# Rechnernetze - Computer Networks

## Lecture 12: Applications

Prof. Dr.-Ing. Markus Fidler

Institute of Communications Technology
Leibniz Universität Hannover

July 5, 2024

# Outline



Principles of Application Layer Protocols

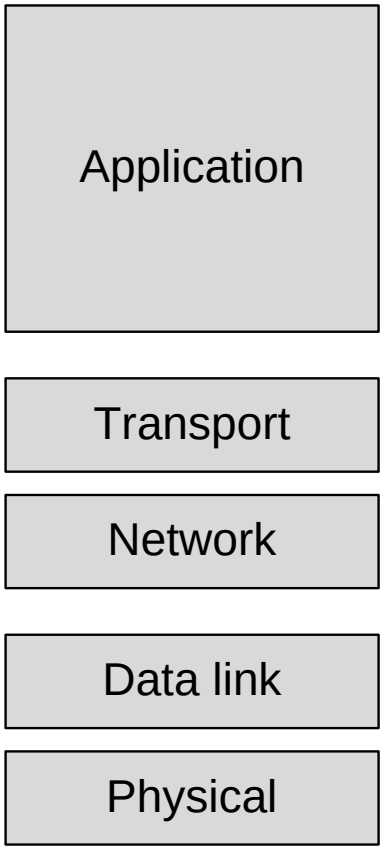
TCP-based Applications
- Remote Login: Telnet
- File Transfer: FTP
- World Wide Web: HTTP
- Email: SMTP

UDP-based Applications
- Domain name system (DNS)

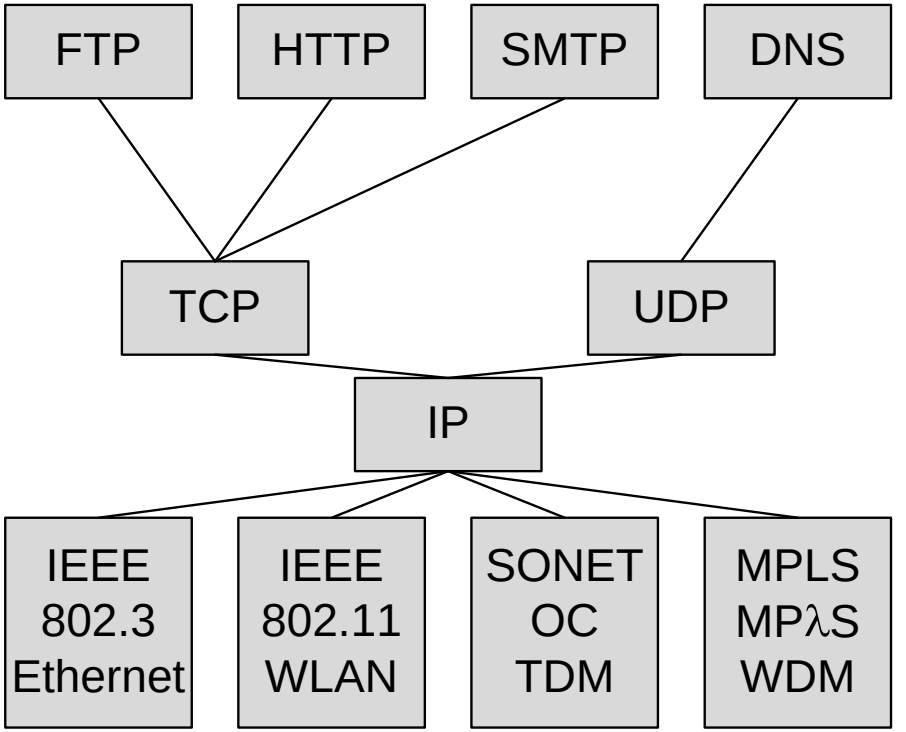
# Previous lectures



TCP/IP model




Internet protocols



# How to locate an application process?



An application process that sends a message to another application process just hands over the message at the socket interface.

Likewise the message is delivered afterwards to the receiving process at its socket interface.

Locating the receiving process involves

- locating the target host
- locating the target application process on that host

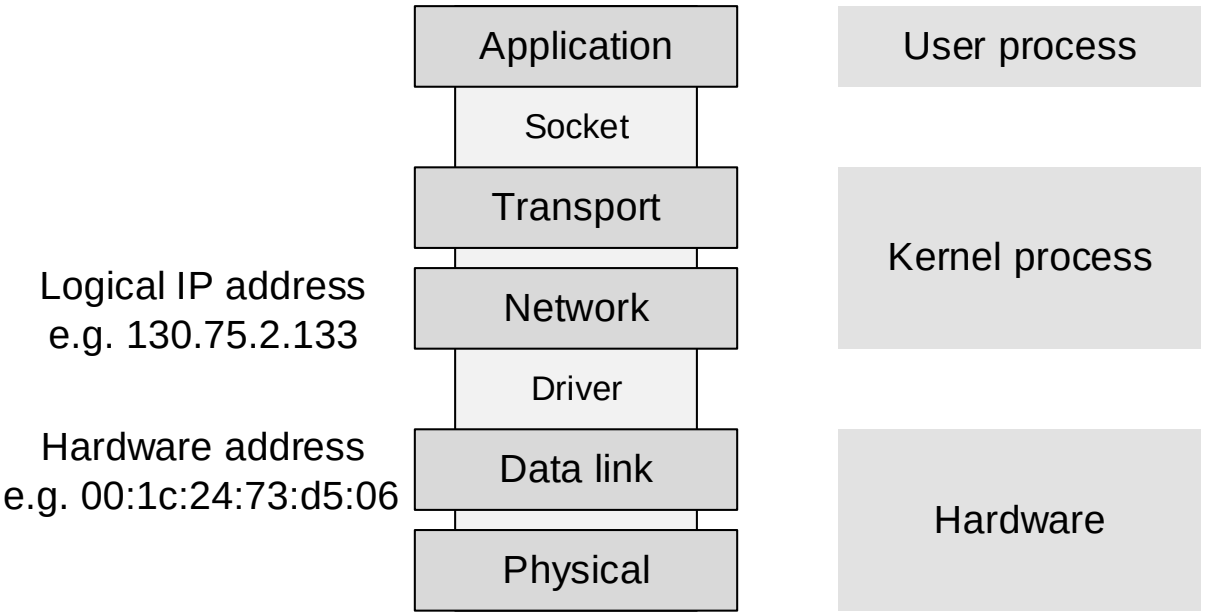
It uses

- addresses to identify (a network interface on) the host
- identifiers (ports) that direct the message to the right process

Packets include source and destination addresses and ports.

