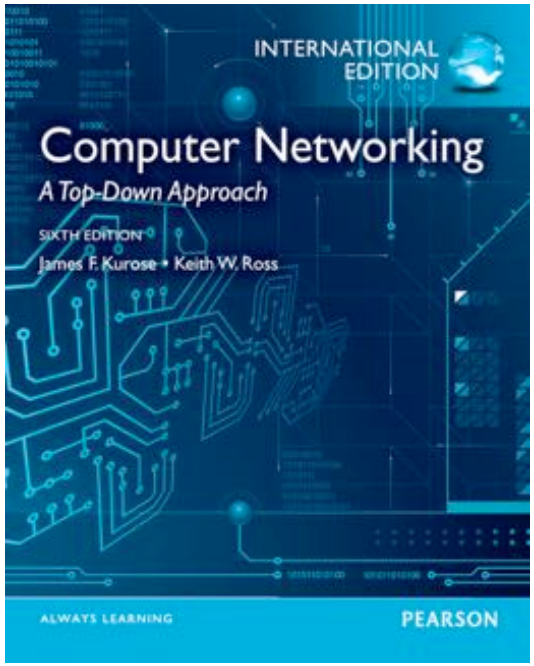
Application layer
Chapter 2

Kjersti Moldeklev, Prof II
Department of Telematics
kjmoldek@item.ntnu.no



The *content* of most of these slides is based on slides available from the web-site of the book by J.F Kurose and K.W. Ross. are

This week, January 18-19

Week	Date & Time	Topic	Room	Responsible	**Remark (ref.to 6th edition)**
2	Thursday 12:15 – 14:00	Practical Information; Course Introduction Network and Internet Overview	R1	Norvald Kjersti	 Chapter 1 + brief introduction to Physical layer.
2	Friday 09:15 – 11:00	Network and Internet Overview (cont)	R1	Kjersti	Chapter 1 Chapter 8.1
3	Thursday 12:15 – 14:00 Thursday 14:15 – 15:00	Application Layer Theory Assignment 1: *Overview of Computer Networks and the Internet* Wireshark Lab 1: *Intro (optional but highly recommended!)*	R1 R1	Kjersti Assistants/ Ida/Norvald	Chapter 2 One must deliver and pass at least 5 of the 8 theory assignments.
3	Friday 09:15 – 11:00	Application Layer (cont)	R1	Kjersti	Chapter 2 Chapter 8.2-3 and 8.5.1

Application layer: Roadmap

2.1 Principles of network applications

2.2 Web and HTTP

~~2.3 FTP~~

2.4 Electronic mail

- SMTP, POP3, IMAP
- 8.2-3 + 8.5.1 Secure email

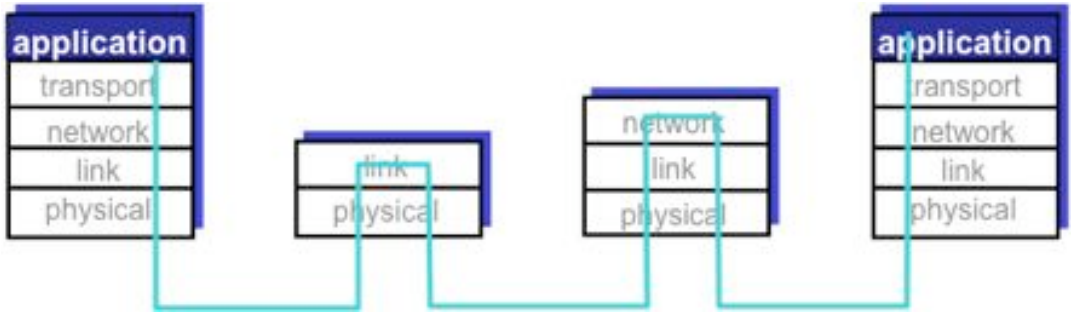
2.5 DNS

2.6 P2P applications

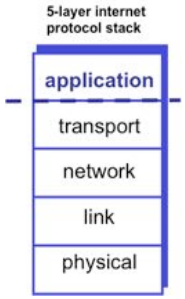
2.7 Socket programming with TCP

2.8 Socket programming with UDP

5-layer internet protocol stack



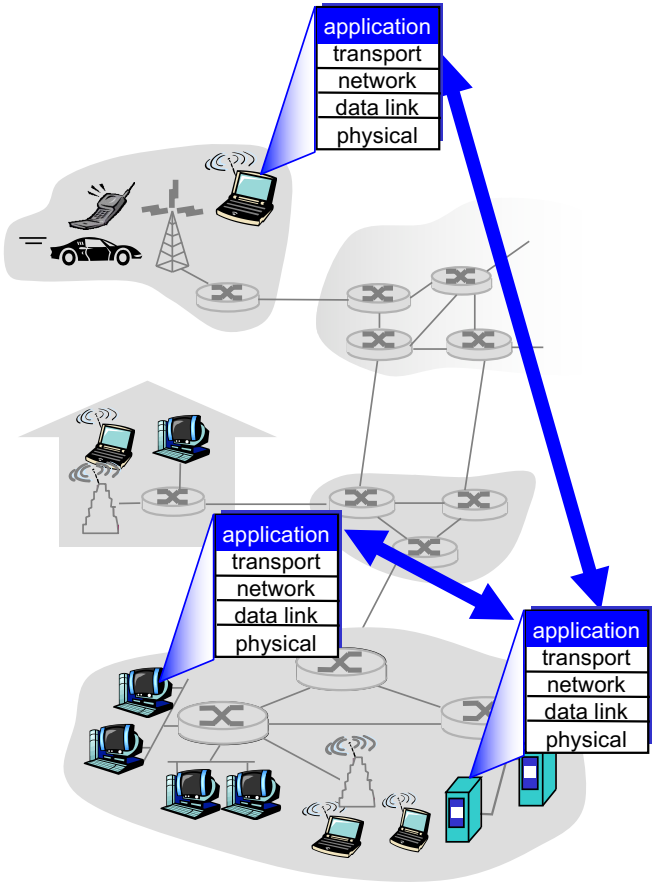
Application layer protocols and their interface towards the communication protocols in the operating system



- Conceptual, implementation aspects of distributed application protocols
  - **transport-layer service** models
  - **client-server** vs **peer-to-peer** paradigm
- Application-layer protocols
  - Web surfing: **HTTP**
  - ~~File transfer: FTP~~
  - Mail: **SMTP** (POP3 / IMAP)
  - Name server: **DNS**
  - ~~Peer-to-peer: BitTorrent~~
- Programming network applications
  - **socket API**

