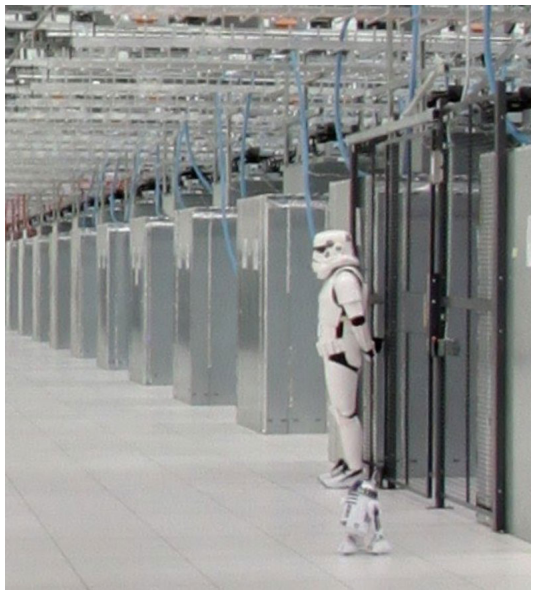
HPC

HPC Clusters

- High Performance Computing
- Solían emplear otras tecnologías como Infiniband o Myrinet
- Les ofrecían bajo retardo y alto throughput, así como técnicas de *Remote Direct Memory Access* (RDMA)
- Hoy en día Ethernet ofrece rendimiento similar
- Clusters basados en paso de mensajes
  - También llamado IPC (Inter-Process Communication)
  - Aplicación distribuida entre los nodos del cluster con acoplamiento fuerte
  - Eso quiere decir que se comunican frecuentemente los nodos entre sí y alto retardo tiene un impacto fuerte en el rendimiento
  - Los supercomputadores son de este tipo
- (…)

HPC Clusters

- High Performance Computing
- Solían emplear otras tecnologías como Infiniband o Myrinet
- Les ofrecían bajo retardo y alto throughput, así como técnicas de *Remote Direct Memory Access* (RDMA)
- Hoy en día Ethernet ofrece rendimiento similar
- Clusters basados en paso de mensajes
- Clusters para el procesado de datos de I/O
  - Las peticiones de los clientes se reparten balanceadas
  - Procesado en paralelo de múltiples peticiones
- (…)



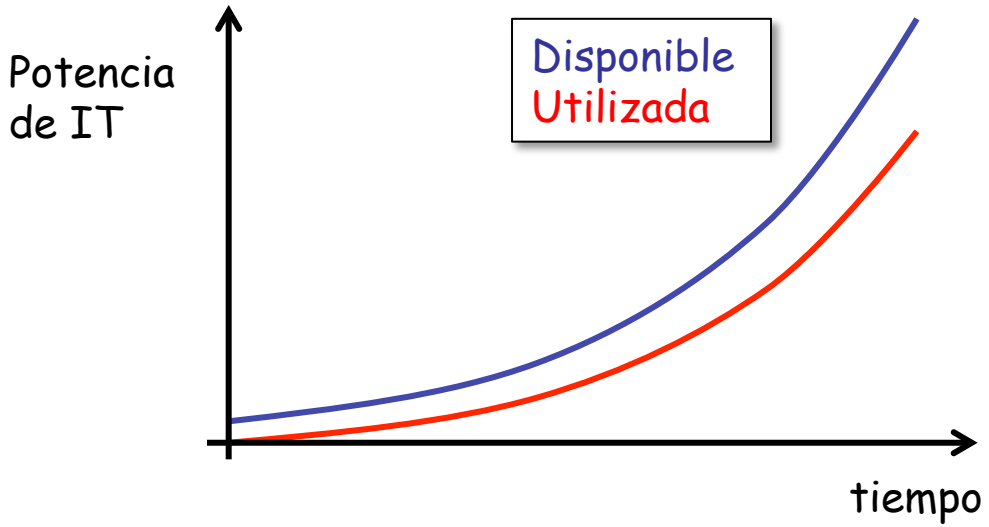
HPC Clusters

- High Performance Computing
- Solían emplear otras tecnologías como Infiniband o Myrinet
- Les ofrecían bajo retardo y alto throughput, así como técnicas de *Remote Direct Memory Access* (RDMA)
- Hoy en día Ethernet ofrece rendimiento similar
- Clusters basados en paso de mensajes
- Clusters para el procesado de datos de I/O
- Clusters de procesado de ficheros
  - La petición se divide y se distribuyen las piezas para el procesado
  - Posteriormente se unen los resultados

Cloud

Hace una década

- Un proveedor de un servicio en red necesita planificar su crecimiento
- Eso le lleva a comprar recursos hardware por adelantado
- Los tiene instalados a la espera de que aumente la demanda
- Eso consume potencia eléctrica inútilmente mientras no hacen falta
- Si crece muy rápido pueden ser muchos recursos
- Puede plantearse comercializar esos recursos sobrantes
- Cuando no los necesita los usan otros clientes
- Si los necesita se los “quita” a esos clientes
- *“Elastic computing”, “hyper virtualization”*

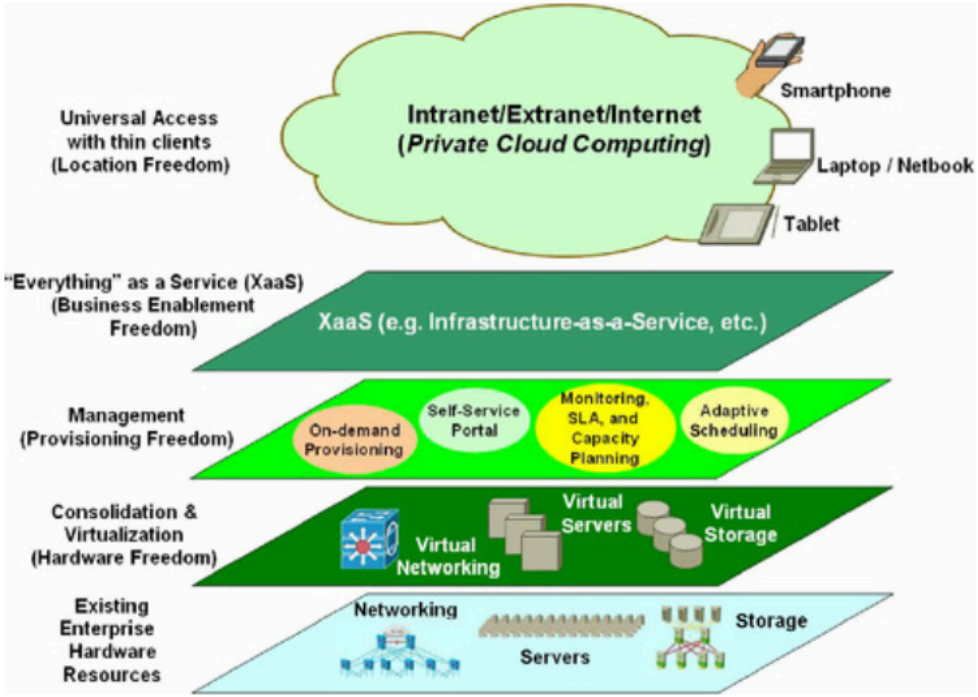


Virtualización en la red

- Para la separación de la red entre departamentos o clientes (*tenants*)
- Sencillo cuando son departamentos de una empresa, complejo cuando son centenares o miles de clientes
- En especial si además queremos tener movilidad de VMs (requiere extender la LAN capa 2)
- La solución era las VPNs (layer 2 ó 3)
- Reconfiguración de servidores y red en un entorno con decenas o centenares de equipos, de diferentes fabricantes
- La “orquestración” es un problema

SOI

- *Service Oriented Infrastructure framework*
- Hardware físico (servidores, almacenamiento, routers, switches)
- Virtualización (de servidor, red o almacenamiento)
- Gestión y *provisioning*
- "X" as a Service



Cloud Computing

- *On-demand self-service*
  - El usuario puede crear nuevas instancias de servidores, almacenamiento o red por su cuenta
- *Universal network access*
  - Acceso mediante tecnologías estándar desde cualquier plataforma
- *Resource pooling*
  - Recursos compartidos entre diferentes *tenants*
- *Rapid elasticity*
  - *Provisioning* rápido o automático para un rápido *scale-out* y *scale-in;* parecen recursos ilimitados para el usuario
- *Pay per use*

Public cloud

- Disponible para el público general o una gran industria
- Propiedad de una organización que vende estos servicios
- Ofrecidos típicamente a través de la Internet

Community cloud

- Compartida por varias organizaciones
- Tienen características similares (misión, requerimientos de seguridad, políticas, cumplimiento necesario de regulación, etc)
- Gestionada por las organización o por un tercero
- En sus propios edificios o en otros (*on-premises vs off-premises)*

Private cloud

- Empleada por una única organización
- Puede ser gestionada por la misma organización o por otra (servicio externalizado)
- Puede encontrarse en sus edificios o en otros

Hybrid cloud

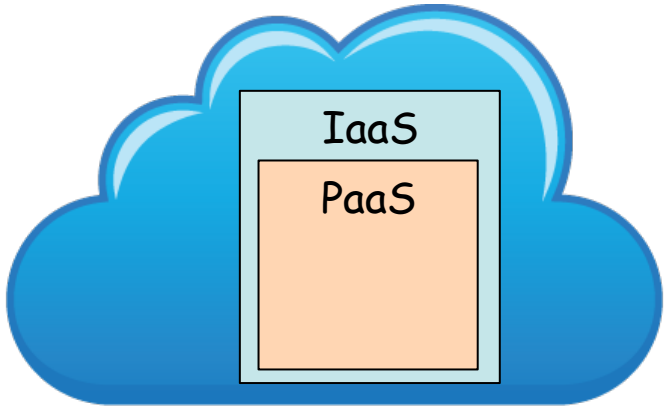
- Utilización de infraestructura de al menos dos de los tipos anteriores para las mismas aplicaciones
- Por ejemplo:
  - Una empresa tiene su *private cloud* con recursos limitados
  - Cuando alcanza los límites, las peticiones en exceso se redirigen a una *public cloud*

IaaS

- El cliente puede instanciar recursos de servidor, almacenamiento y/o red
- Tiene acceso a los servidores (virtuales) para poder emplear el sistema operativo que quiera
- Tiene control sobre esos sistemas operativos para instalar el software que necesite
- Ejemplos: Amazon EC2, CenturyLink Cloud, Microsoft Azure, Terremark vCloud Express, Arsys Cloud, Fujitsu IaaS Trusted Public S5

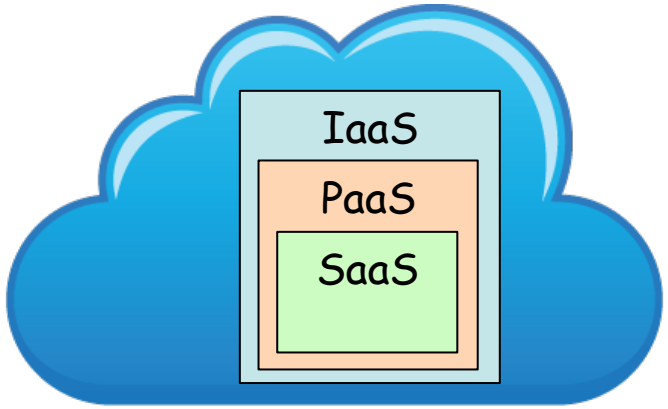
PaaS

- El cliente puede desplegar sus aplicaciones sobre la infraestructura
- Deben estar creadas empleando los lenguajes de programación y utilidades soportadas por ella
- No tiene control sobre la infraestructura
- Tiene control sobre las aplicaciones y tal vez su entorno de *hosting*
- Ejemplos: Google App Engine, AWS Elastic Beanstalk, OpenShift, Salesforce



SaaS

- El cliente emplea las aplicaciones ofrecidas que están “en la nube” en lugar de instalarlas en sus equipos *on premises*
- Son accesibles desde diversos tipos de dispositivos
- No tiene control sobre la infraestructura, ni sistemas operativos, ni almacenamiento ni instalación de aplicaciones
- Ejemplos: Google Apps, Office 365, SharePoint Online, Cisco WebEx



SDN

Internet

- Simple
- Control distribuido
- Ha permitido su gran crecimiento
- Muy bueno para los fabricantes de routers
- Sin embargo, ese crecimiento lleva a ser *commodity* (mercancía)

Equipos hardware

- La funcionalidad de red la dan equipos dedicados: switches, routers, ADCs, etc
- Dependientes de la implementación de funcionalidades en ASICs
- Eso hace su evolución muy lenta
- Y los hace propietarios respecto al desarrollador del ASIC
- Hoy en día estos ASICs y la conmutación en general son "*commodity*"
- El problema está en implementar nuevas funcionalidades
- La implementación de nuevos servicios es muy lenta en ASICs
- Es un entorno mucho menos flexible que el entorno software
- Por ejemplo si tenemos que esperar a que el fabricante del equipo corrija un bug o implemente una funcionalidad

¿Qué se necesita?

- Virtualización
  - Para emplear mejor los recursos
  - En el servidor, en la red, en el almacenamiento
- *Orchestration*
  - Una forma de controlar y gestionar miles de dispositivos, físicos y virtuales
- Programable
  - Poder crear software que cambie cómo se comporta
  - No necesitar cambiar el hardware, que es más lento
- Escalado dinámico
  - Poder aumentar el tamaño según necesidades
- Automatización
  - Reducir el OpEx
- Visibilidad
  - Monitorizar los recursos, conectividad, etc
- Rendimiento
  - Traffic engineering, balanceo de carga, alta disponibilidad, etc
- Multi-tenancy
- Openness

¿SDN?

