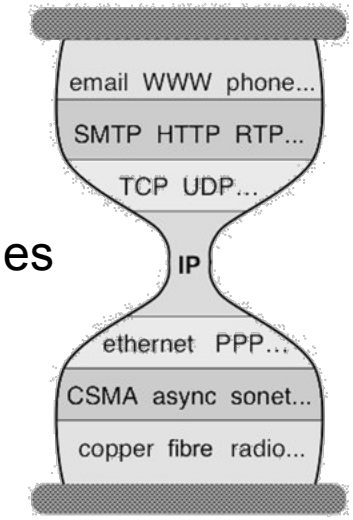
# Principios

Area de Ingeniería Telemática
http://www.tlm.unavarra.es

Máster en Ingeniería de Telecomunicación

# End-to-End principle

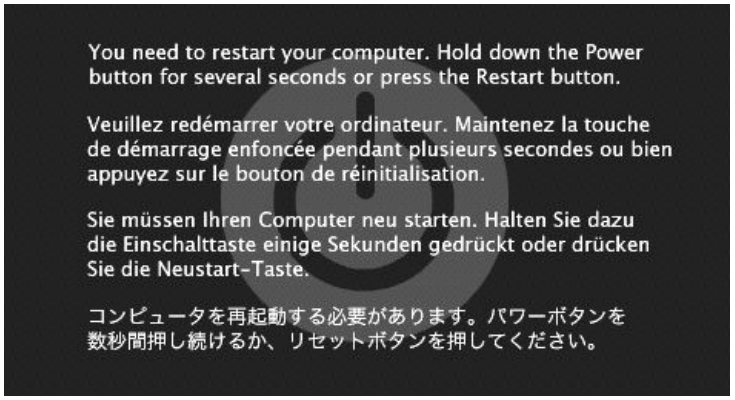
- [RFC3439] *"end-to-end protocol design should not rely on the maintenance of state (i.e., information about the state of the end-to-end communication) inside the network. Such state should be maintained only in the end points, in such a way that the state can only be destroyed when the end point itself breaks."*
- No quiere decir que no haya estado en la red, que lo hay (por ejemplo las tablas de rutas)
- Quiere decir que no interactúa directamente con los protocolos en los hosts
- Esto busca hacer simple la red
- La complejidad impide que escale
- La complejidad lleva a mayor CAPEX y OPEX
- Pero no penséis que los routers de hoy en día son simples




RFC 3435, "Some Internet Architectural Guidelines and Philosophy", R. Bush y D. Meyer, 2002

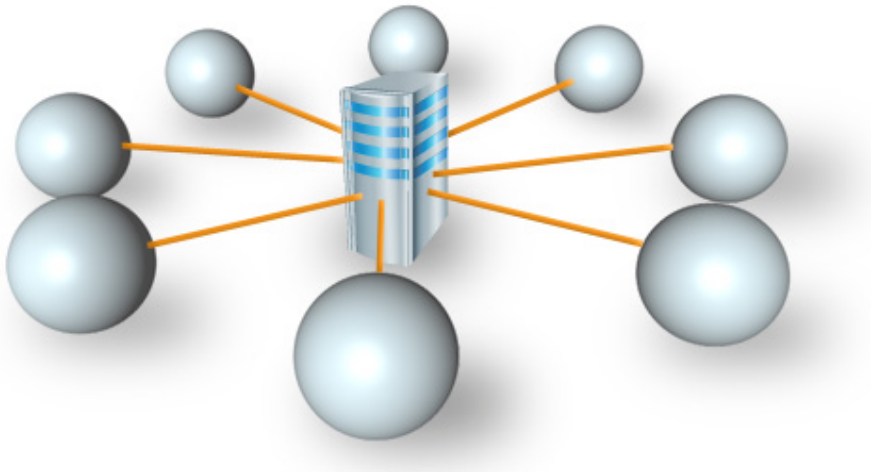
# Internet architecture

- De hecho, ¿podríamos decir que se diseñó la Internet “simple” para permitir esta escalabilidad?
- ¿Un protocolo que luego ha agotado sus direcciones parece creíble que se diseñara para gran escala?
- Bueno, era una época en la que quién iba a pensar que casi todos tendríamos (una o dos) computadoras en nuestros bolsillos
- ¿O tal vez lo hicieron simple para hacerlo rápidamente… porque se les colgaban los equipos?