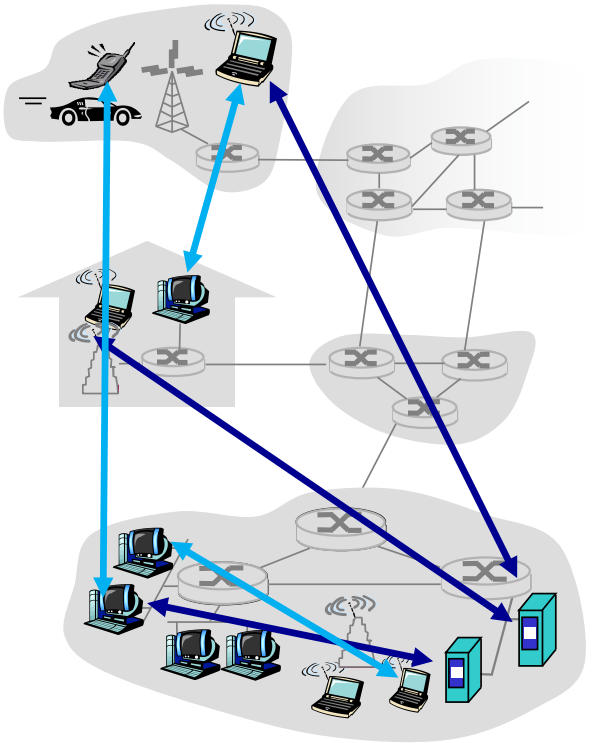
Distributed applications in end systems communicate over the network

- No need to write software for in-network devices
  - In-network devices do not run user applications
- Applications on end systems allow for rapid application development and propagation



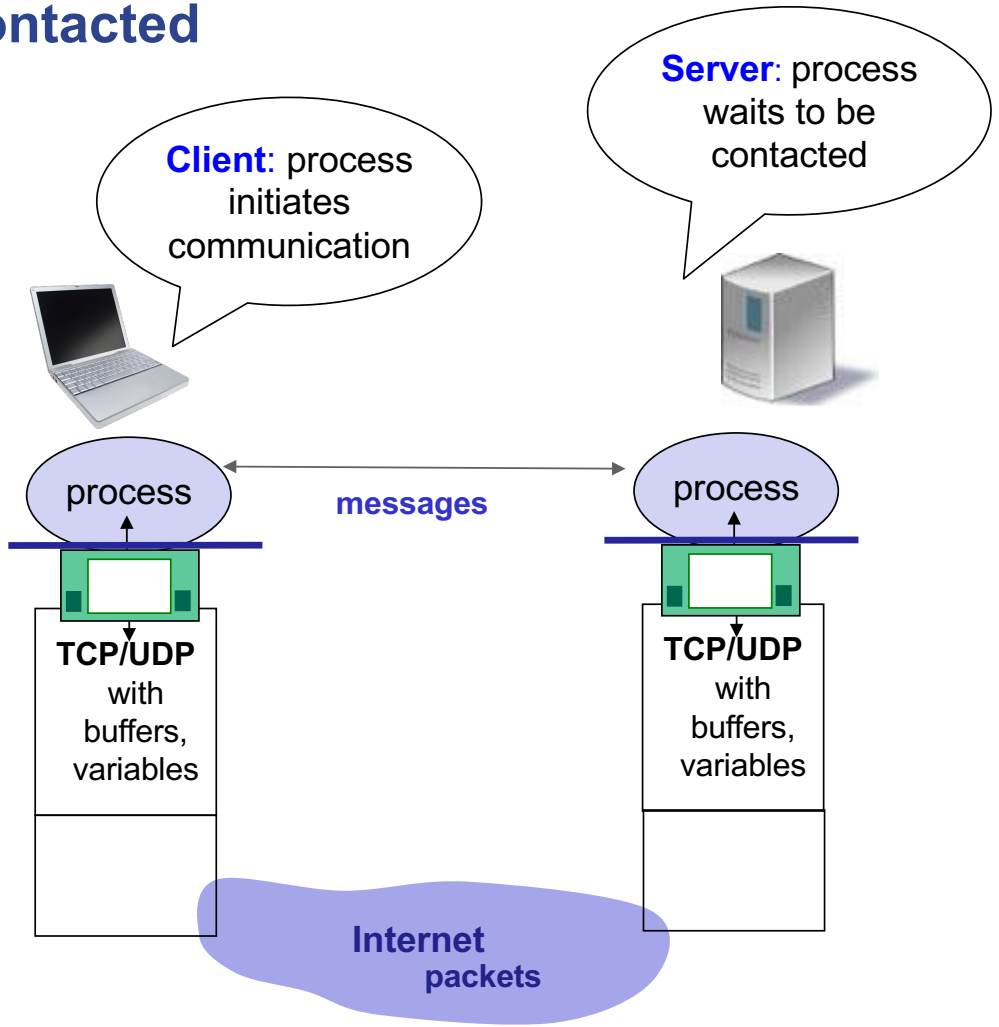
Applications run in the end systems of the network edge

- **Client/server model**
  - client requests a service from an always-on server
  - e.g. web browser/server, mail, file transfer, directories
- **Peer-peer model**
  - minimal (or no) use of dedicated servers
  - e.g. BitTorrent



Client initiates, server awaits to be contacted

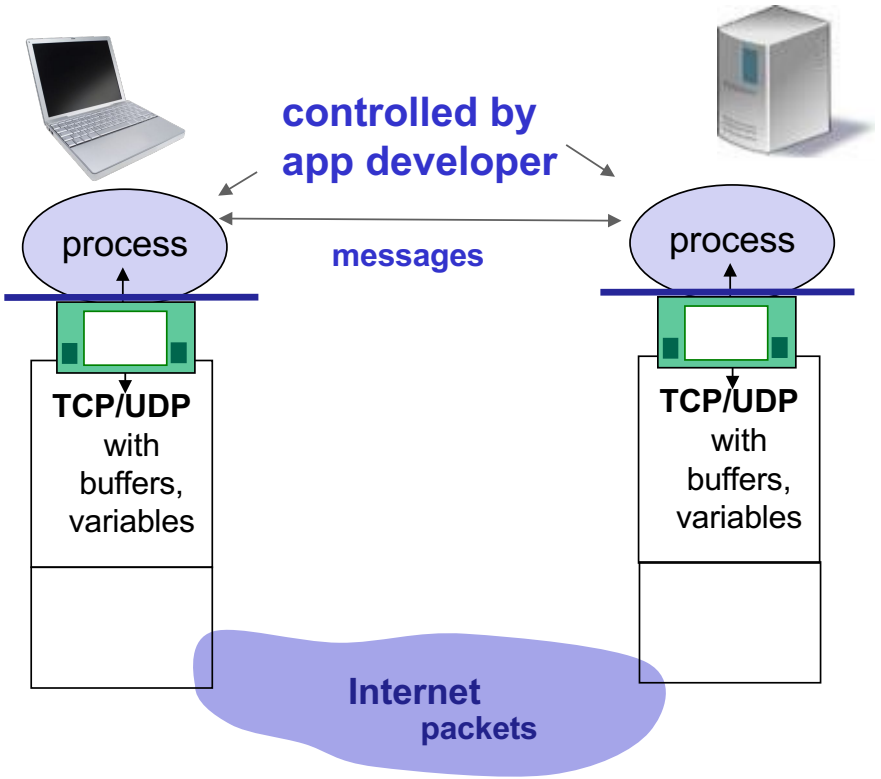
- **Process**: program running within a host
- **Inter-process communication** between two processes within same host (defined by operating system)
- Processes in **different hosts** communicate by **exchanging messages**
- Applications with P2P architectures have both client & server processes



