Hálózatok építése és üzemeltetése

SDN: Software Defined Networking

Mai téma

- Hálózatok “szoftverizálása” - network softwarization
  - control plane szoftverizálása
    - SDN: Software Defined Networking
  - data plane szoftverizálása
    - NFV: Network Function Virtualization
  - hálózati szolgáltatások/alkalmazások szoftverizálása
    - SFC: Service Function Chaining
- Egy konkrét példa: OpenFlow

Kiktől loptam slide-okat?

- Rob Sherwood
  - “GENI Engineering Workshop June 2010”
- Guido Appenzeller
- Nick McKeown
- Guru Parulkar
- Brandon Heller
- Marco Cello
- Seyed Kaveh Fayazbakhsh
- és még sokaktól…

- (Még ez a slide is lopott…)
- (Persze, ők is lopták egymástól…)

SDN koncepció

Marketing

- **Egy-két hír a ~~közelmúltból~~:**
- “**Google** is using **OpenFlow** on custom-designed hardware for all the internal networks it runs connecting its global **data centers**, said **Urs Holzle**, senior vice president of technology infrastructure at Google”

- “**How Google is using OpenFlow to lower its network costs?** Google is checking out a new form of networking protocol known as OpenFlow, in the communications networks that run between its data centers. The search giant is testing the use of software defined networks in order to lower the cost of delivering a bit of information.” (gigaom.com)

- “Virtualization and cloud infrastructure provider **VMware** (NYSE: VMW), announced this week that it will pay **$1.05 billion** in cash plus approximately $210 million in assumed unvested equity awards to **acquire Nicira**, a software-defined networking (**SDN**) **specialist** and provider of network virtualization for open source initiatives.“ (RCR Wireless News – Americas)

Probléma

- Számítógép-hálózat
  - bonyolult, elosztott rendszer
  - különböző HW eszközökből áll
    - switch, router, middlebox, …
    - zárt, gyártóspecifikus HW, FW, SW
    - bonyolult, elosztott kontroll funkciók (pl. routing protokollok)
    - heterogén eszközök konfigurálása
    - különböző interfészek

Probléma

- Számítógép-hálózat
  - nehéz/költséges tervezés és üzemeltetés
  - konfigurálás ↔ programozás
  - lassú innováció (akadémia problémája)
  - drága (ipar problémája)
    - üzemeltetés
    - fejlesztés
    - új szolgáltatások bevezetése/beüzemelése

Cél

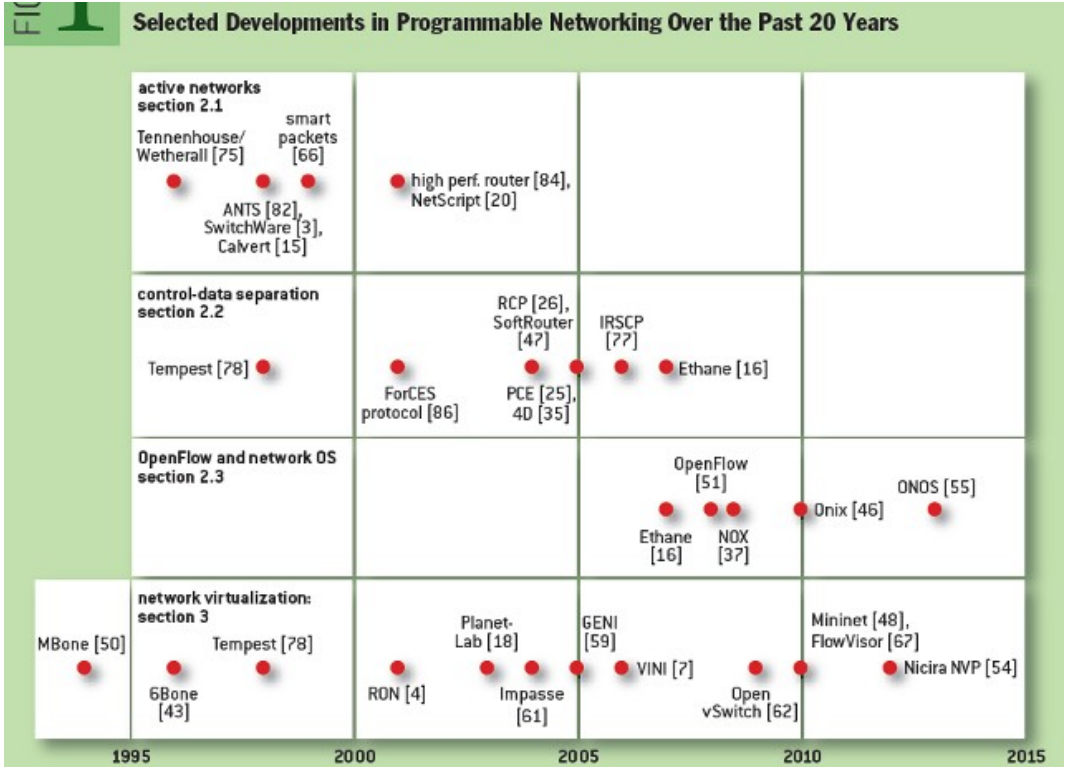
- Számítógép-hálózatok minél “jobb” programozhatósága
- programozhatóság
  - hálózat mint egész működésének meghatározása
  - több mint az egyes elemek működésének befolyásolása
  - konfigurálás ↔ programozás
  - időskála!
- “jobb”
  - könnyebb, gyorsabb
  - flexibilisebb
  - szélesebb körű
  - kevesebb hiba(lehetőség)
  - gyorsabb javíthatóság

(Egy) Megoldás(i irány)

- Software Defined Networking
  - kontrollsík szoftverizálása
  - kontrollsík ↔ adatsík szeparálása
    - kontroll: döntés hogy mi történjen az adott forgalommal
    - adatsík: csomagok továbbítása
  - kontrollsík centralizálása / konszolidálása / egységesítése
  - közöttük: nyílt interfész(ek)
  - korábban: elosztott rendszer, „sok-sok” kapcsolat
  - most: elosztott rendszer, „egy-sok” kapcsolat!
- Nem új ötlet!
  - több évtizedes út
  - sok korábbi ötlet felhasználása
  - pl. PSTN: kontroll- és adatsík szeparálás

Hogyan jutottunk az SDN-ig?

Hogyan jutottunk az SDN-ig?



N. Feamster, J. Rexford, E. Zegura, **The Road to SDN**, *ACM Queue*,, Volume 11, Issue 12, Dec. 2013

Hogyan jutottunk az SDN-ig?

- aktív hálózatok
  - clean-slate architektúra
  - vízió, elrugaszkodott az akkori hálózati valóságtól
  - új adatsík funkció kódja leküldhető az eszközökbe (~ Java kód)
    - kapszula modell (in-band, adatcsomagokban)
    - programozható router/switch model (out-of-band)
  - egyéges végrehajtási környezet az adatsík csomópontjaiban (EE, execution environment)
  - middleboxok egységes kezelése
  - adatsík programozása
  - sok ötlet mára újra előjött!

Hogyan jutottunk az SDN-ig?

- kontroll-adat szeparálás
  - ForCES (Forwarding and Control Element Separation)
    - IETF szabvány
    - nyílt interfész az adatsík felé
    - innováció a kontrollsík szoftvereiben
  - SoftRouter
    - ForCES API-t használja
    - kontroll program forwarding tábla bejegyzéseket tud elhelyezni az adatsík eszközeiben
    - ForCES-t nem fogadták el a nagy router gyártók!
  - RCP (Routing Control Platform)
    - BGP (Border Gateway Protocol) használata
    - flow bejegyzések elhelyezésére
    - meglévő routerekkel működik

Hogyan jutottunk az SDN-ig?

- kontroll-adat szeparálás
  - kontrollsík programozása
  - eszközszintű konfiguráció helyett → hálózat vezérlése

  - Ethane (elődje: SANE)
    - Stanford University, Clean Slate Project
    - centralizált logika
    - magas szintű hálózati policy-k leképzése
    - eszközszintű flow bejegyzésekre
    - OpenFlow közvetlen elődje!

Hogyan jutottunk az SDN-ig?

- MPLS
  - Cél
    - hálózati HW-ek egyszerűsítése
    - hálózati kontroll flexibilitásának javítása
    - (SDN ezt viszi tovább)
  - edge – core szeparáció
    - edge: komplex logika
    - core: hatékony csomagtovábbítás
    - (OpenFlow esetében újra előjött)

Hogyan jutottunk az SDN-ig?

- OpenFlow
  - kontroll-adat szeparálás +
  - hálózati eszköz általánosítása (absztrakció)
  - műveletek általánosítása (bizonyos mértékben)
  - új koncepció: hálózati operációs rendszer

  - SDN (koncepciók) egy népszerű realizációja
  - DE más realizációk is vannak
    - pl. láttuk, BGP-alapú megoldás
    - gyártó specifikus megoldások pl.: Cisco ONE platform, Juniper JunOS SDK

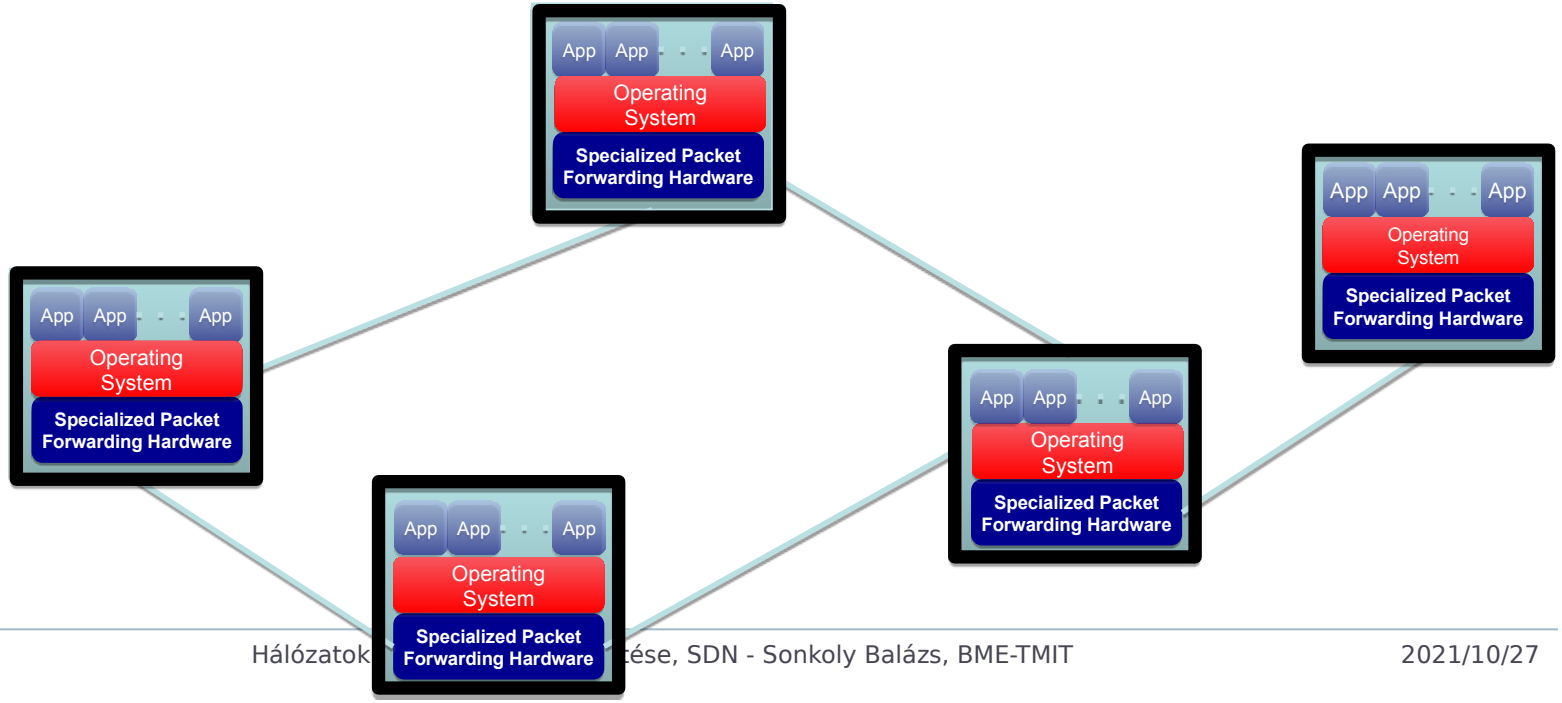
Hogyan jutottunk az SDN-ig?

