Cloud y SDN

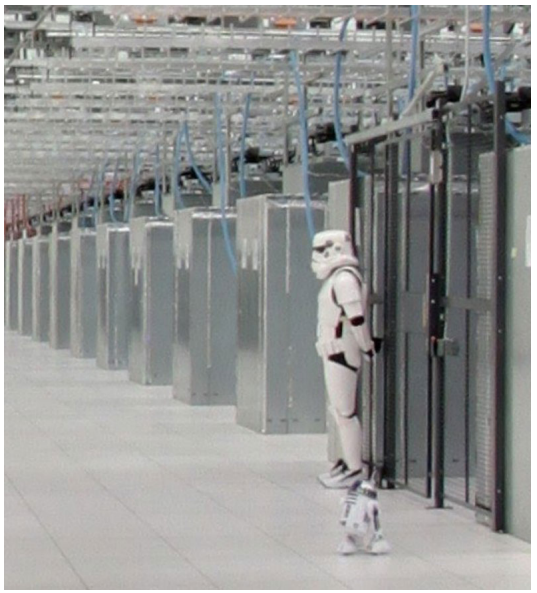
HPC

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
- (…)



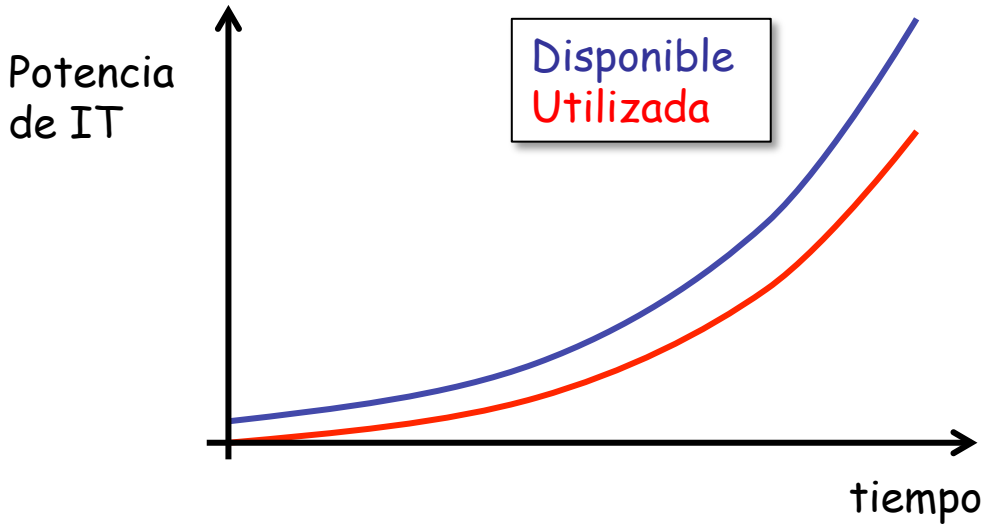
HPC Clusters

- 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”*

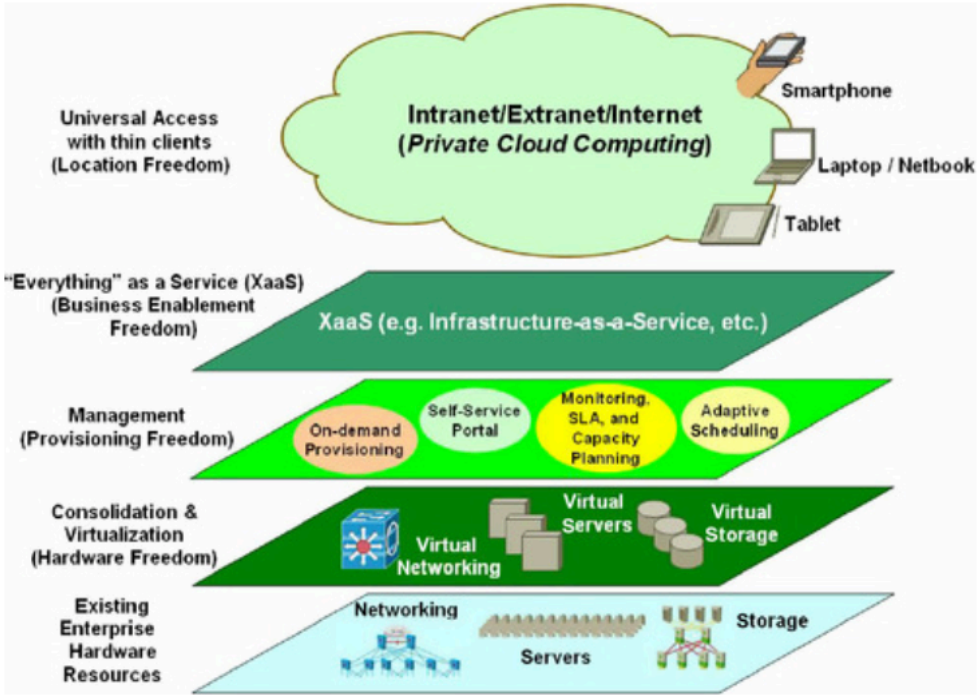


Virtualización en la red

- 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 “orquestación” 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

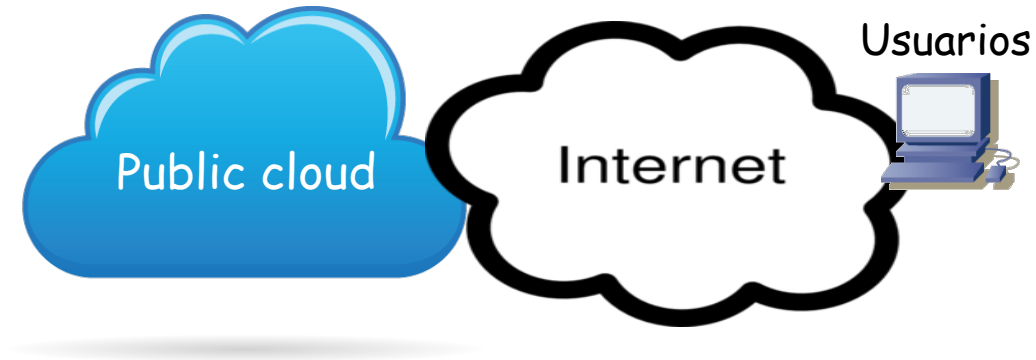


Cloud Computing

- *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*

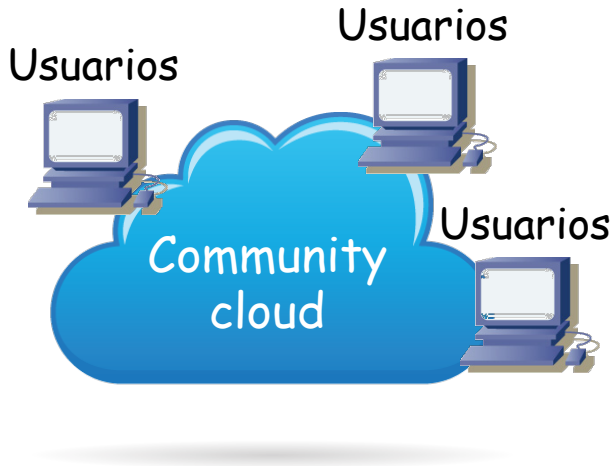
Public cloud

- 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 organizaciones 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

