Netzwerke und Kommunikation
B-LS-MI 004
Basisdienste
DNS Verzeichnisdienst

rolf.schmutz@fhnw.ch

FHNW

27. Oktober 2020

n|w

Ziele

- Sie kennen die Aufgaben von DNS
- Sie kennen die Funktionsweise von DNS
- Sie können gezielt DNS-Server abfragen

DNS: Domain Name System

- `http://www.eff.org/`
- `ping www.google.com`
- `mailto: president@whitehouse.gov`
- `telnet kremvax.kreml.ru`

Hostnamen

→ Menschen[a] arbeiten lieber mit *(Host-, Domain-) Namen* als mit IP-Adressen. . . → wir brauchen ein *Telefonbuch*

[a] und auch Programme. . .

DNS: Grundlage

- IP arbeitet *nur* mit *IP-Adressen* und kennt keine *Host/Domainnamen*
- das entspricht genau dem Telefonnetz wo Nummern zu Namen auch zuerst im Telefonbuch nachgeschlagen werden müssen
- genau wie beim Telefonnetz braucht IP keinen DNS/Verzeichnisdienst – der dient nur zur vereinfachten Handhabung für Menschen und/oder Programme

DNS: . . .die Zeit *vor* [3] DNS

- SRI[1] verwaltete eine zentrale `HOSTS.TXT` Datei mit Namen↔IP-Adressen
- der *Namensraum*[2] war *flach*, d.h. nur eine Ebene von Namen möglich
- Änderungen mussten an SRI geschickt werden
- alle "Kunden" mussten regelmässig eine Kopie der Daten erstellen

Nachteile:

- Last und Datenverkehr auf dem zentralen Server
- *Single point of failure*, alles hängt von einem Server ab
- Konsistenz: Änderungen waren nicht gleichzeitig bei allen "Kunden" sichtbar
- *Name clash*, jeder Name musste sorgfältig ausgewählt werden um Kollisionen zu verhindern
- exponentielles Wachstum der Datei

[1] Standford Research Institute: beteiligt an der Entwicklung des Internet

[2] namespace

[3] in der *Steinzeit*: vor 1983

DNS: Factlets

- DNS ist das *Domain Name System*, ein spezialisierter Verzeichnisdienst
- klassisches Client-/Server-Konzept mit UDP[4]
- Definiert in STD13, RFC1034, RFC1035 und *viele* zusätzliche RFCs
- häufigste Anwendung: IP↔Hostname Verzeichnis
- *hierarchisch*[5] aufgebaut mit *Delegationen* von *subtrees* (subdomains, Unterbäumen)
- *verteilte* Datenbank: eine Ebene der Daten wird von einer Partei verwaltet. Kein Server kennt den gesamten Namensraum
- *Replikation*[6] der Daten vom *Master* auf *Slave* Server[7]
- Kommuniziert über UDP für normale Anfragen und über TCP für Replikation

DNS

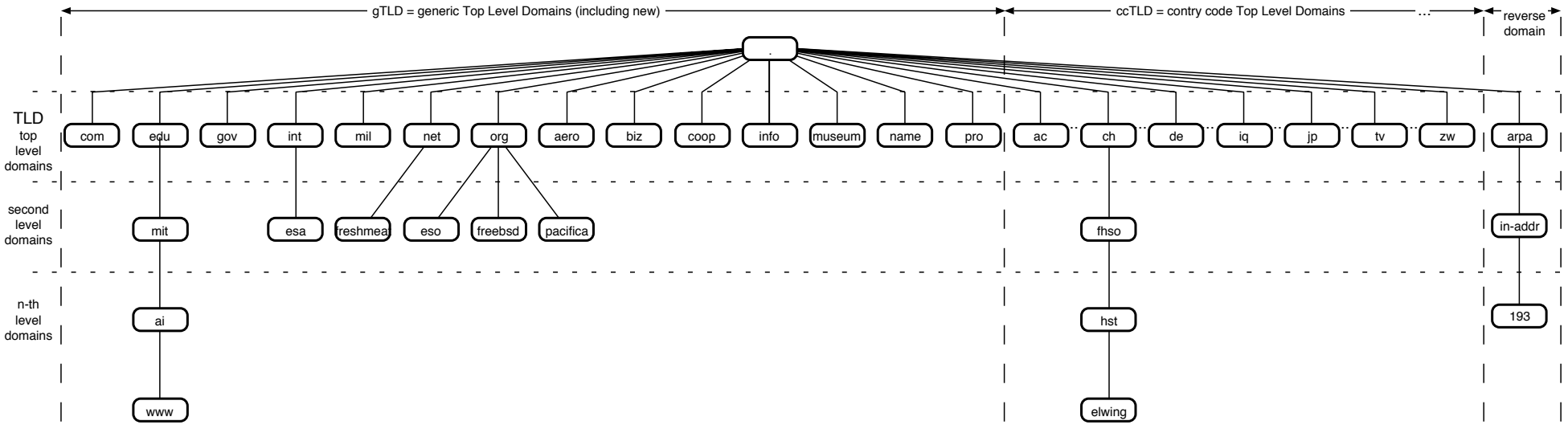
→ scalable, reliable and flexible: Skalierbar, zuverlässig und Erweiterungsfähig

