Naming and DNS [*]

Jens Lechtenbörger

VM Neuland im Internet 2021

Contents

1 Introduction

1.1 Today's Core Questions

- How are names typically managed in computer systems?
- What is the DNS?
  - How to interpret names such as `www.uni-muenster.de`?
  - How does it work?

1.2 Learning Objectives

- Explain DNS as example for namespace management
  - Explain notions of name server, zone, resource record
  - Explain iterative DNS resolution
- Discuss properties of Zooko's triangle in relation to DNS
- Perform DNS lookups for different resource types via nslookup

1.3 Recall: DNS in Internet Protocol Stack

- DNS is application layer request-reply protocol between clients and servers
- Finding IP addresses for names such as `www.uni-muenster.de` is among the first steps of Internet communication

[*]This PDF document is an inferior version of an OER HTML page; free/libre Org mode source repository.

1.4 Namespaces and Name Services

- **Name service** = System managing **namespace**, in particular mapping names to values
  - E.g., DNS names and IP addresses, E-Mail addresses and public keys, file names and storage locations
  - Names may be flat or hierarchical, potentially infinitely many
  - Namespace often structured
    - Similar name parts without conflict
      - E.g., server named `www` in lots of organizations
    - Grouping of related names
      - E.g., all university computers with names ending in `uni-muenster.de`
    - Typically directed graphs
      - Leafs: Named entities
      - Inner nodes: Directories
    - Enable transparent restructuring

2 Domain Name System (DNS)

2.1 Basics

- DNS specified in RFC 1034 and RFC 1035
  - Updated/augmented by various other RFCs
    - `http://bind9.net/rfc`
- DNS is a distributed **name service**
  - Collectively manages DNS names
    - Names mapped to **resource records** by **name servers**
      - Resource record: Next slide
  - Scalable (today's Internet)
    - Caching
    - Partitioned database
  - Fault tolerance via replication

2.1.1 Resource Records

- Name servers manage **resource records**
  - (`NAME`, `TTL`, `CLASS`, `TYPE`, `VALUE`)
    - E.g., (`www.uni-muenster.de.`, `7200`, `IN`, `A`, `128.176.6.250`)
    - `NAME`, `VALUE`: Mostly names and IP addresses
    - `TTL`: Time to live (caching period in seconds) of resource record
    - Sample `TYPE` values
      - `A`: DNS name `NAME` has IPv4 address `VALUE` (IPv6 is `AAAA`)

      - NS: VALUE is DNS name for authoritative name server of domain NAME
      - CNAME: VALUE is canonical name; used for aliasing
      - MX: VALUE is DNS name for mail server of domain NAME
      - SOA: Start of Authority, management and name server information
    - CLASS: Constantly IN in Internet

- Live examples in-class

2.2 DNS Names

- DNS defines namespace with absolute and relative names
  - **Absolute**: ending in dot (`.`)
    - `www.uni-muenster.de.`
    - `.` (**Root**)
  - **Relative**: not ending in dot
    - `www.uni-muenster.de`, `www`
    - `de`, `org` (Top Level Domains, **TLDs**)
    - Completed in local domain
      - E.g., `www` might mean `www.wiwi.uni-muenster.de.`, `www.uni-muenster.de.`, or `www.wwu.de`, while `www.uni-muenster.de` means `www.uni-muenster.de.`
      - Final dot almost always omitted when typing names
  - Reserved names: RFC 2606, e.g., `.example.com`

DNS specifies names consisting of labels that are separated by dots. DNS distinguishes absolute names, ending in a dot, from relative ones. I would not be surprised if you have not seen absolute names so far. A special absolute name is the dot by itself. It denotes the root of the address space. Underneath the root, lots of top level domains are defined such as DE and ORG.

When relative names show up, they need to be completed into absolute names before DNS information can be retrieved. Locally administered rules are applied towards that end. For example, inside the University of Muenster the name "www" might be completed into "www.wiwi.uni-muenster.de." or "www.uni-muenster.de.".

2.3 DNS Zones

- DNS namespace partitioned into so-called **zones**
  - Zones reflect administrative structure of Internet
    - E.g., `wwu.de.` is a subzone of `de.`, which in turn is a subzone of the root zone `.`
    - Various registries (e.g., DENIC for `.de.`, university for `wwu.de.`)
  - Zones managed by authoritative name servers
    - A form of decentralization
    - At least two replicated name servers per zone for fault tolerance