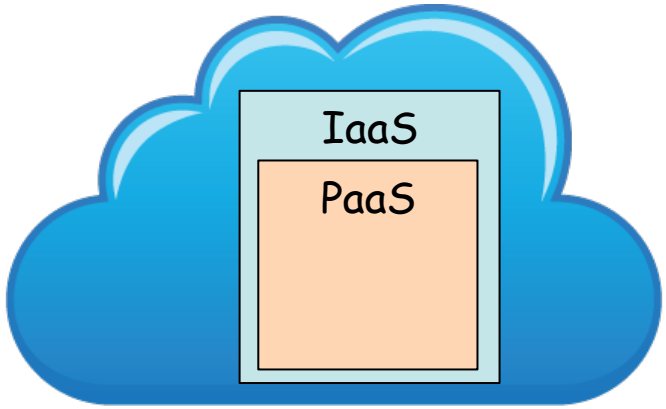
- Utilización de infraestructura de al menos dos de los tipos anteriores para las mismas aplicaciones
- Por ejemplo:
  - Una empresa tiene su *private cloud* con recursos limitados
  - Cuando alcanza los límites, las peticiones en exceso se redirigen a una *public cloud*

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, Google Compute Engine

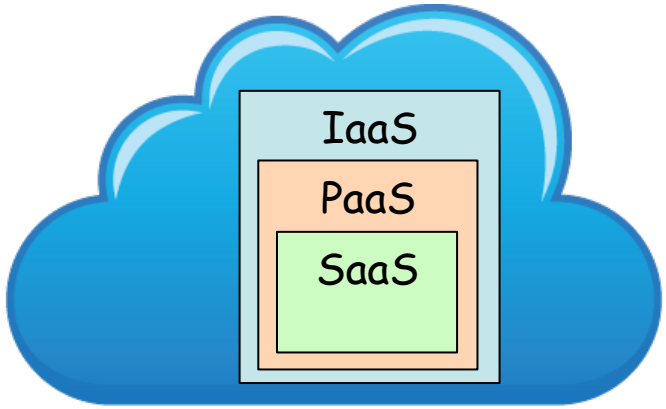
PaaS

- 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 (<10 años)
- Aunque se basa en ideas que tienen bastante 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

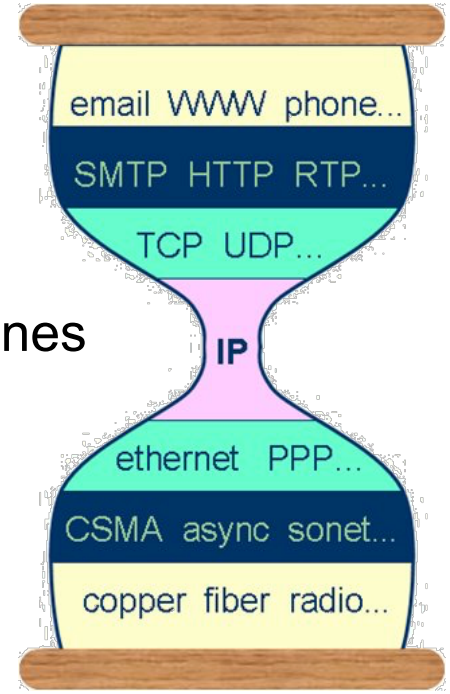
Scott Shenker
ICSI Berkeley

Martin Casado
Stanford, Nicira, VMware

Nick McKeown
Stanford

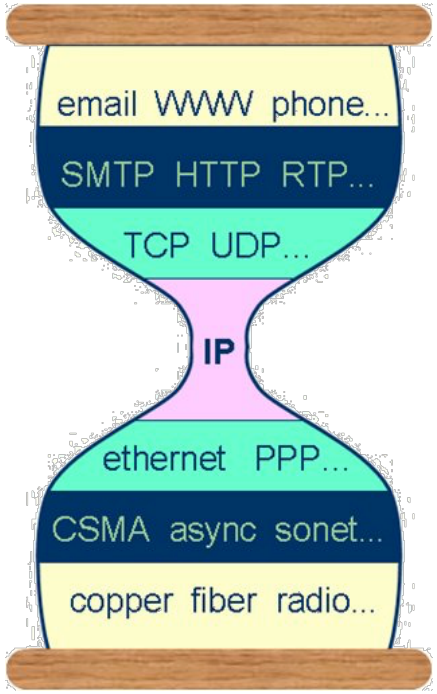
Internet: Por qué

- ¿Por qué ha tenido éxito Internet?
  - Porque ha permitido que surjan gran cantidad de nuevos servicios (innovación)
- ¿Cómo lo hemos hecho?
  - Desarrolladores han construido 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”



