DNS

Karst Koymans & Niels Sijm

Informatics Institute

University of Amsterdam

Friday, September 14, 2012

Outline

Wire?

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

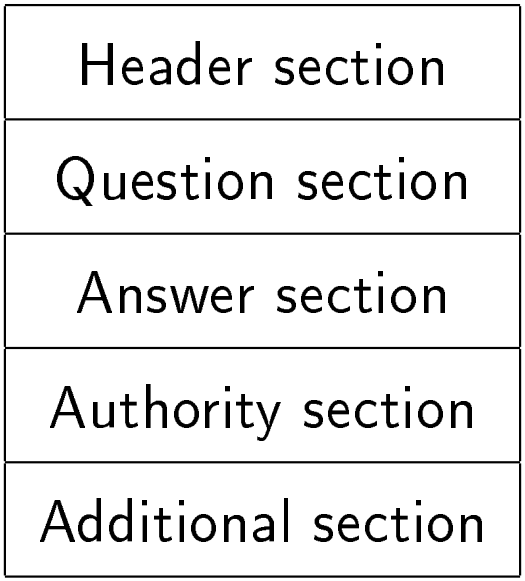
How to package DNS messages

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 :)

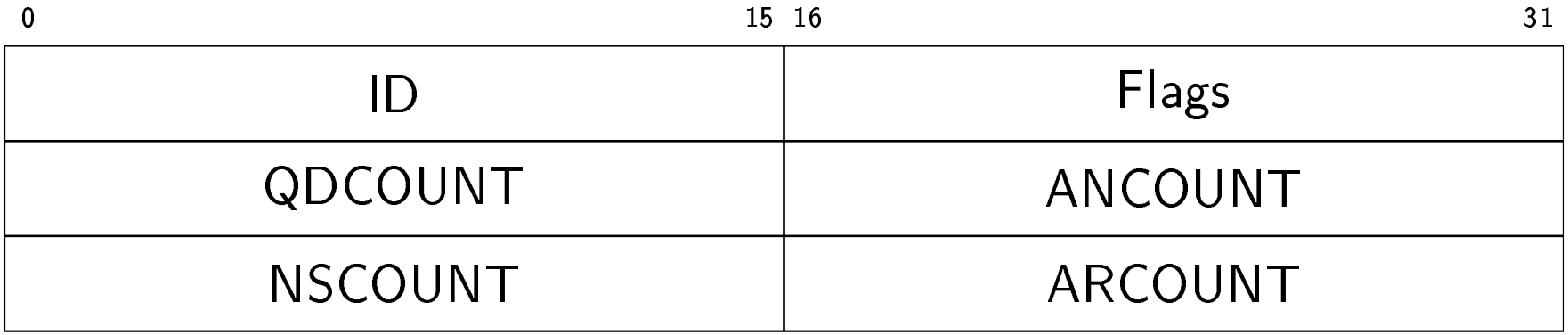
How to tap the wire?

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



DNS packet header



DNS header fields

ID	Transaction Identifier
Flags	See next slide
QDCOUNT	Number of questions
ANCOUNT	Number of answers
NSCOUNT	Number of authority records
ARCOUNT	Number of additional records

DNS header flags

Bit(s)	Mnemonic	Meaning
0	QR	Query(0) or Response(1)
1-4	OPCODE	Kind of Query
5	AA	Authoritative Answer
6	TC	TrunCation or Truncated Response
7	RD	Recursion Desired
8	RA	Recursion Available
9	-	Reserved
10	AD	Authentic Data (DNSSEC)
11	CD	Checking Disabled (DNSSEC)
12-15	RCODE	Result Code

DNS opcodes

0	Query	Standard query
1	IQuery	Inverse Query (obsolete)
2	Status	Status query (not standardized)
4	Notify	Change of master data
5	Update	Dynamic update

DNS result codes

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

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))