[4] "Telegramm"

[5] wie die "alten" Telefonbücher mit Vorwahl, Ortskreis, etc

DNS: hierarchischer Namensraum

- der *Namenspfad* wird von den Blättern (leaves) her ausgelesen: `www.ai.mit.edu` ("most-specific" zu "least-specific", vergleichen Sie das mit Windows oder UNIX Filesystemen)
- der vollständige Pfadname, *FQDN* (fully-qualified-domain-name) und der relative Name *RDN* (relative-domain-name) entsprechen einem Pfad- und einem Dateinamen



DNS: Traditionelle Top-level Domains

- gTLDs (generic top-level domain):
  - `com` International: Kommerzielle Unternehmen: `www.sun.com`
  - `edu` USA: (Hoch-) Schulen (educational): `www.mit.edu`
  - `gov` USA: Regierung (governement): `www.whitehouse.gov`
  - `int` Internationale Organisationen: `www.esa.int`
  - `mil` USA: Militär: `www.norad.mil`
  - `net` International: Netzbetreiber/Infrastruktur: `www.sprint.net`
  - `org` International: Nichtkommerzielle Organisationen: `www.eff.org`
- Reverse: IP-zu-Hostnamen[8]: `67.242.135.193.in-addr.arpa`
- ccTLDs: country-code TLD: ISO-Code länderspezifische Domains: `ch`, `de`, `us`, `tv`, `jp`, `to`, `tv`, `io`, ...

[8] siehe Slide 10

DNS: ICANN neue Top-Level Domains

Seit 2011 werden von ICANN (fast) beliebige Top-Level-Domains[9] erteilt[10].

- Insbesondere wurden auch viele IDN[11] neu geschaffen[12]
- Dies hat zu einem oft kritisierten Wildwuchs geführt, mit mittlerweilen[13] 1509 Top-Level-Domains (Liste).
- Gerade grosse Firmen mit dem Bedürfnis ihren Namen auf allen Top-Level-Domains zu registrieren haben dagegen protestiert

[9] wie .com, .org, .ch, ...