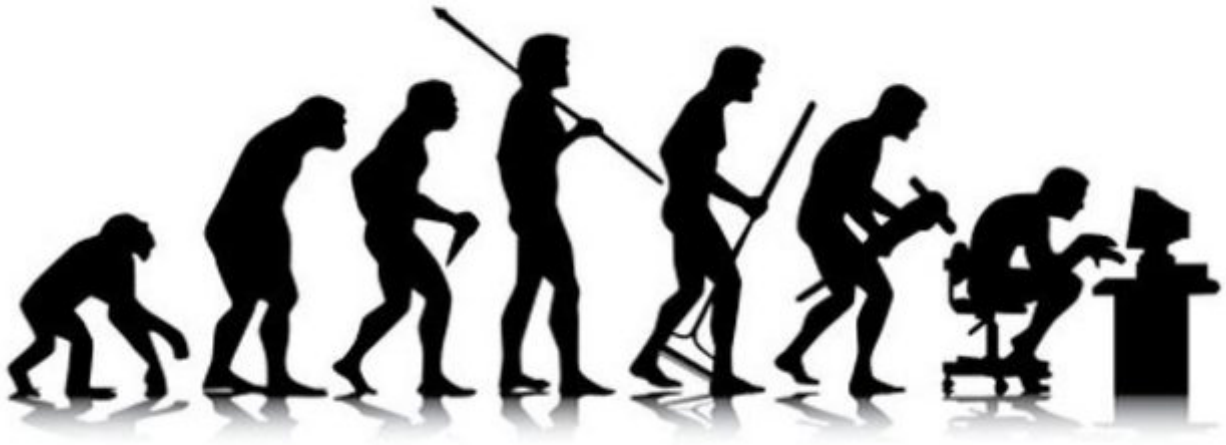
Layers: Por qué

- 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



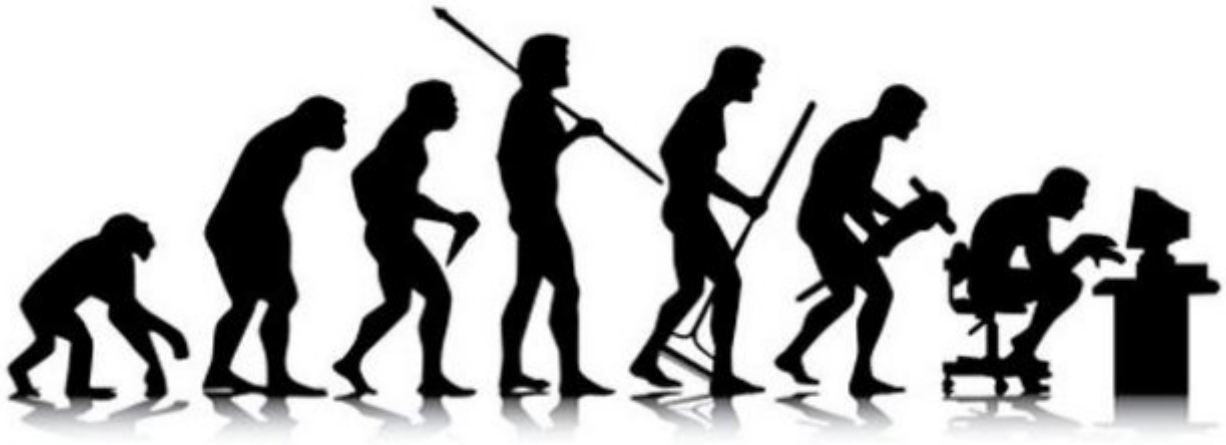
Evolución

- 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 (…)