- *Software Defined Networking*
- ¿Qué es?
- Esto tiene poco tiempo (5-10 años)
- Aunque se basa en ideas que tienen bastantes más edad
- Hay confusión en los últimos años en lo que significa SDN
- ¿SDN = Un API estándar para configurar switches?
- ¿SDN = Separación del plano de datos y de control?
- ¿SDN = Plano de control centralizado?
- ¿SDN = OpenFlow?
- El movimiento actual sí empezó con OpenFlow pero hoy en día consideramos que no son lo mismo

¿Por qué SDN?

¿Por qué SDN?

- (Basado en la presentación de Scott Shenker en el ONS 2011)
- https://youtu.be/YHeyuD89n1Y

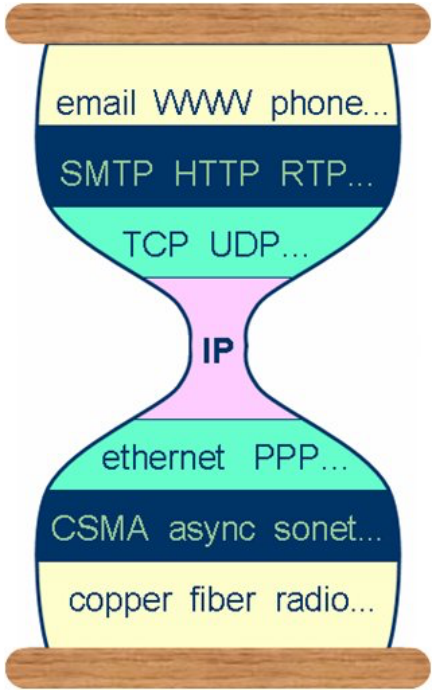
Scott Shenker
ICSI Berkeley

Martin Casado
Stanford, Nicira, VMware

Nick McKeown
Stanford

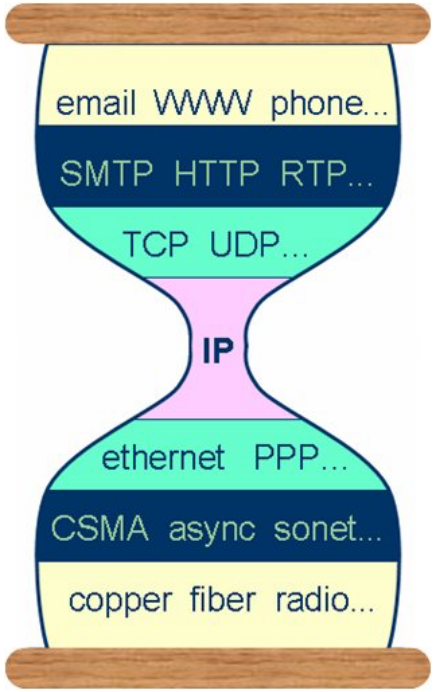
Internet: Por qué

- ¿Por qué ha tenido éxito Internet?
  - Hemos podido construir aplicaciones
  - Sobre un transporte fiable
  - Sobre una red global best-effort
  - Sobre redes locales principalmente best-effort
  - Sobre el transporte de bits en un medio
- En realidad de todo eso queremos las aplicaciones
- El resto es algo que “necesitamos”
- No que “queramos”



Layers: Por qué

- Nos permiten separar responsabilidades
- Puedes cambiar una capa sin modificar el resto
- Eso te permite innovar sin tener que cambiar todo
- Por ejemplo todas las aplicaciones que se han desarrollado no han necesitado la colaboración de la red

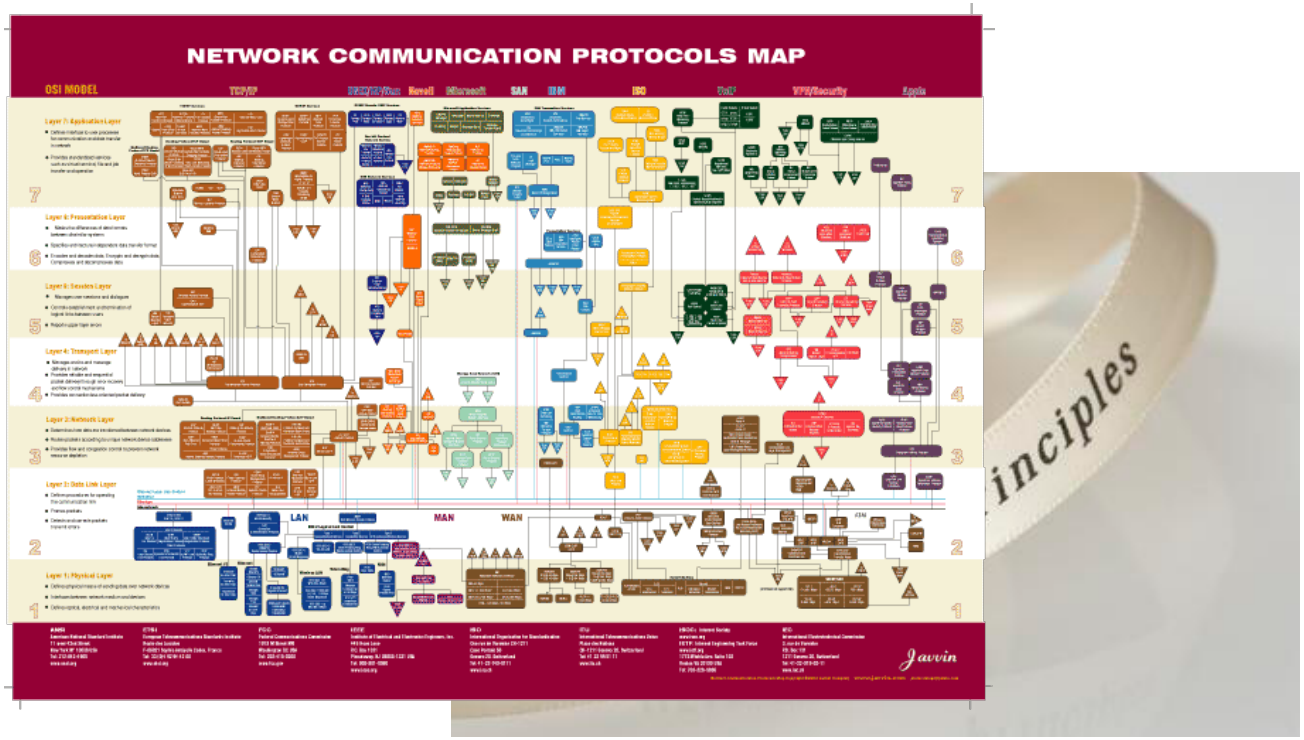


Networking como disciplina

- En física se estudian los principios del universo, leyes físicas
- En Computer Science (Informática) principios sobre sistemas operativos (memoria, procesos, planificación, sincronización), bases de datos, etc.
- En Networking se estudian…

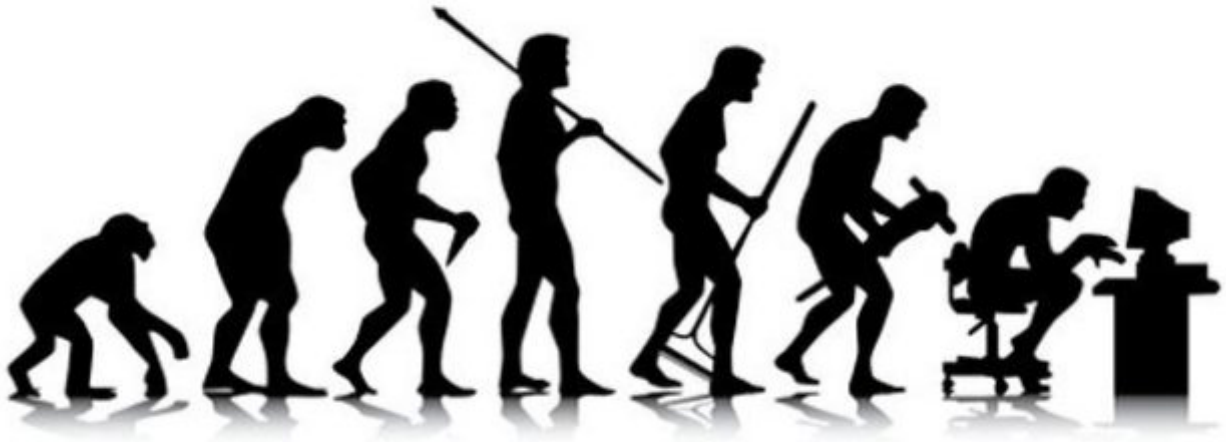
Networking como disciplina

- En física se estudian los principios del universo, leyes físicas
- En Computer Science (Informática) principios sobre sistemas operativos (memoria, procesos, planificación, sincronización), bases de datos, etc.
- En Networking se estudian guías de diseño, protocolos y más protocolos???



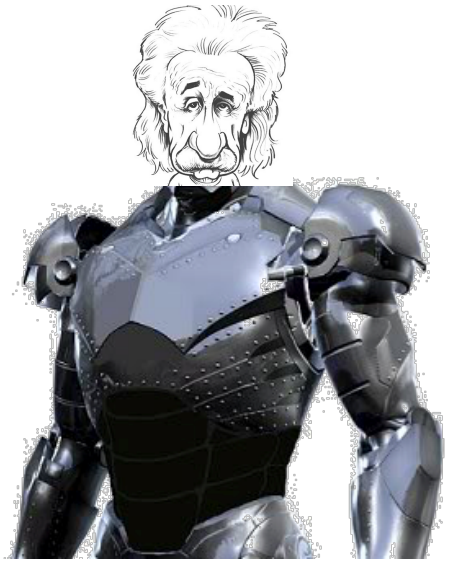
Evolución

- El entorno informático evoluciona rápidamente
  - Tenemos virtualización en el sistema operativo y en el almacenamiento
  - Es hoy en día sencillo de gestionar y provisionar (mediante software)
  - Nuevos lenguajes de programación, nuevos sistemas operativos, todo ello simplifica el desarrollo de nuevos servicios
- Las redes no
  - Seguimos empleando el CLI a cada equipo; gestión primitiva
  - Cambiar los protocolos es lento pues requiere la colaboración de los fabricantes de los equipos
  - Y muchas veces cambiar el hardware

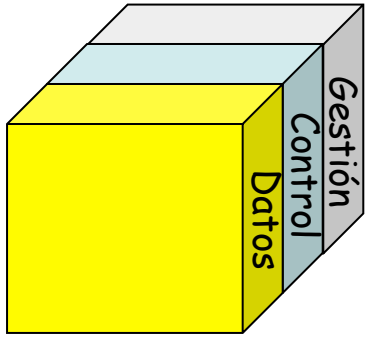


Evolución

- Las redes eran simples
- Ethernet, IP, son tecnologías muy simples
- Pero les hemos añadido nuevas necesidades
- En la parte del control de las mismas
- Y se nos han descontrolado
- ACLs, VLANs, Traffic engineering, Firewalls, VPNs, ADCs, DPI, TRILL, SPB, VXLAN, etc
- Las hacemos funcionar, porque tenemos en la cabeza cómo funcionan todos esos protocolos



Planes



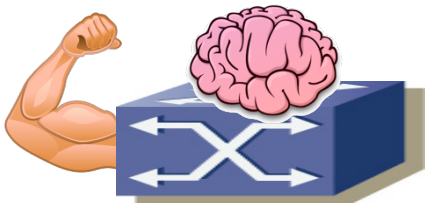
- *Data plane*
  - Paquetes de usuarios
  - Reenvío, fragmentación, replicación para multicast, etc
  - Simple y rápido (*fast path*)
- *Control plane*
  - Actividades necesarias para que funcione el plano de datos
  - Crear tablas de rutas, configurar políticas, anunciar servicios, etc
  - Complejo pero no necesita ser rápido (*slow path*)
- *Management plane*
  - Control opcional
  - Gestión de fallos, configuración, accounting, performance, seguridad, monitorización, provisionamiento, etc
  - Complejo pero no necesita ser rápido

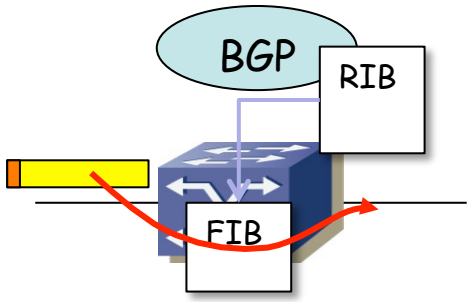
Complejidad y abstracciones

- La forma que tenemos de simplificar el problema es la modularidad
- Dividir el problema en problemas/módulos más pequeños
- Crear interfaces entre esos módulos

Control vs Data Plane

- *Data plane*
  - Conmutación, reenvío layer 2, reenvío IP (el “músculo”)
  - Aquí tenemos las abstracciones de las capas
- *Control plane*
  - Señalización y control, routing protocols (aprendizaje, el “cerebro”)
  - Los datos empleados para conocer la topología
  - Aquí no tenemos una forma abstracta de resolver el problema
- (…)





