Introducción al “Capacity planning” para servicios

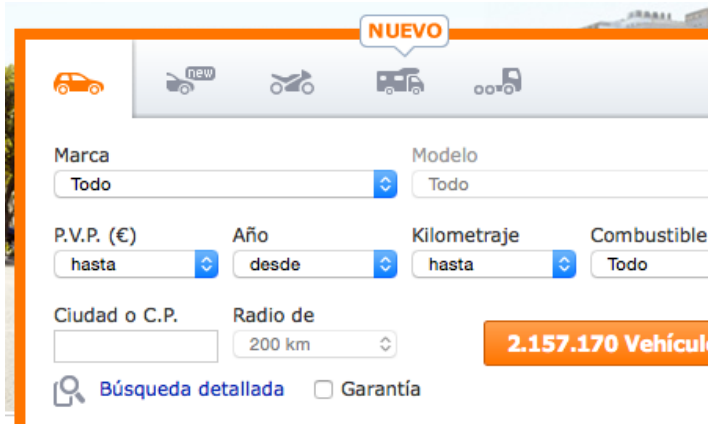
Area de Ingeniería Telemática
http://www.tlm.unavarra.es

Grado en Ingeniería en Tecnologías de Telecomunicación, 4º

Capacity Planning

Ejemplo

- Empresa de venta de coches virtual
- Aglutina ofertas de 1300 concesionarios afiliados
- La base de datos almacena información sobre los vehículos
- Recibe peticiones de:
  - Documentos e imágenes de la web
  - Búsquedas en la base de datos
  - Órdenes de compra



Ejemplo: “El cambio”

- Se van a añadir nuevos concesionarios
- Eso añade más modelos y ofertas
- Se espera que incremente el número de búsquedas recibidas
- Se hará en 3 fases que implicarán:
  - Un incremento del 10% en la llegada de peticiones
  - Un incremento del 20%
  - Un incremento del 30%

Ejemplo: “La calidad”

- Las búsquedas son las peticiones críticas
- Se supone que el 5% de las búsquedas generan una venta, que lleva a un beneficio medio de unos 300€
- Si su tiempo de respuesta está entre 4s y 6s se pierde el 60% de las búsquedas porque los usuarios las abortan
- Si el tiempo de respuesta excede los 6s se pierden el 95%
- Es decir, se deben cumplir unos “niveles de servicio” para no perder posibles ventas

Ejemplo: “Las preguntas”

- ¿Sopotará el sitio web el incremento de carga manteniendo tiempos de respuesta de menos de 4s?
- Si se satura, ¿con qué carga lo hace?
- ¿Cuánto dinero se pierde/deja de ganar si se satura?

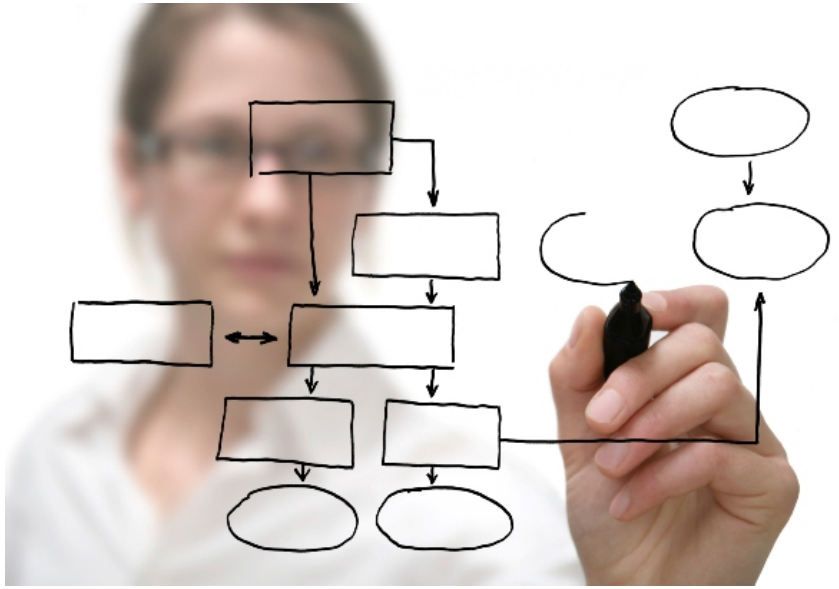
Ejemplo: "El resultado"

- El resultado de la predicción y análisis podría ser algo así:

	Hoy	Hoy +10%	Hoy +20%	Hoy +30%
Búsquedas al día	48638	53501	58365	63229
Tiempo de respuesta (s)	2.9	3.8	5.7	11.3
Ventas perdidas (%)	0	0	60	95
Ventas diarias	2432	2675	1750	158
Beneficio diario (x1000€)	729	802	525	47
Beneficio diario potencial (x1000€)	729	802	875	948
Pérdidas diarias (x1000€)			350	899

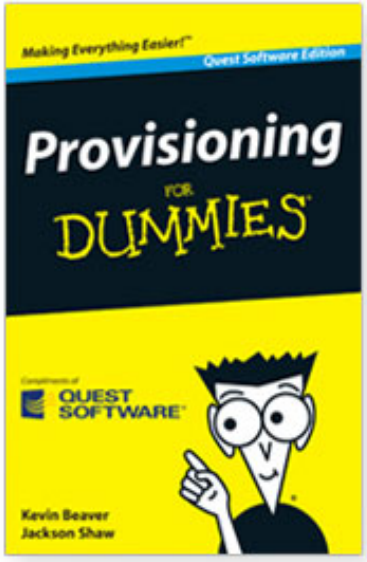
Capacity Planning

- Predecir cuándo la carga futura saturará el sistema
- Determinar la mejor forma (menor coste) de evitar la saturación
- Necesitamos predecir la evolución de la carga, debido a cambio en aplicaciones o usuarios
- Se busca mantener unos niveles de servicio
- Debemos predecir; si esperamos a que se dé el problema se pueden perder clientes para cuando se resuelva (aumentar número de servidores, capacidad de enlaces, etc)