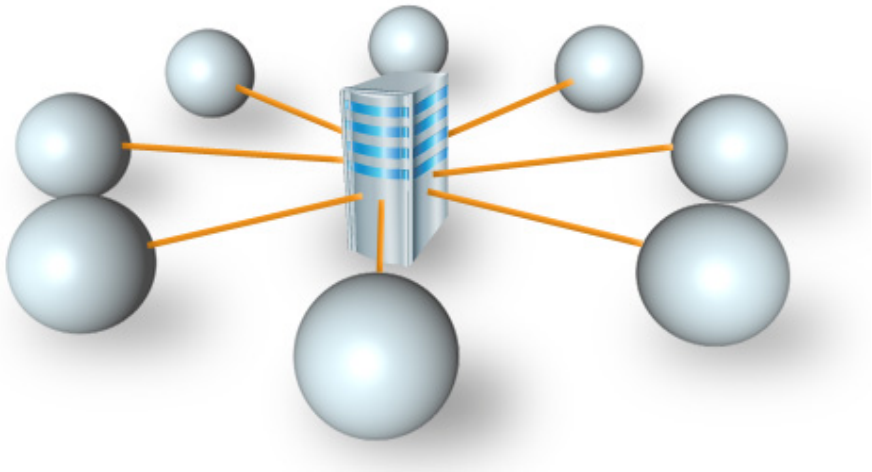

# Internet architecture

- El objetivo que tenían era que si los hosts de los extremos no se colgaban debían poderse comunicar si los routers funcionaban
- Es decir, no depender de otros servidores que controlaran la red, como sucedía en la red telefónica
- El problema es que no se sabía cómo construir algoritmos distribuidos que permitieran recuperar la red ante fallos en enlaces
- Los algoritmos distribuidos no son sencillos
- En los últimos años hemos aprendido bastante sobre ellos
- Muchas veces “por las malas”



# Internet architecture

- ¿Una arquitectura con elementos de control centralizados es mala?
- Es más parecido al control en la red telefónica
- Tienes puntos críticos de fallo…
- …pero puedes tener replicada su funcionalidad
- También son cuellos de botella…
- …pero puedes repartir la carga entre varios
- Va en contra de los principios de Internet…
- …pero es el fundamento de las SDNs (Software Defined Networks)



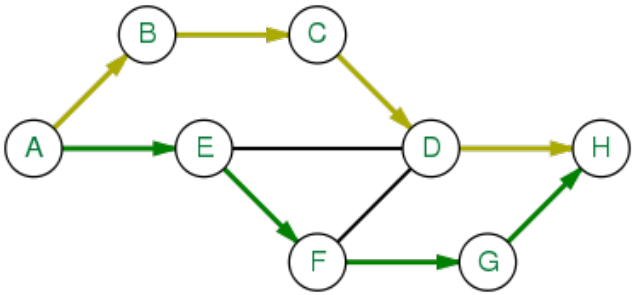
OpenFlow

# Internet architecture

- ¿Una arquitectura con elementos de control centralizados es mala?
- Es más parecido al control en la red telefónica
- ¿Es tan mala la red telefónica?
  - La voz funciona bastante bien
  - Lleva décadas dando QoS extremo a extremo
  - Sin necesitar gran sobredimensionamiento
  - Ni la complejidad de los mecanismos de QoS en la red de conmutación de paquetes
  - ¿Quién se dejaría operar remotamente donde el médico controla el robot a través de la Internet?
  - Pensemos en un parámetro básico: mantener la conectividad (no hablemos de throughput o delay) (…)