# Addressing

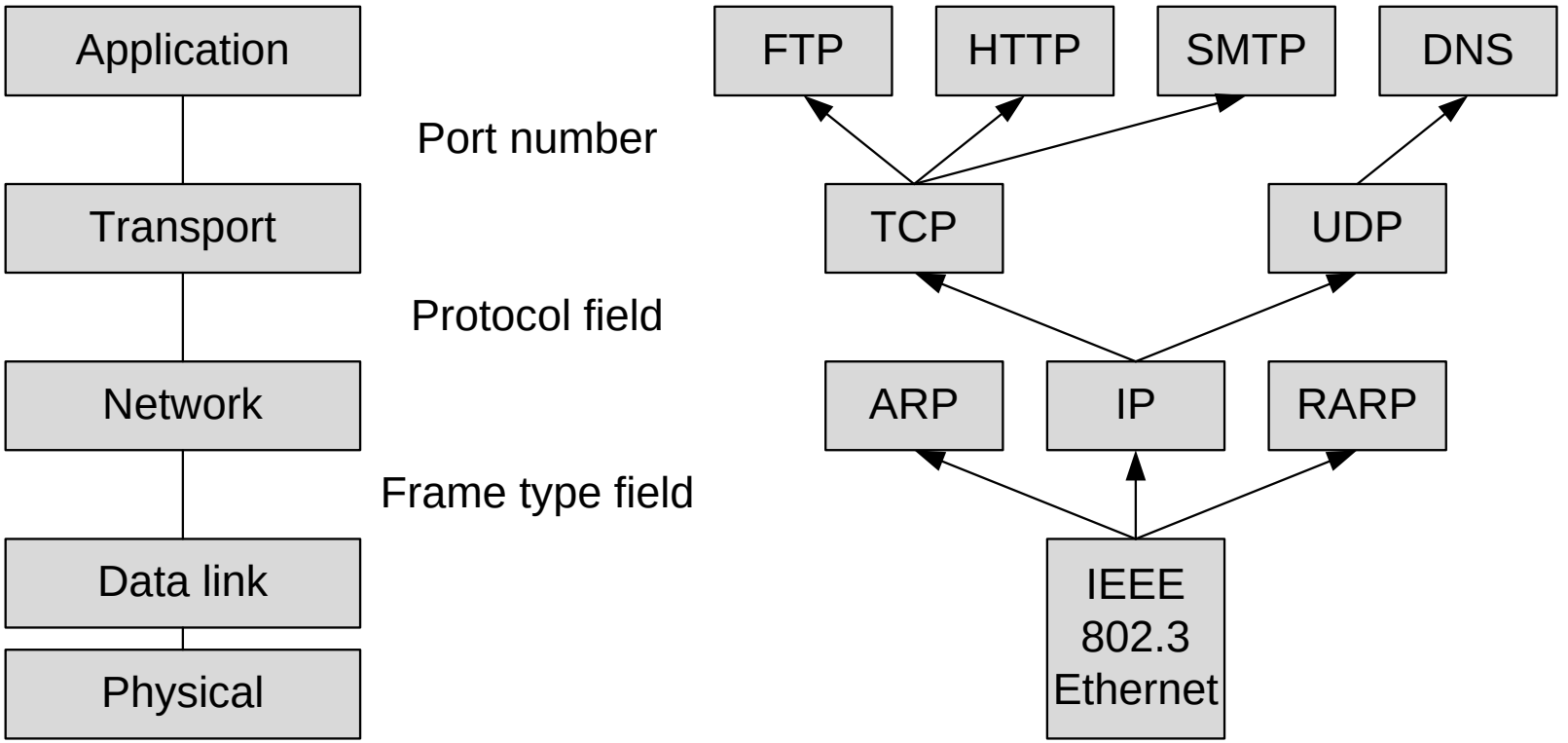





- Socket
  - interface (API) between operating system and application
  - IP addresses are assigned, e.g., dynamically by DHCP
- Driver
  - interface between hardware and operating system
  - physical hardware addresses are assigned by manufacturers

# Multiplexing and demultiplexing






- the Ethernet frame type field identifies the network protocol
- the IP protocol field identifies the transport protocol
- the TCP/UDP port fields identify the application protocol

# Client/Server paradigm



Server

- offers a defined service to clients
- waits for incoming requests or queries from clients
- answers requests and sends replies to clients
- examples: WWW server, FTP server, mail server
- a host can run several servers in parallel

Client

- makes requests or queries to use services of the server
- after issuing a request waits for replies from the server
- examples: WWW browser, FTP client, mail client
- a host can request services from several servers simultaneously

# Pull vs. push services



Pull services (previous slide)

- passive server that waits for requests
- active client that requests services from the server
- bidirectional data exchange

Push services

- passive client that waits for messages
- server sends messages to the client, e.g. periodically
- clients receive messages
- unidirectional data transfer

How can a server with push service know its clients?

# Application service requirements



- Packet loss
  - applications require loss-free transmission, lost packets have to be retransmitted
  - or applications can tolerate some packet loss, lost packets cause only quality degradation
- Data rate
  - applications request a guaranteed minimum capacity
  - or applications are elastic and use the currently available capacity
- Latency
  - applications require data delivery within a specified time limit
  - or applications tolerate delays within certain bounds

# Common Internet transport protocols



Application protocols rely on the service provided by transport protocols (layer 4). Transport protocols are used to deliver application layer data.

Mainly two types of transport protocols are used in the Internet:

- Transmission Control Protocol (TCP)
  API: Streams Socket
  connection-oriented, reliable, flow and congestion control
- User Datagram Protocol (UDP)
  API: Datagram Socket
  connectionless, unreliable, unlimited data rate

# Classification of applications



| Application | Loss | Data rate | Latency | Transport |
|---|---|---|---|---|
| File transfer | no | elastic | insensitive | TCP |
| EMail | no | elastic | insensitive | TCP |
| WWW | no | elastic | few s | TCP |
| Streaming Audio/Video | ok | kbps-Mbps | few s | UDP |
| Real-time Audio/Video | ok | kbps-Mbps | $\leq$ 100 ms | UDP |
| Online games | ok | kbps | $\leq$ 100 ms | UDP |

Why do particular applications choose TCP or UDP?

# Application layer vs. application



The application layer is the part of the application that defines how distributed applications pass messages to each other.

Examples

- the Web is an application that provides access to documents. It comprises
  - Web browsers (e.g. Firefox, Internet Explorer)
  - Web servers (e.g. Apache, Microsoft)
  - a document format (HTML)
  - an application layer protocol (HTTP)
- Email is an application. It comprises
  - mail clients (e.g. Thunderbird, Outlook)
  - mail servers (e.g. Postfix, Sendmail)
  - a standard that defines the structure of emails (e.g. MIME)
  - an application layer protocol (SMTP)

# Scope of application layer protocols



Application-layer protocols define

- the types of messages that are exchanged
- the syntax of messages, i.e., different fields
- the semantics of the fields, i.e., their meaning
- rules for message exchange
  - when to send which messages
  - and how to respond to certain messages

# Outline

# Telnet



Task

- bidirectional byte-oriented communication channel
- connection to a remote host respectively terminal
- the user must have an account on the remote host to log in

Specification

- Telnet stands for telecommunications network protocol that dates back to 1969 on the ARPAnet
- defined in RFC 854 and 855 that are Internet standard 8

Characteristics

- client/server architecture, stateful server
- uses reliable TCP connection, server port 23
- inband signalling
- data are not encrypted including username and password

# Data format and inband signalling



Data can be encoded as

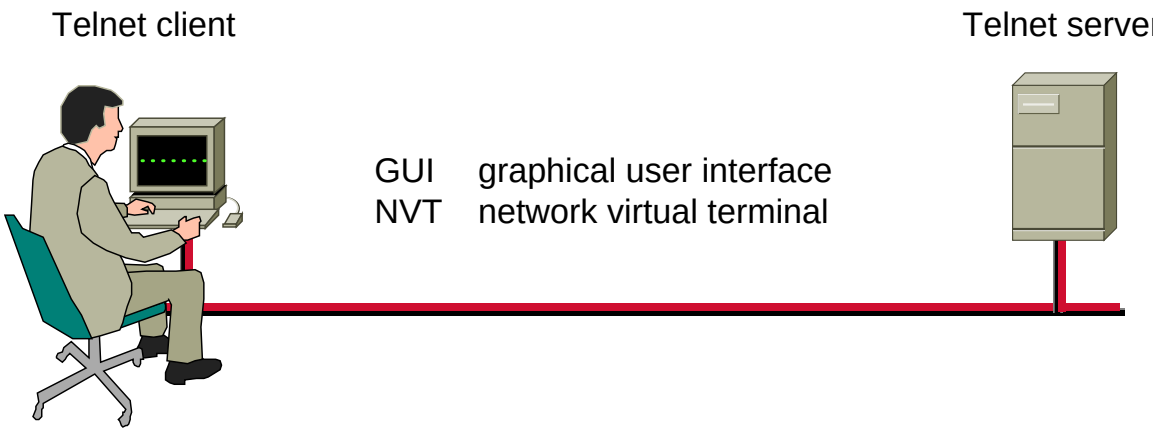
- 7 bit ASCII, byte stream with most significant bit (MSB) set to zero
- alternatively binary format using all 8 bits per byte