TCP Transmission Control Protocol
UDP User Datagram Protocol

Processes send/receive messages to/from their socket

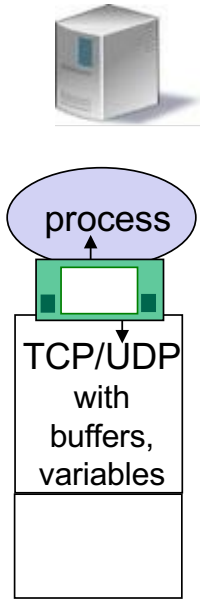
- **Socket** analogous to door
  - sending process shoves message out door
  - transport/network infrastructure brings message to socket at receiving process
- Accessed through **API** (**application programming interface**)
  - choice of transport protocol
  - ability to set a few parameters



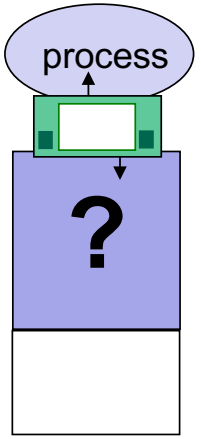
TCP Transmission Control Protocol
UDP User Datagram Protocol

Processes are addressed through: IP address + port number field in transport protocol

- Host device has unique 32-bit **IP address**
- Q: does IP address of host, on which process runs, suffice for identifying the process?
  - A: No, many processes can be running on same host
- **Port numbers** associated with process on host
- Example port numbers
  - FTP server: 21
  - HTTP server: 80
  - Mail server: 25
  - Domain Name Server: 53
- HTTP message to item.ntnu.no
  - IP address: 129.241.200.19
  - Port number: 80



What transport service does an application need?



Throughput

- multimedia apps require a minimum amount of throughput to be "effective"
- elastic applications make use of whatever throughput they get

Delay

- telephony, interactive games require low delay to be "effective"
- file transfers/large downloads bandwidth more important than delay

Data loss

- audio can tolerate some loss
- file transfer, telnet require 100% reliable data transfer

Security

- bank service, password: encryption, data integrity
- open web pages no such requirements

Different applications have different requirements

Application	Data loss	Throughput	Time sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5kbps-1Mbps video:10kbps-5Mbps	yes, 100's ms
stored audio/video	loss-tolerant	same as above	yes, few s
interactive games	loss-tolerant	few kbps up	yes, 100's ms
instant messaging	no loss	elastic	yes and no

Application protocols use an underlying transport protocol

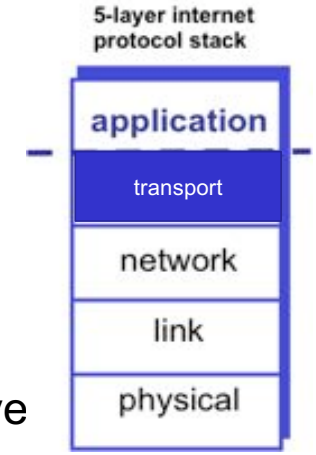
Application	Application layer protocol	Transport protocol
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	HTTP (e.g. Youtube), RTP [RFC 1889]	TCP or UDP
internet telephony	SIP, RTP, proprietary (e.g. Skype)	typically UDP

SMTP Simple Mail Transfer Protocol
HTTP HyperText Transfer Protocol
RFC Requests for comments

FTP File Transfer Protocol
RTP RealTime Protocol
SIP Session Initation Protocol

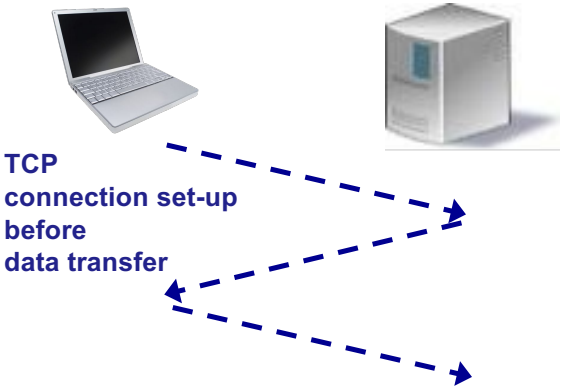
TCP Transmission Control Protocol
UDP User Datagram Protocol

Internet transport protocol services –> a short preview



TCP service

- **connection-oriented**: setup required between client and server processes
- **reliable** transport between sender and receiver

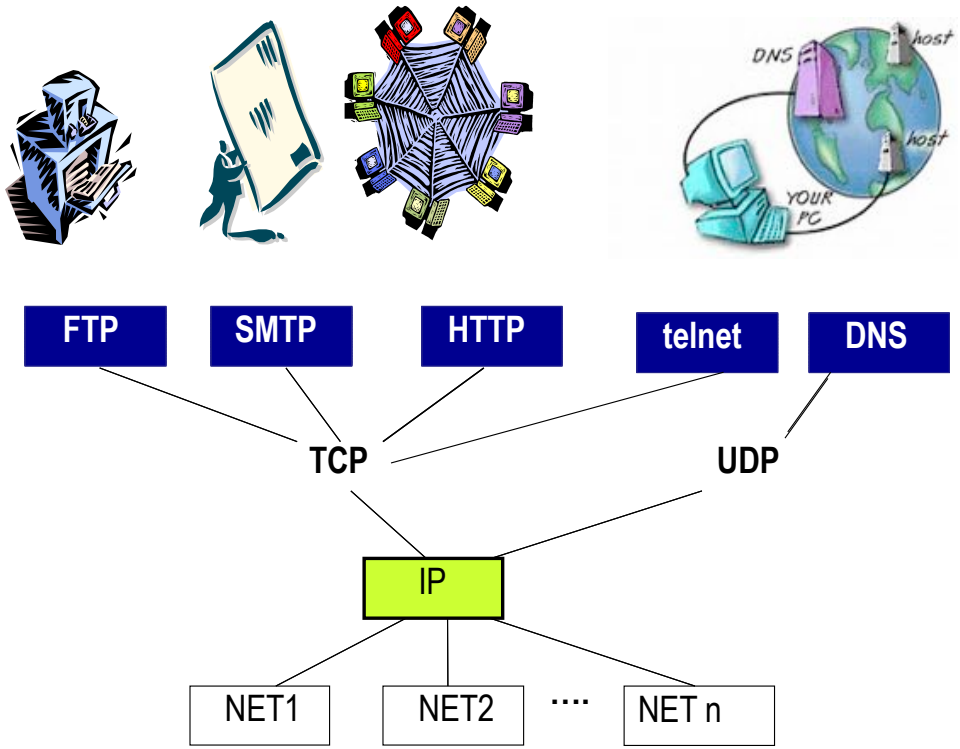


UDP service

- **unreliable** data transfer between sender and receive
- does not provide: connection setup, reliability, flow control, congestion control, timing, throughput guarantee, or security