# Internet architecture

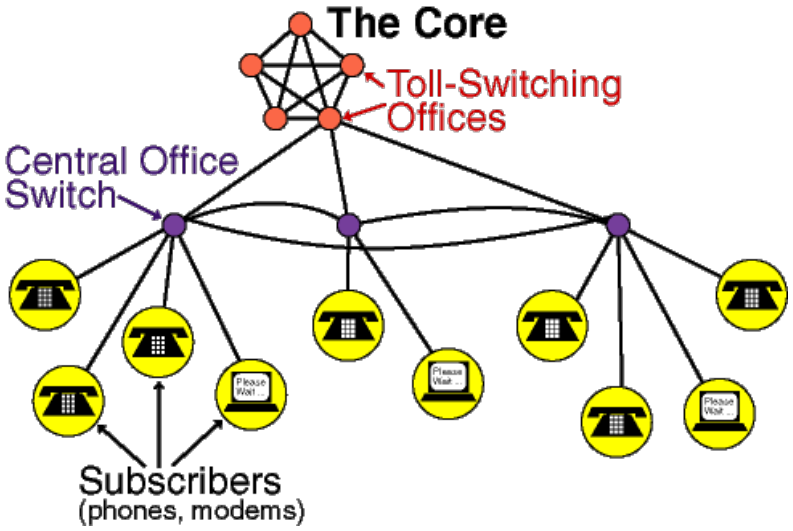
- Mantener la conectividad ante fallos en equipos y en enlaces
- Necesitamos calcular caminos alternativos rápidamente
- Con algoritmos distribuidos
- Para sobrevivir ante fallos deben ser caminos disjuntos




- IP dentro de LSPs MPLS, sobre wavelengths, dentro de fibras, dentro de grupos de fibras
- ¿Pero cómo sabe el nivel IP si dos caminos que ha calculado son físicamente disjuntos?
- ¿Y si van por la misma fibra? ¿O las fibras por el mismo tubo? ¿O los tubos por la misma canalización?
- Por ejemplo se hunde un túnel por donde pasan muchas canalizaciones

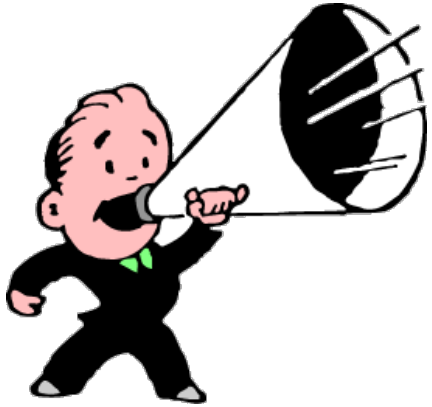
# Arquitectura de la PSTN

- La red telefónica juega con ventaja
- Normalmente desde la fibra entre centrales hasta el teléfono final era controlado por la misma operadora
- También su tráfico es muy predecible
- No solo por las fuentes y destinos involucrados sino por la cantidad y tipo de tráfico en cada flujo
- Eso permite calcular la capacidad necesaria en los enlaces
- Y permite decidir caminos óptimos en la red
- Incluso se pueden adaptar a patrones horarios (se pre-calculan rutas para franjas horarias)



