Computer Networks

Instructor: Niklas Carlsson

Email: niklas.carlsson@liu.se

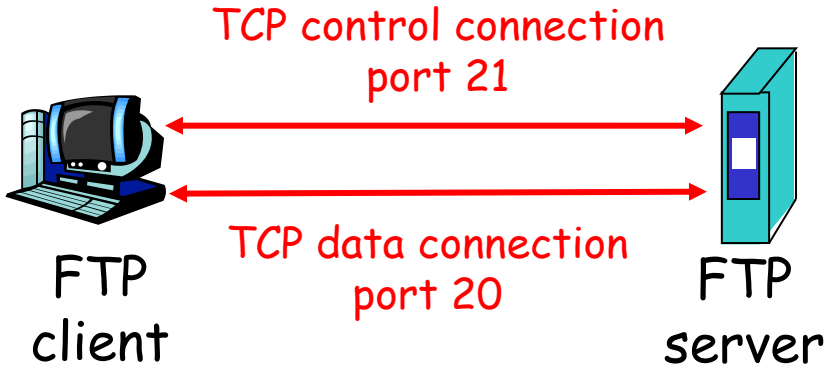
Notes derived from "*Computer Networking: A Top Down Approach*", by Jim Kurose and Keith Ross, Addison-Wesley.

The slides are adapted and modified based on slides from the book's companion Web site, as well as modified slides by Anirban Mahanti and Carey Williamson.

FTP

File Transfer Protocol (FTP)

- FTP client contacts FTP server at port 21, specifying TCP as transport protocol
- Client obtains authorization over control connection
- Client browses remote directory by sending commands over control connection.
- When server receives a command for a file transfer, the server opens a TCP data connection to client
- After transferring one file, server closes connection.



- Server opens a second TCP data connection to transfer another file.
- Control connection: "out of band"
- FTP server maintains "state": current directory, earlier authentication

FTP commands, responses

Sample commands:

- sent as ASCII text over control channel
- **`USER`** ***`username`***
- **`PASS`** ***`password`***
- **`LIST`** return list of file in current directory
- **`RETR filename`** retrieves (gets) file
- **`STOR filename`** stores (puts) file onto remote host

Sample return codes

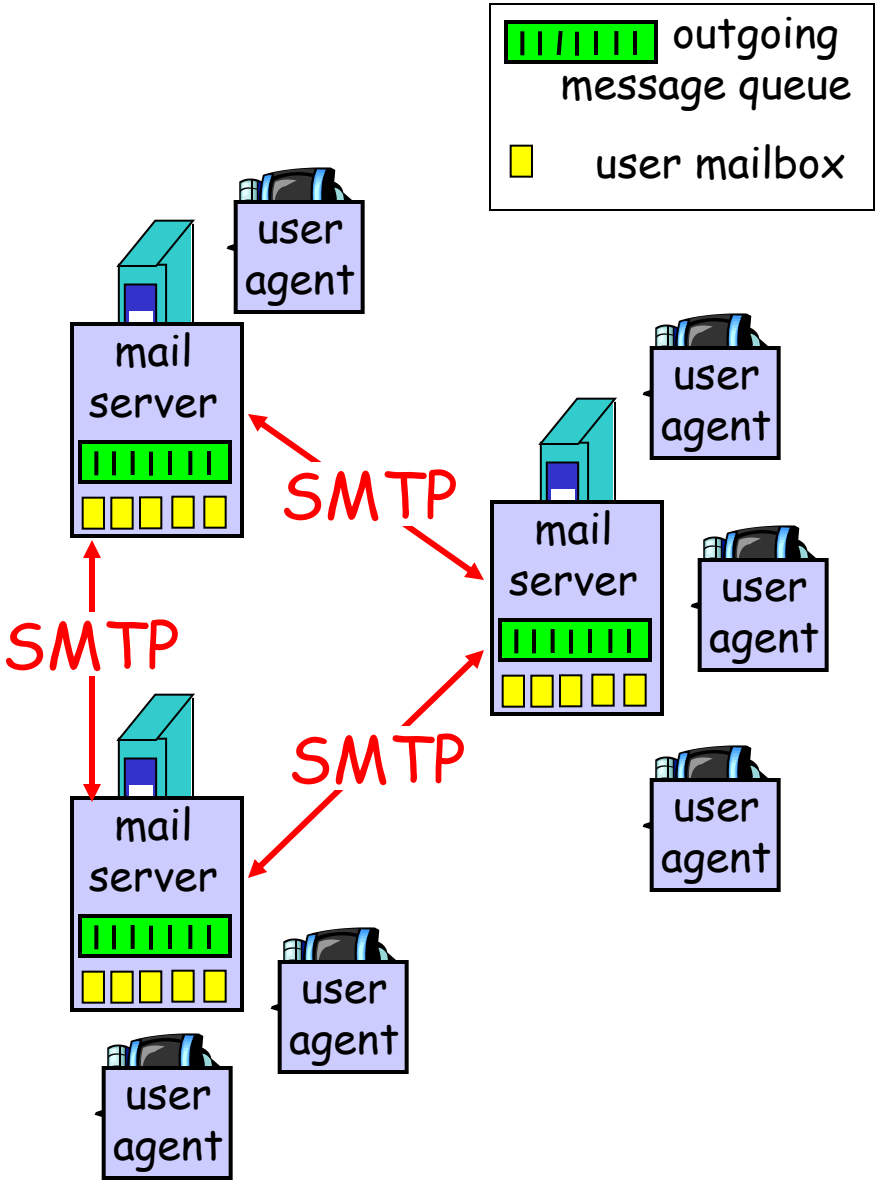
- status code and phrase (as in HTTP)
- **`331 Username OK, password required`**
- **`125 data connection already open; transfer starting`**
- **`425 Can't open data connection`**
- **`452 Error writing file`**

Mail

Electronic Mail

Three major components:

- user agents
  - e.g., Eudora, Outlook, Pine, Netscape Messenger
- mail servers
  - Incoming, outgoing messages
- Simple Mail Transfer Protocol: SMTP



Electronic Mail: SMTP [RFC 2821]

- Client's SMTP mail server establishes a TCP connection to the recipients SMTP server using Port 25
- three phases in messg. transfer
  - handshaking (greeting)
  - transfer of messages
  - closure
- command/response interaction
  - commands: ASCII text
  - response: status code and phrase
- messages must be in 7-bit ASCII

Sample SMTP interaction

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250  Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

Try SMTP interaction for yourself:

- **`telnet servername 25`**
- see 220 reply from server
- enter HELO, MAIL FROM, RCPT TO, DATA, QUIT commands

above lets you send email without using email client (reader)

SMTP: final words

- SMTP uses persistent connections
- SMTP requires message (header & body) to be in 7-bit ASCII
- SMTP server uses `CRLF.CRLF` to determine end of message
- SMPT is a "chatty" protocol

Comparison with HTTP:

- HTTP: pull
- SMTP: push
- both have ASCII command/response interaction, status codes
- HTTP: each object encapsulated in its own response msg
- SMTP: multiple objects sent in multipart msg

