Hálózatok építése és üzemeltetése

A hálózat mint platform (SDN)

Hálózatok gyors helyzetkép

- Switch-ek különböző gyártóktól + interfészek, linkek
- Különböző config felületek/support
- Ki kell találni a hálózat hogy működjön
- A switch-eket router-eket a hozzájuk való config felületen be kel állítani egyesével
- Ha mindent jól csináltuk, akkor működik a hálózat
- De kell egy szakértő csapat, aki a felmerülő hibákat folyamatosan figyeli és javítja.
- Kinek jó ez?

Mai téma

- A hálózat mint platform (SDN)
  - A hálózati eszközöket egységben kezelve egy (szoftver) platformot kapunk, amire alkalmazások fejleszthetők
- Gyökeres váltás a hagyományos gondolkodáshoz képest
- Miért kell ezt tanulni?

"SDN is still young. But understanding OpenFlow controllers and virtualization, and having these skills on your CV, will be crucial to stay relevant."

--- Joachim Bauernberger

Miért most?

- Voltak már próbálkozások
- Most mi változott?
- Google, datacenters “Networks are in my way.”
- Evolúciós dolgok (vízvezeték)
- Ehhez egy stabil hálózat kell, most jutottunk el oda, hogy kifejlődtek a stabil hálózatok világméretű szabványai
- Stabilitás, hibatűrés már nem akkora kihívás → extra szolgáltatások (vö. autóipar)

Hagyományos hálózatok

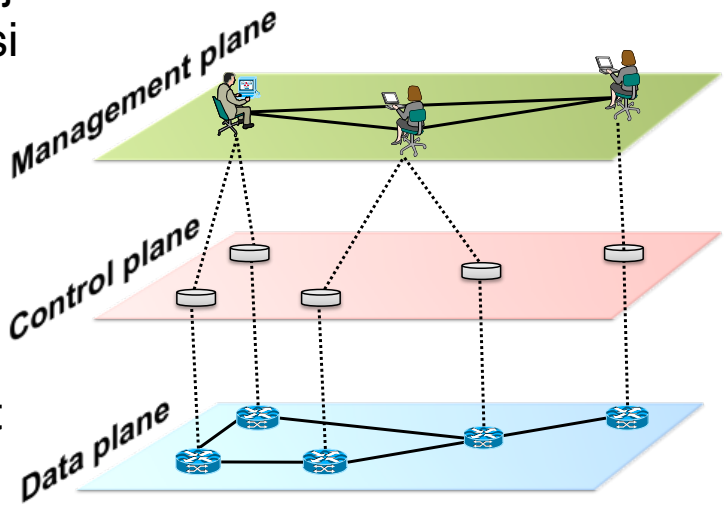
Összefoglalás

Hagyományos hálózatok

- Síkok (miért?)
- Milyen időskálán dolgozik és milyen szintűek a feladatai
- Pl. Szoftverfejlesztő cég:
- Menedzsment (milyen irányba menjen a cég)
- Projekt (adott projekt feladatainak összehangolása)
- Programozás: jól körülhatárolt lokális problémák megoldása gyorsan
- Pl. törvényhozás síkjai

Hagyományos hálózatok síkjai

- Adatsík: Az adatsík gyakorlatilag a hálózati (kapcsoló) eszközöket tartalmazza, amelyek feladata nem más, mint a csomagok hatékony és extrém gyors továbbítása.
- Vezérlősík: A vezérlő sík különböző protokolljai oldják meg, hogy a kapcsolóeszközökben levő kapcsolási logika (táblázatok) megfelelően működjön
- Menedzsment sík: A menedzsment síkban lehet megadni és nyomon követni a kívánt hálózati funkcionalitást.
- Tehát a menedzsment sík definiálja, a vezérlő sík kikényszeríti, az adatsík pedig végrehajtja a kívánt működést.

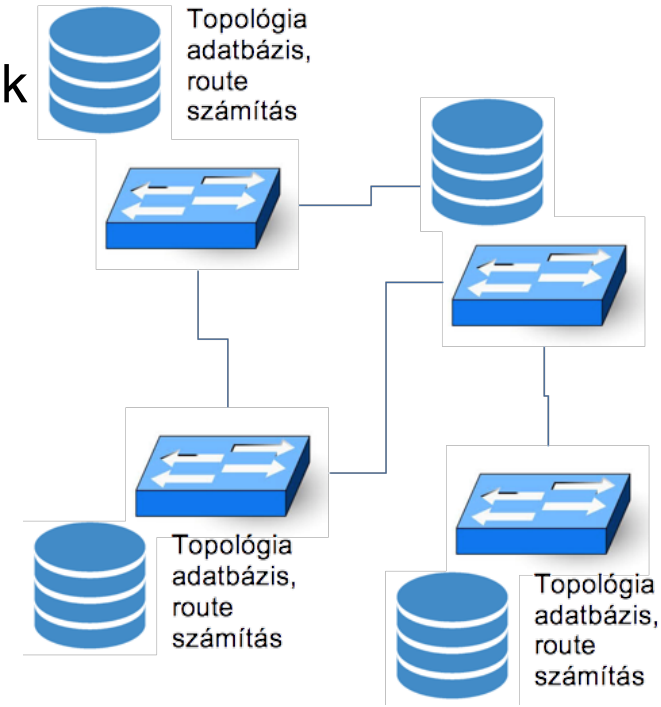


Síkok példa OSPF-el

- Menedzsment sík: legyen OSPF
- Vezérlősíkban az OSPF egyedek monitorozzák a topológiát, kiszámolják a legrövidebb utakat és beállítják az útválasztási táblákat
- Az adatsík meg a táblázatok alapján villámgyorsan pakolja a csomagokat.