Capacity Planning

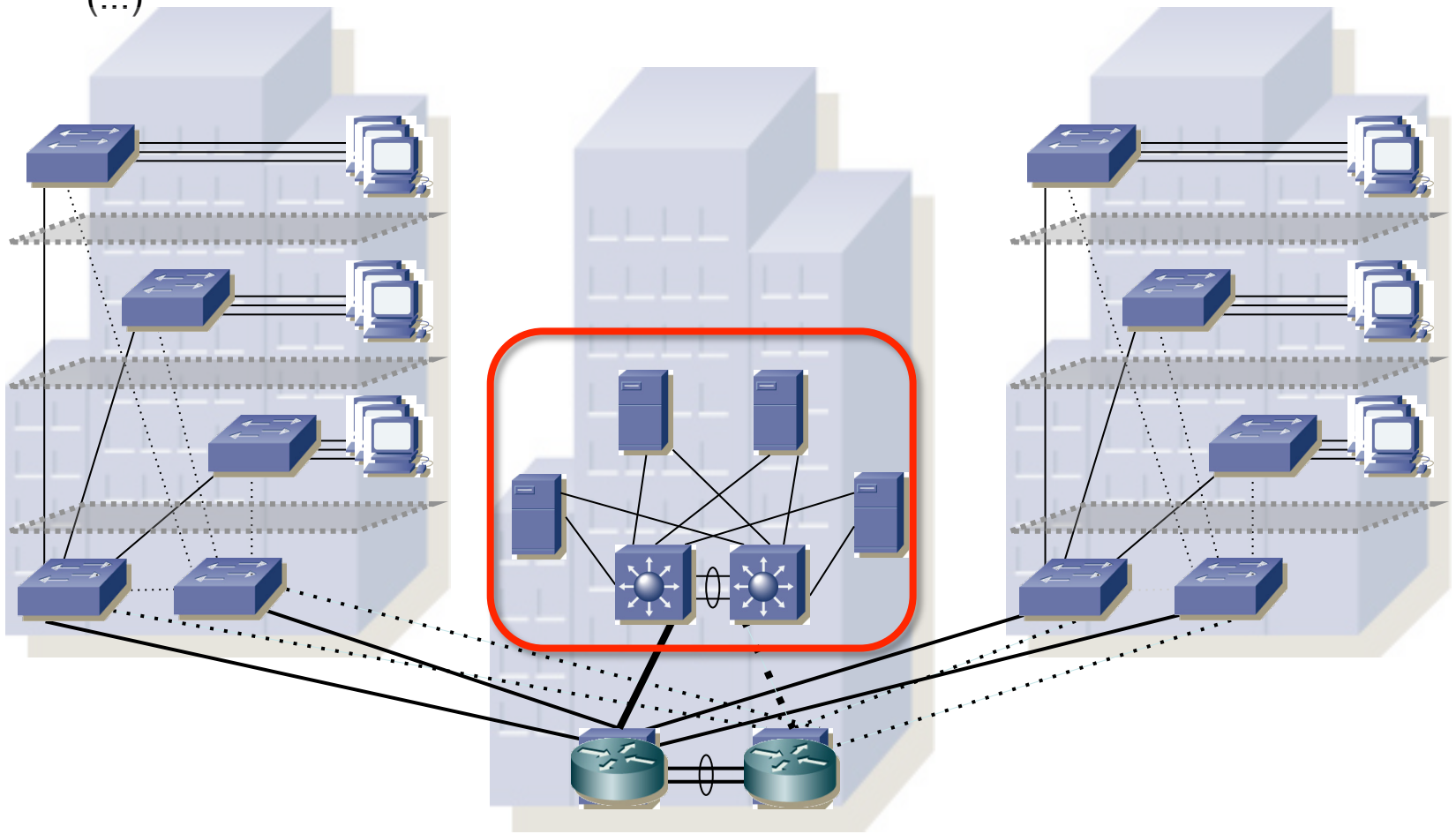
- ¿Cuánto gastar en estar preparado para un problema? Pregúntate cuánto pierde la empresa si se da el problema
- Resolverlo puede ser aumentar el número de servidores, su capacidad de proceso, la capacidad de enlaces, rediseñar la arquitectura del software, rediseñar la red, etc
- Un *provisioning* rápido es muy interesante (hoy en día, mediante virtualización)
- Si resolver la saturación requiere cambiar la arquitectura de la aplicación, eso no es rápido
- Si resolver la saturación requiere cambiar la arquitectura de la red, eso tampoco lo es (empiezan a aparecer las SDNs)