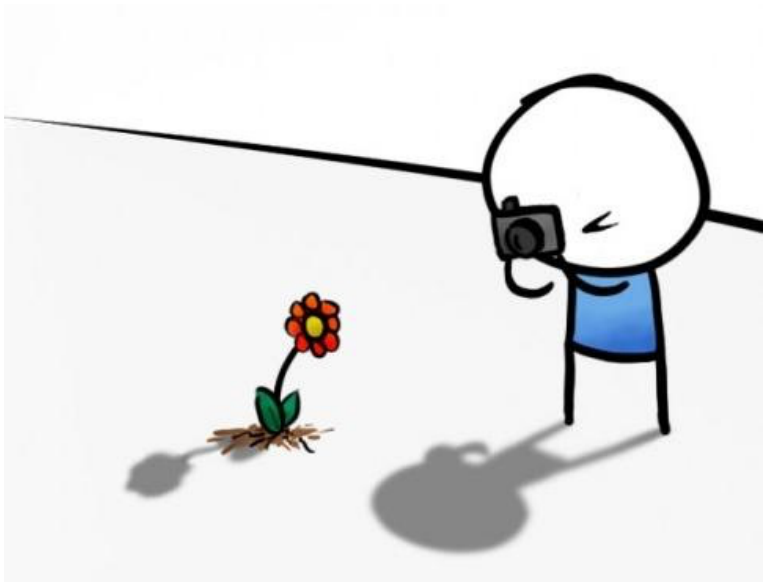
Evolución

- 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

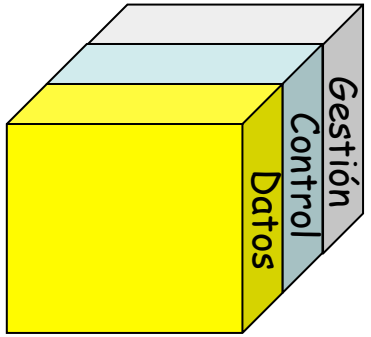
- Las redes eran simples
- Ethernet, IP, son tecnologías muy simples
- Pero les hemos añadido nuevas necesidades
- En la parte del control de las mismas
- Y se nos han descontrolado (…)



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



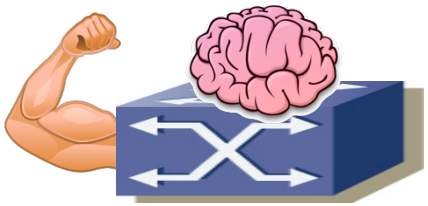
- *Data plane*
  - Paquetes de usuarios
  - Reenvío, fragmentación, replicación para multicast, etc
  - Simple y rápido (*fast path*)
- *Control plane*
  - Actividades necesarias para que funcione el plano de datos
  - Crear tablas de rutas, configurar políticas, anunciar servicios, etc
  - Complejo 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

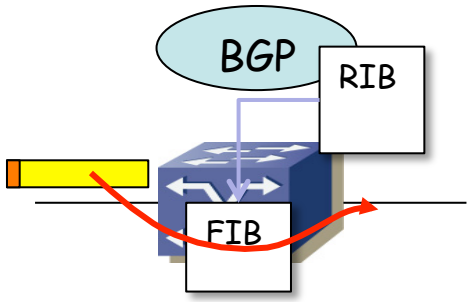
Complejidad y abstracciones

- 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

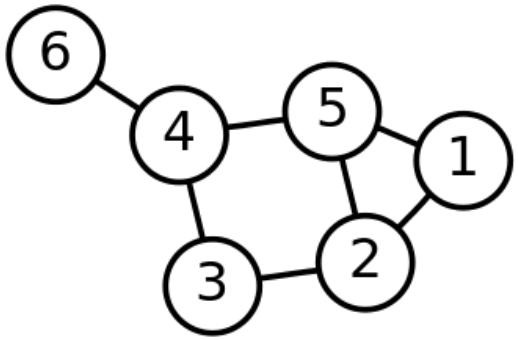
- ¿Qué hacemos en el plano de control?
- Calcular la configuración de los dispositivos
  - Tablas de rutas
  - Listas de filtrado
  - Etc
- Tenemos que desplegar esta configuración a través de una red sin garantías
- Tenemos que desplegarla empleando el protocolo que tenemos