- OpenFlow
  - Siker oka: sokan álltak mögé
  - Akadémiai szereplők
    - legnagyobb egyetemek (USA, EU)
  - Ipari szereplők
    - gyártók
      - NEC, HP, Cisco, Pronto, Brocade, Broadcom, Ericsson, IBM, …
    - felhő szolgáltatók
      - Amazon, Google, Microsoft, …
    - szolgáltatók / adatközpont üzemeltetők
      - Facebook, …
    - carriers
      - DT, Telecom Italia, Telefonica, NTT, …
  - ma már: szabványosító szervezetek
    - Open Networking Foundation
    - OpenDaylight initiative

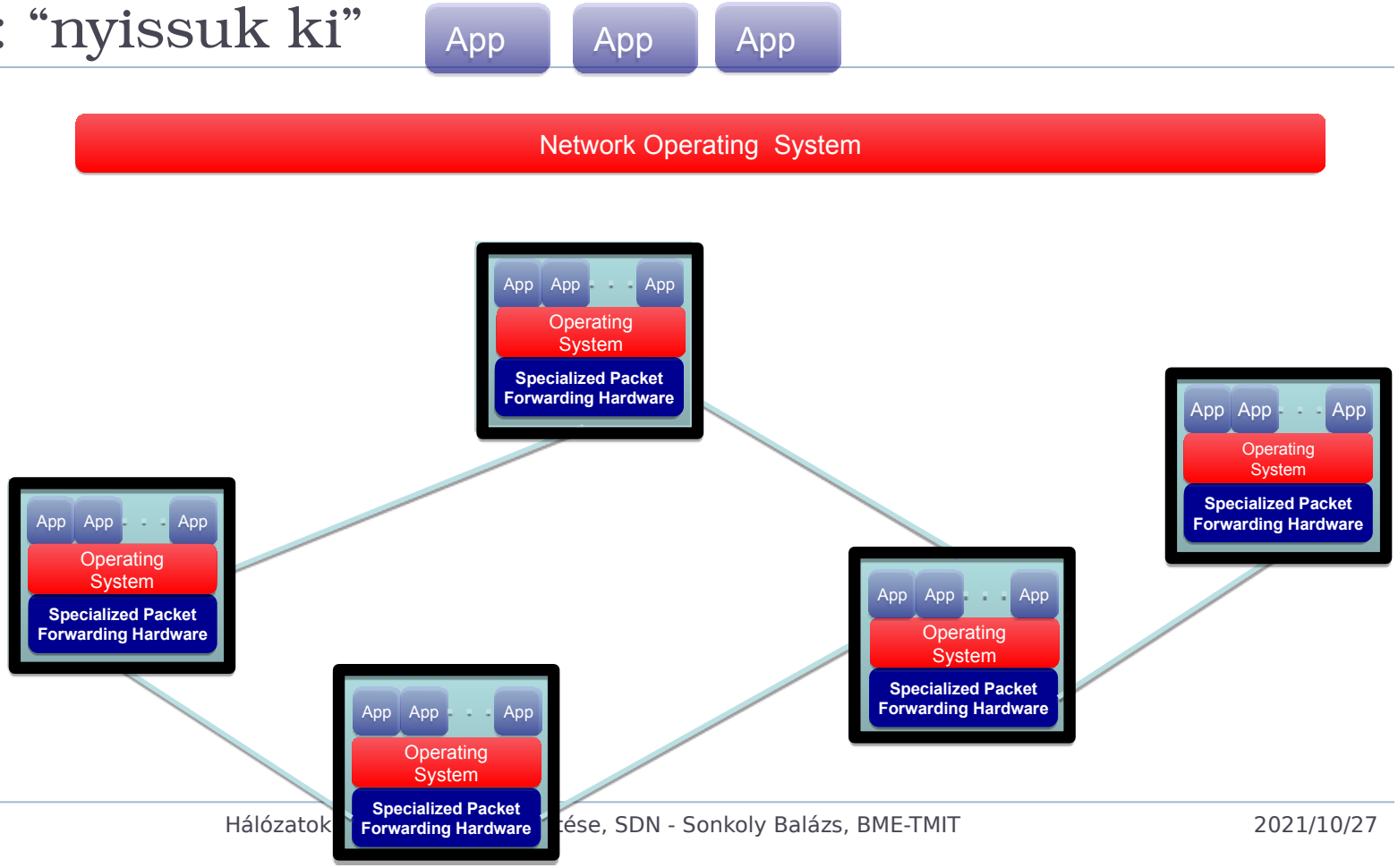
OpenFlow

Alapok

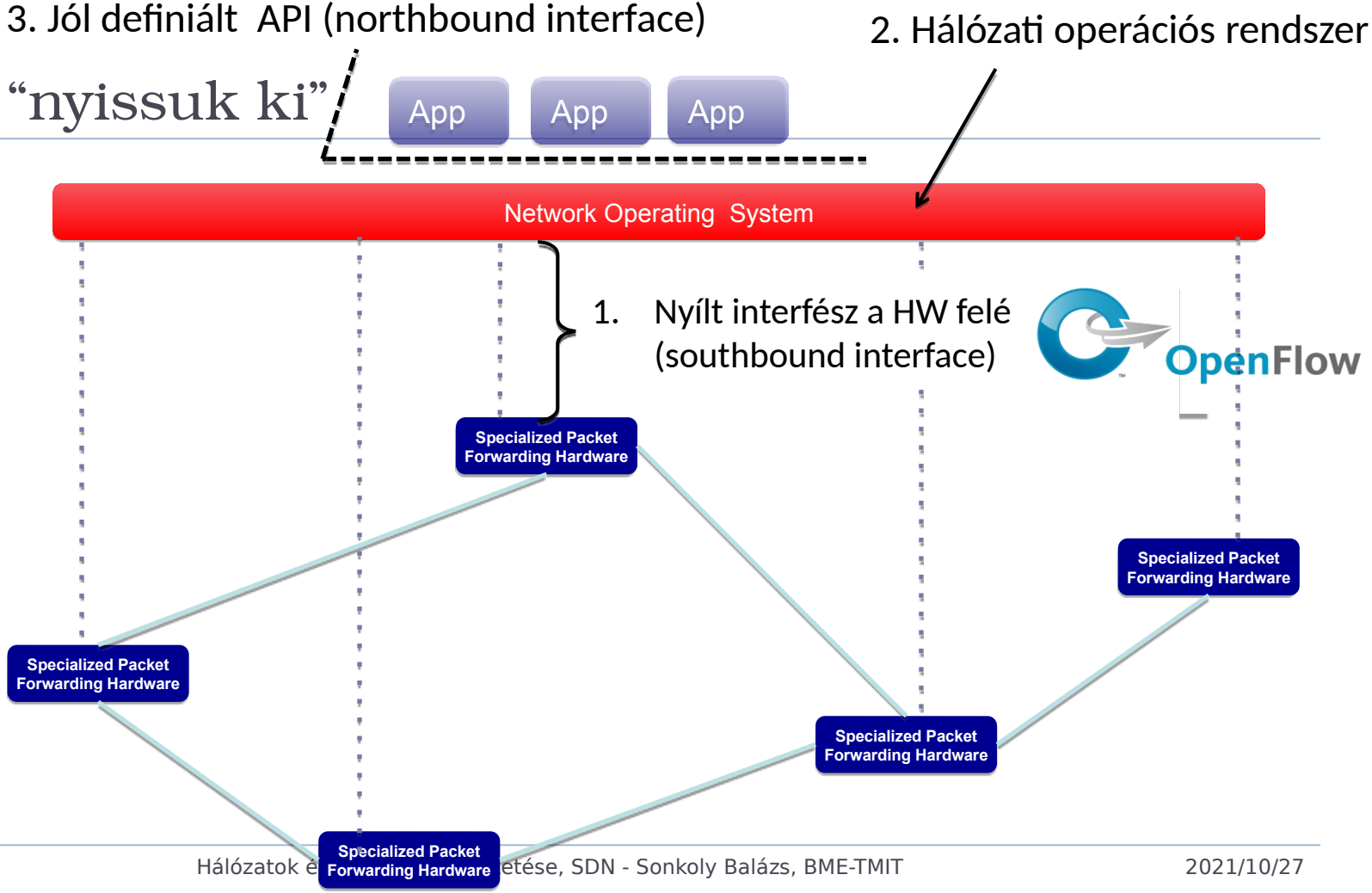
Internet ma: zárt infrastruktúra



SDN: “nyissuk ki”



SDN: “nyissuk ki”



Mi is az az OpenFlow?

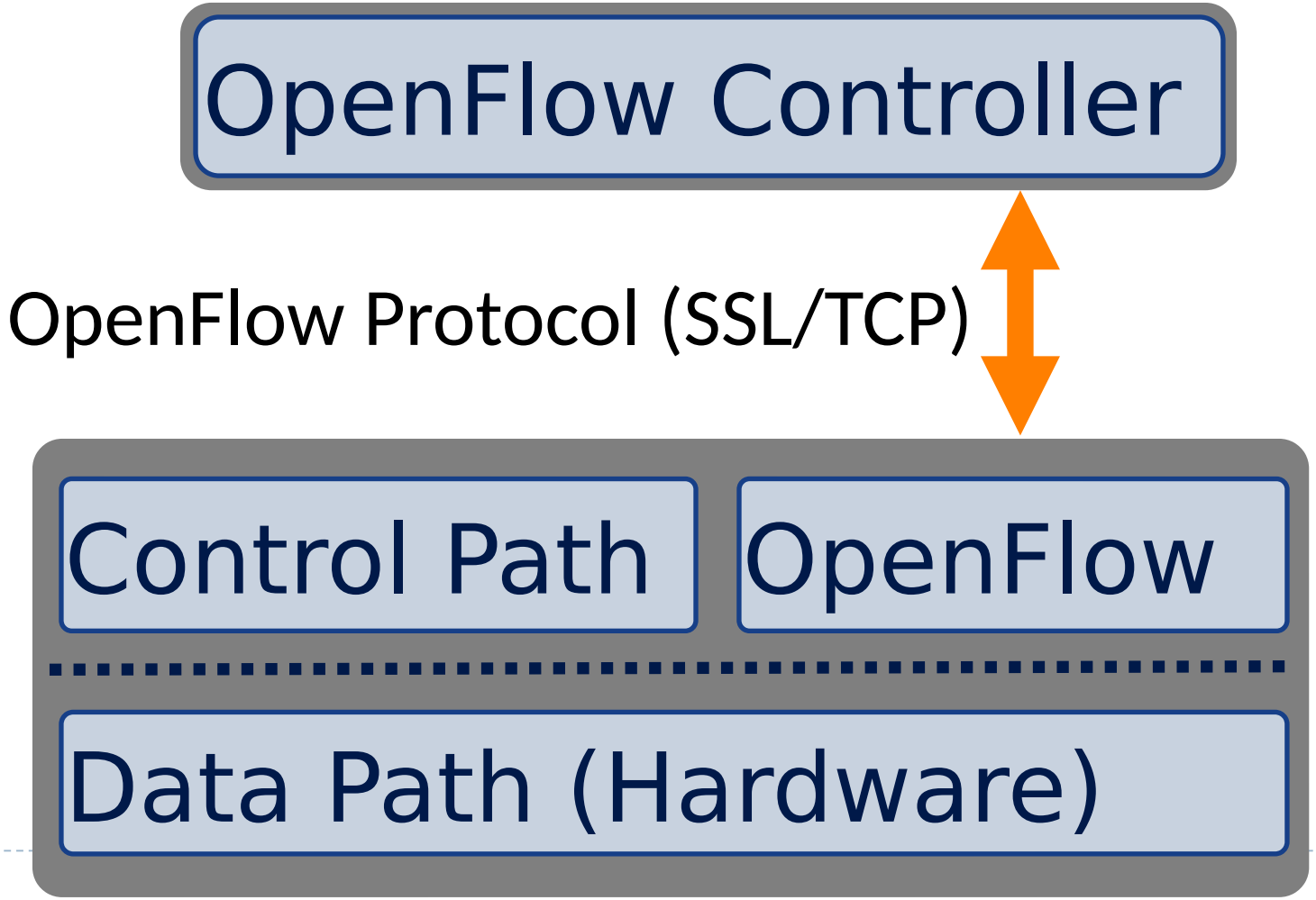
- OpenFlow egy API, interfész

- Ezen keresztül kontrollálható a csomag-továbbítás (forwarding)
- olcsó HW-en is implementálható
- Üzemeltetett hálózat programozható lesz
  - nem csak konfigurálható!
- Egyszerűbb innováció
- (egyszerűbb üzemeltetés, új szolgáltatások bevezetése)

