DNS - Domain Name Services

Design and Implementation of a High Performance Domain Name Service on Commodity Hardware

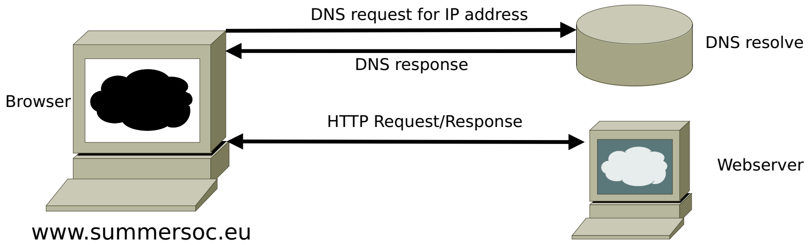
Florian Heinz (OTH Regensburg)
Martin Kluge (Vautron Rechenzentrum AG)

florian.heinz@oth-regensburg.de
martin.kluge@vautron.de

Domain Name System

The Internet Domain Name System is a large, hierarchical, distributed database.

- Used to look up information associated with domain names like **www.summersoc.eu**
- For example: Convert the domain name into an IP address for connecting to the webserver
- Other uses: Look up mail servers, anti-spam information, SIP endpoints and much more...



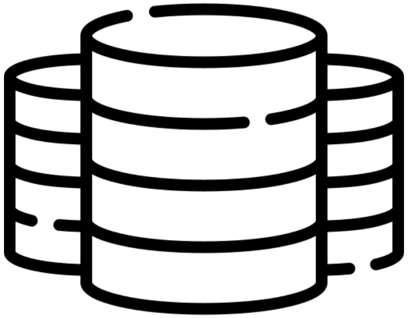
Domain Name System: Zone example.com

example.com.	IN	SOA	ns1.example.com. root.example.com. 2024062301 10000 1800 1209600 3600
example.com.	IN	NS	ns1.provider.net.
example.com.	IN	NS	ns2.provider.net.
www.example.com.	IN	A	192.168.31.37
mail.example.com.	IN	A	192.168.23.42
mail2.example.com.	IN	A	192.168.24.42
example.com.	IN	MX	10 mail.example.com.
example.com.	IN	MX	20 mail2.example.com.
example.com.	IN	TXT	"v=spf1 mx a ptr ?all"

Domain Name System

Estimated amount of data stored in the DNS:

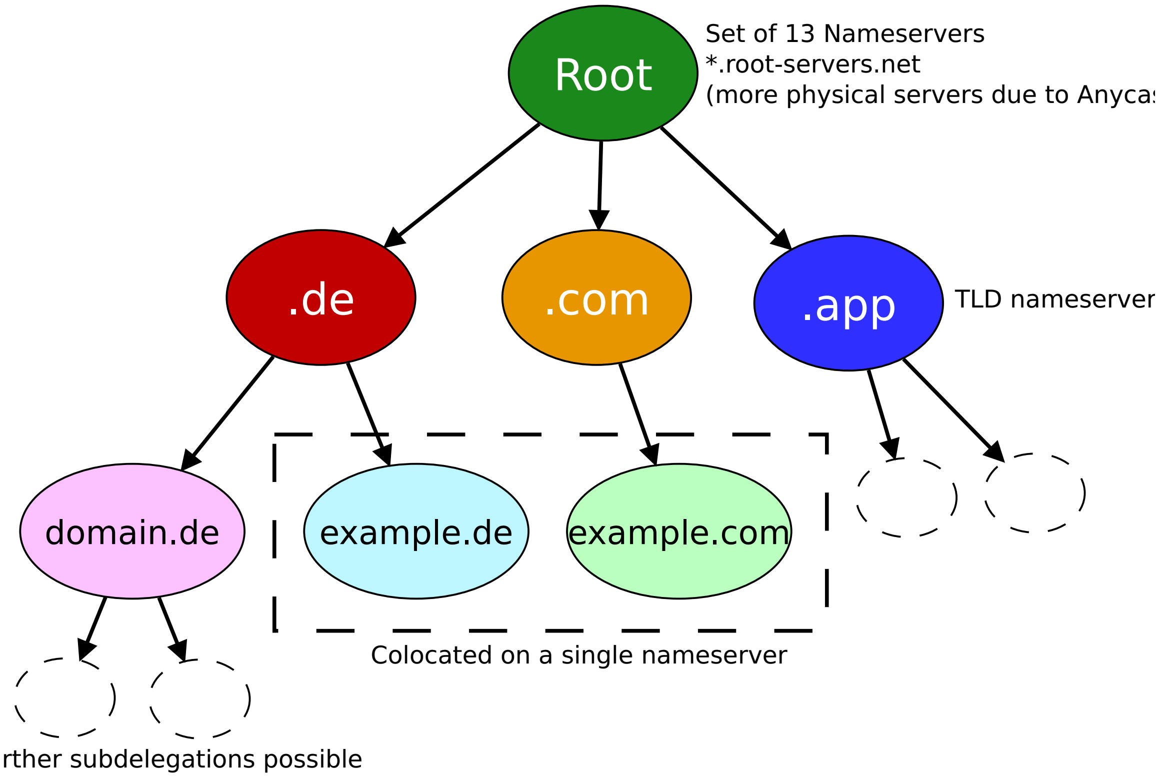
- Domain names registered: **~359 million**
- Average DNS records per Domain Name: **10-20**
- Total: **~4-7 billion records**