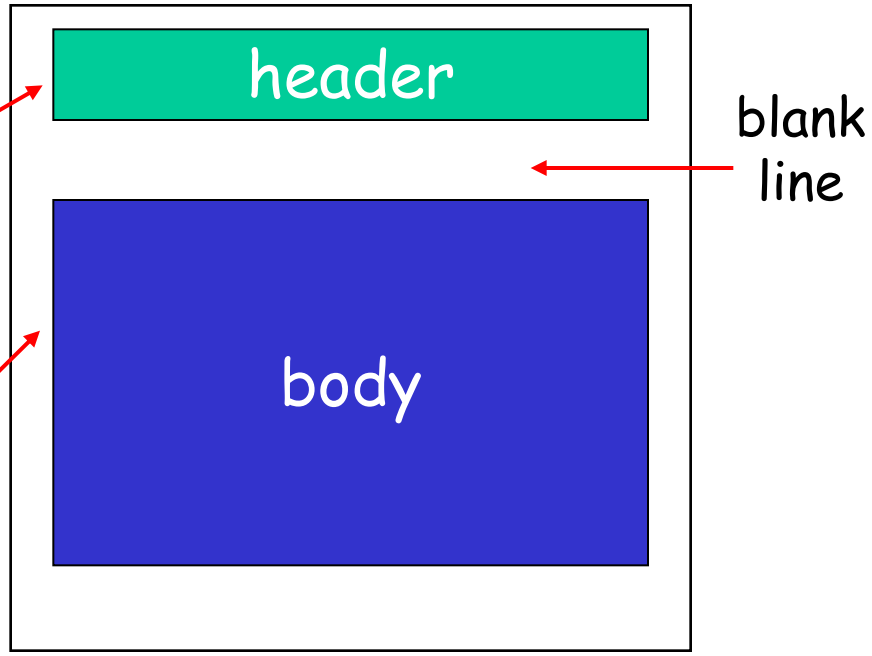
Mail message format

SMTP: protocol for exchanging email msgs

RFC 822: standard for text message format:

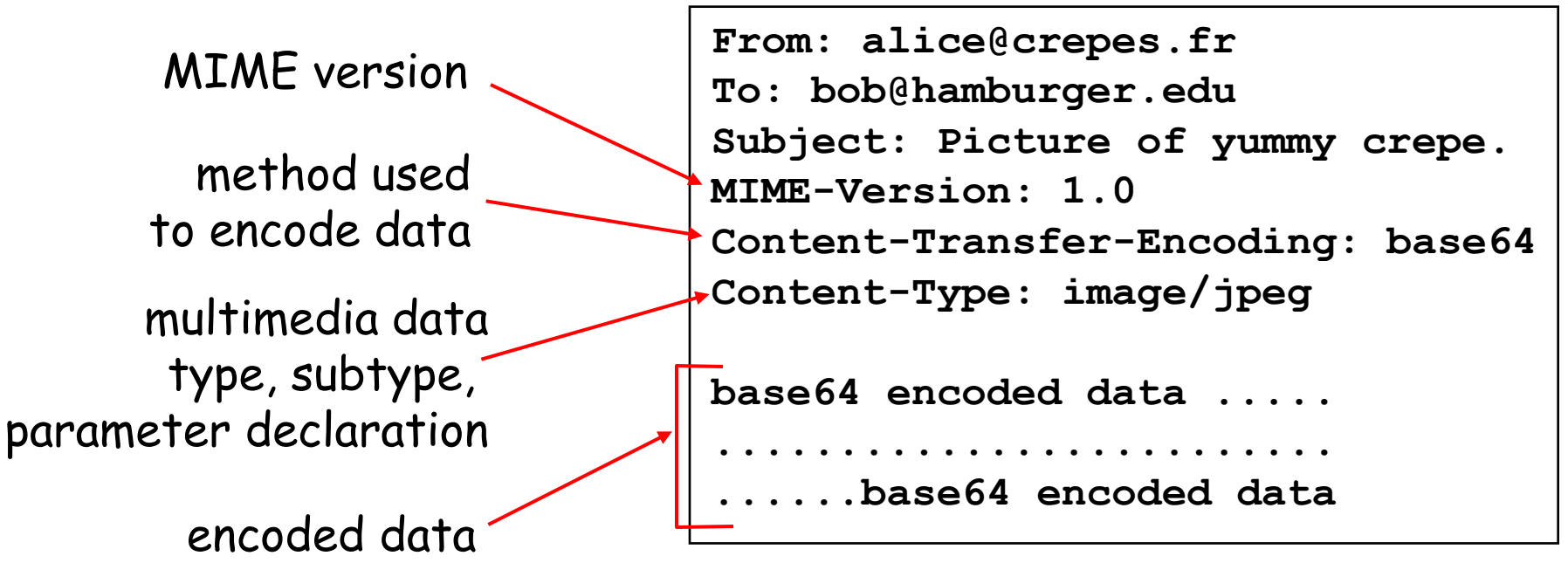
- header lines, e.g.,
  - To:
  - From:
  - Subject:

  different from SMTP commands!
- body
  - the "message", ASCII characters only

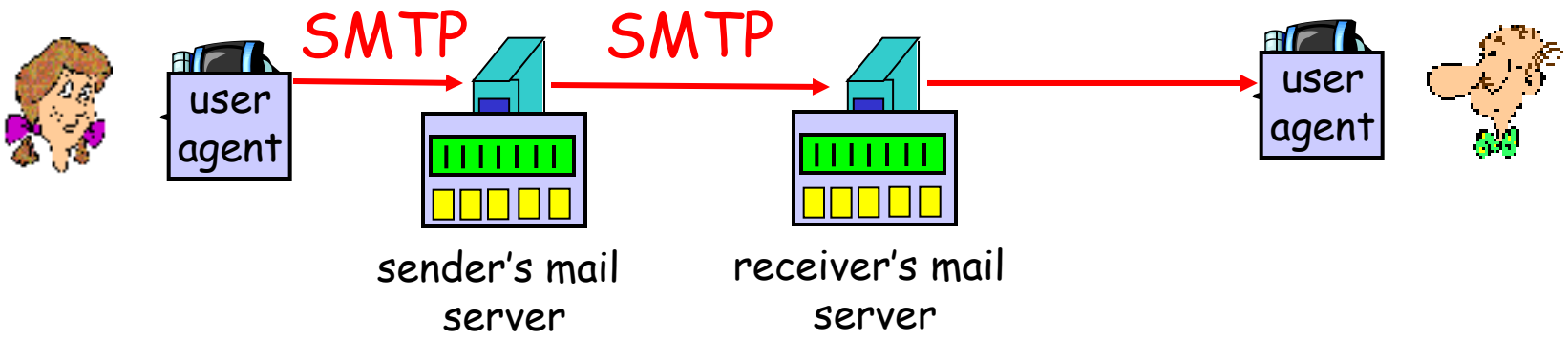


Message format: multimedia extensions

- MIME: multimedia mail extension, RFC 2045, 2056
- additional lines in msg header declare MIME content type

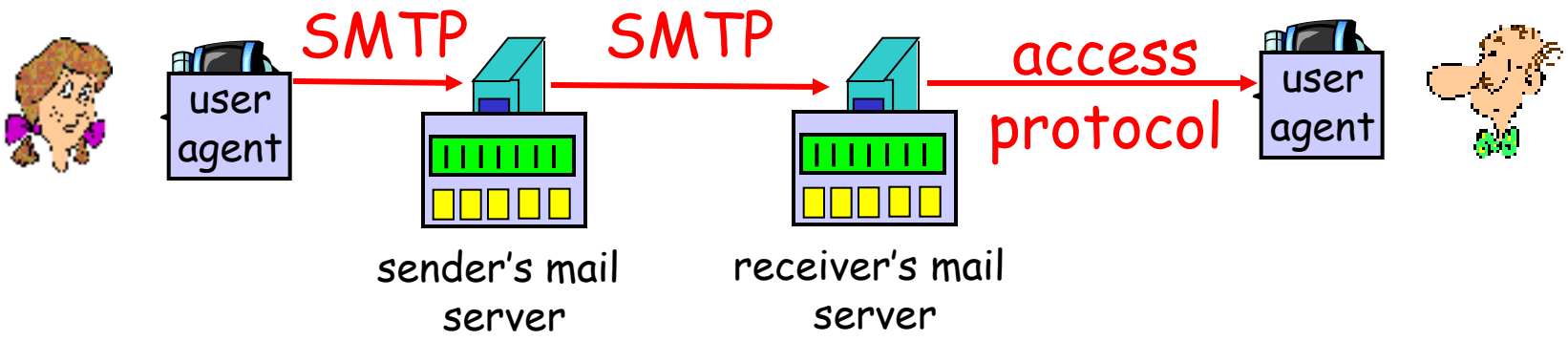


Mail access protocols



- SMTP is a push protocol. How will a user access emails?