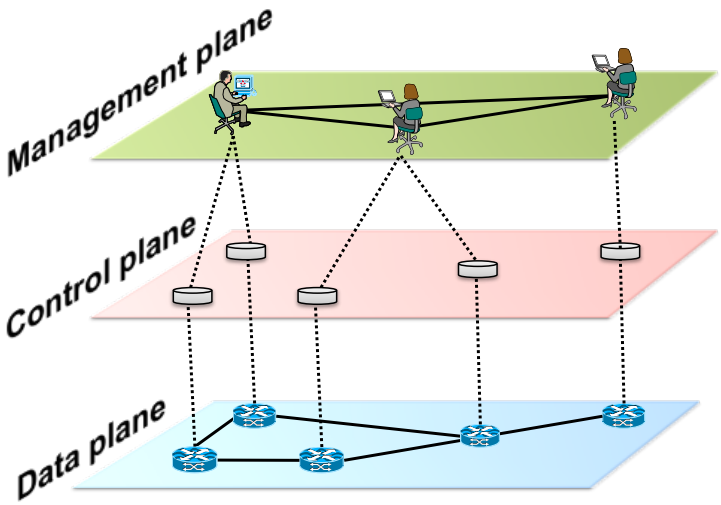
```
File  Edit  View  Terminal  Help

R4#show ip route ospf
     172.16.0.0/24 is subnetted, 6 subnets
O IA    172.16.133.0 [110/66] via 10.1.124.2, 00:09:59, FastEthernet1/0
                     [110/66] via 10.1.124.1, 00:09:59, FastEthernet1/0
O IA    172.16.101.0 [110/2] via 10.1.124.1, 00:09:59, FastEthernet1/0
O IA    172.16.102.0 [110/2] via 10.1.124.2, 00:09:59, FastEthernet1/0
O IA    172.16.103.0 [110/66] via 10.1.124.2, 00:09:59, FastEthernet1/0
                     [110/66] via 10.1.124.1, 00:09:59, FastEthernet1/0
     10.0.0.0/24 is subnetted, 3 subnets
O IA    10.1.13.0 [110/65] via 10.1.124.1, 00:09:59, FastEthernet1/0
O IA    10.1.23.0 [110/65] via 10.1.124.2, 00:09:59, FastEthernet1/0
R4#
```



Hagyományos hálózatok működési logikája

- Síkok megvalósítása, vezérlő - és adatsík egy eszközben (olyan mintha nem lenne projektvezető, vagy mindenki az lenne)
- Elosztott protokollok
- Drága (CAPEX, OPEX)
- Lassú innováció
- Extra funkciók middlebox-okban



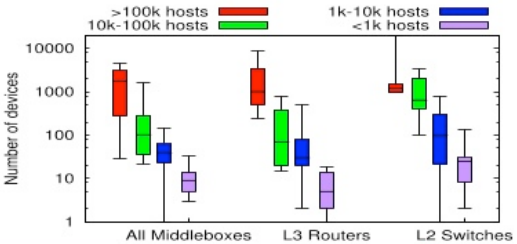
Middlebox-ok

Why middleboxes?

Data from a large enterprise

Type of appliance	*Number*
Firewalls	166
Intrusion detection	127
Media gateways	110
Load balancers	67
Proxies	66
VPN gateways	45
WAN Optimizers	44
Voice gateways	11
Total Middleboxes	*636*
Total routers	*~900*

Survey across 57 network operators



Critical for security, performance, compliance
But painful to manage

2

Az SDN alapkövei

Mi nem az SDN?

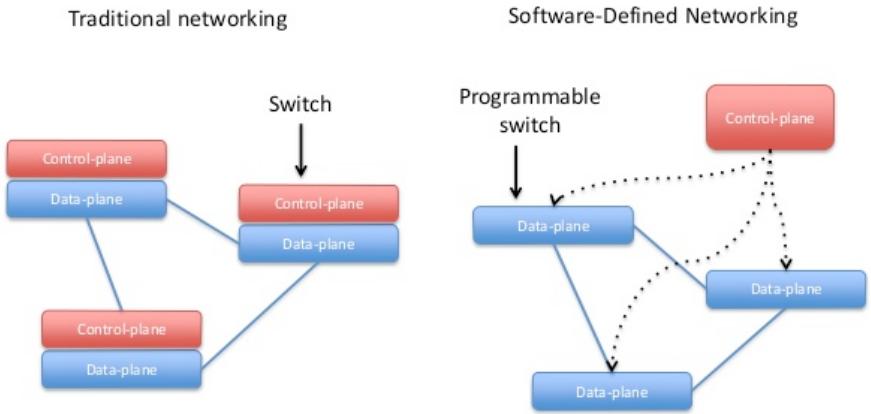
- Nem egy új hálózati működési elv
- Nem egy új algoritmus
- Nem egy új protokoll
- Az SDN nem változtatja meg a hálózatok alapvető lehetőségeit és korlátait

Mi akkor az SDN?

- Egy újfajta szemléletmód és technológia, amivel a hálózat funkcionalitását megadjuk, nyomon követjük és teljesítményét ellenőrizzük.

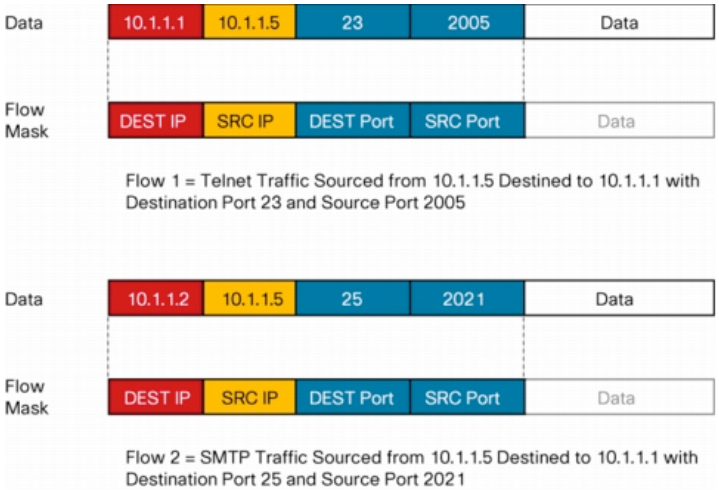
Az SDN alapkövei I.

- A vezérlő és adatsíkok szétválasztása. A vezérlési funkciókat kivesszük a kapcsolóeszközökből, amik ezek után egyszerű csomagtovábbító elemekké válnak mindenféle intelligencia nélkül.



Az SDN alapkövei II.

