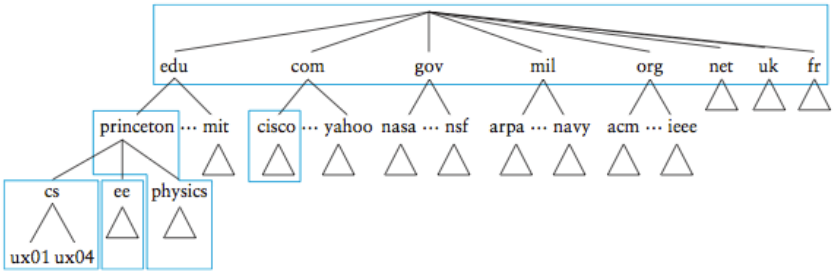
Unidad 9: Capa de Aplicación

DNS

El servidor de nombres es uno de los primeros servicios implementados para permitir que el resto se desentienda de las direcciones de red. En el caso de internet se usa un sistema distribuído llamado **DNS** (Domain Name System).

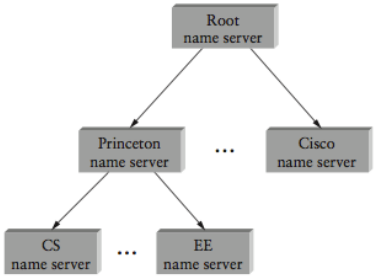
Características del **namespace**:

- Es jerárquico, procesado de derecha a izquierda
- Se puede pensar como un árbol, donde cada nodo es un **dominio**
- Un dominio es un contexto en el cuál se pueden definir más nombres



La estructura del namespace se mantiene de forma distribuída del siguiente modo:

- Jerarquía particionada en **zonas**
- Cada zona puede pertenecer a unidades administrativas independientes
- Las zonas son las unidades básicas de implementación: cada una tiene asignada dos o más **name servers** encargados de resolver los nombres de la misma.
- Cuando un host realiza un pedido a DNS, el sevidor puede dar una respuesta o derivar en otro servidor que tenga más información de la zona.
- Hay siempre más de un name server por zona para lograr redundancia.
- Un name server puede estar asignado a más de una zona.



Cada servidor contiene entradas de tipo **<Name, Value, Type, Class, TTL>**:

- **Name**: clave de la query

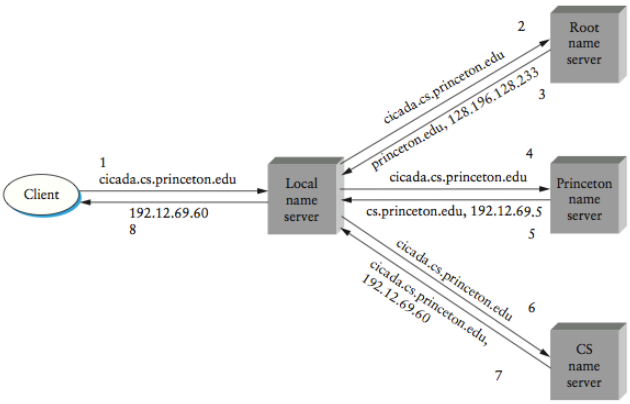
- **Value**: valor asociado
- **Type**: qué es el valor
- **Class**: punto de extensión
- **TTL**: cuánto falta hasta que expire el registro en servidores que lo tienen cacheado

El campo **type** indica cómo se debe interpretar el valor. Ejemplos:

- **A**: el valor es la dirección IP buscada
- **NS**: el valor es el domain name de un name server que puede conocer la respuesta
- **CNAME**: el valor es un alias del dominio buscado. se utiliza para agregar un nivel de indirección.
- **MX**: el valor es el domain name de un servidor que acepta mails a ese dominio

Configuración:

- Cada host debe conocer la dirección de un name server a utilizar.
- En la práctica, se conoce la dirección de un name server local que a su vez conoce algún root name server.
- Esto permite que el servidor cachee respuestas y evita configurar manualmente el root server en todos los hosts.



