Nagy Márton JNTVXJ
Dengyel Bendegúz T6HC11

Önálló laboratórium jegyzőkönyv a

# Hibrid SDN hálózat kialakítása

és

# Hibrid SDN hálózat automatizálása

témákban.

A témán Dr. Zsóka Zoltán konzulens vezetésével Nagy Márton és Dengyel Bendegúz hallgatók dolgoztak. Köszönet Varga Györgynek is, aki a környezetet biztosította, és Szabó Csabának a rendelkezésre állásáért.

Nagy Márton témája a *Hibrid SDN hálózat kialakítása*, Dengyel Bendegúz témája a *Hibrid SDN hálózat automatizálása* volt.

A munkához felhasznált, hivatkozott és általunk készített tartalmak ezen a linken[1] érhetők el.

[1] https://bmeedu-my.sharepoint.com/personal/dengyel_b_edu_bme_hu/_layouts/15/onedrive.aspx?ga=1&id=%2Fpersonal%2Fdengyel%5Fb%5Fedu%5Fbme%5Fhu%2FDocuments%2F%C3%96nlab

# Tartalom:

# Téma kiírás: Hibrid SDN hálózat kialakítása

Félév: 2021-2022/II.
Konzulens: Dr. Zsóka Zoltán
A szoftveralapú hálózatok (SDN) összekapcsolása hagyományos hálózatokkal kihívást jelent, ha közös adminisztrációt szeretnénk felettük megvalósítani. Egy ilyen hibrid hálózatban az SDN kontrollereknek együtt kell működniük a hagyományos routerekkel, támogatva az ott futó routing protokollokat. Ez a kontroller szoftverének kiegészítését igényli.

A hibrid hálózatot többféle virtualizálációs technológia ötvözésével alakítottuk ki. A feladat ennek beüzemelése és a hiányzó szoftverelemek elkészítése.

A munka egy korábban készült, de a végső tesztelésig el nem jutott munka befejezését, illetve folytatását jelenti.

# Téma kiírás: Hibrid SDN hálózat automatizálása

Félév: 2021-2022/II.
Konzulens: Dr. Zsóka Zoltán
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 működésbe 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:

- hibrid hálózatok kialakítása virtualizált környezetben
- kontroller NBI megismerése
- általános automatizáló eszköz (pl. ansible) megismerése és alkalmazása a virtualizált hálózati eszközöknél
- az NBI és az automatizálás integrált kezelése

