Computer Networks (WS23/24)

L10: Domain Name System and the Web

Prof. Dr. Lin Wang
Computer Networks Group
Department of Computer Science
Paderborn University

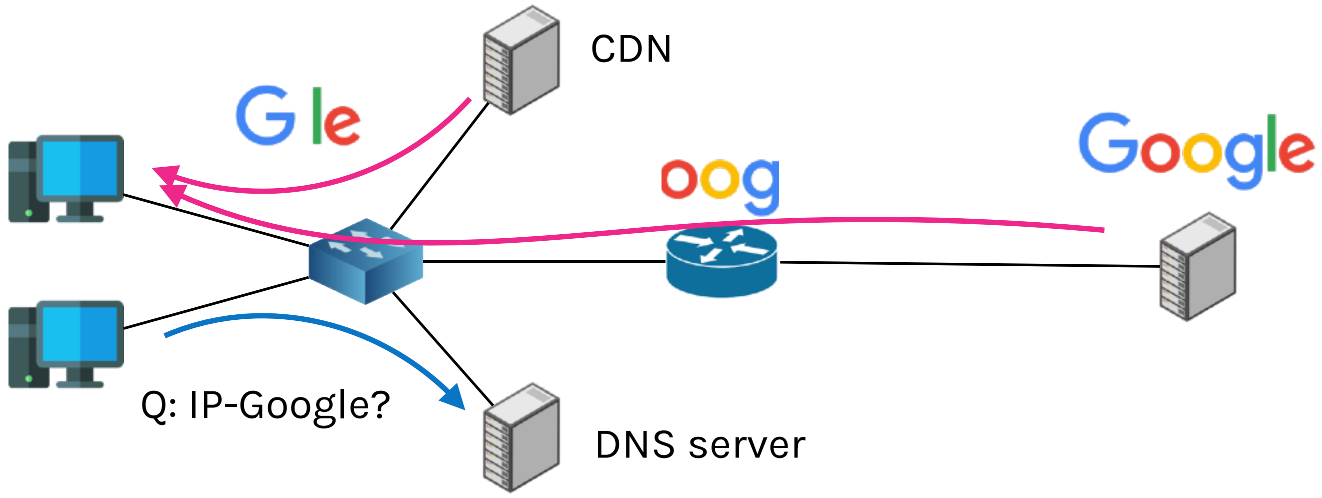
Materials inspired by Laurent Vanbever

Learning objectives

- Application
- Transport
- Network
- Link
- Physical

Part 1

- Domain Name System (DNS)
- The web (HTTP, caching and CDN)



Domain Name System (DNS)

DNS

A distributed database to resolve a domain name into an IP address

Name	DNS ⟷	IP address	
www.google.com		142.251.209.142	Names can be mapped to more than one IP
www.google.com		142.250.179.206	
www.upb.de		131.234.142.33	IPs can be mapped by more than one name
www.uni-paderborn.de		131.234.142.33	

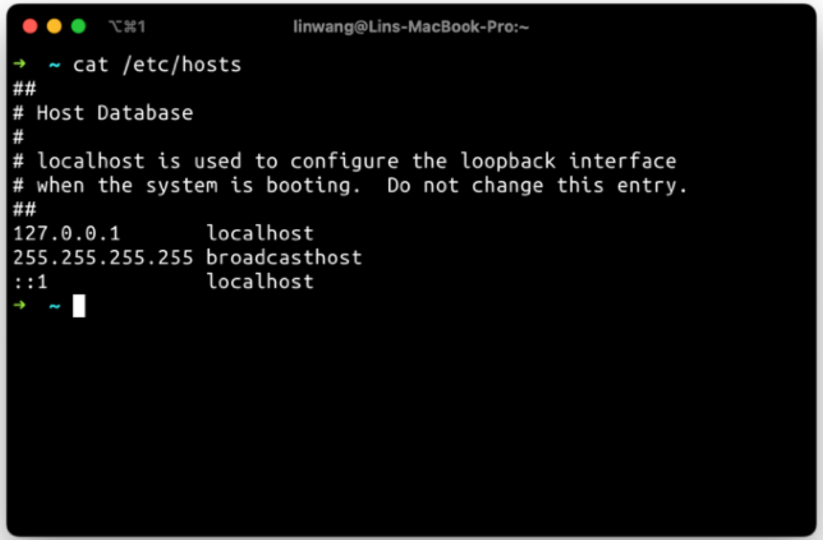
How to resolve a name into an IP?

Initially

- All host-to-address mappings were stored in a file called hosts.txt
- Unix-like systems: /etc/hosts

Problems

- Scalability
- Consistency
- Availability



Two universal tricks in Computer Science

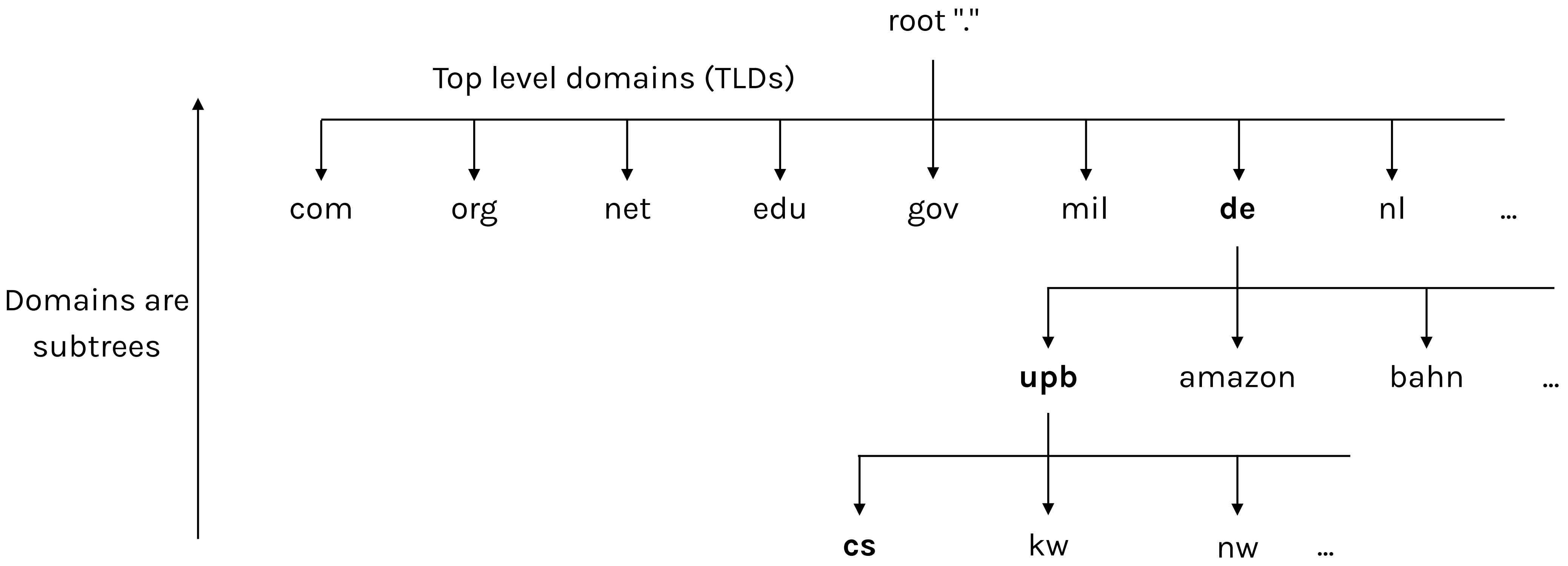
When you need…	more **flexibility**
You add…	a layer of **indirection**

When you need…	more **scalability**
You add…	a **hierarchical** structure

Hierarchies in DNS

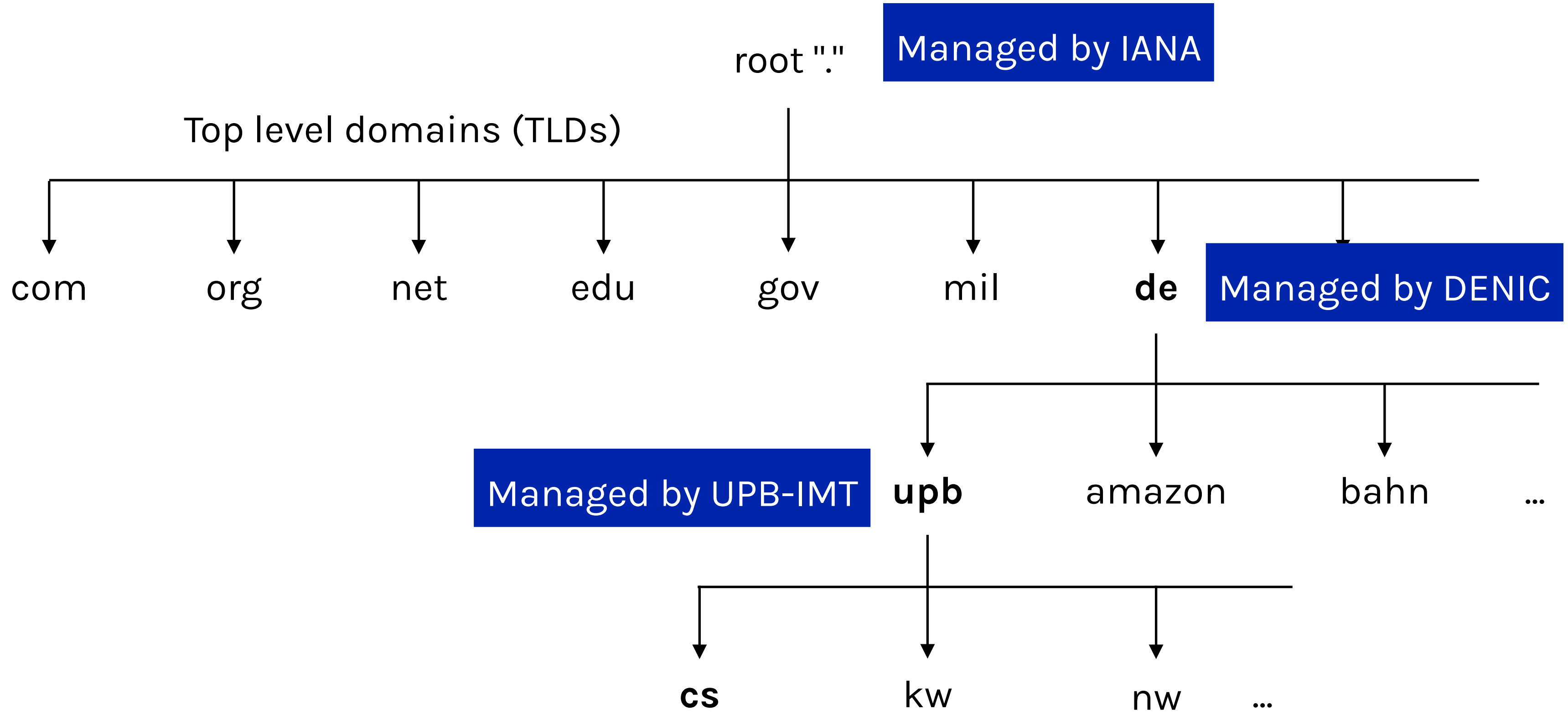
Naming structure	Hierarchy of addresses https://en.cs.uni-paderborn.de/pbnet
Management	Hierarchy of authority over names
Infrastructure	Hierarchy of DNS servers