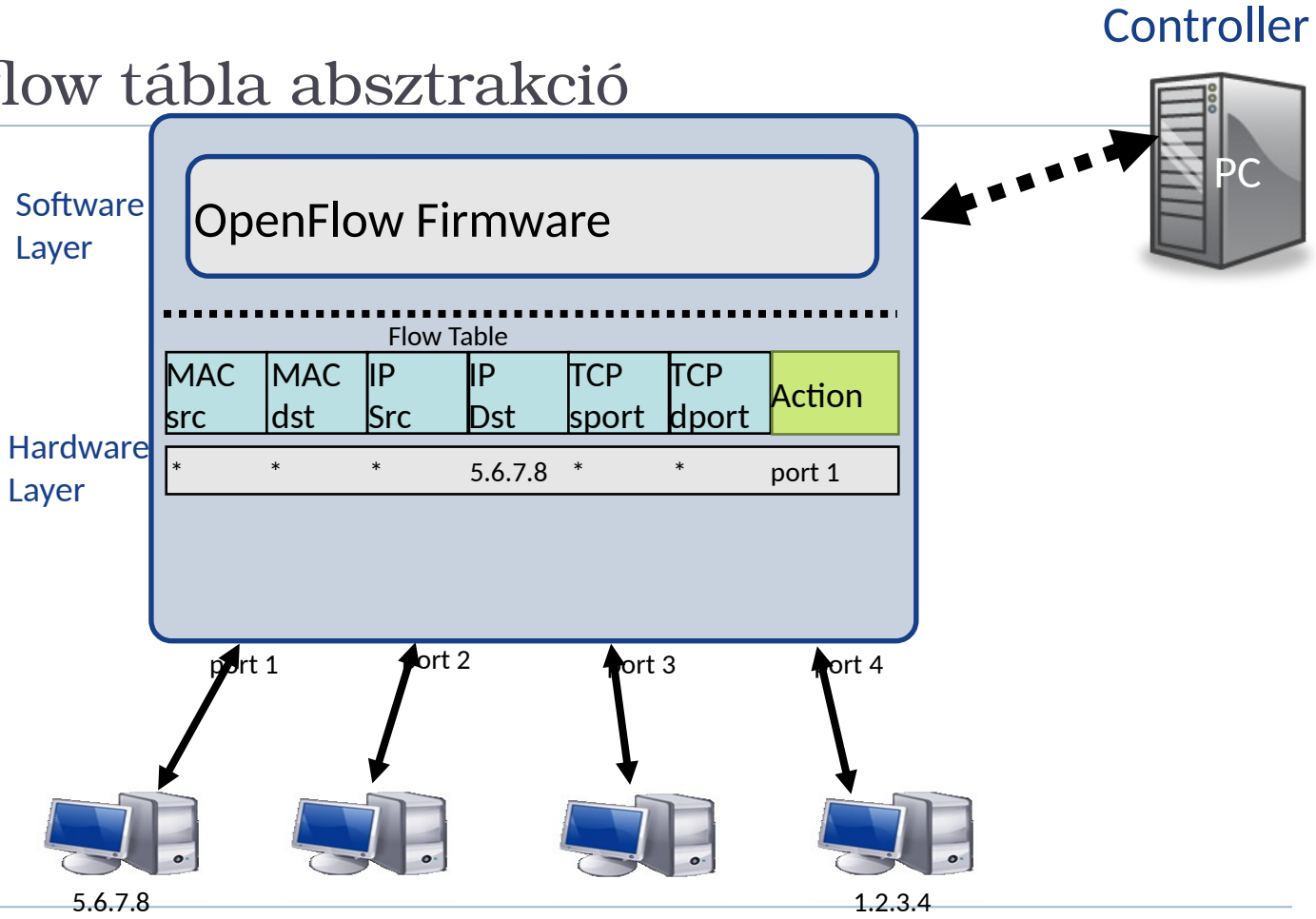
- Fő célok
  - Ne kelljenek speciális testbedek
  - Kísérleti megoldások **valós hálózaton**, **valós forgalom** mellett, **vonali sebesség**en

Ethernet Switch

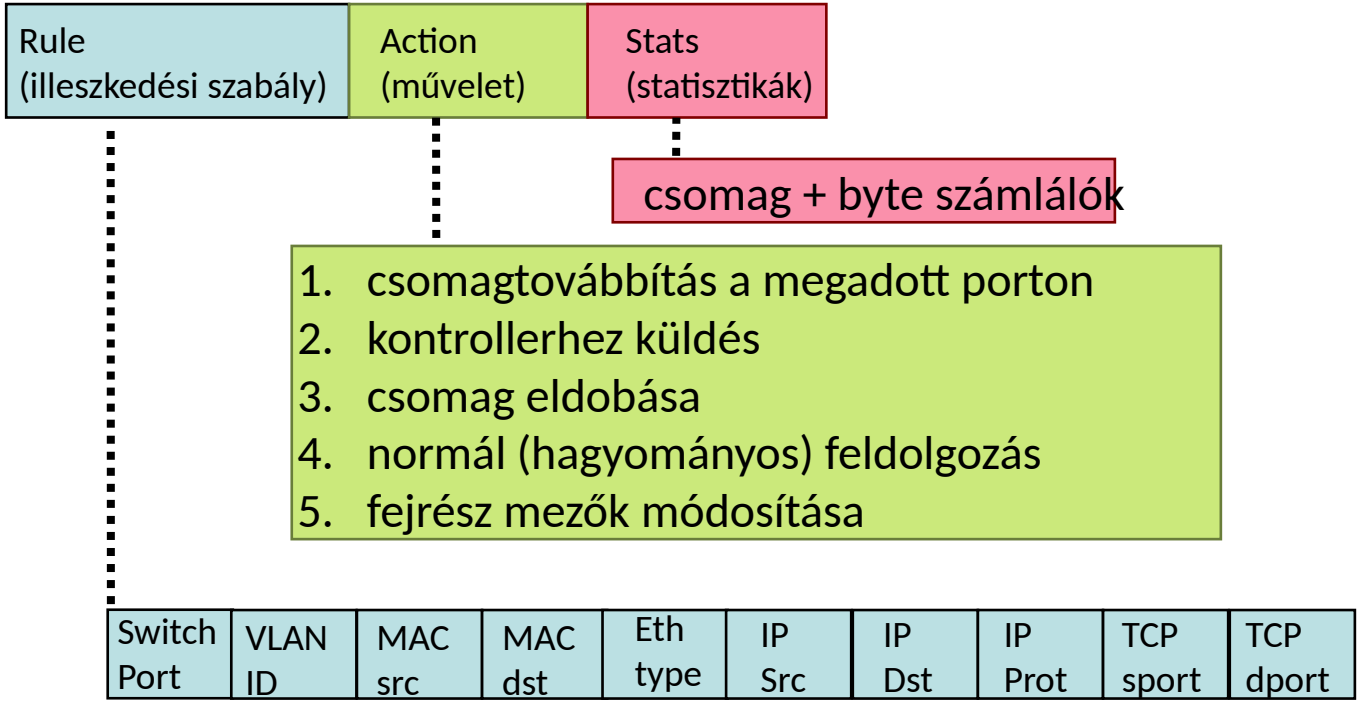

Control Path (Software)
Data Path (Hardware)


OpenFlow Controller
OpenFlow Protocol (SSL/TCP)
Control Path
OpenFlow
Data Path (Hardware)

OpenFlow flow tábla absztrakció



Flow tábla bejegyzések



+ a nem szükséges mezők maszkolhatók (wildcard)

Flow tábla bejegyzések: példák

Switching (L2 kapcsolás)

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f:..	*	*	*	*	*	*	*	port6

Routing (L3 útvonalválasztás)

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	5.6.7.8	*	*	*	port6

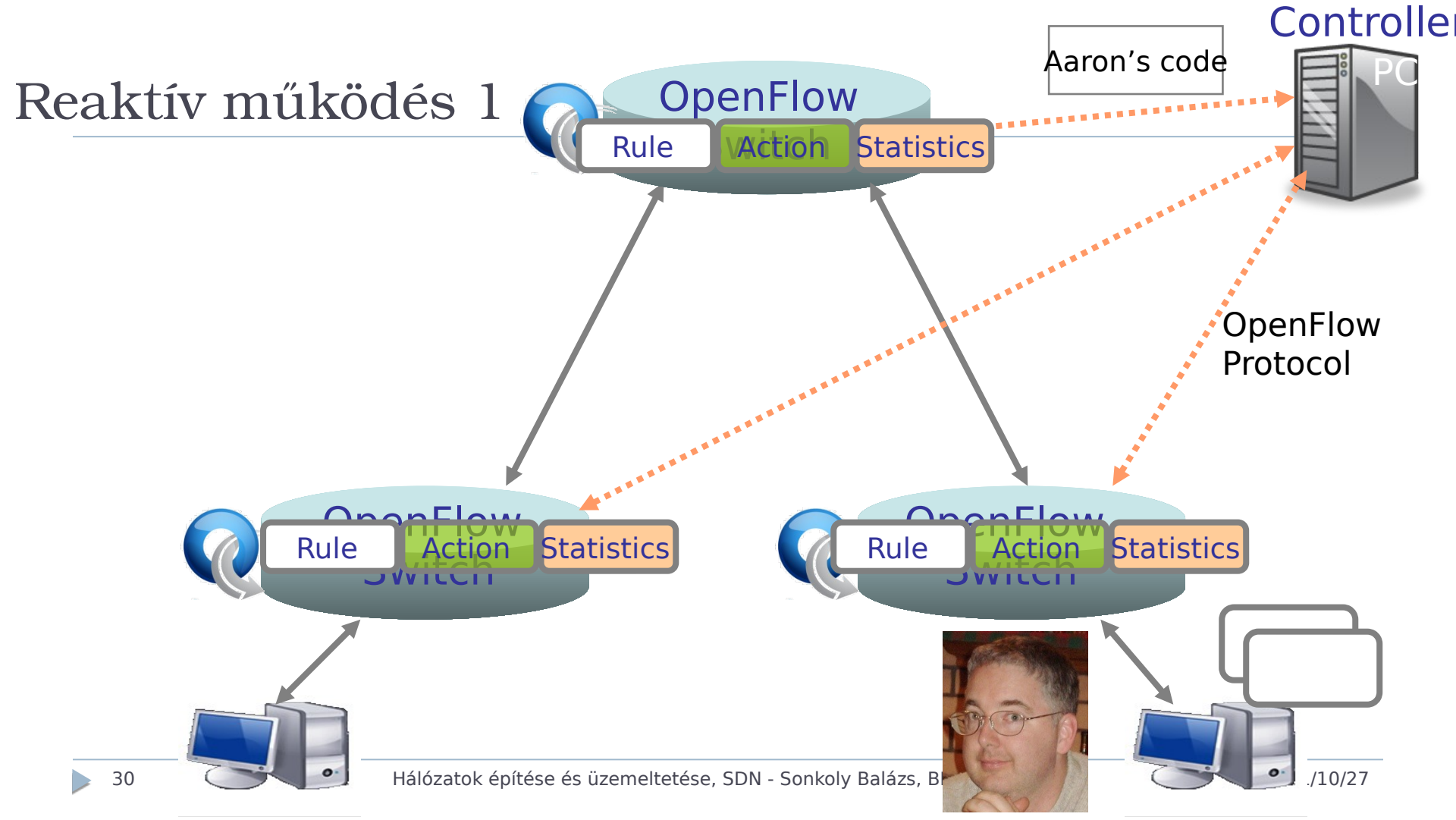
VLAN Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f..	*	vlan1	*	*	*	*	*	port6, port7, port9