Arquitectura del servicio

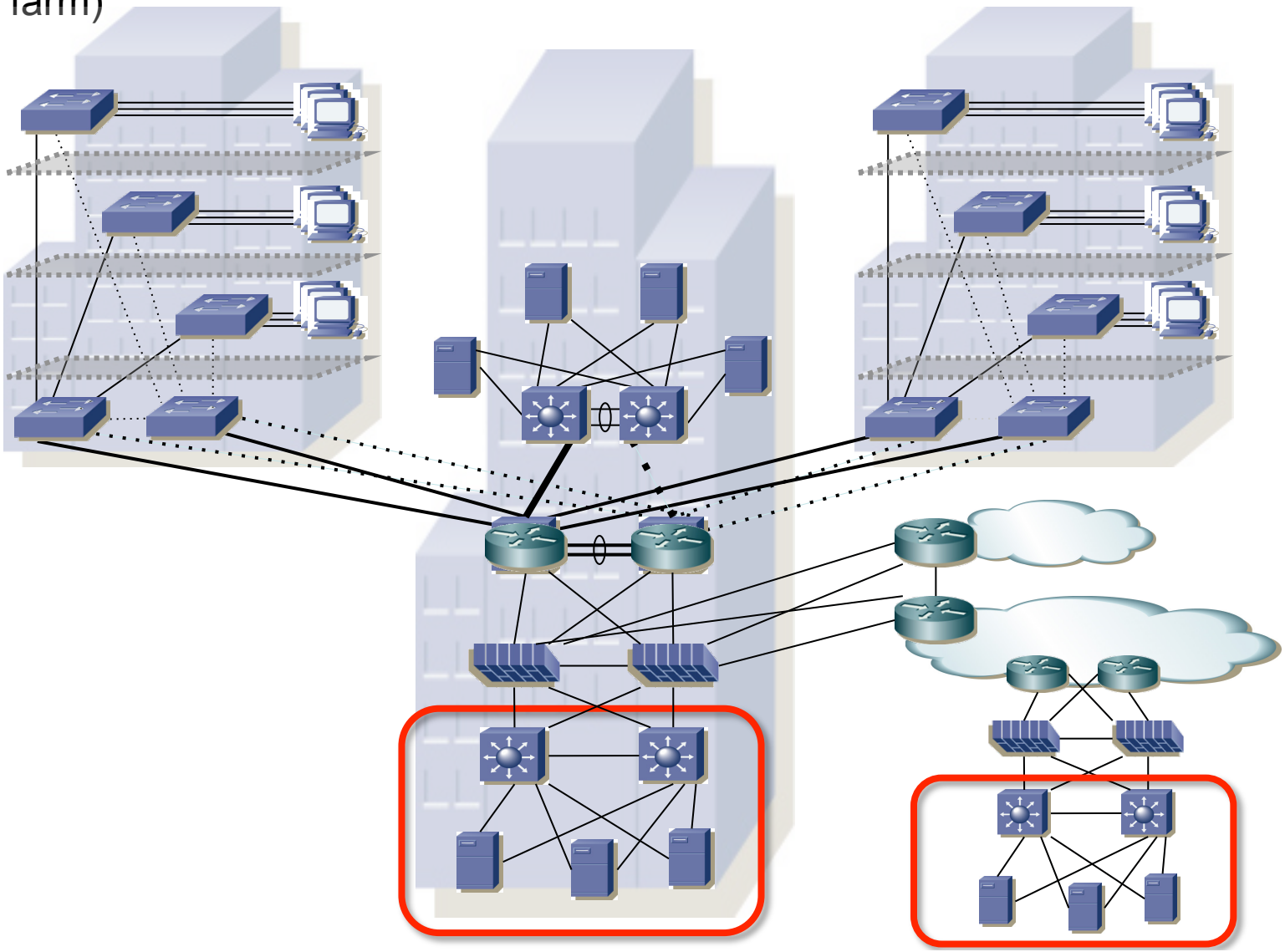
Arquitectura de red

- Nos centraremos en la parte de servidores y el efecto de su arquitectura
- La red interna pero fundamentalmente la externa podrá tener un efecto notable, o no
- Puede ser un data center y servidores internos de la empresa (Intranet)
- (...)



Arquitectura de red

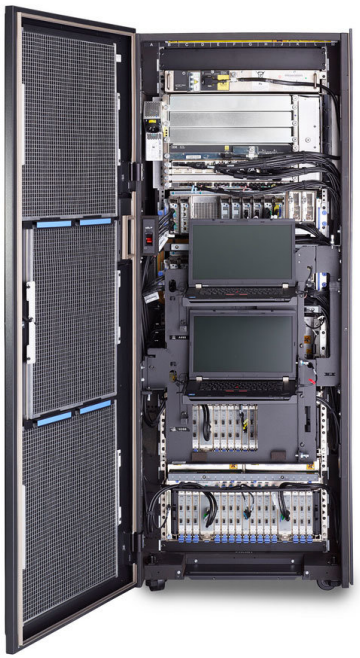
- O servidores accesibles desde el exterior (Extranet o Internet server farm)



Arquitectura de aplicación

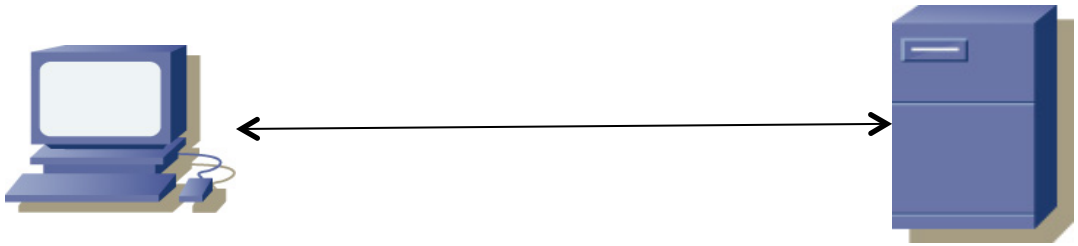
- Mainframe
  - Solución centralizada con clientes simples (*thin client*) (años 60)
  - Capacidades de virtualización
  - Hoy en día grandes servidores, muy flexibles
  - Ejemplo: IBM zEnterprise BC12 H06 (2014) 18 CPUs hexacore a 4.2GHz con hasta 496 GB de RAM con capacidades criptográficas dedicadas por CPU

IBM System/360 (Picture Courtesy IBM)