Hierarchy of addresses



A name (e.g., cs.upb.de) represents a leaf-to-root path in the hierarchy

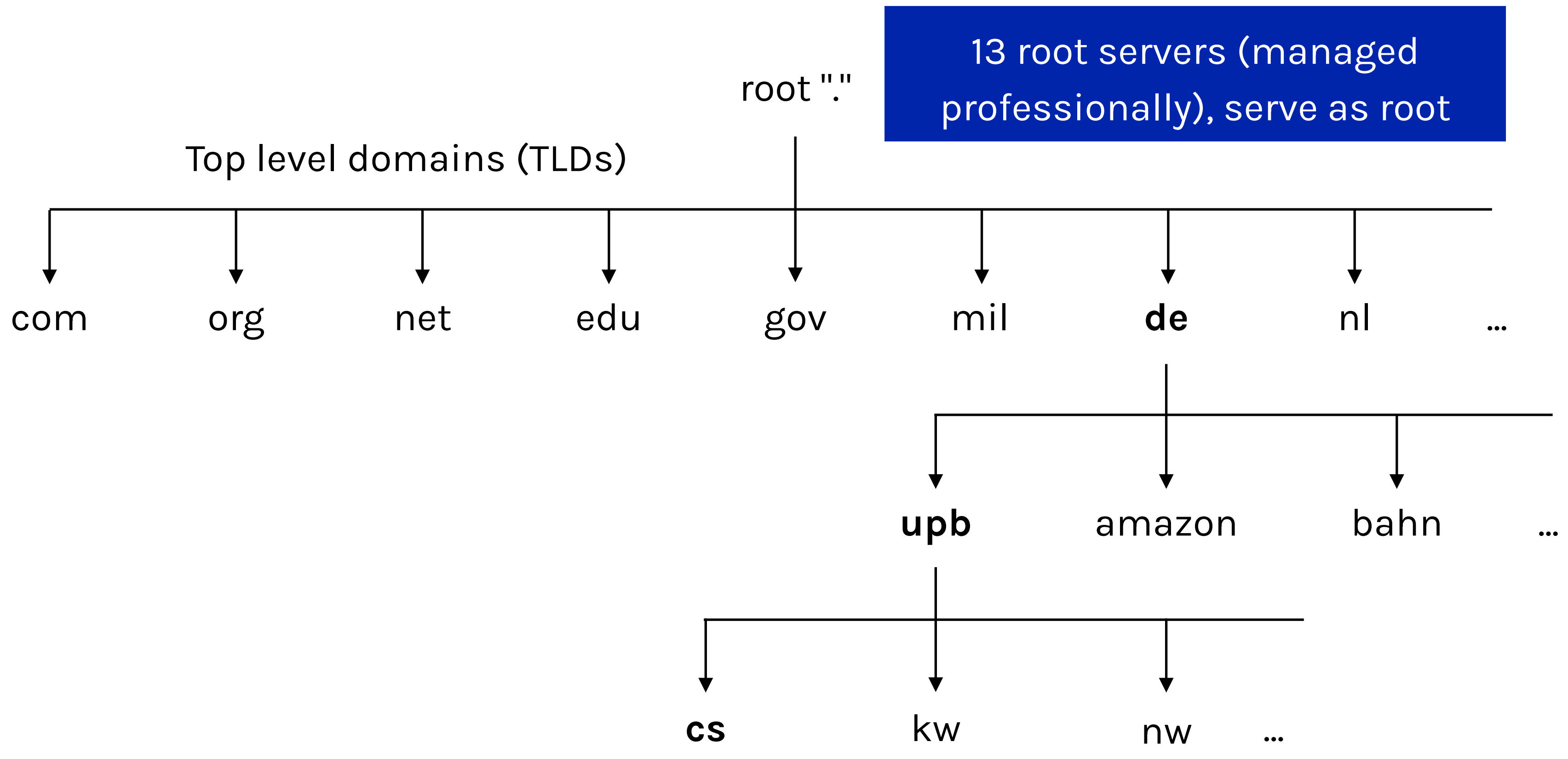
Hierarchy of authorities



https://www.iana.org/domains/root/db
https://en.wikipedia.org/wiki/DENIC
https://imt.uni-paderborn.de/en/network-services-dns-dhcp

Name collision is trivially avoided

Hierarchy of DNS servers



Root servers

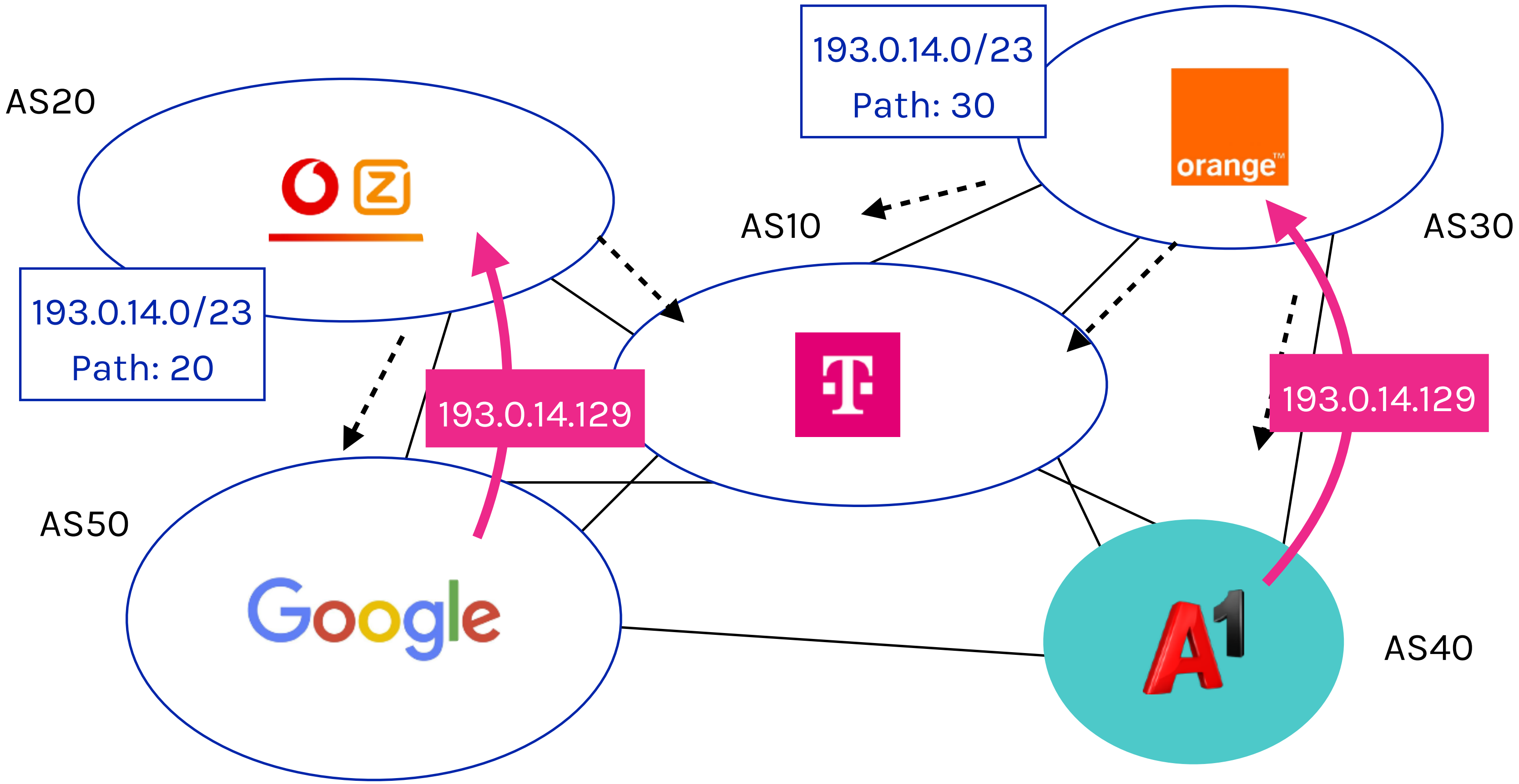
a.root-servers.net	VeriSign, Inc.
b.root-servers.net	USC
c.root-servers.net	Cogent Communications
d.root-servers.net	University of Maryland
e.root-servers.net	NASA
f.root-servers.net	Internet Systems Consortium
g.root-servers.net	US Department of Defense
h.root-servers.net	US Army
i.root-servers.net	Netnord
j.root-servers.net	VeriSign, Inc.
k.root-servers.net	RIPE NCC
l.root-servers.net	ICANN
m.root-servers.net	WIDE Project

