Önálló laboratórium 1.

Hibrid SDN hálózatok automatizálása

Barta Máté Zsombor

Konzulens: Dr. Zsóka Zoltán

Feladat:

A részben kontrollerrel vezérelt SDN kapcsolókból, részben hagyományos berendezésekből álló *hibrid hálózatok* költséghatékony megoldást jelenthetnek olyan felhasználási területeken, ahol a sok gyorsan módosítható kapcsoló mellett a hatékony, nagykapacitású routerekre is szükség van. Ilyen elemeket használhatnak fel például adatközpontokban, ahol nagyon sok végpontot kell összekötni.

Az SDN-ben használt kontroller-alapú megközelítésben a NorthBoundInterface-en keresztül tudjuk menedzselni, programozni a hálózatot. A hagyományos berendezéseket pedig például általános automatizáló eszközökkel lehet hatékonyan lekérdezni és konfigurálni. A hibrid megoldás esetén ezeket a technikákat integrálni kell, hogy a teljes hálózaton értelmezhető lekérdezéseket, illetve változtatásokat tudjunk végrehajtani.

Kapcsolódó feladatok, melyekből két-három teljesítendő egy félév során:

- SDN kontrollerek felderítése, értékelése az NBI szempontjából
- hibrid hálózatok kialakítása virtualizált környezetben
- egy kiválasztott kontroller NBI-jének megismerése
- általános automatizáló eszköz alkalmazása a virtualizált hálózati eszközöknél
- az NBI és az automatizálás integrált kezelése

Az idei félév célkitűzése:

- Megismerkedni az SDN hálózatok alapjaival
- Megismerni a különböző SDN kontrollerek működését
- Egy választott SDN kontroller segítségével egy hálózat felépítése és hálózati működésének szimulálása
- SDN hálózat összekapcsolása egy nem SDN vezérelt hálózati elemmel

1. Bevezetés

A modern infrastruktúra és hálózatok gyors ütemű fejlődése szükségessé tette olyan technológiák kifejlesztését, amelyek képesek dinamikusan alkalmazkodni a gyorsan változó igényekhez. Az egyik ilyen technológia a Software-Defined Networking (SDN), amely egy új, egyszerűbb megközelítést kínál a hálózati menedzsment és működés terén.

Az SDN alapvetően elkülöníti a hálózati eszközök irányítási síkját és adatforgalmi síkját, lehetővé téve a hálózati forgalom központi, szoftveres vezérlését. Ez az elkülönítés rugalmasságot és nagyobb kontrollt biztosít a hálózati adminisztrátorok számára, mivel a hálózat viselkedése központilag, szoftveresen programozható és módosítható anélkül, hogy fizikai beavatkozásra lenne szükség.

2. Openflow

Az OpenFlow az SDN egyik alapvető protokollja, amely kulcsfontosságú szerepet játszik a hálózati forgalom irányításában és vezérlésében. Az OpenFlow lehetővé teszi a hálózati eszközök, például switch-ek és routerek, programozását központi vezérlők segítségével, ezáltal jelentősen növelve a hálózat rugalmasságát és hatékonyságát.

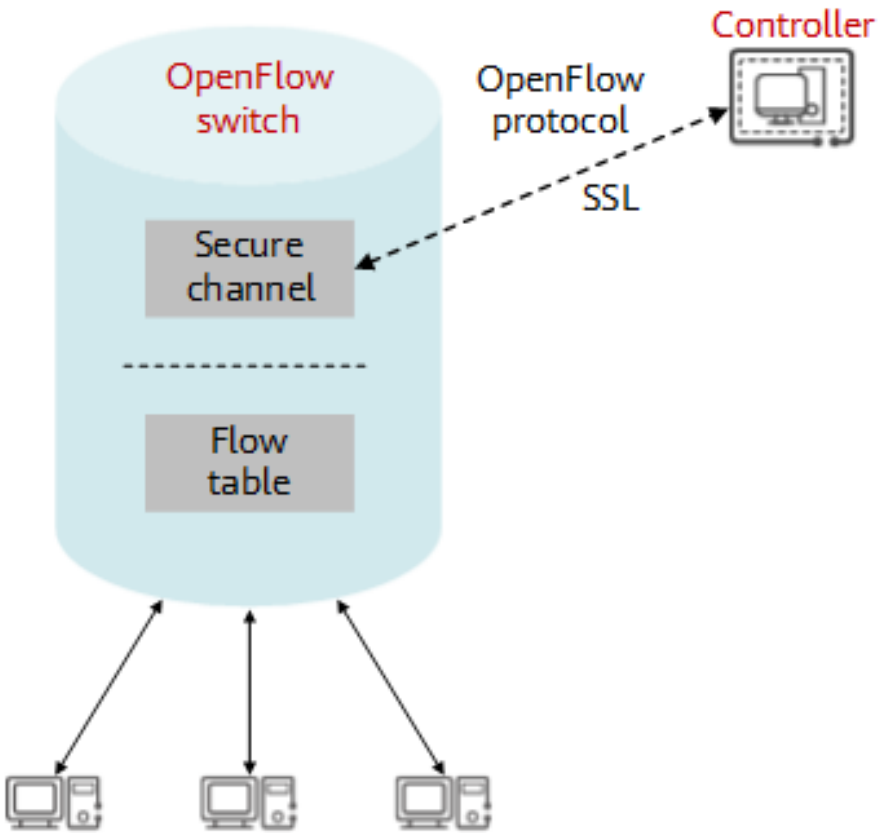
Az egyik legfontosabb tulajdonsága, ami igazából lehetővé teszi SDN kontrollerek használatát az az adat és irányítási sík szétválasztása. Ez a szeparáció lehetővé teszi, hogy a vezérlő szoftver (kontroller) központilag irányítsa a hálózati forgalmat, míg az adatforgalmi sík leköveti a vezérlő által adott utasításokat, kapcsolásokat, kapcsolatokat.

Az OpenFlow architektúrája három fő komponensből áll:

1. **OpenFlow vezérlő:** A vezérlő a hálózat központi agya, amely döntéseket hoz a forgalom irányításáról. A vezérlő legtöbbször szoftveres alkalmazás, amely globálisan rálát a hálózatra és ez alapján kezeli, és küldi el a vezérlő bejegyzéseket a lentebbi rétegekbe. A vezérlő akár mi magunk is lehetünk, amennyiben közvetlen kezeljük a hálózatban a flow bejegyzéseket és azokat mi írjuk be a közvetítő switchbe.