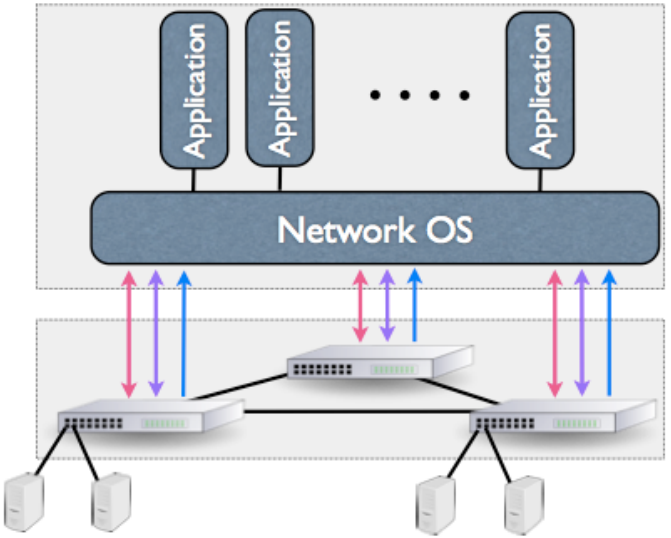
- A kapcsolási döntéseket nem csomag, hanem folyamszinten hozzuk meg.



Az SDN alapkövei III.

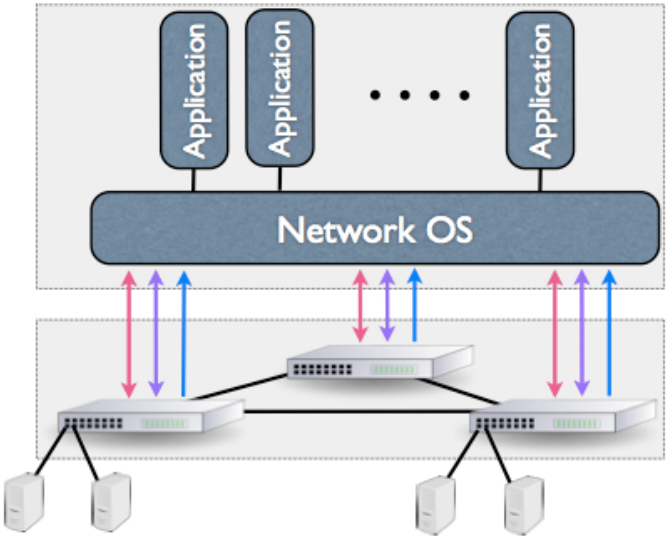
- A vezérlési logikát (ami hagyományos IP hálózatokban a kapcsolókban van) egy külső entitásba, a kontrollerbe más nevén a hálózati operációs rendszerbe (Network Operating System (NOS)) költöztetjük.
- Kontrollhálózat

Igénytelen, egyszerű (pl. broadcast) hálózat
akár alacsony átviteli sebességgel
de magas rendelkezésreállással

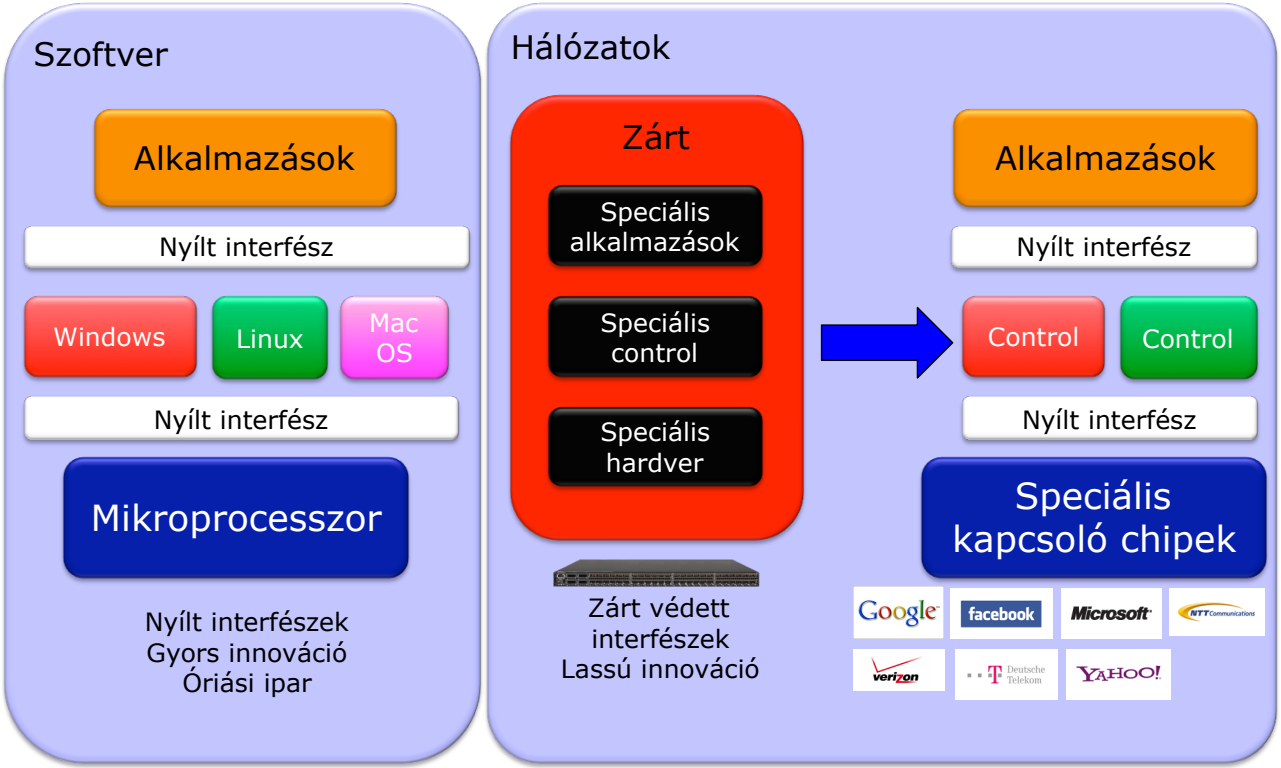


Az SDN alapkövei IV.