Email

Formato

RFC 822:

- Header (*From:*, *To:*, *Subject:*, etc.)
- Body
- Todo ASCII

MIME:

- Extensión para poder enviar datos
- Header lines adicionales (*MIME-Version:*, *Content-Description*, *Content-Type:*, *Content-Transfer-Encoding:*)
- Definición de content types estándar (ejemplo: *image/jpeg*, *text/plain*, *application/msword*)
- Definición de formas de encodear esos contenidos en ASCII (base64)
- Definición de content type **multitype**, que define estructura de mensajes que llevan varios content

types.

Por ejemplo, para enviar texto con archivos adjuntos, se utilizaría el content type *multipart/mixed* definido por MIME, y cada una de estas partes tendría sus propios headers que definen en tipo y encoding.

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="-------417CA6E2DE4ABCAFBC5"
From: Alice Smith <Alice@cisco.com>
To: Bob@cs.Princeton.edu
Subject: promised material
Date: Mon, 07 Sep 1998 19:45:19 -0400

---------417CA6E2DE4ABCAFBC5
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

Bob,

Here's the jpeg image and draft report I promised.

--Alice

---------417CA6E2DE4ABCAFBC5
Content-Type: image/jpeg
Content-Transfer-Encoding: base64

. . . unreadable encoding of a jpeg figure

---------417CA6E2DE4ABCAFBC5
Content-Type: application/postscript; name="draft.ps"
Content-Transfer-Encoding: 7bit

. . . readable encoding of a PostScript document
```

Transferencia

SMTP:

- Protocolo utilizado entre los **mail daemons** de cada host.
- Establece una sesión en donde se intercambian mensajes.
- Los mensajes de control y contenido de los correos son en ASCII (los correos respetando el formato de mensajes de RFC 822)
- El servidor avisa si recibe correo para cada uno de los destinatarios.

- Los mensajes atravesan un camino de **mail gateways**, que hacen store-and-forward de los mensajes.

- A diferencia de los gateways IP (routers), los mail gateways guardan en disco los mensajes y pueden llegar a reintentar el envío durante días.
- Los gateways sirven para esconder el host final de donde se lee el mail (enviando a un gateway "institucional") y para almacenar los mensajes en caso de que el host final no esté online.

Lectura

Podrían leerse los correos con un programa corriendo localmente en el mismo equipo de la casilla de mail, pero generalmente se accede remotamente desde otro equipo. Para esto se utilizan típicamente dos protocolos:

- POP: Descarga los mensajes y se maneja una copia de los datos de la casilla.
- IMAP: Se maneja la casilla real localmente.

Web

HTTP

- Protocolo de comunicación entre clientes y servidores
- Stateless
- El cliente envía requests y el servidor responde

Los requests incluyen:

- Start line: definen la operación a realizar, el recurso solicitado y versión del protocolo
- Headers
- Body (suele ser vacío)

Las respuestas incluyen:

- Start line: versión del protocolo y status code
- Headers
- Body

Operation	Description
OPTIONS	request information about available options
GET	retrieve document identified in URL
HEAD	retrieve metainformation about document identified in URL
POST	give information (e.g., annotation) to server
PUT	store document under specified URL
DELETE	delete specified URL
TRACE	loopback request message
CONNECT	for use by proxies

Mejoras de HTTP 1.1

A partir de HTTP 1.1 se soportan **conexiones persistentes**, que permiten utilizar una conexión para varios ciclos de request/response (y además mejoran el funcionamiento de los mecanismos de control de congestión de TCP).

Además de las conexiones persistentes, se puede utilizar **request pipelining** para enviar varios request antes de que llegue el primer response.

Caching

En la web se suelen utilizar caches a distintos niveles (host, ISP, servers) para mejorar la performance y reducir la carga. HTTP define headers especiales para ayudar a controlar caches, evitando transmitir recursos si el cliente ya posee la última versión.

Por ejemplo: el header *Expires:* indica al cliente por cuánto tiempo puede guardar un recurso sin pedirlo nuevamente, y también se pueden realizar "gets condicionales" utilizando el header *If-Modified-Since:*