Instances of k.root-servers.net

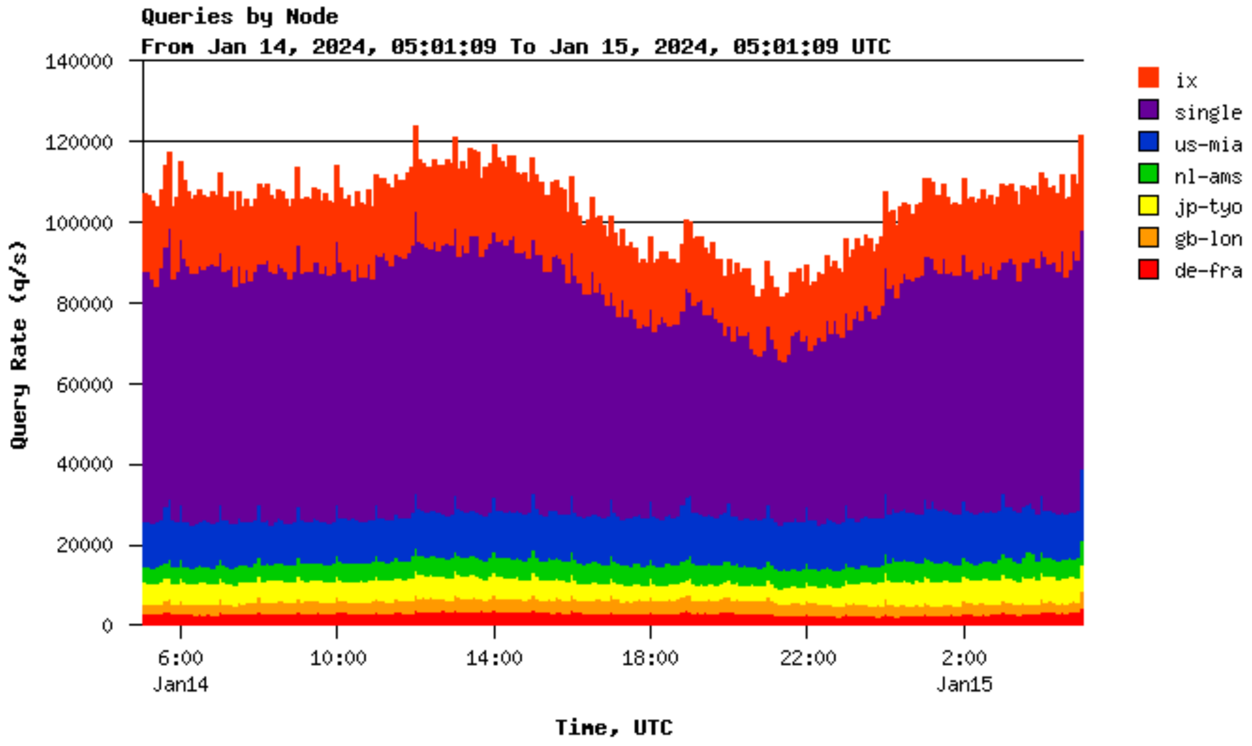
All locations announce 193.0.14.0/23 in BGP, with 193.0.14.129 being the IP of the server

https://www.ripe.net/analyse/dns/k-root/

BGP anycast to scale root servers

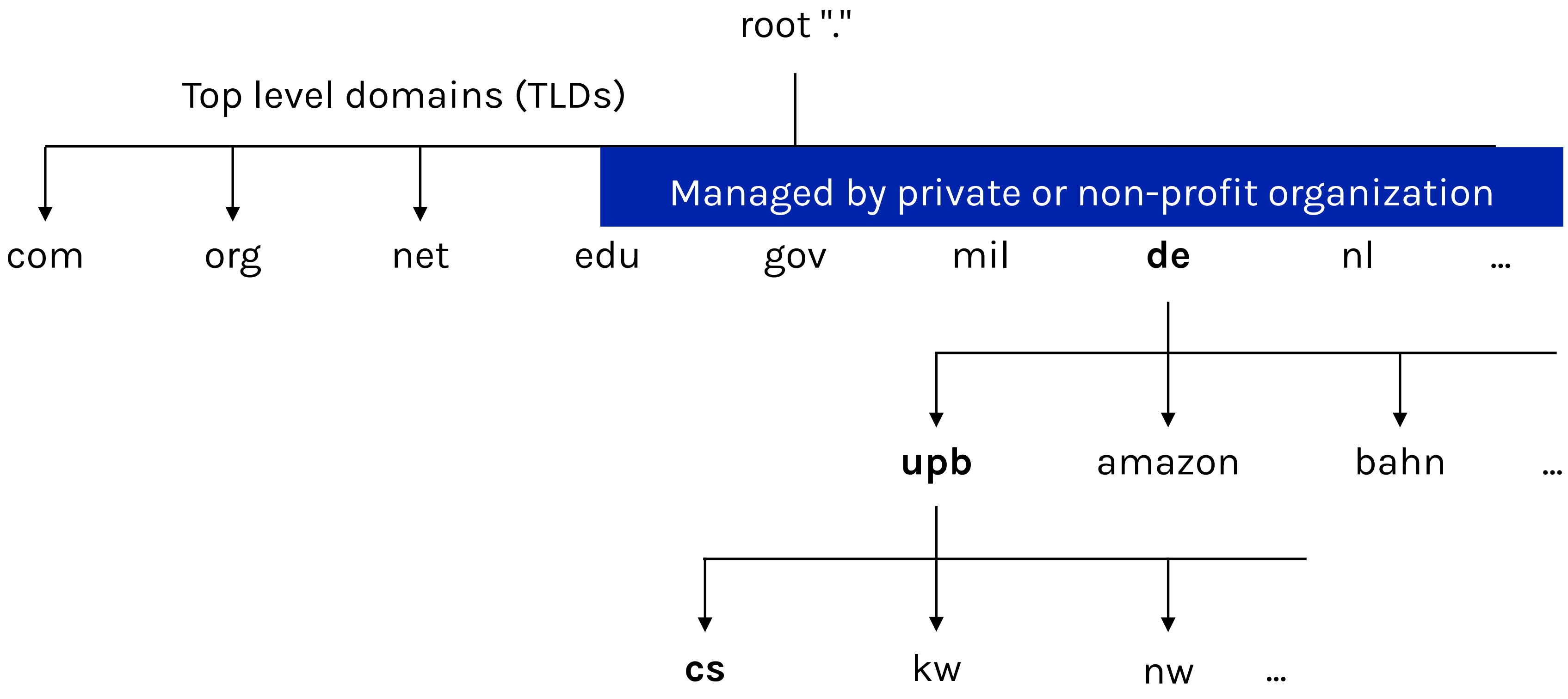


Query load: around 110K queries per second



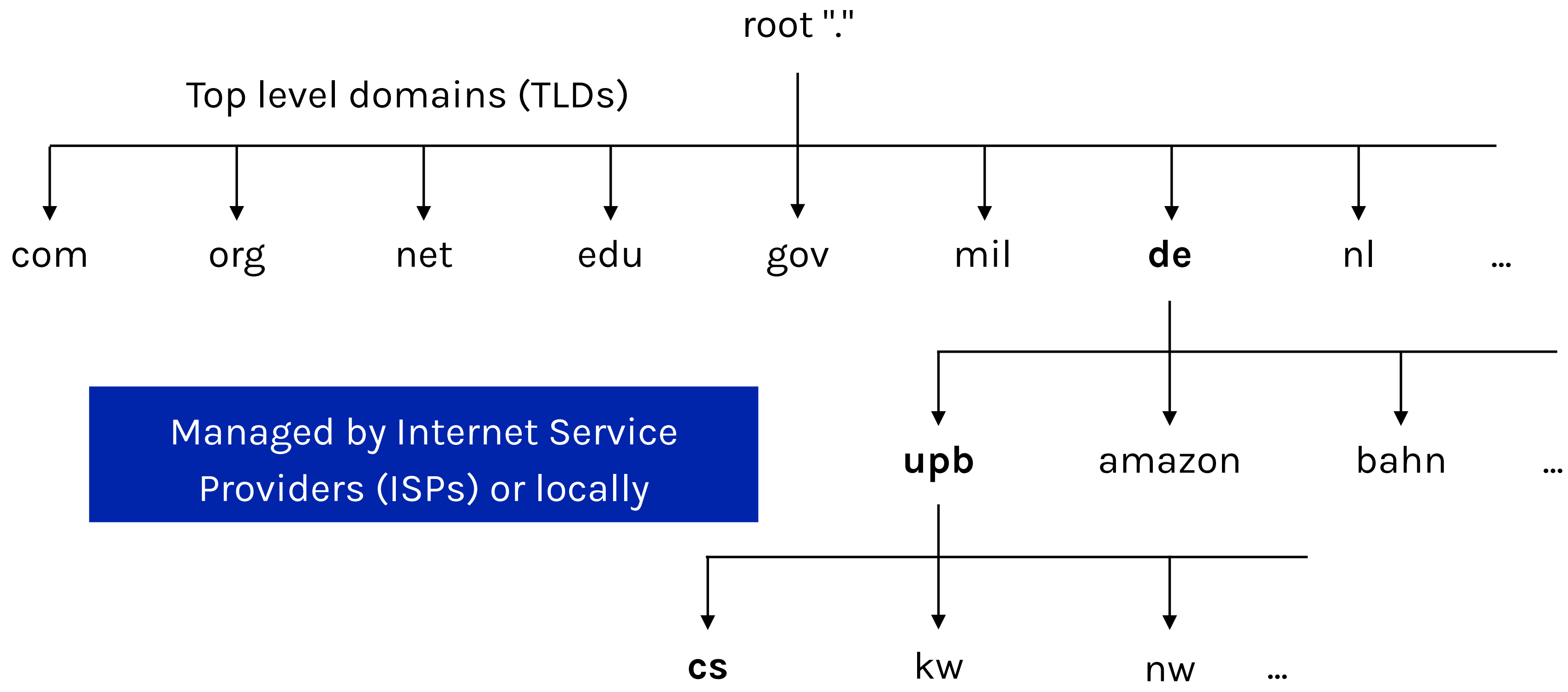
https://www.ripe.net/analyse/dns/k-root/statistics?type=ROOT&increment=daily&