- A hálózat programozható a NOS felett futó alkalmazások segítségével. Az alkalmazások kommunikálhatnak a kapcsolóeszközökkel és dinamikusan változtathatják azok viselkedését.
- Alkalmazások pl.: routing, tűzfal

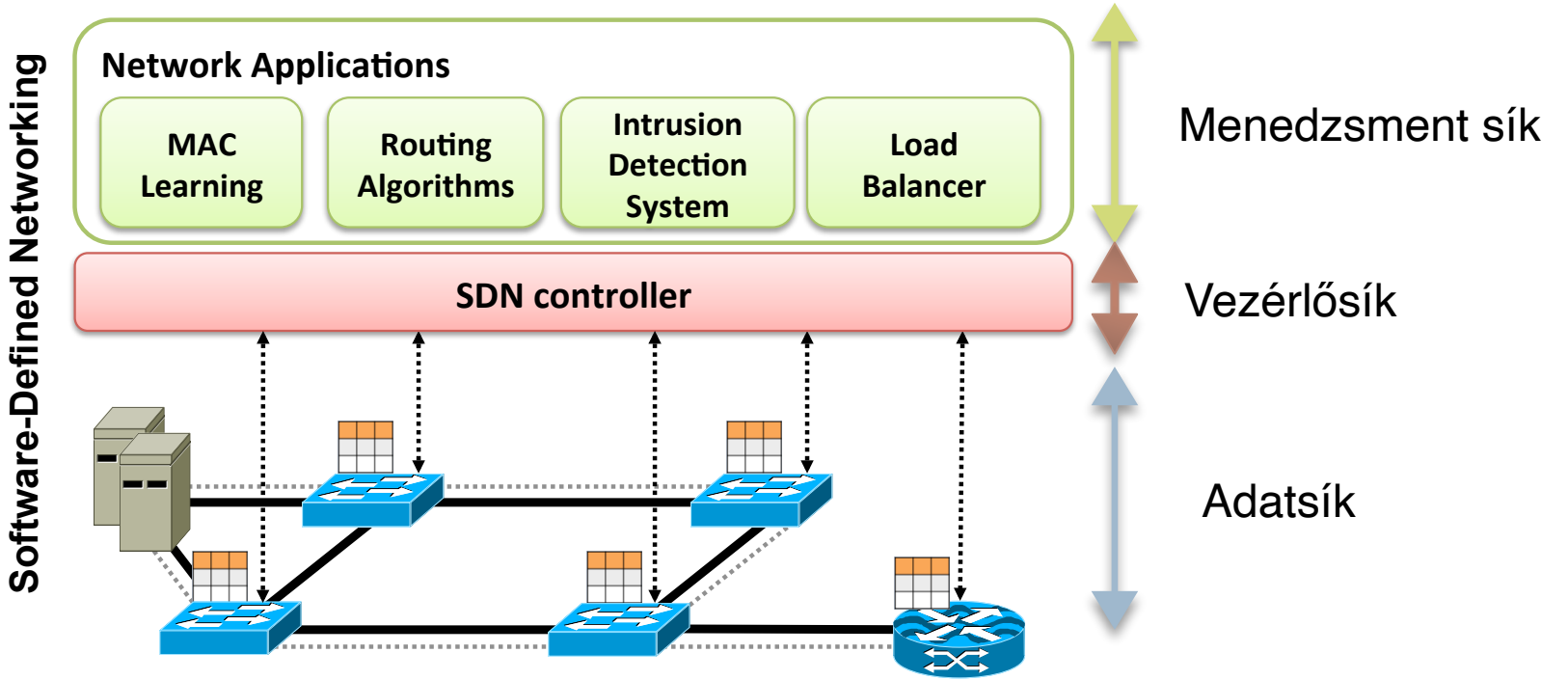


Analógia a szoftver platformokkal

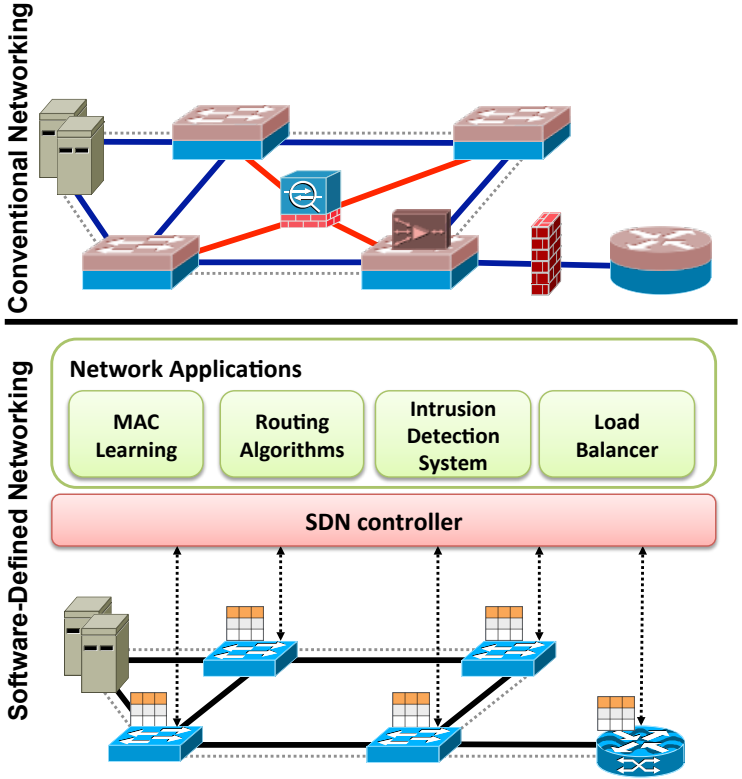


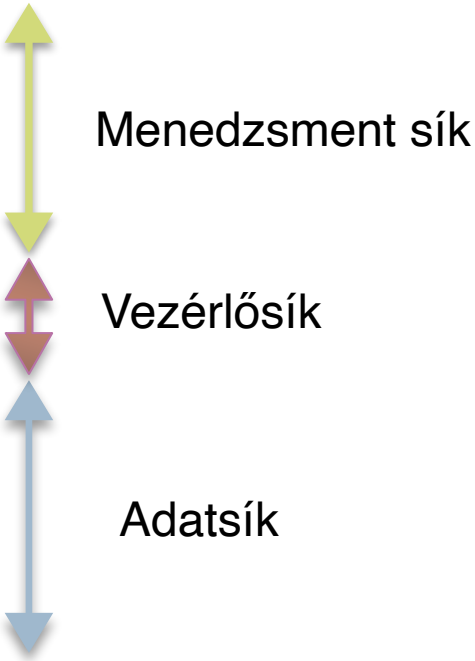
Az SDN síkjai

Az SDN síkjai

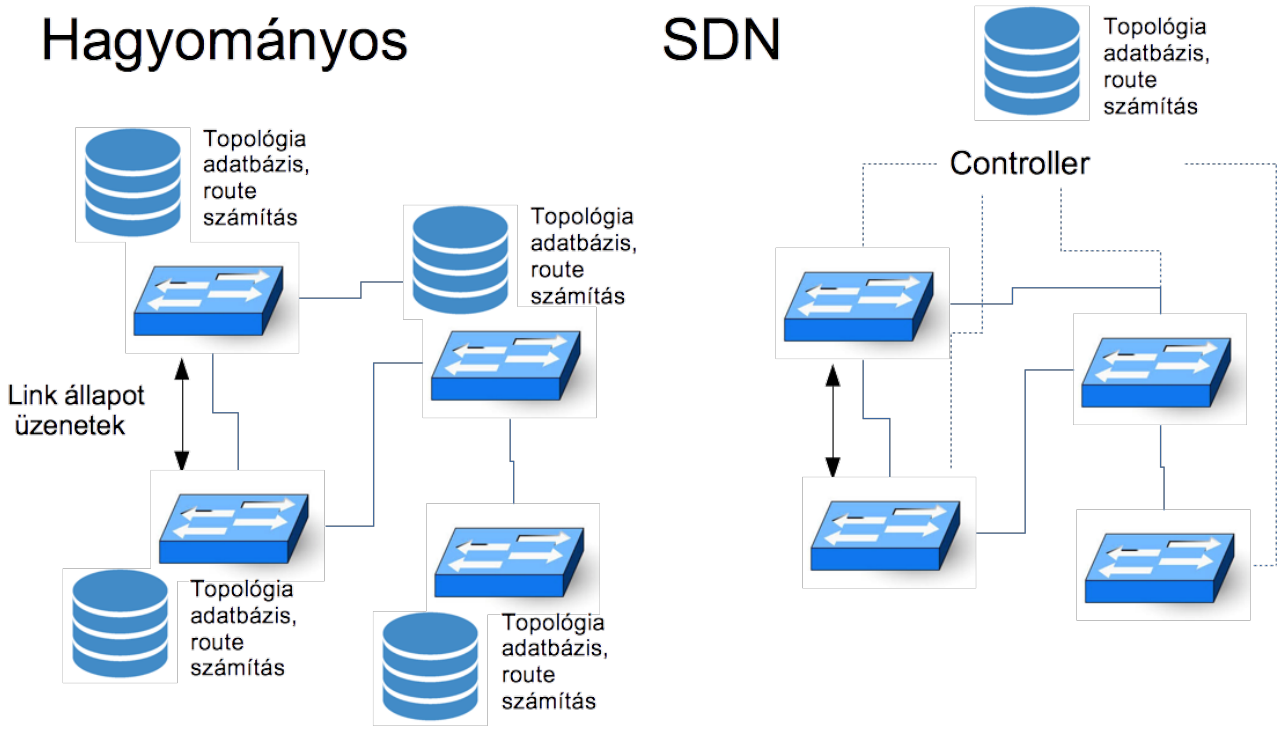


SDN vs hagyományos





Legrövidebb útválasztás OSPF példa



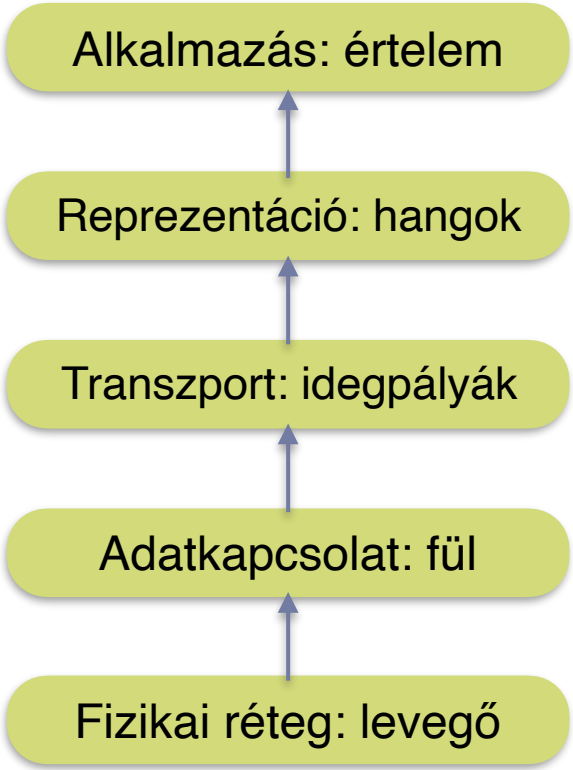
Az adat és vezérlősík szétválasztásából adódó előnyök

- Sokkal könnyebb a hálózatot új funkciókkal bővíteni, hiszen egy SDN alkalmazás egyszerre használhatja, a kontroller által nyújtott információkat és a magas szintű programozási nyelveket.
- Minden alkalmazás használhatja a hálózatról rendelkezésre álló információkat ezért sokkal hatékonyabb szolgáltatások készíthetők és a vezérlősík szoftvermoduljai több modulnál is újrahasznosíthatók.
- Az alkalmazások nagyon könnyen újrakonfigurálhatják a hálózat bármely részében levő kapcsolókat, ezért nincs szükség előre eldönteni, hogy hova helyezzük az egyes funkciókat (pl. terheléselosztó, tűzfal)
- A szolgáltatások integrációja sokkal egyszerűbb. Pl. egyszerűen megadhatjuk, hogy a terheléselosztó alkalmazásnak nagyobb prioritása legyen a routing alkalmazásnál.