→ Large amount of information

→ High availability crucial, otherwise risk of service interruption

Domain Name System: Hierarchy

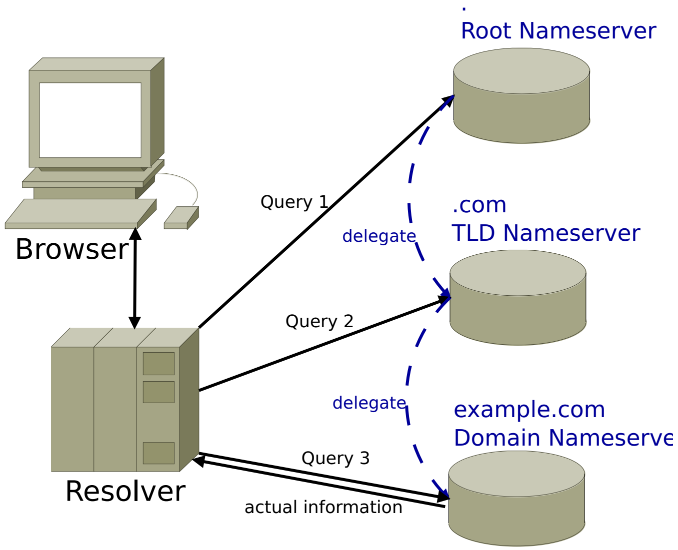


Domain Name System: Distributed database

Hierarchical tree structure:

Example: Lookup of www.example.com

- DNS-Root (Root nameservers)
    - com (TLD nameservers)
        - example (Domain nameserver)
            - www
                - A record: 192.168.95.65



Domain Name System: System critical

DNS servers are critical infrastructure: On failure, many services are not available, e.g., e-mail, web services and more.

Strategies for high-availability:

- Redundant name servers (typical number: 2-4)

  → Attack all nameservers
- Traffic filtering on network or dns request floods

  → Craft packets that cannot be distinguished from legitimate queries
- Using fast caches for dns requests

  → Ask for different domain names to prevent successful caching

However: A skillful attacker can overcome these strategies

Domain Name System: Resilience

Optimal strategy:

The nameserver is able to answer <u>all</u> queries that it receives over the network

→ Shift bottleneck from the processing of queries to the available network bandwidth