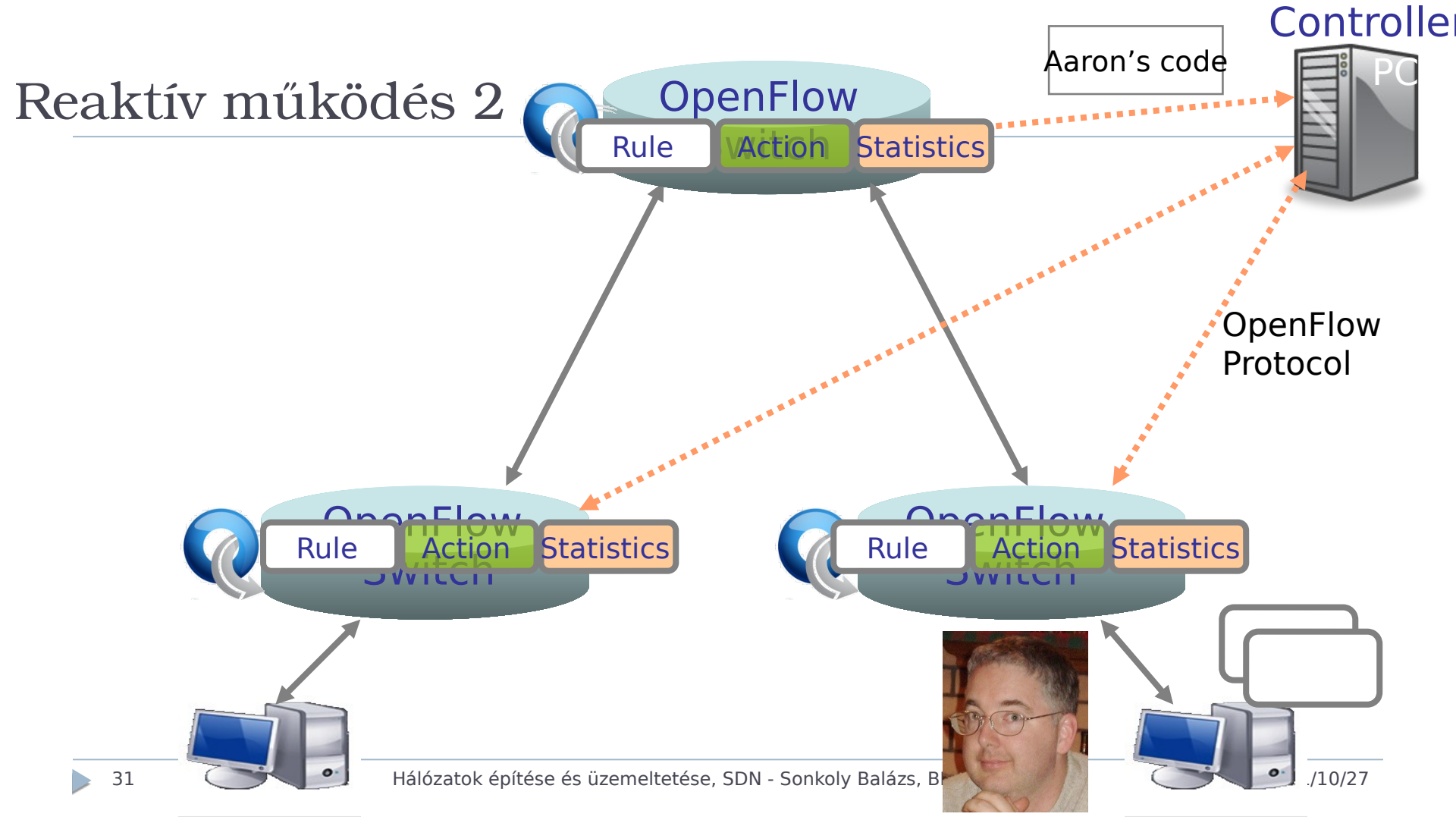
OpenFlow

Működés

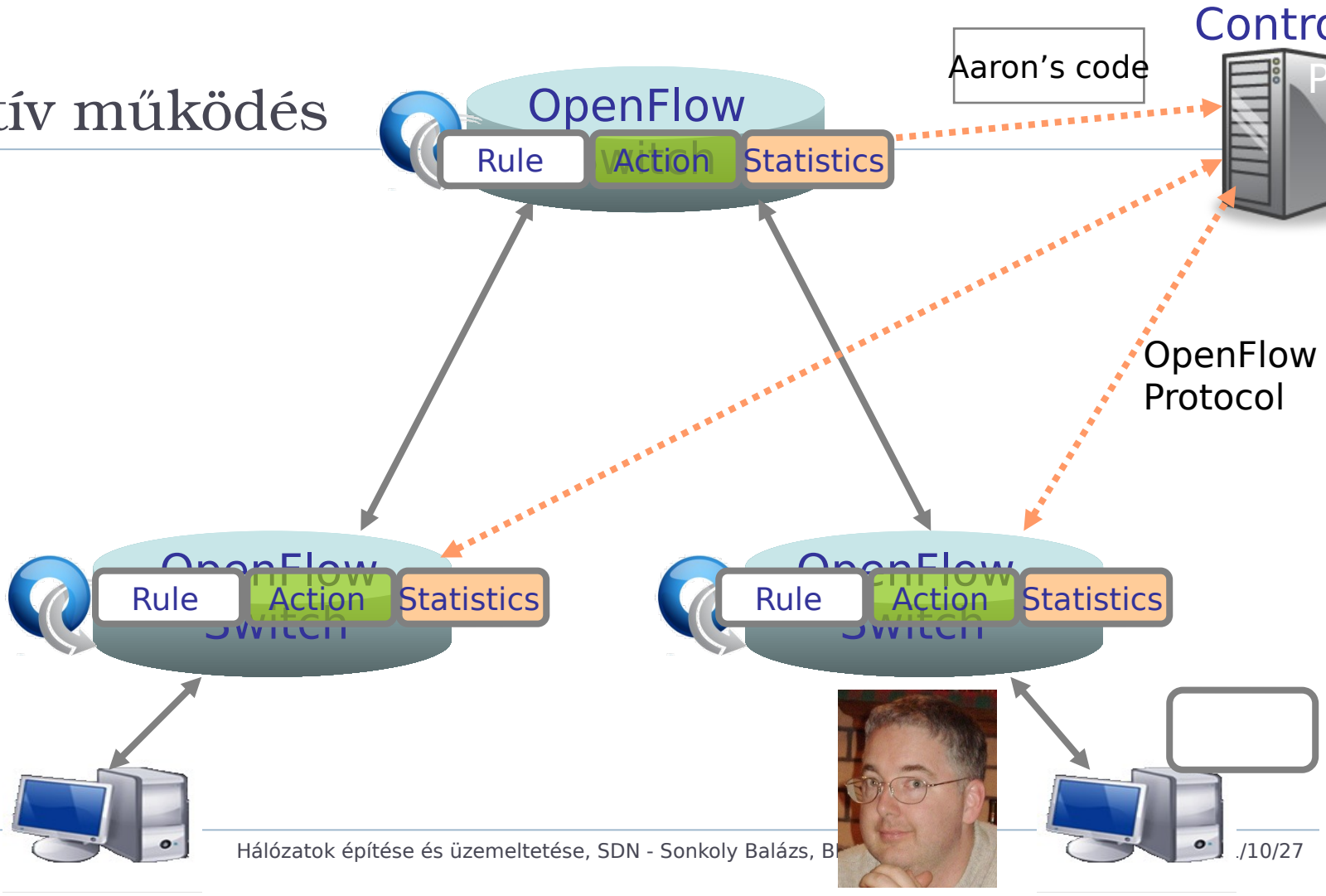
Reaktív működés 1



Reaktív működés 2



Proaktív működés



Új problémák & kihívások

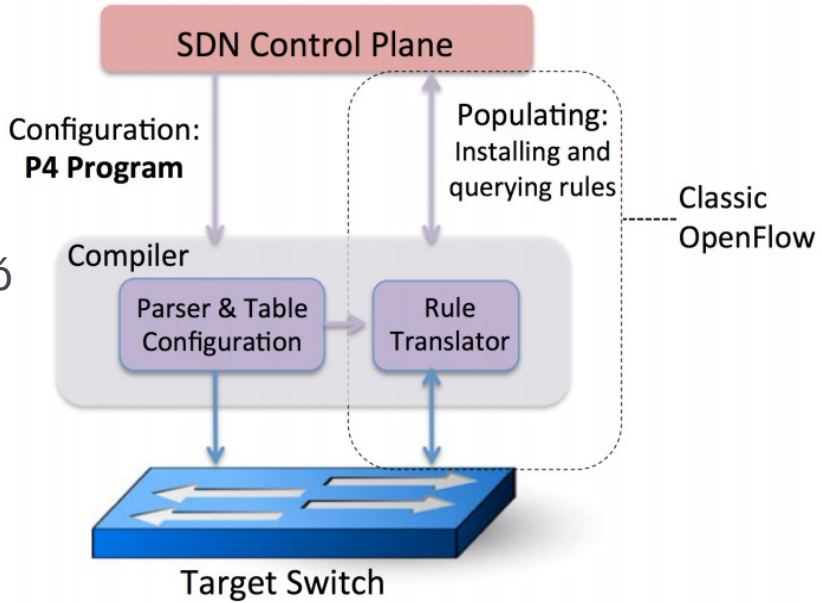
- centralizált kontrollsík
  - skálázhatóság (→ ONOS, ODL)
  - megbízhatóság
- adatsík – kontrollsík szeparálás
  - késleltetés
  - különböző késleltetések
- out-of-band ↔ inband kontroll csatorna
- biztonság
- core-edge szeparáció
  - más elvárások
    - edge: intelligencia, gyors bővíthetőség, SW
    - core: egyszerű, gyors, hatékony, HW
  - OpenFlow a kettő között

OpenFlow evolúciója

- **OF v1.0**
  - legelterjedtebb
  - HW-ek is támogatják
- OF v1.1
  - WAN kiterjesztések
  - több folyam tábla (pipeline)
- OF v1.2
  - IPv6
  - általánosított matching
- **OF v1.3**
  - már több gyártó eszköze támogatja (bizonyos halmazát)
  - (sokszor Open vSwitch alapon)
- OF v1.4, v1.5…

P4: az OpenFlow-n túl

- P4
  - Programming Protocol-Independent Packet Processors
  - data plane programozási nyelv
    - domain-specific nyelv
    - csomagtovábbítás programozására
  - különböző típusú switch-ekre (target) fordítható (compiler)
    - általános célú CPU (SW switch)
    - FPGA
    - NPU (Network Processor)
    - ASIC (HW switch)
      - ma már az ASIC is programozható bizonyos mértékben
      - vonali sebességű működés mellett!!
      - Barefoot Networks: Tofino Chip



Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, David Walker, **P4: Programming Protocol-Independent Packet Processors**, *ACM SIGCOMM CCR*, Volume 44, Number 3, Jul. 2014

Tehát akkor az SDN...

Összefoglalás (Gulyás András slide-jaiból)

Az SDN

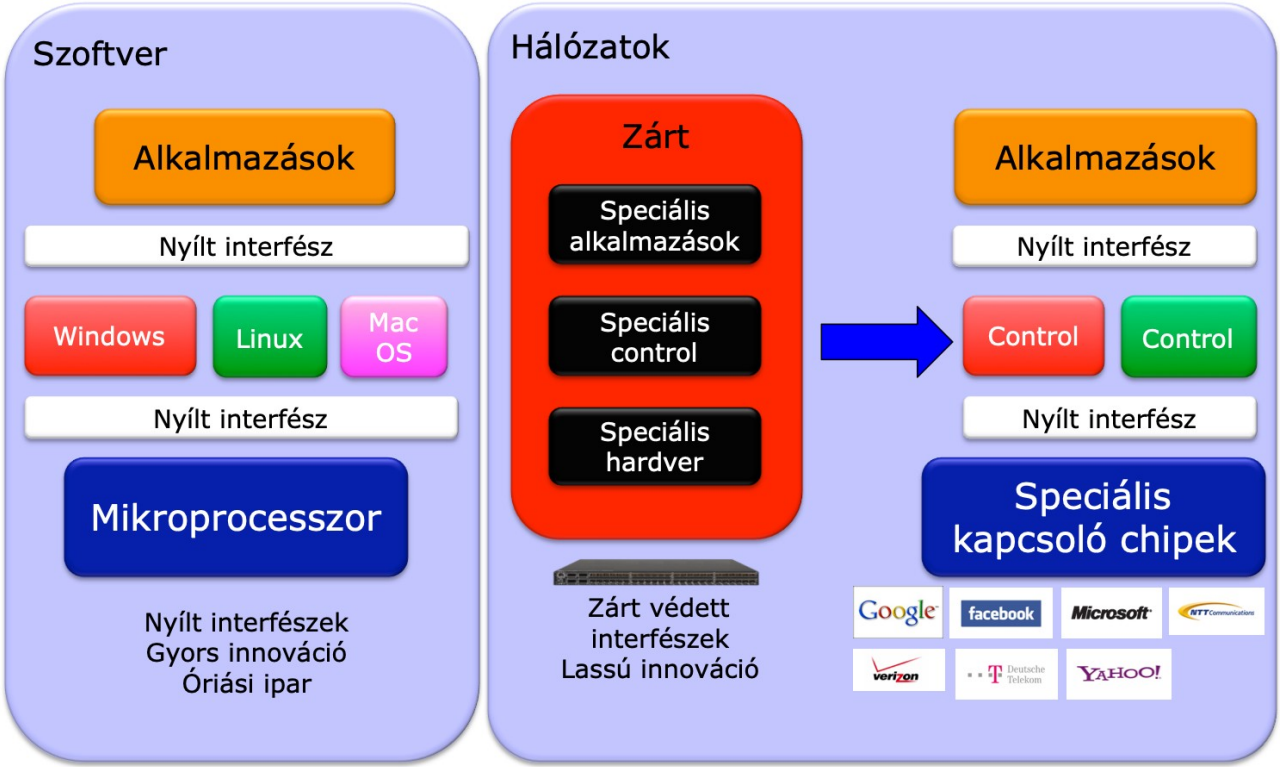
- NEM egy új hálózati működési elv
- NEM egy új algoritmus
- NEM egy új protokoll
- NEM változtatja meg a hálózatok alapvető lehetőségeit és korlátait