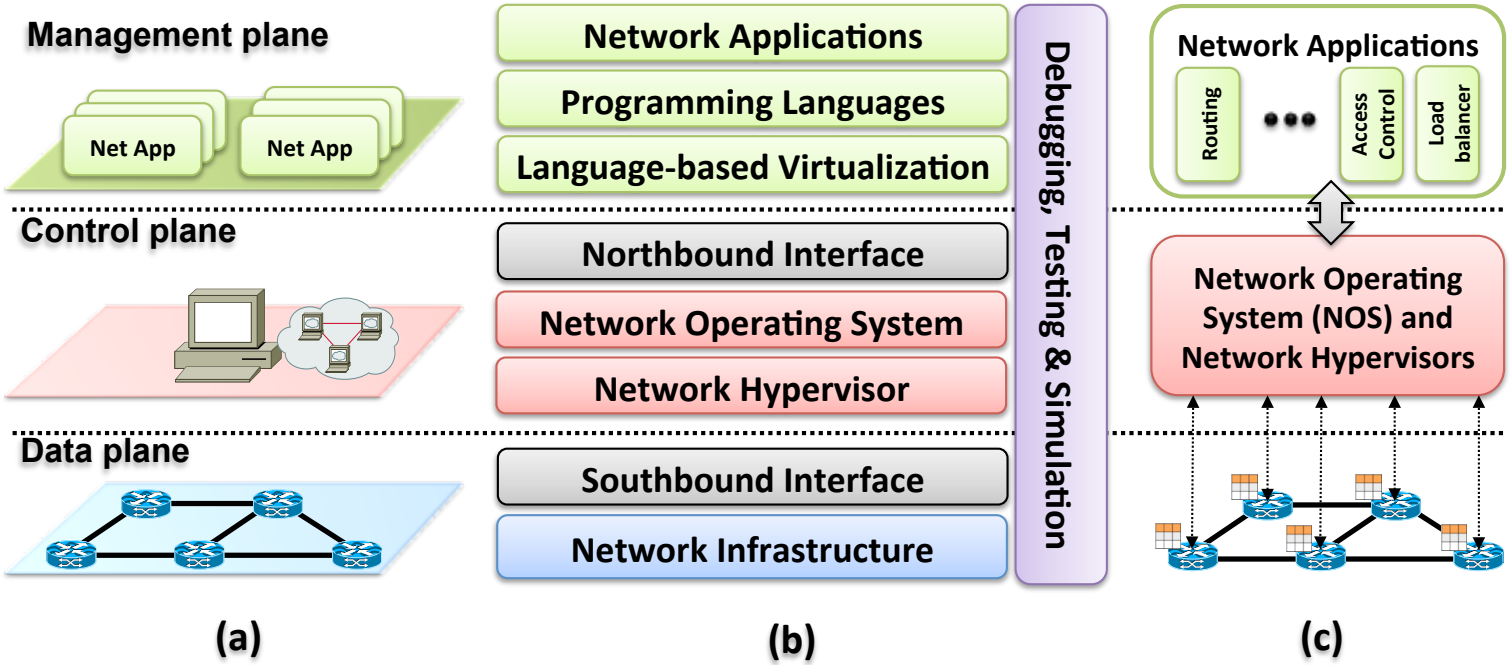
Az SDN rétegei

Rétegek

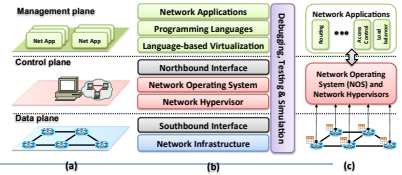
- Miért beszélünk rétegekről? Specifikus-e ez?
- Pl: a hallás rétegei
- Ha az interfészek letisztultak és változatlanok, akkor a fejlődés (innováció) párhuzamosan mehet a rétegeken belül.



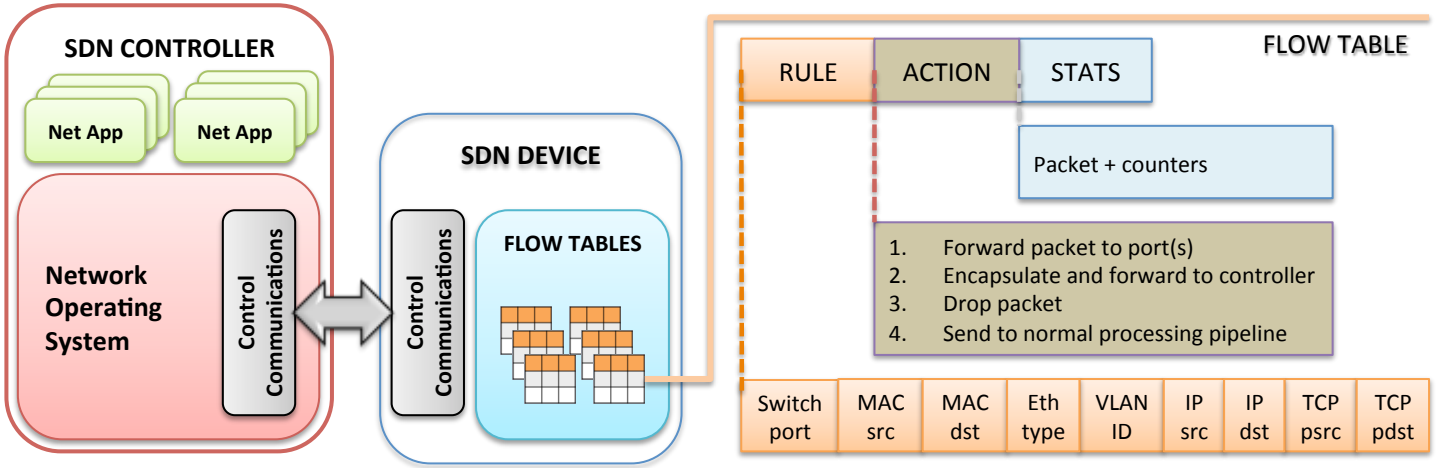
Az SDN rétegei



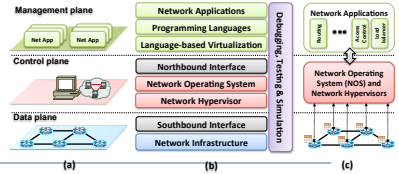
Infrastruktúra



- A hálózat fizikai kapcsolóelemeinek és a köztük levő összeköttetések halmaza
- Mit csinál egy SDN kapcsoló: OpenFlow