Goal: Saturate outgoing network interface with answers to incoming queries

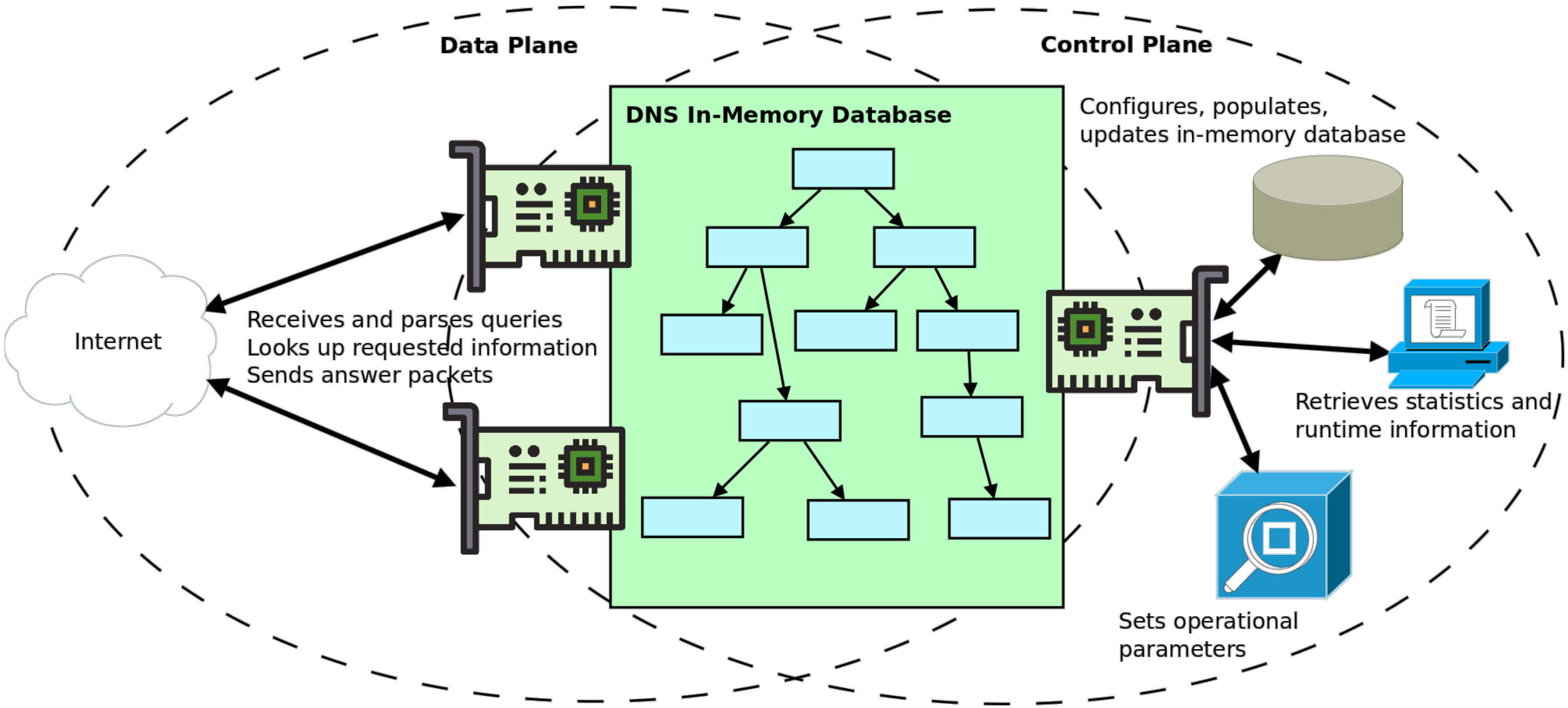
Domain Name Server: Design

Design concepts:

- Division of system into high-speed data plane and control plane
- In-memory radix-tree based structure for fast lookup of domain names
- In-advance preparation of DNS answer packets
- Stateless implementation for answering TCP-based queries

→ Saturate 10-GBit/s link under worst-case conditions with inexpensive commodity hardware (~500 EUR)