Mail access protocols



- SMTP is a push protocol. How will a user access emails?
- Mail access protocol: retrieval from server
  - POP: Post Office Protocol [RFC 1939]
    - Users can't create folders on mail server
  - IMAP: Internet Mail Access Protocol [RFC 1730]
    - more features (more complex)
    - manipulation of stored msgs on server
  - HTTP: Hotmail , Yahoo! Mail, etc.

DNS

DNS: Domain Name System

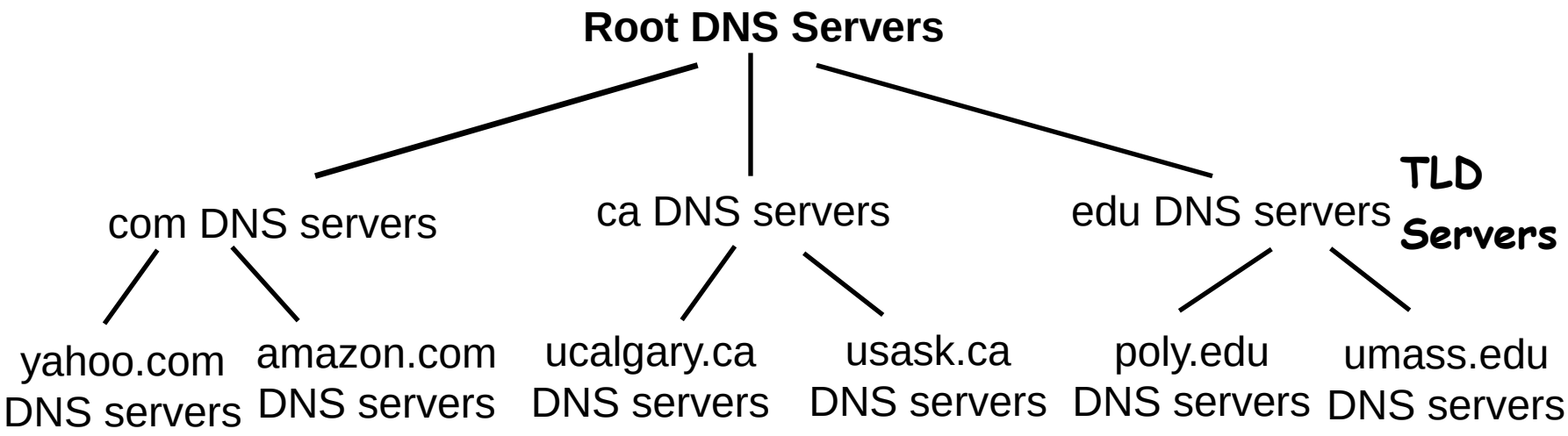
Internet hosts:

- IP address (32 bit) - used for addressing datagrams
- "name", e.g., www.yahoo.com - used by humans

DNS: provides translation between host name and IP address

- *distributed database* implemented in hierarchy of many *name servers*
- *distributed for scalability & reliability*

Distributed, Hierarchical Database



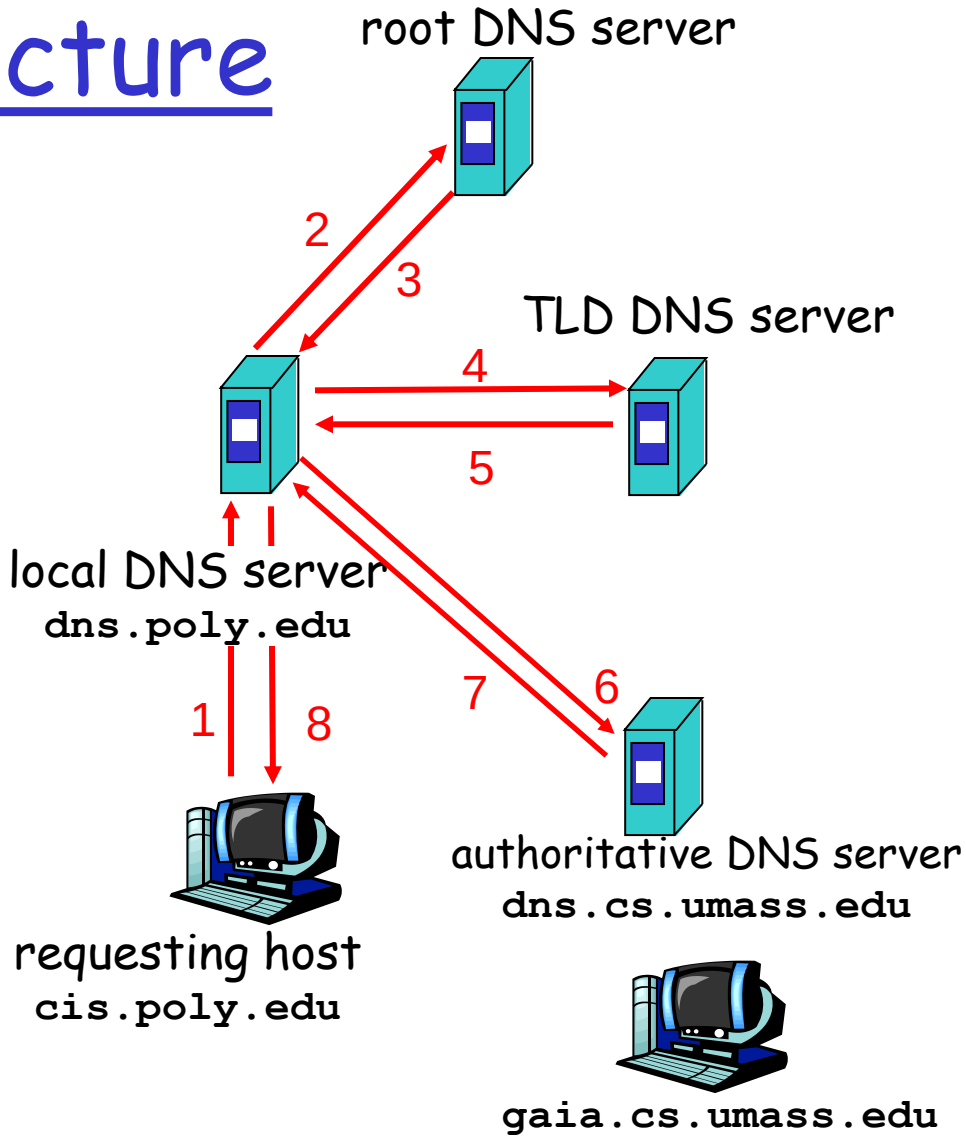
- Root servers and TLD servers typically do not contain hostname to IP mappings; they contain mappings for locating authoritative servers.