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

Az SDN alapkövei

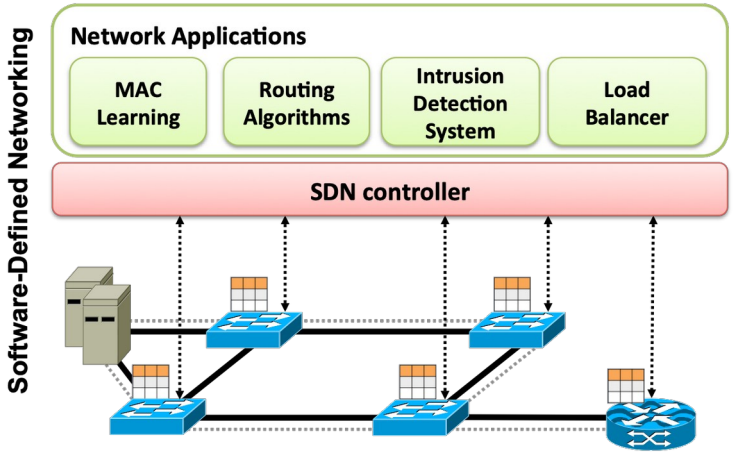
- A vezérlő- és adatsíkok szétválasztása
  - a vezérlési funkciókat kivesszük a kapcsolóeszközökből,
  - amik ezek után egyszerű csomagtovábbító elemekké válnak mindenféle intelligencia nélkül
- A kapcsolási döntéseket nem csomag-, hanem folyamszinten hozzuk meg
- A vezérlési logikát (ami hagyományos IP hálózatokban a switch-ekben/routerekben van) egy külső kontrollerbe, a hálózati operációs rendszerbe (Network Operating System (NOS)) költöztetjük
- A hálózat programozható a NOS felett futó alkalmazások segítségével
  - az alkalmazások kommunikálhatnak a kapcsolóeszközökkel és dinamikusan változtathatják azok viselkedését
  - alkalmazások pl.: routing, tűzfal, terheléselosztó, ...

Analógia a szoftver platformokkal



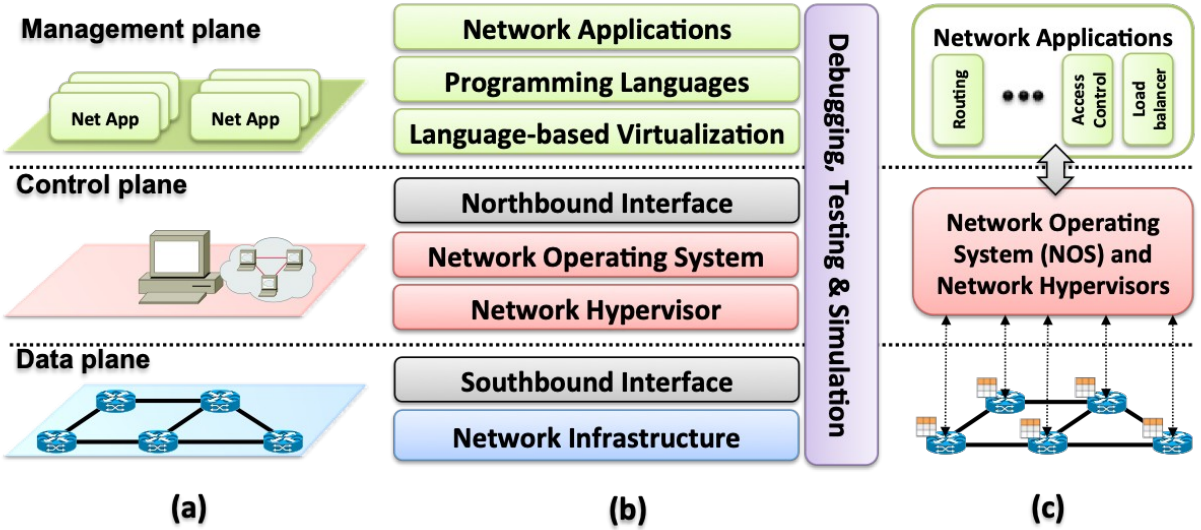
SDN architektúra: síkok és rétegek

- Működési síkok: milyen időskála, milyen szintű feladat
- Menedzsmentsík (SDN-ben net.app. sw)
  - a hálózati működés megadása és nyomon követése
  - pl.: legyen OSPF a hálózatban
- Kontrollsík (SDN-ben szintén net.app. sw, nincs éles határ)
  - kapcsolók tábláinak feltöltése, a működési logika meghatározása
  - **gyors** működés
  - pl.: OSPF futtatása
- Adatsík
  - hálózati (kapcsoló) eszközök + összeköttetések
  - csomagok hatékony és **extrém gyors** továbbítása
  - pl.: OSPF által beállított táblák alapján csomagok gyors továbbítása
- A menedzsmentsík definiálja, a kontrollsík kikényszeríti, az adatsík pedig végrehajtja a kívánt működést



SDN architektúra: síkok és rétegek

- Infrastruktúra
- Déli interfész (pl. OpenFlow)
- Hálózati hypervisor
  - hálózatvirtualizáció
  - bizonyos kapcsolókat, linkeket vagy folyamteret (flow-space) külön NOS vezérelhet
- NOS
  - izoláció, biztonság, konkurens hozzáférés, ...
  - alapvető, közösen használt szolgáltatások (~rendszer könyvtárak)
  - pl.: topológia felderítés, monitorozás
- Északi interfész
  - Hálózati absztrakció
  - NOS API, ~POSIX az OS-eknél



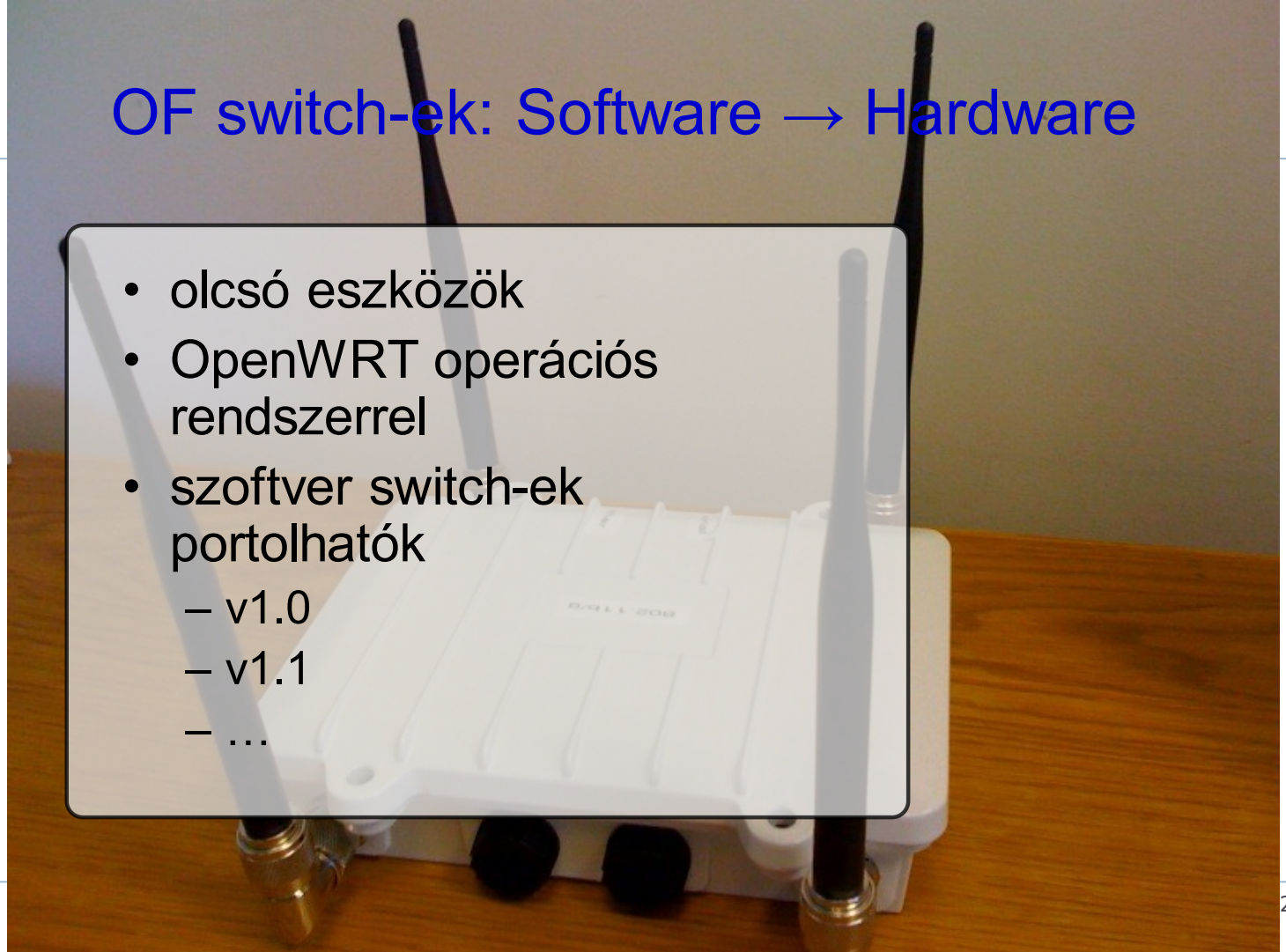
- Programozási nyelvek
  - OpenFlow - Assembly
  - magas szintű nyelvek előnyei
  - “hálózati compiler”
- Hálózati alkalmazások
  - hálózati funkciók, működési logika
  - pl.: routing, multipath, load balancing, firewall, hibatűrés, QoS, mobilitás kezelés, hálózat opt., ...

OpenFlow switch-ek

OF switch-ek: Software → Hardware

- Stanford Referencia implementáció v1.0
- Ericsson, CPqD implementáció v1.1, v1.2, v1.3, v1.4
  - Linux-alapú **szoftver switch (User Space)**
  - hasznos fejlesztéshez & teszteléshez
  - jó alap az egyéb implementációkhoz
- Open vSwitch
  - Linux-alapú **szoftver switch (Kernel Space)**
  - nem csak egy OF switch, virtuális gépek is használják (VirtualBox, XEN, OpenStack)
  - valós HW-ek firmware-e (SW rétege) sokszor Open vSwitch-re épül

OF switch-ek: Software → Hardware



- olcsó eszközök
- OpenWRT operációs rendszerrel
- szoftver switch-ek portolhatók
  - v1.0
  - v1.1
  - …

OF switch-ek: Software → Hardware

- NetFPGA kártya
  - PC-be tehető (PCIe) kártya
  - FPGA-alapú megoldás
  - 4 x 1 Gb/s vagy
  - 4 x 10 Gb/s
- 1G: $500 (akadémiai használat)
- 10G: $1,675 (akadémiai használat)

OF switch-ek: Software → Hardware

Prototype → Product

Core Router

Cisco Catalyst 6k (prototype)

Juniper MX-series (prototype)

HP ProCurve 5400 and others

Enterprise Campus Data Center

Cisco Catalyst 3750 (prototype)

Arista 7100 series (Q4 2010)

Pronto

NEC IP8800

Circuit Switch

Ciena CoreDirector

Wireless

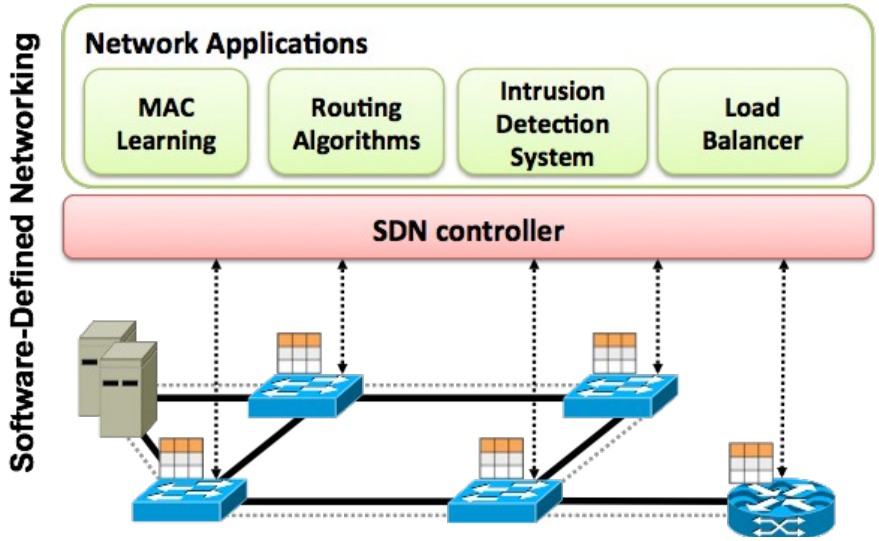
WiMAX (NEC)

more to follow…

OpenFlow kontrollerek

OF kontrollerek

- Mára számos kontroller platform alakult ki
- programozás
  - különböző szoftver környezetben
  - különböző programozási nyelveken
- különböző célok
- különböző teljesítmény