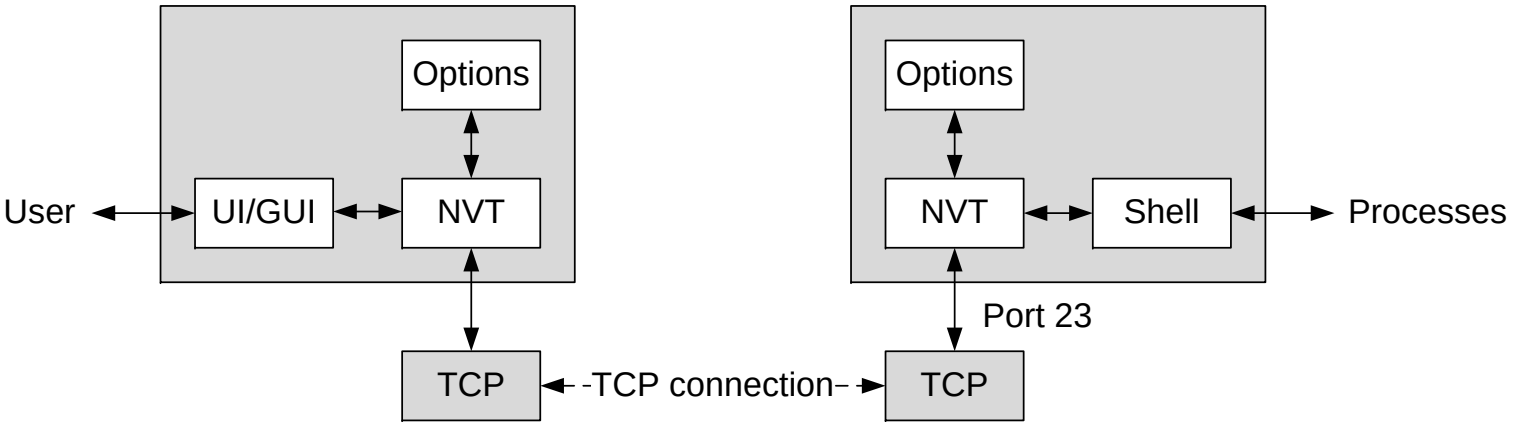
Inband signalling

- 0xff stands for interpret as command (IAC)
- inband commands, preceded by an IAC, are e.g.
  - end of file (EOF) is 0xec
  - interrupt process (IP) is 0xf4
- data bytes that are 0xff are encoded as 0xff 0xff

Secure Shell (SSH, port 22) adds authentication and encryption

# Architecture




NVT: standard terminal for inter-operability of heterogenous systems.

# File transfer protocol (FTP)



Task

- copy files from or to a remote host
- the user must either
  - have an account on the remote host or
  - the remote system is configured to use anonymous FTP

Specification

- dates back to 1971
- defined in RFC 959 that is Internet standard 9

Characteristics

- client/server architecture, stateful server
- uses reliable TCP connections, server ports 20 and 21
- out-of-band signalling
- data are not encrypted including username and password

# Data format and out-of-band signalling



Data can be encoded using a (limited) number of file types and file structures

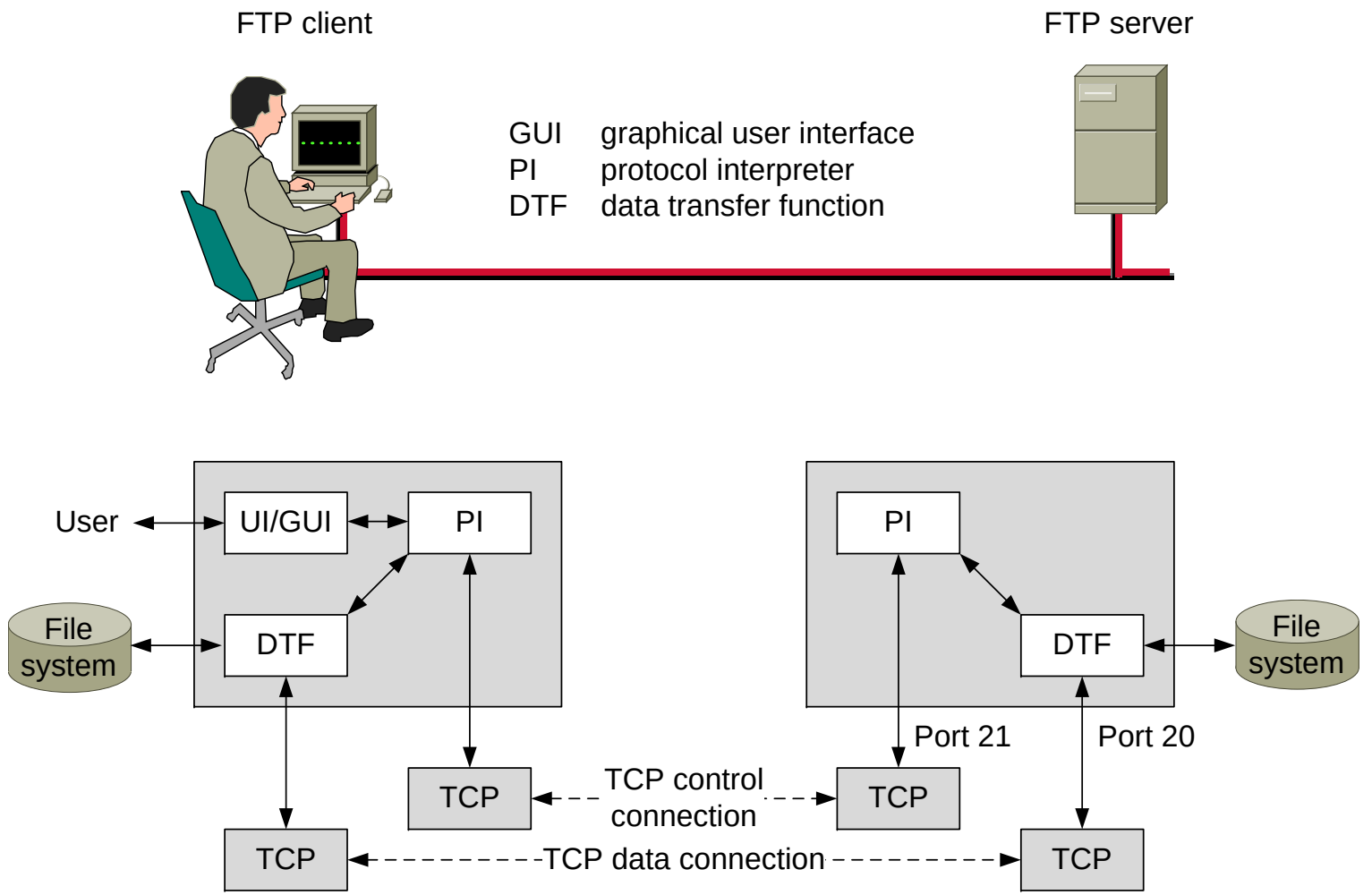
- 7 bit ASCII
- binary
- ...
- conversion of characters during transfer is possible
  - e.g. line breaks are encoded differently under UNIX and Windows

Out-of-band signalling

- data connection, server port 20
- control connection, server port 21

Secure Copy (SCP, uses SSH) adds authentication and encryption

# Architecture



# Commands and replies



The FTP client uses the control connection to send commands

- USER and PASS: username and password on the server
- PORT: open data connection
- LIST: list files and directories on the server
- RETR: retrieve a file from the server
- STOR: store a file on the server
- QUIT: logoff from server

The FTP server replies a 3-digit code with defined meaning, as well as an optional human readable message, e.g.

