1. DNAME resource record
2. DNS on the wire
3. Tea break
4. Zone transfers
5. Encoding of domain names
6. Limitations and extras
Outline

1. DNAME resource record
2. DNS on the wire
3. Tea break
4. Zone transfers
5. Encoding of domain names
6. Limitations and extras
**DNAME workings**

- **DNAME** provides aliasing at all levels below the owner-name
  - **CNAME** $\approx$ symlink for a *file* on a filesystem
  - **DNAME** $\approx$ symlink for a *directory* on a filesystem

- **test.os3.nl.** IN **DNAME** **os3.nl.**

- **DNAME** added later, so not supported everywhere

- **Name Server** also synthesizes **CNAME** record itself
DNAME example

- `test.os3.nl. IN DNAME os3.nl.`

  Question: A record for `www.test.os3.nl.`

  Answer:

  - `test.os3.nl. IN DNAME os3.nl.`
  - `www.os3.nl. IN A 145.100.96.70`
Outline

1. DNAME resource record
2. DNS on the wire
3. Tea break
4. Zone transfers
5. Encoding of domain names
6. Limitations and extras
Wire?

- Not the 1980s punk band ;-) 
- Wire == Network
- Queries and Responses are packaged in packets
- Packets are transferred over the wire
  - OSI Layer 2: does not matter
  - OSI Layer 3: IPv4 or IPv6
  - OSI Layer 4: UDP or TCP
How to package DNS messages

1. Define what information you want to exchange
2. Specify a format in which to encode that information
   - Serialization, “flattening” data structures
3. Implement that format in software
4. Start doing DNS :)
How to tap the wire?

1. **Wiretapping == Copying digital information during transport**

2. **Command line: tcpdump**
   - “tcpdump -i eth0” for real-time wiretapping
   - tcpdump can also write data to a file
   - PCAP (packet capture) format is used to store data

3. **GUI: Wireshark**
   - Provides a GUI to “inflate” serialized data
   - Wireshark can wiretap and process data in real-time
   - Wireshark can also read from PCAP files
DNS Message packet format

- Header section
- Question section
- Answer section
- Authority section
- Additional section
### DNS packet header

<table>
<thead>
<tr>
<th>ID</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>QDCOUNT</td>
<td>ANCOUNT</td>
</tr>
<tr>
<td>NSCOUNT</td>
<td>ARCOUNT</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>ID</td>
<td>Transaction Identifier</td>
</tr>
<tr>
<td>Flags</td>
<td>See next slide</td>
</tr>
<tr>
<td>QDCOUNT</td>
<td>Number of questions</td>
</tr>
<tr>
<td>ANCOUNT</td>
<td>Number of answers</td>
</tr>
<tr>
<td>NSCOUNT</td>
<td>Number of authority records</td>
</tr>
<tr>
<td>ARCOUNT</td>
<td>Number of additional records</td>
</tr>
</tbody>
</table>
# DNS header flags

<table>
<thead>
<tr>
<th>Bit(s)</th>
<th>Mnemonic</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>QR</td>
<td>Query(0) or Response(1)</td>
</tr>
<tr>
<td>1-4</td>
<td>OPCODE</td>
<td>Kind of Query</td>
</tr>
<tr>
<td>5</td>
<td>AA</td>
<td>Authoritative Answer</td>
</tr>
<tr>
<td>6</td>
<td>TC</td>
<td>Truncate or Truncated Response</td>
</tr>
<tr>
<td>7</td>
<td>RD</td>
<td>Recursion Desired</td>
</tr>
<tr>
<td>8</td>
<td>RA</td>
<td>Recursion Available</td>
</tr>
<tr>
<td>9</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>10</td>
<td>AD</td>
<td>Authentic Data (DNSSEC)</td>
</tr>
<tr>
<td>11</td>
<td>CD</td>
<td>Checking Disabled (DNSSEC)</td>
</tr>
<tr>
<td>12-15</td>
<td>RCODE</td>
<td>Result Code</td>
</tr>
<tr>
<td>Opcodes</td>
<td>Operation</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>0</td>
<td>Query</td>
<td>Standard query</td>
</tr>
<tr>
<td>1</td>
<td>lQuery</td>
<td>Inverse Query (obsolete)</td>
</tr>
<tr>
<td>2</td>
<td>Status</td>
<td>Status query (not standardized)</td>
</tr>
<tr>
<td>4</td>
<td>Notify</td>
<td>Change of master data</td>
</tr>
<tr>
<td>5</td>
<td>Update</td>
<td>Dynamic update</td>
</tr>
</tbody>
</table>
### DNS result codes

<table>
<thead>
<tr>
<th>Value</th>
<th>Mnemonic</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NoError</td>
<td>No Error</td>
</tr>
<tr>
<td>1</td>
<td>FormErr</td>
<td>Format Error</td>
</tr>
<tr>
<td>2</td>
<td>ServFail</td>
<td>Server Failure</td>
</tr>
<tr>
<td>3</td>
<td>NXDomain</td>
<td>Non-eXistent Domain</td>
</tr>
<tr>
<td>4</td>
<td>NotImp</td>
<td>Not Implemented</td>
</tr>
<tr>
<td>5</td>
<td>Refused</td>
<td>Query Refused</td>
</tr>
<tr>
<td>6-10</td>
<td>...</td>
<td>Related to dynamic updates</td>
</tr>
<tr>
<td>11-15</td>
<td>...</td>
<td>Not assigned</td>
</tr>
<tr>
<td>16-...</td>
<td>...</td>
<td>Extended result codes (EDNS0)</td>
</tr>
</tbody>
</table>
Queries