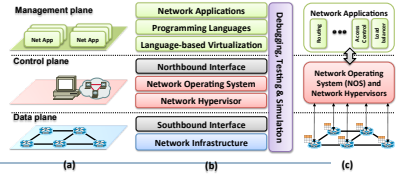


Déli interfész

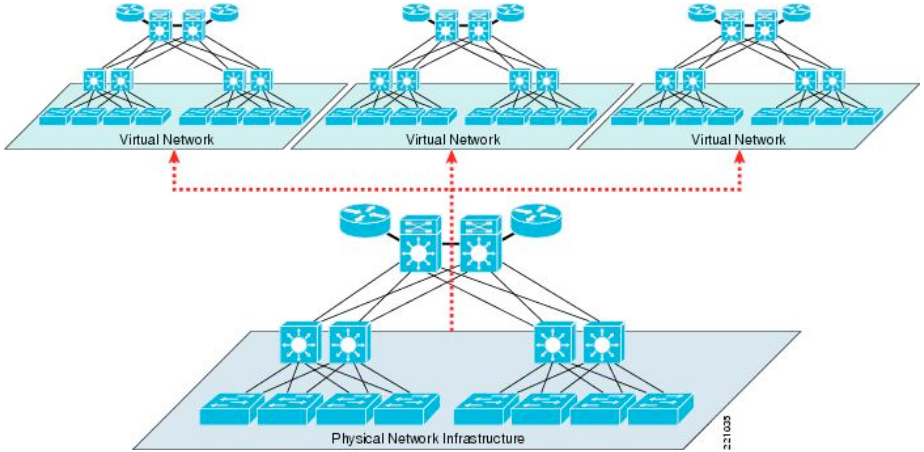


- A kapcsolóelemek és a NOS közötti információátadás módja
- Lényege, hogy implementációtól függetlenül specifikálja a kontroll és adatsík kommunikációját
- OpenFlow esetében az OpenFlow protokoll
  - Esemény alapú működés
    - Link, port események küldése a NOS-nak
    - Flow statisztikák küldése (pl. hány csomag volt része egy adott folyamnak)
    - Packet-in és flow-mod: a kapcsoló jelzi, ha nem tudja mit kezdjen egy csomaggal a NOS erre táblabejegyzést helyezhet el.

Hálózati hypervisor

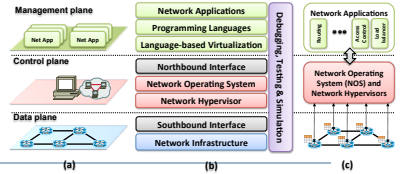


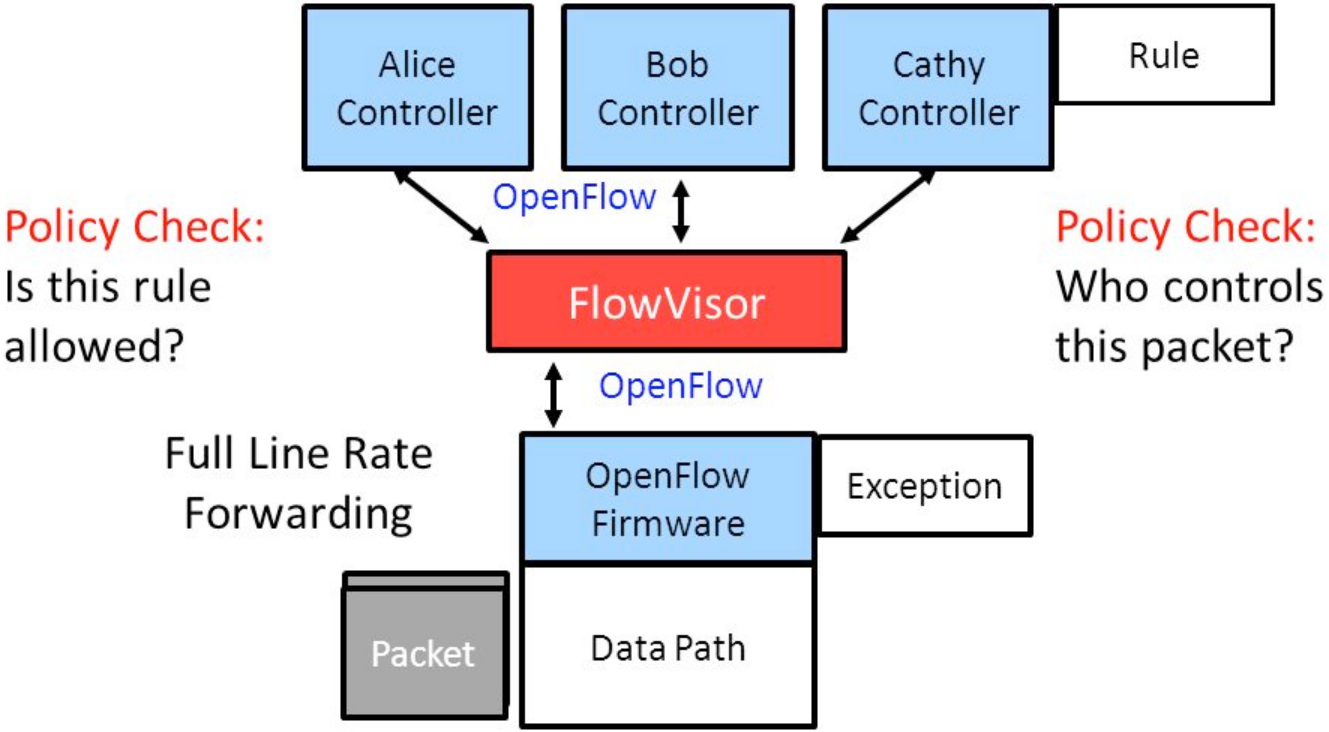
- Virtualizációnak van-e értelme?
- Hálózati virtualizáció:



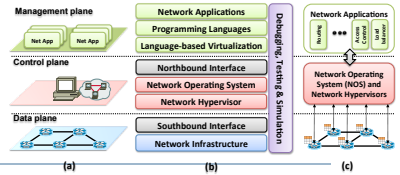
- Lehet pl. eszköz szintű: bizonyos kapcsolókat és linkeket külön NOS-hoz rendeljük hozzá.
- Lehet pl. folyamtér (flow-space) szintű, amikor adott folyamokat rendelük külön NOS-hoz, vagy alkalmazáshoz

FlowVisor


Management plane
Net App
Net App
Control plane
Data plane
Network Applications
Programming Languages
Language-based Virtualization
Northbound Interface
Network Operating System
Network Hypervisor
Southbound Interface
Network Infrastructure
Debugging, Testing & Simulation
(a)
(b)
Network Applications
Network Operating System (NOS) and Network Hypervisors
(c)


Alice Controller
Bob Controller
Cathy Controller
Rule
OpenFlow
Policy Check:
Is this rule allowed?
FlowVisor
Policy Check:
Who controls this packet?
OpenFlow
Full Line Rate Forwarding
OpenFlow Firmware
Exception
Packet
Data Path