NOX

- Egyik legelső kontroller
- célok
  - hatékony működés
  - jó skálázhatóság
  - hálózati operációs rendszer
- C++ alapú
- sima PC-n: több 10 ezer új folyam kezelése másodpercenként
- jól definiált programozói interfész a hálózathoz
- egyszerűbbé válik különböző gyártók eszközeinek, együttes, centralizált vezérlése

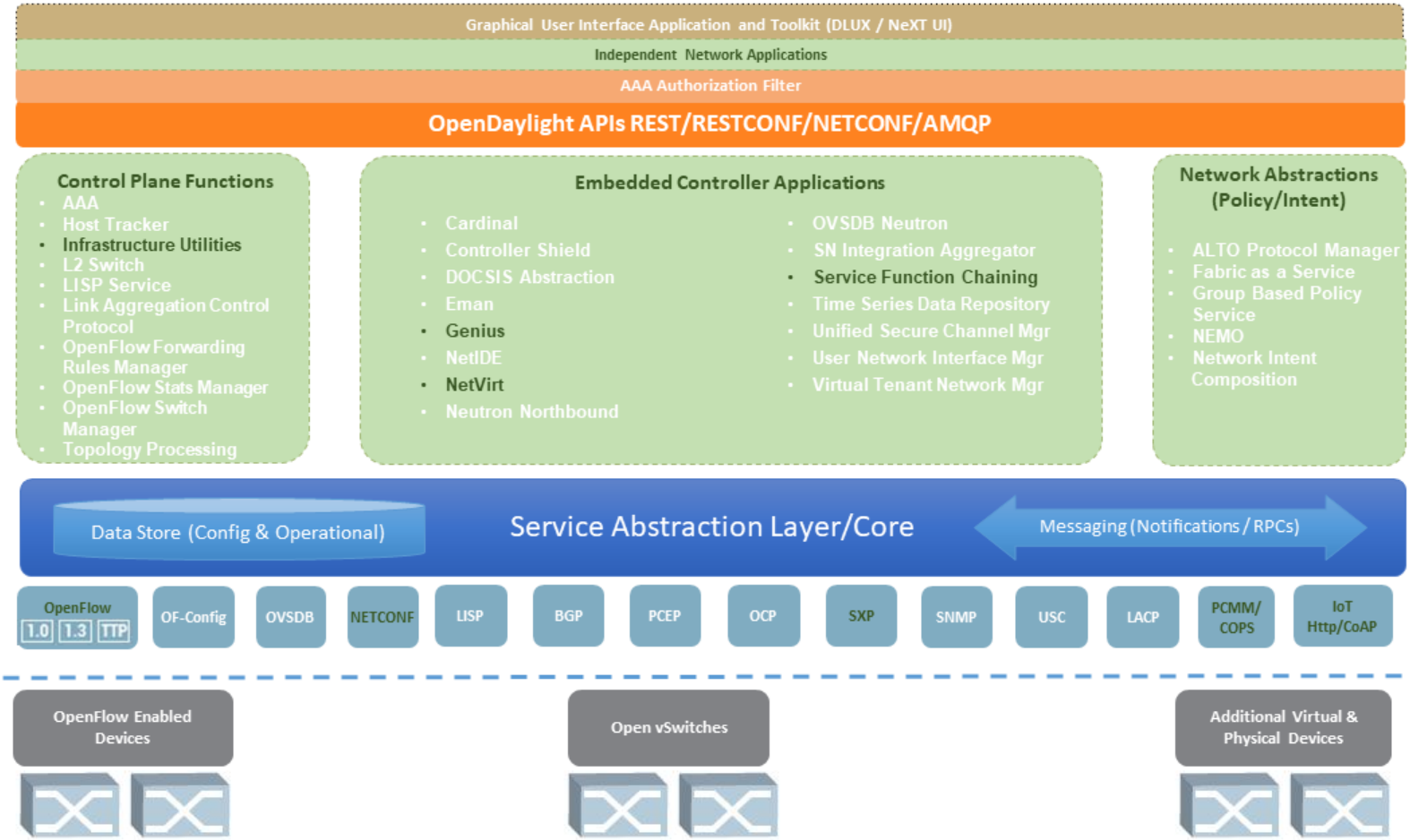
POX

- Python nyelven implementált OpenFlow kontroller
- letisztult, egyszerű programozási környezet
- kontroller alkalmazások Python nyelvű fejlesztéséhez
- gyors prototípus implementálás
- oktatási célok
- hátránya: jelenleg csak az 1.0-ás OpenFlow verziót támogatja
- számos “beépített” alkalmazás elérhető
- POX API-n keresztül nagyon sok hasznos funkció elérhető és felhasználható saját alkalmazások implementálása során

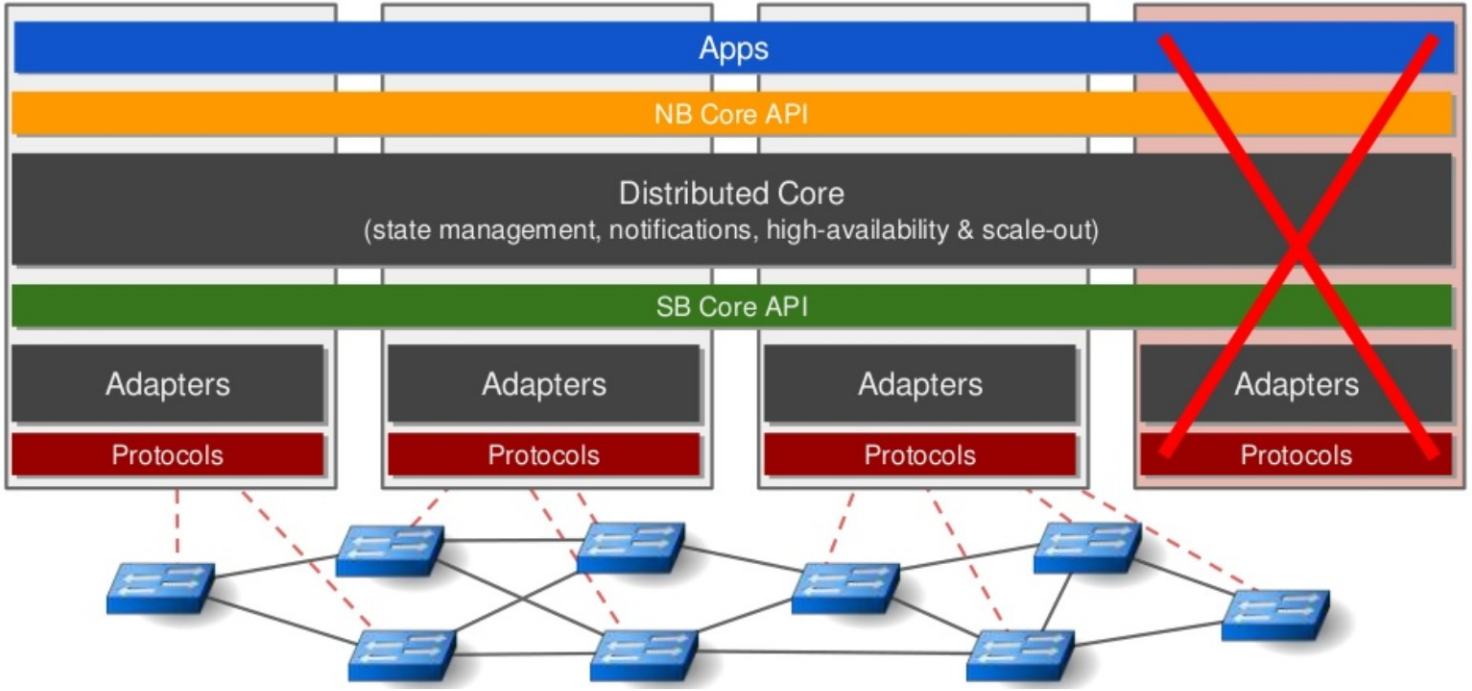
Termékek

- Floodlight
  - Big Switch Networks terméke
  - Java alapú
  - Apache licensz
  - northbound API
- OpenDaylight, ONOS
  - “ipari” SDN platformok

Carbon: Proliferating Use Cases



Distributed Architecture ONOS



- logikai centralizációt biztosít
  - a hálózat fennakadás nélkül üzemel, még ha egy kontrollerpéldányt futtató hardver le is hal
  - az adatok sokszorozását a keretrendszer végzi, a programozónak nem kell ezzel foglalkoznia
- OSGi komponensek (OpenDaylighthoz hasonlóan)

ONOS – intent
Példa egy északi interfészre

- Leírja, hogy minek kéne lennie ahelyett, hogy leírná, hogy hogyan kéne elérni a célt
- Pl.: “Host A és Host B között legyen összeköttetés’’
- ONOS a topológia függvényében ezt egy úttá alakítja és beállítja a kapcsolókban a megfelelő szabályokat
- Ha a topológia változik, akkor az út is változhat, de az északi interfészen nincs forgalom

Egyéb megoldások

- Ryu
  - Python alapú
- Trema
  - Ruby, C
- Frenetic, Nettle
  - deklaratív nyelvek (Haskell, OCaml)
- ...

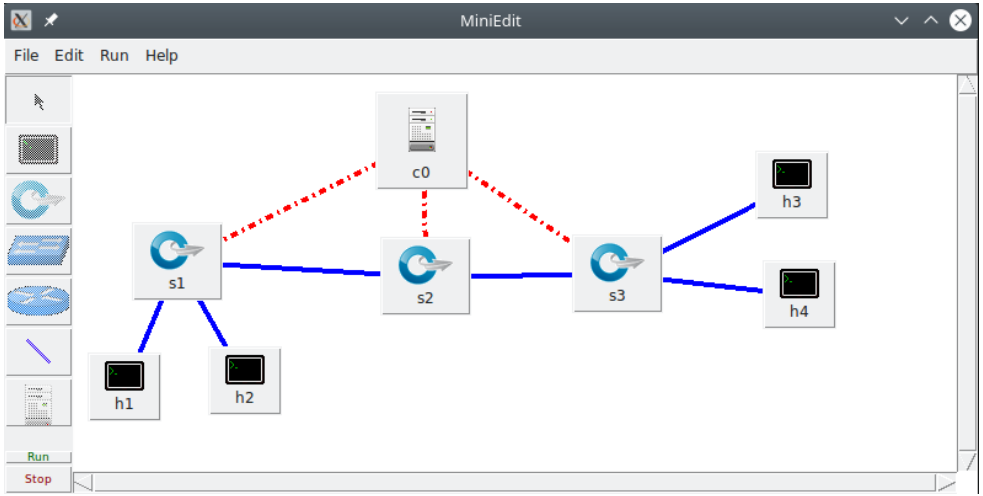
OpenFlow hálózat Mininetben

Példa (HaEpUz VM-ben kipróbálható)

Mininet teszthálózat

- Egyszerű OF (1.0) hálózat
  - 3 OF switch
  - kontroller
  - h1, h2: kliensek
  - h3: ssh szerver
  - h4: web szerver
- Kontroller
  - először manuálisan
  - utána: POX

- ~/mininet/examples
- sudo ./miniedit.py &



Hasznos parancsok

- Switch: Open vSwitch (ezt használják pl. adatközpontokban is)
- parancsok
  - sudo ovs-vsctl show
    - szoftver switch-ek listája, port adatok
  - sudo ovs-ofctl dump-flows s1
    - s1 switch flow táblájának listázása
  - sudo ovs-ofctl add-flow s1 icmp,nw_dst=10.0.0.3,actions=output:3
    - új flow bejegyzés: icmp forgalom 10.0.0.3-as cél IP címre a 3-as porton lesz kiküldve
  - sudo ovs-ofctl del-flows s1 icmp,nw_dst=10.0.0.3
    - előző szabály törlése
  - sudo ovs-ofctl dump-ports-desc s1
    - fizikai portok és OF portok összerendelése
- és a szokásos toolok
  - tcpdump, wireshark, nc, ...