- 220 Service ready for new user.
- 331 Username OK, password required.
- 230 User logged in, proceed.
- 226 Closing data connection. Requested file action successful.
- 550 Requested action not taken, file unavailable.
- 221 Service closing control connection.

# Connection management



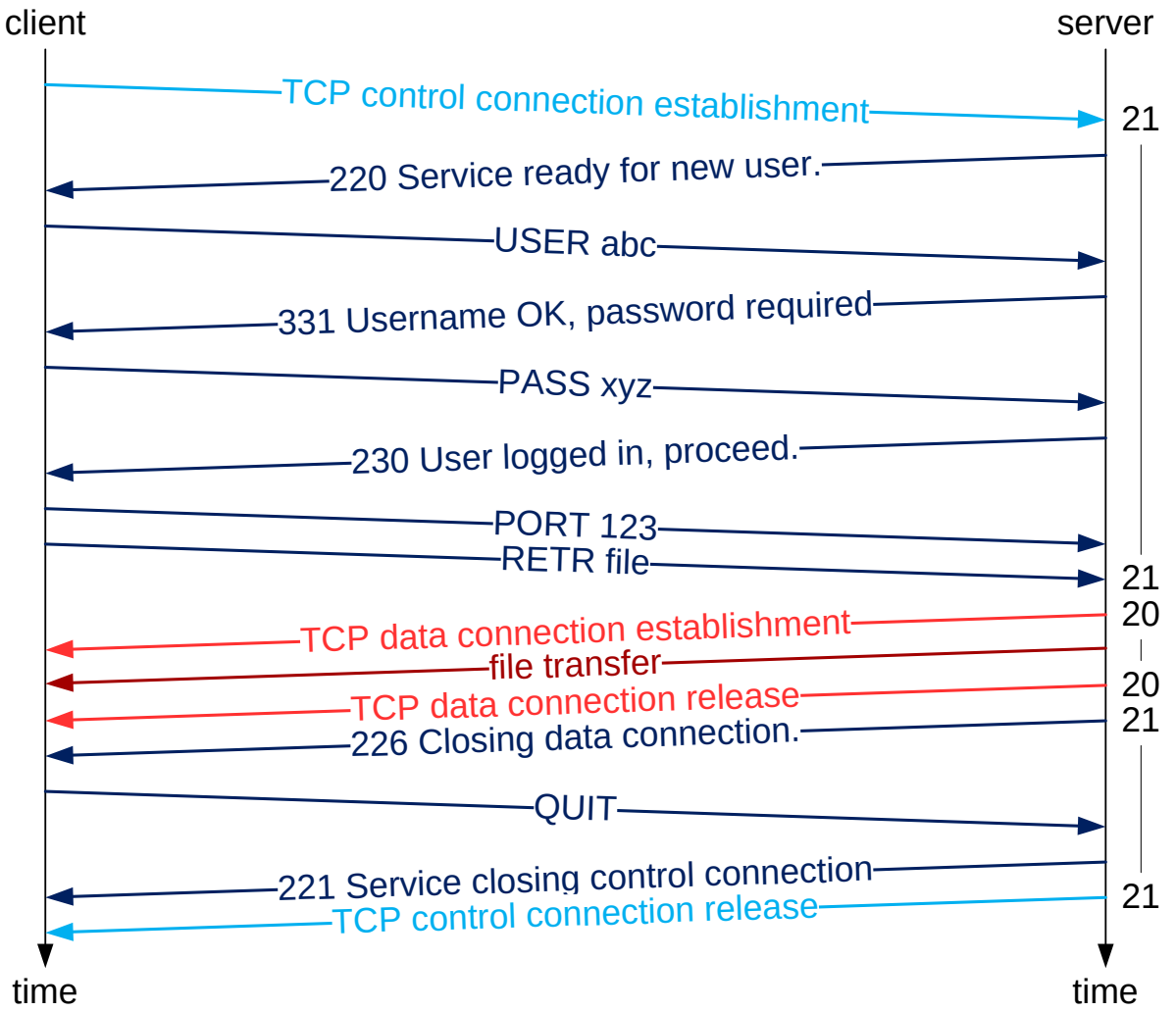
The control connection persists during the entire FTP session whereas the data connection is established only on demand and released afterwards. The data connection is used to

- send a file from the client to the server
- send a file from the server to the client
- send a list of files and directories from the server to the client

Procedure

- the client controls the creation of a data connection
- the client sends a PORT command to the server
- the client waits for the data connection
- the server opens the data connection
- the server closes the data connection at the end of the file

# Example: message sequence



# Hypertext transfer protocol (HTTP)



Task

- retrieve Web pages (documents) consisting of objects (simply files) from Web servers

Specification

- HTTP version 1.0 defined in RFC 1945 (world wide wait)
- HTTP version 1.1 defined in RFC 2616 (pipelining)
- HTTP version 2.0 defined in RFC 7540 (speculative push)
- versions are compatible with each other

Characteristics

- client/server architecture, stateless server (state in Cookies)
- uses reliable TCP connection, server port 80
- inband signalling
- data are not encrypted

# Components of the World Wide Web

WWW servers

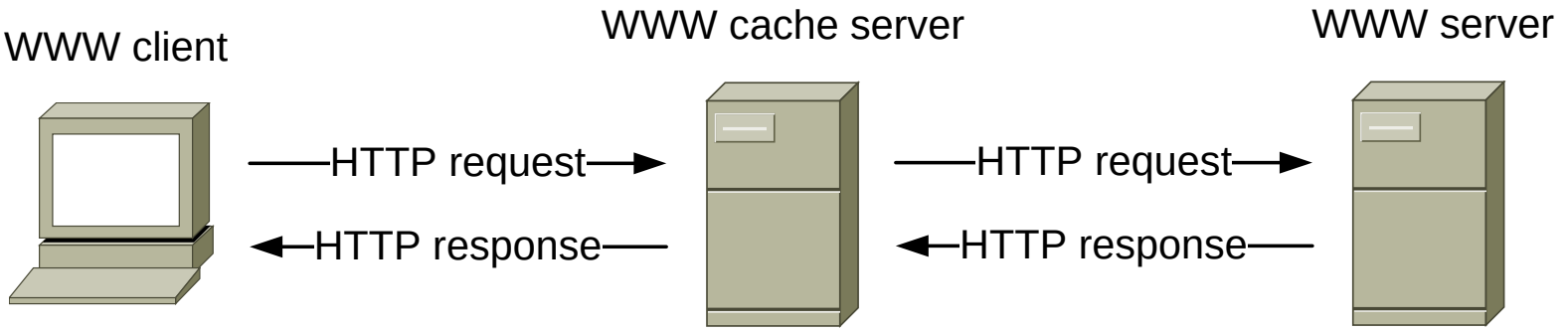
- receive HTTP requests for defined objects from clients
- return HTTP responses with the requested objects
- dynamically generate requested content/objects