2.3.1 DNS Namespace

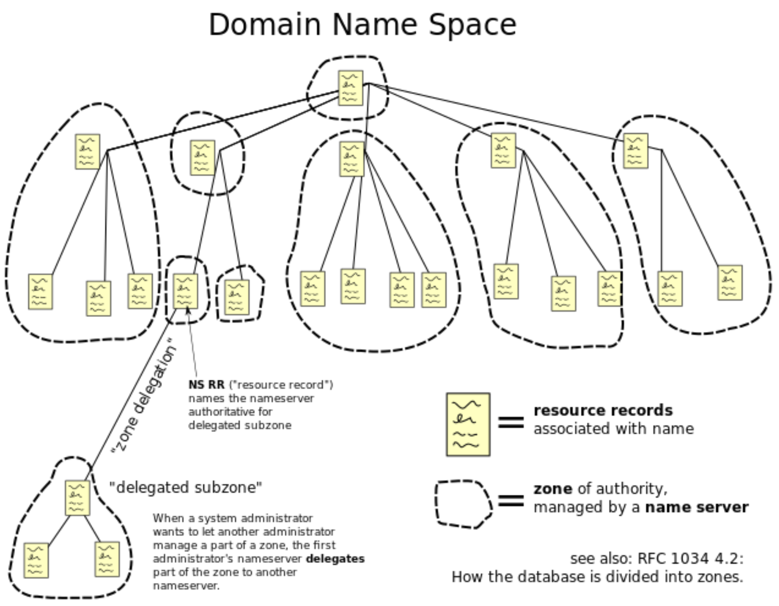
Domain Name Space
"zone delegation"
NS RR ("resource record") names the nameserver authoritative for delegated subzone
"delegated subzone"
When a system administrator wants to let another administrator manage a part of a zone, the first administrator's nameserver **delegates** part of the zone to another nameserver.
= **resource records** associated with name
= **zone** of authority, managed by a **name server**
see also: RFC 1034 4.2: How the database is divided into zones.

Figure 1: “Domain name space” by LionKimbro under Public domain; from Wikimedia Commons

2.3.2 DNS Root Zone and Servers

- Root zone `.` managed by ICANN (Internet Corporation for Assigned Names and Numbers)
  - Names and addresses of root name servers
    - named.cache
    - Starting points for name lookups
  - 13 root name server names as redundant entry points
    - Single letter names `a.root-servers.net`, . . . , `m.root-servers.net`
      - Those 13 names are implemented by hundreds of replicated servers
      - (Using a routing technique called anycast)
    - Root servers are responsible for TLD lookups

2.3.3 Resolution Challenge

- No DNS server knows all names
- E.g., university’s name server knows **everything** about names ending in `wwu.de.`, but **nothing** about names ending in `wikipedia.org.`
  - What to do if client in university asks for `www.wikipedia.org`?

  - Answer: Our name server asks somebody else
    * Our name server does **not** know a name server for `wikipedia.org`.
    * Every name server is **preconfigured** with names and addresses of root name servers . . .

2.4 Name Resolution

- Obtaining information for a DNS name is called **name resolution**
- Components responsible for resolution are called **resolvers**
  - Each with cache for recent results (up to time-to-live (TTL) of resource record)
- Side note
  - DNS name does not necessarily represent single machine
    * Multiple resource records can exist for any name
      · E.g., multiple `A` records for multiple IPv4 addresses
    * Subsequent DNS requests for name can lead to different IP addresses, e.g., round-robin DNS

2.4.1 DNS Resolvers

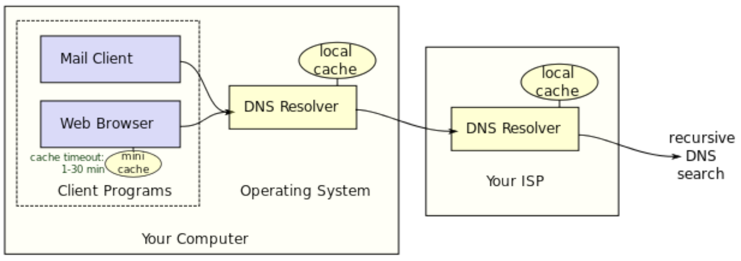

Figure 2: “DNS in the real world” by Lion Kimbro under Public domain; from Wikimedia Commons

2.4.2 Local Resolver

- Application calls OS, e.g., `InetAddress.getByName(String)` in Java
- OS uses local configuration and network services
  - “hosts file”: `/etc/hosts`, `%systemroot%\system32\drivers\etc\hosts`
    * List of DNS names with IP addresses
  - Name services such as DNS, LDAP, NIS
    * OS configured with IP address(es) of DNS name server(s)
      · E.g., `128.176.0.12`, `128.176.0.13` within university

            · Automatic configuration with DHCP

- Beware of profiling
    - Your DNS server learns every hostname contacted by your computer

2.4.3 Iterative DNS Resolution

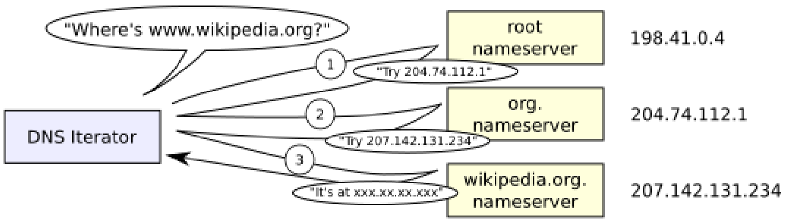

Figure 3: "Example of an iterative DNS resolver" by Lion Kimbro under Public domain; from Wikimedia Commons

What is called DNS Iterator here, could be our university's name server, which is contacted by your operating system when you enter an address in your browser. That server is called Iterator here because you see the iterative variant of DNS resolution. A recursive variant exists as well but will not be covered here.