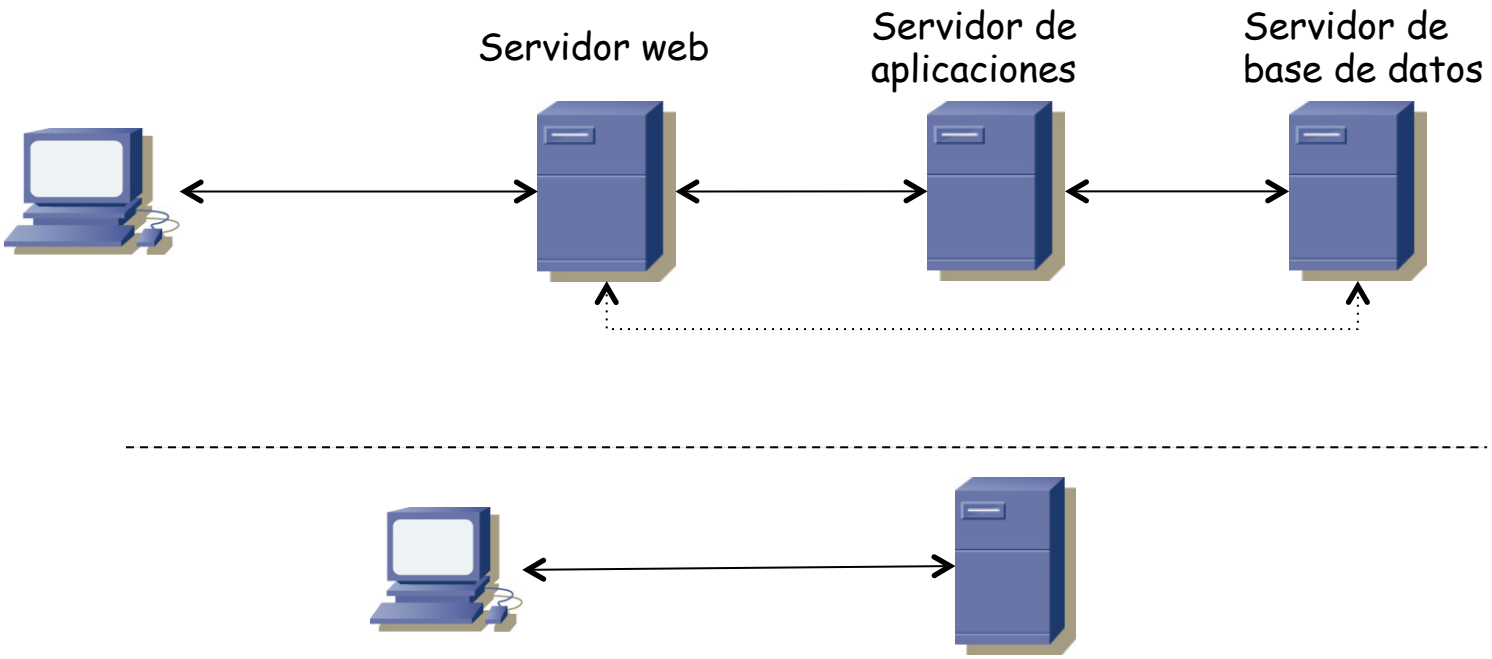
Arquitectura de aplicación

- Cliente-servidor
  - Servidores de menor capacidad que *Mainframe*
  - Clientes de mayor capacidad (*thick client*)
  - Interfaces propietarios hasta llegar la web
  - Se migra de una arquitectura básica c/s a una basada en web
  - Se sigue lo que se conoce como el modelo *n-Tier*



