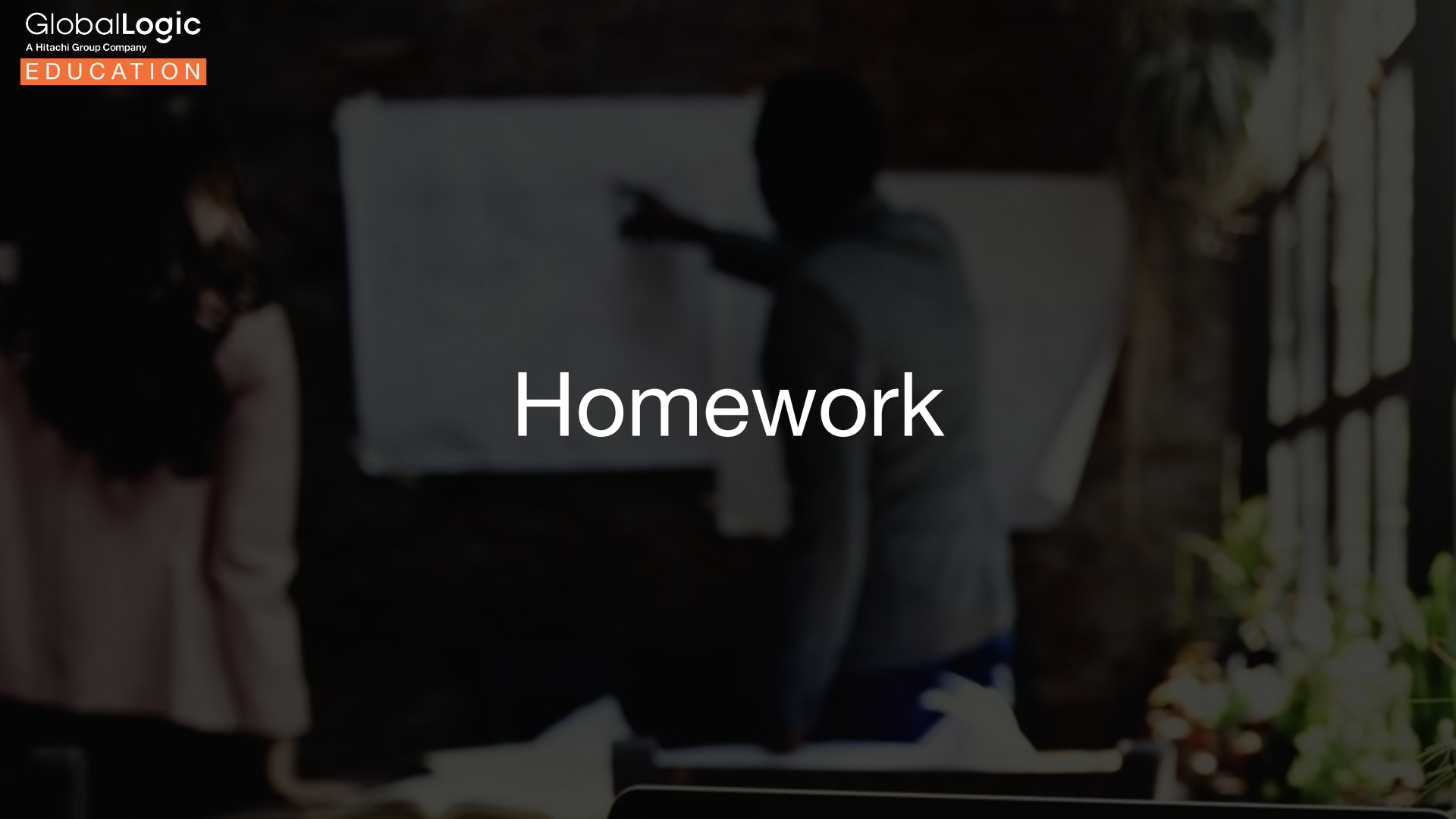
```
jet@nuc1:~$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         172.26.136.65   0.0.0.0         UG        0 0          0 eno1
10.0.1.202      172.26.136.65   255.255.255.255 UGH       0 0          0 eno1
10.0.1.206      172.26.136.65   255.255.255.255 UGH       0 0          0 eno1
10.13.15.0      0.0.0.0         255.255.255.0   U         0 0          0 wlx6045cb95d377
172.17.0.0      0.0.0.0         255.255.0.0     U         0 0          0 docker0
172.26.136.64   0.0.0.0         255.255.255.192 U         0 0          0 eno1
172.26.136.65   0.0.0.0         255.255.255.255 UH        0 0          0 eno1
172.26.140.2    172.26.136.65   255.255.255.255 UGH       0 0          0 eno1
jet@nuc1:~$
```

Debugging name resolution

With first with host and than, more advanced, with dig

```
jet@nuc1:~$ host google.com
google.com has address 142.250.75.14
google.com has IPv6 address 2a00:1450:401b:801::200e
google.com mail is handled by 10 smtp.google.com.
jet@nuc1:~$
jet@nuc1:~$ dig @8.8.8.8 a google.com

; <<>> DiG 9.18.18-0ubuntu0.22.04.1-Ubuntu <<>> @8.8.8.8 a google.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 59033
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             300     IN      A       142.250.75.14

;; Query time: 28 msec
;; SERVER: 8.8.8.8#53(8.8.8.8) (UDP)
;; WHEN: Wed Jan 10 11:58:27 UTC 2024
;; MSG SIZE  rcvd: 55

jet@nuc1:~$
```

