Sistemas Distribuidos

Servicio de Nombres

Índice

Una historia basada en hechos reales

- Quiero contactar con persona en un contexto para pedirle algo
  - Contexto: una organización, una ciudad, un país, el mundo, ...
  - Necesito su dirección de contacto (p.e. nº teléfono en ese contexto)
  - *Nombre* (*quién*) [permanente] → *Dirección* (*dónde*) [transitorio]
    - Veremos que nombres y direcciones no son tan diferentes…
- Servicio telefónico "páginas blancas" (Servicio de nombres)
  - Necesito conocer nº teléfono de servicio de guía del contexto dado
  - Y especificar la persona con "nombre" unívoco en ese contexto
    - Nombre/apellidos | nº empleado | nº DNI
  - Guía proporciona nivel de indirección respecto a dirección contacto
    - Permite que persona cambie nº tfno (cuidado con agenda-caché)
  - Puede requerirse cadena de consultas; ¿nº tfno empleado?:
    - 1º obtengo nº tfno empresa; 2º centralita empresa me da tfno empleado

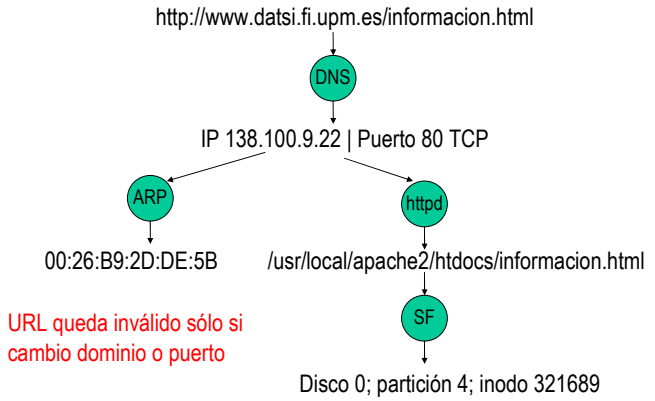
Una historia basada en hechos reales

- Nombres y direcciones suelen tener carácter jerárquico
  - Facilita su administración y gestión
  - Ejs. Nombres: ID empleado internacional (ISBN, cuenta bancaria, ...)
    - Cambio de recurso en jerarquía puede invalidar el nombre
  - Ejs. Direcciones: nº teléfono o dirección postal
  - Encaminamiento jerárquico
- A veces quiero contactar con cualquiera que dé un servicio
  - Necesito conocer condiciones de servicio para elegir
- Servicio telefónico "páginas amarillas" (Servicio de directorio)
- ¿Y si ni siquiera sé nº tfno. de servicios de guía (o no los hay)?
  - Quizás debería gritar pidiendo ayuda
  - Descubrimiento de servicios

URI: Uniform Resource Identifier

- 2 tipos de identificadores de recursos URIs en Internet:
  - Nombres URNs y direcciones URLs
- *Uniform Resource Name*: *Nombre* (*qué*) [permanente]
  - Identifican recurso sin incluir información de localización
  - Requiere un proceso de traducción
- *Uniform Resource Locator*: *Dirección* (*dónde*) [¿transitorio?]
  - Pueden verse afectados si recurso "se mueve"
  - Algunos URL se pueden considerar permanentes
- Ejemplos wikipedia URN vs. URL
  - *urn:ietf:rfc:3187*
  - *http://tools.ietf.org/html/rfc3187.html*

Ejemplo: Niveles de traducción de URL



Servicio de nombres

- Nombre de entidad en SD → punto(s) de acceso a la entidad
  - Sockets: Dir(s) IP+ puerto(s)+ protocolo(s)
  - RMI o CORBA: referencia(s) a objeto(s)
- Nombre permite referirse a una entidad única en SD
  - Aunque puede estar replicada (p.e. fichero en Coda)
  - y puede haber varios nombres para la misma entidad (alias)
- Hay diversos tipos de entidades en SD
  - ficheros, usuarios, grupos, procesos, dispositivos, máquinas, ...
- Serv. de nombres específicos para algunos tipos de entidades
  - para ficheros (SFD), para máquinas (DNS), ...
- Ideal: servicio de nombres integral para todas las entidades
  - Excepto ficheros por gran volumen y frecuencia de actualizaciones

Jerarquía de nombres

- SD incluye muchas entidades muy diversas
  - Como SFD, organización jerárquica facilita asignación y gestión
    - Impresoras de distintos departamentos con el mismo nombre
- Espacio de nombres jerárquico
  - Entidades contenedoras de otras entidades (directorios)
- Traducción de nombres (*pathnames*):
  - Proceso iterativo que parte de un nodo inicial
    - Necesidad de conocer traducción de nodo inicial
    - Absoluta (nodo raíz) vs. Relativa (nodo intermedio)
  - Proceso costoso: Uso de caché en traducción
    - Información inválida si migración
    - Garantía de validez vs. coste de mantener la coherencia de la caché
    - Muchos servicios de nombres pueden devolver información obsoleta

Implementación de servicio de nombres

- Operaciones del servicio de nombres
  - Asociar/Desasociar nombre con una entidad; Crear directorio
  - Traducir nombre
- No factible uso de servidor de nombres único:
  - Problemas de escala, rendimiento y fiabilidad
- Tres técnicas para paliarlos:
  - Caché de traducciones
  - Distribución de espacio de nombres
  - Replicación de espacio de nombres
- Caché de traducciones:
  - Problema de coherencia
  - En DNS: No hay garantía
    - Servidor retorna TTL ("tiempo de vida") de información

Distribución y replicación

- Espacio de nombres partido y distribuido entre servidores
  - Se requiere info. que "monte" particiones para formar árbol único
  - Cada partición gestionada por (al menos) un servidor
    - Posibilita administración distribuida
  - Mismas alternativas de navegación que en SFD
    - Iterativa, Transitiva y Recursiva
- Partición replicada en varios servidores
  - Fiabilidad y rendimiento, pero hay que asegurar coherencia
    - Fiabilidad: mejor réplicas en distintas subredes
  - Esquema simétrico:
    - Consulta a cualquier réplica
    - Actualización simultánea en todas las réplicas
  - Esquema asimétrico: 1 primario/maestro y N secundarios/esclavos
    - Consulta a cualquier réplica
    - Actualización en primario con propagación a réplicas (Modo *push* o *pull*)

Implementaciones alternativas

- Uso de broadcast/multicast:
  - ¿Dónde está recurso de nombre N?
  - No aplicable a SD propósito general (mala escalabilidad y eficiencia)
  - Uso en descubrimiento servicios o para aspectos específicos (ARP)
- Uso de DHT
  - Al fin y al cabo es un mecanismo de traducción
- Uso de cadenas de ubicación probable
  - Aplicable a sistemas con recursos móviles
  - Cuando recurso migra, antigua ubicación almacena la nueva
  - Se crean cadenas de ubicación probable de un objeto
  - Localización sigue cadena hasta encontrar ubicación actual
  - Si respuesta atraviesa cadena, intermediarios saben nueva ubicación
  - El recurso puede tener un nodo *home*

Domain Name System (DNS)

- Servicio de nombres de máquinas en Internet: nombre → IP
  - No es un serv. nombres general pero ilustrativo por escalabilidad
  - Diseño genérico: aunque uso habitual nombre de máquinas Internet
  - Inicios de Internet: fichero HOST que se actualizaba periódicamente
- Espacio de nombres de DNS jerárquico
  - Nombre: secuencia de dominios (≈directorios) de dcha. a izda.
    - www.datsi.fi.upm.es. → . + es + upm + fi + datsi
  - Dominio raíz: . → Caminos absolutos (FQDN) terminan con .
  - Dominios nivel superior (TLD)
    - gTLDs: genéricos (com, org, ...)
    - ccTLDs: por país (¿qué pasa con el de Tuvalu?)
  - De segundo nivel, de tercero, ...
- Implementación más usada BIND

Espacio de nombres distribuido: Zonas

- Zona DNS: partición del árbol global (zona ≠ dominio)
  - Información recursos de un dominio y sus subdominios no delegados
  - Delegación de dominios
    - Un subdominio puede tener su propia zona
    - Dominio padre incluye "punto de montaje" a esa zona subordinada
  - Diseño habitual: delegar todos los subdominios
    - Una zona para cada dominio (zona ≈ dominio)
    - Incluso a veces a los mismos servidores que el dominio del que cuelgan
- Cada zona está replicada:
  - 1 servidor maestro/primario y *N* (al menos 1) esclavos/secundarios
  - Fiabilidad: mejor réplicas en distintas subredes
- Información contenida en una zona:
  - Colección de *Resource Records* (RR) que describen sus recursos