2. **OpenFlow kompatibilis switch-ek**: Ezek a hálózati eszközök fogadják és végrehajtják a vezérlő utasításait. Az OpenFlow switch-ek képesek feldolgozni és a bejegyzések alapján irányítani az adatforgalmat a vezérlő által meghatározott szabályok szerint.

3. **OpenFlow protokoll**: Ez a protokoll határozza meg a kommunikációt a vezérlő és a switch-ek között. Az OpenFlow protokoll segítségével a vezérlő képes utasításokat küldeni a switch-eknek, valamint információt gyűjteni a hálózati forgalomról és az eszközök állapotáról.

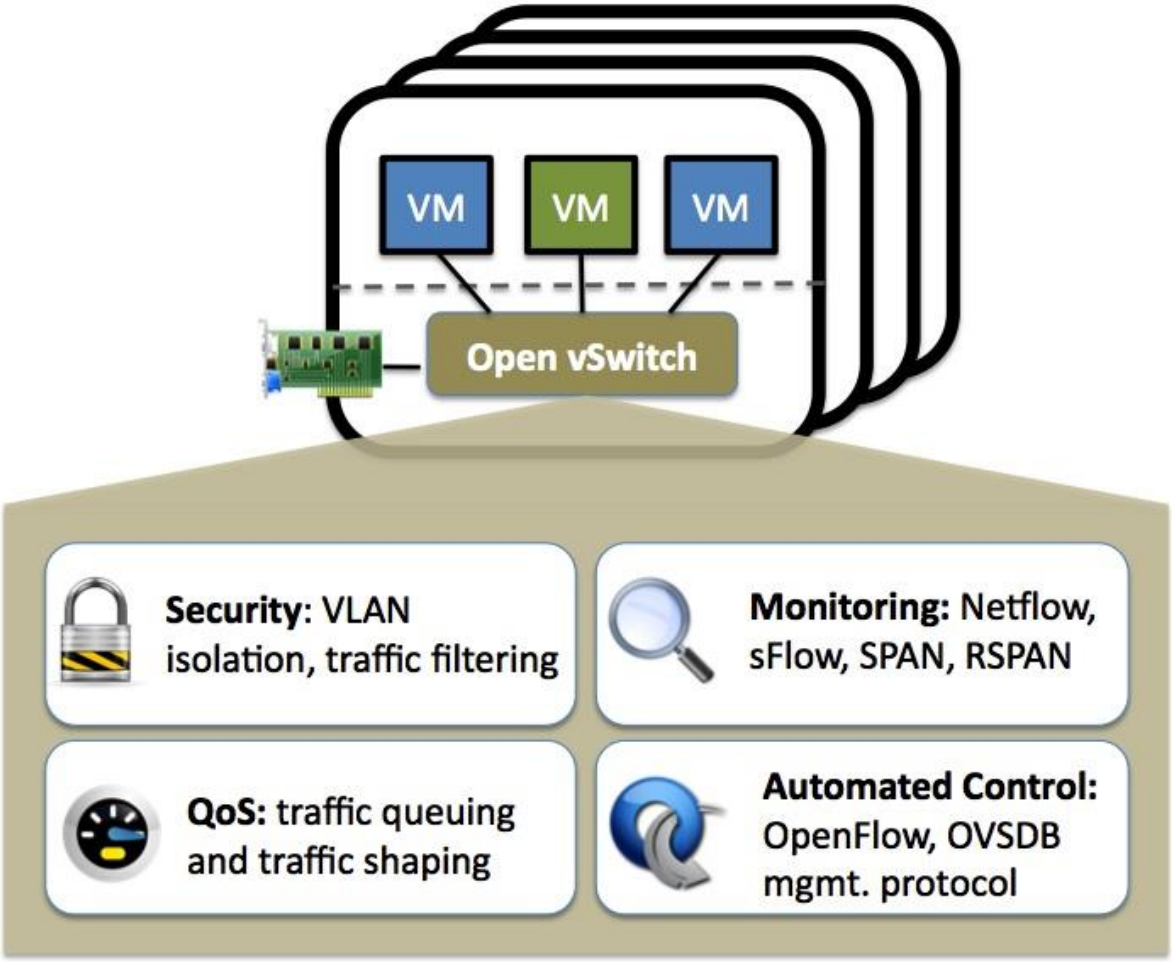


Az OpenFlow működésének alapja a flow tábla, amely a switch-ekben található. A flow tábla tartalmazza azokat a szabályokat, amelyek alapján a switch-ek eldöntik, hogyan kezeljék az érkező adatcsomagokat. A vezérlő dinamikusan frissíti a flow táblákat, hogy optimalizálja a hálózati forgalmat.

Amikor egy switch kap egy csomagot, ellenőrzi a flow tábláját, hogy talál-e megfelelő szabályt a csomag kezelésére. Ha talál, alkalmazza azt, és továbbítja a csomagot a megfelelő kimeneti porton. Ha nincs megfelelő szabály, a switch a vezérlőhöz fordul egy új szabályért. A vezérlő elemzi a csomagot és frissíti a switch flow tábláját a szükséges szabállyal.

3. Open vSwitch

Az Open vSwitch (OVS) egy nyílt forráskódú virtuális switch, amelyet kifejezetten virtuális hálózatokhoz és adatközpontokhoz fejlesztettek ki. Az OVS célja, hogy egy szoftveres megoldást biztosítson a fizikai switch-ek funkcionalitásának emulálására a virtualizált környezetekben. Képes támogatni a standard protokollokat, mint például az OpenFlow-t, és integrálható különféle virtualizációs platformokkal, például az OpenStackkel.