- In most cases QDCOUNT is 1
- Query consists of
  - QNAME (sequence of labels, coded with length/value)
  - QTYPE (2 bytes)
  - QCLASS (2 bytes, almost always IN (≡≡1))
Answers, Authorities and Additionals (1/2)

- Answers
  - Answers to question(s)
  - Special treatment of CNAME’s

- Authorities
  - Adds NS records as referral information

- Additionals
  - Courtesy information
  - Dangerous... if accepted too easily,
    especially if the information is not related to the question
Each of these are a list of resource records:

- Answers
- Authorities
- Additionals

Data per resource record group:

- NAME, TYPE, CLASS (as in queries)
- TTL (4 bytes)
- RDLENGTH (2 bytes)
- RDATA (RDLENGTH bytes)
Outline

1. DNAME resource record
2. DNS on the wire
3. Tea break
4. Zone transfers
5. Encoding of domain names
6. Limitations and extras
15 minute break

Enjoy your coffee/tea/lemonade/Club-Mate/Optimel!
Outline

1. DNAME resource record
2. DNS on the wire
3. Tea break
4. Zone transfers
5. Encoding of domain names
6. Limitations and extras
Use of zone transfers

- Copy data from master to slave server
  - ns1.os3.nl → master
  - ns2.os3.nl → slave
  - ns1.zurich.surf.net → slave

- Zone transfers often limited to slave servers
  - DNS-data: public or semi-public?
  - Prevent zone transfers using ACLs on IP level
How zones are transferred

- **Pull**
  - When starting without cache, or
  - when data is changed
  - DNS query type AXFR

- **Push**
  - Tells slave servers to pull :)  
    - Uses serial number to decide whether data is changed
  - DNS opcode 4 ("notify")
Outline

1. DNAME resource record
2. DNS on the wire
3. Tea break
4. Zone transfers
5. Encoding of domain names
6. Limitations and extras
Composition of domain names

- Domain name is a sequence of labels
- Start at leaf
  1. www
  2. os3
  3. nl
- Start at root
  1. nl
  2. os3
  3. www
- DNS starts at leaf (least significant label)
1. Use a delimiter to separate labels
   - `<label> <delimiter> <label>`
   - `www . os3 . nl .`

2. Specify length of labels in label encoding
   - `<label> <label>`
   - `{3,www} {3,os3} {2,nl} {0,}`

3. DNS uses latter way of encoding
“Normal label length” encoding

- First byte used for length
  - First 2 bits are flags
    - 00 means “normal label length”
  - Remaining 6 specify label length
    - Hence the maximum label length of $2^6 - 1 = 63$ octets
- Remaining bytes contain the label itself
- Number of remaining bytes is encoded in first byte of label
“Normal label length” encoding

R
L
F

Label length
Flags
“Normal label length” encoding
Compressed encoding

- Domain name with compressed encoding has fixed length of 2 bytes
  - First 2 bits are flags
    - 11 means “compressed label”
  - Remaining 6 bits + 8 subsequent bits are used as pointer
    - Points to label at other position in packet
    - Value is offset from beginning of packet
- Saves space when a domain name is used more than once
Compressed encoding
Compressed encoding
Reading domain names

Read first 2 bits of domain name field:

- If values is 00 (normal label length):
  1. If label length is 0 (empty label, thus root):
     - Return sequence of noted labels as domain name
  2. Read 6 subsequent bits and determine length of label
  3. Read first 2 bits of next byte and iterate

- If value is 11 (compressed encoding):
  1. Read 14 subsequent bits and determine position of domain name
  2. Jump to position and decode domain name
One more type: extended

- First byte value: 01000001
  - First two bits are 01
- Defines the use of EDNS0
- Can be used for binary labels
  - IPv6 PTR resource records
DNS limitations

- DNS is usually based on UDP
  - RFC 1035 maximum size is 512 bytes of DNS content
  - Option to use TCP was present from the start
    but was not recommended for ordinary use
- DNS has weak security
  - DNS packets can easily be spoofed
  - Initially no support for message authentication
    except for a clear text Transaction ID
Message Authentication

- **TSIG mechanism added in RFC 2845**
  - **TSIG** == Transaction Signature
  - Used mainly for updates (dynamic DNS)
  - Calculates HMAC-MD5 over the complete DNS packet
  - Uses secret keys

- **SIG(0) mechanism added in RFC 2931**
  - DNS Request and Transaction Signatures
  - Uses public keys
  - Uses DNSSEC mechanisms
  - Extends DNSSEC to cover complete DNS packets
Extension Mechanisms for DNS

- **EDNS0**
  - Specified in RFC 2671
  - Published in 1999
  - Necessary for DNSSEC
  - Extends maximum size of UDP-based requests and responses
  - Extends possible flags, result codes and label types
  - Uses a “pseudo”-OPT-RR
  - Used by DNSSEC for DO (DNSSEC OK) extended flag