Resource Record

- Definición de un recurso: *Nombre Tipo Clase TTL Datos*
  - Clase *IN* para Internet (otros *HS*, para Hesiod, y *CH*, para Chaos)
  - NOTA: *Nombre* puede tener * a la izqda. (*wildcard RR*; no lo tratamos)
- Fichero de zona:
  - Fichero de texto en primario define RRs de una zona: 1 RR/línea
    - Aunque RRs se transmiten en binario
  - Incluye RRs de recursos del dominio y de subdominios no delegados
  - Sintaxis definida para facilitar introducción de datos en fichero de zona
    - Macros, caracteres especiales, caminos relativos, omisión de campos,...
- Diversos tipos de RRs
  - Nos centramos en SOA, A, AAAA, PTR, CNAME, MX, SRV, TXT y NS
  - No tratamos los RRs relacionados con la extensión DNSSEC
    - Proporciona autenticación e integridad en DNS

RR de tipo SOA (*Start of Authority*)

- Comienzo de definición de una zona
- Ejemplo de definición en fichero de zona (wikipedia)

```
example.com.  IN  SOA  ns.example.com. username.example.com. (
        2007120710 ; serial number of this zone file
        1d         ; slave refresh (1 day)
        2h         ; slave retry time in case of a problem (2 hours)
        4w         ; slave expiration time (4 weeks)
        1h         ; maximum caching time in case of failed lookups (1 hour)
        )
```

- Ejemplo de consulta: *dig fi.upm.es. SOA*

```
fi.upm.es.        86400  IN    SOA    chita.fi.upm.es. hostmaster.fi.upm.es. 2013102101 28800 7200 2419200
    3600
```

Dominio; TTL; Clase Internet; Start Of Authority; S. maestro; responsable; nº serie (incrementar si cambio);
Periodo de actualización de secundario; Tiempo de reintento de secundario antes actualización fallida;
Tiempo de expiración de info. de secundario ante actualización fallida; TTL para cache negativa (tiempo en cache de consultas erróneas)

RR de tipo A o AAAA

- Dirección de máquina: A (IPv4) y AAAA (IPv6)
- Ejemplo de definición en fichero de zona (wikipedia)

```
www.example.com.  A      192.0.2.1       ; IPv4 address for example.com
www.example.com.  AAAA   2001:db8:10::1  ; IPv6 address for example.com
```

- Ejemplo de consulta: *dig www.fi.upm.es. A*

```
www.fi.upm.es.      86400  IN    A     138.100.243.10
```

- Múltiples recursos con mismo nombre (reparto de carga)

```
www.google.es.      300    IN    A     130.206.193.48
www.google.es.      300    IN    A     130.206.193.59
www.google.es.      300    IN    A     130.206.193.26
..... Hasta 16 .....
```

  - Nótese TTL bajo en RR para favorecer el reparto de carga

RR de tipo PTR

- Traducción inversa dirección IP → Nombre
- Gestionada también por DNS: mediante dominios especiales
- Para IPV4: *in-addr.arpa*.
  - Traducir 138.100.243.10 → 10.243.100.138.in-addr.arpa.
- Para IPV6: *ip6.arpa*.
  - Traducir 2001:720:41c:40:12:100:4:4 → 4.0.0.0.4.0.0.0.0.0.1.0.2.1.0.0.0.4.0.0.c.1.4.0.0.2.7.0.1.0.0.2.ip6.arpa.
- Ejemplos de consulta:
  - *dig 10.243.100.138.in-addr.arpa. PTR*

```
10.243.100.138.in-addr.arpa. 86400 IN  PTR    www.fi.upm.es.
```

  - *dig 4.0.0.0.4.0.0.0.0.0.1.0.2.1.0.0.0.4.0.0.c.1.4.0.0.2.7.0.1.0.0.2.ip6.arpa. PTR*

```
4.0.0.0.4.0.0.0.0.0.1.0.2.1.0.0.0.4.0.0.c.1.4.0.0.2.7.0.1.0.0.2.ip6.arpa. 86302IN PTR
    galileo.ccupm.upm.es.
```


RR de tipo CNAME (Canonical NAME)

- Alias: Nuevo nombre para mismo recurso

```
www.datsi.fi.upm.es.      86400  IN    CNAME   avellano.datsi.fi.upm.es.
avellano.datsi.fi.upm.es. 86400  IN    A       138.100.9.22
```

- Frente a:

```
www.datsi.fi.upm.es.      86400  IN    A       138.100.9.22
avellano.datsi.fi.upm.es. 86400  IN    A       138.100.9.22
```

- Más flexibilidad ante cambios pero ineficiencia por indirección
- Pueden encadenarse:

```
www.elpais.com.          967     IN    CNAME   elpais.es.edgesuite.net.
www.elpais.es.           1456    IN    CNAME   elpais.es.edgesuite.net.
elpais.es.edgesuite.net. 10467   IN    CNAME   a1749.g.akamai.net.
a1749.g.akamai.net.      20      IN    A       130.206.192.24
a1749.g.akamai.net.      20      IN    A       130.206.192.49
```


RR de tipo MX

- Servidores de correo para dominio con orden de preferencia
- Formato:

  name TTL class MX priority target

- Ejemplo de consulta: *dig upm.es. MX*

```
upm.es.          73788  IN    MX    10 relay.upm.es.
upm.es.          73788  IN    MX    30 relay4.upm.es.
upm.es.          73788  IN    MX    50 correo.upm.es.
```

  - Número indica orden de preferencia:↓ prioridad → ↑ preferencia
- Remitente de correo debe contactar con servidor de menor nº
  - Si caído con el siguiente, …

RR de tipo SRV

- Permite especificar qué máquinas dan un servicio en el dominio
- Formato:

  _service._proto.name TTL class SRV priority weight port target

- Permite especificar prioridades y reparto entre misma prioridad
- Ejemplo de wikipedia:

```
_sip._tcp.example.com. 86400 IN SRV 10 60 5060 bigbox.example.com.
_sip._tcp.example.com. 86400 IN SRV 10 20 5060 smallbox1.example.com.
_sip._tcp.example.com. 86400 IN SRV 10 10 5060 smallbox2.example.com.
_sip._tcp.example.com. 86400 IN SRV 10 10 5066 smallbox2.example.com.
_sip._tcp.example.com. 86400 IN SRV 20 0 5060 backupbox.example.com.
```

- ¿Por qué se usan tan poco? ¿Por qué no se usan para la Web?

RR de tipo TXT

- Permite asociar texto con un nombre
- Una forma de añadir funcionalidad a DNS sin nuevos RRs
- Ejemplos de aplicación:
  - *Sender Policy Framework* (SPF):
    - Validar qué máquinas de un dominio pueden enviar correo
      - NOTA: existe también un RR de tipo SPF
  - Verificar la propiedad de un dominio para Google
- Ejemplo de consulta: *dig fi.upm.es. TXT*

```
fi.upm.es.      86400 IN TXT "v=spf1 ip4:138.100.8.0/24 ip4:138.100.198.0/24 ip4:138.100.4.67 -all"
fi.upm.es.      86400 IN TXT "google-site-verification=rJT2Yatvyyg4HepVHZ-
    nk6LrxHNNlArZHaNxhkFCSgU“
```


RR de tipo NS (*Name Server*)

- Primer uso: especificar servidores de nombres para un dominio
- Ejemplo: *dig fi.upm.es. NS*

```
fi.upm.es.      86400 IN NS chita.fi.upm.es.
fi.upm.es.      86400 IN NS zape.fi.upm.es.
fi.upm.es.      86400 IN NS tarzan.fi.upm.es.
fi.upm.es.      86400 IN NS galileo.ccupm.upm.es.
fi.upm.es.      86400 IN NS ns.fi.upm.es.
```

- Ejemplo: *dig 8.100.138.in-addr.arpa. NS*

```
8.100.138.in-addr.arpa. 86400 IN NS zape.fi.upm.es.
8.100.138.in-addr.arpa. 86400 IN NS chita.fi.upm.es.
8.100.138.in-addr.arpa. 86400 IN NS galileo.ccupm.upm.es.
8.100.138.in-addr.arpa. 86400 IN NS tarzan.fi.upm.es.
8.100.138.in-addr.arpa. 86400 IN NS ns.fi.upm.es.
```


RR de tipo NS para delegación

- Segundo uso: delegar subdominio a s.nombres (pto. montaje)
  - Aparece como nombre del RR el del subdominio
  - Lista subdominios delegados no se puede obtener mediante consulta
  - Ejemplo: UPM delega administración de sus recursos DNS a FI:
    - fichero de zona de *upm.es.* debe incluir:

```
fi.upm.es.      86400 IN NS chita.fi.upm.es.
fi.upm.es.      86400 IN NS zape.fi.upm.es.
fi.upm.es.      86400 IN NS tarzan.fi.upm.es.
fi.upm.es.      86400 IN NS galileo.ccupm.upm.es.
fi.upm.es.      86400 IN NS ns.fi.upm.es.
```

    - fichero de zona de *100.138.in-addr.arpa.* debe incluir:

```
8.100.138.in-addr.arpa. 86400 IN NS zape.fi.upm.es.
8.100.138.in-addr.arpa. 86400 IN NS chita.fi.upm.es.
8.100.138.in-addr.arpa. 86400 IN NS galileo.ccupm.upm.es.
8.100.138.in-addr.arpa. 86400 IN NS tarzan.fi.upm.es.
8.100.138.in-addr.arpa. 86400 IN NS ns.fi.upm.es.
```