Control plane

- ¿Cómo hacemos evolucionar hoy en día el plano de control?
- Nos inventamos un nuevo protocolo desde cero (y esperamos a que se implemente, compremos hardware que lo soporte, etc)
- O reconfiguramos algún mecanismo existente (por ejemplo para hacer ingeniería de tráfico)
- O hacemos configuración manual (ACLs, middleboxes, routers domésticos, etc)

Control plane

- ¿Qué hacemos en el plano de control?
- Calcular la configuración de los dispositivos
  - Tablas de rutas
  - Listas de filtrado
  - Etc
- Tenemos que desplegar esta configuración a través de una red sin garantías
- Tenemos que desplegarla empleando el protocolo que tenemos

Analogía informática

- Calcular la configuración de los dispositivos
- Desplegarla a través de una red sin garantías
- Y empleando el protocolo que tenemos
- Especificar cómo es y dónde está cada bit del programa en memoria
- Gestionar los recursos hardware y todos los problemas que surjan para emplear la CPU, la memoria, etc
- Y lo tiene que hacer en Cobol

- ¿En realidad cómo lo hace un programador?
  - No se preocupa de cuáles son los bits resultado del programa ni de dónde está cada uno
  - El sistema operativo se encarga de esa gestión y comunicación
  - Desarrolla un lenguaje de programación de más alto nivel
- Es decir, resuelve el problema en cuestión, no todos los añadidos colaterales para hacerlo funcionar

Abstracciones

- Estado distribuido
  - Tenemos que distribuir el estado (la configuración)
  - Esto ya está resuelto, no queremos volver a resolverlo
  - ¿Cuántas páginas de la RFC de OSPF son para distribuir LSAs?
  - Para un problema queremos poder suponer que lo resolvemos de forma centralizada
  - No preocuparnos de cómo luego se distribuya
- Especificación
  - El programa de control lo que quiere es especificar algo
  - Su problema no debería ser implementarlo
  - Necesitamos una abstracción de la red (virtualización)
- Forwarding
  - Modelo para el comportamiento del plano de datos
  - Independiente del equipo, fabricante, etc

SDN

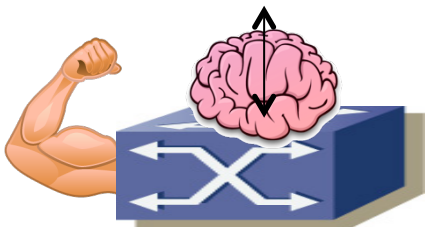
- Pretende ofrecer estas abstracciones
- No el “cómo”
- No es una solución a un problema, no es un nuevo protocolo o una nueva tecnología
- Es una nueva arquitectura
- Es un facilitador de nueva innovación
- *“Software Defined Networking”*
- Networking quiere evolucionar hacia el software
- El software definirá las redes, cómo se comportan, etc
- La red pasa a ser infraestructura abstracta como para las redes lo son los enlaces
- Que podremos administrar y controlar automática y dinámicamente
- Hablamos de esto solo desde hace menos de 5-10 años
- Todas las organizaciones están hablando ahora de ello: IETF, ITU, ONF, ETSI, etc.

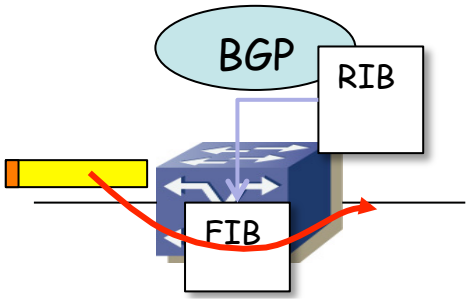
Elementos en SDN

Control vs Data Plane

- En Internet el desarrollo se hizo basado en un control distribuido
- Es decir, ambos están en el mismo equipo, implementados por el fabricante
- *Software Defined Networking* (SDN) se basa en la separación de ambos y comunicación mediante un interfaz abierto (…)
- La propuesta del Open Networking Forum (ONF) es OpenFlow
- Eso no quiere decir que SDN sea igual a OF

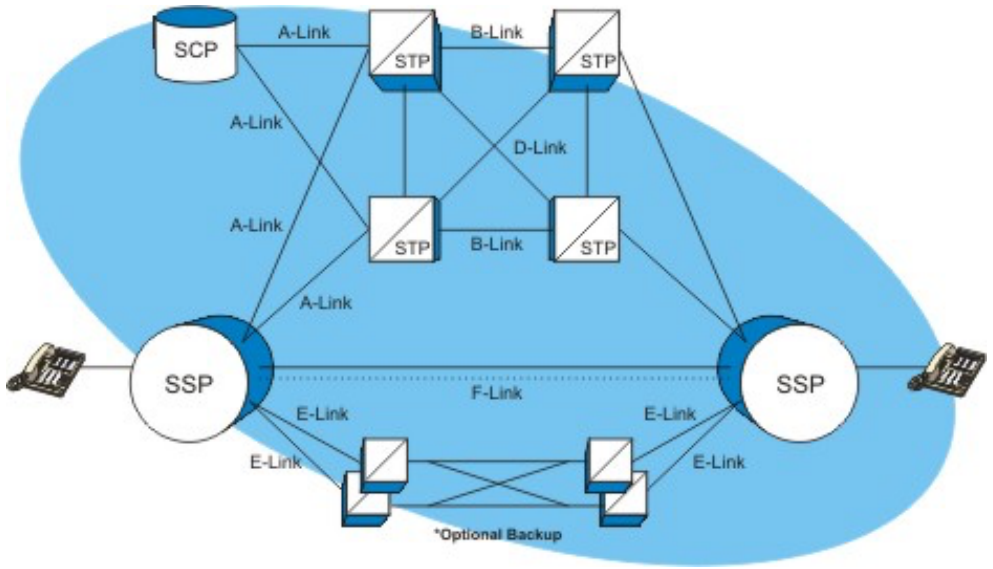
¿ OpenFlow ?





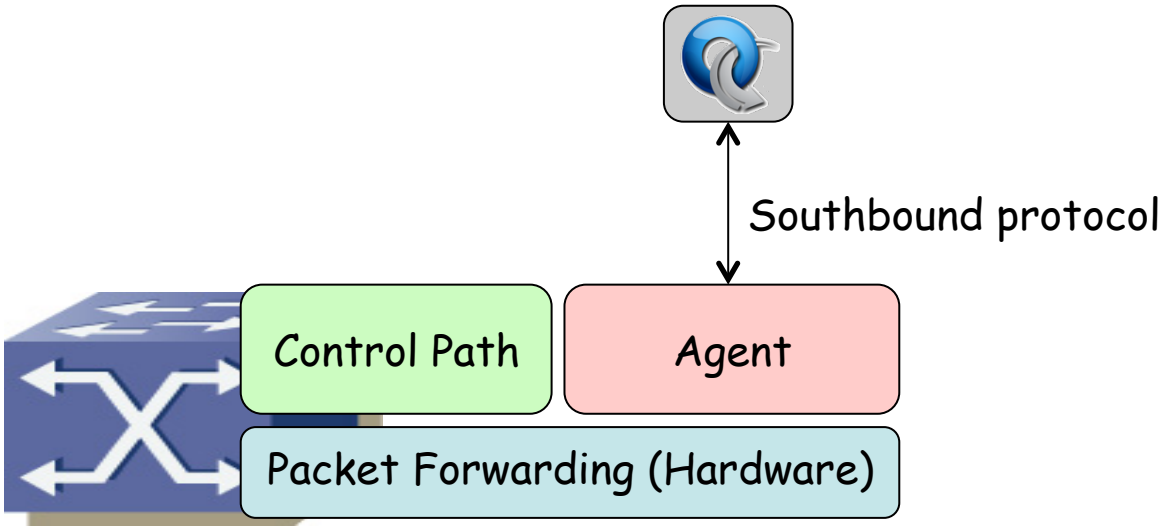
Separación control-datos

- Simplifica la evolución del control pues no está “atado” al hardware
- Permite el desarrollo de software de más alto nivel, así como su depuración, testing, simulación, etc
- La red telefónica ya tenía separado el control a elementos de señalización y control de red
- Especialmente útil en data centers y en IXPs
- Permite la optimización de los flujos
- También para una arquitectura con middleboxes
- También en el entorno WAN controlado por la misma empresa



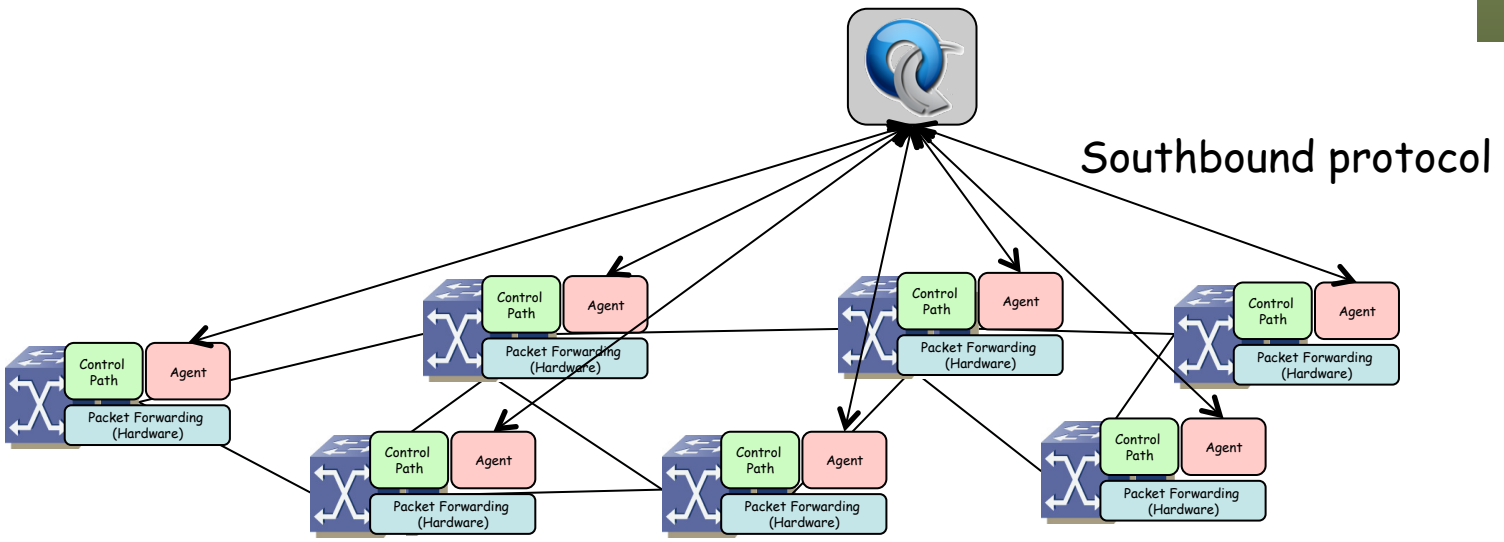
Controlador

- Esta arquitectura permite tener centralizado el plano de control
- Hoy en día el concepto de SDN no obliga a tenerlo centralizado
- Se comunica con el dispositivo mediante un *Southbound protocol*
- Para un gran número de dispositivos (…)



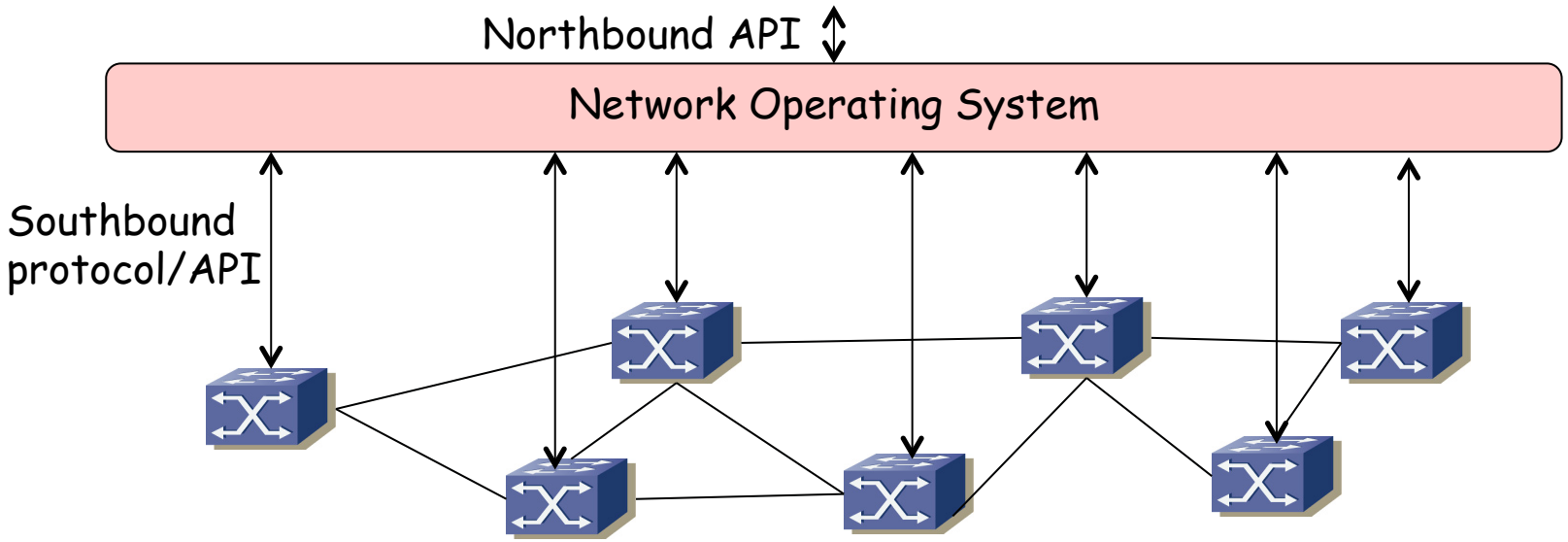
Controlador

- Esta arquitectura permite tener centralizado el plano de control
- Hoy en día el concepto de SDN no obliga a tenerlo centralizado
- Se comunica con el dispositivo mediante un *Southbound protocol*
- Para un gran número de dispositivos