Label types

- First two bits of the length byte denote the label type
  - A label length may not exceed 63 octets (6 bits needed)
  - 00: Normal label length
  - 11: Compressed label: 6+8 bits used as pointer
  - 01: Extended label type (EDNS0)
    - 01000001: Binary labels (for use with IPv6 PTR types)

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

Answers, Authorities and Additionals (2/2)

- 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

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 transfered

- 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

Wildcards (1)

From RFC 1034, section 4.3.3

The contents of the wildcard RRs follows the usual rules and formats for RRs. The wildcards in the zone have an owner name that controls the query names they will match. The owner name of the wildcard RRs is of the form "*.<anydomain>", where <anydomain> is any domain name. <anydomain> should not contain other * labels, and should be in the authoritative data of the zone. The wildcards potentially apply to descendants of <anydomain>, but not to <anydomain> itself. Another way to look at this is that the "*" label always matches at least one whole label and sometimes more, but always whole labels.

Wildcards (2)

From RFC 1034, section 4.3.3

Wildcard RRs do not apply:

- When the query is in another zone. That is, delegation cancels the wildcard defaults.
- When the query name or a name between the wildcard domain and the query name is know to exist. For example, if a wildcard RR has an owner name of "*.X", and the zone also contains RRs attached to B.X, the wildcards would apply to queries for name Z.X (presuming there is no explicit information for Z.X), but not to B.X, A.B.X, or X.

Wildcard synthesis in query matching algorithm

From RFC 1034, section 4.3.2, algorithm step 3.c

If at some label, a match is impossible (i.e., the corresponding label does not exist), look to see if a the "*" label exists.

If the "*" label does not exist, check whether the name we are looking for is the original QNAME in the query or a name we have followed due to a CNAME. If the name is original, set an authoritative name error in the response and exit. Otherwise just exit.

If the "*" label does exist, match RRs at that node against QTYPE. If any match, copy them into the answer section, but set the owner of the RR to be QNAME, and not the node with the "*" label. Go to step 6.

Problems with wildcards in RFC 1034

- Notions are intuitive, not well-defined
  - When does a domain name "exist"?
  - How does matching work exactly?
  - What about empty non-terminals?
- RFC 4592 tries to clarify all of this
  - Defines "existence of a domain name"
  - Defines "asterisk label" and "wildcard domain name"
  - Defines "source of synthesis" and "closest encloser"