[10] für mindestens 1 Million $US

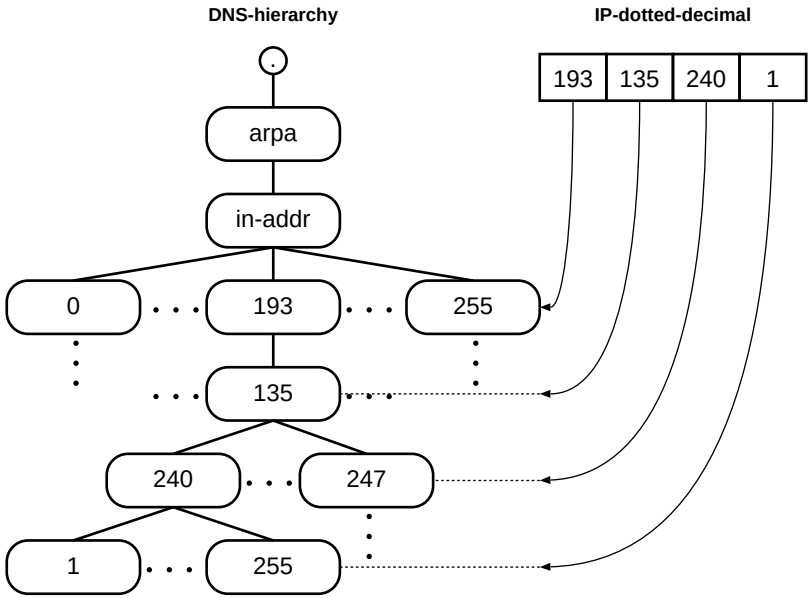
[11] Internationalized Domain Names

[12] vorallem Chinesische

[13] 2020

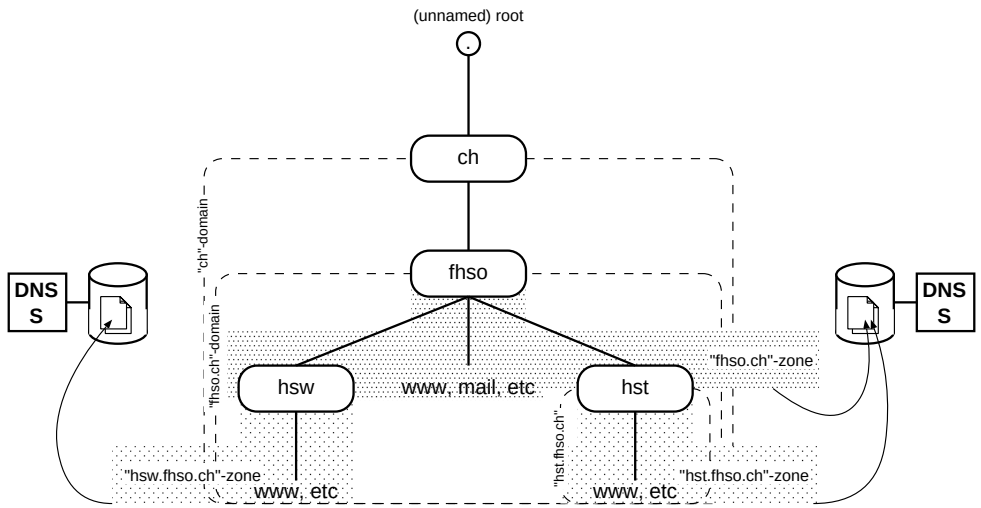
DNS: Reverse Domain

- für Abfragen von IP-Adresse → Domainname wird eine spezielle TLD `in-addr.arpa`[14] verwendet
- die Namen werden ebenfalls von den Blättern (unten) her ausgelesen – deshalb wird eine IP-Adresse "verkehrt herum" dargestellt: `www.fhnw.ch` → `50.3.86.147.in-addr.arpa`



DNS: Verteilte Datenbasis, Zonen und Delegation

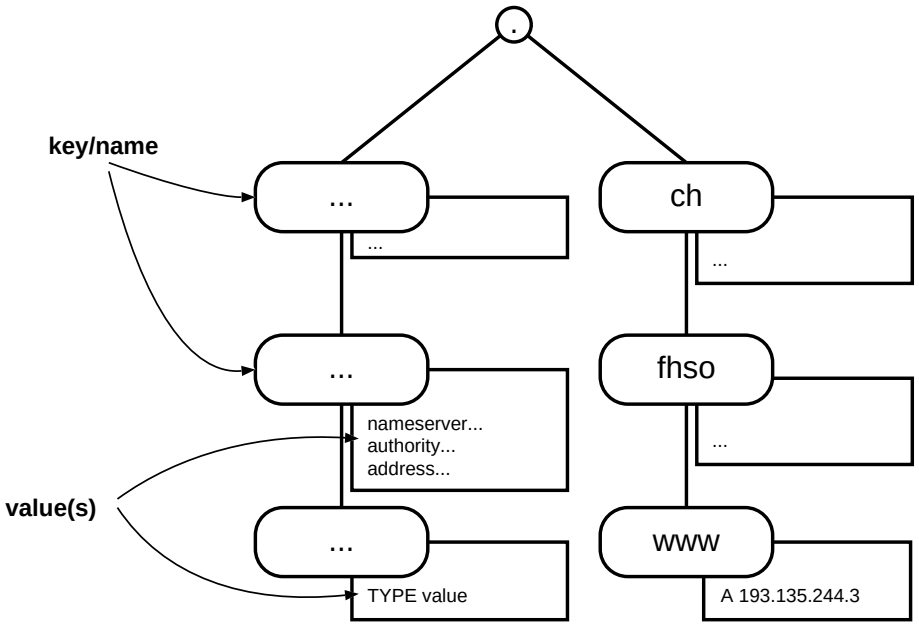
- die Verwaltung *authority* über einzelne Teilbäume (subtree/domain) wird *delegiert* (per NS-record, der Verwalter ist zuständig für die Daten und die korrekte Funktion der Server)
- die Daten werden in *Zonen* (entspricht einer Verzeichnisebene im Filesystem) aufgeteilt und sind deshalb nicht mehr zentral verwaltet
- die *skalierung* (Wachstum) dieses Systems ist fast unbegrenzt möglich
- ein Teilbaum *subtree* wird als *Domain* bezeichnet
- eine Verzeichnisebene wird als *Zone* bezeichnet und wird üblicherweise in einer Datei auf dem Server abgelegt



