High Performance Computing and Networking

Arquitectura de Redes 1:

Práctica 2: Cliente DNS

José Luis García Dorado

Javier Ramos

Práctica 2

- Inicio:
  - 28/29 octubre.
- Entrega:
  - antes del 25 (L) / 26 (M) de noviembre a las 00:00
- Leer enunciado en la web y seguir instrucciones de “¿Cómo empezar?”.
- Objetivo: implementación de un cliente sencillo DNS.

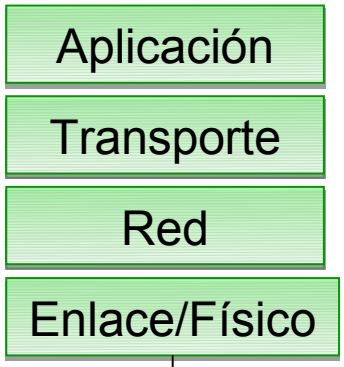
Contexto

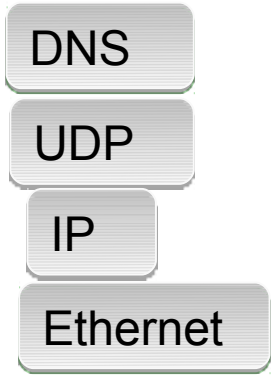
NSLOOKUP

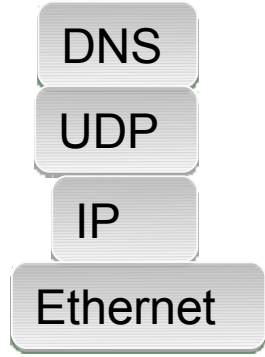
10.0.0.2

Servidor DNS

10.1.0.3





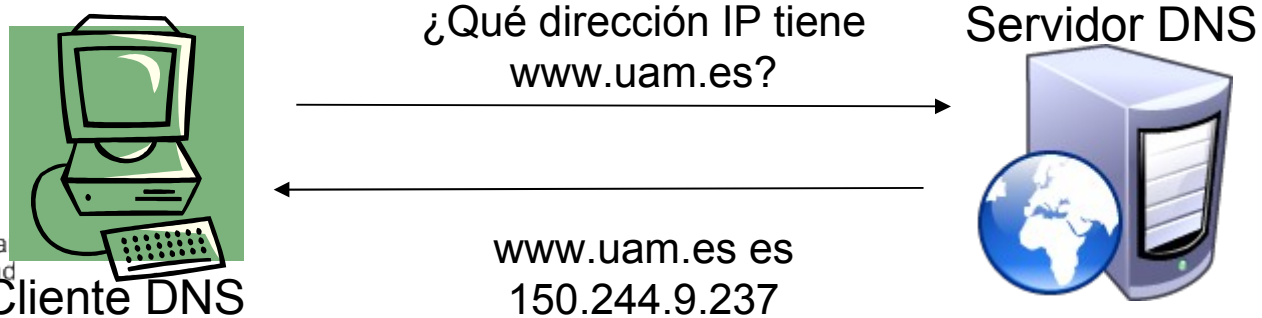


(1)
¿Qué IP es
www.uam.es?

(2)
www.uam.es
es 150.244.9.237

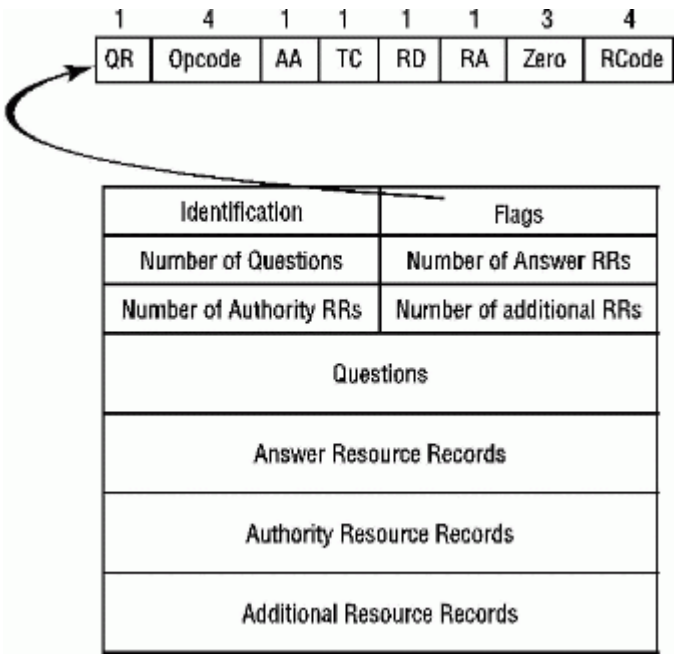
Fundamentos protocolo DNS

- RFC 1035: http://www.faqs.org/rfcs/rfc1035.html
- Funciona sobre UDP puerto 53.
- Objetivo: traducir nombres de máquinas a IPs.
- Modelo petición-respuesta:
  - Cliente envía petición de la traducción de un nombre de recurso (una máquina o un servidor de correo).
  - El servidor consulta su base de datos y devuelve la dirección del recurso.