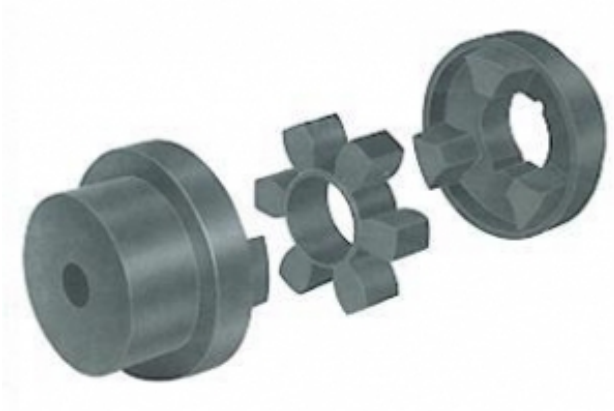
# Amplification principle

- [RFC3439] *"…there are non-linearities that occur at large scale which do not occur at small to medium scale"*
- En redes grandes incluso sucesos pequeños pueden tener grandes consecuencias
- Es decir, pequeñas perturbaciones pueden desestabilizar el sistema
- Ejemplo: añadir un pequeño número de enlaces puede hacer la resolución del routing mucho más compleja
- Ejemplo: descartar una celda ATM lleva a perder un paquete completo
- Para evitarlo se intenta que los cambios locales tengan efectos locales, no globales



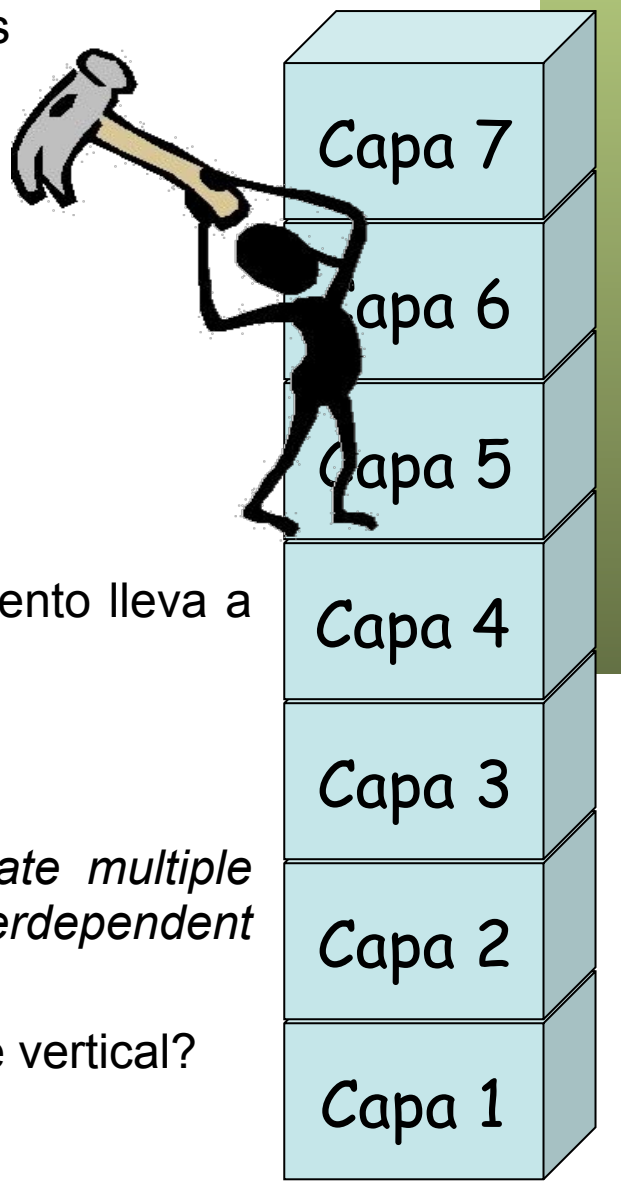
# Coupling principle

- [RFC3439] *"…as things get larger, they often exhibit increased interdependence between components."*
- Acoplamiento horizontal: en la misma capa de protocolos
- Acopamiento vertical: entre capas
- Ejemplo: conexiones TCP sincronizan el comportamiento de su ventana de control de congestión al compartir cuello de botella
- Una forma de reducir el acoplamiento es introducir aleatoriedad