Manuális “kontroller”

- Step 1
  - h1 – h3 között minden IP forgalom továbbítása
    - tesztelés: pl. ping, ssh, web (emulálás: nc)
- Step 2
  - h1,2 – h3,4 között minden ICMP forgalom továbbítása
    - flow rule-ok
- Step 3
  - forgalomirányítás Layer 4 címek alapján
  - h1,2 – h4 (webszerver) közötti forgalom bekonfigurálása
    - flow rule-ok
    - tesztels: web (emulálás: nc)
- Közben flow táblák követése
  - egyszerre több switch-et pl:
    - watch -d 'for i in s1 s2 s3; do sudo ovs-ofctl dump-flows $i; echo "-------------------------\n"; done'

POX kontroller

- POX kontroller indítása (pox alatt a HaEpUz VM-ben)
  - ./pox.py log.color log.level --DEBUG forwarding.l2_learning
  - elindítja a POX platformot +
  - egy Layer 2 switch-et
    - ami megtanulja, milyen MAC cím melyik portján érhető el
- Közben
  - Wireshark: kontroll csatorna (lo) forgalmának vizsgálata
  - flow táblák követése
  - POX log üzenetek
- Tesztek
  - ping: h1 – h3
  - web: h2 – h4

POX, kicsit részletesebben

Németh Felicián slide-jai

POX

- Elavult: csak OpenFlow 1.0-t támogat
  - Konkurensek: OF 1.4+, OF-config, netconf, snmp
- Fejlesztése gyakorlatilag leállt
  - Hibajavítások kivételével
- Ipari igényeket nem elégít ki

- Minimális függőségi lista (python 2.7)
- Könnyen installálható
- Szkript nyelvet használ:
  - gyors edit/(compile)/debug ciklus
  - gyors prototípus implementálás
- Rendkívül elegáns eseménykezelő keretrendszer
- Single threaded: így is gyors, de nehezebb hibázni
- Keretrendszer, amiben alkalmazások írhatók

http://www.noxrepo.org

http://github.com/noxrepo/pox/

https://openflow.stanford.edu/display/ONL/POX+Wiki

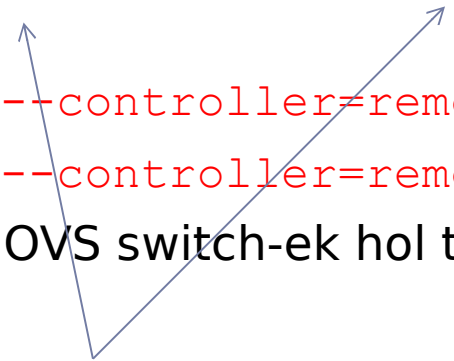
Installáció, futtatás

```
~$ git clone http://github.com/noxrepo/pox
~$ cd pox
~/pox$ git checkout eel

~/pox$ ./pox.py samples.pretty_log forwarding.l2_learning

$ mn --topo=linear --mac --controller=remote
$ mn --topo=linear --mac --controller=remote,ip=192.168.56.1
```

- Controller argumentum: az OVS switch-ek hol találják az OF kontrollert.



POX komponensek (amik azért nem NOS alkalmazások)

Pox komponensek

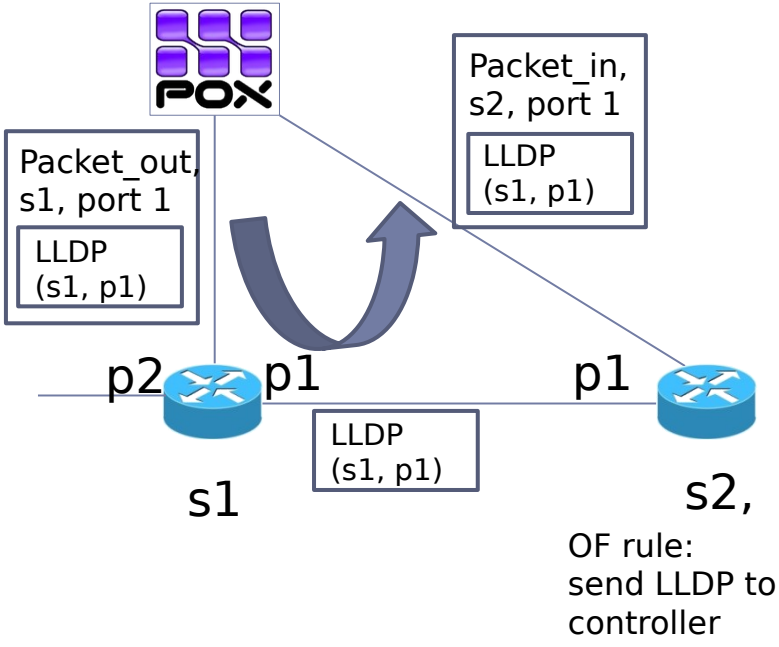
- forwarding.hub
- forwarding.l2_learning
- forwarding.l3_learning
- forwarding.topo_proactive

- Az összes komponens: find ~/pox/pox

- openflow.of_01
  - --port=<X>
  - OF üzenetek küldése, fogadása, POX eseménnyé alakítása
- openflow.discovery
  - Topólogia feltérképezése LLDP üzenetekkel
- openflow.webservice
  - Északi interfész alacsonyszintű OF protokollhoz

openflow.discovery

LLDP: Link Layer Discovery Protocol (EtherType == 0x88cc)



- A packet_in vétele után
  - a kontroller megtanulja, hogy van egy **s1.p1-s2.p1** link
  - Az openflow.discovery küld egy **LinkEvent** üzenetet.
- LinkEventet az kapja meg, aki feliratkozik rá, pl:
- `./pox.py openflow.discovery` **`misc.gephi_topo`** `\`
  `host_tracker forwarding.l2_learning`

POX: eseménykezelés

- Ryu: dekorátorokkal hasonló eredmény:

```
@set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
def _packet_in_handler(self, ev):
    msg = ev.msg
```

```
class GephiTopo (object):
  def __init__ (self):
    core.listen_to_dependencies(self)
    ...

  def _handle_openflow_ConnectionUp (self, event):
    ...

  def _handle_openflow_discovery_LinkEvent (self, event):
    ...


def launch (port = 8282, __INSTANCE__ = None):
  if not core.hasComponent("GephiTopo"):
    core.registerNew(GephiTopo)

  ...
```

az osztály metódusnevei alapján fog az eseményekre reagálni

- Automatikusan meghívódik az openflow osztály ConnectionUp küldésekor
- Illetve az openflow.discovery LinkEvent esemény küldésekor

Regisztálja a GephiTopo osztályt, aminek egy példánya ezután a globális core.GephiTopo változóként elérhető.

Eseménykezelés

```
class GephiTopo (object):
  def __init__ (self):
    core.listen_to_dependencies(self)
    ...

  def _handle_openflow_ConnectionUp (self, event):
    ...

  def _handle_openflow_discovery_LinkEvent (self, event):
    ...


def launch (port = 8282, __INSTANCE__ = None):
  if not core.hasComponent("GephiTopo"):
    core.registerNew(GephiTopo)

  ...
```

- NB:
  - core.**o**penflow.addListeners()
  - core.registerNew(Class, “name”)

```
class GephiTopo (object):
  def __init__ (self):
    core.openflow.addListeners(self)
    core.Discovery.addListeners(self)
    ...

  def _handle_ConnectionUp (self, event):
    ...

  def _handle_LinkEvent (self, event):
    ...


def launch (port = 8282, __INSTANCE__ = None):
  if not core.hasComponent("GephiTopo"):
    core.registerNew(GephiTopo)
  ...
```

Események küldése

```
class LinkEvent (Event):
  """
  Link up/down event
  """
  def __init__ (self, add, link, event = None):
    self.link = link
    self.added = add
    self.removed = not add
    self.event = event # PacketIn which caused this, if any

  ...

class Discovery (EventMixin):

  _eventMixin_events = set([
    LinkEvent,
  ])

  def _handle_openflow_PacketIn (self, event):
    """
    Receive and process LLDP packets
    """
    ...

    link = Discovery.Link(originatorDPID, originatorPort, event.dpid,
                          event.port)
    self.raiseEventNoErrors(LinkEvent, True, link, event)
    ...
```

- raiseEvent vs raiseEventNoErrors
  - Utóbbi esetben a kivételeket automatikusan elkapja a keretrendszer (hiba esetén a program működése nem áll meg)

Lazán csatolt komponensek

- GephiTopo LinkEventre vár; nem számít ki küldi
- Discovery LLDP alapján térképezi fel a hálózati topológiát
- De lehetne írni egy komponenst, ami pl. OSPF hello üzeneteket használna, de ugyanúgy LinkEventeket küldene

```
mininet@mininet-vm:~/pox/pox$ ls lib/packet
arp.py    ethernet.py  ipv4.py  packet_base.py   vlan.py
dhcp.py   icmp.py      ipv6.py  packet_utils.py
dns.py    icmpv6.py    llc.py   rip.py
eapol.py  igmp.py      lldp.py  tcp.py
eap.py    __init__.py  mpls.py  udp.py
```

- libopenflow_01
  - Python objektumok és a bináris hálózati formátum között végez átalakításokat
- openflow.of_01
  - Python objektumok segítségével valósítja meg az OF protokollt
  - Egyes protokolleseményekhez eseményeket küld (pl.: PacketIn)
- Discovery
  - OF eseményeket lekezeli, absztrakt eseményeket (LinkEvent) küld
  - Csomaggenerálásra, -feldolgozásra a lib.packet.* osztályokat használja
- GephiTopo
  - Az absztrakt eseményeket dolgozza fel.
  - RPC adatforrást nyújt a Gephi programnak.

Hub alkalmazás POX kontrollerben

```
def _handle_ConnectionUp (event):
  """
  Be a proactive hub by telling every connected switch to flood all packets
  """
  msg = of.ofp_flow_mod()
  msg.actions.append(of.ofp_action_output(port = of.OFPP_FLOOD))
  event.connection.send(msg)
  log.info("Hubifying %s", dpidToStr(event.dpid))


def _handle_PacketIn (event):
  """
  Be a reactive hub by flooding every incoming packet
  """
  msg = of.ofp_packet_out()
  msg.data = event.ofp
  msg.actions.append(of.ofp_action_output(port = of.OFPP_FLOOD))
  event.connection.send(msg)


def launch (reactive = False):
  if reactive:
    core.openflow.addListenerByName("PacketIn", _handle_PacketIn)
    log.info("Reactive hub running.")
  else:
    core.openflow.addListenerByName("ConnectionUp", _handle_ConnectionUp)
    log.info("Proactive hub running.")
```

proaktív hub:
- ConnectionUp esemény kezelése
- (switch csatlakozása)
- flow bejegyzés összeállítása & leküldése

reaktív hub:
- PacketIn esemény kezelése
- (csomag sw→ ctrl)
- itt csak packet_out (most nincs flow bejegyzés)