Hierarchy of DNS servers



https://root-servers.org/

Hierarchy of DNS servers



Managed by Internet Service Providers (ISPs) or locally

Hierarchy of DNS servers

Every server knows the address of the root servers (for bootstrapping)

Command: `dig @dns-server domain-name`

Each root server knows the address of all TLD servers

```
;; AUTHORITY SECTION:
de.                     172800  IN      NS      s.de.net.
de.                     172800  IN      NS      n.de.net.
de.                     172800  IN      NS      a.nic.de.
de.                     172800  IN      NS      f.nic.de.
de.                     172800  IN      NS      l.de.net.
de.                     172800  IN      NS      z.nic.de.
```

All .de DNS server knows the addresses of the DNS servers of all sub-domains

```
;; AUTHORITY SECTION:
upb.de.                 86400   IN      NS      dns-1.dfn.de.
upb.de.                 86400   IN      NS      dns1.uni-paderborn.de.
upb.de.                 86400   IN      NS      dns2.uni-paderborn.de.
upb.de.                 86400   IN      NS      dns3.uni-paderborn.de.
```

Each DNS server knows the IP address of all children

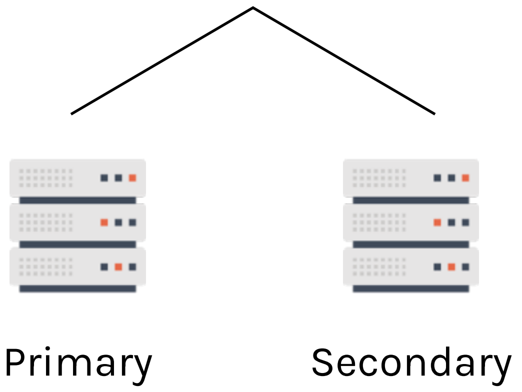
```
;; ANSWER SECTION:
upb.de.                 3600    IN      DNAME   uni-paderborn.de.
cs.upb.de.              3600    IN      CNAME   cs.uni-paderborn.de.
cs.uni-paderborn.de.    71154   IN      A       131.234.9.34
```

https://www.internic.net/domain/named.root

Availability

Each domain must have at least a primary and secondary DNS server

- Ensure name service availability as long as one of the servers is up
- DNS queries can be load-balanced across the replicas over servers
- On timeout, client uses alternative servers (applying exponential backoff when trying the same server)




```
;; AUTHORITY SECTION:
upb.de.                 86400   IN      NS      dns-1.dfn.de.
upb.de.                 86400   IN      NS      dns1.uni-paderborn.de.
upb.de.                 86400   IN      NS      dns2.uni-paderborn.de.
upb.de.                 86400   IN      NS      dns3.uni-paderborn.de.
```

DNS system properties

Scalable

- #names, #updates, #lookups, #users
- Also in terms of administration

Available

- Domains replicate independently of each other

Extensible

- Any level (including the TLDs) can be modified independently

Inserting a name into the DNS system

You have founded next-startup.de and want to host it yourself

Step 1: You register next-startup.de at a registrar X

- Examples: GoDaddy, name-cheap.org, and many more

Provide X with the name and IP of your DNS servers

- Example: [ns1.next-startup.de, 131.234.20.112]

You set up a DNS server at 131.234.20.112

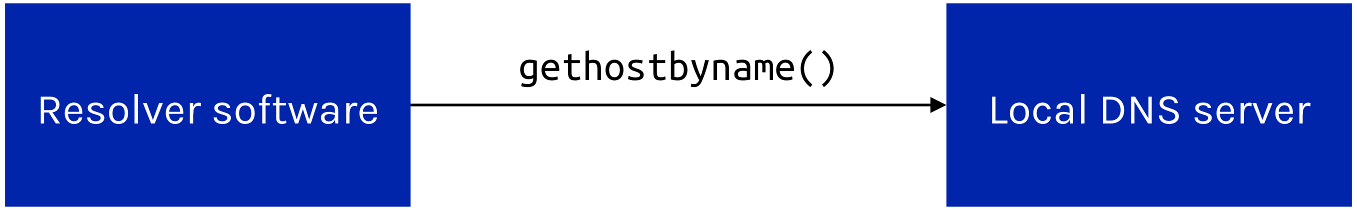
- Define A records for www, MX records, etc.

Records on DNS servers

A DNS server stores Resource Records composed of (name, value, type, TTL)

Records	Name	Value
A	Hostname	IP address
NS	Domain	DNS server name
MX	Domain	Mail server name
CNAME	Alias	Canonical name
PTR	IP address	Corresponding host name

Using DNS

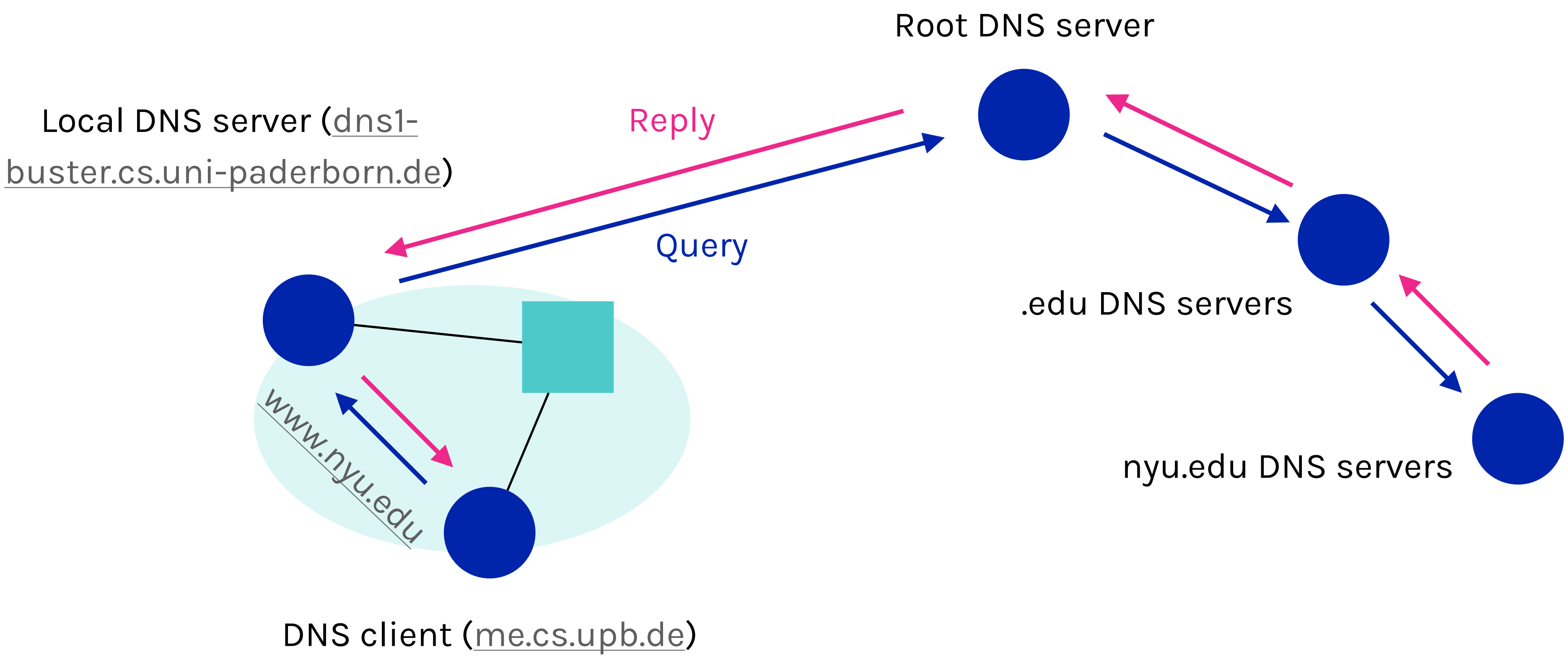


Trigger resolution process and send requests to local DNS server

Usually near the end-hosts, configued statically (resolv.conf) or dynamically (DHCP)

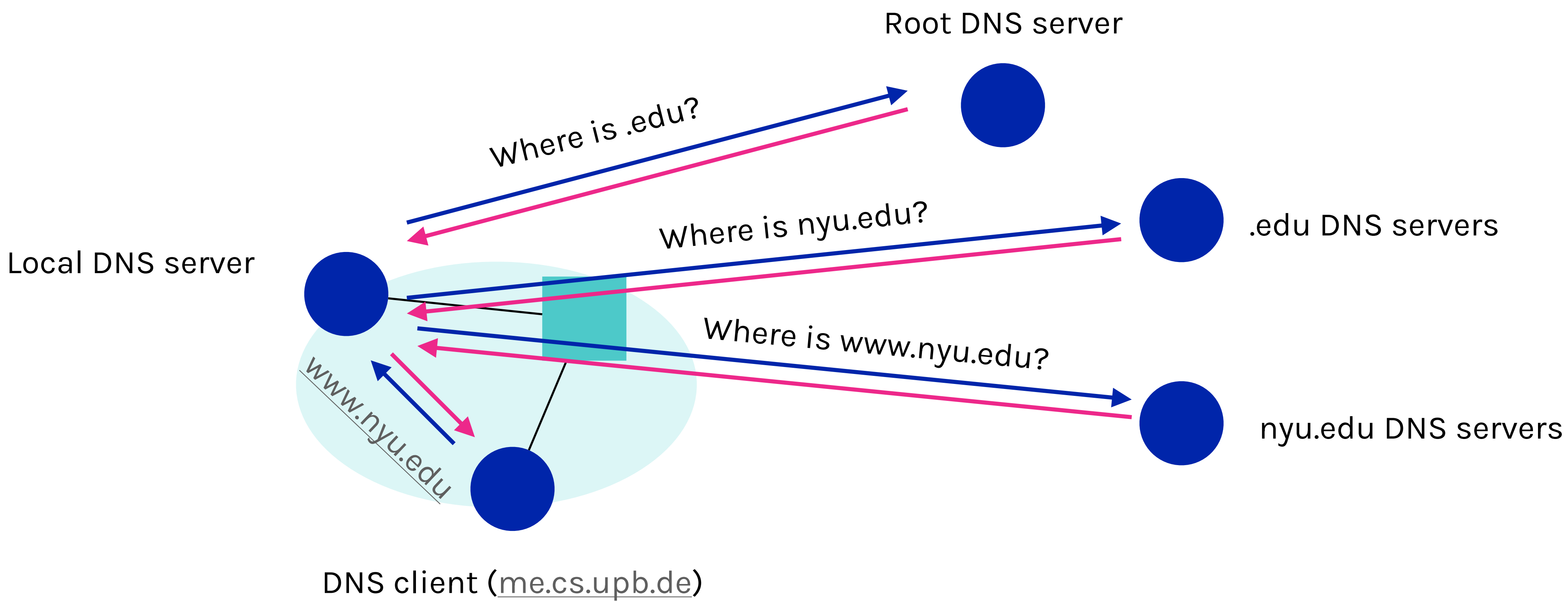
DNS query and reply uses UDP (port 53); reliability is implemented by repeating requests