A kód nagy része platformfüggetlen C nyelven íródott, így könnyen átültethető más környezetekbe. Az Open vSwitch jelenlegi kiadása a következő funkciókat tartalmazza:

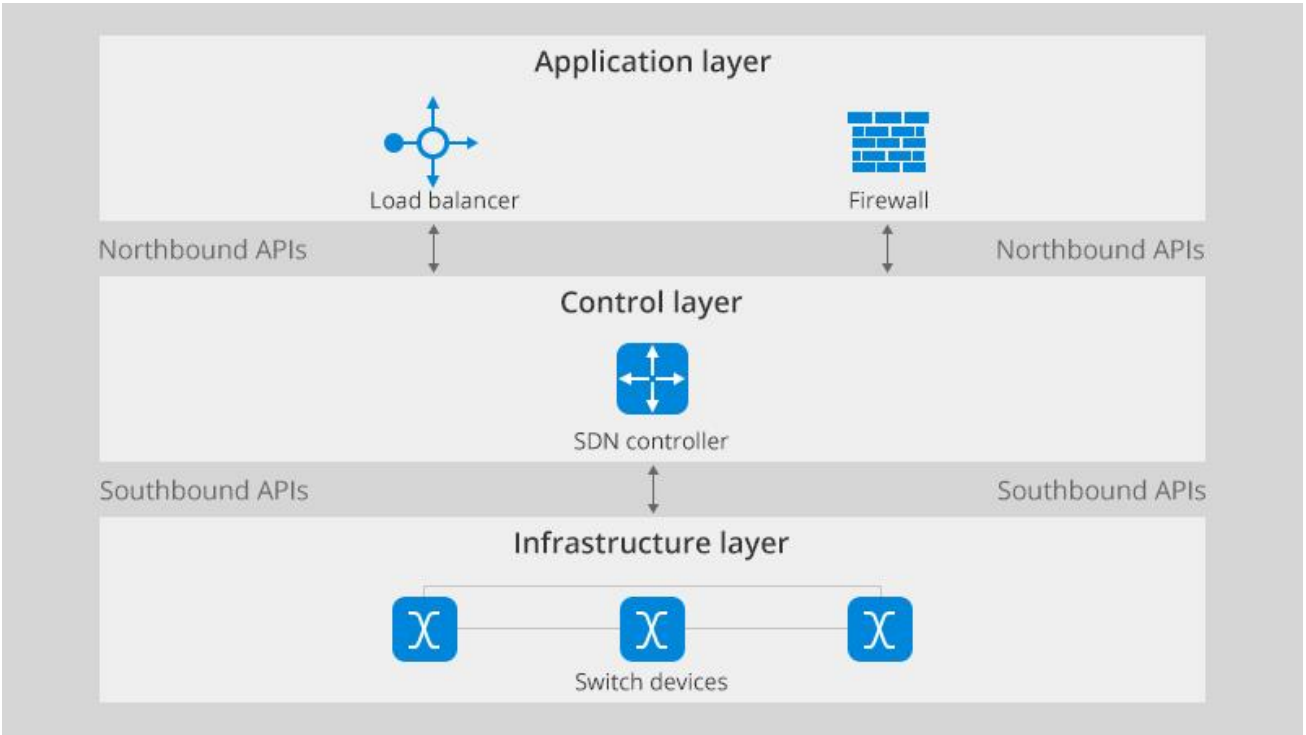
- Szabványos 802.1Q VLAN modell trunk és access portokkal
- NIC bonding, akár LACP-vel, akár anélkül
- NetFlow, sFlow(R) és mirroring a jobb láthatóság érdekében
- QoS (Quality of Service) konfiguráció, valamint forgalomkorlátozás

- Geneve, GRE, VXLAN, STT, ERSPAN, GTP-U, SRv6, Bareudp és LISP tunneling protokollok támogatása
- 802.1ag kapcsolat hibakezelés
- OpenFlow 1.0 és számos kiterjesztés
- Tranzakciós konfigurációs adatbázis C és Python interfészekkel
- Nagy teljesítményű adatforgalom irányítás Linux kernel modullal

Az Open vSwitch egy hatékony eszköz a virtuális hálózatok kezeléséhez, különösen nagy skálájú adatközponti környezetekben. Támogatja az összes főbb virtualizációs platformot, és lehetővé teszi a bonyolult hálózati topológiák egyszerű megvalósítását és kezelését. Telepítése és konfigurálása egyszerű, és számos haladó funkcióval rendelkezik, amelyek révén a hálózati adminisztrátorok teljes körű ellenőrzést gyakorolhatnak a virtuális hálózatok felett.

4. SDN alapjai

Az előbb ismertetett technológiák teszik számunkra lehetővé az SDN kontrollerek hatékony használatát. Az OpenFlow lehetővé teszi a logikai szétválasztás, ami az SDN megközelítés alapja, az Open vSwitch pedig, hogy egy azonos szabványos köztes médiumon keresztül történjen minden fajta hálózati kommunikáció. Így könnyen lehet adaptálni különböző gyártók között is a konfigurációt.



A SDN felépítése egyszerű. Két fő interfészt definiál, egy délit és egy északit. A déli interfész a kontroller és a hálózati infrastruktúra közötti kommunikáció megvalósításáért felel. Az északi pedig számunkra ad lehetőséget a kontroller konfiguráció módosításához.

Északi interfész:

Az északi interfész az SDN kontroller és az alkalmazások, valamint a felsőbb szintű szolgáltatások között helyezkedik el. Segítségével a kontroller szabványos kommunikációs interfészeket nyújt a felette lévő applikációk vagy akár a felhasználók felé. Ezen keresztül tudja a felhasználó módosítani a kontroller beállításait, a hálózati topológiát, a különböző szolgáltatásait, például QoS. Általában a kontrollerek nagy része REST és gRPC technológiát támogat, de előfordulnak más megoldások is.

Déli interfész:

A déli interfész az SDN kontroller és az adatforgalmat kezelő réteg (data plane) között helyezkedik el. Ez felel a kontroller ás a hálózatot irányító eszközök közötti kommunikációért. Ezen keresztül képes a kontroller leküldeni a kontroll üzeneteket az eszközöknek, ami meghatározza, hogy a flow bejegyzések hogyan érvényesüljenek. Az SBI-n (Southbound interface) keresztül információt tud gyűjteni a kontroller a hálózati eszközök állapotáról, például a forgalmi statisztikákról, hibákról és egyéb hálózati eseményekről. Több különböző protokollt támogat, a legelterjedtebb az Openflow, de ezen kívül általában képes NETCONF vagy RESTCONF-ra is.