    - Problema: delegación y CIDR (no lo tratamos)

Ejemplo hipotético

- Empresa con sede central y tres departamentos
  - Un administrador gestiona sede central, dep1 y dep2
  - Dep3 con administrador propio (y servidor de correo propio)
- Detalles de servidores de nombres de cada dominio:
  - Sede central de la empresa (emp.es.): 2 s. de nombres
    - maestro en dominio (ns.emp.es.);esclavo externo (ns.isp.com.)
  - Dep1 (dep1.emp.es.): subdominio no delegado
  - Dep2 (dep2.emp.es.): subdominio delegado a mismos servidores
  - Dep3 (dep3.emp.es.): 3 s. de nombres
    - Maestro (ns1.dep3.emp.es.); esclavo interno (ns2) y externo (ns.isp.com.)
- Empresa tiene asignada red clase B 138.99.0.0
  - 138.99.1 central; 138.99.2 dep1; 138.99.3 dep2; 138.99.4 dep3
  - Sólo está delegado subdominio de 138.99.4

F. zona emp.es. (ns.emp.es.)

```
emp.es.                          IN  SOA  ns.emp.es ....
emp.es.          86400           IN  NS   ns.emp.es. ; servidor maestro del dominio
emp.es.          86400           IN  NS   ns.isp.com. ; servidor esclavo externo
; dep2 delegado a mismos servidores
dep2.emp.es.     86400           IN  NS   ns.emp.es. ; servidor maestro en el dominio padre
dep2.emp.es.     86400           IN  NS   ns.isp.com. ; servidor esclavo externo
; dep3 delegado a servidor maestro en el subdominio
dep3.emp.es.     86400           IN  NS   ns1.dep3.emp.es. ; servidor maestro en su propio subdominio
dep3.emp.es.     86400           IN  NS   ns2.dep3.emp.es. ; servidor esclavo en su propio subdominio
dep3.emp.es.     86400           IN  NS   ns.isp.com. ; servidor esclavo externo
emp.es.          86400           IN  MX   10 mail1.emp.es. ; servidor de correo preferente para la empresa
emp.es.          86400           IN  MX   20 mail2.emp.es. ; servidor de correo de reserva para la empresa
dep1.emp.es.     86400           IN  MX   10 mail1.emp.es. ; servidor de correo preferente para dep1
dep1.emp.es.     86400           IN  MX   20 mail2.emp.es. ; servidor de correo de reserva para dep1
; Máquinas en el dominio de la empresa
ns.emp.es.       86400           IN  A    138.99.1.1
mail1.emp.es.    86400           IN  A    138.99.1.2
mail2.emp.es.    86400           IN  A    138.99.1.3
www.emp.es.      86400           IN  A    138.99.1.4
www.dep1.emp.es. 86400           IN  A    138.99.2.1; RR de subdominio no delegado en misma zona
; Glue records para subdominio dep3
ns1.dep3.emp.es. 86400           IN  A    138.99.4.1
ns2.dep3.emp.es. 86400           IN  A    138.99.4.2
```


F. zona dep2.emp.es. (ns.emp.es.)

```
dep2.emp.es.                     IN  SOA  ns.emp.es ....
dep2.emp.es.     86400           IN  NS   ns.emp.es. ; servidor maestro del dominio (=padre)
dep2.emp.es.     86400           IN  NS   ns.isp.com. ; servidor esclavo externo
; Correo
dep2.emp.es.     86400           IN  MX   10 mail1.emp.es. ; s. correo preferente para dep2
dep2.emp.es.     86400           IN  MX   20 mail2.emp.es. ; s. correo de reserva para dep2
; Máquinas en el dominio de dep2
www.dep2.emp.es. 86400           IN  A    138.99.3.1
backup.dep2.emp.es. 86400        IN  CNAME www.dep2.emp.es.
```


F. zona dep3.emp.es. (ns.dep3.emp.es.)

```
dep3.emp.es.                     IN  SOA  ns1.dep3.emp.es ....
dep3.emp.es.     86400           IN  NS   ns1.dep3.emp.es.; servidor maestro del dominio (!=padre)
dep3.emp.es.     86400           IN  NS   ns2.dep3.emp.es.; servidor esclavo en el propio dominio
dep3.emp.es.     86400           IN  NS   ns.isp.com. ; servidor esclavo externo
; Correo
dep3.emp.es.     86400           IN  MX   10 mail.dep3.emp.es. ; s. correo preferente para dep3
dep3.emp.es.     86400           IN  MX   20 mail2.emp.es. ; s. correo de reserva para dep3
; Máquinas en el dominio de dep3
ns1.dep3.emp.es. 86400           IN  A    138.99.4.1
ns2.dep3.emp.es. 86400           IN  A    138.99.4.2
mail.dep3.emp.es. 86400          IN  A    138.99.4.3
www.dep3.emp.es. 120             IN  A    138.99.4.4; reparto de carga en servicio web
www.dep3.emp.es. 120             IN  A    138.99.4.5 ; reparto de carga en servicio web
```


F. zona 99.138. (ns.emp.es.)

```
99.138.in-addr.arpa.             IN  SOA  ns.emp.es ....
99.138.in-addr.arpa.  86400      IN  NS   ns.emp.es.
99.138.in-addr.arpa.  86400      IN  NS   ns.isp.com.
; dir. IP de dep3 delegadas a servidor maestro en el subdominio (no se necesitan glue records)
4.99.138.in-addr.arpa. 86400     IN  NS   ns1.dep3.emp.es.
4.99.138.in-addr.arpa. 86400     IN  NS   ns2.dep3.emp.es.
4.99.138.in-addr.arpa. 86400     IN  NS   ns.isp.com.
; Máquinas de sede central, dep1 y dep2
1.1.99.138.in-addr.arpa. 86400   IN  PTR  ns.emp.es.
2.1.99.138.in-addr.arpa. 86400   IN  PTR  mail1.emp.es.
3.1.99.138.in-addr.arpa. 86400   IN  PTR  mail2.emp.es.
4.1.99.138.in-addr.arpa. 86400   IN  PTR  www.emp.es.
1.2.99.138.in-addr.arpa. 86400   IN  PTR  www.dep1.emp.es.
1.3.99.138.in-addr.arpa. 86400   IN  PTR  www.dep2.emp.es.
```

F. zona 4.99.138. (ns.dep3.emp.es.)

```
4.99.138.in-addr.arpa.          IN  SOA   ns1.dep3.emp.es. ....
4.99.138.in-addr.arpa.  86400   IN  NS    ns1.dep3.emp.es.
4.99.138.in-addr.arpa.  86400   IN  NS    ns2.dep3.emp.es.
4.99.138.in-addr.arpa.  86400   IN  NS    ns.isp.com.
; Máquinas de dep3
1.4.99.138.in-addr.arpa.  86400  IN  PTR  ns1.dep3.emp.es.
2.4.99.138.in-addr.arpa.  86400  IN  PTR  ns2.dep3.emp.es.
3.4.99.138.in-addr.arpa.  86400  IN  PTR  mail.dep3.emp.es.
4.4.99.138.in-addr.arpa.  86400  IN  PTR  www.dep3.emp.es.
5.4.99.138.in-addr.arpa.  86400  IN  PTR  www.dep3.emp.es.
```


Glue records

- Posibles círculos viciosos en la traducción de nombres
- Si s. nombres de subdominio (Ssub) pertenece a subdominio
  - En el ejemplo: ns1.dep3.emp.es. y ns2.dep3.emp.es.
  - Para obtener IP de cualquier máq. subdominio → contactar con Ssub
    - ¿IP de www.dep3.emp.es? → contactar con ns1.dep3.emp.es.
  - Pero para hacerlo necesito IP de Ssub → contactar con Ssub
    - ¿IP de ns1.dep3.emp.es.? → contactar ns1.dep3.emp.es.
- *Glue record (GR)*
  - RR de tipo A/AAAA que se incluye en un dominio ajeno
  - Solución c. vicioso: padre debe incluir RR tipo A con dir. IP de Ssub
  - Aumenta problemas de coherencia
    - Cambios en IP de Ssub deben reflejarse también en dominio padre
  - No necesario *glue record* para servidor externo o en dominio padre
    - Siempre se puede obtener su traducción

Ejercicio: delegaciones en UPM

- *dig upm.es. SOA*

```
upm.es.  86400  IN  SOA  einstein.ccupm.upm.es. hostmaster.upm.es. 2013101401 86400 7200 1209600 3600
```

- *dig etsia.upm.es. SOA*

```
etsia.upm.es.  86400  IN  SOA  einstein.ccupm.upm.es. hostmaster.upm.es. 2011060701 86400 7200 1209600 7200
```

- *dig fi.upm.es. SOA*

```
fi.upm.es.  86400  IN  SOA  chita.fi.upm.es. hostmaster.fi.upm.es. 2013102101 28800 7200 2419200 3600
```

- *dig datsi.fi.upm.es. SOA*

```
datsi.fi.upm.es.  86400  IN  SOA  chita.fi.upm.es. hostmaster.fi.upm.es. 2012121801 28800 7200 2592000 3600
```

- *dig dlsiis.fi.upm.es. SOA*

```
No hay respuesta
```

- Analizar qué delegaciones existen
  - ¿dónde se necesitan *glue records*?