Wildcard supporting definitions

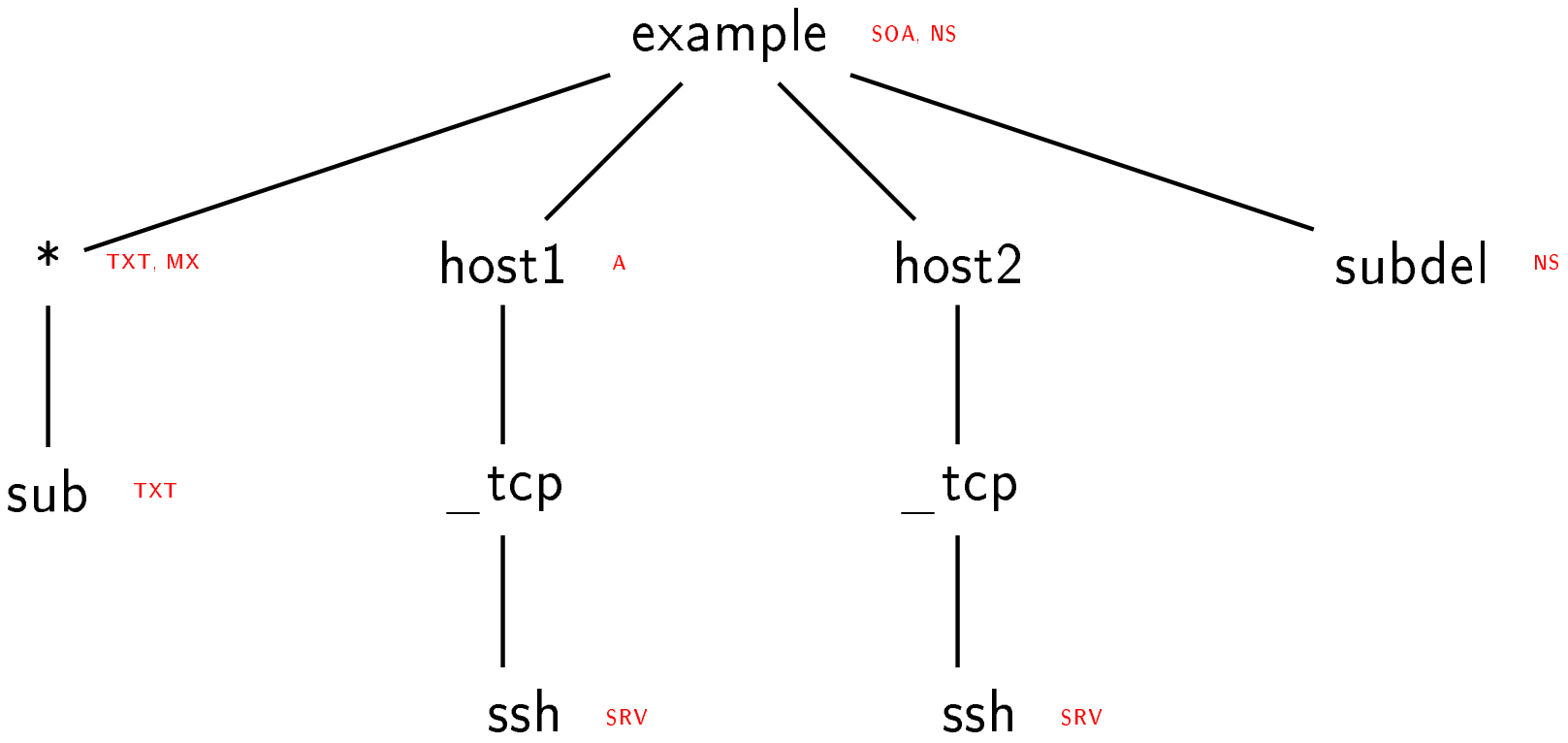
Definitions

- A domain name exists if itself or
  any of its descendants has at least one RR
  - In particular empty non-terminals exist
- An asterisk label is a label of length 1
  containing as only octet the ASCII equivalent of "*"
- A wildcard domain name is a domain name
  with an asterisk label as its leftmost label
- The closest encloser of a query name
  is the longest matching ancestor that exists
- The source of synthesis of a query name
  is the domain name "*.<closest encloser>" (if it exists)

RFC 4592 example

```
$ORIGIN example.
example.                    3600 IN  SOA   <SOA RDATA>
example.                    3600     NS    ns.example.com.
example.                    3600     NS    ns.example.net.
*.example.                  3600     TXT   "this is a wildcard"
*.example.                  3600     MX    10 host1.example.
sub.*.example.              3600     TXT   "... not a wildcard"
host1.example.              3600     A     192.0.2.1
_ssh._tcp.host1.example.    3600     SRV   <SRV RDATA>
_ssh._tcp.host2.example.    3600     SRV   <SRV RDATA>
subdel.example.             3600     NS    ns.example.com.
subdel.example.             3600     NS    ns.example.net.
```

RFC 4592 example tree



RFC 4592 example queries

QNAME	QTYPE	synthesized?	result
host3.example.	MX	yes	non-empty
host3.example.	A	yes	empty
foo.bar.example.	TXT	yes	non-empty
host1.example.	MX	no	empty
sub.*.example.	MX	no	empty
_telnet._tcp.host1.example.	SRV	no	no such domain
host.subdel.example.	A	no	referral
ghost.*.example.	MX	no	no such domain

Outline

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