Recursive query resolution



The client offloads the task of resolving to the server recursively

Iterative query resolution



DNS servers return the address of the "next" DNS server

DNS caching

DNS servers cache responses to former queries

- And your client and the application

Authoritative servers associate a lifetime to each record

- Time-To-Live (TTL)

DNS records can only be cached for TTL seconds

- The record is cleared after TTL

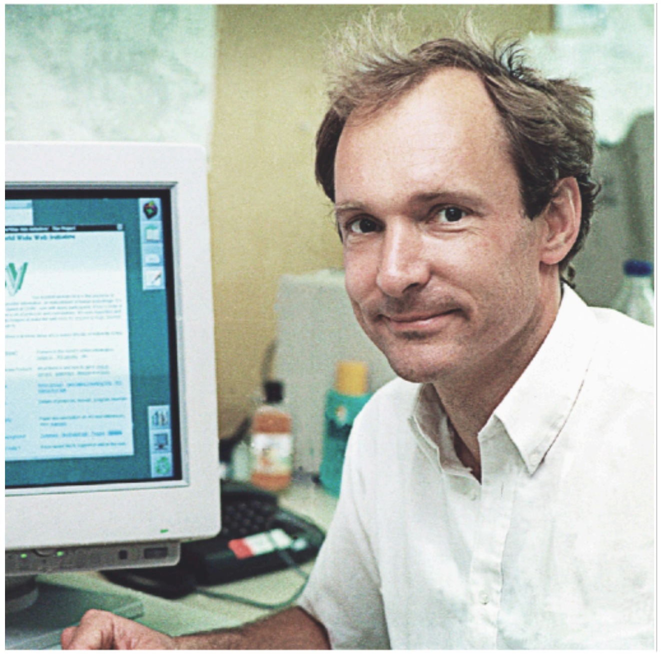
Caching is effective: top-level servers rarely change & popular website visited often

- Practical cache hit rates: ~75%

https://pdos.csail.mit.edu/papers/dns:ton.pdf

The Web

World Wide Web



The web as we know it was founded in ~1990, by Tim Berners-Lee, physicist at CERN

His goal

- Provide distributed access to data

The World Wide Web (WWW)

- A distributed database of "pages" linked together via the Hypertext Transport Protocol (HTTP)

What the web enables?

Self-publishing on the web is easy

- Independent, free, and accessible to everyone

People were not looking for technical perfection

- Little interest in collaborative or idealistic endeavor

People essentially want to make their mark and find something neat

Three key components of the web

Our focus

↓

Infrastructure	Content	Implementation
Client/browser Servers Proxies	Objects (text, images, videos) organized in websites (as a collection of objects)	URL: name content HTTP: transport content

Uniform resource locator (URL)

Refers to an Internet resource

Default to protocol's standard
(HTTP: 80, HTTPS: 443)

protocol://hostname[:port]/directory_path/resource

- protocol: HTTP(s), FTP, SMTP…
- hostname: Domain names, IP address
- directory_path/resource: Identify the resource on the destination

HTTP

A simple synchronous request/reply protocol

Layered over a bidirectional byte stream

- Typically TCP, but QUIC is ramping up

Text-based (ASCII)

- Human readable, easy to reason about

Stateless

- Maintains no information about past client requests

Protocol

Performance

HTTP request

method <sp> URL <sp> version	<cr><lf>
header field name: value	<cr><lf>
...	
header field name: value	<cr><lf>
<cr><lf>	
Body	

HTTP request

method <sp> **URL** <sp> **version**	<cr><lf>
header field name: value	<cr><lf>
...	
header field name: value	<cr><lf>
<cr><lf>	
Body	

Method

GET: return resource

HEAD: return headers only

POST: send data to server (forms)

URL

Relative to server (e.g., /index.html)

Version

1.0, 1.1, 2.0, 3.0

HTTP request

method <sp> URL <sp> version	<cr><lf>
header field name: value	<cr><lf>
...	
header field name: value	<cr><lf>
<cr><lf>	
Body	

Header fields names

- Authorization info
- Acceptable document types/encoding
- From (user email)
- **Host (identify the server to which the request is sent)**
- If-Modified-Since
- Referrer (cause of the request)
- User Agent (client software)

Multiple domain names can map to the same IP

Name	DNS ⟷	IP address	
www.google.com		142.251.209.142	Names can be mapped to more than one IP
www.google.com		142.250.179.206	
www.upb.de		131.234.142.33	IPs can be mapped by more than one name
www.uni-paderborn.de		131.234.142.33	

The host header indicates to the server (given by the IP) the desired domain know (known as virtual hosting)

Virtual hosting

One IP address hosts multiple websites

Connect	openssl s_client -crlf -quiet -connect linwang.info:443
Request	GET / HTTP/1.1 Host: linwang.info
Reply	HTTP/1.1 200 OK Date: Mon, 15 Jan 2024 14:03:21 GMT+1 Server: nginx/1.14.0 (Ubuntu) <head>... <title>Lin Wang's Home Page</title> ...

linwang.info → Resolved through DNS → 134.209.197.20

Virtual hosting

One IP address hosts multiple websites

Connect openssl s_client -crlf -quiet -connect edgecomp.org

Resolved through DNS

134.209.197.20

Request

```
GET / HTTP/1.1
Host: edgecomp.org
```

Reply

```
HTTP/1.1 200 OK
Date: Mon, 15 Jan 2024 14:03:21 GMT+1
Server: nginx/1.14.0 (Ubuntu)

<head>…
<title>Edge Computing</title>
…
```

HTTP response

version <sp> status <sp> phrase	<cr><lf>
header field name: value	<cr><lf>
…	
header field name: value	<cr><lf>
<cr><lf>	
Body	

HTTP response

version <sp> **status** <sp> **phrase**	<cr><lf>
header field name: value	<cr><lf>
…	
header field name: value	<cr><lf>
<cr><lf>	
Body	

Status

1XX: informational

2XX: success

- 200: OK

3XX: redirection

- 301: moved permanently
- 303: moved temporarily
- 304: not modified

4XX: client error

- 404: not found

5XX: server error

- 505: not supported

HTTP response

version <sp> status <sp> phrase
header field name: value
...
header field name: value
<cr><lf>
Body

Header fields names

- Location (for redirection)
- Allow (list of methods supported)
- Content encoding (e.g., gzip)
- Content-Length
- Content-Type
- Expires (caching)
- Last-Modified (caching)

HTTP is stateless

Advantages	Disadvantages
Server-side scalability Failure handling is trivial	Some applications need state! (Example: shopping cart, user profiles, tracking)

How to maintain state in a stateless protocol?

Cookies

HTTP makes the client maintain the state

Client stores small state (on behave of the server X)

Client sents state in all future requests to X

Can provide authentication

Cookies example

Request

GET / HTTP/1.1
Host: www.google.de

Reply

HTTP/1.1 200 OK
Date: Mon, 15 Jan 2024 14:03:21 GMT+1
Cache-Control: private, max-age=0
Content-Type: text/html, charset=ISO-8859-1
Server: gws

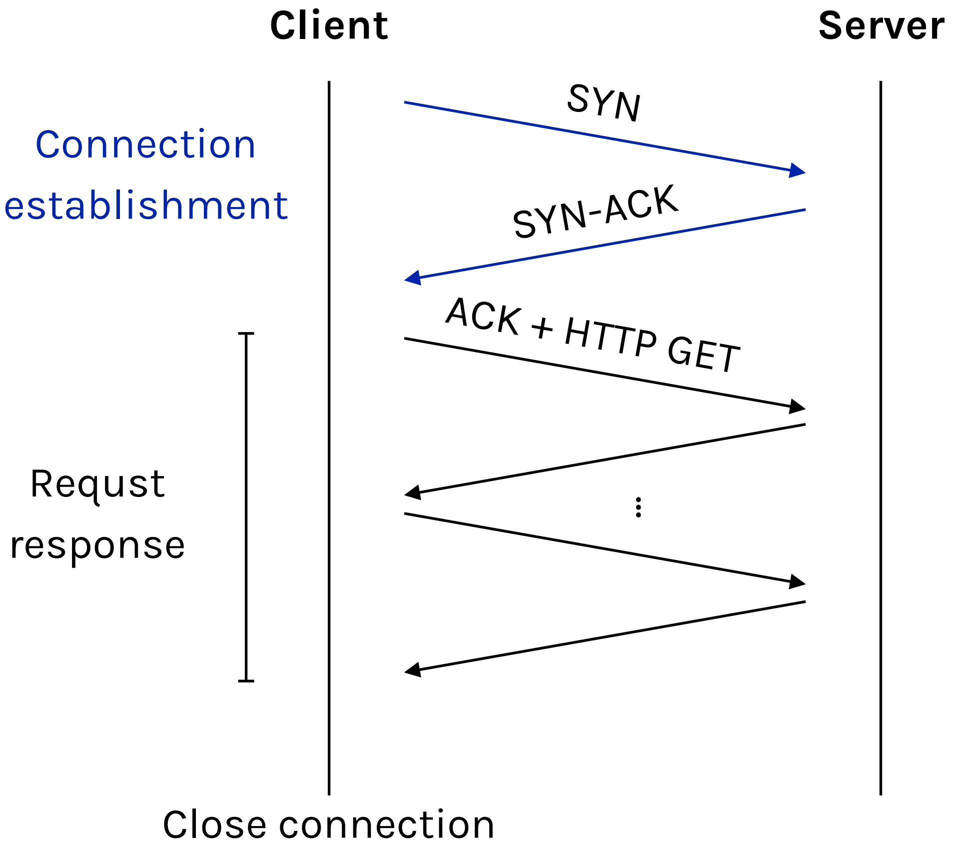
Set-Cookie:
NID=79=g6lgURTq_BG4hSTFhEy1gTVFmSncQVsy
TJI260B3xyiXqy2wxD2YeHq1bBlwFyLoJhSc7jmcA
6TlFIBY7-
dW5lhjiRiQmY1JxT8hGCOtnLjfCL0mYcBBkpk8X4
NwAO28; expires=Mon, 30-Jan-2024 14:20:30
GMT+1; path=/; domain=.google.de; HttpOnly