DNS: Verzeichniseinträge, `Key [Type] Value`

- schlage `Key` im Verzeichnis nach und finde zugehörige `Values`
- *alle* Knoten (=`Key`) in der Verzeichnishierarchie haben zugehörige `Values` – sonst würden sie nicht existieren
- die Paarung `Type Value` wird als *resource-record* (RR) bezeichnet

(vergleichbar mit Telefonbuch: zu einem Namen gibt es den Datentyp "Telefonnummer" und den Inhalt "0318921122")



DNS: Resource Records 1/2

RR-Type	Description	Syntax
A	**A**ddress: name → IP-address	*name* A *ip-address*
PTR	**P**oin**t**e**r**: IP-address → name	*rev-ip-name* PTR *name*
SOA	**S**tart-**O**f-**A**uthority: zone-head	*domain* SOA *origin mail* (serial refresh retry expiration minimum-TTL)
NS	**N**ame **S**erver	*domain* NS *name-of-ns*
MX	**M**ail E**x**change: mail-server	*name* MX *priority name-of-mails*
CNAME	**C**anonical **Name**: alias-name	*alias* CNAME *original*
...	...	...

DNS: Resource Records 2/2

- `SOA`: jede Zone (Verzeichnisebene) muss einen solchen Record haben. Darin werden Verwaltungsdaten (Seriennummer, Abfrageintervall für *Slave*-Server, etc abgelegt
- `NS`: *Delegation*: zeigt auf den zuständigen Nameserver für die Zone in `Key`
- `A`: der "normale" Eintrag: `Key`=Name und `Value`=IP-Adresse
- `PTR`: der "reverse" Eintrag: IP-Adresse auf Namen
- `CNAME`: ein Alias-Name der üblicherweise für *Dienste* (versus Host/Server) verwendet wird
- `MX`: Mailserver für eine Domain
- das `TTL`-Feld bezeichnet für jeden RR die Lebensdauer als Cache-Inhalt
- es können mehrere `Values` eines `Types` für einen `Key` definiert werden

DNS: Tools

normalerweise werden DNS-Abfragen durch das Betriebssystem[15] automatisch erledigt...
zur manuellen Abfrage des DNS-Dienstes gibt es einige Kommandozeilenwerkzeuge[16]:

- `nslookup`: der "Urvater"
  `nslookup [-q=`*`Type`*`] `*`Key`*` [`*`nameserver-to-use`*`]`
  oder einfach
  `nslookup`
  `set type=`*`Type`*
  `Key` (interaktiver Modus)
- `host`: moderner Nachfolger von `nslookup` (UNIX):
  `host [-t `*`Type`*`] `*`Key`*` [`*`nameserver-to-use`*`]`
- `dig` ist ein weiterer Nameserver-Client (UNIX)

[15] genauer: durch `gethostbyname()`

[16] auf den meisten Betriebssystemen finden Sie auch GUI-Tools oder Sie benutzen: http://www.dnsstuff.com/

DNS: Interlude

- finden Sie die IP-Adresse von www.fhnw.ch
- finden Sie den Namen von 64.147.188.3 (versuchen Sie auch explizit `Type=PTR`
- finden Sie die Mailserver von kyoto-u.ac.jp, microsoft.com und apple.com
- finden Sie die Nameserver von "." (root nameservers)
- finden Sie die Nameserver (DNS-Server) für mit.edu
- finden Sie den "echten" Namen von www.netlabs.ch und von postoffice.netlabs.ch

DNS: Server Rollen

ein Nameserver kann verschiedene Rollen annehmen (auch gleichzeitig)

- Authoritative: der Server ist in den NS records angegeben und hat die Zonendaten lokal (Datei, DB) gespeichert. Antworten werden mit dem "aa" (authoritative-answer) bezeichnet
- Forwarding/Caching: der Server erledigt "recursive-queries"[17] für Clients[18] – er ist nicht *authoritative* für die angefragten Daten. Typischerweise finden Sie forwarding-servers in Organisationen oder Providern für "interne" Clients. Die Antworten werden auf dem caching-server zwischengespeichert