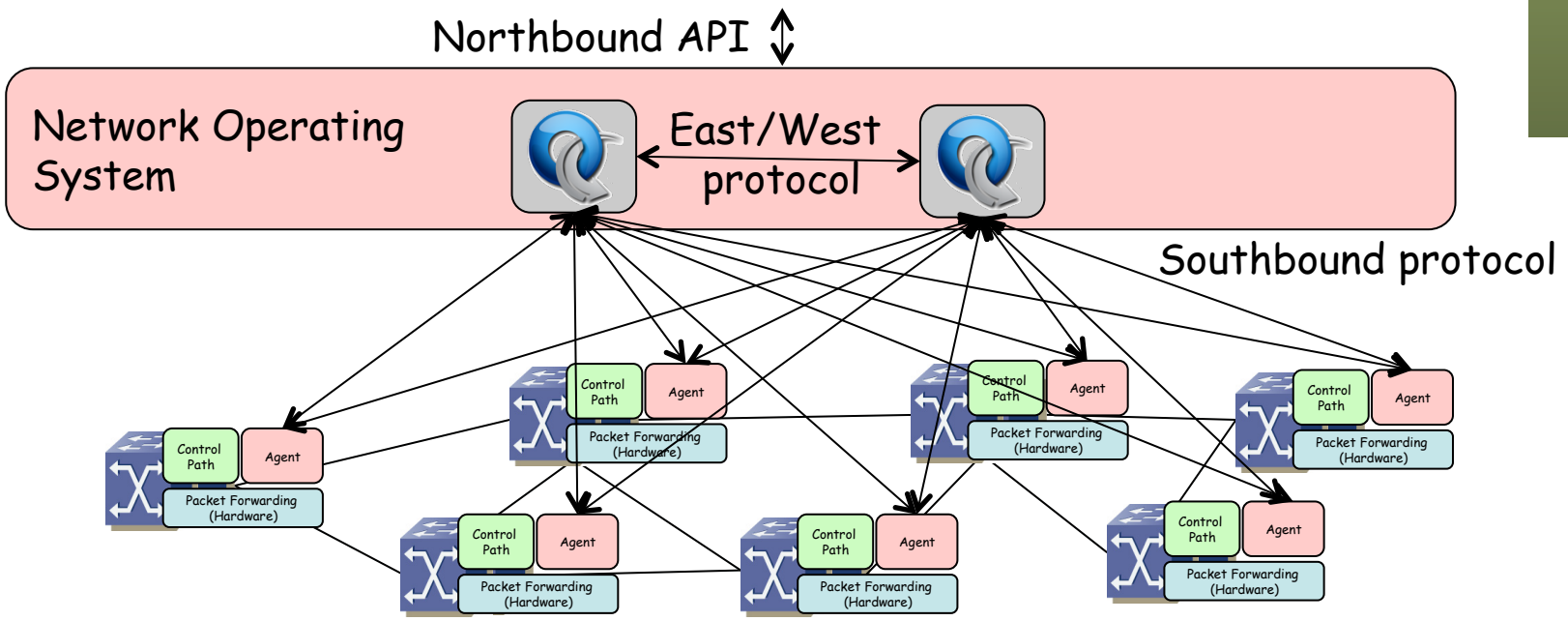
Network Operating System

- Tenemos una visión global de la red
- Mediante lo que se está viniendo a llamar un NOS
- El NOS es software en servidores que habla con los conmutadores
- El NOS da una visión virtualizada de la red, un grafo y un API ("*northbound*")
- Sobre ella podemos escribir los programas de control
- Nos aísla del hardware, igual que un OS del hardware del PC



High Avilability

- Para mayor disponibilidad no tendremos un solo controlador sino varios
- La comunicación entre los controladores se lleva a cabo mediante lo que se llama un *East/West Protocol*

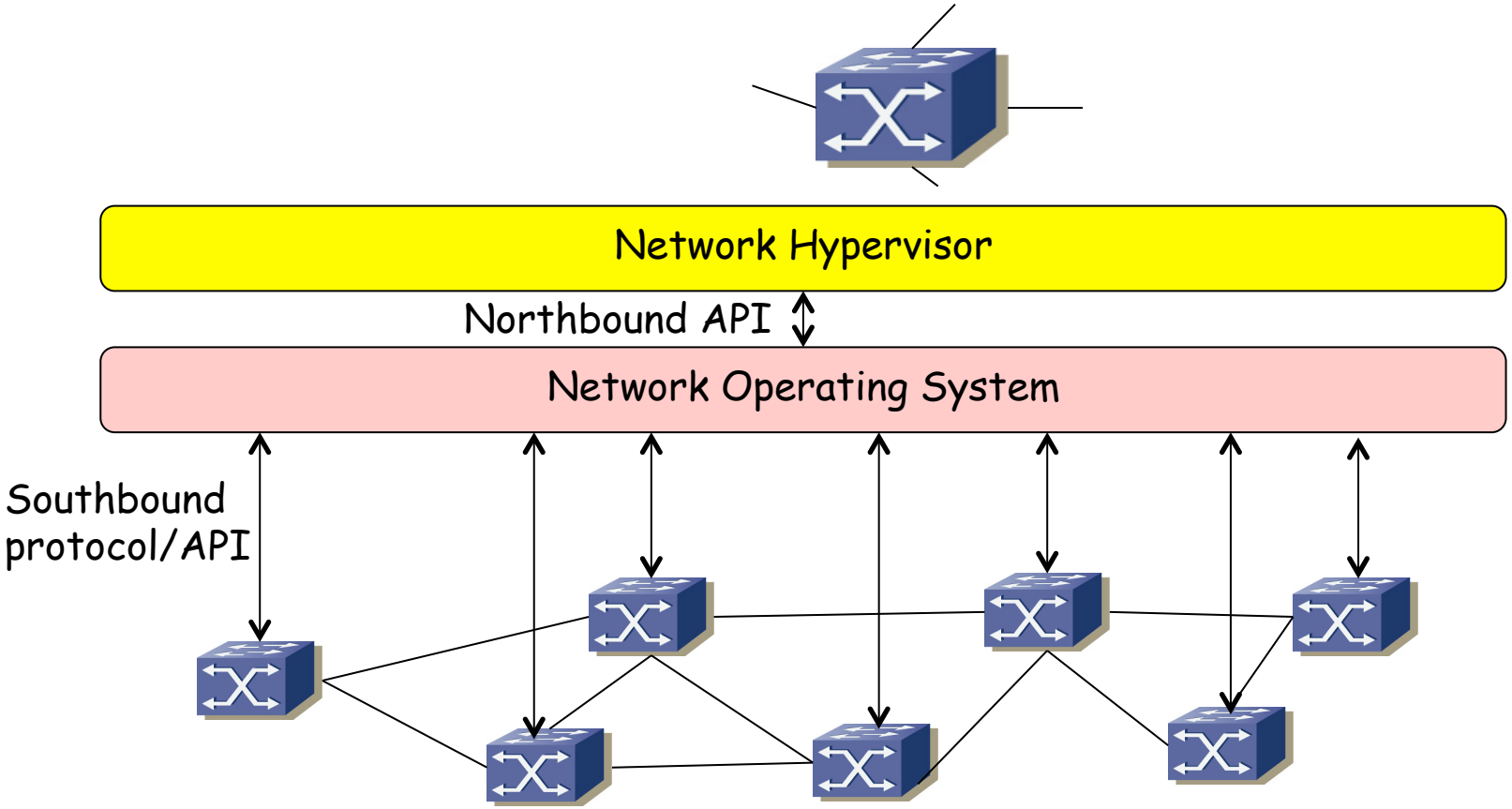


upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Redes de Nueva Generación
Área de Ingeniería Telemática

Network Virtualization

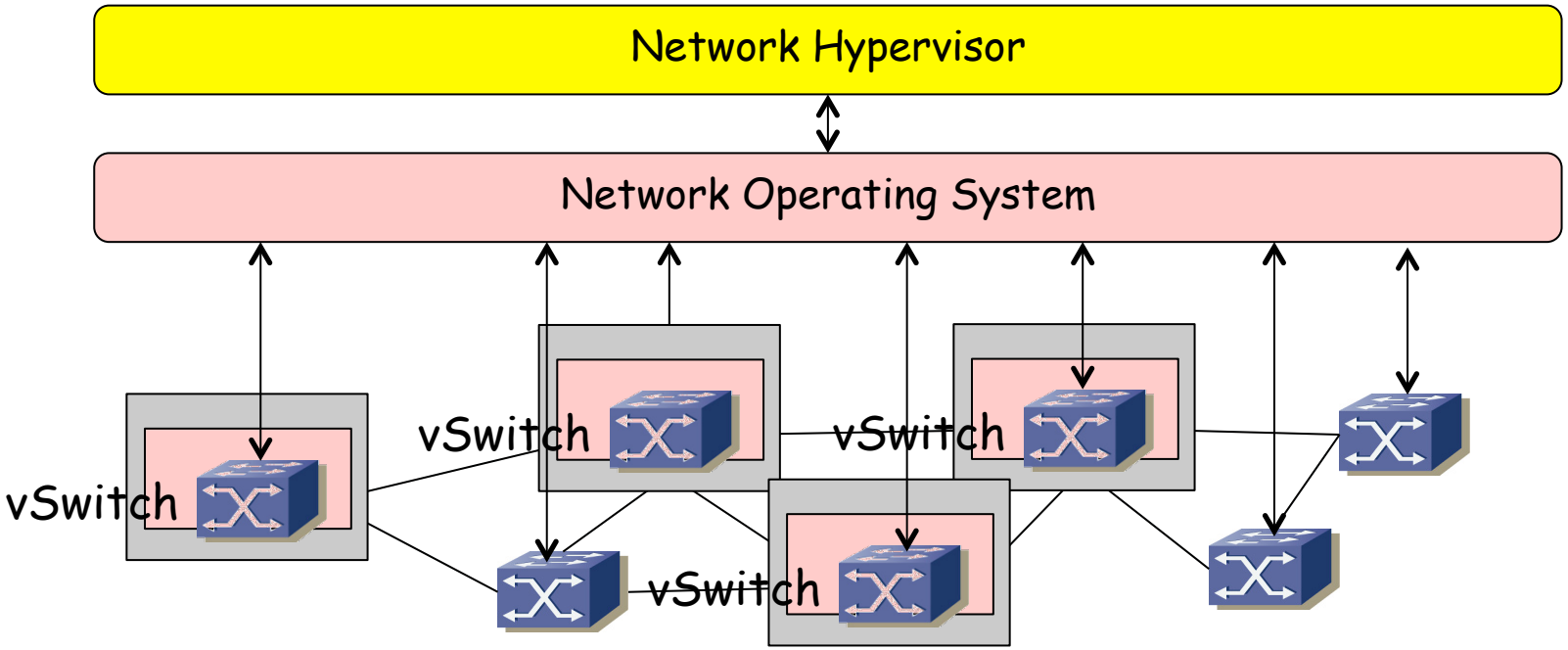
Network Virtualization

- La visión que da el NOS es virtual
- Puede ser la más adecuada para el problema que tenga que resolver el programa de control
- Sobre el NOS un *Network Hypervisor*



Network Virtualization

- ¿Y esos conmutadores?
- Ya no son simplemente conmutadores hardware, también vSwitches
- Hoy en día tenemos ya más puertos de hosts virtuales que físicos
- Un core x86 puede reenviar más de 20Mpps IPv4
- 1Mpps de 64bytes = 500Mbps; 1Mpps de 1518bytes = 12Gbps
- La frontera (edge) puede implementarse en software
- Podemos simplificar el core y volver el edge controlado por software



Ejemplo

- Middleboxes: lo más frecuente es que estén basados en arquitectura x86
- Están en general en el camino del tráfico
- Hacen mucho más que simple reenvío capa 2 ó 3
- Y pueden con ello
- Por cierto, ¿hay muchos?