Servidores de nombres raíces

- Hay "13" servidores de dominio raíz (.) replicados
  - Desde *a.root-servers.net* hasta *m.root-servers.net*
  - "13" porque esa información cabe en paquete UDP
    - DNS usa UDP (53); y sólo TCP(53) cuando tamaño lo aconseja
  - ¿Problemas de escalabilidad?
    - Detrás de cada uno hay múltiples servidores (uso de *anycast*)
  - Incluyen NS y *glue records* de dominios de nivel 1º (TLDs)
    - Aunque también gestionan algunos dominios de primer nivel → *arpa*.
  - Cada serv. DNS tiene dir. de servidores raíz (fichero *root.servers*)
    - Se debe actualizar periódicamente
- Lista y localización: *http://root-servers.org*

Servidores DNS

- Servidor gestiona (*authoritative*) *N* (≥0) zonas directas/inversas
  - De algunas puede ser maestro de otras esclavo
- Servidor DNS puede tener doble rol (algo confuso):
  - Proporciona acceso a sus zonas
  - Puede actuar como cliente en navegación recursiva
- Servidor debe ofrecer navegación iterativa; recursiva opcional
  - Si no ofrece recursiva (no caché) → sólo acceso a sus zonas
  - Recursiva (caché)→por seguridad sólo peticiones de ciertas máquinas
- Ejemplos de diversos tipos de servidores de nombres:
  - Servidor sólo caché (*nonauthoritative*), recursivo para clientes internos
    - Sirve a clientes de una organización actuando de proxy
  - Servidor *authoritative* y recursivo sólo para clientes internos
    - Sirve a clientes de una organiz. y da acceso a sus zonas a todo el mundo
  - Servidor *authoritative* no recursivo → p.e. servidor raíz o de TLD

Resolver

- Parte cliente de DNS: da servicio a aplicaciones en un nodo
- Implementado habitualmente como biblioteca (⊂ en *libc*)
- Proporciona API para traducción directa e inversa:
  - UNIX: *gethostbyname/getaddrinfo* y *gethostbyaddr/getnameinfo*
- Configurado con servidores de nombres a los que consulta
  - Requiere también las direcciones IP de todos esos s. de nombres
  - En UNIX: */etc/resolv.conf*
- Esos servidores de nombres deben ser recursivos
  - *Resolver* no sabe navegar
- Puede usar caché (las aplicaciones también)
- UNIX permite configurar mecanismo de traducción de *hosts*
  - */etc/hosts*, DNS, NIS, LDAP (*/etc/nsswitch.conf*)

Resolución de consultas

- Servidor *S* recibe una consulta *C* de *N*:
  - Compara con RRs de todas sus zonas y de su caché (si usa)
  - Selecciona mejor encaje → RR (*RRX*) que sea sufijo más largo de *C*
  - Si encaje completo → envía a *N* consuelta resuelta
    - La marca como *authoritative* si no proviene de la caché
    - Si varios RRs satisfacen consulta, se envía a *N* lista con todos
      - Servidor rota la lista cada vez (*Round-robin DNS*) para reparto de carga
    - Se incluye información adicional para agilizar la operación
      - P.e. consulta MX puede retornar los RRs de tipo A de servidores de correo
  - Si encaje no completo, *RRX* → NSs de dominio por donde continuar
    - En el peor caso, los NSs de servidores raíz
    - Si op. recursiva: S envía consulta a uno de los NSs encontrados
    - Si op. no recursiva: *S* envía a *N* los RRs de los NSs encontrados
    - Si *S* conoce direcciones de NSs encontrados
      - Si no recursiva: las incluye como info. adicional en mens. de respuesta a *N*
      - Si recursiva: *S* las usa para contactar; sino tiene que obtenerlas

Ejemplos reales de traducción

- Operación de traducción directa: *www.fi.upm.es.*
- Operación de traducción inversa: *138.100.243.10.*
- *Resolver* tiene configurado como SN (3 opciones):
  - SN1: *ns.miempresa.com.*; IP 139.100.1.1
    - Zonas gestionadas: *miempresa.com.* y *100.139.in-addr.arpa., …*
  - SN2: *einstein.ccupm.upm.es.*; IP 138.100.4.8
    - Zonas gestionadas: *upm.es.* y *100.138.in-addr.arpa., …*
  - SN3: *zape.fi.upm.es.*; IP 138.100.8.1
    - Zonas gestionadas: *fi.upm.es.* y *243.100.138.in-addr.arpa., …*
- Supuestos:
  - traducción recursiva resolver-SN e iterativa desde SN
  - cachés vacías

Traducción directa usando SN1

- Aplicación llama a *gethostbyname("www.fi.upm.es.")* de *resolver*
- *Resolver* envía petición DNS de tipo A a dir. de SN1: 139.100.1.1
- SN1 mejor encaje: . → elige un s. raíz: *a.root-servers.net.* (198.41.0.4)
  - *a.root-servers.net.* mejor encaje: *es.* → envía a SN1 los NSs de *es.*
    - Y sus *glue records* como información adicional
- SN1 elige *a.nic.es.* (194.69.254.1)
  - *a.nic.es.* mejor encaje: *upm.es.* → envía a SN1 los NSs de *upm.es.*
    - Y sus *glue records* como información adicional
- SN1 elige *einstein.ccupm.upm.es.* (138.100.4.8)
  - *einstein.ccupm.upm.es.* mejor encaje: *fi.upm.es.*
    - envía a SN1 los NSs de *fi.upm.es.* y sus *glue records*
- SN1 elige *zape.fi.upm.es.* (138.100.8.1)
  - *zape.fi.upm.es.* Encaje completo: *www.fi.upm.es.*
    - envía a SN1 el NS de tipo A → *www.fi.upm.es.* | 138.100.243.10
      - SN1 se lo envía al *resolver* y éste retorna la IP a la aplicación

Traducción directa usando SN2 y SN3

- Aplicación llama a *gethostbyname("www.fi.upm.es.")* de *resolver*
- *Resolver* envía petición DNS de tipo A a dir. de SN2: 139.100.4.8
- SN2 mejor encaje: *fi.upm.es.* → elige *zape.fi.upm.es.* (138.100.8.1)
  - *zape.fi.upm.es.* encaje completo: *www.fi.upm.es.*
    - envía a SN2 el NS de tipo A → *www.fi.upm.es.* (138.100.243.10)
      - SN2 se lo envía al *resolver* y éste retorna la IP a la aplicación

---

- Aplicación llama a *gethostbyname("www.fi.upm.es.")* de *resolver*:
- *Resolver* envía petición DNS de tipo A a dir. de SN3: 138.100.8.1
- SN3 encaje completo: *www.fi.upm.es.* (138.100.243.10)
  - SN3 se lo envía al *resolver* y éste retorna la IP a la aplicación

Traducción inversa usando SN1

- Aplicación llama a *gethostbyaddr(...138.100.243.10...)* de *resolver*
- *Resolver* envía PTR *10.243.100.138.in-addr.arpa.* a SN1: 139.100.1.1
- SN1 mejor encaje: . → elige un s. raíz: *a.root-servers.net.* (198.41.0.4)
  - *a.root-servers.net.* mejor encaje: *in-addr.arpa.*
    - envía a SN1 los NSs de *in-addr.arpa.* y sus *glue records*
- SN1 elige *a.in-addr-servers.net.* (199.212.0.73)
  - *a.in-addr-servers.net.* mejor encaje: *138.in-addr.arpa.*
    - envía a SN1 los NSs de *138.n-addr.arpa.* (no sus *glue records*)
- SN1 elige *r.arin.net.* → tiene que hacer la traducción directa completa
  - No se muestra el detalle por ya conocido; Obtiene 199.180.180.63
  - *r.arin.net.* mejor encaje: *100.138.in-addr.arpa.*
    - envía a SN1 NSs de *100.138.n-addr.arpa.* (no sus *glue records*)

Traducción inversa usando SN1 (cont.)