Beispiel: dns.fhnw.ch

Authoritative: für `fhnw.ch`
Caching/Forwarding: für alles andere

[17] Slide 20

[18] für die Anfrage auf andere Nameserver nimmt er dann die Client-Rolle an

DNS: Server-Replikation

- in der Server-Konfiguration *für eine Zone wird*
  - *ein* master-Server[19]
  - *0 bis viele* slave-Server

  definiert. Diese werden als NS-records für die Zone erfasst
- von "aussen" ist diese Rollenverteilung unwichtig/nicht eindeutig feststellbar[20]: alle NS-Server sind gleichberechtigt (authoritative)
- die Replikation/Informationsabgleich kann auf zwei Arten gelöst werden:
  - im SOA-Record der Zone wird das Zeitintervall für "Neuigkeiten"-Abfrage slave→master festgelegt[21] wird eine Änderung festgestellt → "polling"
  - bei Änderungen der Daten auf dem master-Server werden die slave-Server mit einer speziellen Nachricht informiert → "event triggered"
- in beiden Fällen läuft die Synchronisation normalerweise über TCP

[19] früher "primary" und "secondary"

[20] das entsprechende Informationsfeld im SOA-Record ist nur "informativ"

[21] über das Versions-Feld im SOA-Record

DNS: Clients

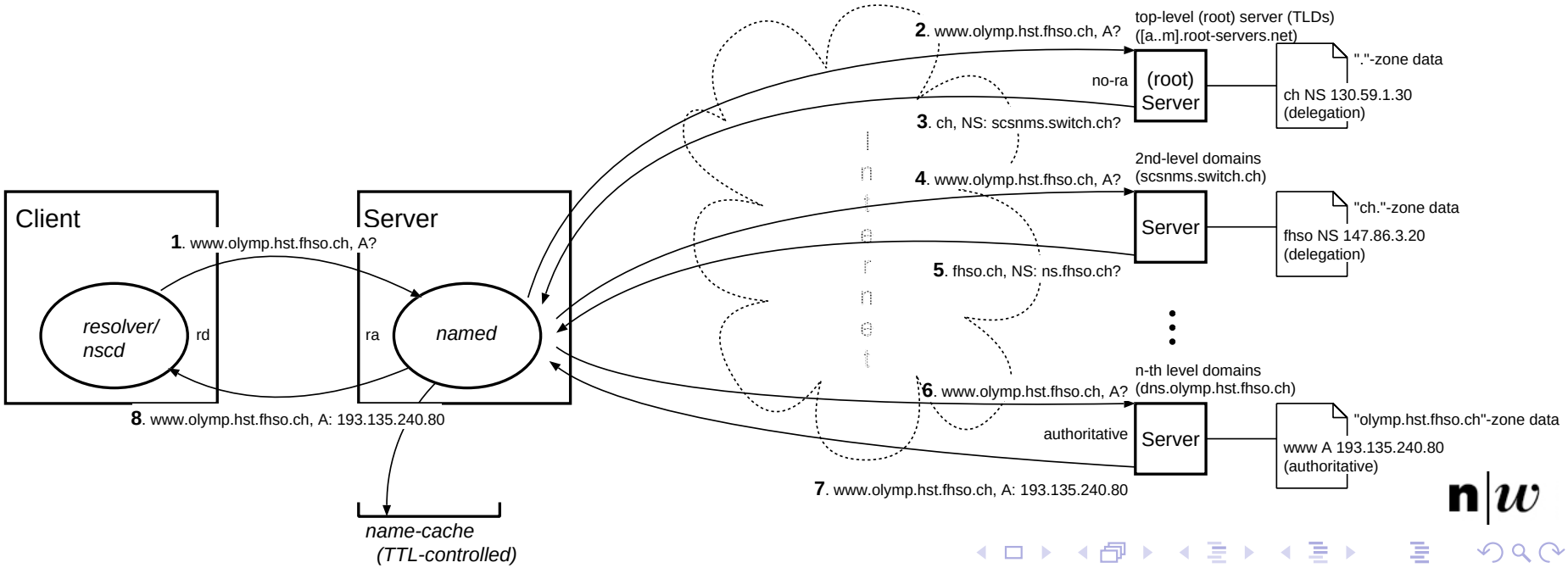
- Clients benutzen eine der Funktionen `gethostbyname`, `gethostbyaddr` oder `getaddrinfo` in den System-Bibliotheken[22] um DNS-Abfragen zu senden
- moderne Betriebssysteme benutzen einen *lokalen* Cache-Dienst (`nscd`, ein eigener Prozess) um Anfragen von verschiedenen Prozessen/Programmen effizient zu verwalten (mit internem cacheing)
- `nslookup`, `host`, etc benutzen im Normalfall *nicht* diese Library
- zur korrekten Funktion muss mindestens ein Forwarding/Caching-Nameserver pro Client definiert werden (`/etc/resolv.conf`). Das passiert normalerweise über DHCP
- mit dem `AA`-Flag[23] in der Antwort des Servers kann festgestellt werden ob die Antwort gecacht oder authoritative war