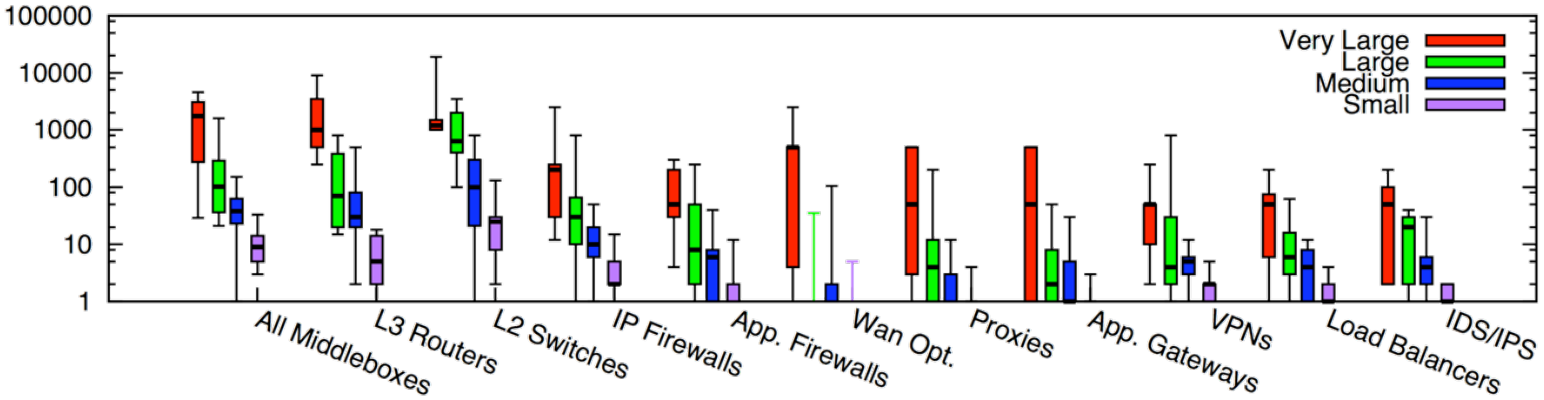


Figure 1: Box plot of middlebox deployments for small (fewer than 1k hosts), medium (1k-10k hosts), large (10k-100k hosts), and very large (more than 100k hosts) enterprise networks. Y-axis is in log scale.

J.Sherry et al., "Making Middleboxes Someone Else's Problem: Network Processing as a Cloud Service", ACM SIGCOMM 2012

OpenFlow

OpenFlow

- Su origen en proyectos de investigación en la Universidad de Stanford
- En 2011 se funda el consorcio ONF
  - *Open Networking Foundation*
  - https://www.opennetworking.org
  - Más de 140 empresas (fabricantes, operadoras, ISPs, startups, etc)
- OpenFlow es un protocolo “southbound”
- No hace “nada” sin una aplicación que lo emplee

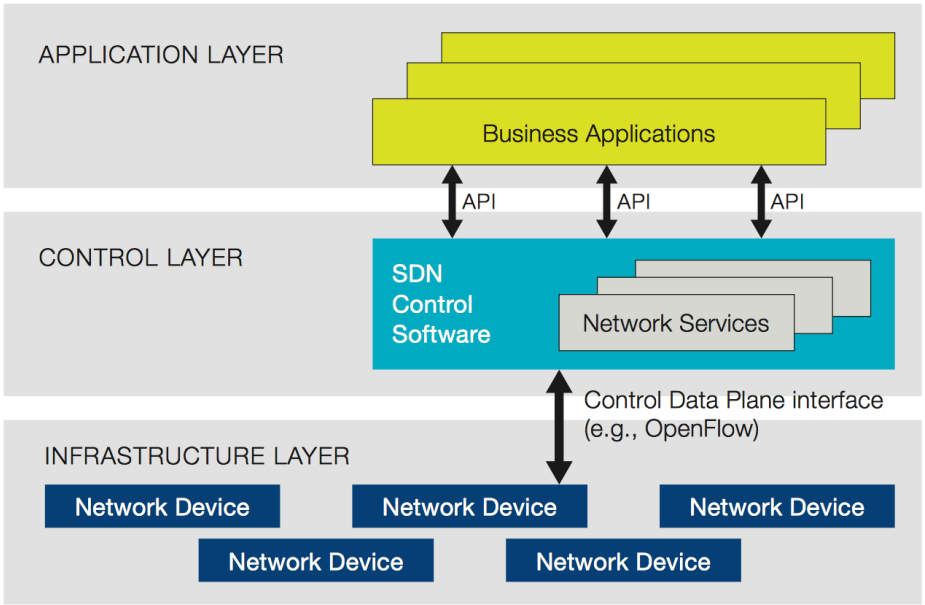
OPEN NETWORKING FOUNDATION

Martin Casado

ONF y SDN

- "The aim of SDN is to provide open interfaces that enable the development of software that can control the connectivity provided by a set of network resources and the flow of network traffic though them, along with possible inspection and modification of traffic that may be performed in the network."
- "In the SDN architecture, the control and data planes are decoupled, network intelligence and state are logically centralized, and the underlying network infrastructure is abstracted from the applications. "

OpenFlow



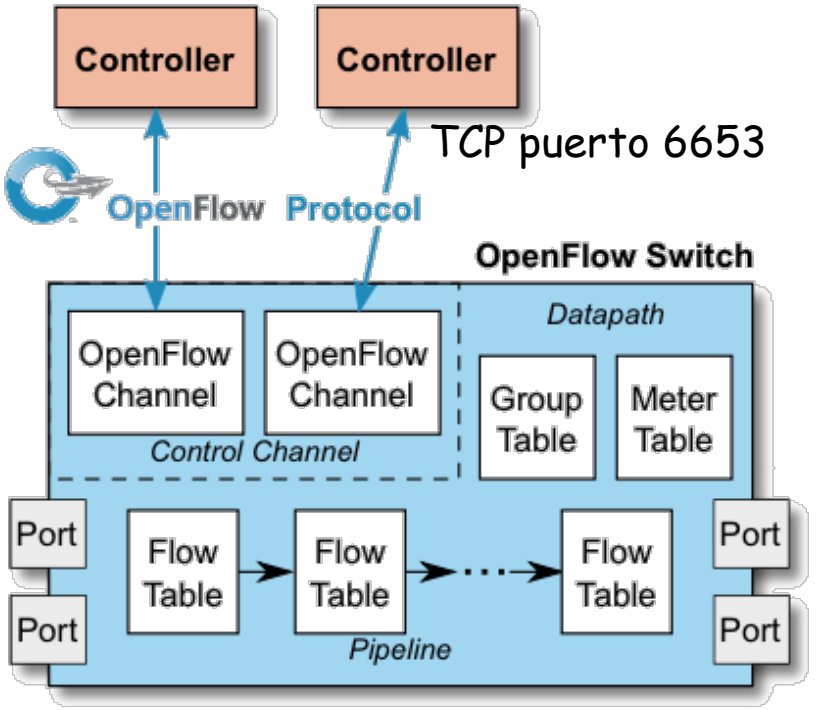
https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf

OpenFlow

- Dos tipos de conmutadores:
  - *OpenFlow-only*: solo soportan el modo de funcionamiento OpenFlow
  - *OpenFlow-hybrid*: también soportan funcionamiento "normal" (conmutación L2, conmutación L3, VLANs, ACLs, etc)
  - Los híbridos deberán tener alguna forma de clasificar si los paquetes pasan por procesado "normal" u OpenFlow

Flow Tables

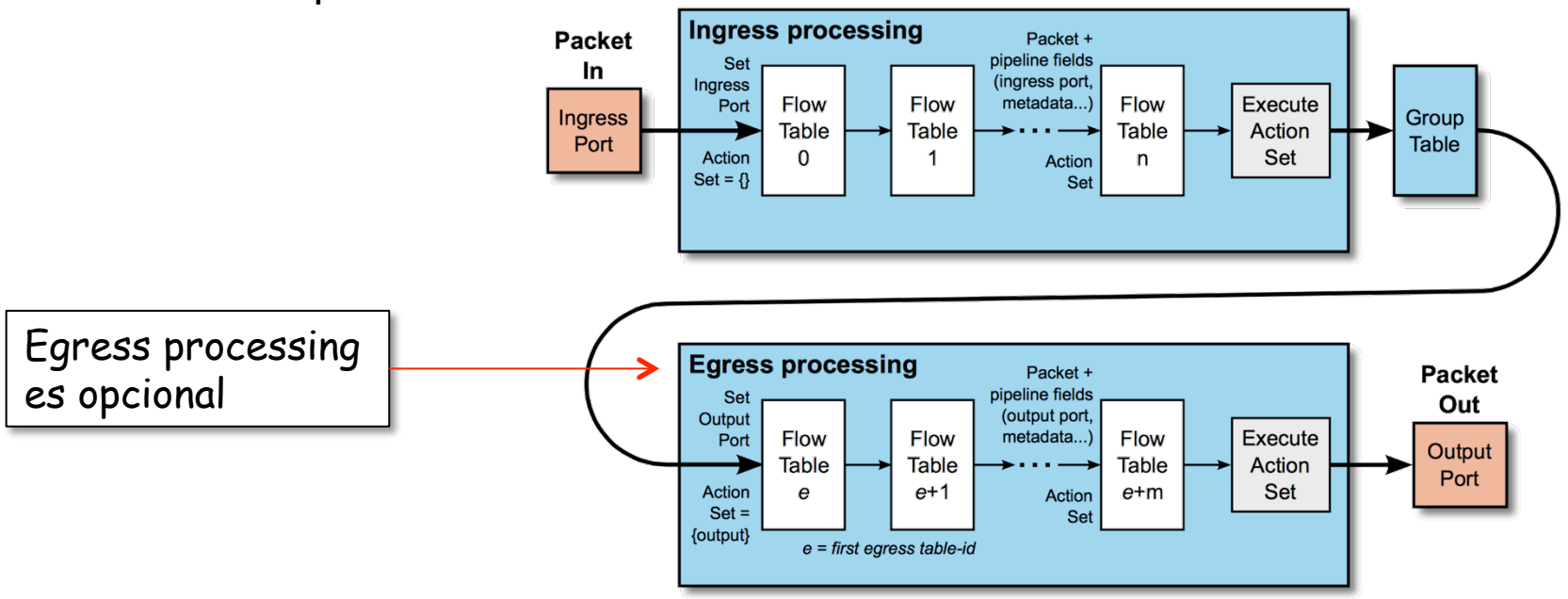
- Contienen la información sobre los campos a comprobar (*match fields*) en los paquetes y qué hacer con ellos
- El controlador puede añadir, modificar y borrar entradas empleando OF
- Las "acciones" son las operaciones en caso de que el paquete verifique la entrada en la tabla
- Puede reenviar el paquete, mandárselo al controlador, pasarlo a otra tabla, actualizar contadores, etc



https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf

OpenFlow pipeline

- Debe tener al menos una tabla aunque pueden ser más (desde 1.1, permite procesado de etiquetas MPLS)
- Hay procesado a la entrada del paquete (al menos una tabla)
- Si se decide reenviarlo pasa por tablas de salida (desde 1.5)
- Las tablas se comprueban en orden
- Si el paquete verifica una regla se ejecuta la acción que indique
- Si no verifica ninguna es un *"table miss"* y hay una acción por defecto en la tabla para este caso



https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf

Acciones

- Incluimos aquí la acción por defecto para el caso de *"table miss"*
- La acción puede ser pasar a otra tabla posterior (no anterior)
- Puede ser hacer inundación
- O reenviar por un puerto en concreto
- O puede ser reenviar el paquete al controlador (dentro de un mensaje OF)
- O pasar el paquete a un reenvío tradicional si es un conmutador híbrido
- O modificar campos de cabeceras del paquete (una modificación afecta a las comprobaciones en egress tables)
- etc

Entradas en las tablas

- *Match Fields:*
  - Puede valer ANY (comodín) o soportarse bitmasks
  - Hasta la versión 1.1 se miraban ciertos campos:
    - Puerto de entrada, metadatos provenientes de tabla anterior
    - Direcciones MAC origen y destino, Ethertype, VLAN ID, PCP
    - Etiqueta MPLS, TC
    - Direcciones IP origen y destino, protocolo, ToS
    - Puertos origen y destino TCP/UDP/SCTP
    - Tipo y código ICMP
  - Otros que se han ido añadiendo:
    - Bits ECN
    - Flags TCP
    - Código de opción de ARP, direcciones MAC e IP en el mensaje ARP
    - Direcciones IPv6, flow label IPv6, tipo y código ICMPv6
    - Etc
  - (…)

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie	Flags

Entradas en las tablas

- Prioridad:
  - Pueden verificarse varias entradas de la tabla
  - En ese caso se selecciona solo la de mayor prioridad
- (…)

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie	Flags

Entradas en las tablas

- Contadores:
  - Se actualizan cuando la entrada es seleccionada
- (…)

Counter	**Bits**	
Per Flow Table		
Reference Count (active entries)	32	*Required*
Packet Lookups	64	*Optional*
Packet Matches	64	*Optional*
Per Flow Entry		
Received Packets	64	*Optional*
Received Bytes	64	*Optional*
Duration (seconds)	32	*Required*
Duration (nanoseconds)	32	*Optional*
Per Port		
Received Packets	64	*Required*
Transmitted Packets	64	*Required*
Received Bytes	64	*Optional*
Transmitted Bytes	64	*Optional*
Receive Drops	64	*Optional*
Transmit Drops	64	*Optional*
Receive Errors	64	*Optional*
Transmit Errors	64	*Optional*
Receive Frame Alignment Errors	64	*Optional*
Receive Overrun Errors	64	*Optional*
Receive CRC Errors	64	*Optional*
Collisions	64	*Optional*
Duration (seconds)	32	*Required*
Duration (nanoseconds)	32	*Optional*
Per Queue		
Transmit Packets	64	*Required*
Transmit Bytes	64	*Optional*
Transmit Overrun Errors	64	*Optional*
Duration (seconds)	32	*Required*
Duration (nanoseconds)	32	*Optional*
Per Group		
Reference Count (flow entries)	32	*Optional*
Packet Count	64	*Optional*
Byte Count	64	*Optional*
Duration (seconds)	32	*Required*
Duration (nanoseconds)	32	*Optional*
Per Group Bucket		
Packet Count	64	*Optional*
Byte Count	64	*Optional*
Per Meter		
Flow Count	32	*Optional*
Input Packet Count	64	*Optional*
Input Byte Count	64	*Optional*
Duration (seconds)	32	*Required*
Duration (nanoseconds)	32	*Optional*
Per Meter Band		
In Band Packet Count	64	*Optional*
In Band Byte Count	64	*Optional*