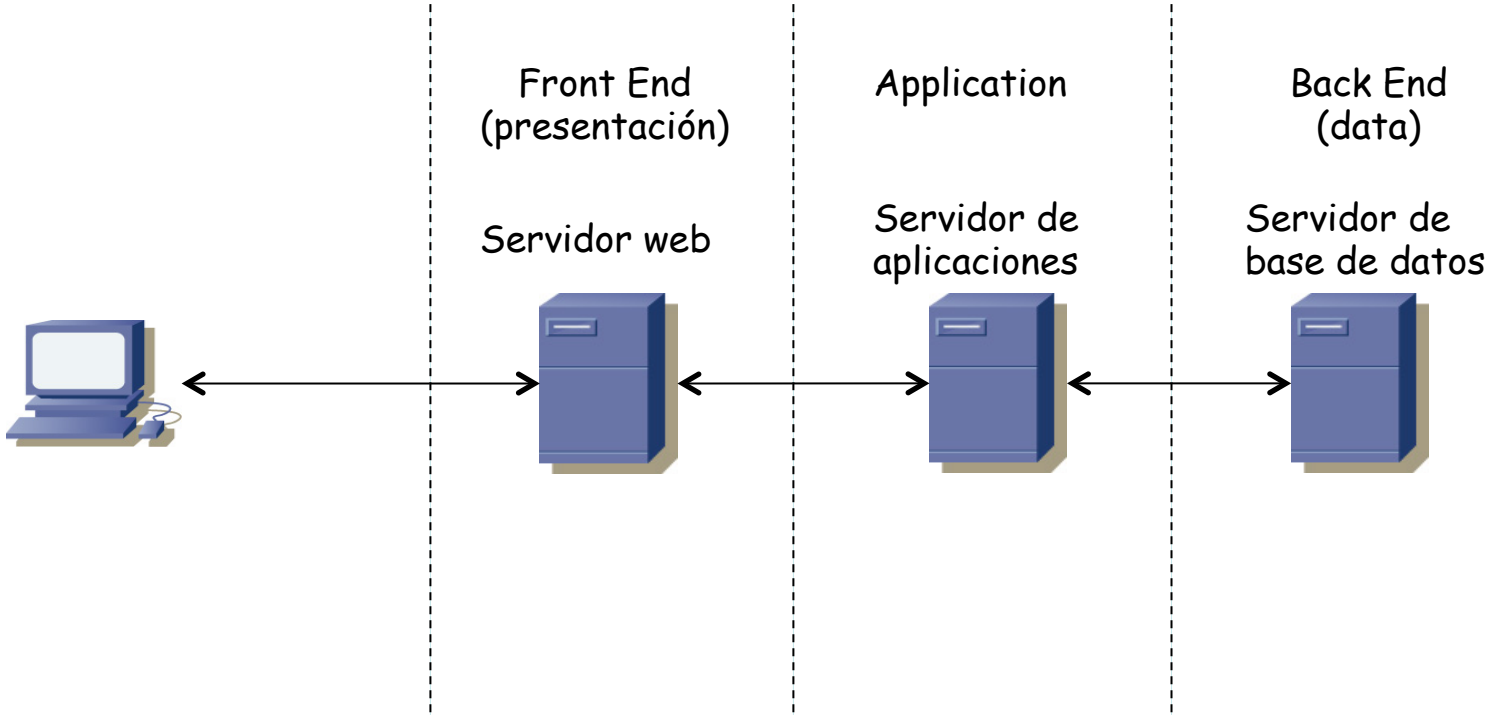
n-Tier model

- Las funcionalidades del servidor se dividen en niveles/capas/*tiers*
- En lugar de estar todo en un solo servidor pueden distribuirse en 2, 3 o más capas
- Permite avanzar a una computación distribuida
- Eso permite escalar (*scale-out*) el sistema para mayores cargas
- Simplifica y distribuye el control de la aplicación



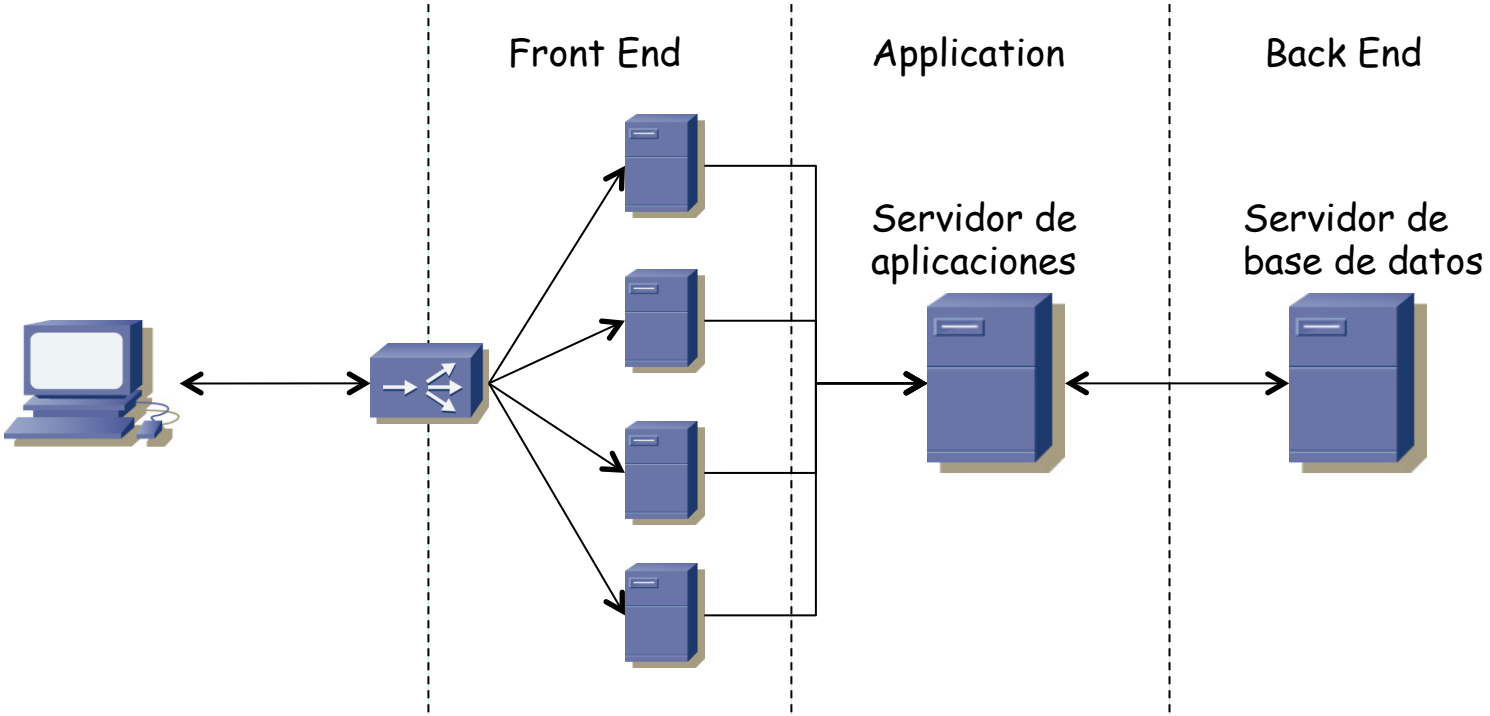
Arquitectura *multitier*

- La arquitectura de red ofrece separación física y lógica entre las capas de la aplicación
- Segmentos de red:



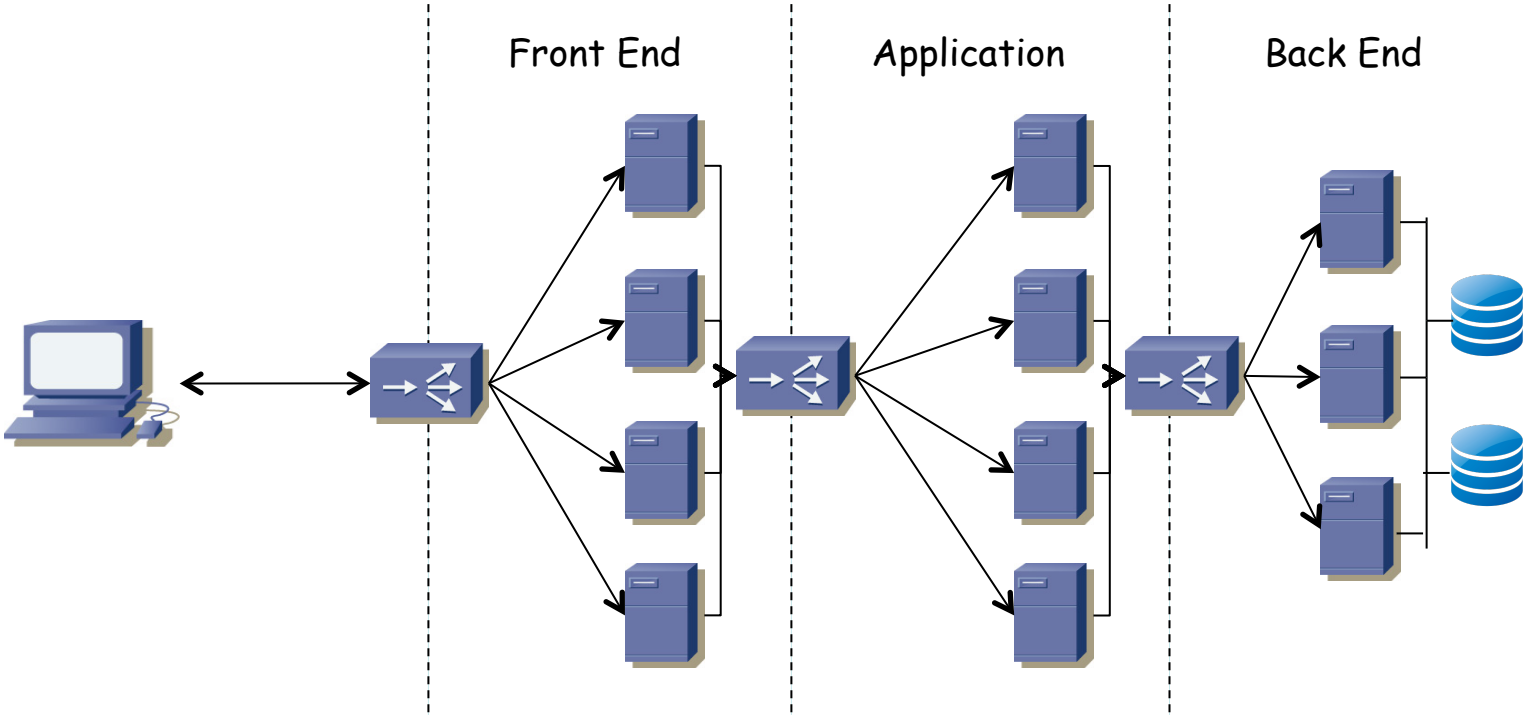
Arquitectura *multitier*

- No vamos a tratar la arquitectura específica de la electrónica de red
- Sí nos interesa recordar que el escalado se consigue mediante equipos que hacen balaceo de carga (...)



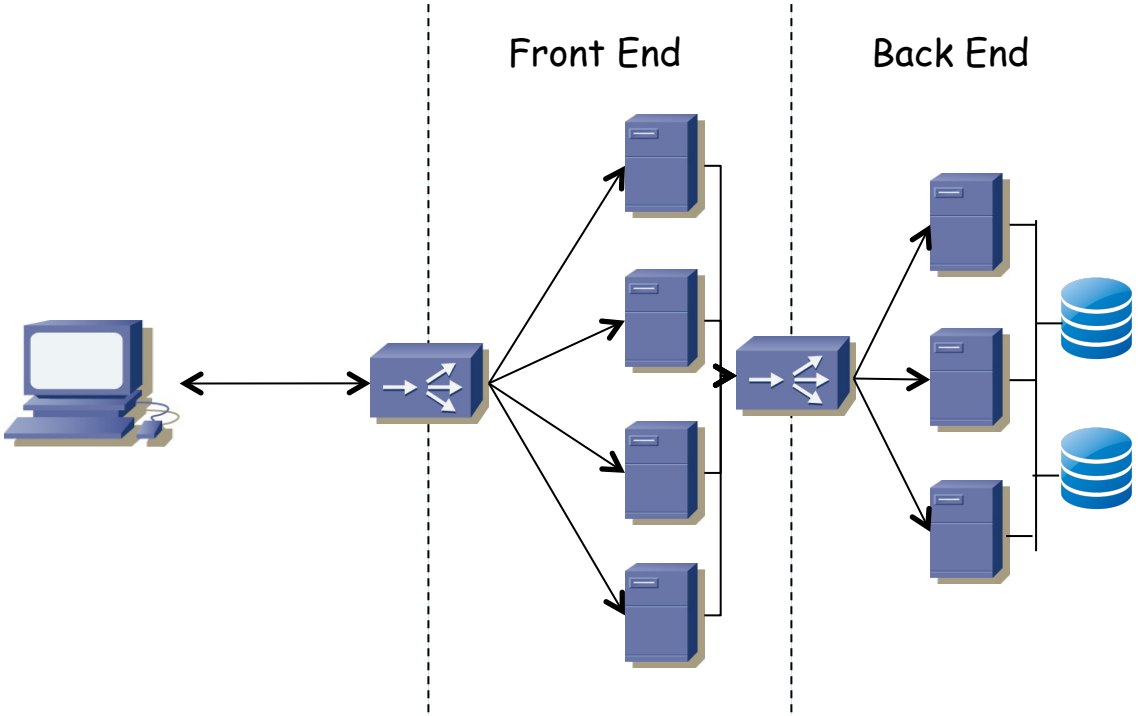
Arquitectura *multitier*

- No vamos a tratar la arquitectura específica de la electrónica de red
- Sí nos interesa recordar que el escalado se consigue mediante equipos que hacen balaceo de carga
- Y para la interconexión habrá conmutadores L2, routing L3, firewalls, los discos pueden estar en red, etc