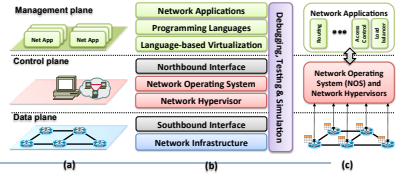
Hálózati operációs rendszer NOS



- Az agy: ide fut be minden információ az infrastruktúrából
- Magas szintű API az infrastruktúrával (kapcsolóelemekkel) való kommunikációra (hasonlóan az OS-ekhez), izoláció, biztonság, konkurens hozzáférés
- Alapvető szolgáltatások, topológia felderítés, monitorozás
- Kapcsolóelemek konfigurációja, menedzsmentje (korábban kézzel, zárt struktúrában), akár többféle déli interfész támogatásával
- Erre épülnek a hálózati alkalmazások

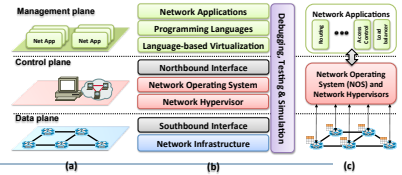
Hálózati operációs rendszerek OpenFlow-hoz

Északi interfész



- Magát a hálózat absztrakcióját valósítja meg
- A déli interfésszel szemben ez egy szoftver rendszer
- Az alkalmazások és a NOS közötti kommunikációt adja meg
- Még nincs gyakorlatban elfogadott verzió
- Legtöbb NOS-nak az API-ja jelenti az északi interface
- De vannak már próbálkozások (frenetic REST API)
- Hasonló kéne legyen mint az OS-ek esetében a POSIX

Programozási nyelvek



- Assembly—OpenFlow üzenetek
  - Tisztán az OpenFlow üzenetekkel is lehet hálózatot vezérelni,
  - nagyon bonyolult, hiszen csomó hardver közeli apróságra kell figyelni, mint pl. konkurens folyamtábla-szerkesztés, átfedő folyamtábla bejegyzések, dinamikus működésből adódó inkonzisztens állapotok stb.
  - Ráadásul ilyen alacsony szintű nyelven nehéz jól strukturált újrafelhasználható kódot írni.
- Magasabb szintű nyelvek: c/c++, java, python, ruby

Magas szintű programozási nyelvek nyelvek

- A fő céljuk ezeknek a nyelveknek az, hogy a fejlesztőknek ne kelljen annyit a hálózat megvalósításával foglalkozniuk, ehelyett a konkrét megoldandó feladatra (logikai működés) tudjanak fókuszálni
- Előnyeik:
  - A hálózat magas szintű absztrakciója, melyen keresztül a hálózat funkcionalitása (pl. legrövidebb útválasztás) könnyen megadható. A fordító mechanizmusok feladata ezután, hogy a magas szinten megadott leírást végül konkrét OpenFlow üzenetekké transzformálják.
  - Produktív, problémaközpontú környezet kialakítása, a fejlesztés és innováció gyorsítása, hibalehetőségek minimalizálása.
  - Moduláris, újrahasznosítható kód létrehozása.
  - Nyelv alapú hálózati virtualizáció. Ez kb. azt jelenti, hogy a hálózat elemeihez virtuális objektumokat rendelhetünk (egy kapcsolóhoz akár többet is), így magán a hálózati alkalmazáson belül hálózatvirtualizációt valósíthatunk meg.

Magas vs. alacsony szint példa

Simple Repeater

```
def switch_join(switch):
        # Repeat Port 1 to Port 2
        p1 = {in_port:1}
        a1 = [forward(2)]
        install(switch, p1, DEFAULT, a1)

        # Repeat Port 2 to Port 1
        p2 = {in_port:2}
        a2 = [forward(1)]
        install(switch, p2, DEFAULT, a2)
```

Web Traffic Monitor

```
def switch_join(switch)):
        # Web traffic from Internet
        p = {inport:2,tp_src:80}
        install(switch, p, DEFAULT, [])
        query_stats(switch, p)

def query_stats(switch, p, bytes, ...)
        print bytes
        sleep(30)
        query_stats(switch, p)
```

Composition: Repeater + Monitor

```
def switch_join(switch):
        pat1 = {inport:1}
        pat2 = {inport:2}
        pat2web = {in_port:2, tp_src:80}
        install(switch, pat1, DEFAULT, None, [forward(2)])
        install(switch, pat2web, HIGH, None, [forward(1)])
        install(switch, pat2, DEFAULT, None, [forward(1)])
        query_stats(switch, pat2web)

def query_stats(switch, xid, pattern, packets, bytes):
        print bytes
        sleep(30)
        query_stats(switch, pattern)
```

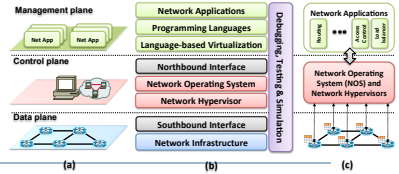
```
# Static repeating between ports 1 and 2
def repeater():
        rules=[Rule(inport:1, [forward(2)]),
        Rule(inport:2, [forward(1)])]
        register(rules)

# Monitoring Web traffic
def web_monitor():
        q = (Select(bytes) *
        Where(inport:2 & tp_src:80) *
        Every(30))
        q >> Print()

# Composition of two separate modules

def main():
        repeater()
        web_monitor(
```

Alkalmazások



- Az SDN architektúra (SDN stack) tetején ülnek az alkalmazások. Ezek
- adják meg a hálózat funkcióját, működési logikáját.
- Alap (általában a NOS-al együtt jár) topology, discovery, monitoring
- Load balancing, firewall, routing, multipath, hibatűrés, energiafogyasztás minimalizálás, QoS, mobilitás kezelés, hálózatoptimalizálás stb.
- Nézzük meg hogyan töltünk alkalmazásokat egy OpenFlow hálózatra

NFV (Network Function Virtualization)

- Az NFV nem más, mint a hardveres middlebox-ok virtualizált verziója.
- NFV-be általában olyan funkcionalitást szoktak tenni, ami a kapcsolókban nem valósítható meg (pl. részletes csomagvizsgálat, forgalomanalízis, hálózati kódolás, behatolás detektálás stb.).
- A middlebox-ok virtualizációja kényelmessé teszi használatukat, könnyen menedzselhetővé varázsolja őket, a virtualizált verziókat könnyű áthelyezni, frissíteni vagy éppen kivonni a forgalomból.