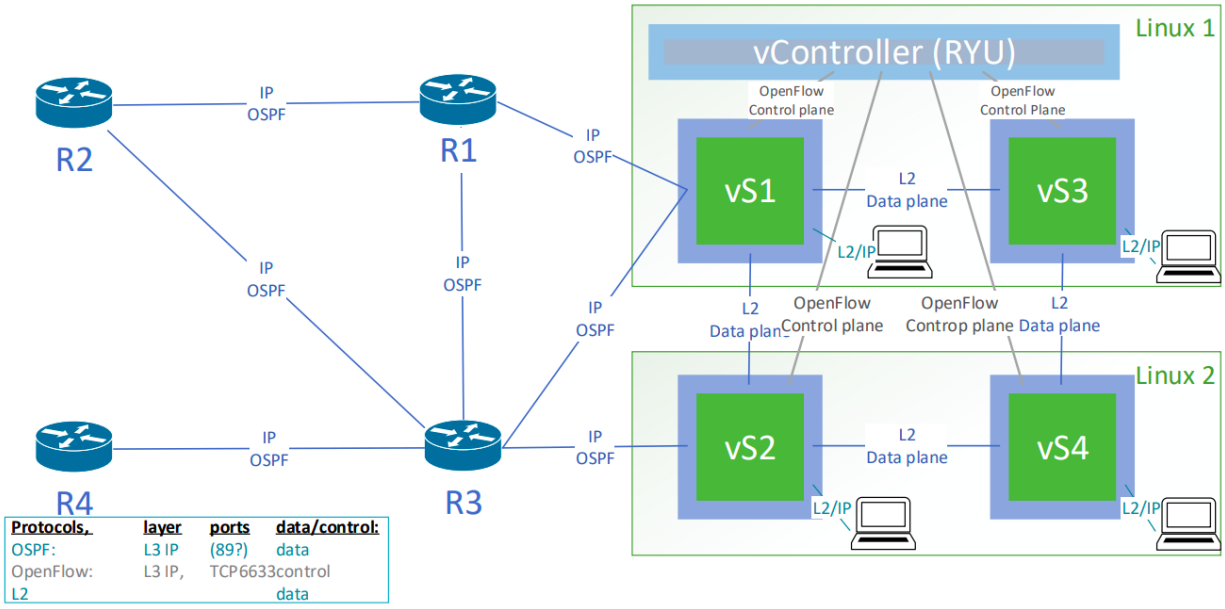
# A teszthálózat és környezete

A tervünk a fent leírt 2 téma feldolgozása, a környezet kialakítása, és a témában való mélyebb elmélyedés volt. Az első lépés a teszthálózat és környezetének kialakítása volt, mely Szabó Csaba munkája[2] (önálló laboratórium és diplomamunka is). Ezt a környezetet állítottuk fel, és fejlesztettük tovább, mivel mindkettőnk témájának alapja ez a környezet, és a méréseket itt terveztük elvégezni.

Ez a környezet részletesen le van írva, topológiákkal és más ábrákkal is szemléletesen dokumentálva van Szabó Csaba diplomamunkájában, de röviden összefoglalva a következő:

- Egy tanszéki (HIT) szerveren futó ESXi (VMware) virtualizációs környezetben
- fut 2 db Linux virtuális gép (Ubuntu 18.04): linux1, linux2
- 4 db Cisco router image (szintén virtualizálva)
- ezzel egy hagyományos
- és egy SND hálózatot alkotva
- közösen egy ún. hibrid hálózatot alkotva.
- Az SDN hálózat 4 db OVS-ből (Open Virtual Switch), és
- 1 db Ryu SDN kontrollerből áll.
- Az SDN hálózatot alkotó OVS-ek és kontrollerük Docker konténerként fut a 2 Linuxon úgy, hogy
- a linux1 nevű VM-en fut a Ryu kontroller, valamint az OVS1 és OVS3
- a linux2 VM-en pedig az OVS2 és OVS4
- amik mind macvlan technológiával kommunikálnak egymással az ESXi switchen, hogy a kialakított környezet egy valóságos hálózathoz legjobban hasonlítson működésben,
- a (Cisco) routerek is az ESXi switchen keresztül, de számukra transzparensen kommunikálnak (ez a switch minden hálózati eszköz számára transzparens). Ennek szemléltetésén az **esxitopo.pdf** segít.
- Ezen felül a hosztokat Alpine lightweight linux konténerekkel valósítjuk meg (még folyamatban van), melyek célja, hogy végpontként a hálózatunk működését tudjuk tesztelni.

[2] https://diplomaterv.vik.bme.hu/hu/Theses/Halozati-eszkozok-osszekotese-SDN-halozatokkal




*A hálózat vázlatos felépítése (Szabó Csaba szakdolgozata, 4.4 ábra)*

A tervünk megvalósításában a félév végére a környezet felállításáig jutottunk. Jelenleg a 2 Linux (bennük az 5 SDN konténer) és 4 router fut. Internet van a gépeken, és részben látják is ezek az SDN eszközök egymást (egy gépen belül képesek kommunikálni egymással, valamint az adott gépen futó alpine konténer „hoszttal"). Számos script készült el, melyek a környezetet automatizálják, és sok hibajavítás is történt. Ezekről később részletesen beszámolunk. Célunk az lett, hogy a környezetet kijavítsuk, és részletes dokumentációval lássuk el, hogy ha más is dolgozna ezen a környezeten, akkor gyorsan (akár fél óra alatt is) fel tudja állítani azt, mely robusztusan működik. Ehhez készítettük a részletesen dokumentált **„README BUILDING ENVIRONMENT.txt"** nevű segédletet, mely a szükséges szoftverek telepítésétől kezdve, a környezet felállításának egyes lépésein végigmegy, egészen odáig, ahol most tartunk (macvlan kialakítása). Ilyen lépések pl.: az egyes scriptek mire valók, hogy lehet futtatni őket, milyen sorrendben érdemes haladni, hogy gyorsan elkészüljön a hálózatunk. Lent ezeket is részletesen kifejtjük.