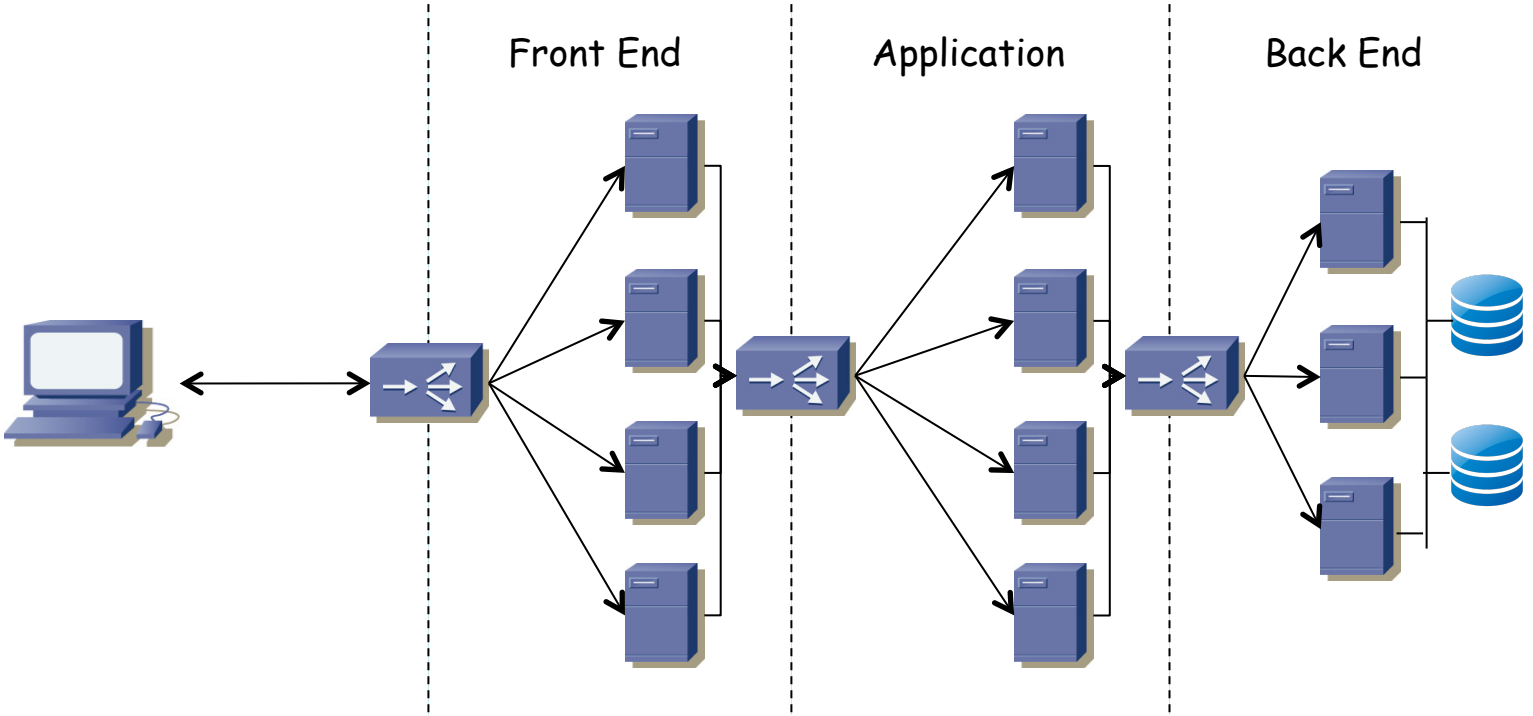
2-tier vs 3-tier

- ¿Necesitamos 3 tiers?
- Depende de dónde necesitemos crecer al aumentar la carga



Arquitectura *multitier*

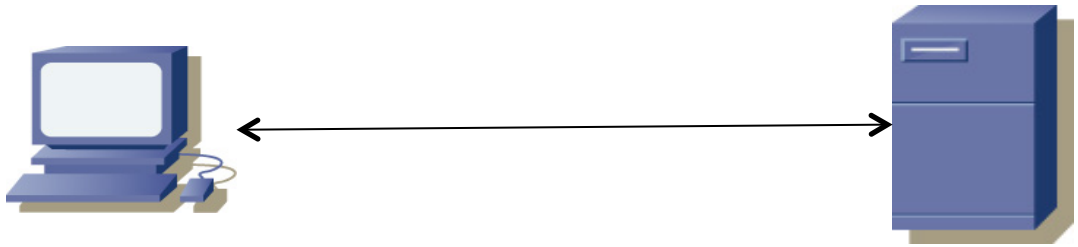
- ¿Cuellos de botella?
  - Servidor/es
  - Base de datos
  - Protocolos
  - Acceso a web services externos
  - (...)