[22] Library: `libc` oder `net`

[23] "AuthoritativeAnswer"

DNS: Recursive Query

- ein Client kann eine “recursive query” an einen Server stellen, der dann die ganze “Arbeit” übernimmt und nur das Endergebnis liefert
- dazu muss der Client das “rd” recursion-desired flag setzten und der Server “ra” recursion-available anbieten
- meistens wird vom Server “ra” nur an *interne* Clients angeboten
- “Caching”-Server ist eine spezielle Rolle, die Queries weiterleiten (recursive) und lokal zwischenspeichern. Gleiche Anfragen werden aus dem Cache bedient



DNS: Caching

- anders als z.B. bei ARP ist das Caching bei DNS über den Parameter – TTL[24] – genau geregelt
- TTL gibt an, wielange ein Caching-Server[25] die zwischengespeicherte Antwort weiter ausliefern darf. Bei abgelaufener TTL muss der Authoritative-Server neu angefragt werden
- wenn eine Antwort aus einem Cache[26] ausgeliefert wird, wird in der Antwort das "AA" (authoritative-answer) Flag *nicht* gesetzt
- eine lange[27] Zeitangabe – z.B. mehrere Tage – führt bei Änderungen am Resource-Record zu Problemen weil viele Caching-Server und auch Clients noch die alte Antwort weiterverwenden
- eine kurze[28] Zeitangabe (einige Sekunden) ist zwar flexibel führt aber zu wesentlich mehr Anfragen auf den "Authoritative"-Server

[24] Time-To-Live, hier tatsächlich eine Zeitangabe in Sekunden

[25] oder ein Client-seitiger Cache

[26] oder generell von einem nicht-Authoritativen-Server

[27] oft bei top-level Domainserver, z.B. a.nic.ch zu finden um die Last gering zu halten

[28] oft bei Webseiten, etc zu finden

DNS: Loadbalancing

- manchmal werden auf eine Anfrage mehrere Antworten[29] geliefert – entspricht "mehrere Telfonnummern"
- der Client wählt eine[30] der gleichwertige Antworten für weitere Kommunikation aus
- Server "rotieren" die Antworten bei jeder Auslieferung und erreichen so eine primitive Art des Loadbalancings[31]

[29] für einen Resource-Record-Type

[30] meistens die erste

[31] verteilen von Anfragen über mehrere – z.B. Web – Server

DNS: Praktische Übungen

finden Sie Beispiele von:

- kurze TTL
- lange TTL
- "round-robin"-Loadbalancing/mehrere Antworten
- beobachten Sie den "decay"/das Ablaufen der TTL bei Abfrage eines Caching-Servers (z.B. 1.1.1.1, 8.8.8.8, 9.9.9.9)

verwenden sie wenn möglich die Authoritative-Server (ausser bei Caching-Frage):

```
# mit dig, der @-Parameter ist der anzufragende Host
# (hier der NS von post.ch, d.h. der authoritative-nameserver
dig www.post.ch @dns1.post.ch

# oder mit host
host -h dns1.post.ch www.post.ch
```

n|w

DNS: DoH und DoT[33]

Eine neuere Entwickung ist die Implementierung einer verschlüsselten Kommunikation ("privacy") bei DNS-Abfragen

- Caching-Server (z.B. die des Providers, des Campus) können so nicht erkennen welche Abfragen gemacht werden (und auch nicht verändern)
- die tatsächliche IP-Kommunikation findet aber trotzdem vom Client zum Server statt – die IP-Adressen sind wiederum ersichtlich
- der Benutzer vertraut einfach einem DoH/DoT-Betreiber[32] anstatt dem lokalen Providers, Campus, etc

Kritik: Privacy ist auch ein Geschäft

Die DoT und DoH Server "sehen" natürlich die Anfragen und können daraus wichtige Meta/Statistik/Business-Daten erzeugen.
Zudem gehen wichtige Sicherheitstechniken verloren (gesperrte Malware-Domains).

[32] die auch ein Geschäftsmodell haben...

[33] DNS-over-HTTPS, DNS-over-TLS