DNS Services

- Hostname to IP address translation
- Host aliasing
  - Canonical and alias names
- Mail server aliasing
- Load distribution
  - Replicated Web servers: set of IP addresses for one canonical name

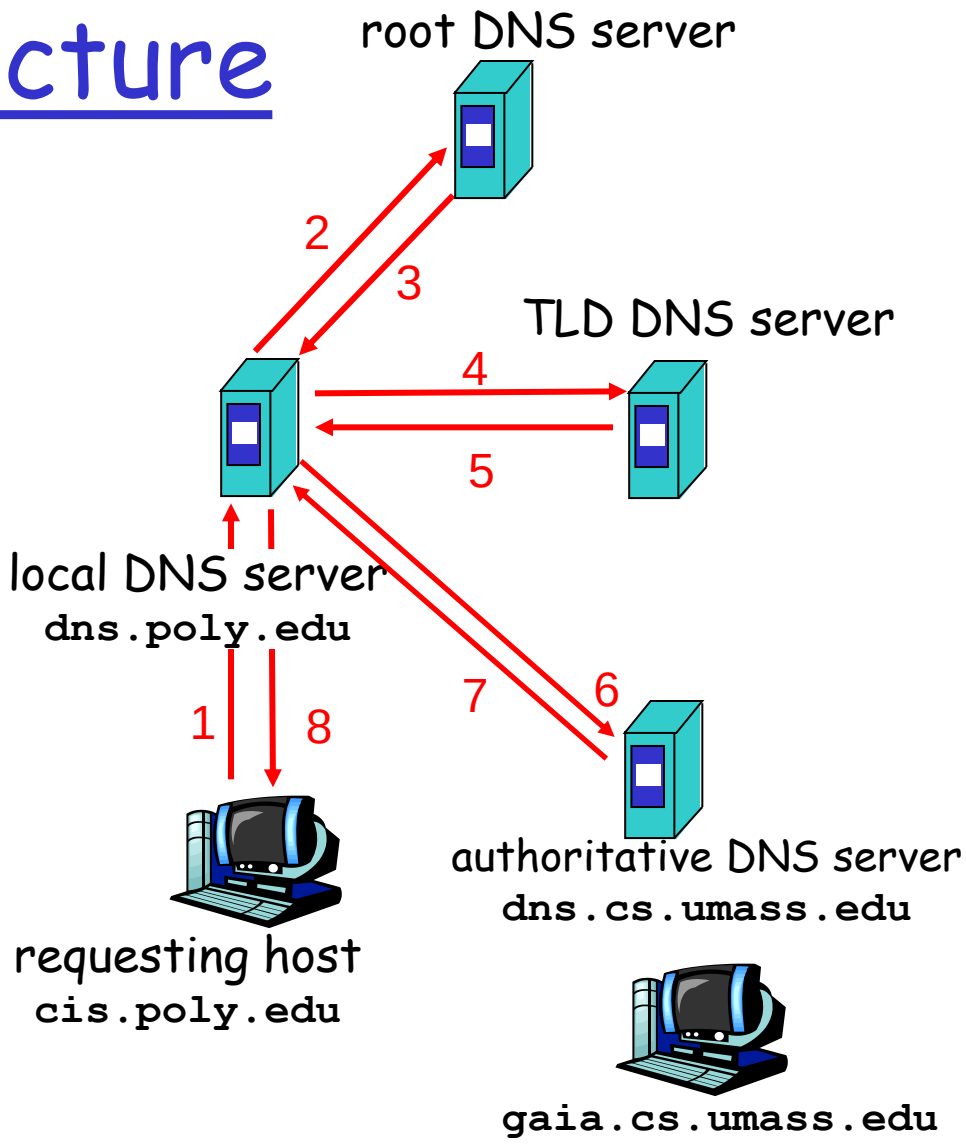
DNS Infrastructure

- Host at cis.poly.edu wants IP address for gaia.cs.umass.edu
- Infrastructure:
  - Client resolver
  - Local DNS server
  - Authoritative DNS Server
  - Root DNS Server
  - Top-Level Domain DNS Server
- Transport protocol?



DNS Infrastructure

- Host at cis.poly.edu wants IP address for gaia.cs.umass.edu
- Infrastructure:
  - Client resolver
  - Local DNS server
  - Authoritative DNS Server
  - Root DNS Server
  - Top-Level Domain DNS Server
- Transport protocol?
  - UDP (port 53)



DNS: Root name servers

- contacted by local name server that cannot resolve name directly
- root name server:
  - contacts authoritative name server if name mapping is not known
  - gets mapping
  - returns mapping to local name server

TLD and Authoritative Servers

- Top-level domain (TLD) servers: responsible for .com, .org, .net, .edu, .gov, .mil, and all top-level country domains (e.g., .uk, .fr, .ca, .jp)
  - Network Solutions maintains servers for .com TLD
  - Educause for .edu TLD
- Authoritative DNS servers: organization's DNS servers, providing authoritative hostname to IP mappings for organization's servers (e.g., Web and mail).
  - Can be maintained by organization or service provider

Local Name Server

- Each ISP (residential ISP, company, university) has one.
  - Also called "default name server"
- When a host makes a DNS query, query is sent to its local DNS server
  - Acts as a proxy, forwards query into hierarchy.
  - Reduces lookup latency for commonly searched hostnames