Modelo del servidor

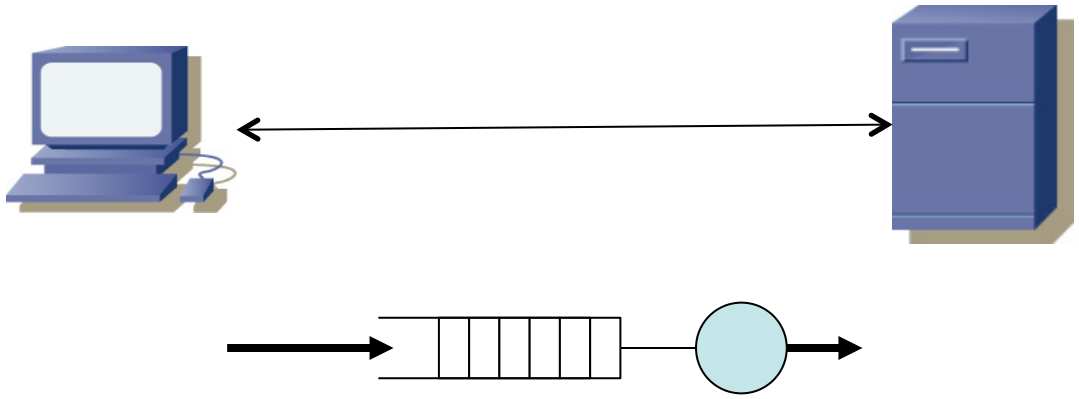
Modelo de un servidor

- Recibe peticiones de múltiples usuarios
- Las atiende en serie
- Puede tener que dejar otras a la espera mientras atiende a una
- Tiene un tiempo de respuesta que vendrá condicionado por:
  - El tiempo de espera a ser atendido
  - El tiempo que tarde en calcular la respuesta (CPU)
  - El tiempo de acceso a disco o base de datos
  - El tiempo debido a la red
- Algunos de esos tiempos pueden paralelizarse (ej: mientras espera un ACK sigue leyendo del disco)
- (...)



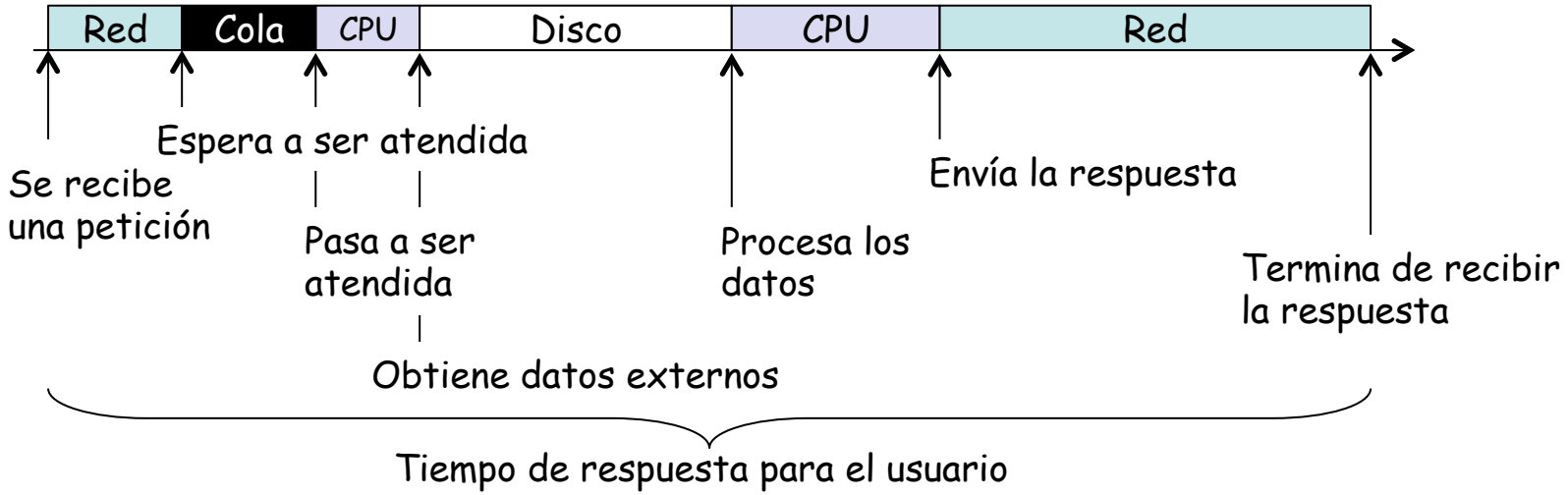
Modelo de un servidor

- Es decir, podemos modelar su comportamiento con un sistema con 1 servidor y una cola
- El tiempo de servicio engloba todos los tiempos mencionados
- La cola son las peticiones que se han recibido y que aún no se han atendido
- Normalmente esa cola tendrá una profundidad máxima acotada
- Es un problema modelar ese tiempo de servicio
- Por ejemplo depende de las condiciones de la red



Tiempo de respuesta

- Podemos descomponerlo para modelar cada componente
  - La componente de red vendrá condicionada por los mecanismos de TCP
  - El tiempo de espera depende de las peticiones anteriores y sus tiempos de servicio (modelo de colas)
  - El uso de CPU depende del tipo de trabajo y la máquina
  - El tiempo de acceso a disco depende de la cantidad de información, dónde esté en el disco y las características de éste



Tiempo de respuesta

- En mayor o menor medida, todos esos tiempos son variables
  - El tiempo de CPU depende del tipo de petición
  - El tiempo de espera en cola depende de qué otras tareas tiene la CPU
  - El tiempo de disco depende del tamaño del documento, dónde esté, el tipo de disco, otros accesos concurrentes, etc
  - El tiempo de red depende de los tamaños, el estado de la red, la implementación de TCP de los extremos, etc
- Según las circunstancias puede que algunos sean despreciables frente a otros