Data Plane vs. Control Plane



Data Plane vs. Control Plane

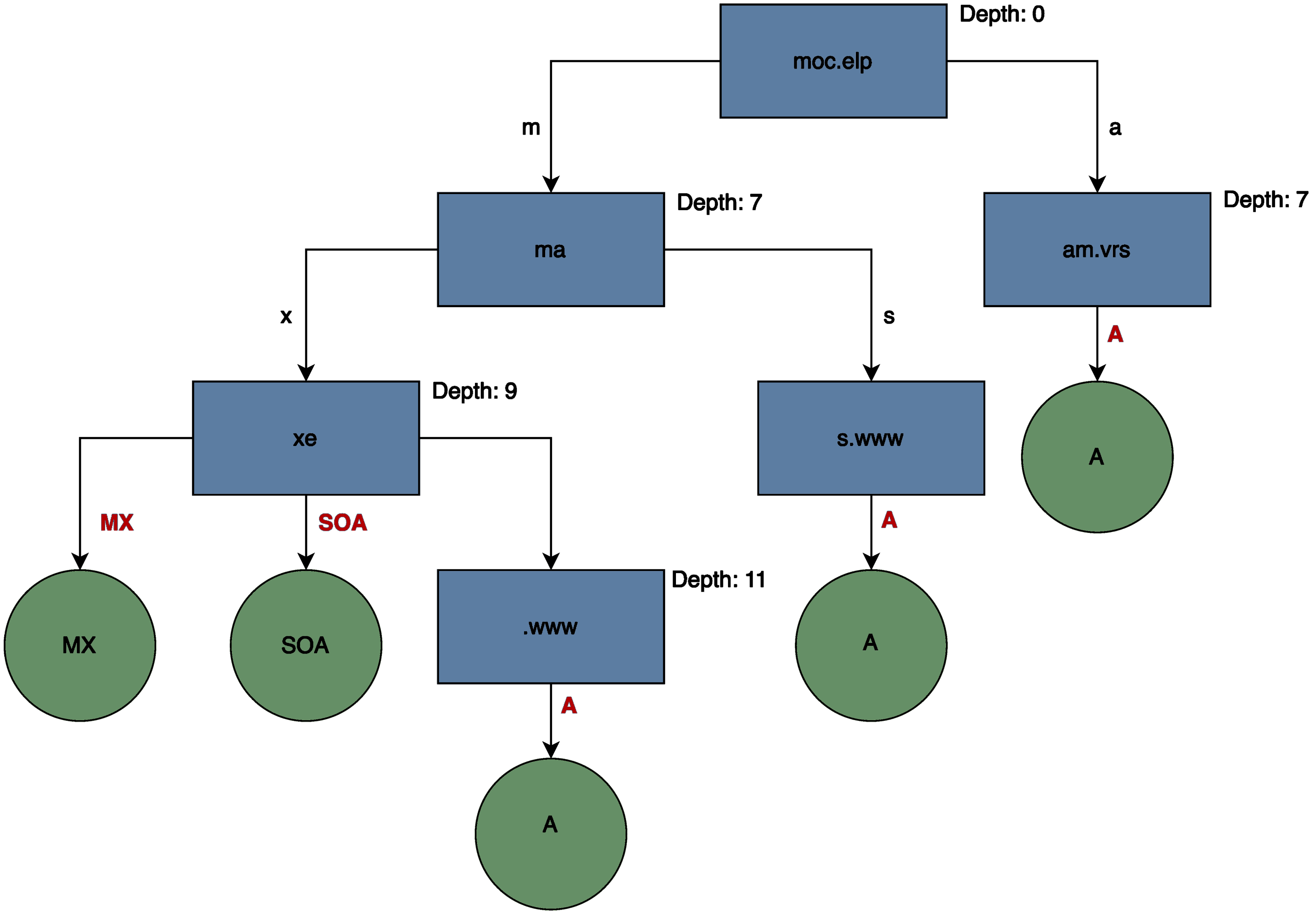
Data Plane

- Uses Data Plane Development Kit (dpdk.org)
- Direct NIC access (without OS layer)
- Implementation of all network layers (up to layer 2)
- Fast read-only access to the DNS information tree
- Avoids any blocking operations