5. SDN kontrollerek

Továbbiakban szeretném ismertetni az általam megismert SDN kontrollereket.

5.1. ONOS

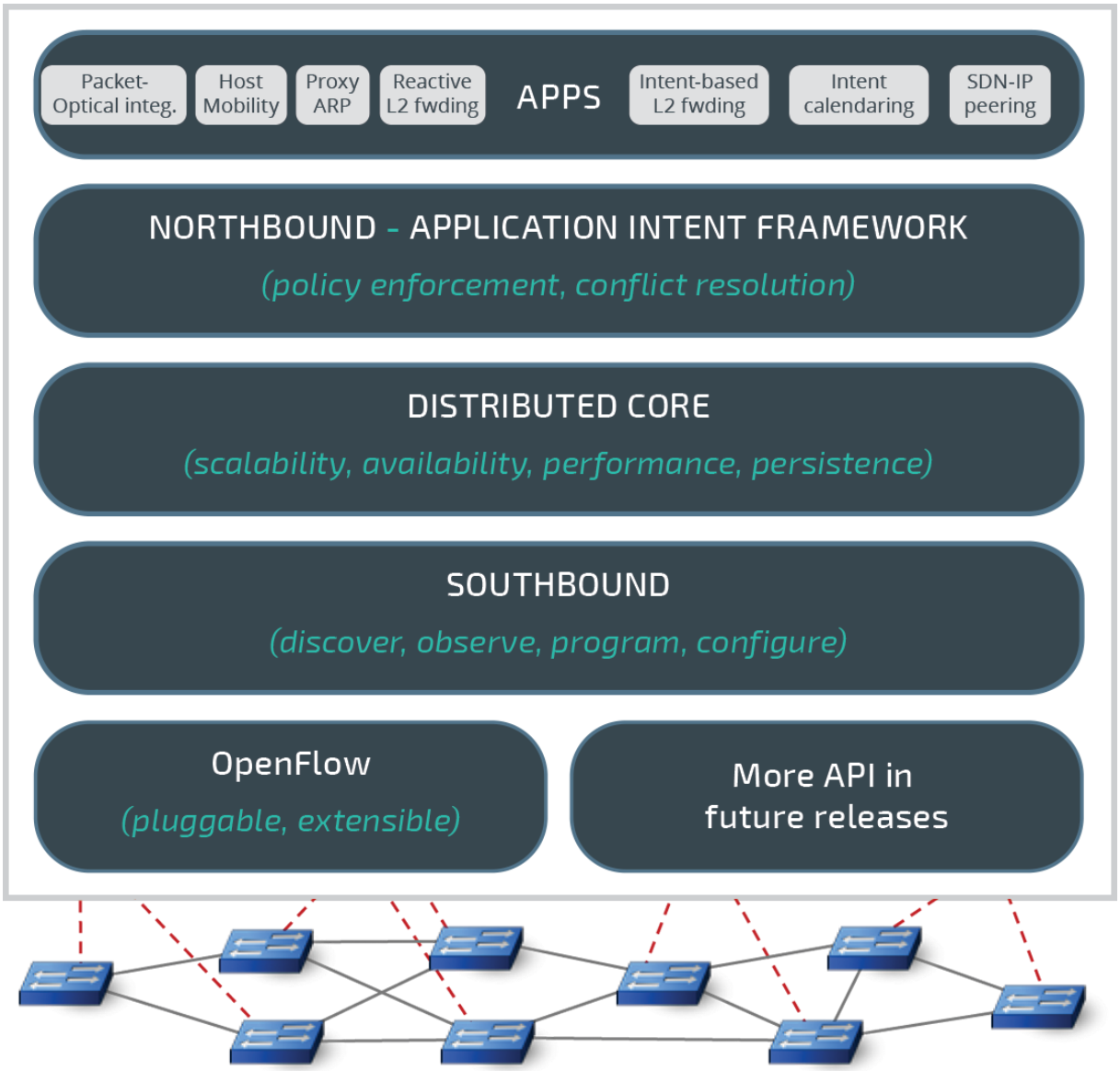
Ezt a kontrollert kifejezetten az SD-WAN hálózatokban való használatra találták ki, tehát a célja, hogy olyan kapcsolatokat működtessen, amik lehet, hogy földrajzilag több kilométeren átívelők. Ezen célból tud georedundanciát biztosítani, ezért több példányt is lehet egyszerre üzemeltetni akár másik földrészen is és képes mindegyik példányban valós képet tartani az egész hálózatról.

Főbb jellemzői:

- Eszközöket valós időben, „runtime” lehet ki/becsatlakoztatni
- Skálázható – több példány is működtethető egyszerre

- Atomi datastore konzisztenciát biztosít a kontrollerek között
- Natívan támogatja a BGP-t
- Hamar leterhelődik – nem sok hostra tervezték
- Déli interfésze támogatja többek között: OpenFlow, P4, NETCONF, TL1, SNMP, BGP, RESTCONF and PCEP
- Északi interfész: gRPC és REST API
- High avaliability átállást biztosít
- JAVA

Jelenleg a Linux Foundation Networking támogatása alatt van.



