7.5 FRR Alpine

ตัวอย่าง 7.1 ใช้ Alpine Linux เพื่อสร้าง เร้าเตอร์และรันโปรโตคอล OSPF

1. ดาวน์โหลดอิมเมจ Alpine

```
docker pull alpine:latest
```

2. Create 2 router containers: สร้างเร้าเตอร์จากนั้นกำหนดค่าต่าง ๆ ด้วยการสร้างเน็ตเวิร์กที่ชื่อว่า routernet และสร้าง router1 และ router2

```
docker network create --subnet=172.20.0.0/16 routernet

docker run -dit --name router1 --hostname router1 \
 --network routernet --ip 172.20.0.2 \
 --cap-add=NET_ADMIN --cap-add=NET_RAW \
 alpine:latest

docker run -dit --name router2 --hostname router2 \
 --network routernet --ip 172.20.0.3 \
 --cap-add=NET_ADMIN --cap-add=NET_RAW \
 alpine:latest
```

พารามิเตอร์	รายละเอียด
--cap-add	Add Linux capabilities
--cap-drop	Drop Linux capabilities

ตารางที่ 7.1: Physical features of LoRa and NB-IoT

Capability Key	รายละเอียด
NET_RAW	Use RAW and PACKET sockets
NET_ADMIN	Perform various network-related operations.

ตารางที่ 7.2: Network Capability

3. ติดตั้ง FRR ให้กับ router1 และ router2 ด้วยการเข้าไปใน shell ของเร้าเตอร์ทั้ง 2 ตัว

```
docker exec -it router1 sh

apk update
apk add frr frr-openrc ip6tables iptables
```

4. ติดตั้ง ospf ตั้งค่า /etc/frr/daemons เป็นรองรับการทำงานของ OSPF และ zebra (สามารถทำผ่าน nano) เพื่อกำหนดให้ค่า ospfd=yes และ zebra=yes

```
# Edit /etc/frr/daemons
sed -i s/ospfd=no/ospfd=yes/ /etc/frr/daemons
sed -i s/zebra=no/zebra=yes/ /etc/frr/daemons

# Enable IP forwarding
echo "net.ipv4.ip_forward=1" >> /etc/sysctl.conf
sysctl -p
```

5. ติดตั้ง OSPF:

```
# Start FRR
/usr/lib/frr/frrinit.sh start

# Configure via vtysh
vtysh

configure terminal
router ospf
 network 172.20.0.0/16 area 0
exit
exit
write memory
```

6. Verify:

```
# Check OSPF neighbors
vtysh -c "show ip ospf neighbor"

# Check routes
vtysh -c "show ip route"
```

กลับไปทำขั้นตอนที่ 3 - 5 ใหม่ที่ router 2 เพื่อให้ router ทั้งสองสามารถสื่อสารกันได้

ตัวอย่าง 7.2 ทดสอบการสร้าง เน็ตเวิร์กประกอบด้วย โฮสต์ สวิตช์และเร้าเตอร์

สร้าง Tolopogy hosts -- OVS -- routers -- OVS -- hosts.

```
# Install OVS on host
apk add openvswitch
rc-service ovs-modules start
rc-service ovsdb-server start
rc-service ovs-vswitchd start

# Create 2 OVS bridges
ovs-vsctl add-br ovs1
ovs-vsctl add-br ovs2

# Create router containers with multiple networks
docker run -dit --name router1 --hostname router1 \
 --cap-add=NET_ADMIN --cap-add=NET_RAW --network none alpine:latest

docker run -dit --name router2 --hostname router2 \
 --cap-add=NET_ADMIN --cap-add=NET_RAW --network none alpine:latest

# Create host containers
docker run -dit --name host1 --network none alpine:latest
docker run -dit --name host2 --network none alpine:latest

# Connect using ovs-docker (utility script)
ovs-docker add-port ovs1 eth0 host1 --ipaddress=10.1.1.10/24
ovs-docker add-port ovs1 eth0 router1 --ipaddress=10.1.1.1/24
ovs-docker add-port ovs2 eth0 host2 --ipaddress=10.2.2.10/24
ovs-docker add-port ovs2 eth0 router2 --ipaddress=10.2.2.1/24

# Create router interconnect network
docker network create --subnet=192.168.0.0/30 router-link
docker network connect router-link router1 --ip=192.168.0.1
docker network connect router-link router2 --ip=192.168.0.2
```