Control Plane

- Uses standard OS networking stack
- Configures the operational parameters
- Provides read/write interface to the DNS information tree
- Operations not time critical

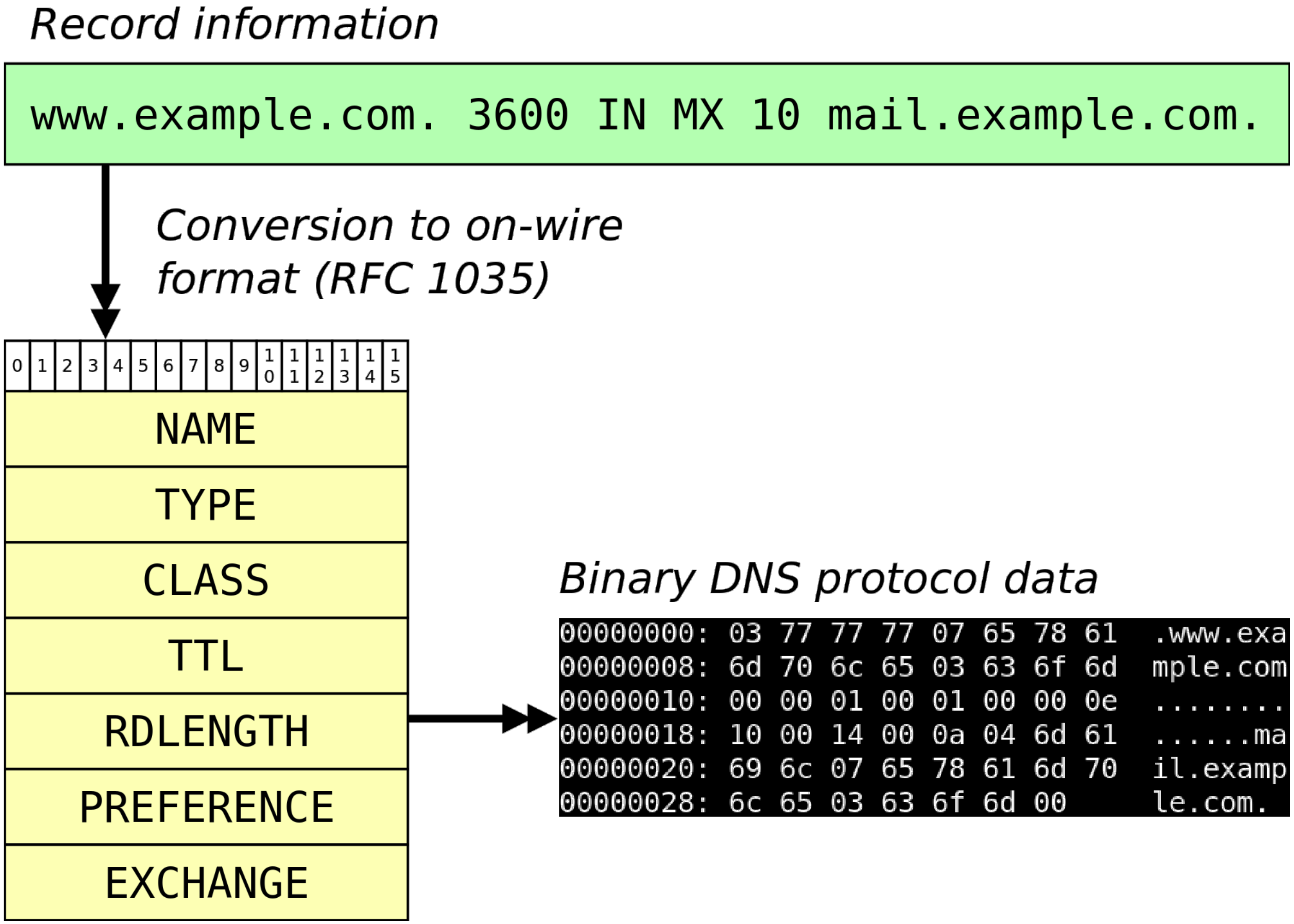
DNS information tree:



DNS information tree

- Central element, access from data plane (ro) and control plane (rw)
- Modified radix tree
- Contains records for all served domains in-memory
- Designed for one-pass traversal:
  - Tracks wildcards and delegations
  - Pointers to CNAME referrals
- Built and maintained by Control Plane:
  - from secondary sources, e.g., Databases or Zone Files
  - via Network APIs
  - from LUA Scripts

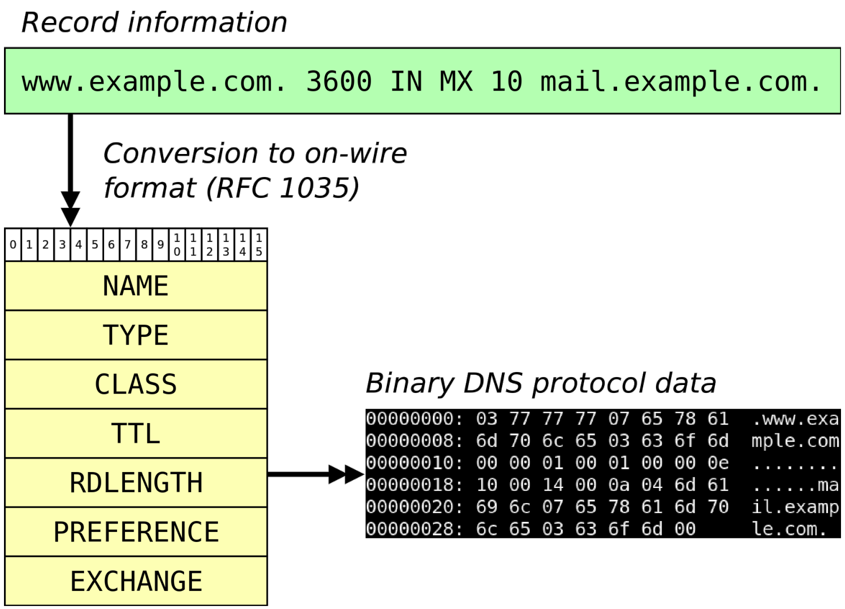
DNS information tree: Records



DNS information tree: Records

Possible (inefficient) strategy:
(e.g. PowerDNS with MySQL backend)

- Wait for request
- Fetch dataset from database
- Convert to on-wire format
- Send answer packet

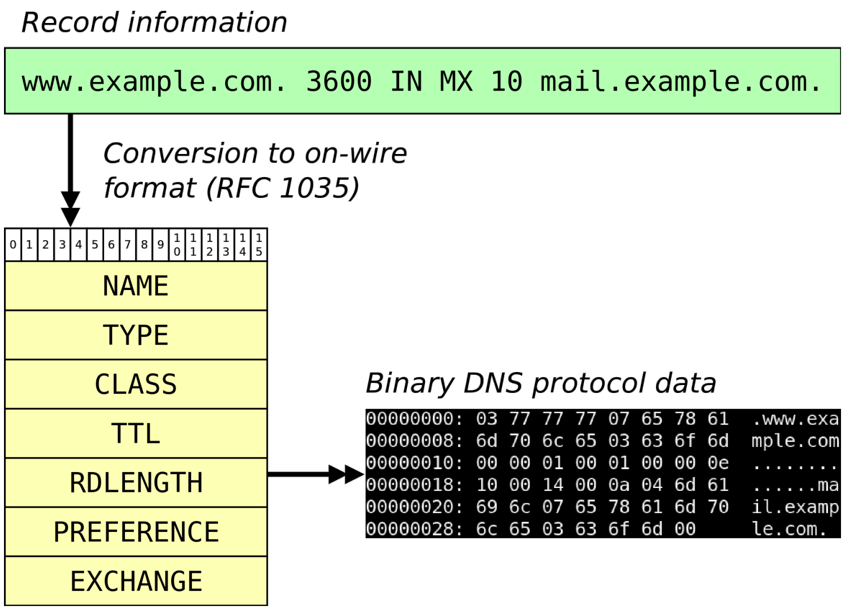


DNS information tree: Records

Implemented strategy:

- Fetch all records (CP)
- Build DNS information tree (CP)
- Build on-wire buffer for all records (CP)

- Wait for request (DP)
- Look up information in tree (DP)
- Send buffer (DP)



ℹ Time-Memory Tradeoff!