# Felmerülő problémák és megoldásaik

Mint a fenti vázlatából is látszik, ez egy rendkívül összetett környezet, mely nagyon sok technológiát ötvöz (virtualizálás, Linux OS kezelése, Docker, SDN eszközök, hagyományos routerek stb.). A környezetet Szabó Csaba is épp befejezte, szakdolgozatában ki is tért rá, hogy sok helyen kell javítani rajta.

A félév első heteiben megismerkedtünk ezzel, és elkezdtük a routerek felkonfigurálását, mellyel hamar végeztünk, bár itt is előjött, hogy nem tudtuk, hogy az ESXi switch miatt egyes interfészek a routereken nem képesek VLAN-ra (**esxitopo.pdf**), így nem rögtön lett kész azok konfigurálása sem, valamint a konfiguráció, mely Csaba dolgozatában volt valószínűleg nem a legfrissebb volt, mert a környezetünkben nem működött. Végül az „**R1 config.txt**”, „**R2 config.txt**”, „**R3 config.txt**”, és „**R4 config.txt**” konfigurációs fájlokat készítettük.

Több script file is módosult a szakdolgozat befejezte óta, melyet Csaba rendelkezésünkre bocsájtott. A dolgozata és a mellékelt anyagok (scriptek, ábrák stb.) alapján a **GetStarted.txt** szerint kezdtünk el dolgozni, mely 3 pontból áll: a **linux_config.sh** script futtatása, a szükséges fileok felmásolása a Ryu kontrollerre, majd a kontrolleren adott állományok futtatása, így szimulálva bizonyos működéseket.

Első megoldandó feladat az internet elérés volt a Linux VM-eken, melyet viszonylag hamar megoldottunk, erre szolgál a **vlan_up.sh** script, mely a 999 vlan-t használja internetelérésre (a routerek is ezen keresztül érik el az internetet).

Ezután futtattuk a kapott **linux_config.sh**-t, mely rengeteg hibát dobott, és hiányos volt sok helyen. Pár hét után, miután Csabát elértük, kiderült, hogy neki van egy frissebb scriptje, mely már sikeresen lefutott. Igaz, később ebben is kellet számos helyen módosítani.

A 2 Linux VM-en nehézséget okozott, hogy nem lehetett a konzolra másolni, onnan kimásolni. A megoldásunk az lett, hogy a felmásolandó nagyobb állományokat, azaz ami pár sornyi parancs, **.iso** file-á alakítottuk. Ezeket felcsatoltuk a gépre, majd onnan átmásoltuk őket más mappába (és még pár egyéb lépés, melyek a segédletünkbe leírtunk). Ezeket a műveleteket végül az **iso_import.sh** scripttel automatizáltuk. A **scriptify.sh** pedig a felcsatolt **.sh** állományokat teszi futtathatóvá Linuxon („/” és „\” jelek cseréje, futtatási jog biztosítása).

Minután már a frissebb **linux_config.sh** script lefutott, mely a docker telepítéséért, az egyes konténerek futtatásáért, illetve a köztük lévő kapcsolat kialakításáért felelt, észrevettük, hogy pár óra futás után „elmegy a net” a Linuxokról. Sok vizsgálódás után kiderült, hogy internet van a gépeken, de a konténerek hibás konfigurációja miatt hurok keletkezett a hálózatban, ami miatt annyira ellepik a csomagok a hálózatot (több 10 millió csomag néhány másodperc alatt), hogy pl. ping sem jut ki a gépről. Végül a scriptben javítottuk a hibás konfigurációt, és ez a hálózati flood-ot megjavította.

További probléma volt, hogy ha a VM-eket lekapcsoltuk, vagy újraindítottuk, akkor a korábbi konténereket, docker-ben kialakított hálózatokat már nem tudtuk elindítani, mert mindenféle daemon processek befoglalták azokat, akármilyen docker prune parancsokkal próbálkoztunk. Ennek „megoldása” az lett, hogy a gépeket újból telepítettük, a korábban elvégzett munkákat végrehajtottuk egészen odáig, hogy a linux_config lefut, ekkor snapshotot készítettünk a gépekről (és még régebben a kész routerekről is), és így ettől a ponttól kezdve futtattuk a config scriptet. Ezután ha valami „elromlott” a dockerben, akkor egyszerűen visszaállítottuk a snapshotot, és megint futtattuk a scripteket. Ez azért is jobb volt, mert időközben valami dockeres probléma okán (melyet nem sikerült megtalálni) a VM-ek tárhelye 2-3 nap alatt teljesen megtelt, gyakorlatilag használhatatlan lett, egy egyszerű, üres file-t sem lehetett létrehozni rajtuk. Ekkor mindig visszaállítottuk a snapshotokat, és újból volt 2-3 nap lehetőség munkára…