WWW clients

- send HTTP requests for objects to the respective servers
- interpret returned objects and render information graphically
- automatically request embedded objects
- execute embedded program code, e.g. JavaScript
- execute additional programs and plug-ins

# Architecture






- clients send HTTP requests to request objects from servers
- servers send HTTP responses to send objects to clients
- caches pass through requests and responses and store objects

# Data format and inband signalling



HTTP messages consist of header and body, encoded using 8 bit ISO Latin 1 (ASCII with Western European extensions)

Header

- contains control information, inband signalling

Body

- response messages contain the requested object
- request messages may also contain information, e.g. form fields, passwords
- MIME Content-Type is used to specify objects
  - Hypertext Markup Language (HTML)
  - Extensible Hypertext Markup Language (XHTML)
  - all other MIME types

# Uniform resource locator (URL)



Objects are referenced by uniform resource locators (URLs) that have two components

- host name of the server
- path name of the object
- e.g. `www.uni-hannover.de/pics/h_start.jpg`

In addition the application layer protocol and the port can be specified resulting in

- syntax: <protocol>://<host name>:<port><path name>
- e.g. `http://www.uni-hannover.de:80/pics/h_start.jpg`

# HTTP 1.0 connection management

HTTP 1.0

- for each object a separate TCP connection is established
- after transmitting an object the TCP connection is released (usually by the server)
- after reception objects are scanned for embedded objects
- embedded objects are requested successively using the above procedure
- simple to implement but low performance

Netscape work-around

- scan received object for embedded objects while receiving it
- immediately open additional TCP connections in parallel to download these objects

# HTTP 1.1 connection management



Persistent TCP connections

- TCP connections are not released after an object is transmitted
- TCP connections are used for transmission of several objects
- clients have to detect the end of an object (the size is given in the header)
- signaling procedures for connection release are required
- timeouts to terminate erroneous connections are required

Request pipelining

- several HTTP requests can be issued in parallel on a single TCP connection
- responses are sent sequentially
- used to request embedded objects

# Components of electronic mail



User agent for asynchronous email communication

- read, reply to, forward, save, and compose messages
- send messages to a mail server
- access received messages from a mail server
- manage the mailbox on a mail server

Mail servers

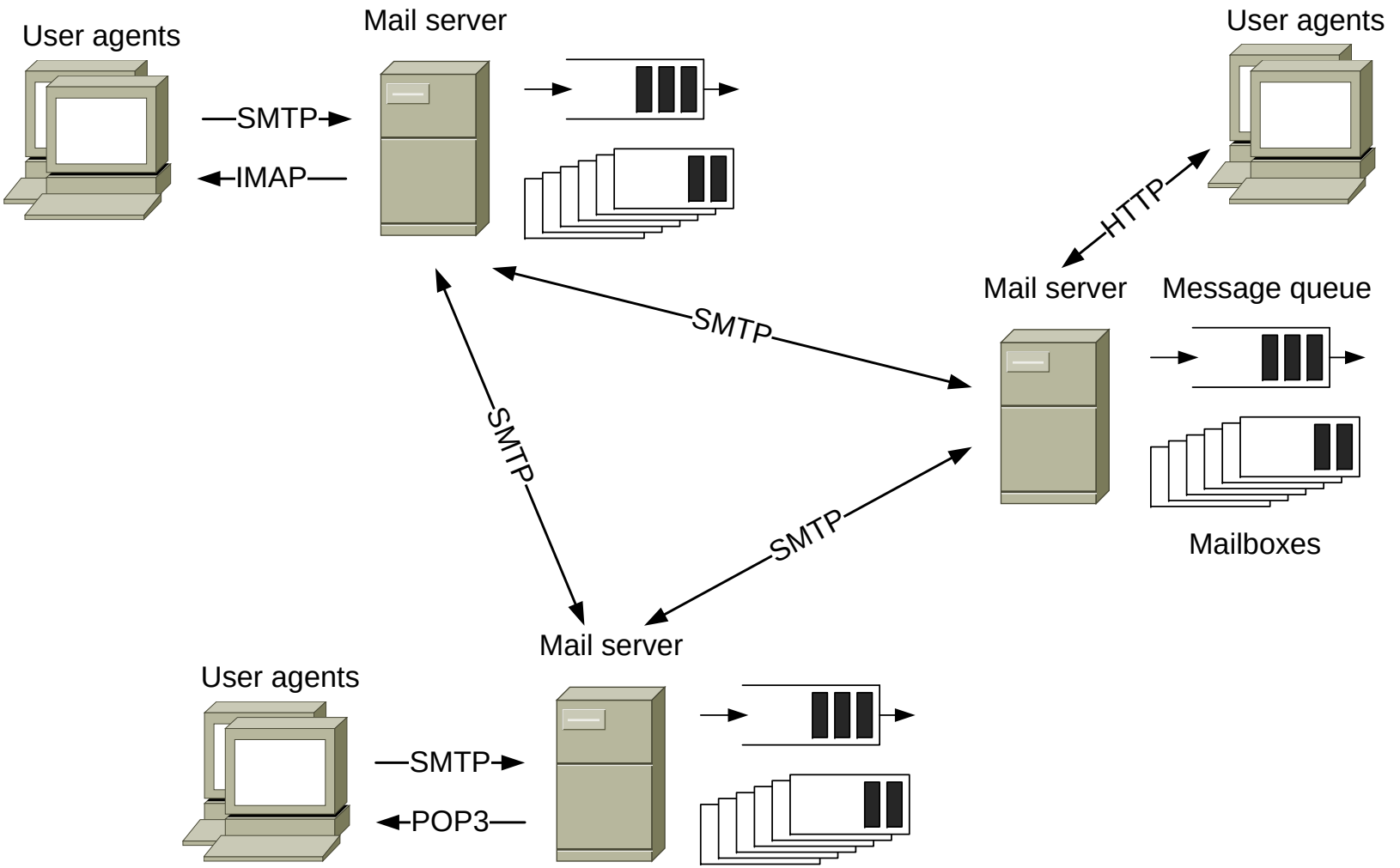
- receive and place incoming mail into the mailbox of the recipient
- receive mail from users and forward it to other mail servers
  - message routing (DNS mail exchange record)
  - message queue for outgoing mail, reattempt sending in case of failure, e.g. every 30 minutes for several days

Protocols

- simple mail transfer protocol (SMTP)
- post office protocol version 3 (POP3)
- Internet message access protocol (IMAP)

# Architecture





# Simple mail transfer protocol (SMTP)



Task

- transfer emails from the user agent to the mail server
- transfer emails between mail servers

Specification

- defined in RFC 821 and 1870 that are Internet standard 10, email message routing RFC 974, all obsoleted by RFC 5321

Characteristics

- client/server architecture
  - client side is executed on the sender's mail server
  - server side is executed on the recipient's mail server
- uses reliable TCP connection, server port 25
- inband signalling
- data are not encrypted
- push protocol

# Commands and replies



The structure of SMTP commands and replies is similar to FTP.
Typical commands used in this order are

- HELO: announce the name of the SMTP client to the server
- MAIL FROM: address of the mail sender
- RCPT TO: address of the mail recipient
- DATA: the mail text
- QUIT: terminate the SMTP session

The SMTP server replies 3-digit codes to each command, e.g.

- 250 Sender OK, Recipient OK, Message accepted for delivery
- 221 Closing connection.