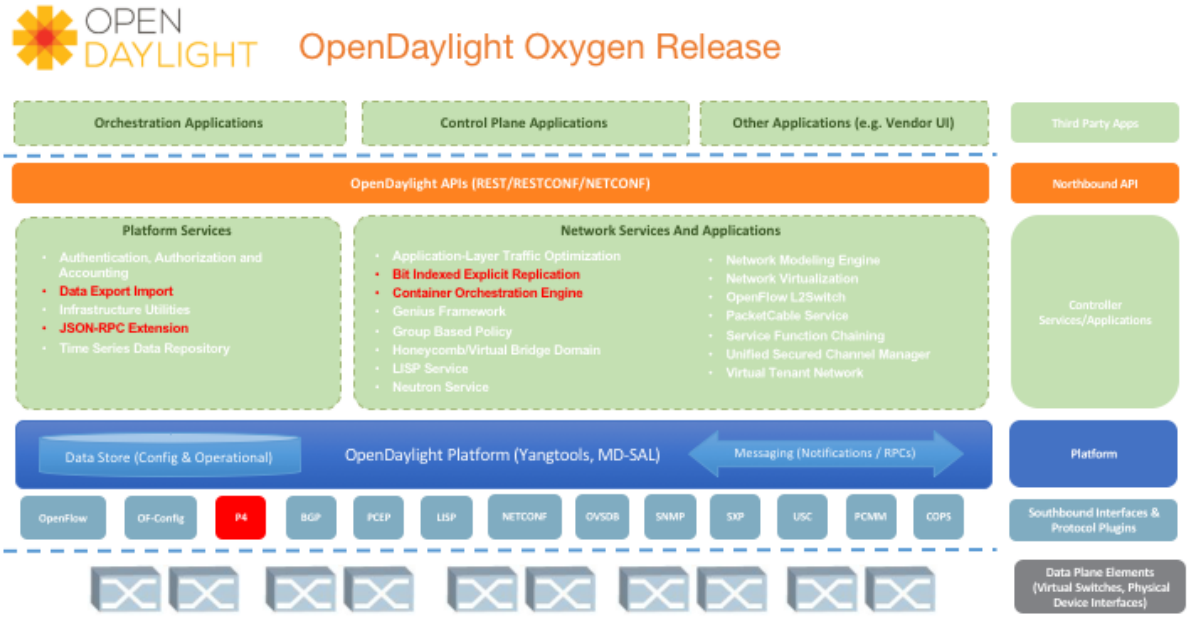
5.2. Opendaylight

Ezen disztribúció arra törekedett, hogy minél könnyebben adaptálható legyen különböző környezetekben, ezért technológiailag egy Java-s konténerizációs alapon nyugszik, ezzel garantálva, hogy különböző operációs rendszereken is ugyanúgy lehessen futtatni.

Főbb jellemzők:

- Egyszerű felhő integráció – pl.: Openstack
- Nagy skálázhatóság és flexibilitás
- In-memory valós kép a hálózatról
- Natív BGP támogatás
- Déli interfész: OpenFlow, P4, NETCONF, SNMP, BGP, RESTCONF and PCEP
- Északi interfész: gRPC, REST API

Ez szintén a Linux Foundation Networking kezei alatt fut.



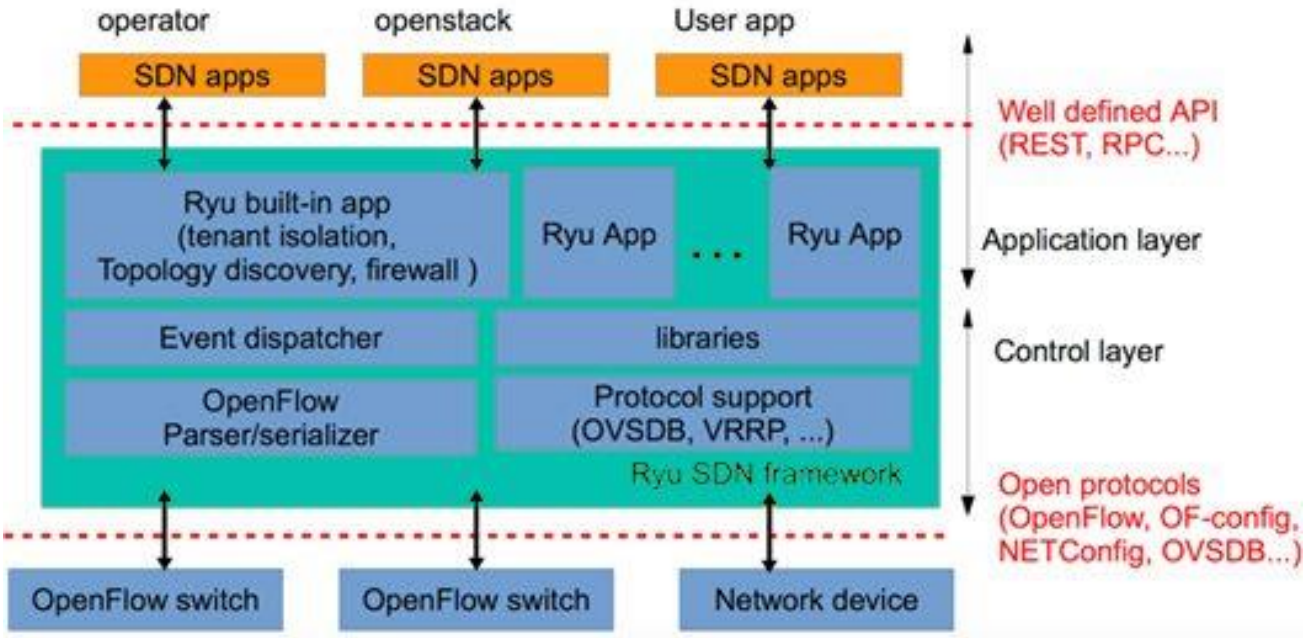
5.3. Ryu

Neve Japán eredetű, lefordítva angolra a ryu flow-t jelent. Ez az eddigiektől logikában eltérő konténer. A hálózatot python nyelven lehet leírni és azt az interpreter fordítja és indítja el. Ebből kifolyólag ha bármi változást akarunk eszközölni a hálózatban mindig újra kell indítani a kontrollert, de viszont ezt kevesebb, mint egy szekundum alatt képes megtenni, ezért a felhasználók számára ez a kiesés észrevehetetlen.

Főbb jellemzők:

- Nagy flexibilitás
- Nincs cooperatív cluster kezelés
- Jól definiált API-k
- Külső eszközök lehetővé teszik új példányok használatát (pl.: Zookeeper)
- Déli interfész: Openflow, NETCONF, OF-config… (mint az eddigiek)
- Északi interfész: REST API

Teljes mértékben az érdeklődők közösségének a szabad fejlesztése alatt van.