A GetStarted.txt-ben a következő pont a szükséges python, és egyéb fileok felmásolása volt a konténerbe. Ez nagyon súlyos probléma volt, nagyon sok módszert megpróbáltunk, végül a Dockerfile-os **docker build** paranccsal sikerült leküzdeni, mely már tényleg felmásolta a szükséges fájlokat. Ekkor a kontrollert is a fent említett 999 VLAN segítségével kapcsoltuk az internetre, mivel szükséges volt telepíteni erre is bizonyos csomagokat. Itt a megfelelő verziójú pip (python csomagkezelő, mivel a Ryu kontroller python alapú) telepítése okozott nehézséget, mivel annyira régi volt a rajta lévő pip, hogy nem akarta frissebb verzióra frissíteni magát, és rendszeresen hibát dobott, mikor futtatunk volna akármilyen python filet. Végül sikerült ezt is leküzdeni a **ryu_init.sh** scripttel, mely még ezen felül az internetet is felkapcsolja kontrolleren, és pár hasznos eszközt is letölt (pl. ping, traceroute stb).

A legutolsó szembejövő feladat a konténerek közti kommunikáció megvalósítása macvlan segítségével, mely még folyamatban van. Itt a legfőbb problémát a macvlanhoz tartozó internetes források és leírások inkonzisztenciája okozta. Először azt hittük, úgy dolgoztunk vele, mintha vlan egy altípusa lenne. Nagyjából a munkánk felénél jöttünk rá, hogy a macvlan inkább al-interfészként működik, ami lehetővé teszi, hogy egy fizikai interfészre több konténert kössünk. Az Linuxokon belül a konténerek közötti kommunikáció ennek segítségével már működik, azonban a különböző Linuxokon lévő konténerek közötti kommunikáció megvalósítása még hátra van.

A macvlan hálózatok létrehozása közben észrevettük, hogy az OVS-eken keresztül nem érik el egymást a host-ok. Az OVS-ek konfigjában az eredetileg Csabától átvett, majd általunk kiegészített konfigurációkat használtuk, ami viszont így hibásnak bizonyult. Miután töröltük az OVS-ek konfigjait, a hálózati forgalom helyreállt. A korábban használt/jelenleg törölt konfiguráció megtalálható a **docker_env_start.sh** scriptben, kikommentezve.

Végül említendő még, hogy a könnyebb munka érdekében felállítottunk egy hasonló környezetet saját számítógépen, mely a 2 Linuxból állt, azokon az 5 konténer (+2 Alpine konténer is, mely a namespace hosztok helyett lett, mivel namespaceket nem sikerült felállítani), de a 4 router nem lett kialakítva. Először az SDN hálózatot szerettük volna kialakítani, hogy megfelelően működjön. Ez Oracle VirtualBox környezetben lett kialakítva (A VMware-es ESXi helyett), és a gépek közti

kommunikáció NATnetwork megoldással történt. Itt a tesztelés szempontjából jelentősen könnyebb dolgunk volt, mivel a copy/paste gondok, internet felkapcsolása stb. tárgytalanná vált.

Végül a **linux_config.sh** scriptet ketté szedtük egy **docker_env_setup.sh** és **docker_env_start.sh** scriptre. Az előbbi a dockert installálja, és a szükséges docker imageket letölti (ezek nagyobb méretűek, főleg a docker). Lefutás után készítettünk snapshotot a két Linux VM-ről, így a konténerek futtatását, macvlan-ok kialakítását és egyéb konfigurációkat elvégző utóbbi script már gyorsan (<10mp) alatt lefut, és nem kell hozzá internet sem.