Entradas en las tablas

- *Instructions:*
  - Cambio al paquete, acciones, etc, cuando se selecciona la entrada
  - Las hay de implementacion requerida y opcional
  - Ejemplos:
    - Enviar a un puerto de salida, descartar, asignar cola en el puerto out
    - Añadir/retirar etiquetas (MPLS, VLAN, PBB)
    - Modificar valor de un campo de cabecera
- (…)

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie	Flags

Entradas en las tablas

- *Timeouts:*
  - Máximo tiempo inactiva antes de expirar
- (…)

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie	Flags

Entradas en las tablas

- *Cookie*:
  - Ahí el controlador puede guardar un valor
  - El switch no lo emplea para nada
- (…)

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie	Flags

Entradas en las tablas

- *Flags*:
  - Diferentes opciones
  - Ejemplo:
    - Que envíe un mensaje al controlador al eliminarse o expirar una entrada
    - Que no lleve contadores de bytes o de paquetes

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie	Flags

El protocolo

- TCP (puerto 6653), opcionalmente empleando TLS
- Hay mensajes:
  - De controlador a conmutador
    - Petición de capacidades
    - Establecer o preguntar por configuración o estado
    - Entregarle un paquete para enviar por un puerto
  - Asíncronos (desde el conmutador) (…)
  - Simétricos

El protocolo

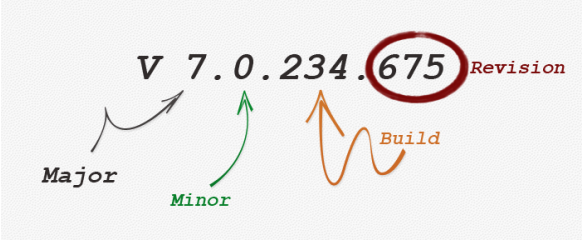
- TCP (puerto 6653), opcionalmente empleando TLS
- Hay mensajes:
  - De controlador a conmutador
  - Asíncronos (desde el conmutador)
    - Envío al controlador de un paquete recibido
    - Notificación de entrada en tabla eliminada
    - Notificación de cambio de estado de un puerto
  - Simétricos (…)

El protocolo

- TCP (puerto 6653), opcionalmente empleando TLS
- Hay mensajes:
  - De controlador a conmutador
  - Asíncronos (desde el conmutador)
  - Simétricos
    - Hello, al establecer la conexión
    - Echo, para comprobar que el otro extremo está vivo y tal vez para medir latencia o bw
    - Error

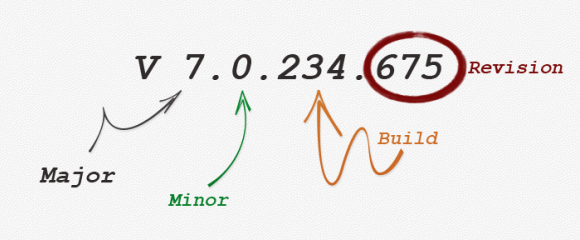
Versiones

- https://www.opennetworking.org/sdn-resources/technical-library
- Versión 1.5.0 Diciembre de 2014
- Probablemente OF 1.0 sea lo más implementado en hardware
- Las siguientes versiones han ido introduciendo mejoras, más flexibilidad, pero también haciéndolo más complejo
- OF 1.1
  - Múltiples tablas
  - Soporte de acciones para MPLS (soporta multi-etiqueta)
  - Acciones sobre el TTL
  - Soporte de VLANs en QinQ
  - Soporte para agrupar puertos de cara a acciones
- (…)



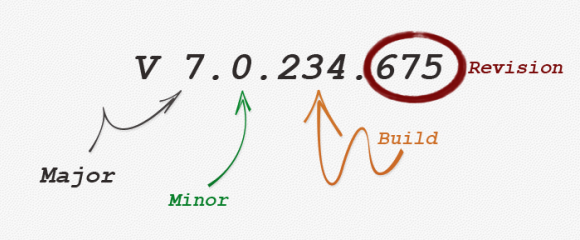
Versiones

- https://www.opennetworking.org/sdn-resources/technical-library
- Versión 1.5.0 Diciembre de 2014
- Probablemente OF 1.0 sea lo más implementado en hardware
- Las siguientes versiones han ido introduciendo mejoras, más flexibilidad, pero también haciéndolo más complejo
- OF 1.1
- OF 1.2
  - Soporte de campos de IPv6, ICMPv6, ND
  - Mejora la extensibilidad de las reglas de *match*
- (…)



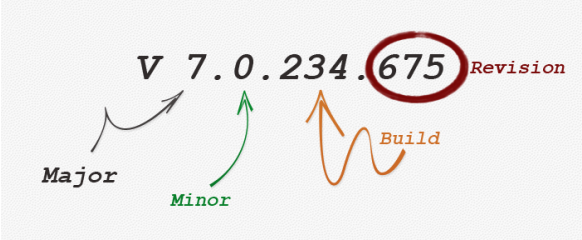
Versiones

- https://www.opennetworking.org/sdn-resources/technical-library
- Versión 1.5.0 Diciembre de 2014
- Probablemente OF 1.0 sea lo más implementado en hardware
- Las siguientes versiones han ido introduciendo mejoras, más flexibilidad, pero también haciéndolo más complejo
- OF 1.1
- OF 1.2
- OF 1.3.x
  - *Meters* por flujo (limitadores para QoS)
  - Soporte de PBB
- (…)



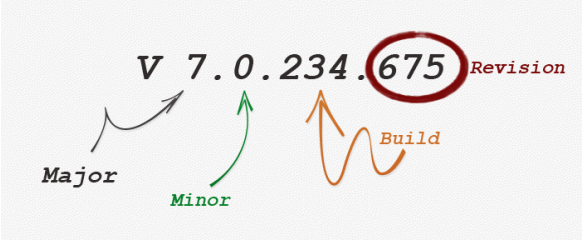
Versiones

- https://www.opennetworking.org/sdn-resources/technical-library
- Versión 1.5.0 Diciembre de 2014
- Probablemente OF 1.0 sea lo más implementado en hardware
- Las siguientes versiones han ido introduciendo mejoras, más flexibilidad, pero también haciéndolo más complejo
- OF 1.1
- OF 1.2
- OF 1.3.x
- OF 1.4
  - Mayor extensibilidad
  - Soporte de puertos ópticos (frecuencias, potencia, etc)
- (…)



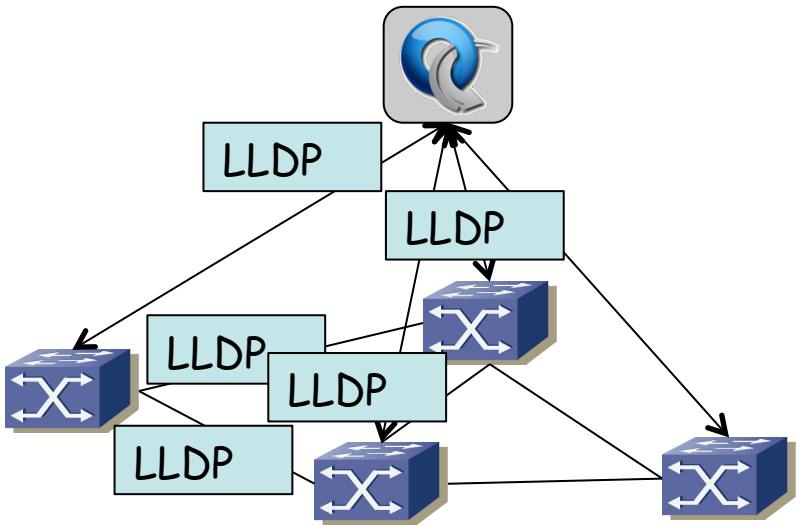
Versiones

- https://www.opennetworking.org/sdn-resources/technical-library
- Versión 1.5.0 Diciembre de 2014
- Probablemente OF 1.0 sea lo más implementado en hardware
- Las siguientes versiones han ido introduciendo mejoras, más flexibilidad, pero también haciéndolo más complejo
- OF 1.1
- OF 1.2
- OF 1.3.x
- OF 1.4
- OF 1.5
  - *Egress tables*
  - Soporte para más que Ethernet
  - Flags TCP



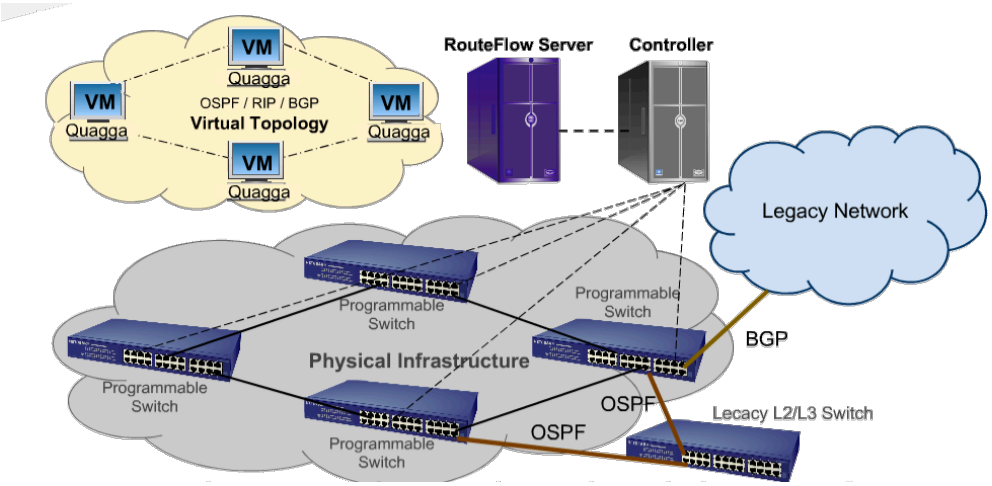
Ejemplo

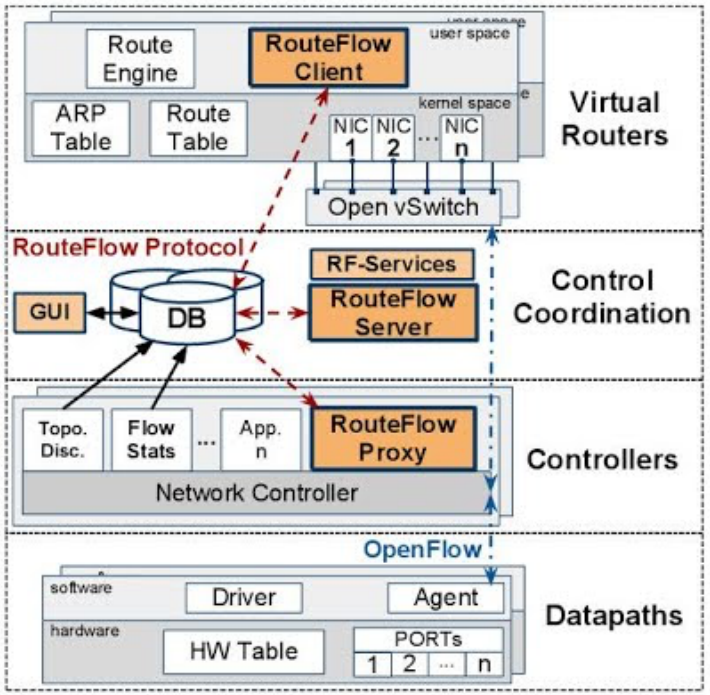
- Descubrimiento de la topología
- El controlador programa en los switches una regla para que le envíen las tramas LLDP
- Envía por el canal de control tramas LLDP para que ellos envíen por sus puertos (. . .)
- Las recibirá del resto de switches adyacentes
- Con eso puede descubrir la topología