- SN1 elige *einstein.ccupm.upm.es.* → necesita trad. directa completa
  - No se muestra el detalle por ya conocido; Obtiene 138.100.4.8
  - *einstein.ccupm.upm.es.* mejor encaje: *243.100.138.in-addr.arpa.*
    - envía a SN1 NSs de *243.100.138.in-addr.arpa.* y sus *glue records*
- SN1 elige *zape.fi.upm.es.* (138.100.8.1)
  - *zape.fi.upm.es.* encaje completo: *10.243.100.138.in-addr.arpa.*
    - envía a SN1 el NS de tipo PTR →
      - *10.243.100.138.in-addr.arpa* | *www.fi.upm.es.*
      - SN1 se lo envía al *resolver* y éste retorna nombre del *host* a la aplicación

Traducción inversa usando SN2 y SN3

- Aplicación llama a *gethostbyaddr(...138.100.243.10...)* de *resolver*
- *Resolver* envía PTR *10.243.100.138.in-addr.arpa.* a SN2: 139.100.4.8
- SN2 mejor encaje: *243.100.138.in-addr.arpa.* → elige *zape.fi.upm.es.*
  - *zape.fi.upm.es.* encaje completo: *10.243.100.138.in-addr.arpa.*
    - envía a SN2 el NS de tipo PTR →
      - *10.243.100.138.in-addr.arpa | www.fi.upm.es.*
      - SN2 se lo envía al *resolver* y éste retorna nombre del *host* a la aplicación

---

- Aplicación llama a *gethostbyaddr(...138.100.243.10...)* de *resolver*
- *Resolver* envía PTR *10.243.100.138.in-addr.arpa.* a SN3: 138.100.8.1
- SN3 encaje completo: *10.243.100.138.in-addr.arpa.*
  - *10.243.100.138.in-addr.arpa | www.fi.upm.es.*
  - SN3 se lo envía al *resolver* y éste retorna nombre *host* a la aplicación

Mantenimiento de info. de zona

- Sincronización de esclavo
  - Esclavo pide info. zona a maestro
    - Periódicamente (tal como lo especifica SOA)
    - O cuando maestro avisa de cambios (*NOTIFY*)
  - Si cambio: transferencia zona completa (*AXFR*) o incremental (*IXFR*)
  - Sólo se debe permitir transferencia de zona entre maestro y esclavos
- Actualización de DNS:
  - Cambio en fichero zona, incrementa nº en SOA y aviso a maestro
  - *Dynamic* DNS: Protocolo DNS incluye ops. para actualizar zona
    - Añadir, modificar y borrar RR pero no crear nuevas zonas
    - Mucho más flexible pero menos seguro
    - Algunas aplicaciones:
      - Permitir que máquinas mantengan mismo nombre en sistemas con DHCP
      - Servidor elige cualquier puerto y usa SRV (requerido por *Active Directory*)

Índice

Servicio de directorio

- Punto de acceso es sólo uno de los atributos de una entidad
  - Nombre impresora → modelo, color, ubicación, formatos soportados, ...
  - Pueden gestionarlos servidores específicos
    - Servicio de impresión gestiona información de impresoras
  - Problema: Duplicidad de funcionalidad
- Solución: Generalización del servicio de nombres
  - Nombre → conjunto de atributos de la entidad
- Aplicable a entidades de infraestructura del SD y de "negocio"
  - En FI: alumnos, profesores, títulos, asignaturas, dptos., servicios, ...
- Servicio de directorio (Sdir):
  - Repositorio de información de entidades de SD
    - Sun: "globalización" de */etc* → *Network Information System* (NIS)
  - No todo atributo ⊂ Sdir: no incluir atributos muy dinámicos
    - Tamaño cola de trabajos debería gestionarlo el servicio de impresión

Tipos de resolución de nombres

- Convencional (*Páginas blancas*): nombre → atributos
- Por atributos (*Páginas amarillas*): atributos → entidades
  - "Quiero imprimir fichero en impresora en color cerca de mi despacho"
- Parámetros típicos en resolución por atributos:
  - Nodo de inicio de búsqueda (base)
    - P.e. Sólo buscar entidades en determinada sucursal de la empresa
  - Profundidad de la búsqueda (sub-árbol, hijos directos, sólo base, ...)
  - Criterio/filtro de búsqueda:
    - Función lógica que deben satisfacer las entidades buscadas
    - P.e. Impresoras en color que estén ubicadas en el tercer piso
  - Límite de tiempo de búsqueda
  - Nº máximo de entidades que se retornarán
  - Atributos que se retornarán de las entidades seleccionadas

Tipos de entidades gestionadas

- ¿Qué tipos de entidades gestiona un servidor de nombres?
- Predeterminado:
  - Tipos de entidades predefinidas
- Configurable:
  - Existe un mecanismo para definir los tipos de las entidades
    - hay que definir: nombre del tipo, atribs., tipos de los atribs., etc.
  - Separación entre definición de tipos de entidades y de entidades
    - Similitud con base de datos: esquemas y datos
    - Similitud con POO: clases y objetos
  - Extensible: tipos predefinidos pero se pueden definir adicionales
    - Puede ser útil la herencia (simple o múltiple)

Serv. directorio vs. Base de datos

- Hay alguna similitud
  - Repositorio de información
  - Permite búsqueda sofisticada
- Pero muchas diferencias. Servicio de directorio:
  - Muchas consultas pero muy pocas modificaciones
  - Transacciones muy simples
  - Uso de esquemas estándar, siempre que sea posible
  - Más facilidad para cambiar esquemas para datos ya creados
    - P.e. Añadir a profesor FI asignaturas que imparte
  - Más adecuado para datos jerárquicos
  - Datos con múltiples valores para cada atributo
  - Si datos replicados, no requiere coherencia estricta
- Aunque muchos Sdir implementados con una base de datos

Lightweight Directory Access Protocol

- Precedente: X.500 servicio de directorio de ISO
  - Concebido para ser un directorio mundial
  - Complejo
  - Pesado: Ejecuta sobre la pila OSI
  - Protocolo de acceso DAP (*Directory Access Protocol*)
- LDAP (*Lightweight Directory Access Protocol*, RFC 4510)
  - Basado en X.500
  - Más sencillo
  - Más ligero: ejecuta sobre la pila TCP/IP
  - Es un protocolo pero define implícitamente un modelo de datos
    - No define aspectos de implementación
  - Distintos sistemas ofrecen una interfaz LDAP (p.e. *Active Directory*)
  - Actualmente versión 3

Objetos y clases

- Entidad → Objeto (entrada) en LDAP
  - Orientado a objetos: Objeto ∈ Clase (atributo *objectClass*)
- Clase define conjunto de atributos del objeto
  - Tipo del atributo | obligatorio(ob) u optativo(op) | valor único o múltiple
- Herencia: clases forman una jerarquía (*top* raíz de jerarquía)
  - Clase derivada hereda atributos de superclases
- Tipos de clases:
  - Abstracta (AB): no pueden definirse objetos de esa clase (p.e. *top*)
  - Estructural (ES): Objeto ∈ Una y solo una clase estructural
    - No puede cambiar la clase estructural de un objeto
  - Auxiliar (AU): Objeto puede estar asociado a varias clases auxiliares
    - Pueden añadirse dinámicamente: Facilitan extensión de objetos
  - Superclase(ES)=ES|AB; Superclase(AU)=AU|AB

Ejemplos de clases

- *top*: raíz; AB; ob: *objectClass*
- *person*: ↓*top;* ES; ob: *cn*, *sn*; op: *telephoneNumber*, ...
- *residentialPerson*: ↓*person;* ES; ob: *l;* op: *postalAddress*, ...
- *organization*: ↓*top;* ES; ob: *o;* op: *postalAddress*, ...
- *organizationalUnit*: ↓*top;* ES; ob: *ou;* op: *postalAddress*, ...
- *dcObject*: ↓*top;* AU; ob: *dc* (valor único)
- *device*: ↓*top;* ES; ob: *cn*; op: *serialNumber*, *o*, *ou*, *owner*, ...
- *groupOfNames*: ↓*top;* ES; ob: *cn*, *member*; op: *o*, *ou*, ...
- *alias*: ↓*top;* ES; ob: *aliasedObjectName*
- *referral*: ↓*top;* ES; ob: *ref*

Extracto de mi entrada en LDAP de FI

Formato de texto LDIF (*LDAP Data Interchange Format*): protocolo LDAP es binario

objectClass: inetOrgPerson ← estructural (*top*→*person*→*organizationalPerson*→*inetOrgPerson*)
objectClass: posixAccount ← auxiliar (*top*→ *posixAccount*)
objectClass: fiEmployee ← auxiliar (*top* → *irisPerson* → *fiPerson* → *fiEmployee*)
objectClass: sambaSamAccount ← auxiliar (*top* → *sambaSamAccount*)
cn: Fernando Perez Costoya
cn: F. P. Costoya
sn: Perez Costoya
telephoneNumber: 913367377
mail: fperez@fi.upm.es
uid: fperez
uidnumber: ....
gidNumber: ....
irisUserStatus: Activo
fiRelationShip: pdi
fiTeaching: .....
sambaSID: .....

person *organizationalPerson* *inetOrgPerson* *posixAccount* *irisPerson* *fiPerson* *fiEmployee* *sambaSamAccount*

Extracto de entrada FI en LDAP de FI

objectClass: dcObject ← auxiliar (*top* → *dcObject*)
objectClass: organization ← estructural (*top* → *organization*)
objectClass: labeledURIObject ← auxiliar (*top* → *labeledURIObject*)
dc: fi ← atributo específico de *dcObject*
o:: RmFjdWx0YWQgZGUgSW5mb3Jtw6F0aWNhIC0gVVBN
postalCode: 28660
l: Boadilla del Monte
st: Madrid
labeledURI: http://www.fi.upm.es ← atributo específico de *labeledURIObject*
telephoneNumber: +34 913367399