# Scriptek leírása

Alább ismertetjük a használt scripteket:

- **docker_env_setup.sh**: felel a Docker letöltéséért, feltelepítéséért, illetve a Ryu kontroller, OVS, és Alpine imageket tölti le. Ennek futása során nagyobb mennyiségű adatot tölt le, ezért érdemes a futtatása után egy snapshotot készíteni az adott környezetről. Hívása: **sudo ./ docker_env_setup.sh linux1** (vagy linux2). Szabó Csaba **linux_config.sh** scriptje alapján készült, annak első felét valósítja meg, kiegészítve az Alpine image letöltésével, illetve a Ryu kontrollert Dockerfile alapján készíti. A **linux_config.sh** scriptet 2 scriptre bontottuk: erre (**docker_env_setup.sh**) és a **docker_env_start.sh** scriptre.

- **docker_env_start.sh**: a korábbi **linux_config.sh** 2. része, mely felel a macvlan-ok létrehozásáért, a konténerek elindításáért, valamint a konténerek macvlan hálózatba kötéséért. Hívása: **sudo ./ docker_env_start.sh linux1** (vagy linux2). Fontos, hogy az OVS parancsokat kikommentelve találjuk meg benne, mert ezek kiadása megakadályozta a kapcsolatot a konténerek között még azonos VM-en belül is.

- **Dockerfile**: segítségével sikerült fájlokat felmásolni a Ryu konténerre. Fontos, hogy amikor futtatjuk a **docker_env_start.sh** scriptet, akkor egy mappában álljunk a **Dockfile**-al. A benne definiált **/code** mappa tartalmát másolja fel a konténerre. A **/code** mappa is egy mappában legyen a **Dockfile**-al. Nem kell külön hívni, **docker_env_start.sh** hívja.

- **iso_import.sh**: feladata a VM konzol menüjén keresztül felcsatolt **.iso** imagek felcsatolása és átmásolása a **/mount** mappába (amit létrehoz). Ezután az lemezkép lecsatolását is elvégzi. Ezzel valósítottuk meg a “copy/paste” műveleteket. Később a https://termbin.com/ segítségével is dolgoztunk. Ha shell scriptet (**.sh**) csatolunk fel, után a **scriptify.sh** scriptet is meg kell hívni. Hívása: **sudo ./ iso_import.sh linux1** (vagy linux2).

- **linux_config_3.sh**: a **docker_env_setup.sh** és **docker_env_start.sh** közvetlen elődje, ebben már benne van az Alpine, az OVS parancsok ki vannak kommentelve, illetve jelentősen olvashatóbbá van téve, teljes mértékben megegyezik a futása a 2 utód scripttel. Meghagytuk, hiszen vissza lehet térni hozzá/ismét kiindulni belőle szükség esetén. Hívása nem szükséges már, de ugyan úgy kell, mint utódait.

- **"R1 config.txt":** az R1 router konfigja. Name Servert, Default GW-t állít be, a GigabitEthernet6 interfészen létrehozza a szakdolgozatban definiált VLAN-okat, majd OSPF routing bejegyzéseket állít be. Mi kézzel vittük be a CLI-be (a router konzolja). A többi router konfig is hasonló módon működik, értelemszerűen mindegyik a hozzárendelt IP címekkel, OSPF routokkal és VLAN-okkal.

- **ryu_init.sh**: a Ryu kontroller konténer első lépéseit végzi el, melyek az internet bekötése (a 999 VLAN-non keresztül, az ESXi környezetben ezen keresztül érhető el a világ), a felmásolt tartalma megfelelő helyre másolása (lásd: **GetStarted.txt**). Ezentúl feltelepíti a megfelelő verziójú pip python csomagkezelőt, és még pár hasznos eszközt (pl. net-tools). Hívása (a kontrolleren belül): **./ryu_init.sh**

- **scriptify.sh**: a felmásolt scriptfájlokon kell futtatni, mert futási, írási, olvasási jogot biztosít azokra, illetve egy karaktermódosító műveletet végez, mely gyakran megakadályozza a Windowson írt scriptek futását Linux környezetben. Hívása: **sudo ./ scriptify.sh <filename>**

- **vlan_up.sh**: a 999 VLAN-ra kapcsolja a gépet, melyen az internet érhető el. Minden bootolás után ki kell adni. Hívása: **sudo ./ vlan_up.sh linux1** (vagy linux2).

- **README BUILDING ENVIRONMNENT**: ez nem script file, de ebben található meg a környezet összeállításához szükséges parancsok sorrendje, és nagyon sok egyéb, hasznos információ a környezetben felmerülő, gyakran előforduló hibák elhárításával kapcsolatban. Érdemes egy használati útmutatóként kezelni a környezethez.