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)

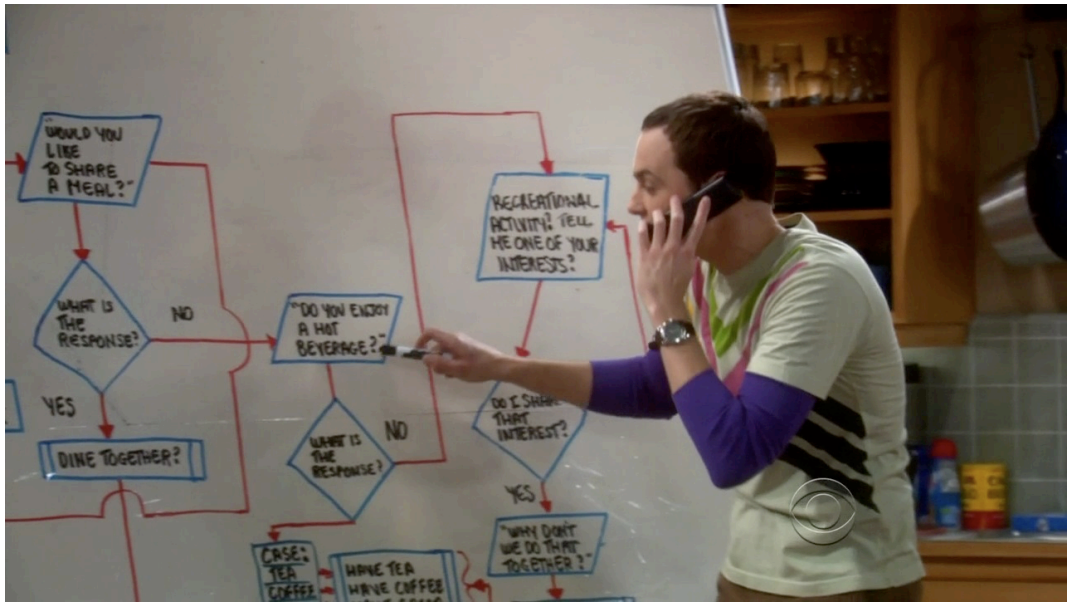
Abstracciones

- Estado distribuido
  - Tenemos que distribuir el estado (la configuración)
  - Esto ya está resuelto, no queremos volver a resolverlo
  - Para un problema queremos poder suponer que lo resolvemos de forma centralizada
  - No preocuparnos de cómo luego se distribuya



Abstracciones

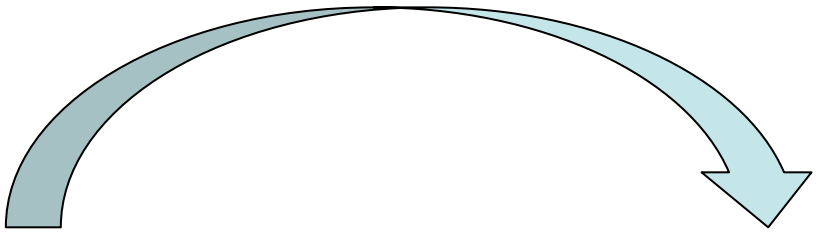
- Especificación
  - El programa de control lo que quiere es especificar algo
  - Su problema no debería ser implementarlo
  - Necesitamos una abstracción de la red (virtualización)



Abstracciones

- Forwarding
  - Modelo para el comportamiento del plano de datos
  - Independiente del equipo, fabricante, etc

- Ejemplo:
  - De unas 240 páginas que tiene la RFC de OSPFv2 solo unas 20 son sobre el cálculo del camino (Dijkstra)
  - El resto es principalmente la distribución del estado



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.