Browser will relay this value in following requests →

Performance goals

	User	Network operator	Content provider
Goal	Fast downloads High availability	No overload	Happy users Cost-effective infrastructure
Solution	Improve HTTP to compensate for TCP weakspots	Caching and replication	

TCP weakspots



Naive HTTP

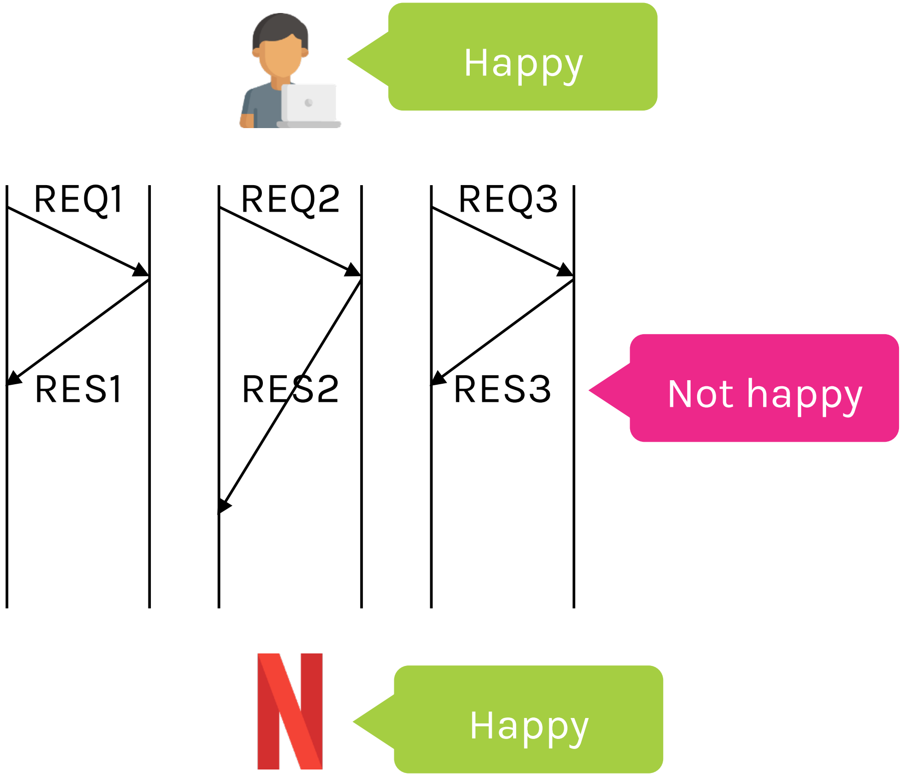
Most web pages have multiple objects

- Text, images, videos, etc.

Naive HTTP opens one TCP connection for each object

- TCP establishment for each connection (one RTT per connection)
- HTTP request/response (one RTT per object)
- Fetching n objects requires around $2n$ RTTs

Solution: parallel TCP connections



Why?

Persistent connections (HTTP/1.1)

Avoid overhead of connection set-up and teardown

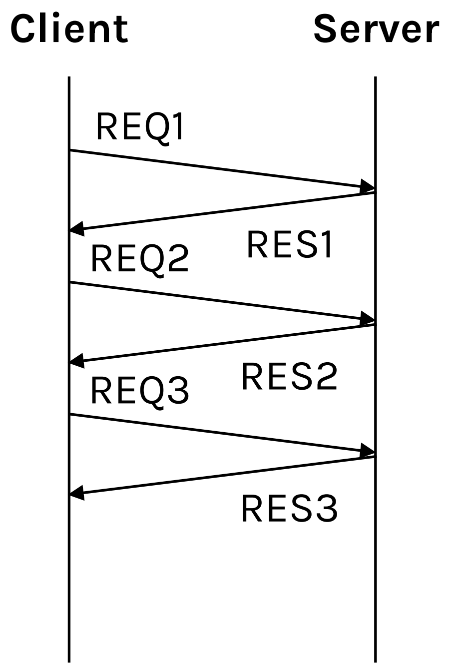
- Clients or servers can tear down the connection

Allow TCP to learn more accurate RTT estimate

- Hence more precise timeout value

Allow TCP congestion window to increase

- Therefore higher bandwidth



Persistent connections (HTTP/1.1)

Avoid overhead of connection set-up and teardown

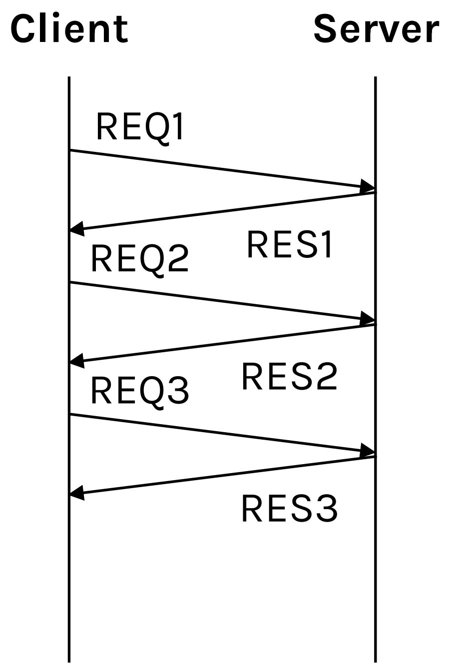
- Clients or servers can tear down the connection

Allow TCP to learn more accurate RTT estimate

- Hence more precise timeout value

Allow TCP congestion window to increase

- Therefore higher bandwidth



Head-of-Line (HoL) blocking at the request leve

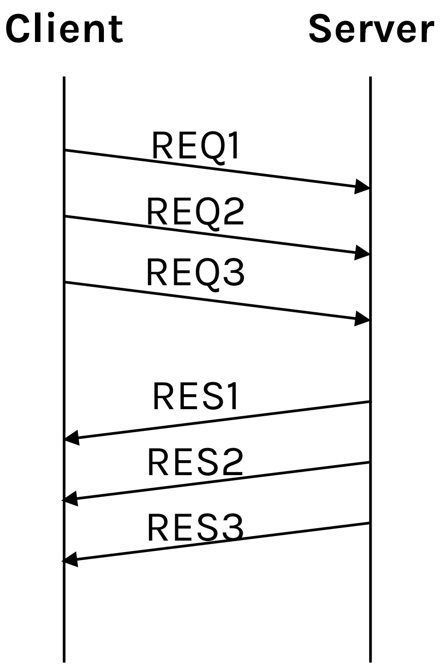
Request pipelining (HTTP/2.0)

Multiple requests can be issued concurrently, without waiting

- Avoid HoL at the request level
- Still HoL at the packet level!
- Multi-streams in QUIC address this issue

Batch requests and responses to reduce the number of packets

- Multiple requests may be packed into one TCP segment



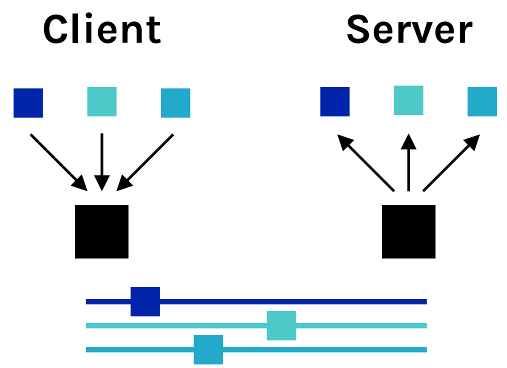
Multi-stream with QUIC and HTTP/3.0

Multiple requests can be issued concurrently, without waiting

Concurrent requests are allocated to different streams

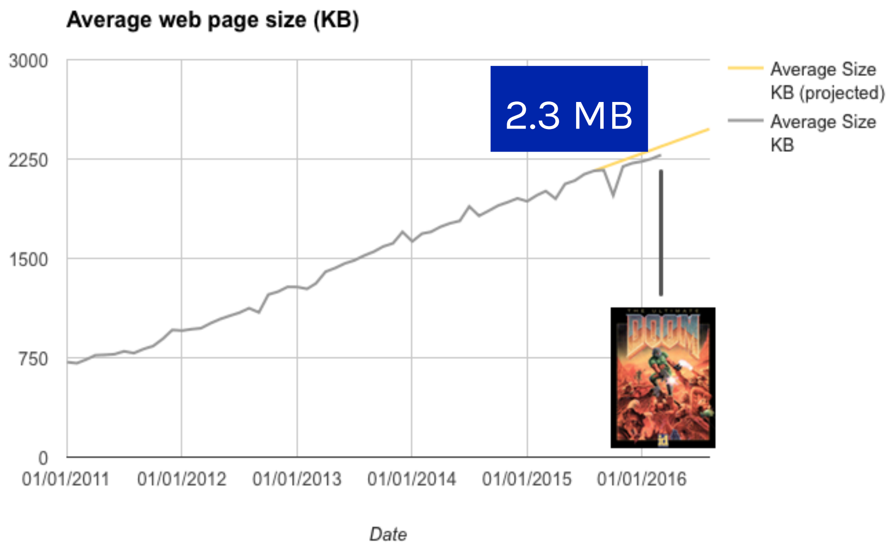
- No HoL at both the request and the packet level