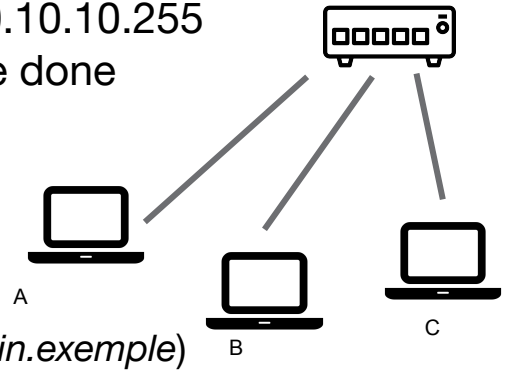
GlobalLogic
A Hitachi Group Company
EDUCATION



Homework

Homework

- There is 3 computers in the physical network: A, B, C. There is a dedicated IP addresses range that can be used 10.10.10.0 up to 10.10.10.255
  Please explain a network configuration that need to be done in order to achieve following results:
  - Computer A can directly communicate to B
  - Computer B can directly communicate with C
  - Computer A can not communicate directly with C
  - Each computer shall have a hostname and FQDN (*hosta*, *hostb*, *hostc* and domain name can be *testdomain.exemple*)
  - Every computer shall be able to resolve both hostname and FQDN into proper IP address

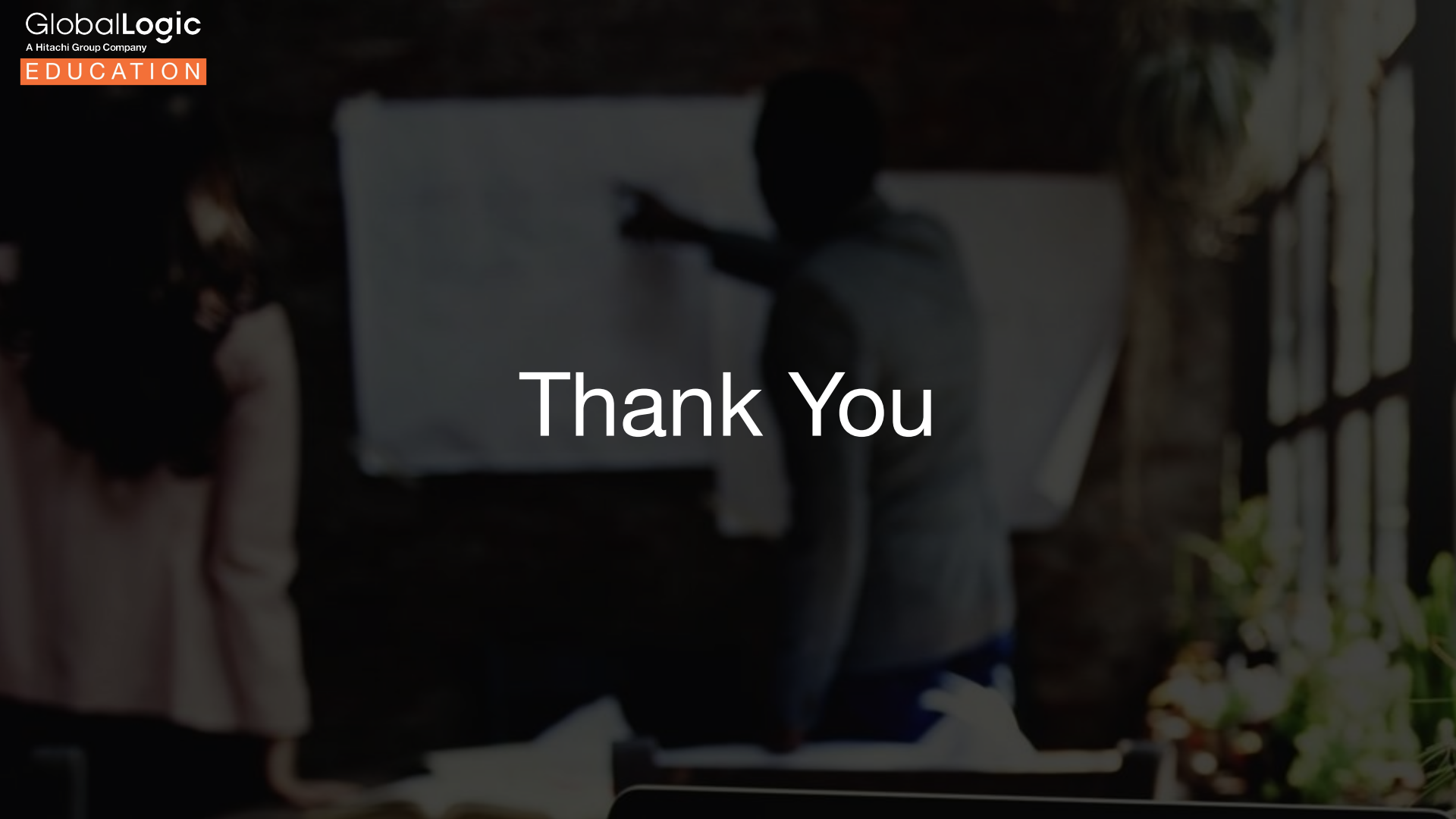


- Please provide all required commands to configure all three computers

Good luck!

GlobalLogic
A Hitachi Group Company
EDUCATION

Q&A


GlobalLogic
A Hitachi Group Company
EDUCATION
Thank You