Ejemplo

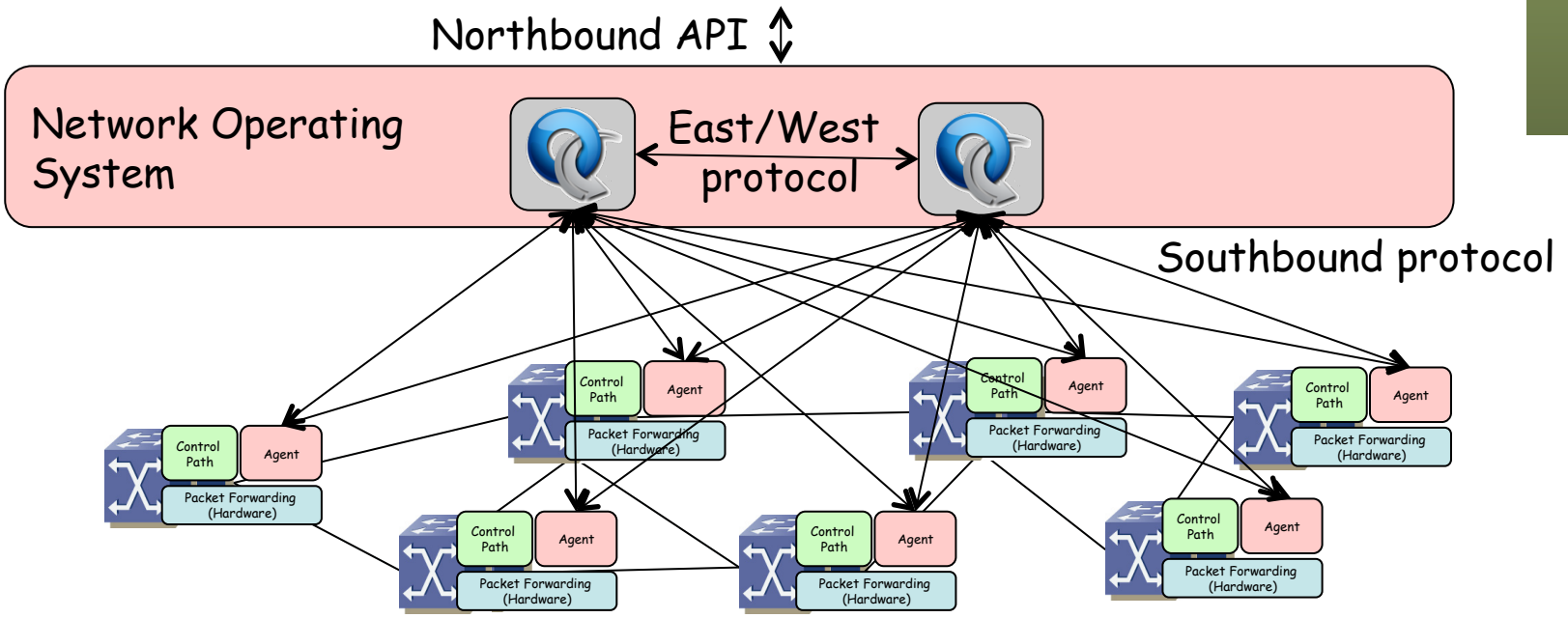
- RouteFlow
- https://sites.google.com/site/routeflow/
- Routing IP virtualizado en controlador (open source)
- Genera la FIB que se traslada mediante OpenFlow al hardware
- Es decir, mantiene el funcionamiento del routing pero ejecutado en elementos centralizados





APIs

- OpenFlow es un *Southbound API*
- El ONF asocia OpenFlow a SDN pero una SDN no necesita emplear necesariamente OpenFlow
- Podríamos considerar OF a día de hoy el API south estándar
- No hay *Northbound API* estandarizada, ni *de facto*
- No hay *East/West API* estandarizada

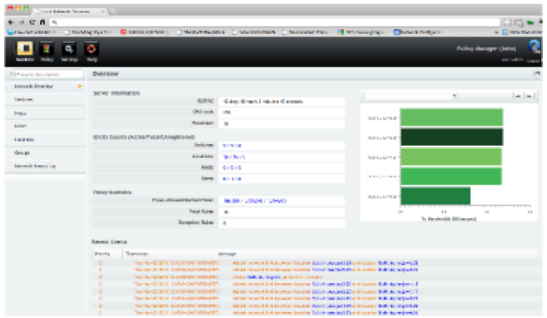


Controladores

- NOX
  - http://www.noxrepo.org
  - Desarrollado por Nicira, cedido el código en 2008
  - Ofrece un API C++ para OF 1.0
  - Muchos otros heredan de su código
  - Incluye componentes de ejemplo para descubrir la topología, implementar un puente transparente y un switch distribuido
  - Open Source
- POX
  - Hereda de NOX
  - Permite el desarrollo en Python
  - Open Source
- Beacon
  - https://openflow.stanford.edu/display/Beacon/Home
  - Java (desarrollo con eclipse)
  - Open Source

Controladores

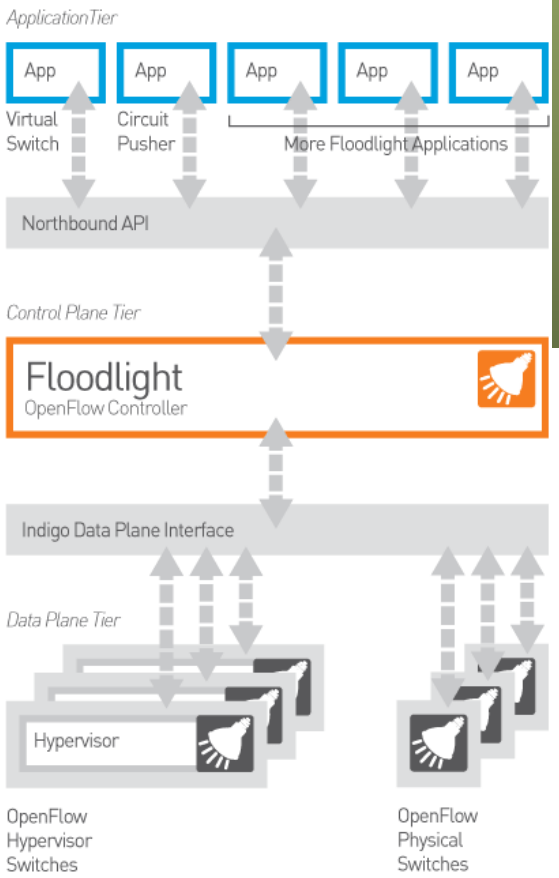
- SNAC
  - http://www.openflowhub.org/display/Snac/SNAC+Home
  - Incluye GUI web
  - Incluye un lenguaje de definición de políticas
  - Open Source



- FloodLight
  - http://www.projectfloodlight.org/floodlight/
  - Basado en Java (basado en Beacon)
  - Apoyado por Big Switch Networks
  - Lo emplean para construir su controlador
  - Open Source

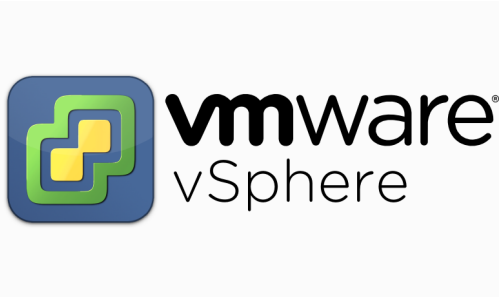
Guido Appenzeller

big switch networks



VMware

- Controlador propietario
- vCenter Server controla los VDS (Virtual Distributed Switches)
- Otros componentes: vSphere, vCloud Director, vCloud Networking and Security, vCloud Automation Center, vCenter Site Recovery Manager, vCenter Operations Management Suite, vFabric Application Director for Provisioning
- Máximos vSphere 6.0:
  - 1024 VMs por host
  - 10 vNICs por VM
  - 1000 hosts por VDS
  - 1016 puertos de VDS activos por host
  - 60.000 puertos por VDS
  - 1000 hosts, 10.000 VMs en funcionamiento y 128 VDS por vCenter
  - 65.536 direcciones MAC por vCenter
  - 4/8 operaciones vMotion simultáneas por host por NIC 1/10Gbps
  - 16 VDS por host
  - etc



http://www.vmware.com/pdf/vsphere6/r60/vsphere-60-configuration-maximums.pdf

Nicira

- Fundada en 2007
- Miembro fundador del ONF
- En 2011 empieza a distribuir su NVP (*Network Virtualization Platform*)
- Es un controlador para OVS (Open vSwitch)
- No emplea solo OF sino OVSDB (Open vSwitch DataBase Management Protocol)
- Adquirida en 2013 por VMware (por unos 1260 millones de $)

Martin Casado

OvS
Open vSwitch

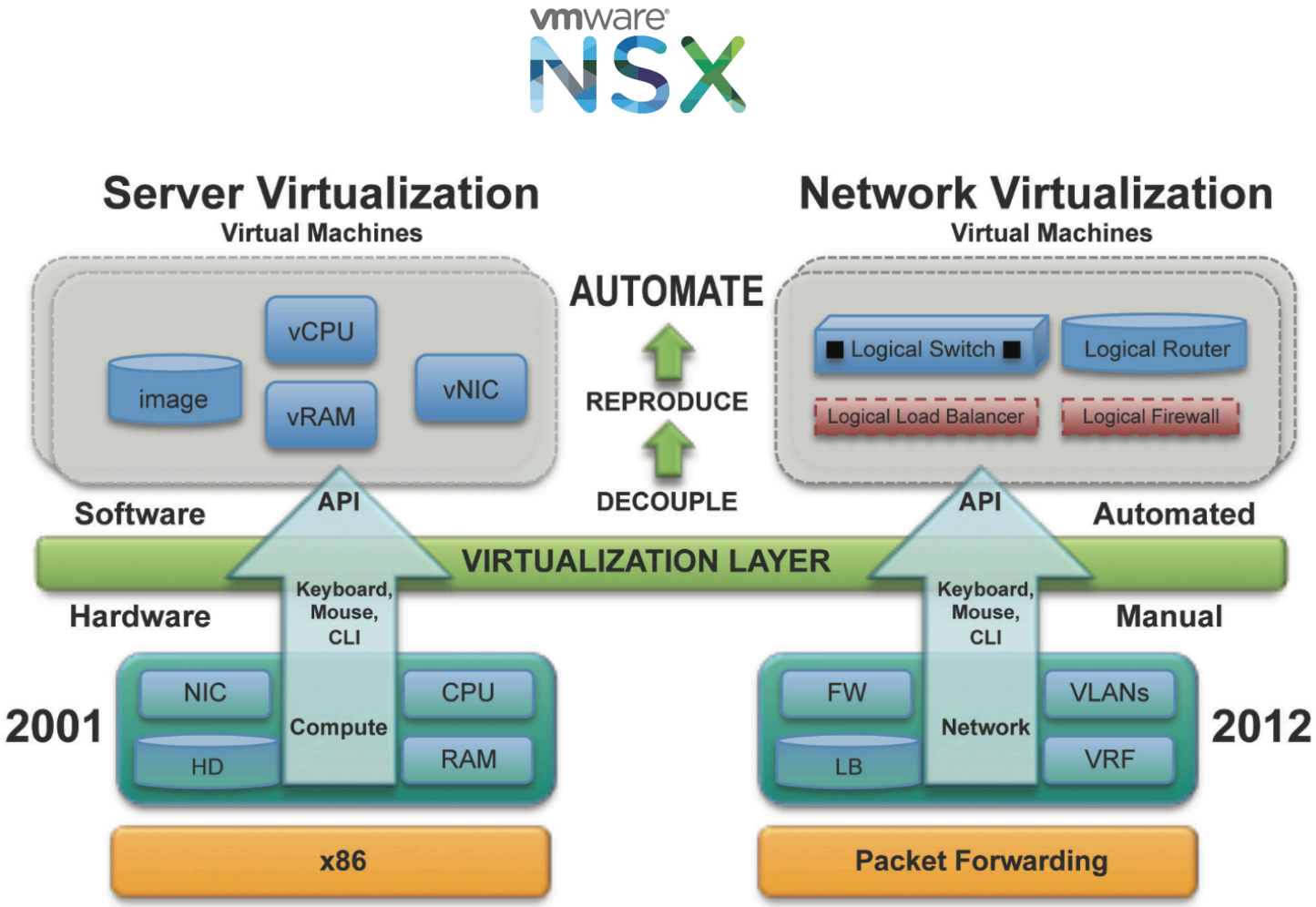
Otro software

- Frameworks
  - Onix, Trema, Maestro, Ryu
  - Indigo (para añadir OF a switches)
- FlowVisor:
  - https://github.com/OPENNETWORKINGLAB/flowvisor/wiki
  - Actúa como un proxy entre los switches y los controladores OF
  - Permite repartir recursos de la red entre varios controladores
- Avior, Oflops, Cbench, Twister, FortNOX, LINC, Pantou, Of13softwitch, Cisco OnePK, Plexxi, etc etc etc

- ¡Se abrió la veda al software!