Decodificación de base 64
o: Facultad de Informática – UPM

Modelo de nombres

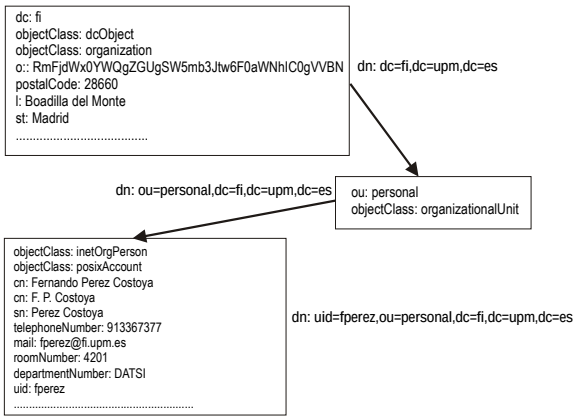
- Entrada tiene un nombre: *Relative Distinguished Name* (RDN)
  - 1 o más atributos de la entrada que la hacen única entre "hermanos"
    - *uid=fperez* (ej. múltiples*: cn=Fernando Perez Costoya+dni=76543210)*
- Jerarquía de nombres (*Directory Information Tree*, DIT)
  - Nombre completo (*path*): *Distinguished Name* (DN)
    - RDN de la entrada + DN del padre (separados por comas)
      - ***dn:*** *uid=fperez,ou=personal,dc=fi,dc=upm,dc=es*
    - No confundir con jerarquía de clases
    - Similar a SF pero directorios también tienen información asociada
  - Nombre del objeto raíz (sufijo o base): a discreción
    - Convenio: a partir de dominio DNS usando clase auxiliar *dcObject*
      - Dominio: *fi.upm.es* → ***dn:*** *dc=fi,dc=upm,dc=es*
  - Servidor LDAP gestiona 1 ó más DIT
  - Servidor devuelve metainformación en objetos/atrib. operacionales
    - DIT gestionados por el servidor, esquemas soportados, ...

Extracto de rama del DIT del LDAP de FI

```
# fi.upm.es
dn: dc=fi,dc=upm,dc=es
dc: fi
objectClass: dcObject
objectClass: organization
.....
# personal, fi.upm.es
dn: ou=personal,dc=fi,dc=upm,dc=es
ou: personal
objectClass: organizationalUnit

dn: uid=fperez,ou=personal,dc=fi,dc=upm,dc=es
uid: fperez
.....
```


Extracto de rama del DIT del LDAP de FI



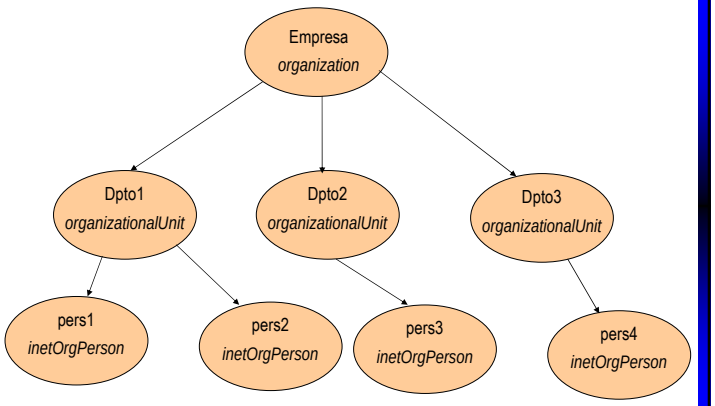
Distribución y replicación

- Espacio de nombres distribuido usando *referrals*
  - Objeto en DIT especifica punto de montaje
  - No definido el modelo de navegación
    - Implementación más habitual iterativa
    - Aunque también recursiva (*chaining*)
- Replicación de espacio de nombres no definida por estándar
  - OpenLDAP admite dos esquemas:
    - Maestro-esclavo: asimétrico
    - Multi-maestro: simétrico
  - OpenLDAP no garantiza coherencia

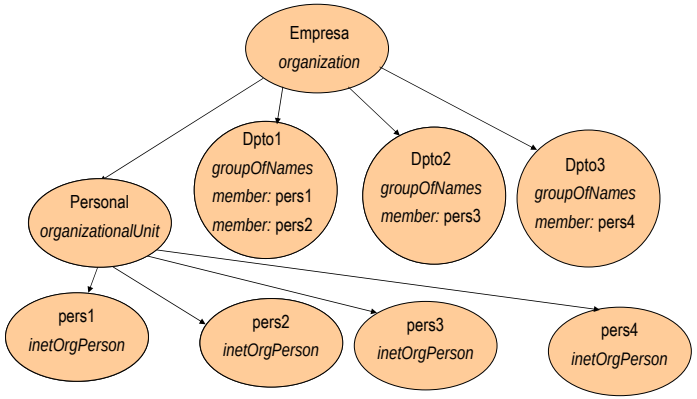
Diseño del DIT

- No trivial: requiere experiencia
- Análisis previo de info. del SD y cómo evolucionará
  - Diseño debería evitar que cambios previstos en info. modifiquen DIT
    - Cambio debería afectar a atributos en vez de a estructura de DIT
  - Mejor árbol poco profundo
- Ej.: empresa donde personal cambia de dpto. con frecuencia
  - Diseño 1
    - 1 *organizationalUnit*/dpto. + 1 *inetOrgPerson*/persona
    - Entrada de persona hija de entrada de su departamento
  - Diseño 2
    - 1 *organizationalUnit* para todo el personal + 1 *inetOrgPerson*/persona
    - 1 *groupOfNames*/dpto. con 1 atributo *member*/persona
    - Persona cambia de departamento: cambio atributos, no cambio DIT
      - Aunque ciertas búsquedas pueden ralentizarse

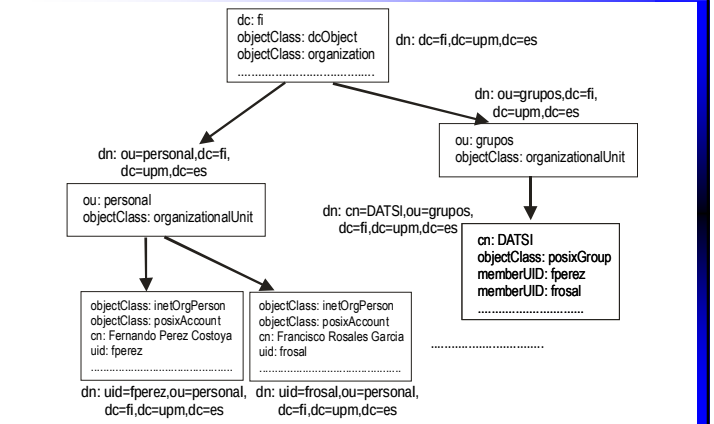
Diseño 1



Diseño 2



Extracto de jerarquía de LDAP de FI



Operaciones de LDAP

- *Bind/Unbind*: conecta y autentica/desconecta
- *Search*: realiza una búsqueda basada en los parámetros:
  - DN base de la búsqueda
  - Ámbito: Sólo la entrada base, sólo hijos o todo el sub-árbol
  - Filtro de búsqueda
  - Atributos que se devuelven (además, si valores o sólo tipos)
  - Si se siguen los alias o no durante la búsqueda
  - Límite de tiempo y máximo nº de entradas retornadas
- *Compare*: comprueba si DN dado tiene un valor en atributo
- *Add/Delete*: Añade/Elimina la entrada del DN dado
- *Modify*: Modifica atributos (añade, elimina o cambia) de un DN
- *Modify DN*: Cambia DN de una entrada
  - Renombra si sólo cambia RDN final; mueve en DIT en caso contrario

Acceso a operaciones de LDAP

- API de programación en C
  - *ldap_bind()*,*ldap_search()*,*ldap_add()*,*ldap_delete()*,*ldap_modify()*, ...
- Mandatos
  - *ldapsearch*, *ldapadd*, *ldapdelete*, *ldapmodify*, *ldapmodrdn*, ...
    - La mayoría usan el formato LDIF como entrada o salida
- Formato URL estándar para LDAP
  - *ldap://máquina:puerto/DNbase?atributos?ámbito?filtro*
    - *ldaps* si usa comunicación segura

Ejemplos de búsquedas (en triqui)

- Leer mi entrada

```
ldapsearch -x -W -H ldaps://info.fi.upm.es -D 'uid=fperez,ou=personal,dc=fi,dc=upm,dc=es'
    -b 'uid=fperez,ou=personal,dc=fi,dc=upm,dc=es'
```

- Nombre de profesores que comparten un despacho dado

```
ldapsearch  -x -W -H ldaps://info.fi.upm.es -D 'uid=fperez,ou=personal,dc=fi,dc=upm,dc=es'
    -b 'ou=personal,dc=fi,dc=upm,dc=es'  '(roomNumber=4201)' cn sn
```

- Nº tel. de personal de nombre Fernando y no sean del DATSI

```
ldapsearch  -x -W -H ldaps://info.fi.upm.es -D 'uid=fperez,ou=personal,dc=fi,dc=upm,dc=es'
    -b 'ou=personal,dc=fi,dc=upm,dc=es' '(&(!(departmentNumber=DATSI))(cn=*Fernando*))' cn
    telephoneNumber
```

- Nombre de “secciones” de la FI

```
ldapsearch -x -W -H ldaps://info.fi.upm.es -D 'uid=fperez,ou=personal,dc=fi,dc=upm,dc=es'
    -b 'dc=fi,dc=upm,dc=es' -s one '(objectClass=organizationalUnit)' ou
```


Esquema

- Paquete que incluye definiciones en ASN.1 y que usan OIDs
- Esquema incluye varios tipos de definiciones:
  - *ldapsyntax*: Define tipos básicos de LDAP
  - *matchingRule*: Op. de comparación sobre tipos básicos
  - *attributetype*: Definición de atributo
  - *objectclass*: Definición de clase
  - *matchingRuleUse*: Para qué atributo se usa una regla de comparación
  - *dITContentRule*: qué clases auxiliares permitidas para una c. estruct.
  - *dITStructureRule*: qué clases pueden ser padres de una c. estructural
  - *nameForm*: qué atributos pueden usarse como RDN de c. estructural
- Se usa herencia tanto en defs. de clases como de atributos
- Hay esquemas estandarizados:
  - *core*, *cosine*, *inetorgperson*, *nis*, ...