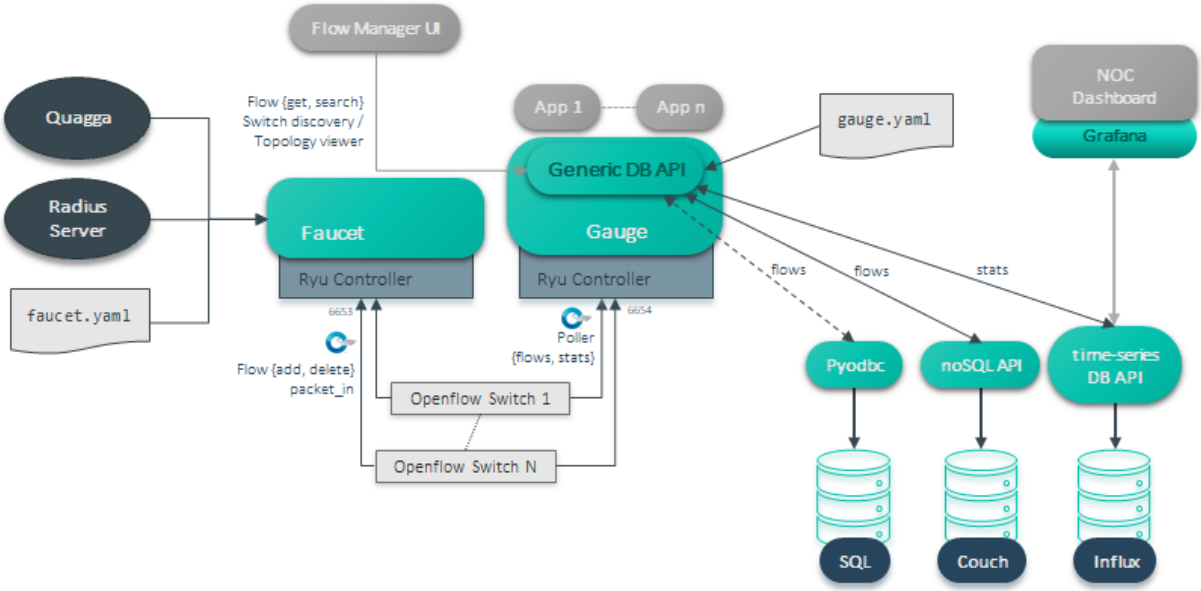


5.4. Faucet

Alapjaiban a Ryu kontrollerre épít. Fentebbi absztrakciós szintekbe viszi a hálózat kezelését, ezért könnyebben le lehet írni a hálózat állapotát, a kapcsolatokat.

Főbb jellemzők:

- Akár bare-metal vagy konténerben is telepíthető
- Idempotens cluster
- Nincs cooperatív cluster támogatás, csak külső eszközökkel
- Gyors újraindulás
- Déli interfész: Openflow, natív BGP
- Északi interfész: YAML



Mivel ezzel a kontrollerrel dolgoztam szeretném ezt részletesebben is bemutatni.

6. Használt kontroller bemutatása

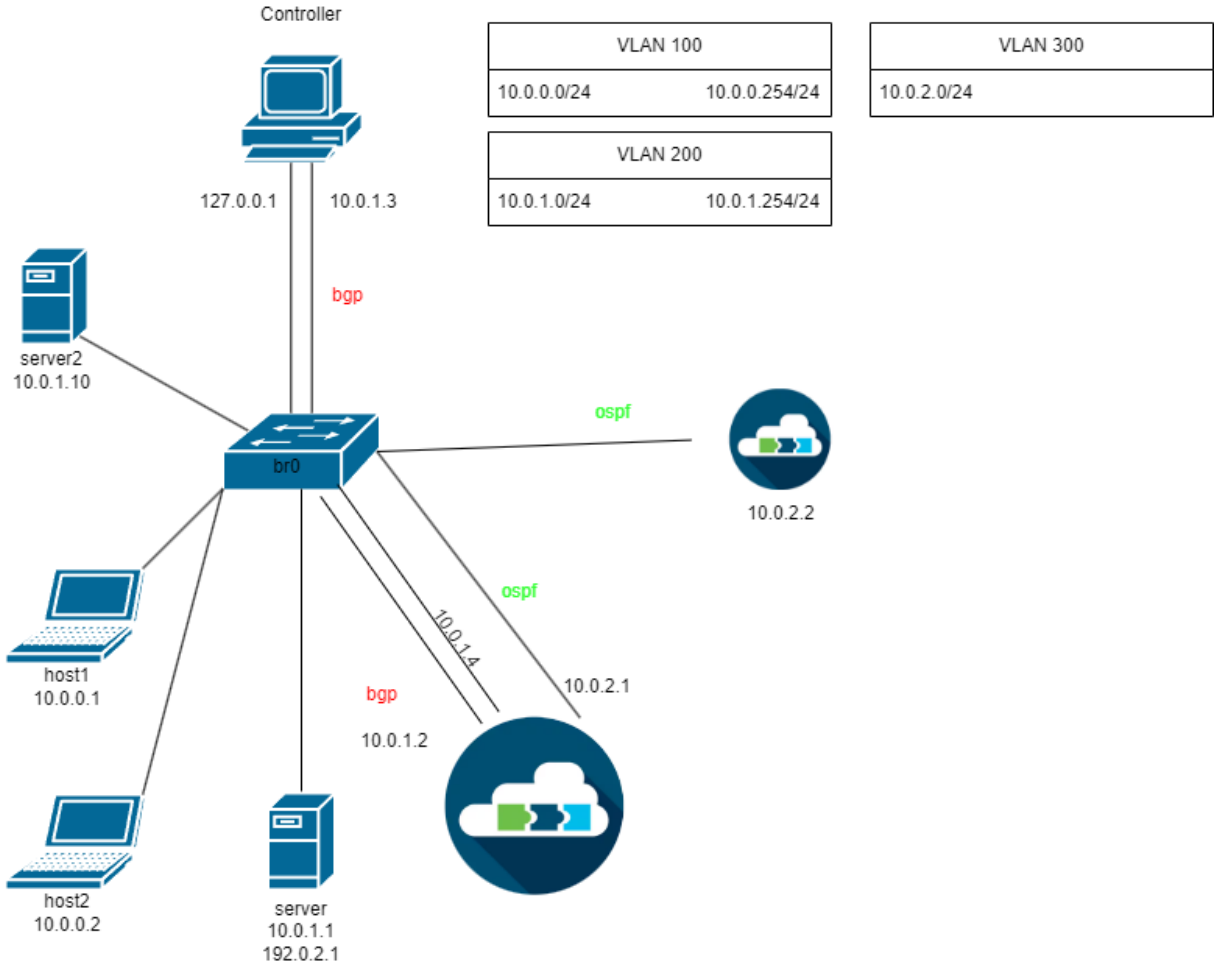
A bemutatáshoz a fentebbi ábrát fogom segítségül venni és ennek segítségével végig megyünk a kontroller pontos működésén és a komponenseinek a felelősségén.