# Data format and inband signalling



Emails consist of header and body, both encoded using 7 bit ASCII

Typical header fields

- Received: (added as a prefix by each receiving mail server)
  - from name of the SMTP client
  - by name of the SMTP server
  - date of reception
- From: senders email address
- To: recipients email address
- Subject: subject line

Body

- CRLF.CRLF (carriage return line feed) marks the body's end

# Multipurpose Internet mail extensions (MIME)

MIME specifies how to encode other data formats than 7 bit ASCII in emails and email attachments.

Specification

- specified in RFC 2045 and 2046
- IANA takes care of types/subtypes as specified in RFC 2048

MIME defines additional header fields for use in SMTP

- MIME-Version: e.g. 1.0
- Content-Transfer-Encoding: encoding technique for conversion into 7 bit ASCII, e.g. base64, binary
- Content-Type: media type/subtype; parameters

# MIME types



- text
  - text/plain; charset=us-ascii
  - text/html
- image
  - image/jpeg
  - image/gif
- video
  - video/mpeg
  - video/quicktime
- application
  - application/msword
  - application/octet-stream
- multipart/mixed; Boundary=xyz
  - --xyz denotes the boundary
  - after each boundary new MIME headers may follow

The subtype field is frequently used to start a selected application.

# Mailbox access



The user agent is executed on a local computer that

- has a rich set of features, e.g. multimedia etc.
- is, however, not always on resp. connected to receive mail

Thus mail servers usually controlled by ISPs receive mail and store it in the recipients mailbox for later access.

- sending mail using SMTP is a push operation
- accessing the mailbox is, however, a pull operation

Protocol options for accessing the mailbox

- post office protocol (POP)
- Internet mail access protocol (IMAP)
- hypertext transfer protocol (HTTP), browser/web frontend

# Post office protocol (POP)



Post office protocol version 3 (POP3)

- specified in RFC 1939, Internet standard 53
- uses reliable TCP connection, server port 110
- mailbox access comprises three phases
  - authorization: username and password
  - transaction: retrieve messages, (un)mark messages for deletion
  - update: quit POP3 session, server deletes marked messages
- two possible replies from server
  - +OK
  - -ERR
- two operating modes
  - download-and-delete mode
  - download-and-keep mode
- state information is kept during but not between sessions
- folder structure is kept only on the local machine

# Internet mail access protocol (IMAP)

Internet mail access protocol (IMAP)

- specified in RFC 2060
- uses reliable TCP connection, server port 143
- includes the basic functionality of POP3 and
- manages multiple message folders at the mail server
  - folder hierarchy for each user on the mail server
  - remote mailboxes can be manipulated like local ones
  - state persists over successive accesses
  - very useful for nomadic users that use several different computers for mail access
- client can obtain mail headers or parts of MIME messages only, e.g. over slow access links

# Outline

# Domain name system (DNS)



Task

- there are two ways of identifying a host on the Internet
  - hostname, mnemonics used by humans, e.g. `www.uni-hannover.de` or `cmsv023.rrzn.uni-hannover.de`
  - IP-address, hierarchical structure used by routers, e.g. 130.75.2.151
- need a directory service that translates hostnames to IP-addresses: Domain name system (DNS)

Specification

- defined in RFC 1034 and 1035 that are Internet standard 13

Characteristics

- client/server architecture
- usually unreliable UDP communication, destination port 53

# DNS example of use



DNS is an infrastructure service that is employed by other application layer protocols, e.g.

- a user uses the browser to request the URL `www.uni-hannover.de/pics/h_start.jpg`
- the browser has to send an HTTP request message to `www.uni-hannover.de` to retrieve `pics/h_start.jpg`
- to route the message to `www.uni-hannover.de` the IP-address of the host has to be known
  - the browser passes the hostname `www.uni-hannover.de` to the client-side DNS application running on the same host
  - the DNS client sends a query for `www.uni-hannover.de` to the DNS server
  - the server sends the IP-address 130.75.2.151 to the client
  - the client passes the IP-address 130.75.2.151 to the browser
- the browser sends an HTTP request message to 130.75.2.151 to retrieve `pics/h_start.jpg`

# DNS services



Services offered by DNS (try `nslookup`)

- translate hostnames to IP addresses
- host aliasing: hosts can have several hostnames
  - alias hostnames, e.g. `www.ikt.uni-hannover.de`
  - canonical hostname, e.g. `cmsv005.w3.uni-hannover.de`
- mail server aliasing: email addresses are localpart@domain
  - the domain name can be an alias for the mail server
  - an organizations web and mail server can have identical aliases
- load balancing: replicated servers with different IP addresses
  - DNS always returns the entire set of IP addresses, e.g.
  - `www.spiegel.de`
    128.65.210.180
    128.65.210.181
    ...
    128.65.210.185
  - the ordering of IP addresses rotates, however

# DNS mode of operation



Client side is simple

- simple library routine returns the IP-address of a hostname
- e.g. `gethostbyname()` on UNIX systems
- sends DNS query messages and waits for DNS reply

Server side is complex

- distributed database
- hierarchy of name servers
- application layer protocol that finds the needed DNS record
  - servers send DNS replies to hosts
  - servers exchange DNS queries and DNS replies
- widely used software: BIND (Berkeley Internet Name Domain)

# Why not a simple centralized approach?



A single DNS server would be simple but has major drawbacks:

- **reliability:** the DNS server is a single point of failure, if it breaks hosts are not reachable by their hostname any more
- **performance:**
  - a single DNS server has to handle a huge traffic volume, i.e. all DNS queries (e.g. WWW, email)
  - a single server cannot be close to all hosts causing large response times
- **management:** a single server is huge since it includes all hosts; it has to be updated for each new host

In fact, the name system was managed centrally at first

- all hostname, IP address pairs were stored in a file hosts.txt
- hosts.txt was maintained by the Network Information Center (NIC) at the Stanford Research Institute (SRI)
- hosts.txt was periodically downloaded by hosts using FTP

# Distributed hierarchical approach



A distributed database solves previous problems:

- **management:** distributed database involving many DNS servers
  - hierarchical tree structure
    - top-level domains, e.g. `.com`, `.edu`, `.de`
    - subdomains `.uni-hannover.de`, `.ikt.uni-hannover.de`
  - decentralized administration
    - delegation in subdomains
    - the university maintains `.uni-hannover.de`
    - the institute maintains `.ikt.uni-hannover.de`
  - but: central root name server in the tree structure
- **reliability:** mirrored servers achieve redundancy
- **performance:** caching