Domain Name System: TCP requests

DNS by default packet-based (UDP)
One request packet → one answer packet

Maximum allowed UDP payload: 512 bytes

DNS answers often much larger today (DNSSEC!)

Stream based answers (TCP) more and more needed

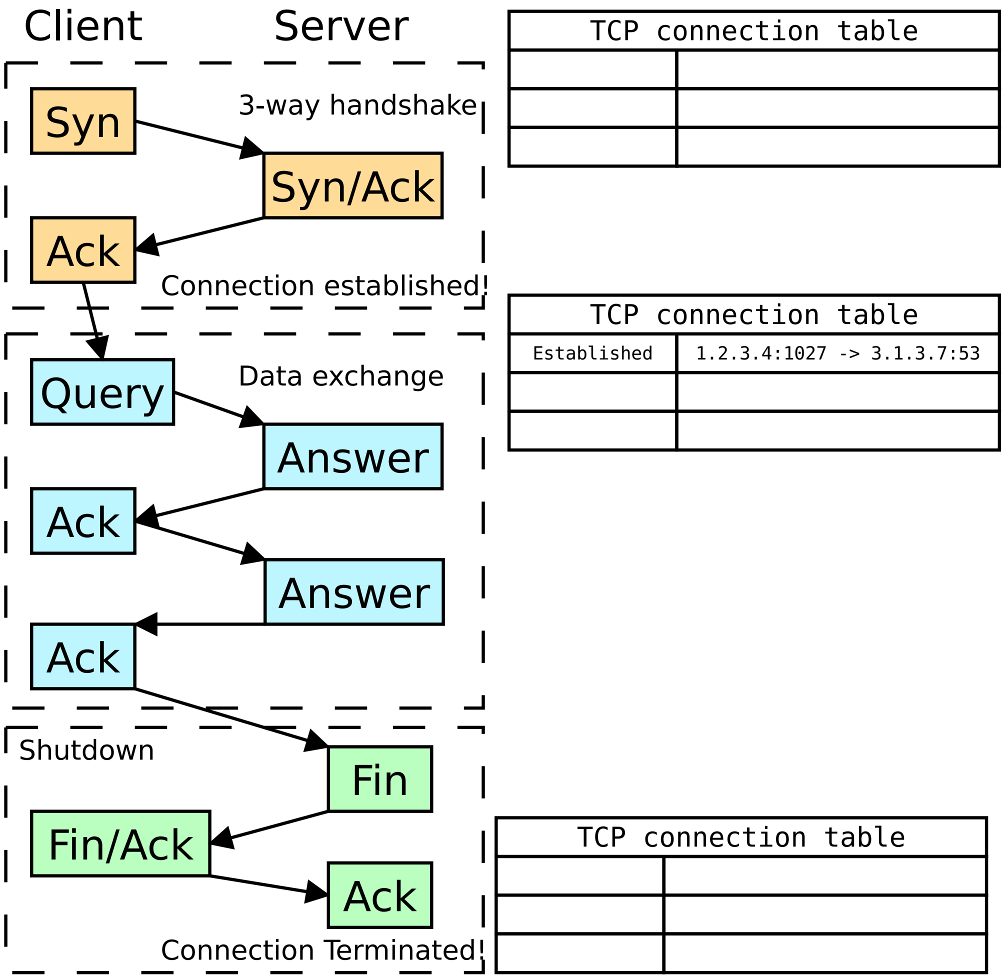
Domain Name System: TCP requests

TCP is a **stateful** protocol:

- All connections tracked
- Complex state machine
- High RAM usage on attack

but:

- Prohibits spoofing
- Reliable data transfer
- Automatic retransmissions
- .. and much more