kétféle üzemmód:
- reaktív
- proaktív

Hálózati Funkciók Virtualizálása

NFV, SFC

Middleboxok

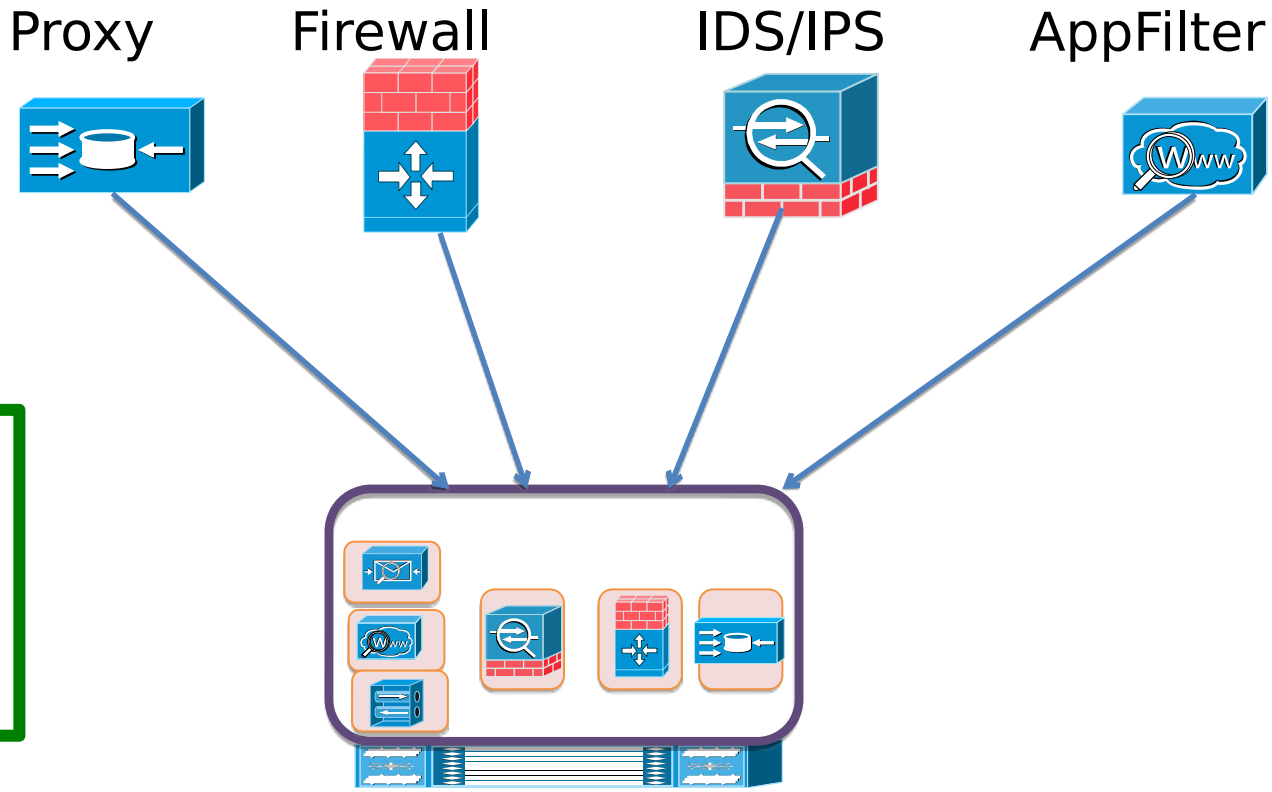
- Mi az a middlebox?
  - minden forgalom/csomag processzáló eszköz, ami nem switch vagy router
  - speciális hálózati funkció
  - pl. NAT, tűzfal, IDS/IPS, DPI, load balancer
- Mennyi van belőlük a hálózatban?
- Felmérés:
  - nagyvállalati hálózati környezet
  - felhasználók >80K
  - telephely n*10

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

Middleboxok ➡ NFV

Implementáció ma:
- önálló egység
- speciális HW berendezés vagy
- switch/router + extra funkció

Tendencia:
- HW → SW
- általános célú HW-en
- SW komponensek
- NFV: Network Function Virtualization

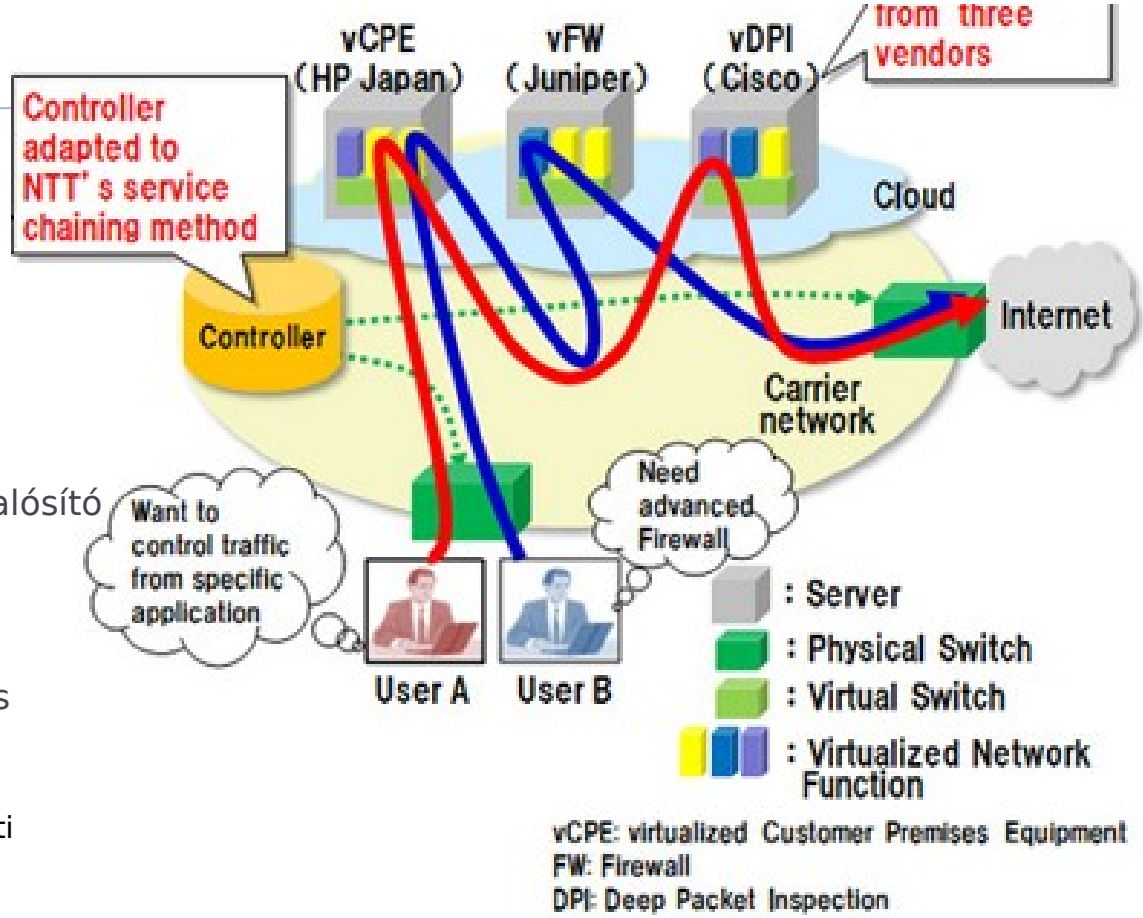


NFV

- Network Function Virtualization
  - vissza az adatsík programozásához
  - aktív hálózatok (lásd korábban)
    - egyéges végrehajtási környezet az adatsík csomópontjaiban (EE, execution environment)
    - akkor nem volt meg a megfelelő HW környezet
  - általános célú szerver HW-ek (pl. x86)
    - hatalmas fejlődés
    - realitás a hatékony csomagfeldolgozás SW alapon!
    - pl. Intel DPDK, Netmap
- NFV-k futtatása
  - cloudban
  - vagy saját adatközpontokban
  - micro/pico adatközpont pl. egy bázisállomásban

SFC

- Service Function Chaining
- Nem új koncepció
- SDN előretörésével fókuszba került
- Tipikus szolgáltatás
  - hálózati funkciók végrehajtása
  - adott sorrendben
  - adott forgalomra
  - csomagok irányítása a funkciót megvalósító blokkok között
- Absztrakció
  - service chain vagy service graph
  - magas szintű szolgáltatások generikus leírására
    - milyen típusú forgalom/felhasználó
    - milyen elemi szolgáltatások (vagy hálózati funkciók)
    - milyen sorrendben



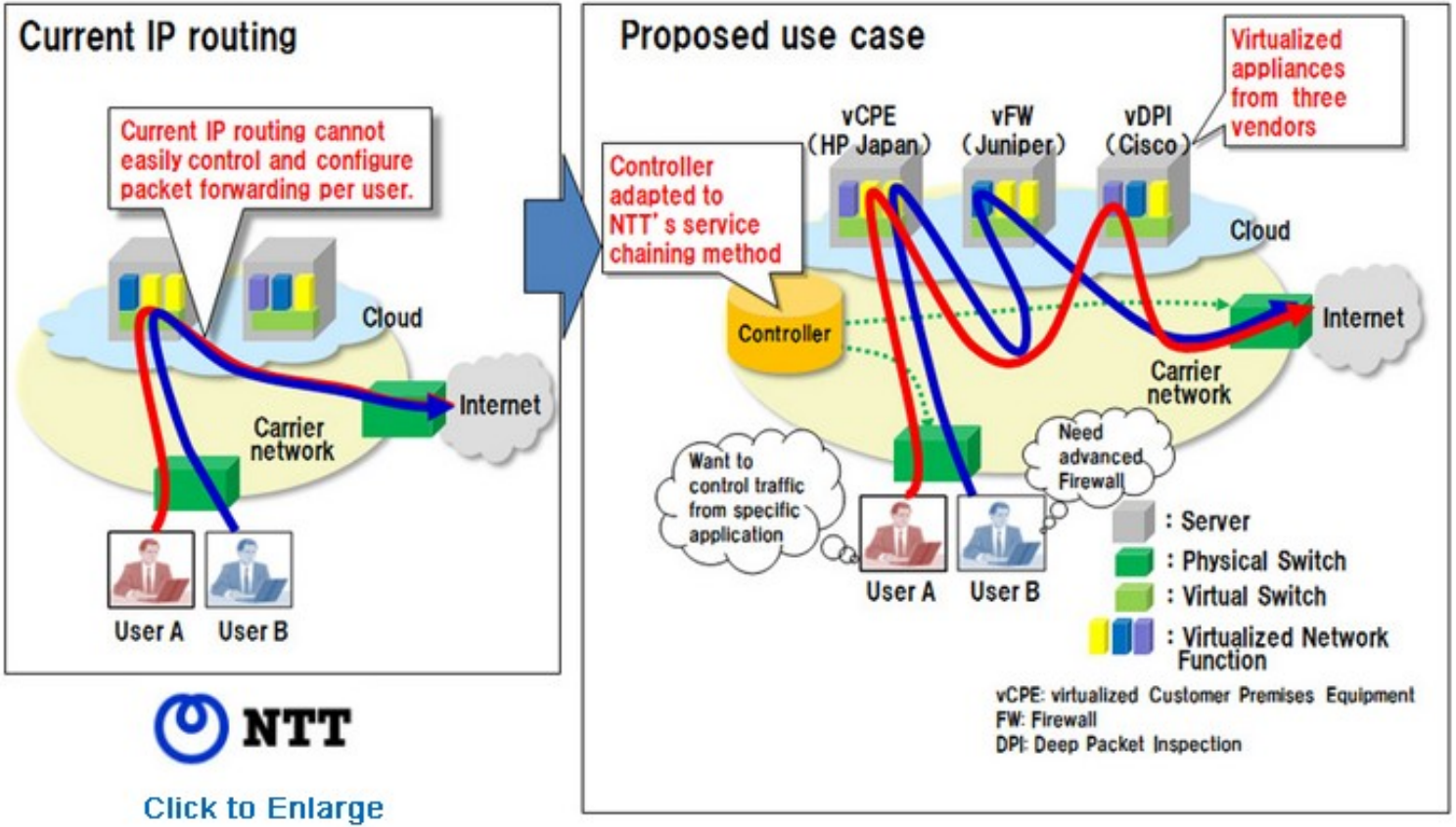
SFC

- Mi a probléma a mai szolgáltatás nyújtással?
  - komoly korlátok
  - erős kötődés a fizikai topológiához
  - speciális képességű, drága middlebox hardverekhez
  - és azok fizikai elhelyezkedéséhez
  - NEM dinamikus
  - NEM flexibilis
  - NEM ad lehetőséget új szolgáltatások gyors bevezetésére
  - NEM jól skálázható
  - NEM garantálható az erőforrások optimális kihasználása
- Következmény
  - service chainek konfigurálása, elhelyezése, üzemeltetése komplex feladat
  - sokszor manuális beavatkozást igényel

Megoldás: SDN+NFV(+cloud)

- SDN
  - forgalom flexibilis irányítása
  - megfelelő funkciók között
  - megfelelő sorrendben
- NFV
  - egy szoftver
  - ami „tetszőleges” környezetben futhat
    - pl. virtuális gép a cloudban
    - Docker konténerben futtatott processz
  - könnyű áthelyezni
  - igény szerint indítani
  - adott szempontok alapján választott fizikai helyen
  - igény szerint le-/felskálázni

- Rapid service provision based on user selection of network function
- Verification of service chaining method in multivendor environment



NTT

Click to Enlarge

Megoldás: SDN+NFV(+cloud)

- Jönnek demók/prezentációk
  - Robotok, robotkarok vezérlése jövőbeli gyárakban
    - Industry 4.0
    - elosztott SW – SFC a felhőben (cloud edge, fog)
    - erőforrás orkesztrációs rendszer
  - Drónok vezérlése 5G hálózatból
    - beltéri drónok vezérlése kamera kép alapján
    - része lehet jövőbeli gyáraknak
    - Industry 4.0

Összefoglalás

- Hálózatok “szoftverizálása” - network softwarization
- control plane szoftverizálása
  - SDN: Software Defined Networking
  - koncepció + történelem
  - egy konkrét példa: OpenFlow
- data plane szoftverizálása
  - NFV: Network Function Virtualization
- hálózati szolgáltatások/alkalmazások szoftverizálása
  - SFC: Service Function Chaining