Faucet: A kontrollernek ez a része az agy. Itt történik minden hálózati forgalom módosítás. Egy ryu kontrollerre épül rá. Ezen egy applikáció helyezkedik el, ami segítségével egyszerűen egy YAML fájlban leírva a hálózatot tudjuk leírni a kívánt kapcsolatokat, ez az északi interfésze. Ennek a segítségével tudunk még a kontrollerhez csatlakoztatni külső eszközöket is, mint például egy radius szervert.

Gauge: Felépítésében nem sokban különbözik a faucet-tól, majdnem megegyeznek. A különbség az, hogy míg a faucet kezeli a hálózat irányítását a gauge semmi változtatást nem visz végbe, csak egy azonos állapotot tart fent a hálózatról. Jól definiált API-k segítségével ez az elem arra szolgál, hogy a hálózatunkból információt nyerhessünk ki. Egy real-time adatbázisban el tudjuk tárolni a belőle kinyert adatokat így ennek segítségével erre monitoring rendszert is képesek vagyunk felépíteni. Ezeket mi is külön beállíthatjuk, de a telepítés során a kontroller ezzel egybe építve telepítődik.

7. Valós szcenárió SDN használatával

A faucet kontroller segítségével könnyedén képes voltam egy a kontroller által vezérelt hálózatot felépíteni. A hálózat a lentebbi ábrán látható:



A teljes hálózatot virtualizálva építettem fel. A hostok és szerverek jelen esetben hálózati namespacek egy gépen belül.

Itt látható egy shell script ami segítségével felépítettem a hálózatot:

```
#!/bin/bash
as_ns () {
    NAME=$1
    NETNS=faucet-${NAME}
    shift
    sudo ip netns exec ${NETNS} $@
}

create_ns () {
    NAME=$1
    IP=$2
    NETNS=faucet-${NAME}
    sudo ip netns add ${NETNS}
```

```
    sudo ip link add dev veth-${NAME} type veth peer name veth0 netns ${NETNS}
    sudo ip link set dev veth-${NAME} up
    as_ns ${NAME} ip link set dev lo up
    [ -n "${IP}" ] && as_ns ${NAME} ip addr add dev veth0 ${IP}
    as_ns ${NAME} ip link set dev veth0 up
}

create_ns host1 10.0.0.1/24
create_ns host2 10.0.0.2/24
create_ns server 10.0.1.1/24
create_ns server2 10.0.1.10/24
create_ns bgp 10.0.1.2/24
as_ns host1 ip route add default via 10.0.0.254
as_ns host2 ip route add default via 10.0.0.254
as_ns server ip route add default via 10.0.1.254
as_ns server2 ip route add default via 10.0.1.254
as_ns bgp ip route add default via 10.0.1.254

sudo ovs-vsctl add-br br0 \
-- set bridge br0 other-config:datapath-id=0000000000000001 \
-- set bridge br0 other-config:disable-in-band=true \
-- set bridge br0 fail_mode=secure \
-- add-port br0 veth-host1 -- set interface veth-host1 ofport_request=1 \
-- add-port br0 veth-host2 -- set interface veth-host2 ofport_request=2 \
-- add-port br0 veth-server -- set interface veth-server ofport_request=3 \
-- add-port br0 veth-server2 -- set interface veth-server2 ofport_request=4 \
-- add-port br0 veth-bgp -- set interface veth-bgp ofport_request=5 \
-- set-controller br0 tcp:127.0.0.1:6653 tcp:127.0.0.1:6654

as_ns server ip address add 192.0.2.1/24 dev veth0


sudo ip link add veth-faucet type veth peer name veth-faucet-ovs
sudo ovs-vsctl add-port br0 veth-faucet-ovs -- set interface veth-faucet-ovs
ofport_request=5
sudo ip addr add 10.0.1.3/24 dev veth-faucet
sudo ip link set veth-faucet up
sudo ip link set veth-faucet-ovs up

sudo ip link add dev veth-ospf-bird type veth peer name veth1 netns faucet-bgp
sudo ip link set dev veth-ospf-bird up
as_ns bgp ip addr add dev veth1 10.0.2.1/24
as_ns bgp ip link set dev veth1 up

sudo ip link add dev veth-bgp-false type veth peer name veth2 netns faucet-bgp
sudo ip link set dev veth-bgp-false up
as_ns bgp ip addr add dev veth2 10.0.1.4/24
as_ns bgp ip link set dev veth2 up
```

```
sudo ovs-vsctl add-port br0 veth-bgp-false -- set interface veth-bgp-false
ofport_request=6

sudo ovs-vsctl add-port br0 veth-ospf-bird -- set interface veth-ospf-bird
ofport_request=7

create_ns ospf 10.0.2.2/24
as_ns ospf ip route add 10.0.0.0/24 via 10.0.2.1
sudo ovs-vsctl add-port br0 veth-ospf -- set interface veth-ospf
ofport_request=9


as_ns bgp bird -P /run/bird-bgp.pid
sudo systemctl reload faucet
```

Ez létrehozza a hostokat különböző namespacekben, hozzá ad mindegyikhez az ábrán megfelelő interfészt és ezekre felvesz privát IP címeket. Majd ezeket hozzáköti egy Open vSwitch-hez és beállítja, hogy ennek a switchnek a kontrollerje a mi faucaet kontrollerünk legyen.

A hostok és szerverek mezei számítógépeket szimulálnak, amik a hálózaton kommunikálni fognak, nincsen kitüntetett szerepük. A 10.0.1.2-es címen egy BIRD routing daemon fog futni. Ennek segítségével fog tudni megtanulni a kontrollerünk olyan utakat is, amik nem mi írunk le a YAML fájlban, hanem egy külső hálózat, amit nem a kontroller vezérel. A 10.0.2.2-es címen pedig egy FRR routing daemon fog futni, ami a külső hálózatot fogja szimulálni. Ez fog a BIRD-el OSPF-en kommunikálni Majd fogjuk látni, hogy így a kontroller BGP-n keresztül a BIRD-től képes lesz megtanulni ezt az utat.