Domain Name System: TCP requests

Most TCP features not needed for DNS communication

💡 Idea: Stateless TCP for DNS requests

- Receive Syn → Send Syn-Ack
- Receive Ack without data → Ignore
- Receive Query → Send <u>all</u> answer packets at once
- Receive Fin → Send Ack

Not all edge cases covered (e.g. fragmented queries), but works well in practice

Improvement: Provide a full-fledged TCP stack and switch to light on demand

Evaluation:

System is deployed at an internet infrastructure provider:

- ~1.040.000 Zones (i.e. Domains) served
- ~8.470.000 Records in total
- DNS Information Tree size: 9.8 GB
- DNS Information Tree depth: 28 Levels
- ~1500 requests/second in normal operation

Evaluation setup:

- Copy of production data
- Direct connection to query client with 10GBit/s link
- Worst-case queries on dataset
- Inexpensive commodity hardware:

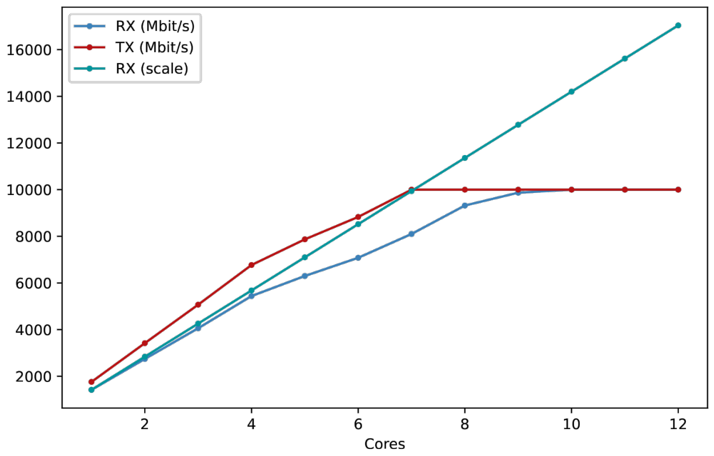
Mainboard:	MSI MAG B550
CPU:	AMD Ryzen 5600X 6-Core
RAM:	64GB DDR4 2666 MT/s
NIC:	Intel X540 10G

Total cost: ~500€ (excl. VAT)

Evaluation results:

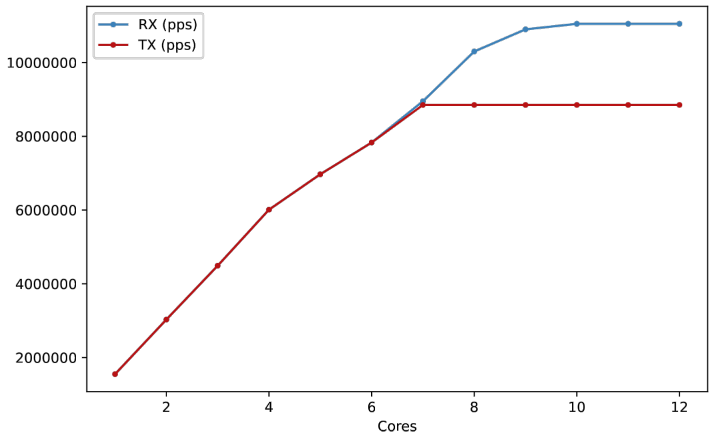
Bandwidth:

- Scales almost linearly with number of cores
- TX saturated with 7 cores
- RX saturated with 10 cores



Packets:

- ~11 mio. queries processed with 10 cores
- ~8.8 mio. responses sent with 7 cores



Summary:

- Domain Name System crucial for a working internet
- Domain Name Servers should be resilient against attacks
- Traditional approaches often have to perform too many or expensive tasks at query time
- This approach: Minimize work at query time:
  - Data plane vs. Control Plane
  - All information stored in-memory (radix-tree)
  - Prepare all responses in advance
  - Stateless TCP handling
- Result: 10GBit/s saturation on commodity hardware

Thanks for Listening!

See you at the poster session for demonstration and questions.