Estructura paquete DNS

- Ver ejemplo y detalles con Wireshark (secc. ¿Cómo empezar del enunciado?).
- ID: identificador (arbitrario)
  - Correlar peticiones y respuestas.
- Flags:
  - QR= 0 petición, 1 respuesta
  - OPCODE=0000 petición
  - RD=1
  - Resto= 0



- OJO!: orden de red para campos “Número de…”

Estructura paquete DNS

- Estructura Question:
  - Query name: Longitud variable
    - Codificacion nombres: los puntos son eliminados y delante de cada “trozo” se coloca su longitud. Se concluye con 0.
    - Ej: www.google.es => 3www6google2es0
  - Tipo: longitud 16 bits
    - 1(A) para direcciones IP (caso petición)
  - Clase: longitud 16 bits
    - 1 (IN) Internet
  - OJO!: Orden de red para campos Tipo y Clase.

Estructura paquete DNS

- Estructura Response:
  - Name: Longitud variable
    - Formato como query name o con compresión (más adelante) cuando la haya.
  - Rdata Length: 16 bits (orden de red!)
    - Longitud del campo Rdata
  - Rdata: longitud dada por el campo anterior.
    - Dato de la respuesta en sí. (IP o nombre canónico)
  - Tipo: longitud 16 bits
    - 1(A) direcciones IP
    - 5(CNAME) Nombres Canónicos
  - Clase: longitud 16 bits
    - 1 (IN) Internet
  - TTL: 32 bits
    - No importante en nuestro caso.

Compresión de nombres

- Para no “malgastar” bytes en escribir nombres que ya están en otra parte del paquete, se utiliza un método de compresión basado en punteros.
- Si el primer byte es 0xC0, el siguiente byte te dice la posición del paquete donde continúa el nombre.
- Puede aparecer a la mitad de un nombre.
- Puede aparecer de manera recursiva. Esto es, dar varios saltos para construir un solo nombre.
- Ver ejemplo con la ayuda de Wireshark.

Objetivos de la práctica

- Implementar un cliente DNS (similar a nslookup) que permita hacer resoluciones directas (de nombres a direcciones IP) .

- Formato:
  - Para peticiones directas:
    - ./nslookup nombre_a_resolver
    - Ej: ./nslookup www.uam.es

Criterios de evaluación

- Petición directa simple (e.g. www.uam.es): 6 pt
- Petición directa múltiple (e.g. www.as.com): 4 pt

Consejos

- Seguir los pasos de “¿Cómo empezar?”:
  - Hacerlo **ahora** en la primera clase para preguntar las dudas que surjan!
  - Antes de ponernos a escribir código, entender muy bien qué tenemos que hacer (cómo funciona el protocolo DNS, cómo son las peticiones, cómo son las respuestas…) y pensar cómo vamos a hacerlo.
- Usar Wireshark para ver si estamos construyendo bien las peticiones y cómo vienen las respuestas.

Sugerencia de implementación

- Una función que dado un nombre de dominio lo cambie al formato de petición de DNS
  - Por ejemplo www.ii.uam.es → 3www2ii3uam2es0
- Una función que haga lo inverso de lo anterior
- Una función que cree la query DNS
  - Rellenar campos de cabecera
  - Crear petición
  - Devolver un array con todos los bytes necesarios para enviar
- Un función que analice las peticiones (Útil cuando en las respuestas devuelve la petición que se ha realizado)
- Una función que analice las respuestas
  - Obtener los recursos tipo A, CNAME, PTR, MX y en función de cada tipo decidir como rellenar la estructura hostent
- Una función que analice las autoridades
  - Obtener los recursos tipo A y NS y rellenar como corresponda la estructura hostent
- Una función que analice los registros adicionales
  - Obtener los recursos tipo A y CNAME y rellenar como corresponda la estructura hostent

Consejos

- Funciones útiles:
  - htons/ntohs:
    - Los números enteros de 16 (o más) bits están guardados de manera distinta en la máquina que como son mandados por la red.
    - htons útil al escribir el paquete de petición (pasa de formato host a formato red).
    - ntohs útil al leer el paquete de respuesta (pasa de formato de red a formato host).
    - Ejemplo: queremos escribir que el número de preguntas es 1
      - `uint16_t nquestions=htons(1);`
      - `memcpy(buffer+4,&nquestions,sizeof(u_int16_t))`