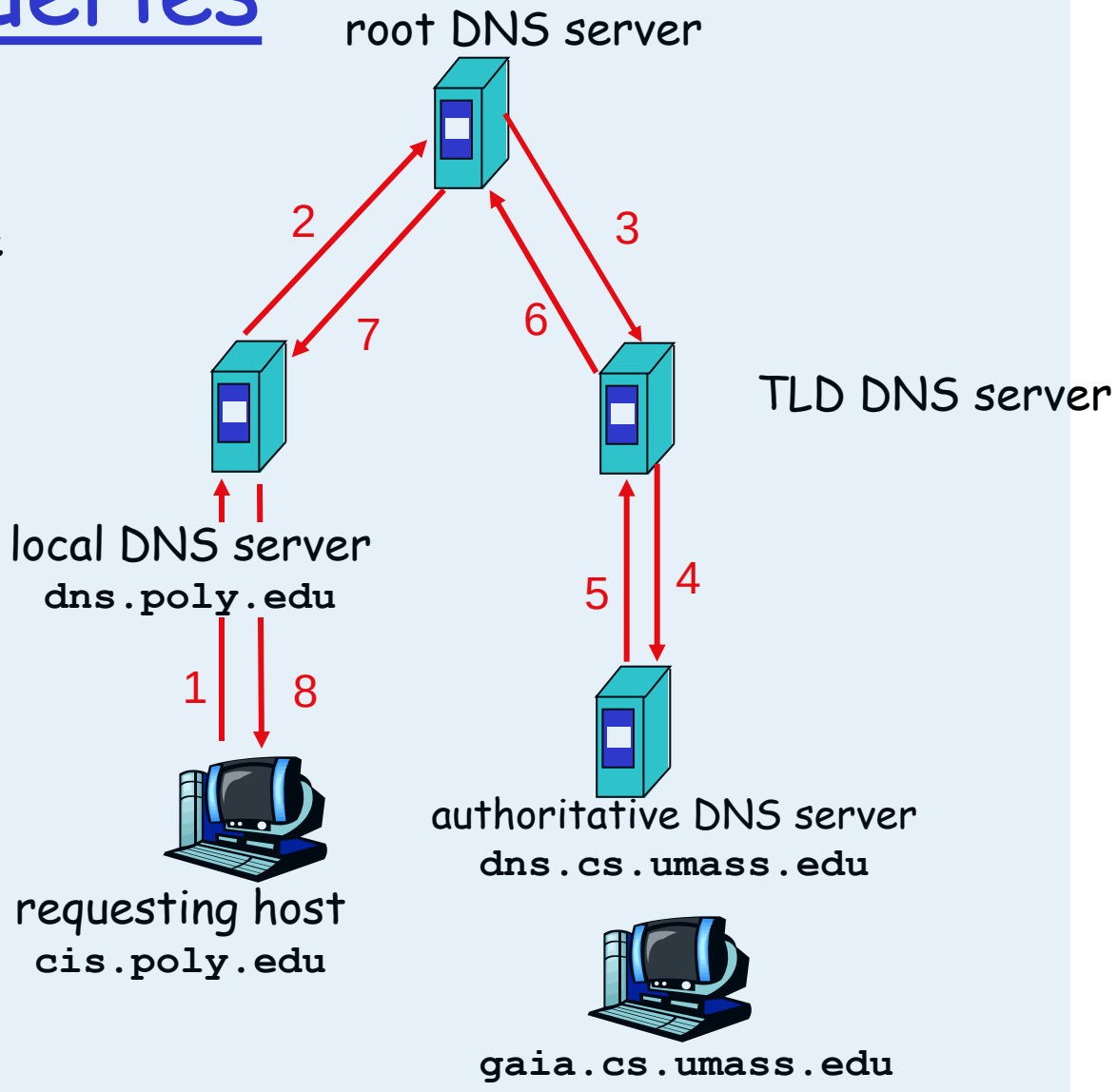
Recursive queries

recursive query:

- puts burden of name resolution on contacted name server
- heavy load?

iterated query:

- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



DNS: caching and updating records

- once (any) name server learns mapping, it *caches* mapping
  - cache entries timeout (disappear) after some time called the Time To Live (TTL)
  - TLD servers typically cached in local name servers
    - Thus root name servers not often visited

DNS records

DNS: distributed db storing resource records (RR)

RR format: **(name, value, type, ttl)**

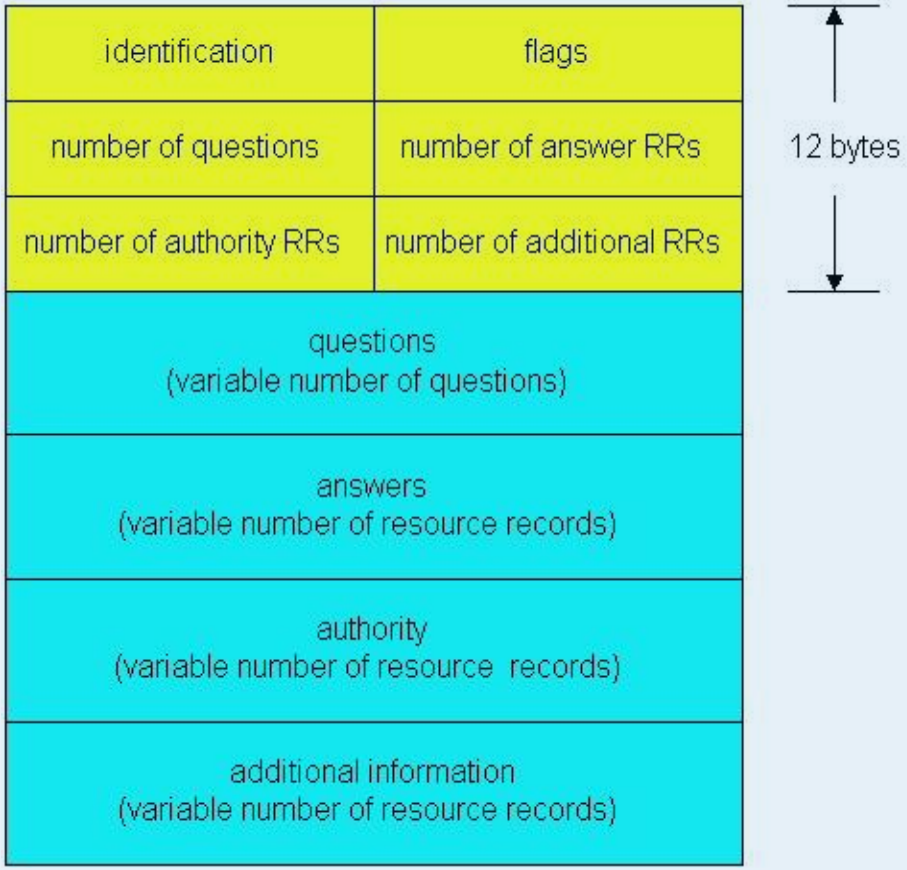
- Type=A
  - **name** is hostname
  - **value** is IP address
- Type=NS
  - **name** is domain (e.g. foo.com)
  - **value** is IP address of authoritative name server for this domain
- Type=CNAME
  - **name** is alias name for some "canonical" (the real) name
    `www.ibm.com` is really
    `servereast.backup2.ibm.com`
  - **value** is canonical name
- Type=MX
  - **value** is name of mail server associated with **name**

DNS protocol, messages

DNS protocol : *query* and *reply* messages, both with same *message format*

msg header

- identification: 16 bit # for query, reply to query uses same #
- flags:
  - query or reply
  - recursion desired
  - recursion available
  - reply is authoritative



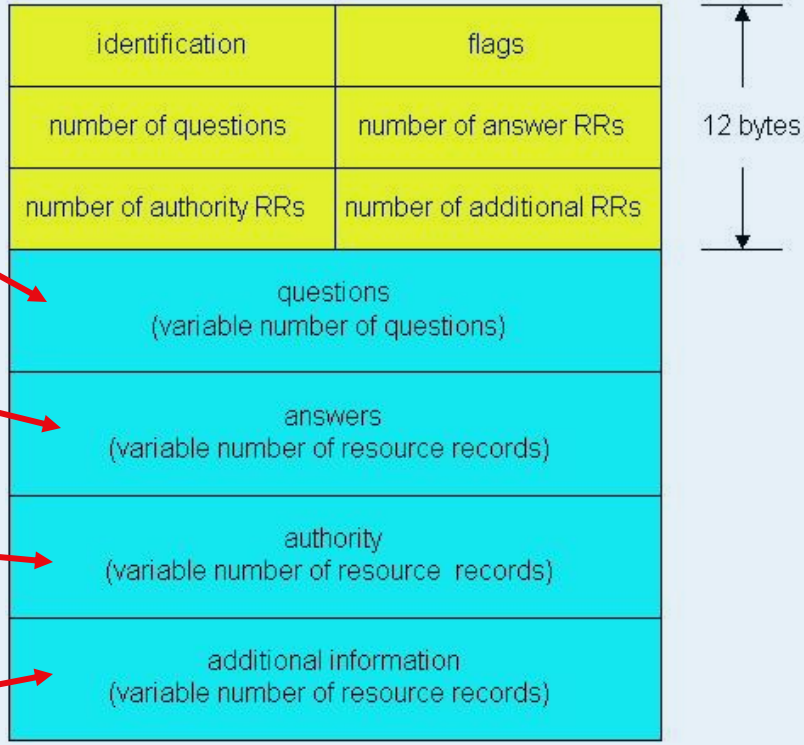
DNS protocol, messages

Name, type fields for a query

RRs in response to query

records for authoritative servers

additional "helpful" info that may be used



DNS messages are carried using UDP on port 53

Inserting records into DNS

- Example: just created startup "Network Utopia"
- Register name networkutopia.com at a registrar (e.g., Network Solutions)
  - Need to provide registrar with names and IP addresses of your authoritative name server (primary and secondary)
  - Registrar inserts two RRs into the com TLD server:

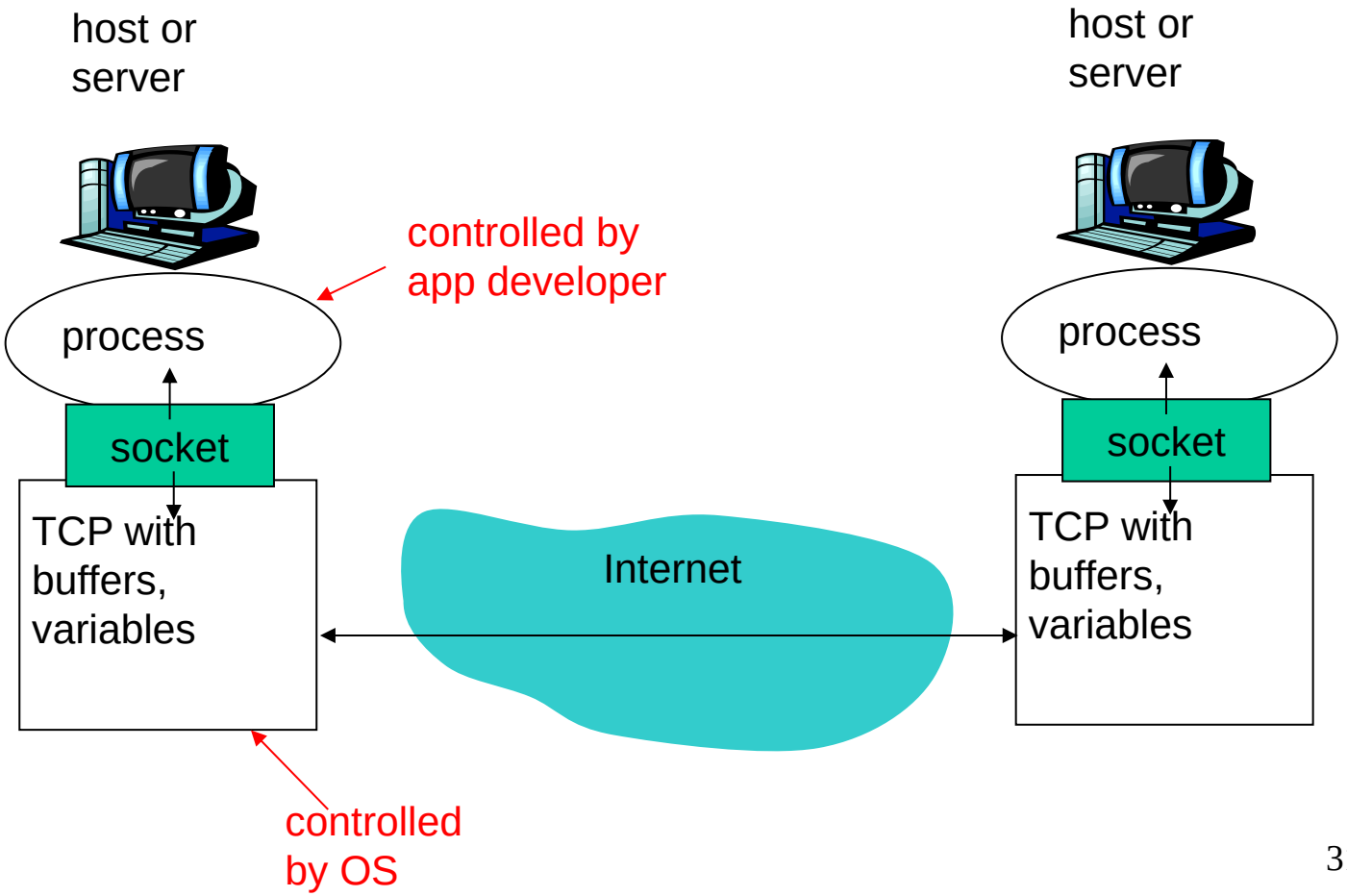
```
(networkutopia.com, dns1.networkutopia.com, NS)
(dns1.networkutopia.com, 212.212.212.1, A)
```

- Put in authoritative server Type A record for www.networkutopia.com and Type MX record for networkutopia.com
- How do people get the IP address of your Web site?

Socket programming

Sockets (recall)

- process sends/receives messages to/from its socket



Socket programming

Goal: learn how to build client/server application that communicate using sockets

Socket API

- explicitly created, used, released by apps
- client/server paradigm
- two types of transport service via socket API:
  - unreliable datagram
  - reliable, byte stream-oriented

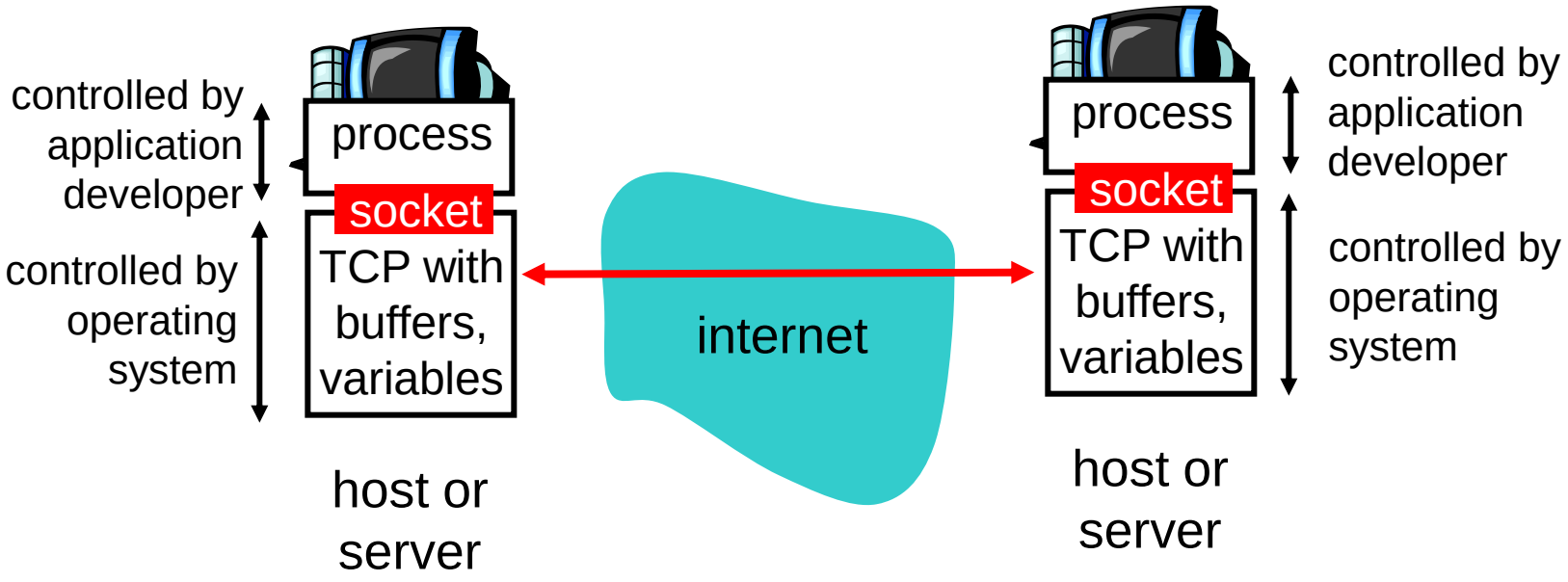
socket

a *host-local*, *application-created*, *OS-controlled* interface (a "door") into which application process can both send and receive messages to/from another application process