# Arquitectura de capas

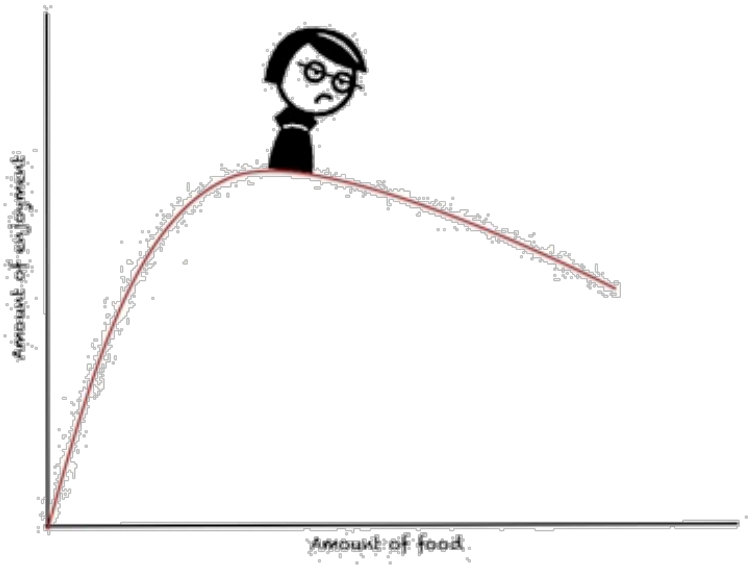
- La integración vertical nos permite repartir tareas
- Permite aislar implementaciones
- Las capas implementan funcionalidades como:
  - Control de errores
  - Control de flujo
  - Fragmentación
  - Multiplexación
  - Control de conexión
  - Direccionamiento
- Sin embargo la misma independencia y aislamiento lleva a que se reimplementen las funcionalidades
- Puede llevar a mayor complejidad
- Y acabamos violando el principio de simplicidad
- [RFC 1925] *"It is always possible to agglutinate multiple separate problems into a single complex interdependent solution. In most cases this is a bad idea."*
- ¿Tal vez es mejor una separación horizontal que vertical?




RFC 1925, "The Twelve Networking Truths", R.Callon, 1996

# La optimización es dañina

- Un poco de optimización está bien
- Pero la optimización pasado cierto punto introduce complejidad y mayor acoplamiento entre las capas
- Esto lleva a sistemas menos fiables
- Ley de los rendimientos decrecientes (*Diminishing Returns*): al aumentar algo que da beneficio, cada vez incrementa en menos el beneficio, hasta llegar a poder reducirse



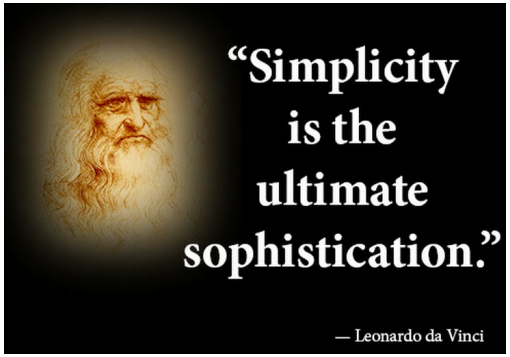
# Packet- vs Circuit- Switching

- ¿Qué es más eficiente?
- La conmutación de paquetes permite mayor eficiencia mediante la multiplexación estadística
- Sin embargo la utilización en los enlaces de conmutación de paquetes es muy baja
- ¿Por qué?
  - Es muy difícil predecir el tráfico así que se suele sobredimensionar grandemente la capacidad
  - Si falla otro enlace puede redirigirse todo su tráfico por éste, así que mejor dimensionemos con capacidad disponible para eso
  - es decir, es como si hiciéramos un 1:1
  - Las tecnologías tienen escasa granularidad (10GE pasa a 100GE)
  - Conseguimos QoS mediante *over-provisioning*

# Packet- vs Circuit- Switching

- ¿Qué es más **simple**?
- El principio end-to-end le da simplicidad a Internet, ¿sí?
- IP es simple, pero los routers y los protocolos (algoritmos distribuidos) no lo son
- Software es más complejo en routers
- Las operaciones que deben hacer por paquete son más complejas que las que hace un conmutador de circuitos
- El hardware del router es más complejo
- El router consume más potencia por esto último
- El control es más simple en CC (estático, out-of-band)
- ¿QoS en Internet? CC la tiene desde su concepción

- ¿Qué es más simple?
- ¿Qué es más eficiente?