Ejemplo: NSX

http://www.vmware.com/products/nsx



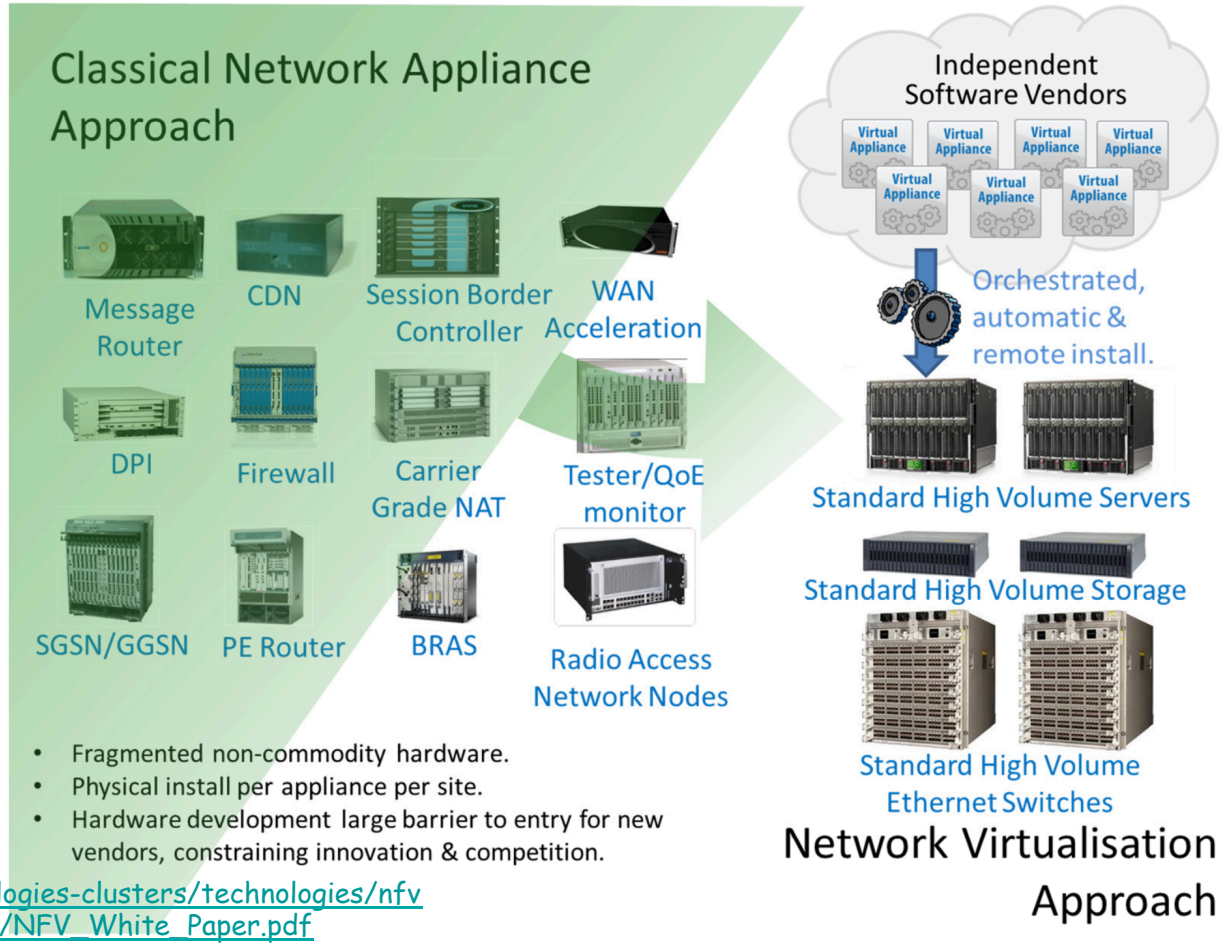
NFV

El problema

- Problema de las operadoras
- Gran cantidad de *appliances*
- Desplegar un nuevo servicio requiere espacio y alimentación para ese nuevo hardware
- Nuevas habilidades de la gente para diseñar, integrar y operar el servicio con ese nuevo hardware
- Ese hardware alcanza su límite de vida con rapidez, lo cual requiere políticas de remplazo que no crean nuevo beneficio
- Los operadores declaran no estar incrementando sus beneficios pero aumentan sus costes (más tráfico, más servicios)

NFV

- *Network Functions Virtualisation* (complementario a SDN)
- Se busca mover de hardware dedicado a máquinas virtuales
- Un ISG (*Industry Specification Group*) de ETSI desde finales de 2012
- Hoy más de 200 compañías



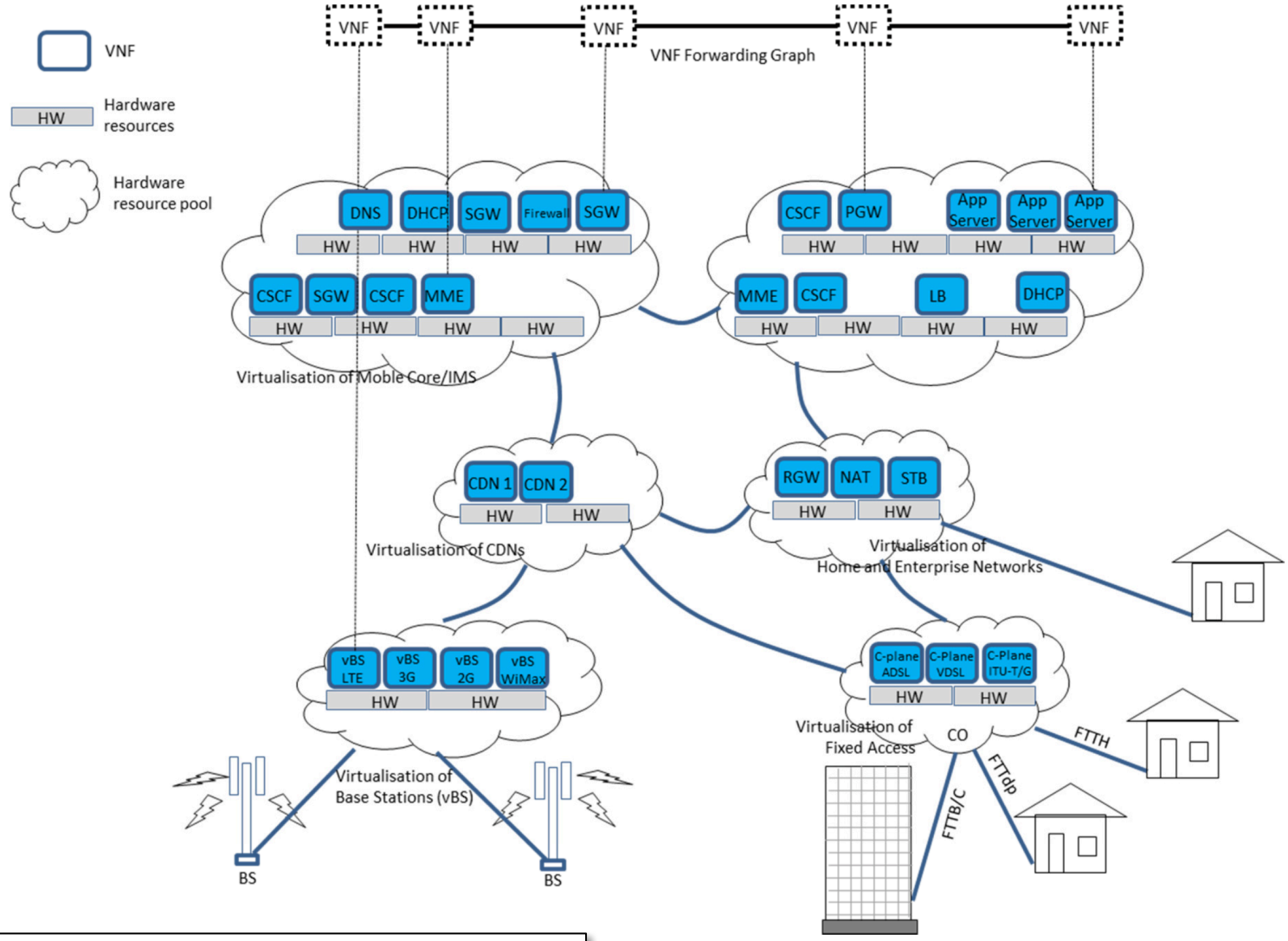
http://www.etsi.org/technologies-clusters/technologies/nfv
https://portal.etsi.org/NFV/NFV_White_Paper.pdf

Use cases

- Switching elements: BNG, CG-NAT, routers.
- Mobile network nodes: HLR/HSS, MME, SGSN, GGSN/PDN-GW, RNC, Node B, eNode B.
- Functions contained in home routers and set top boxes to create virtualised home environments.
- Tunnelling gateway elements: IPSec/SSL VPN gateways.
- Traffic analysis: DPI, QoE measurement.
- Service Assurance, SLA monitoring, Test and Diagnostics.
- NGN signalling: SBCs, IMS.
- Converged and network-wide functions: AAA servers, policy control and charging platforms.
- Application-level optimisation: CDNs, Cache Servers, Load Balancers, Application Accelerators.
- Security functions: Firewalls, virus scanners, intrusion detection systems, spam protection.

https://portal.etsi.org/NFV/NFV_White_Paper.pdf

Ejemplos



VNF = Virtualised Network Function

https://portal.etsi.org/NFV/NFV_White_Paper2.pdf

Algunos beneficios

- Reducción de coste de equipos
- Reducción de consumo eléctrico
- Reducción de tiempo de despliegue de un nuevo servicio
- Posibilidad de tener servicios en producción, prueba y desarrollo en la misma infraestructura
- Escalado rápido del servicio
- Abre el mercado a desarrolladores de soft (no necesitan desarrollar hardware)
- Multi-tenancy
- Mejores habilidades existentes para la gestión de infraestructura IT de gran escala que de equipos de red
- Reducción de tiempos de reparación
- Reducción de tiempos de actualización de software
- Etc etc

Facilitadores

- *Cloud Computing*
  - Virtualización (hypervisores, vSwitch, smart NICs)
  - *Orchestration*
  - Open APIs

- Grandes volúmenes de servidores
  - Componentes estándar (por ejemplo x86), vendidos por millones (escala) e intercambiables (competencia)
  - En lugar de *appliances* que dependen de ASICs

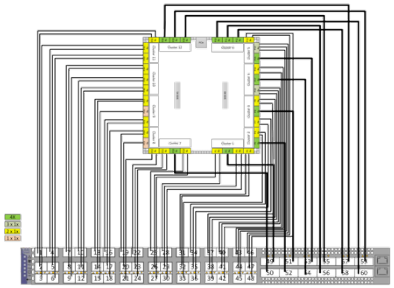
Networking hardware y el software

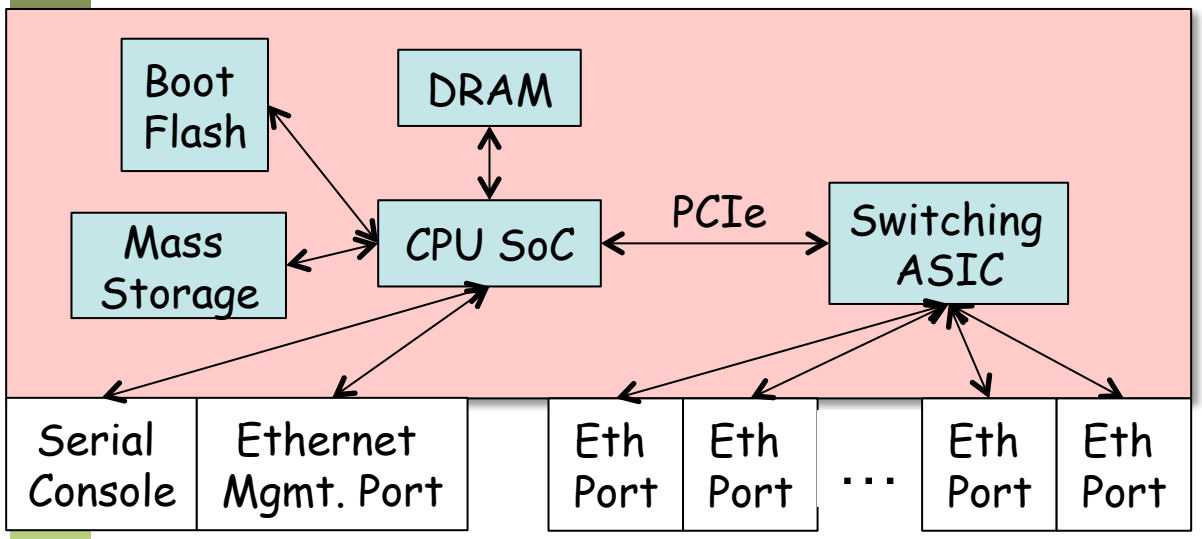
¿Evolución?

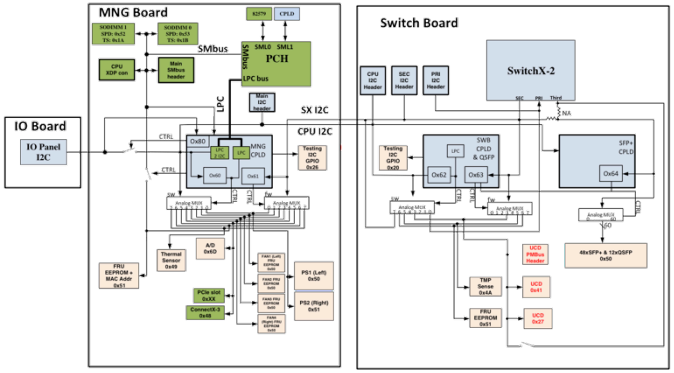
- La infraestructura se está simplificando
- Principalmente el hardware, controlable por software
- *White boxes* no solo servidores sino también switches
- También se venden ya switches *"Bare metal"* = solo el hardware

Bare metal switches

- Menores costes
- El mismo equipo un día es un switch, otro un firewall, otro un balanceador… dentro de las limitaciones del ASIC
- Ejemplo:
  - Open Compute Networking Project
  - http://www.opencompute.org/wiki/Networking
  - Especificaciones completas de conmutadores
- Fabricantes: Mellanox, Quanta, Penguin Computing, Edge-core, Acton, Dell, etc







¿Evolución?

- Para estos equipos sistemas operativos y gran cantidad de software, generalmente basados en linux, muchos de código abierto
- Ejemplos:
  - Open Network Install Environment (ONIE): http://onie.opencompute.org
  - Open Network Linux: http://opennetlinux.org
  - Big Switch's Switch Light OS
  - Pica8 PicOS
  - Cumulus Linux
- Es decir, igual que en el entorno de servidor, puedes cambiar el hardware, instalar el sistema operativo que quieras y desarrollar tus aplicaciones (…)

¿Evolución?

- Para diferenciarse, los proveedores desarrollan software propietario para ofrecer sus servicios
- Porque hoy en día ya es el software por lo que principalmente están cobrando los fabricantes “no-open”
- Muchos modelos ToR de fabricantes conocidos son switches bare-metal que han comprado, cambiado el software y el frontal

Software Defined X

- Software Defined Networking (SDN)
- Software Defined Infrastructure (SDI)
- Software Defined Data Center (SDDC)
- Software Defined Storage (SDS)
- etc