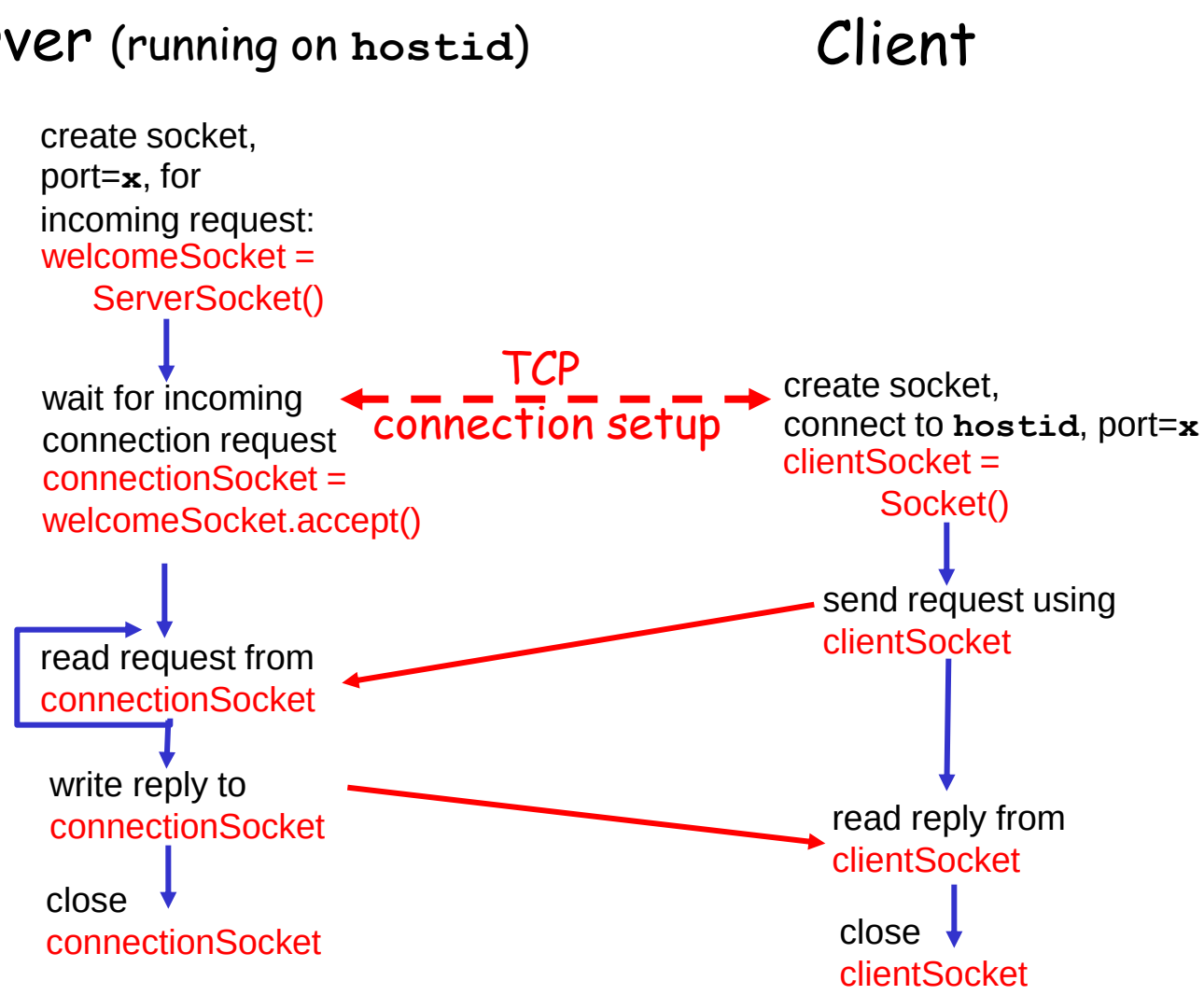
Socket-programming using TCP

Socket: a door between application process and end-end-transport protocol (UDP or TCP)

TCP service: reliable transfer of *bytes* from one process to another

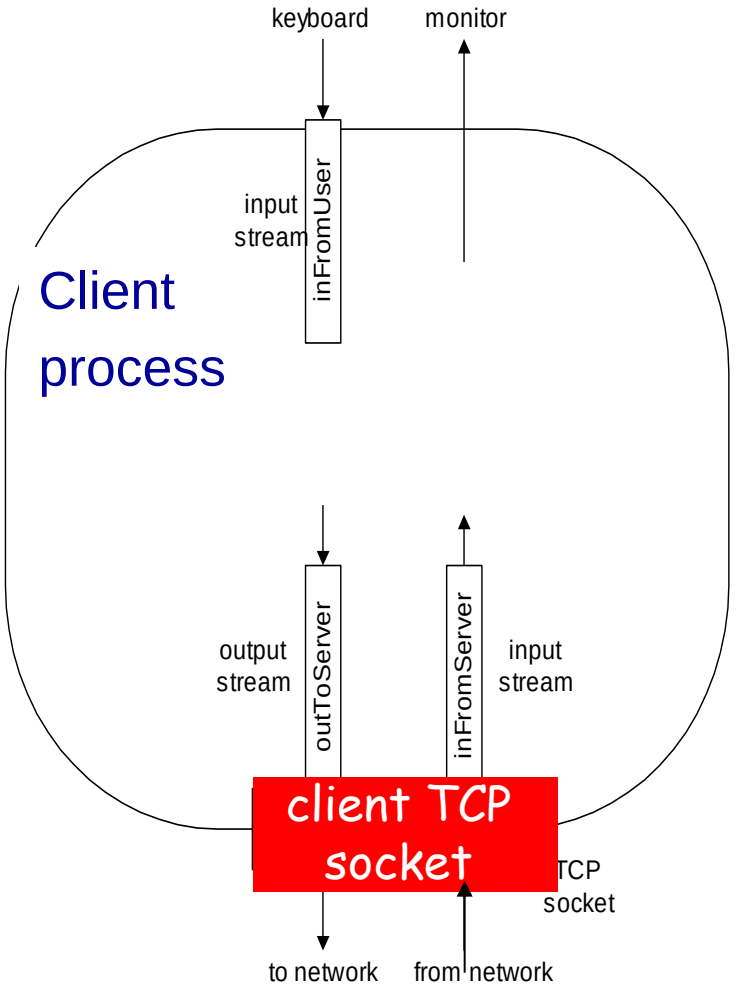


Client/server socket interaction: TCP



Stream jargon

- stream is a sequence of characters that flow into or out of a process.
- input stream is attached to some input source for the process, e.g., keyboard or socket.
- output stream is attached to an output source, e.g., monitor or socket.