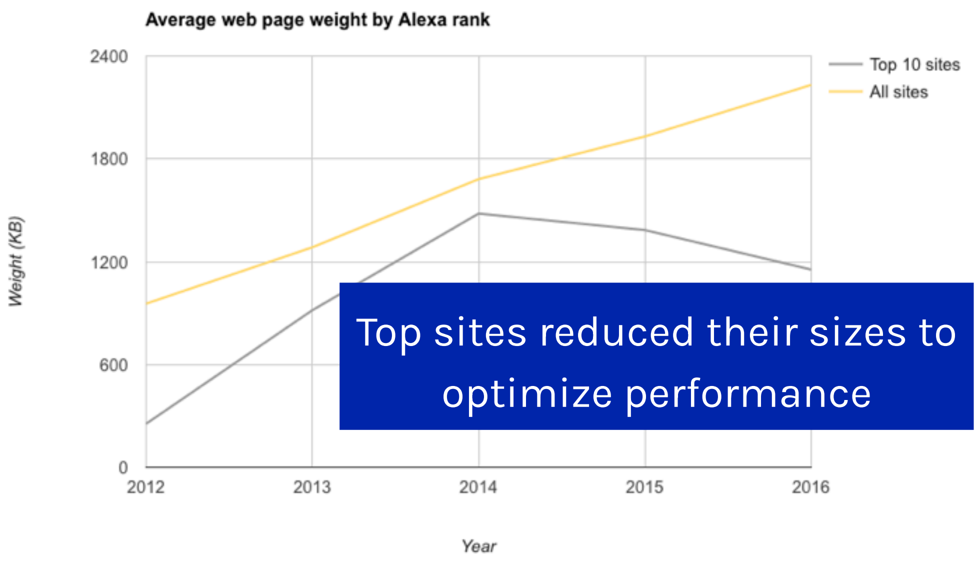
Basis for HTTP/3.0



https://en.wikipedia.org/wiki/HTTP/3

Average webpage size





Caching

Think how often you request the Google logo per day vs. how often it actually changes

Caching saves time for your browser and decreases network and server load

"Uncachable" objects

- Dynamic data: stock price, scores,...
- Scripts: results based on parameters
- Cookies: results may be based on past data
- SSL: cannot cache encrypted data
- Advertising: wants to measure # of hits ($$$)

Limiting cache staleness

Server hints when an object expires (kind of TTL)

- Also the last modified date of an object

Client conditionally requests a resource using "if-modified-since" header in the HTTP request

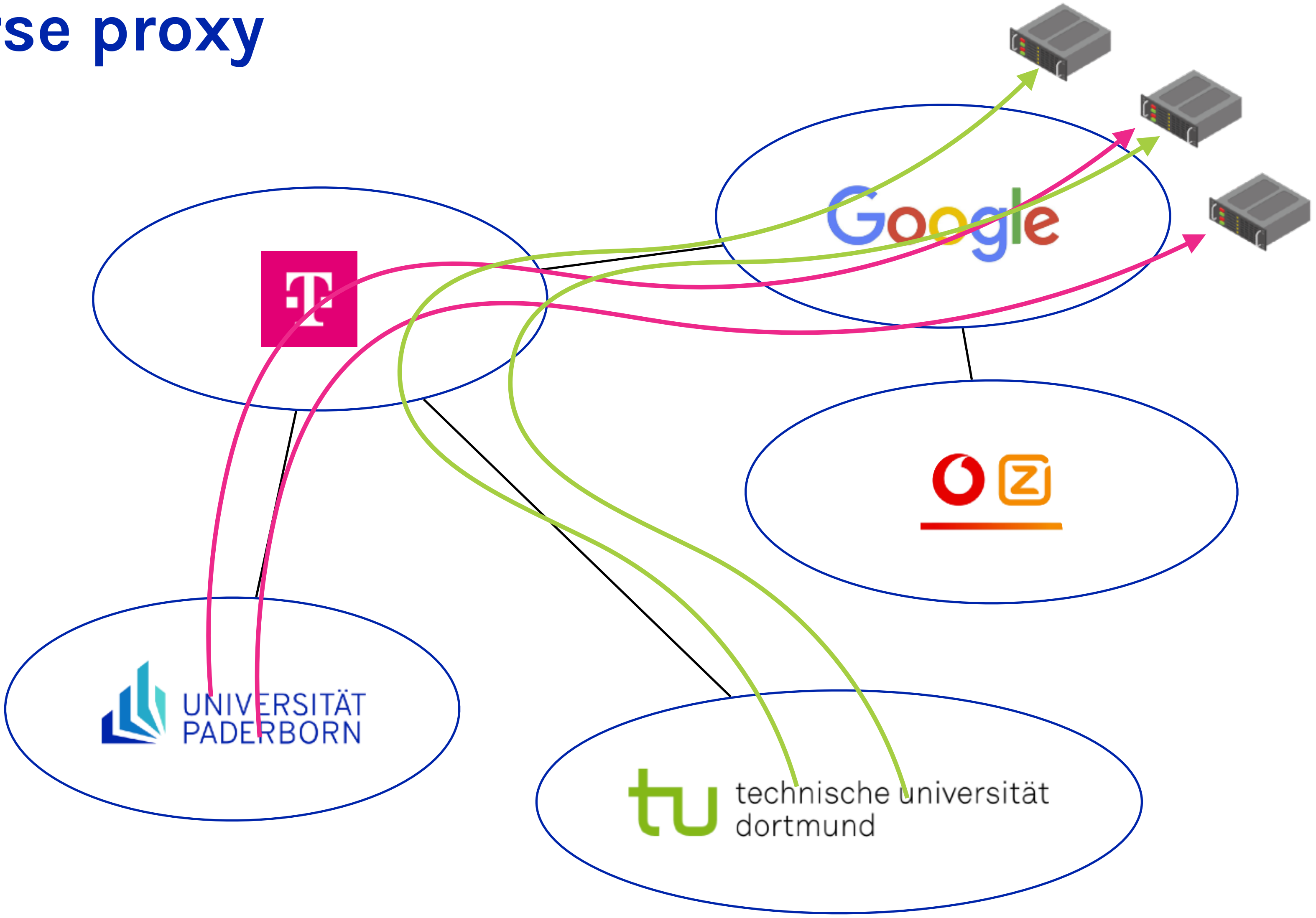
Server compares this against "last modified" time of the resource and returns

- Not modified if the resource has not changed
- OK with the latest version

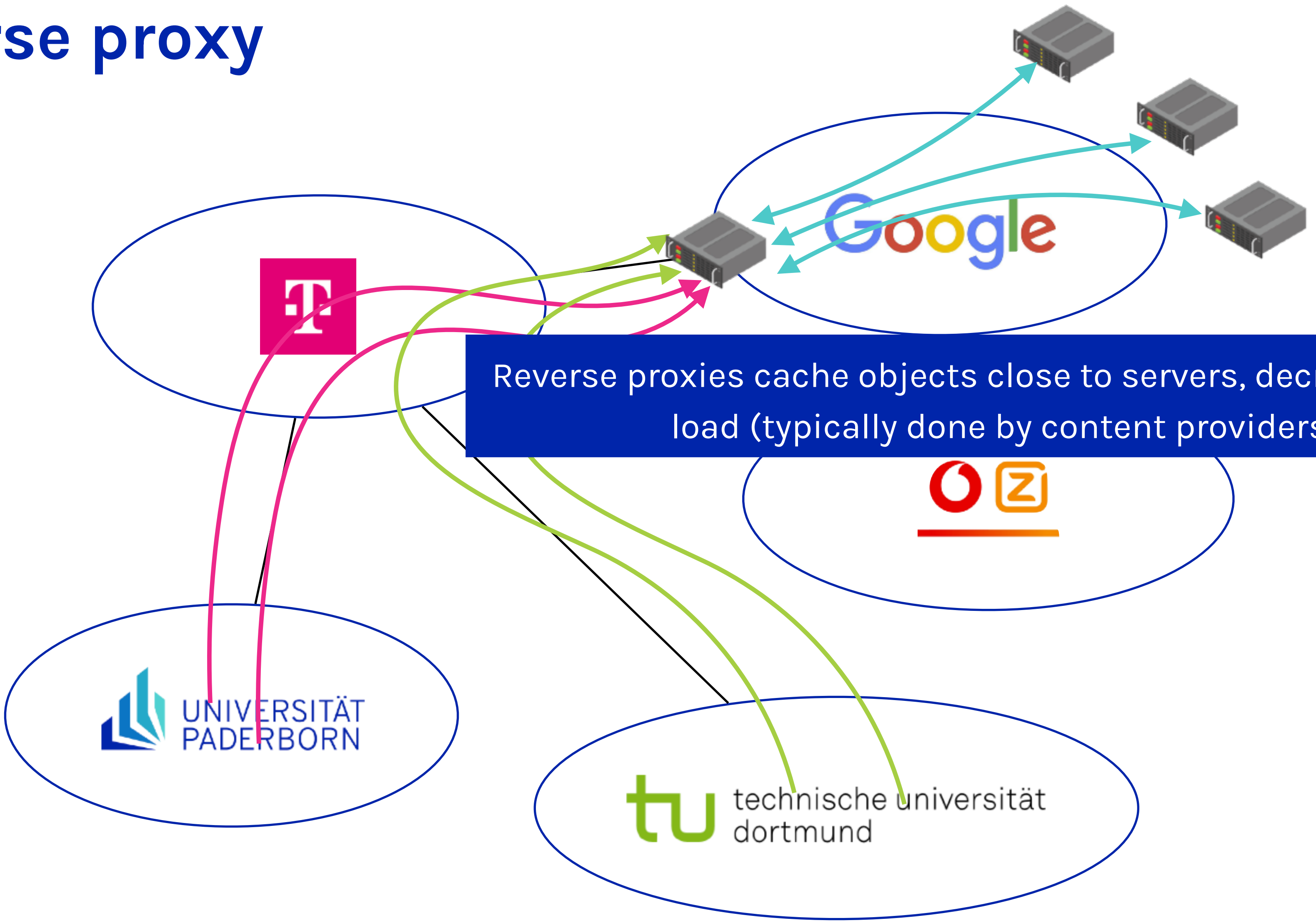
Caching locations

Client	Browser cache
Close-to-the-client	Forward proxy Content Delivery Network (CDN)
Close-to-the-destination	Reverse proxy