In the shown example, the IP address of a web server of wikipedia is requested. When the DNS Iterator is started, it is preconfigured with the names and IP addresses of the root name servers. Based on this information, it can perform iterative resolution starting from the end of the name, here dot org.

In step 1, the Iterator contacts a root name server to learn the IP address of a TLD name server responsible for dot org. Here, the root name server with IP address 198.41.0.4 delivers the IP address 204.74.112.1 of a name server for the TLD dot org. In step 2, the Iterator contacts that name server and asks it for a name server responsible for the domain wikipedia.org, which results in IP address 207.142.131.234. In step 3, finally that name server is contacted to resolve the IP address for the web server www.wikipedia.org.

2.5 DNS as Network Application

- DNS follows request/reply communication pattern
    - Queries and answers are network messages
        * "Small," fit into single IP datagram
        * Delivery either via UDP or TCP
            · Typically, DNS uses UDP on well-known port 53
            · Best effort service model
            · Why is that good enough?
- Originally, no security mechanisms
    - DNS spoofing (cache poisoning)
    - Attackers add fake IP addresses to hosts file to redirect traffic
    - Attackers replace DNS server configuration (e.g., GhostDNS in 2018)

- DNSSEC under deployment since 2010
  - Integrity protection with digital signatures

3 In-Class

3.1 Dynamic DNS (DynDNS)

- Aim: Permanent DNS name for dynamic IP address
- Idea
  - Special DNS servers allow updates in real-time
  - TTL of 60s
  - Special client software to notify DNS server of changed IP address
    - Sometimes integrated into DSL router/modem

3.2 DNS Query Tools

- `dig` (next slide) or `nslookup` as command line tools
- `nslookup` (monitor with Wireshark to see all details)
  - Query for IP address
    - `www.uni-muenster.de`
  - Query for mail server
    - `set type=MX`
    - `uni-muenster.de`
  - Query for name server, ask specific server
    - `set type=NS`
    - `server 128.176.0.12`
    - `gitlab.com`
- Further info on domain names: `whois`
  - `whois gitlab.com`

3.2.1 Example for `dig`

- Why 13 name servers?
- Answer: Compatibility issues
  - UDP replies restricted by DNS (RFC 1035) to 512 bytes
  - `dig @a.root-servers.net .  soa`
    - With 13 servers, response had 497 bytes; no space for 14th
- Now larger responses with Extension mechanisms for DNS

3.3 DNS over HTTPS (DoH)

- DoH: Encapsulate DNS requests in HTTPS messages for security
  - RFC 8484
    - Since 2018
      - Different degrees of adoption in different browsers
  - Controversy
    - Protection of HTTPS
    - “Trusted” DoH resolver
      - Centralization step, single point of failure, single point of surveillance

3.4 Decentralized Namespaces

- Desirable properties
  - **Security**, with variations
    - Self-authenticating: Given name-value pair, anyone can verify that both belong together
      - E.g., name is cryptographic hash of value
    - Strong ownership: Proof of name ownership with digital signatures
  - **User-chosen** names: Readable, memorizable
  - **Decentralization**: Users register names without central authorities
- Above three properties known as Zooko’s triangle
  - [Wil01] Thought to be impossible to have all three
  - Blockchain technology may provide solutions
    - (See [Kal+15] for Namecoin, [Ali+16] for Blockstack)

[Kal+15] presents empirical study of Namecoin

- Namecoin dominated by domain squatting, no secondary market, poor client software
- Real problem is mapping entities to values, which is not addressed by Namecoin; no verification as for Extended Validation certificates
- What about trademarks?

3.4.1 DNS in Relation to Zooko’s Triangle?

4 Conclusions

4.1 Summary

- DNS is prime example for name service and distributed system
- DNS servers manage resource records in zones
- Zooko’s triangle points to limitations

Bibliography

[Ali+16] Muneeb Ali et al. “Blockstack: A Global Naming and Storage System Secured by Blockchains”. In: *USENIX Annual Technical Conference*. 2016, pp. 181–194. URL: https://www.usenix.org/system/files/conference/atc16/atc16_paper-ali.pdf.

[Kal+15] Harry A. Kalodner et al. “An Empirical Study of Namecoin and Lessons for Decentralized Namespace Design”. In: *WEIS*. 2015. URL: https://www.econinfosec.org/archive/weis2015/papers/WEIS_2015_kalodner.pdf.

[Wil01] Zooko Wilcox-O’Hearn. *Names: Decentralized, Secure, Human-Meaningful: Choose Two*. 2001. URL: https://web.archive.org/web/20011020191610/http://zooko.com/distnames.html.

License Information