Socket programming *with UDP*

UDP: no "connection" between client and server

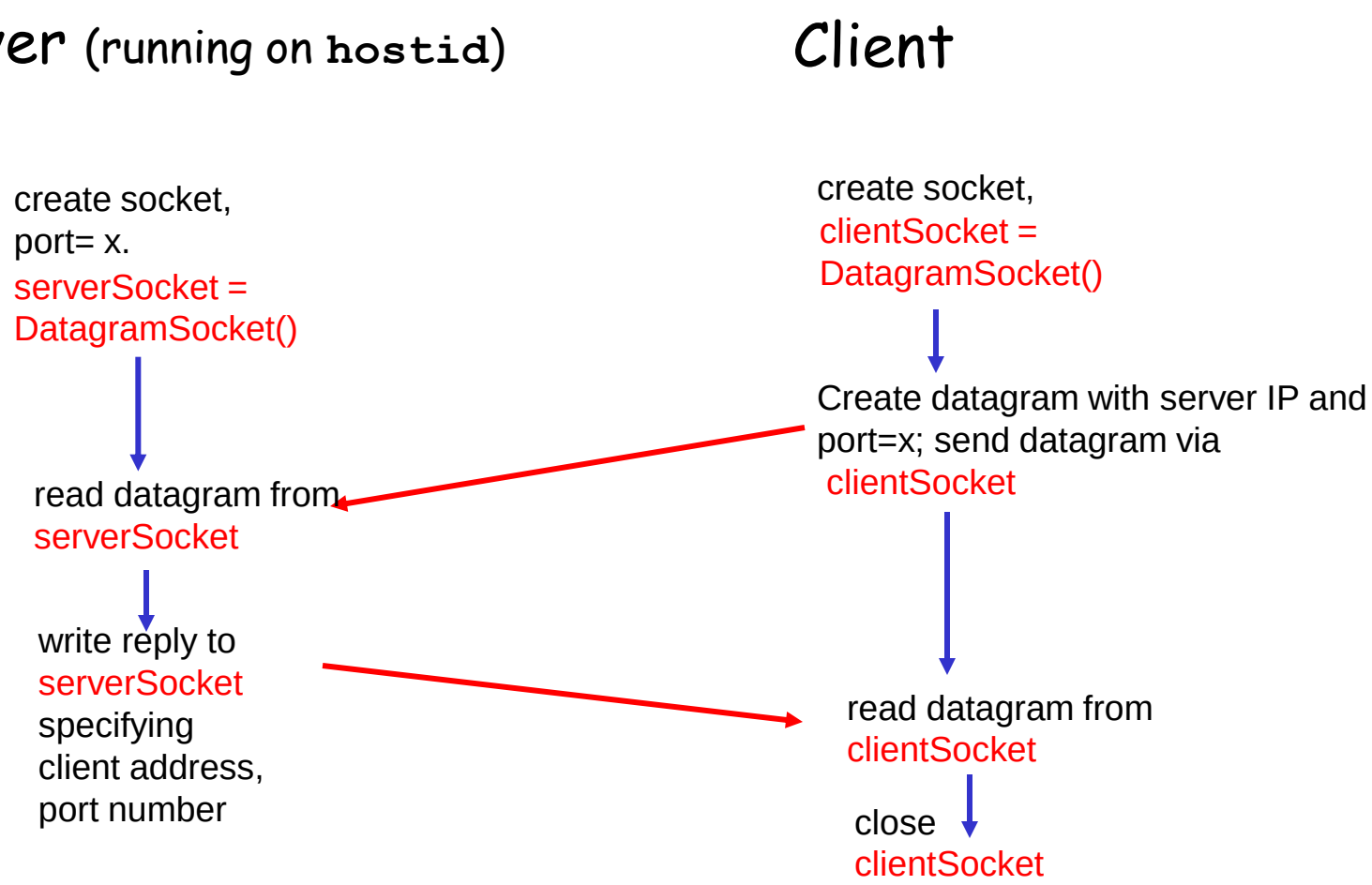
- no handshaking
- sender explicitly attaches IP address and port of destination to each packet
- server must extract IP address, port of sender from received packet

UDP: transmitted data may be received out of order, or lost

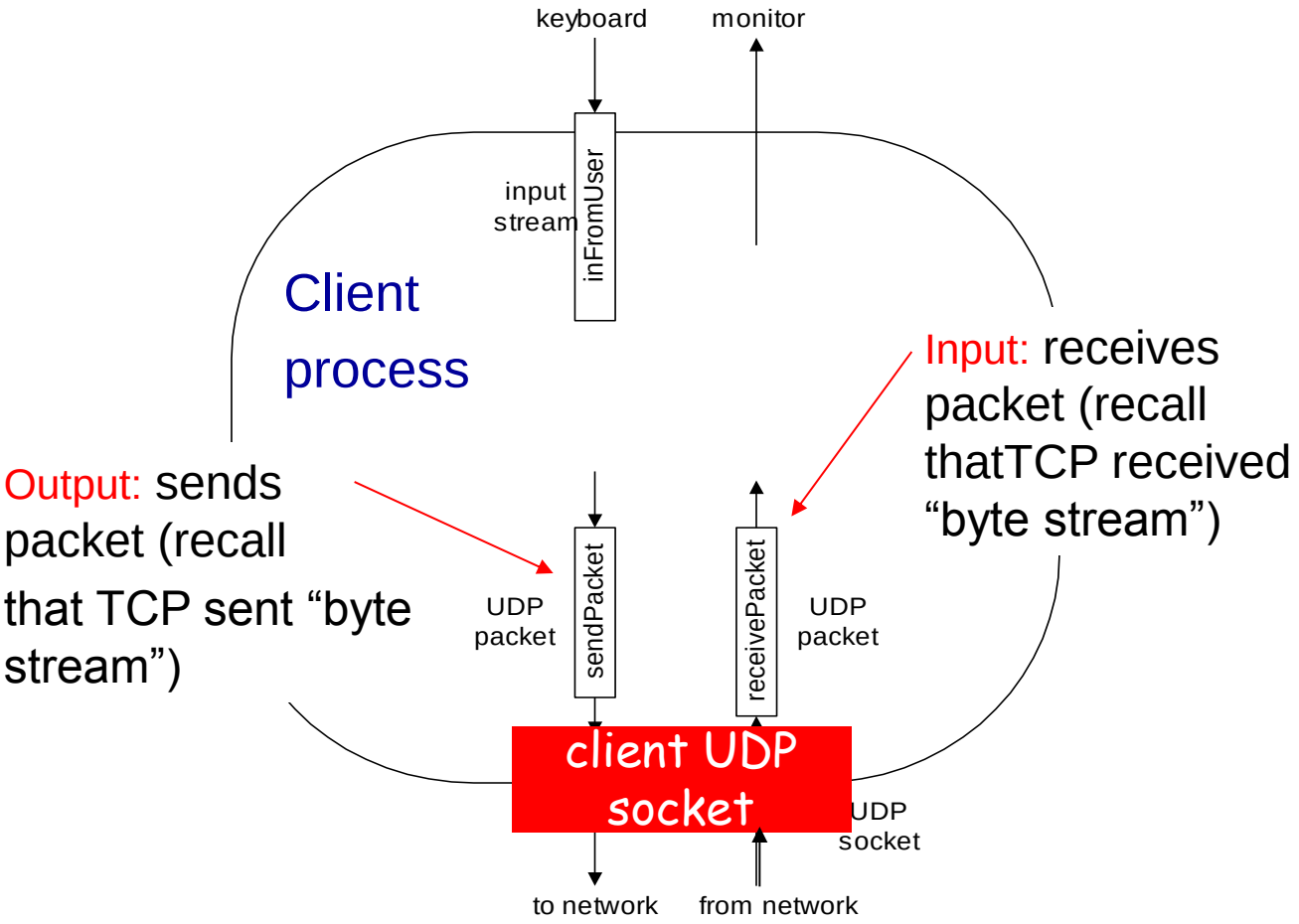
application viewpoint:

UDP provides unreliable transfer of groups of bytes ("datagrams") between client and server

Client/server socket interaction: UDP



Example: client (UDP)



Chapter 2: Summary

- application architectures
  - client-server
  - P2P
  - hybrid
- application service requirements:
  - reliability, bandwidth, delay
- Internet transport service model
  - connection-oriented, reliable: TCP
  - unreliable, datagrams: UDP

- specific protocols:
  - HTTP
  - FTP
  - SMTP, POP, IMAP
  - DNS
  - P2P: BitTorrent, Skype
- socket programming

Chapter 2: Summary

some important lessons about *protocols*

- typical request/reply message exchange:
  - client requests info or service
  - server responds with data, status code
- message formats:
  - headers: fields giving info about data
  - data: info being communicated

Important themes:

- control vs. data msgs
  - in-band, out-of-band
- centralized vs. decentralized
- stateless vs. stateful
- reliable vs. unreliable msg transfer
- "complexity at network edge"