Az így felépült hálózatban még így nincsenek kapcsolatok, mivel nincsen semmi bejegyzés a switch-ünkben ami alapján a csomagok továbbítását intézhetné. Ahhoz, hogy mégis történjen kommunikáció fel kell konfigurálnunk a kontrollerünket. Itt látható az ehhez használt YAML fájl:

```
vlans:
    hosts:
        vid: 100
        description: "vlan for clients"
        faucet_mac: "00:00:00:00:00:11"
        faucet_vips: ["10.0.0.254/24"]
    servers:
        vid: 200
        description: "vlan for servers"
        faucet_mac: "00:00:00:00:00:22"
        faucet_vips: ["10.0.1.254/24"]
    ospf:
```

```
        vid: 300
        description: "vlan for ospf"
        faucet_mac: "00:00:00:00:00:33"
        faucet_vips: ["10.0.2.254/24"]

routers:
    router-hosts-servers:
        vlans: [hosts, servers]
    bird:
        bgp:
            vlan: servers
            as: 65000
            port: 9179
            routerid: '10.0.1.3'
            server_addresses: ['10.0.1.3']
            neighbor_addresses: ['10.0.1.2']
            neighbor_as: 65001
dps:
    br0:
        dp_id: 0x1
        hardware: "Open vSwitch"
        interfaces:
            1:
                name: "host1"
                description: "host1 network namespace"
                native_vlan: hosts
            2:
                name: "host2"
                description: "host2 network namespace"
                native_vlan: hosts
            3:
                name: "server"
                description: "server network namespace"
                native_vlan: servers
            4:
                name: "server2"
                description: "server2 network namespace"
                native_vlan: servers
            5:
                name: "faucet"
                description: "faucet dataplane connection"
                native_vlan: servers
            6:
                name: "bgp-false"
                description: "bgp plus one to server"
                native_vlan: servers
            7:
                name: "bird-ospf"
                description: "frr to bird"
```

```
        native_vlan: ospf
    8:
        name: "bgp"
        description: "bgp ns to server vlan"
        native_vlan: servers
    9:
        name: "ospf"
        description: "ospf"
        native_vlan: ospf
```

A YAML lehetővé teszi számunkra, hogy ember számára is olvasható módon írjuk le a hálózatunkat. Ez összesen annyit csinál most, hogy felvesz 3 VLAN-t, amikbe majd kerülnek az eszközök. Ezeknek megadja a default gateway címeket. Majd egy Inter-VLAN routingot engedélyezünk a hosts és servers VLAN-ok között plusz felkonfiguráljuk a BGP utat, amin keresztül majd a BIRD daemonnal fog tudni kommunikálni. Ezek után csak a kontrollált switchünk-höz felvesszük, hogy melyik interfészen melyik eszköz van és ezeket behelyezzük a VLAN-okba.

Ha ez megvan akkor a `sudo systemctl faucet reload` paranccsal újra tudjuk indítani a kontrollert és az openflow bejegyzéseink létre is jönnek.

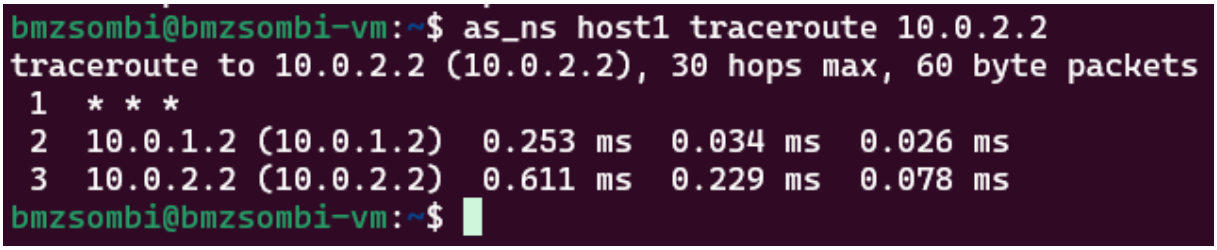
Az `as_ns bgp birdc show route export faucet` paranccsot lefuttatva láthatjuk, hogy melyek azok az utak, amiket a BIRD exportál a kontrollerünknek:

```
bmzsombi@bmzsombi-vm:~$ as_ns bgp birdc show route export faucet
BIRD 1.6.8 ready.
10.0.2.0/24         dev veth1 [myOSPF 2024-05-23] * I (150/5) [10.0.1.2]
192.0.2.0/24        via 10.0.1.1 on veth0 [static1 2024-05-23] * (200)
bmzsombi@bmzsombi-vm:~$
```

Itt jól látható, hogy az első bejegyzés egy OSPF út. Ez az, amin keresztül a daemonunk a másikkal tud kommunikálni.
Megnézhetjük azt is, hogy melyek azok az utak, amiket a daemon tanul meg a kontrollertől:

```
bmzsombi@bmzsombi-vm:~$ as_ns bgp birdc show route protocol faucet
BIRD 1.6.8 ready.
10.0.1.0/24         via 10.0.1.254 on veth0 [faucet 2024-05-23 from 10.0.1.3] * (100) [i]
bmzsombi@bmzsombi-vm:~$
```

Ezek után, ha a host1-től indítunk egy traceroute-ot a 10.0.2.2 címre, ami az FRR daemon címe, azaz egy külső, nem SDN kontroller alatt által vezérelt hálózat címe, szépen kivehető, hogy a csomagok a BIRD routeren keresztül kerülnek kiküldésre:



Így sikerült felépítenünk egy SDN kontroller által vezérelt hálózatot, ami képes a külvilággal, kommunikálni.

8. További lehetőségek

A féléves munkámban csak az SDN megismerésével foglalkoztam leginkább. A jövőbeli célkitűzés, hogy egy egységes eszközt készítsek, aminek segítségével egyaránt tudunk változtatásokat végezni az SDN hálózatban, valamint a „régimódi” hálózatban is.