Application-layer protocol defines

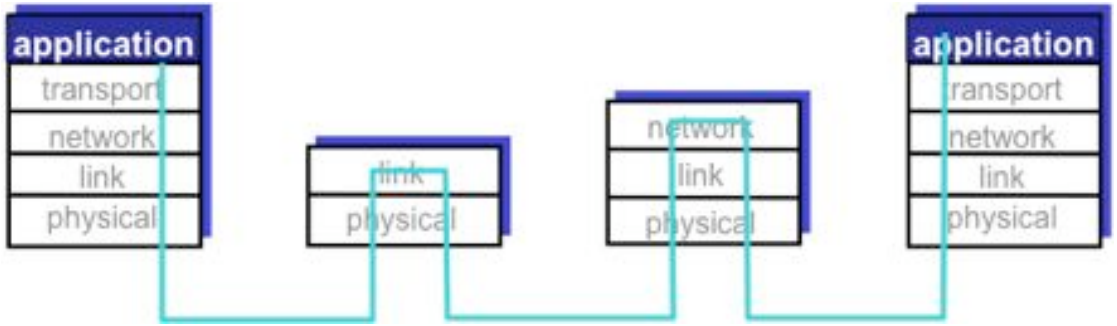
- **Types** of messages exchanged
  - e.g. request, response
- Message **syntax**
  - what fields in messages
  - how fields are delineated
- Message **semantics**
  - meaning of information in fields
- **Rules** for when and how processes send & respond to messages – actions



Application layer: Roadmap

- 2.1 Principles of network applications
- **2.2 Web and HTTP**
- ~~2.3 FTP~~
- 2.4 Electronic mail
  - SMTP, POP3, IMAP
  - 8.5.1 Secure email
- 2.5 DNS
- 2.6 P2P applications
- 2.7 Socket programming with TCP
- 2.8 Socket programming with UDP

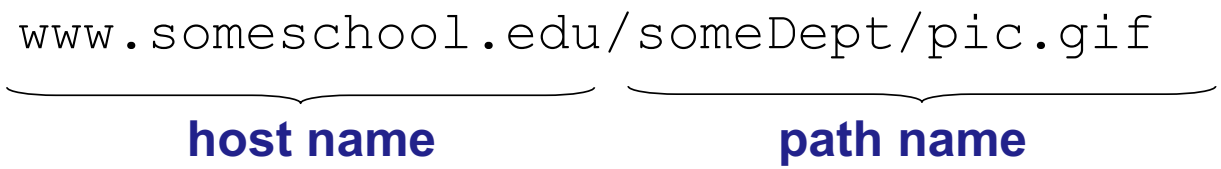
5-layer internet protocol stack



NTNU

Web pages consist of addressable objects

- Web page consists of a **base HTML-file** which includes several referenced objects
- **Object** can be HTML file, image, Java applet, audio file,…
- Each object is addressable by a **URL Uniform Resource Locator**



Sundby fikk lynkurs i smertehåndtering av Tufte

VAL DI FIEMME (NRK): Martin Johnsrud Sundby takker roer Olaf Tufte for at han klarte å holde ut smerten han måtte gjennom for å forsvare en ny Tour de Ski-seier.