Sintaxis: tipos de datos de LDAP

- Definidos por estándar, por interoperabilidad no deberían definirse nuevos tipos

```
ldapsearch -H ldaps://info.fi.upm.es -x  -b cn=subschema -s base ldapsyntaxes

ldapSyntaxes: ( 1.3.6.1.4.1.1466.115.121.1.44 DESC 'Printable String' )
ldapSyntaxes: ( 1.3.6.1.4.1.1466.115.121.1.11 DESC 'Country String' )
ldapSyntaxes: ( 1.3.6.1.4.1.1466.115.121.1.26 DESC 'IA5 String' )
ldapSyntaxes: ( 1.3.6.1.4.1.1466.115.121.1.40 DESC 'Octet String' )
ldapSyntaxes: ( 1.3.6.1.4.1.1466.115.121.1.41 DESC 'Postal Address' )
ldapSyntaxes: ( 1.3.6.1.4.1.1466.115.121.1.50 DESC 'Telephone Number' )
ldapSyntaxes: ( 1.3.6.1.4.1.1466.115.121.1.36 DESC 'Numeric String' )
ldapSyntaxes: ( 1.3.6.1.4.1.1466.115.121.1.27 DESC 'Integer' )
ldapSyntaxes: ( 1.3.6.1.4.1.1466.115.121.1.24 DESC 'Generalized Time' )
ldapSyntaxes: ( 1.3.6.1.4.1.1466.115.121.1.7 DESC 'Boolean' )
ldapSyntaxes: ( 1.3.6.1.4.1.1466.115.121.1.6 DESC 'Bit String' )
.....
```


Reglas de comparación de tipos

- Definidas por estándar, por interoperabilidad no deberían definirse nuevas reglas

```
ldapsearch -H ldaps://info.fi.upm.es -x  -b cn=subschema -s base matchingRules

matchingRules: ( 2.5.13.4 NAME 'caseIgnoreSubstringsMatch' SYNTAX
    1.3.6.1.4.1.1466.115.121.1.58 )
matchingRules: ( 2.5.13.2 NAME 'caseIgnoreMatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
    )
matchingRules: ( 1.3.6.1.4.1.1466.109.114.3 NAME 'caseIgnoreIA5SubstringsMatch'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26)
matchingRules: ( 1.3.6.1.4.1.1466.109.114.2 NAME 'caseIgnoreIA5Match' SYNTAX
    1.3.6.1.4.1.1466.115.121.1.26 )
matchingRules: ( 2.5.13.14 NAME 'integerMatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 )
matchingRules: ( 2.5.13.13 NAME 'booleanMatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 )
.....
```


Definición de atributos

```
ldapsearch -H ldaps://info.fi.upm.es -x  -b cn=subschema -s base attributetypes

attributetype ( 2.5.4.41 NAME 'name'
    EQUALITY caseIgnoreMatch
    SUBSTR caseIgnoreSubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{32768} )

attributetype ( 2.5.4.3 NAME ( 'cn' 'commonName' ) SUP name )

attributetype ( 0.9.2342.19200300.100.1.25
    NAME ( 'dc' 'domainComponent' )
    DESC 'RFC1274/2247: domain component'
    EQUALITY caseIgnoreIA5Match
    SUBSTR caseIgnoreIA5SubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE )
```


Definición de clases

```
ldapsearch -H ldaps://info.fi.upm.es -x  -b cn=subschema -s base objectClasses

objectclass ( 2.5.6.0 NAME 'top' ABSTRACT MUST objectClass )

objectclass ( 2.5.6.6 NAME 'person' SUP top STRUCTURAL
    MUST ( sn $ cn )
    MAY ( userPassword $ telephoneNumber $ seeAlso $ description ) )

objectclass ( 2.5.6.7 NAME 'organizationalPerson' SUP person STRUCTURAL
    MAY ( title $ x121Address $ registeredAddress $ destinationIndicator $
      preferredDeliveryMethod $ telexNumber $ teletexTerminalIdentifier $
      telephoneNumber $ internationaliSDNNumber $
      facsimileTelephoneNumber $ street $ postOfficeBox $ postalCode $
      postalAddress $ physicalDeliveryOfficeName $ ou $ st $ l ) )

objectclass ( 1.3.6.1.4.1.1466.344 NAME 'dcObject'
    DESC 'RFC2247: domain component object'
    SUP top AUXILIARY MUST dc )
```

Uso de reglas de comparación

ldapsearch -H ldaps://info.fi.upm.es -x -b cn=subschema -s base matchingRulesUse

matchingRuleUse: (1.3.6.1.4.1.1466.109.114.2 NAME 'caseIgnoreIA5Match' APPLIES (altServer $ mail $ dc $ associatedDomain $ email $ aRecord $ mDRecord $ mXRecord $ nSRecord $ sOARecord $ cNAMERecord $ janetMailbox $ gecos $ homeDir....))

matchingRuleUse: (2.5.13.13 NAME 'booleanMatch' APPLIES (hasSubordinates $ olcGentleHUP $ olcLastMod $ olcReadOnly $ olcReverseLookup $ olcDbNoSync $ olcDbDirtyRead $ olcDbLinearIndex $ olcChainCacheURI $ olcChainReturnError $ olcDbRebindAsUser $ olcDbChaseReferrals $ olcDbProxyWhoAmI $ olcDbSingleConn $ olcDbUseTemporaryConn $ pwdLockout $ pwdMustChange $ pwdAllowUserChange $ pwdSafeModify $ sambaBoolOption $ pwdReset $ olcPPolicyHashCleartext $ olcPPolicyUseLockout $ olcSpNoPresent $ olcSpReloadHint))

Creación de un nuevo esquema

- Sólo si es estrictamente necesario
  - Nunca cambiar comportamiento de objetos/atrib. estándar
- 2 alternativas para extender clase ya existente
  - Crear nueva clase estructural derivada de clase existente
    - Permite mejor control: se pueden definir reglas de contenido/estructura
    - Pero requiere eliminar y reinsertar todos los objetos existentes
  - Crear clase auxiliar derivada de *top* e incluirla en definición de objetos
    - Se puede añadir directamente usando *Modify*
- C. auxiliar también permite incluir atrib. en objetos de ≠ clases
  - p.e. fecha de alta en organización, tanto personas como dispositivos

Extracto de esquema del LDAP de FI

attributetype (1.3.6.1.4.1.7547.1.19.10.4.2.4 NAME 'fiRelationShip' DESC 'Relacion del usuario con la Escuela' EQUALITY caseIgnoreMatch SUBSTR caseIgnoreSubstringsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype (1.3.6.1.4.1.7547.1.19.10.4.2.1 NAME 'fiGender' DESC 'Sexo de la persona (ISO 5218)' EQUALITY integerMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE)

attributeTypes: (1.3.6.1.4.1.7547.1.19.10.4.2.5 NAME 'fiTeaching' DESC 'Asignaturas impartidas por el profesor' EQUALITY caseIgnoreMatch SUBSTR caseIgnoreSubstringsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{20})

objectclass (1.3.6.1.4.1.7547.4.3.1.2 NAME 'irisPerson' DESC 'Persons inside the IRIS community' SUP top AUXILIARY MAY (sn1 $ sn2 $ irisPersonalTitle $ irisPersonalUniqueID $ irisUserEntitlement $ irisUserPrivateAttribute $ irisUserStatus $ irisMailHost $ irisMailRoutingAddress $ irisMailbox $ irisMailMainAddress $ irisMailAlternateAddress $ irisUserPresenceID $ irisClassifCode))

objectclass (1.3.6.1.4.1.7547.1.19.10.4.1.1 NAME 'fiPerson' DESC 'Persona perteneciente a la Facultad de Informatica (UPM)' SUP irisPerson AUXILIARY MUST (uid $ mail) MAY (fiPwdChangedOperTime $ fiMailQuotaSize $ fiGender $fiRelationShip))

objectClasses: (1.3.6.1.4.1.7547.1.19.10.4.1.3 NAME 'fiEmployee' DESC 'Empleado de la Facultad de Informatica (UPM)' SUP fiPerson AUXILIARY MAY fiTeaching)