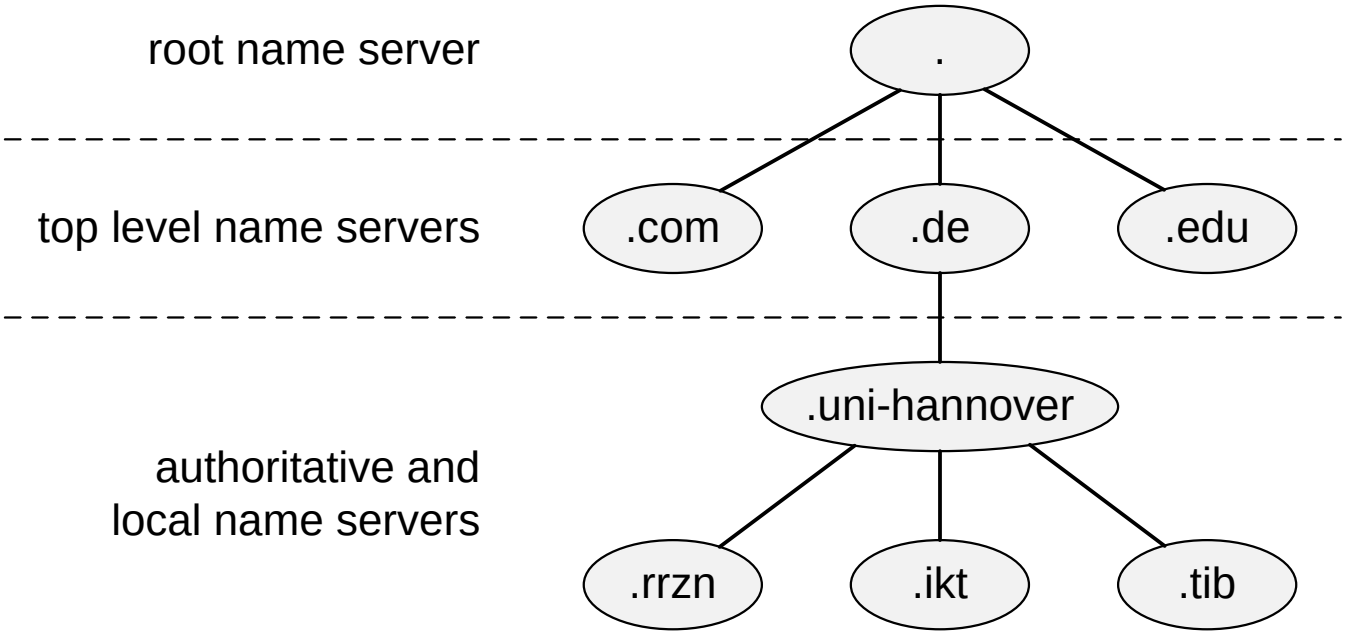
# Classification of name servers



- authoritative name servers:
  - each zone (name space) has an authoritative name server
  - every host of the zone is registered with the name server
- local name servers:
  - maintained by ISPs, organizations, universities, etc.
  - configured as the default DNS server for belonging clients
  - server is geographically close to the clients
  - can reply to DNS queries for clients of the same ISP
  - often authoritative name server and local name server are on the same physical machine
- top-level name servers: name servers for top-level domains
- root name servers: currently 12 independent operators with 1537 instances overall (July 11, 2022), see `http://www.root-servers.org/`

# Tree structure






ICANN controls top level domains

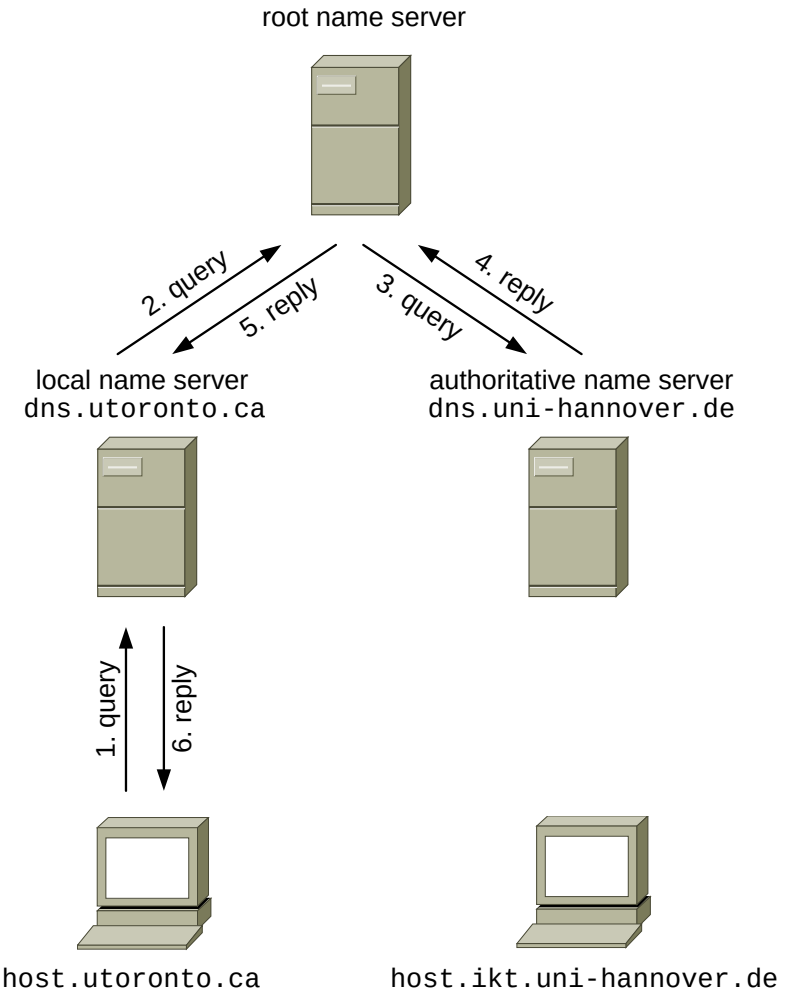
- generic top level domains, e.g. .com, .net, .org
- country code top level domains, e.g. .uk, .ca, .de

subdomains are managed decentrally

# Recursive queries

`host.utoronto.ca` queries for `host.ikt.uni-hannover.de`

1. queries local name server
2. `dns.utoronto.ca` has no entry, queries root server
3. root knows authoritative name server, queries it
4. `dns.uni-hannover.de` replies IP-address of `host.ikt.uni-hannover.de`
5. reply to local name server
6. reply to requesting host

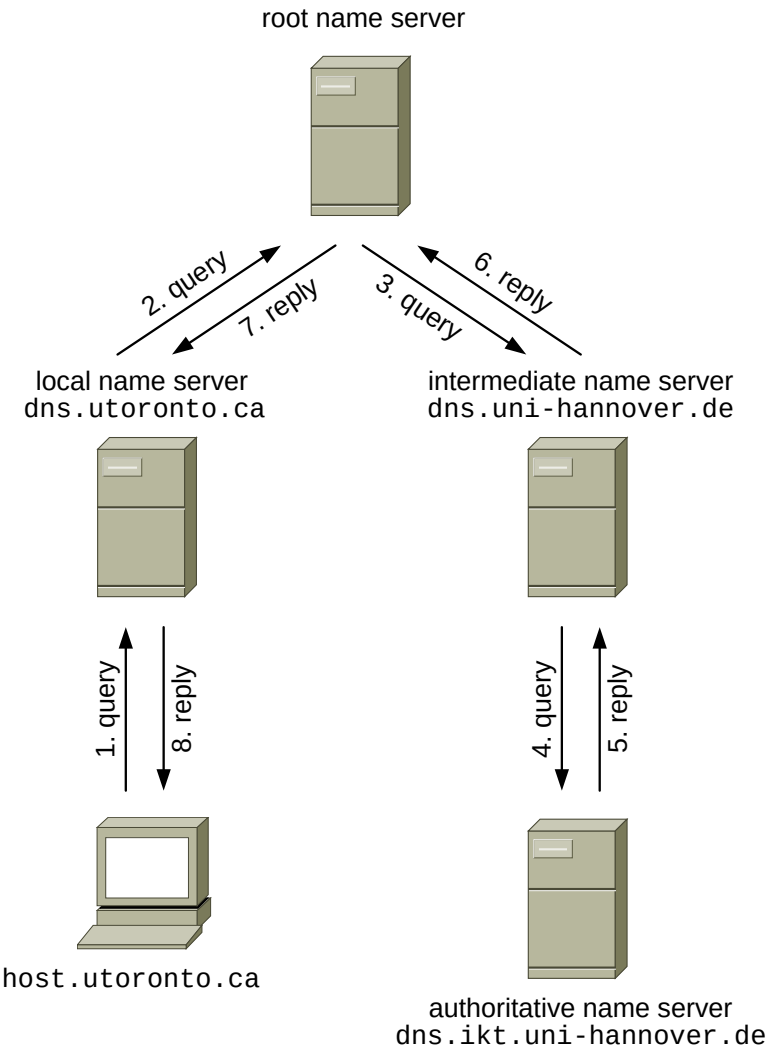


# Recursive queries continued



`host.utoronto.ca` queries for `host.ikt.uni-hannover.de`

3. root does not know authoritative name server, queries intermediate
4. intermediate queries authoritative name server
5. authoritative name server replies IP-address of `host.ikt.uni-hannover.de`
6. reply to root name server
7. reply to local name server
8. reply to requesting host