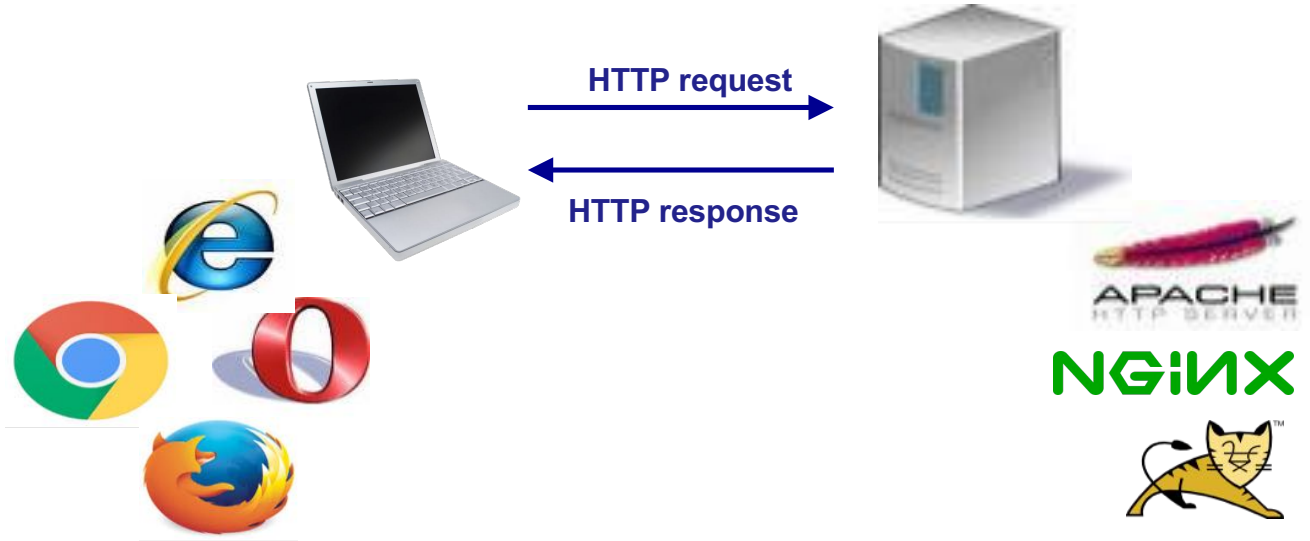
Journalist
Yasmin Sunde Hoel

TOI

Publ

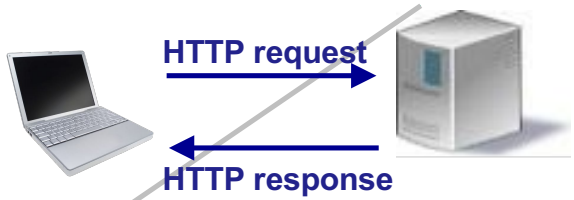
<!--[if lt IE 9]>
<script>
// IE8 and below: For discussion and comments, see: http://remysharp.com/2009/01/07/html5-enabling-script/
(function(){if(!/*@cc_on!@*/0)return;var e = "abbr,article,aside,audio,bb,canvas,datagrid,datalist,details,dialog,eventsource,figure,figcaption,footer,header,hgroup,mark,me
</script>
<![endif]-->
<title>Sundby fikk lynkurs i smertehåndtering av Tufte - Sport</title>
<meta name="description" content="VAL DI FIEMME (NRK): Martin Johnsrud Sundby takker roer Olaf Tufte for at han klarte å holde ut smerten han måtte gjennom for å forsvare en
<meta name="author" content="NRK"/>
<meta name="date" content="2019-01-11T14:47:48+01:00"/>
<link rel="apple-touch-icon-precomposed" sizes="114x114" href="/serum/latest/views-dist/skin/sport/img/apple-touch-icon-114x114.png" />
<link rel="apple-touch-icon-precomposed" sizes="72x72" href="/serum/latest/views-dist/skin/sport/img/apple-touch-icon-72x72.png" />
<link rel="apple-touch-icon-precomposed" href="/serum/latest/views-dist/skin/sport/img/apple-touch-icon.png" />
<link rel="shortcut icon" href="/serum/latest/views-dist/type/page/img/nrk.ico" />
<meta property="og:type" content="article" name="og:type" />
<meta property="og:title" content="Sundby fikk lynkurs i smertehåndtering av Tufte" name="og:title" />
<meta property="og:description" content="VAL DI FIEMME (NRK): Martin Johnsrud Sundby takker roer Olaf Tufte for at han klarte å holde ut smerten han måtte gjennom for å for
<meta property="og:image" content="http://gfx.nrk.no/dnQfl0hlrp426Pw_84FlQgnhYlF7a7okmnOy2MdSYIBQ" name="og:image" />
<meta property="og:site_name" content="NRK" name="og:site_name" />
<meta property="og:url" content="https://www.nrk.no/sport/sundby-fikk-lynkurs-i-smertehandtering-av-tufte-1.12144855" name="og:url" />
<meta name="twitter:card" content="summary" />
<meta name="twitter:site" content="@NRKno" />
<meta name="twitter:title" content="Sundby fikk lynkurs i smertehåndtering av Tufte" />
<meta name="twitter:description" content="VAL DI FIEMME (NRK): Martin Johnsrud Sundby takker roer Olaf Tufte for at han klarte å holde ut smerten han måtte gjennom for å fo
<meta name="twitter:image" content="http://gfx.nrk.no/dnQfl0hlrp426Pw_84FlQymnxKuaAlamnOy2MdSYIBQ" />
<meta name="Robots" content="index,follow" />
<meta name="Googlebot" content="index,follow" />
<meta name="Revisit-after" content="1 Days" />
<meta name="Generator" content="Publ
<link rel="Copyright" title="Opphavsrett NRK" href="http://www.nrk.no/emikk/1.2843522" />
<link rel="Help" title="Hjelp" href="http://www.nrk.no/hjelp/" />

The HTTP = hypertext transfer protocol for requesting objects from servers



- **Client = browser**
  requests, receives, “displays” web objects
- **Server = web server**
  sends objects in response to requests

HTTP is text based and has two message types



REQUEST:

Method URL version

```
GET /somedir/page.html HTTP/1.1\r\n
Host: www.someschool.edu\r\n
User-agent: Mozilla/4.0\r\n
Connection: close\r\n
Accept-language:fr\r\n
\r\n
data ved POST
```

header lines

carriage return character
line-feed character

(ekstra \r\n)

indicates end of message

RESPONSE:

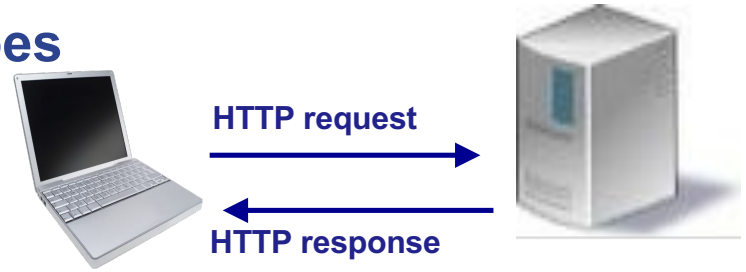
version
status code
status phrase

```
HTTP/1.1 200 OK
Connection: close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 …...
Content-Length: 6821
Content-Type: text/html
data data data data data ...
```

header lines

For more examples
http://gaia.cs.umass.edu/kurose_ross/interactive/

HTTP is text based and has two message types



REQUEST:

Method URL version

```
GET /index.html HTTP/1.1\r\n
Host: www-net.cs.umass.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
data ved POST
(ekstra \r\n)
```

header lines

carriage return character
line-feed character

indicates end of message

RESPONSE:

version
status code
status phrase

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02 GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-1\r\n
\r\n
data data data data data ...
```

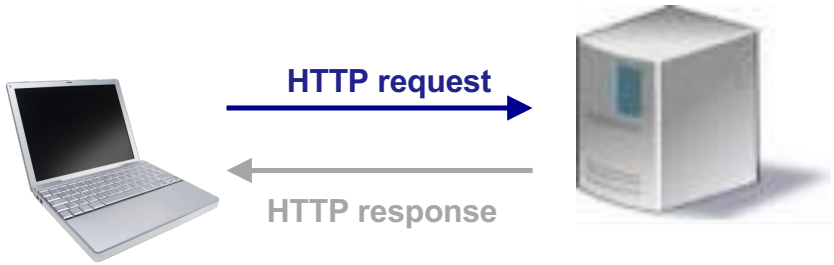
header lines

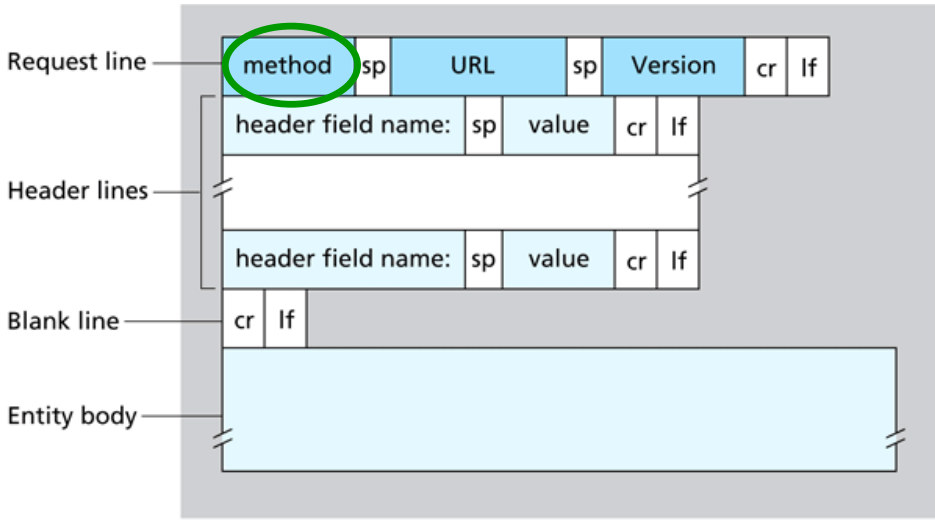
For more examples
http://gaia.cs.umass.edu/kurose_ross/interactive/

NTNU

HTTP request message

Method types





sp = space; cr=carrige return; lf=line feed

HTTP/1.0

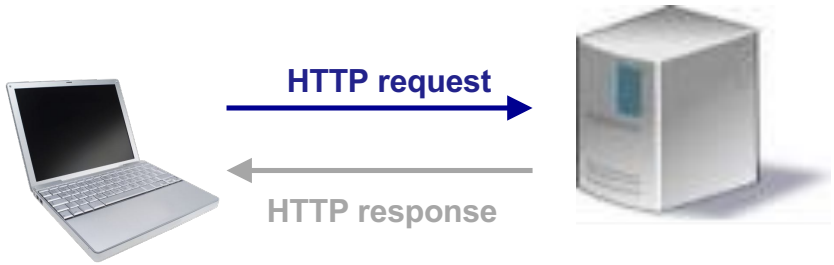
- **GET**
  - request object
- **POST**
  - request object dependent on content
- **HEAD**
  - ask server to leave requested object out of response

HTTP/1.1

- **GET**, **POST**, **HEAD**
- **PUT**
  - upload file in entity body to path specified in URL field
- **DELETE**
  - delete file specified in URL field

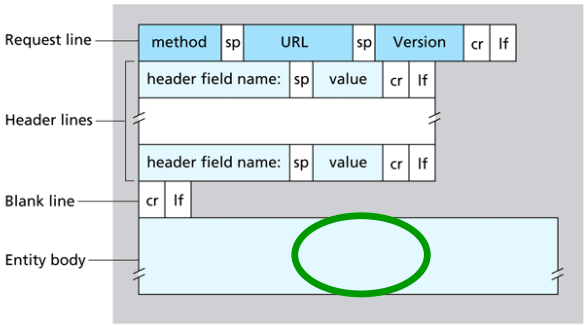
HTTP request message

When web page includes form input



POST method:

- Input is uploaded to server in entity body

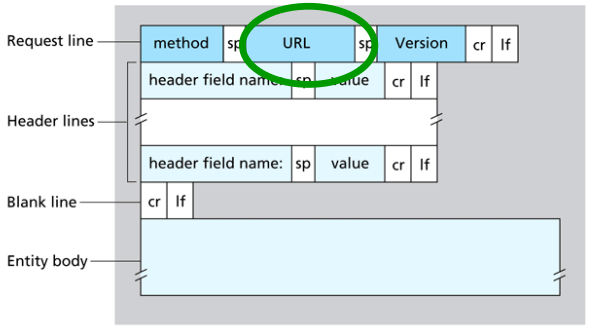


sp = space; cr=carrige return; lf=line feed

URL method:

- Uses GET method
- Input is uploaded in URL field of request line

```
www.somesite.com/animalsearch?monkeys&banana
```



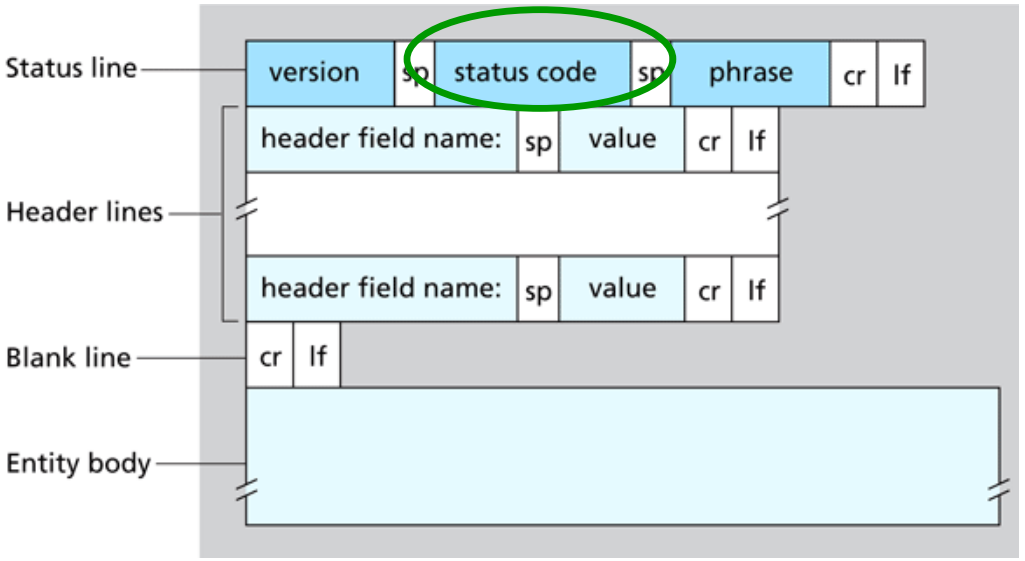
HTTP request message

HTTP POST message

```
POST /gp/product/sessionCacheUpdateHandler.html HTTP/1.1
Host: www.amazon.co.uk
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:26.0) Gecko/20100101 Firefox/26.0
Accept: */*
Accept-Language: nb,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Referer: http://www.amazon.co.uk/
Content-Length: 132
Cookie: x-wl-uid=1QnptmyS6pWLKJYqcs7DoCfJaZiW/S9IS+o7r32IkWPqE20iV1HYygYP29mjsgg
+1yNIIZSVq7Xlvkxz6YnBCTJFhUKoPNJUoc599fRtzoOREgsdwITCywrHbEQTuc99HjMWmC+Ss+JU=; session-id-
time=20827584011; session-id=275-4621766-1475452; ubid-acbuk=278-8688747-5031653; session-token=wZnz3Ov
+UIzZFtmYyaNxlKtR2EBk5Ru/s69C3LWygkywAWudCHSsKL4I1tAlXWAndHnOzKG+jz5SeGIJb6hO+IGosWDYtq5yGeu9gsnK/
LjHMkvirHQ9vnXkT3oiqV2cbtVTLN3nVC4pQ6qT26aGAWClglOPx/yp54jhONA6Sj4bafC+2695F/IXnlOFoIPba2kDMCswBi4xA2N
+pNoilbSmx4+37eCkVSw/iGzPQOeO2hSo7tiyFvtF2jv4lqehr/ICcnGilUU/3lzFllVG4JhCD2405zkN; csm-hit=203.99|
1389385731820; x-acbuk="ys?cyWEwPtgnPFgdy4TrpoODB83CI5@O55GhUxFOxlEf@h3IEfW8900ifbdVPmDP"
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache

sessionCacheUpdateFlag=1&pageType=Gateway&browserWidth=1251&browserHeight=805&token=fVoboEDsYMjWCjzTNBs
COM6p%2FFu4N[illegible]3DHTTP/1.1 200 OK
Date: Fri, 10 Jan 2014 20:28:52 GMT
Server: Server
x-amz-id-1: 1Q5FRCBSRTMSAA17HRY7
p3p: policyref="http://www.amazon.co.uk/w3c/p3p.xml",CP="CAO DSP LAW CUR ADM IVAo IVDo CONo OTPo OUR
DELi PUBi OTRi BUS PHY ONL UNI PUR FIN COM NAV INT DEM CNT STA HEA PRE LOC GOV OTC "
x-amz-id-2: 8tBw/nkrKcINjFZYSFOObH3ZXVh/8qIBsa6AWiyOOA3/yHzdm2oOWEzM5azkcs8P
Vary: Accept-Encoding,User-Agent
Content-Encoding: gzip
Content-Type: text/html; charset=ISO-8859-1
Set-cookie: ubid-acbuk=278-8688747-5031653; path=/; domain=.amazon.co.uk; expires=Tue, 01-Jan-2036
00:00:01 GMT
Set-cookie: session-id-time=20827584011; path=/; domain=.amazon.co.uk; expires=Tue, 01-Jan-2036
00:00:01 GMT
Set-cookie: session-id=275-4621766-1475452; path=/; domain=.amazon.co.uk; expires=Tue, 01-Jan-2036
00:00:01 GMT
Transfer-Encoding: chunked
```

HTTP response message

Status codes



- **`200 OK`**
  - request succeeded, requested object later in this message
- **`301 Moved Permanently`**
  - requested object moved, new location specified later in this message (`Location:`)
- **`400 Bad Request`**
  - request message not understood by server

`404 Not Found`

requested document not found on this server

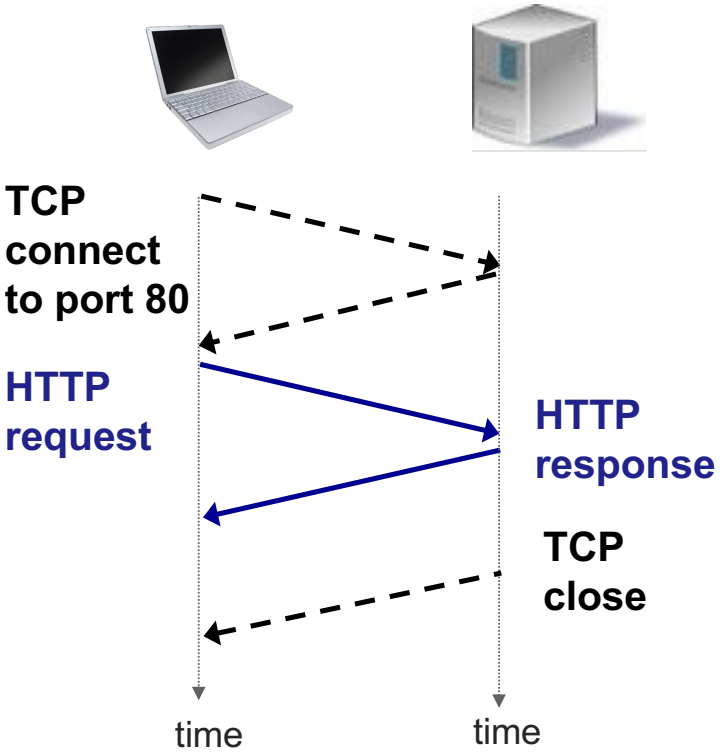
`505 HTTP Version Not Supported`

HTTP uses TCP for transport of application-layer messages

- **Non-persistent HTTP**
  - At most one object per TCP connection
  - HTTP 1.0
- **Persistent HTTP**
  - Multiple objects per TCP connection
  - HTTP 1.1

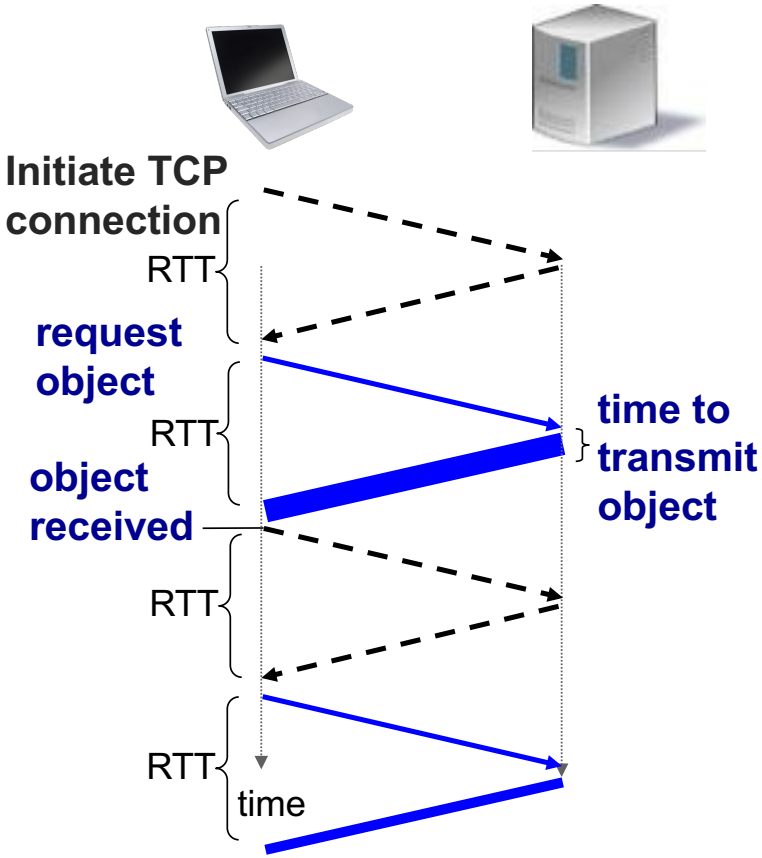
Performance challenges with HTTP 1.0?

➪ Giving different round-trip-times (RTTs) per object



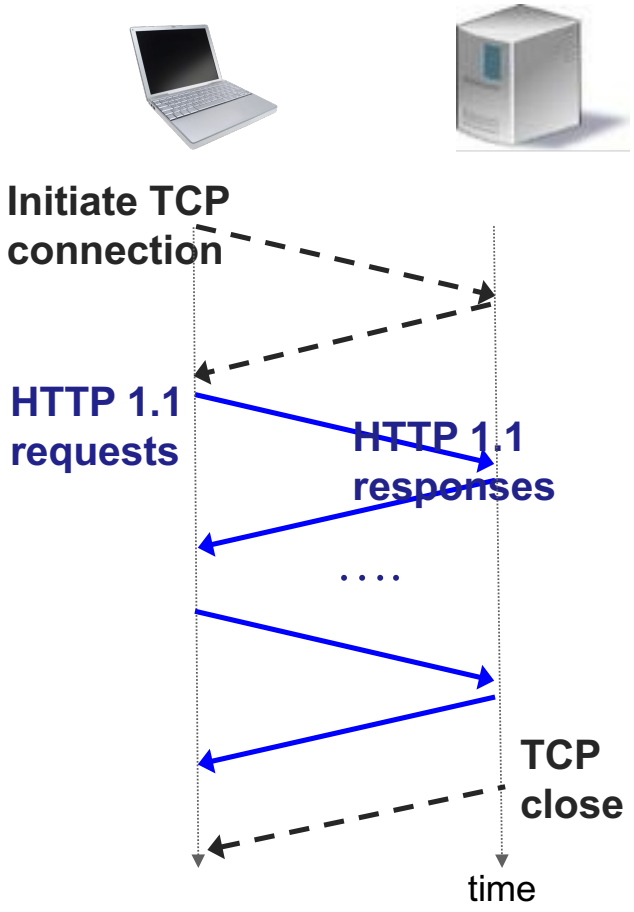
Non-persistent HTTP transfers one object per connection

- RTT to initiate TCP connection
- RTT for HTTP request and first few bytes of HTTP response to return