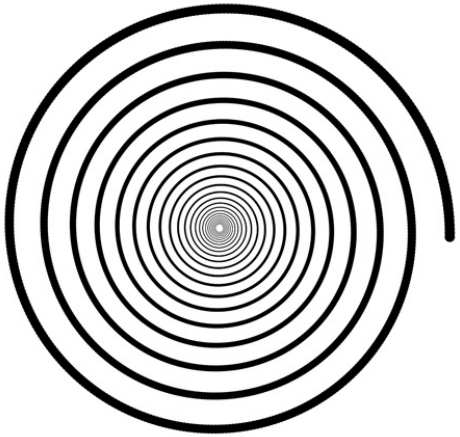


# Debugging

- Requiere experiencia con gran cantidad de componentes, tecnologías y protocolos
- Muchas veces requiere visitar la configuración de los equipos uno a uno
- Las herramientas son primitivas
- El problema puede no estar en la red sino en algún servicio auxiliar (¿DNS?)

# 99.999%

- ¿Podemos mejorar la tecnología para lograr redes fiables durante el 99.999% del tiempo?
- La gran mayoría de los fallos están causados por humanos o malos procedimientos
- Es decir, mejorando la tecnología tal vez logramos eliminar el … ¿20% de los fallos?
- ¿Y cómo lo hacemos? Aumentando la complejidad del sistema
- Mayor redundancia, caminos alternativos, algoritmos distribuidos para recuperarse ante fallos, etc
- Lo cual hace más frágil el sistema porque hay más puntos en los que los humanos nos podemos equivocar
- Es decir, reducimos los fallos de la tecnología pero aumentamos los fallos de los humanos



# Moraleja


RFC 3439

# Moralejas

- Hay mucha herencia e historia
- Las soluciones actuales no tienen por qué ser técnicamente las mejores
- Pueden estar condicionadas
  - Relaciones de poder entre empresas en su momento
  - Visiones subjetivas (“mi protocolo es mejor”)
  - Cuestiones económicas (“no puedo tirar toda la red que tengo para poner esa nueva tecnología”, “esto es ahora más barato”)
  - Desconocimiento de otra tecnología por la gente que defiende una
  - Relaciones entre grupos de “estandarización”
- Al aumentar la escala de la red hablamos de sistemas complejos
- Para hacerlo más fiable lo hacemos más complejo
- A más complejo, más fallos humanos
- A más fallos humanos más frágil

# Otras “verdades fundamentales”

## RFC 1925 “The Twelve Networking Truths”

- It Has To Work
- No matter how hard you push and no matter what the priority, you can’t increase the speed of light
- With sufficient thrust, pigs fly just fine. However, this is not necessarily a good idea. It is hard to be sure where they are going to land, and it could be dangerous sitting under them as they fly overhead
- Some things in life can never be fully appreciated nor understood unless experienced firsthand. Some things in networking can never be fully understood by someone who neither builds commercial networking nor runs and operational network.
- It is always possible to agglutinate multiple separate problems into a single complex interdependent solution. In most cases this is a bad idea.

# Otras “verdades fundamentales”

## RFC 1925 “The Twelve Networking Truths”

- It is easier to move a problem around (for example, by moving the problem to a different part of the overall network architecture) than it is to solve it
- It is always something … Good, Fast, Cheap: Pick any two (you can’t have all three)
- It is more complicated than you think
- For all resources, whatever it is, you need more
- One size never fits all
- Every old idea will be proposed again with a different name and a different presentation, regardless of whether it works
- In protocol design, perfection has been reached not when there is nothing left to add, but when there is nothing left to take away