Reverse proxy

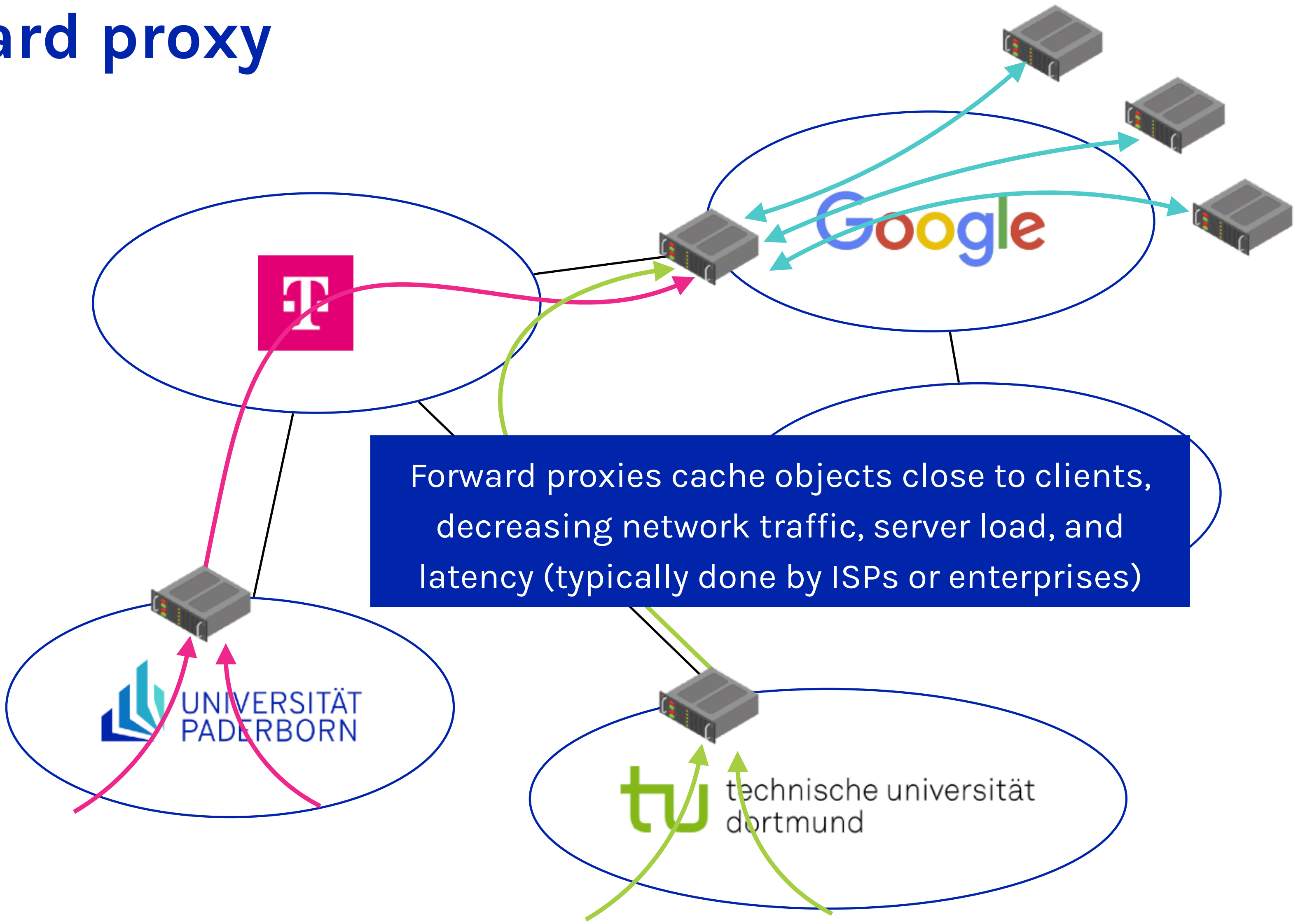


Reverse proxy



Reverse proxies cache objects close to servers, decreasing their load (typically done by content providers)

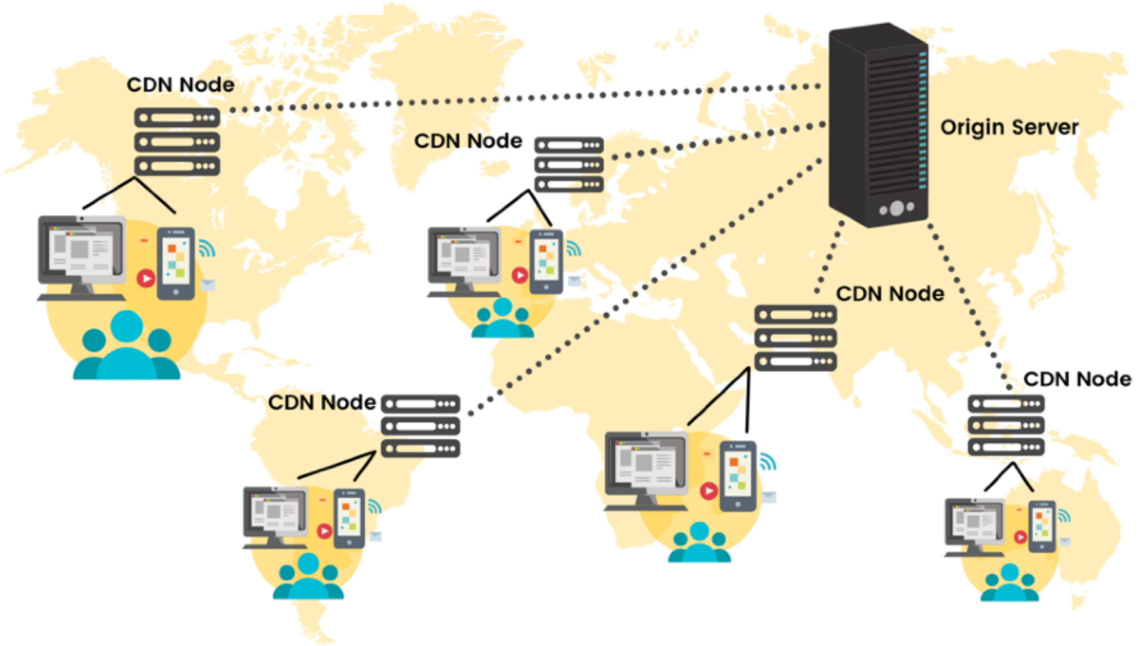
Forward proxy



Forward proxies cache objects close to clients, decreasing network traffic, server load, and latency (typically done by ISPs or enterprises)

Content delivery network (CDN)

Akamai CLOUDFLARE



- Spreads load on servers across multiple data centers
- Places content closer to clients, only way to beat the "speed of light"
- Helps speed up uncachable content (from closer)

Pull caching: direct result of client requests

Push replication: when expecting high access rate

Summary

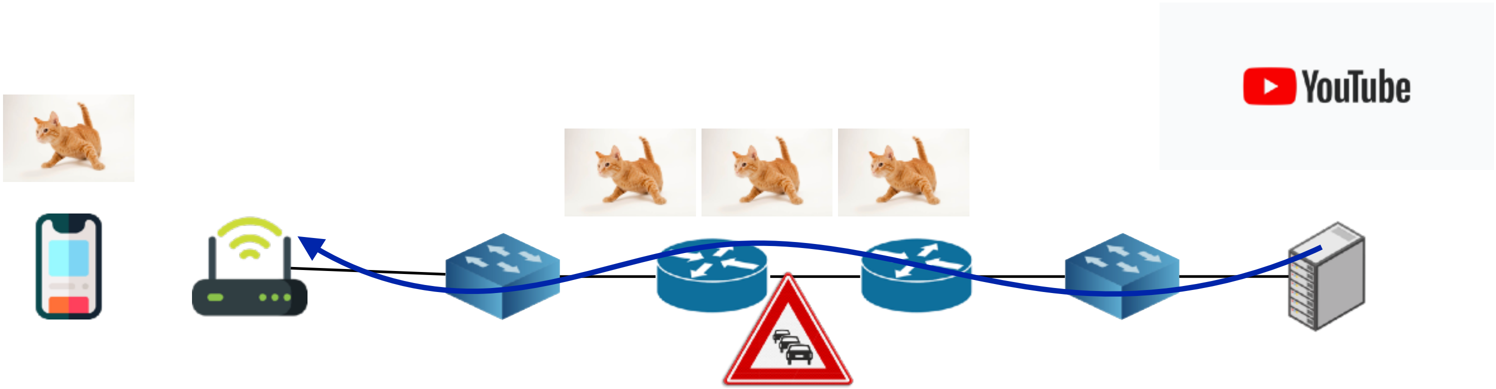
Domain Name System (DNS)

- Address hierarchy
- Authority hierarchy
- DNS server hierarchy
- DNS records
- Recursive vs. iterative query resolution
- Caching

The web

- WWW
- URL
- HTTP
- Performance
- Caching and CDN

Next time: application layer



How to ensure smooth video streaming under **unpredictable** network conditions?

Further reading material

Andrew S. Tanenbaum, David J. Wetherall. Computer Networks (5th edition).

- Section 7.1 DNS---The Domain Name System
- Section 7.3 The World Wide Web
- Section 7.5 Content Delivery

Larry Peterson, Bruce Davie. Computer Networks: A Systems Approach.

- Section 9.1.2 World Wide Web (HTTP)
- Section 9.1.3 Web Services
- Section 9.3.1 Name Service (DNS)