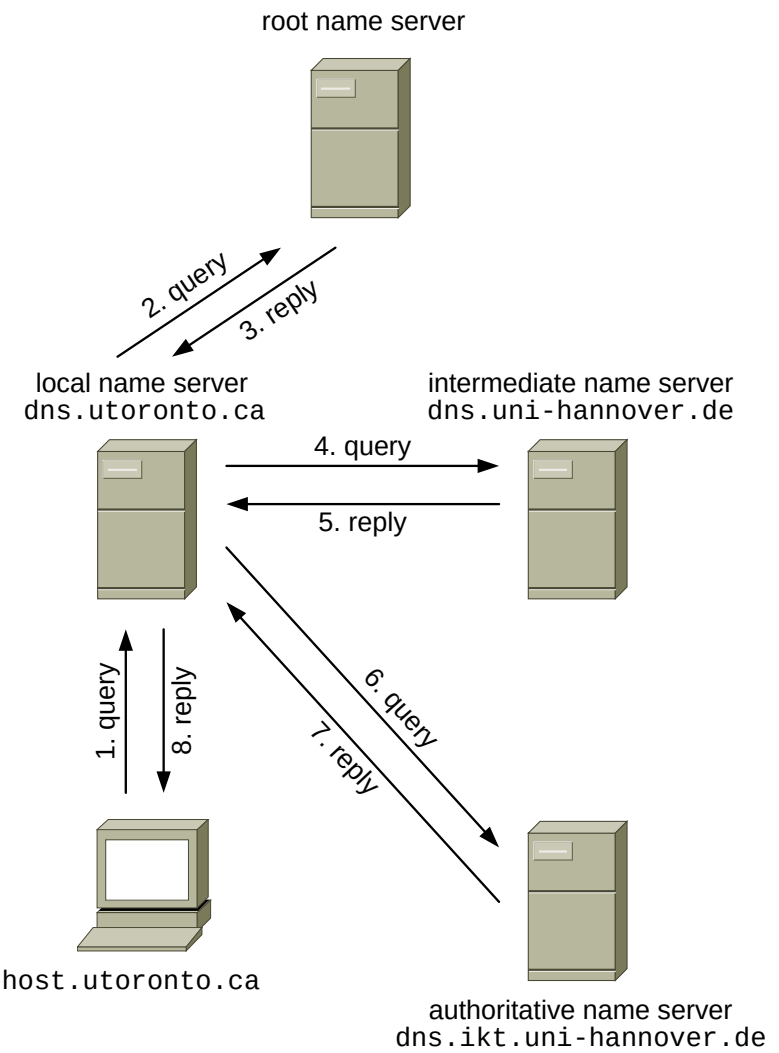


# Iterative queries



`host.utoronto.ca` queries for `host.ikt.uni-hannover.de`

3. root returns intermediate to the local name server
4. local name server queries intermediate name server
5. intermediate returns authoritative name server
6. local name server queries authoritative name server
7. authoritative name server replies IP-address of `host.ikt.uni-hannover.de`
8. reply to requesting host

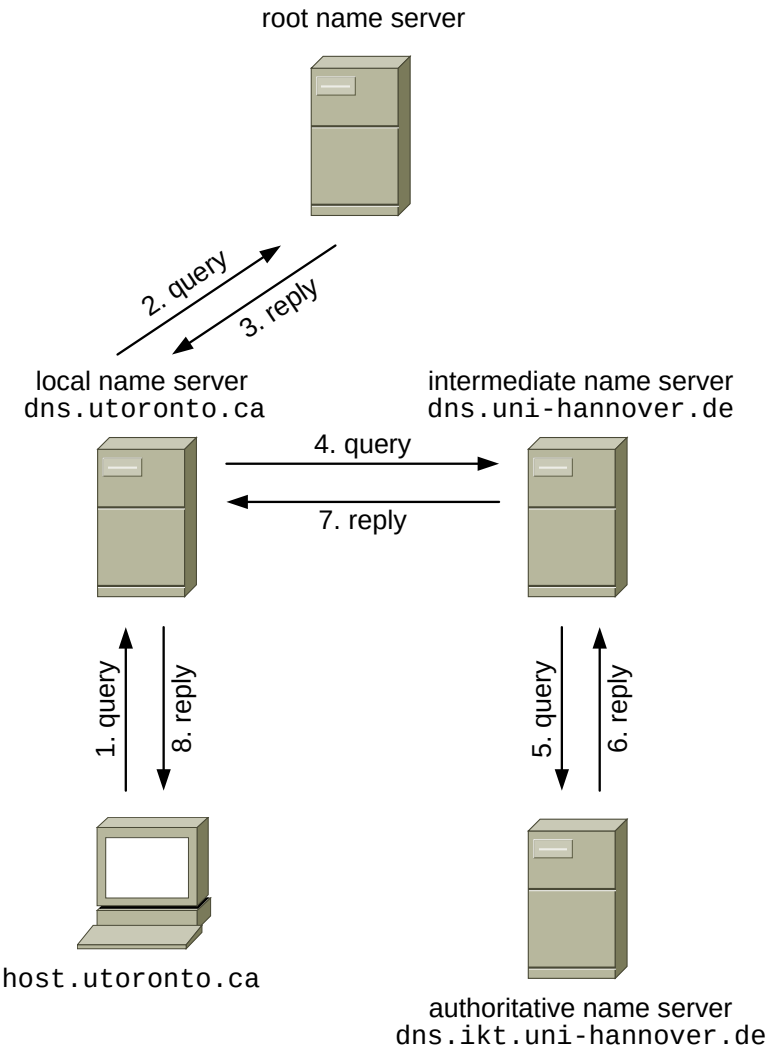


# Mixed recursive and iterative queries



`host.utoronto.ca` queries for `host.ikt.uni-hannover.de`

3. root returns intermediate to the local name server
4. local name server queries intermediate name server
5. intermediate queries authoritative name server
6. authoritative replies IP-address of `host.ikt.uni-hannover.de` to intermediate
7. reply to local name server
8. reply to requesting host



# DNS records



The entries in the DNS data bases are called resource records (RR) consisting of name, value, type and time-to-live (expiry in caches)

- type = A (address)
  - name = hostname
  - value = IP address
- type = CNAME (canonical name)
  - name = alias hostname
  - value = canonical hostname
- type = NS (name server)
  - name = domain
  - value = hostname of authoritative name server
- type = MX (mail exchange)
  - name = domain
  - value = hostname of mail server

# DNS messages



DNS messages contain among other fields

- question section: information about the query that is made
  - name
  - type
- answers section: in a reply; contains resource records
  - value
  - time-to-live
- additional section: contains other "helpful" records, e.g.
  - if the query is of type NS (name server)
  - the answers section contains the NS record i.e. hostname
  - the additional section contains the type A record, i.e. IP-address of the name server