Modelo de seguridad

- 3 métodos de autenticación
  - Sin autenticación: se considera usuario anónimo
  - Autenticación básica: DN del usuario + contraseña
  - *Simple Authentication and Security Layer* (SASL)
    - Entorno genérico de autenticación y seguridad de datos
    - Permite usar múltiples mecanismos (p.e. SASL DIGEST-MD5)
    - SASL EXTERNAL: protocolo nivel inferior proporciona autenticación
      - Como cuando se usa *Transport Layer Security* (TLS)
- Protección de entradas no definida por el estándar
  - Habitualmente se usan listas de control de acceso (ACL)
    - Controlan acceso a cada atributo de una entrada

Índice

- Introducción
- Servicio de nombres
  - Estudio de un ejemplo práctico: DNS
- Servicio de directorio
  - Estudio de un ejemplo práctico: LDAP
- Descubrimiento de servicios
  - Gestión de nombres en sistemas móviles/ubicuos
  - Auto-configuración
  - Servicios de descubrimiento de servicios

Gestión de nombres en SD convencional

- Estructura del SD bastante estática
- Cada nodo se configura con (suponiendo uso de IP)
  - Su dir. IP, máscara de red, dir. *router*, e info. de encaminamiento
  - Su nombre, dominio DNS al que pertenece y direc. servidores DNS
  - Nombre servidor(es) LDAP y conocimiento del esquema usado
  - Los manejadores requeridos para interacción con dispositivos en SD
- Incluso en SD convencional, op. configuración no escalable
  - Uso de DHCP (*Dynamic Host Configuration Protocol*)
- Necesidad limitada de "descubrimiento" de servicios/dispos.
  - Una vez instalada nueva impresora se la da de alta en LDAP
  - Próxima búsqueda de impresoras en LDAP la encontrará
  - Dar de baja impresora (poco frecuente): basta con actualizar LDAP
  - No se requiere "*Plug* & *Play*" en el nivel del SD

Gestión de nombres en SD móvil/ubicuo

- Computación ubicua → "invisible" → auto configuración
- Sistemas dinámicos, "espontáneos" y volátiles:
  - Nodos entran y salen de un SD: de un "espacio inteligente" (EI)
    - Mi cámara digital y yo entramos/salimos en habitación de hotel
    - Un vehículo entra/sale de EI controlado por un semáforo inteligente
- Descubrimiento de servicios clave para computación ubicua
- Nodo entra en EI:
  - Se autoconfigura y descubre, y es descubierto, por nodos restantes
  - Si proveedor de servicios, hace conocerlos a quiénes le interesen
  - Si consumidor de serv., descubre los de otros nodos que le interesen
  - Necesidad de "lenguaje" de definición y búsqueda de servicios
    - Atributos-valores (similar a LDAP), basado en XML, ontologías, ...
    - Suficiente flexibilidad para incorporar nuevos tipos de serv./dispos.

Gestión de nombres en SD móvil/ubicuo

- *Plug & play* de servicios/dispositivos en SD
  - Además de descubrirlos, hay que saber "hablar" con ellos
  - Nuevo tipo puede requerir nuevo manejador
- Nodo abandona EI: servicio/dispositivo desaparece
  - Abandono abrupto → uso de *leases*
- Problema de frontera del espacio inteligente:
  - Delimitación precisa de confines de un espacio inteligente
    - ¡Espero que mis fotos no se impriman en habitación contigua!
  - Necesidad de crear ámbitos (*scopes*)
- Limitación de recursos y volatilidad pueden condicionar:
  - Estrategias de auto-configuración y descubrimiento de servicios
- Ej. Jini, UPnP, Zeroconf, *Service Location Protocol* (RFC 2608)

Ejemplo plantilla de servicio de SLP

```
service:printer://lj4050.tum.de:1020/queue1
scopes = tum, bmw, administrator
printer-name = lj4050
printer-model = HP LJ4050 N
printer-location = Room 0409
color-supported = false
pages-per-minute = 9
sides-supported = one-sided, two-sided
```

Extraído de "*A Comparison Of Service Discovery Protocols And Implementation Of The Service Location Protocol*", Christian Bettstetter y Christoph Renner

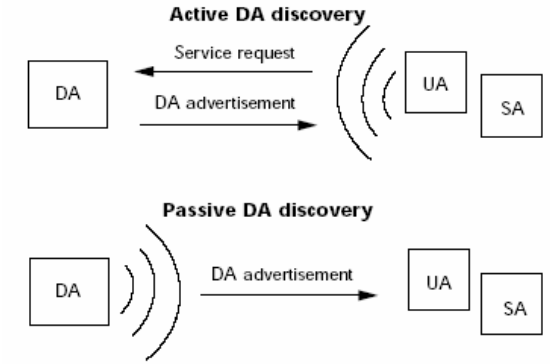
Auto-configuración

- Obtención de dirección IP (e info asociada: máscara, *router*,...)
  - Uso de DHCP:
    - Nodo *broadcast* petición de dirección IP
    - Servidor DHCP asigna dirección IP con *lease* asociado
  - Si DHCP no disponible (por volatilidad o limitación de recursos)
    - *Dynamic Configuration of IPv4 Link-Local Addresses* (RFC 3927)
    - Nodo elige su dir. IP y usa ARP para comprobar que no está en uso
    - Si conflicto, selecciona otra
- Obtención de nombre DNS (si requerido)
  - Uso de DNS con protocolo de actualización: *Dynamic-DNS*
  - Si DNS no disponible (por volatilidad o limit. recursos): *Multicast*-DNS
    - Consultas a dominio `.local.` usan *multicast* dir. fija
    - Nodo correspondiente responde (con *unicast* o *multicast*)

Servicios de descubrimiento de servicios

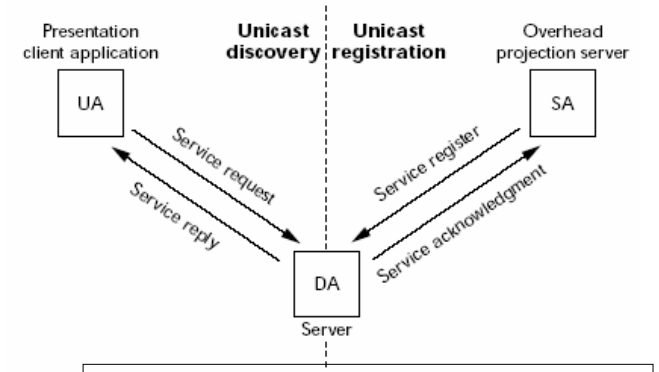
- Tres roles (usando terminología SLP):
  - cliente (UA), proveedor servicio (SA), servidor descubrimiento (DA)
- Alternativa principal: con o sin DA
  - Puede haber múltiples DA: replicación y/o info. de distintos ámbitos
- Con DA (Jini, UPnP):
  - UA y SA deben localizar DAs (posible filtro por ámbitos)
    - Localización activa: UA/SA *multicast* a dirección fija
    - Localización pasiva: DA *multicast* a dirección fija
      - Facilita incorporación de nuevos DAs al sistema
      - SAs pueden registrar servicios y UAs realizar consultas
  - SA registra servicio mediante *unicast* en DAs localizados
  - UA consulta mediante *unicast* a alguno de los DAs localizados
  - UA puede pedir a DA notificación si aparece un tipo de SA → evento

Descubrimiento de DA en SLP



Service Location Protocol: Automatic Discovery of IP Network Services. Erik Guttman

Registro y búsqueda de servicio en SLP



Service Location Protocol: Automatic Discovery of IP Network Services. Erik Guttman

Servicios de descubrimiento de servicios

- Sin DA: *pull* versus *push*
  - *Pull*: UA *multicast* petición; SA la recibe y responde
  - *Push*: SA *multicast* anuncio de servicio; UAs guardan esa info.
  - *push* descubrimiento automático nuevo SA; *pull* uso de *polling*
- Esquema híbrido (SLP)
  - Mientras no haya ningún DA (suponiendo modelo *pull* como SLP):
    - SA escucha dir. *multicast* peticiones de servicio
    - UA envía a dir. *multicast* peticiones de servicio
    - SAs/UAs escuchando dir. *multicast* posibles altas de DAs
  - Cuando aparece un DA no habiendo ninguno antes
    - SAs registra servicio en DA mediante *unicast*
    - UAs consultan DA usando *unicast*
  - Si desaparecen todos los DAs: vuelta al primer punto

Zeroconf (Bonjour, Avahi)

- Configuración automática máquinas/servicios en red IP
- Selección de dirección IP
  - DHCP y, si no disponible, direcciones *Link-Local*
- Configuración nombres de máquina
  - DNS convencional y, si no disponible, *Multicast*-DNS
- Descubrimiento de servicios:
  - Usar como DA → DNS-SD: una extensión de DNS
  - Usa SRV, PTR y TXT para especificación de servicios
  - Añade *leases* para detectar servidor ya no presente
  - y *long-lived queries* para solicitar notificación si aparece nuevo servicio