- Object transmission time
- Non-persistent HTTP response time per object = 2RTT+transmit time
- Operating system overhead for each TCP connection
- Browsers often open **parallel TCP connections** to fetch referenced objects



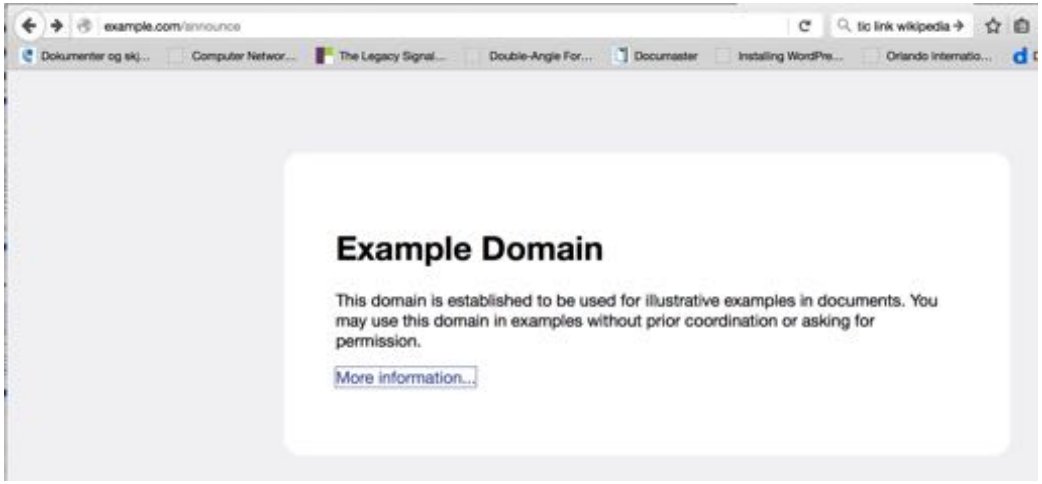
Persistent HTTP transfers several objects per connection

- Subsequent HTTP messages between same client/server sent over one TCP connection
- Client sends requests as soon as it encounters a referenced object
- Reduces delay to fetch a web page: *One* RTT for *all* the referenced objects!



Manual HTTP over telnet 80

telnet example.com 80
Trying 93.184.216.34...
Connected to example.com.
Escape character is '^]'.
get / HTTP/1.1
Host: example.com



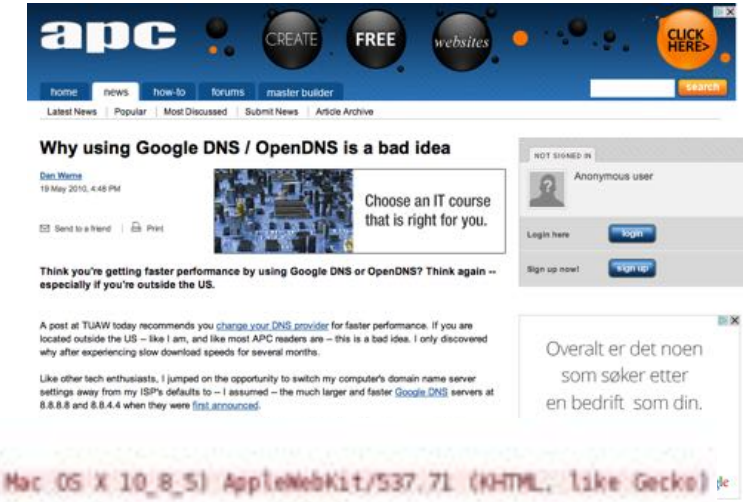
HTTP/1.1 501 Not Implemented
Content-Type: text/html
Content-Length: 357
Connection: close
Date: Sun, 17 Jan 2016 18:51:55 GMT
Server: ECSF (lga/1389)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
            <head>
                <title>501 - Not Implemented</title>
            </head>
            <body>
                <h1>501 - Not Implemented</h1>
            </body>
</html>
```

Connection closed by foreign host.

HTTP request and response message


```
GET /CMSAdmin/CMSScripts.js HTTP/1.1
Host: apcmag.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_5) AppleWebKit/537.71 (KHTML, like Gecko)
Version/6.1 Safari/537.71
Accept: */*
Referer: http://apcmag.com/why-using-google-dns-opendns-is-a-bad-idea.htm
Accept-Language: nb-no
Accept-Encoding: gzip, deflate
Cookie: ASP.NET_SessionId=thxpeoj5nifinjzlhvgmylmu; Mode=
Connection: keep-alive

HTTP/1.1 200 OK
Content-Type: application/x-javascript
Content-Encoding: gzip
Last-Modified: Mon, 19 Nov 2012 22:37:43 GMT
Accept-Ranges: bytes
ETag: "8015247da6c6cd1:0"
Vary: Accept-Encoding
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Mon, 06 Jan 2014 17:03:36 GMT
Content-Length: 2730
```



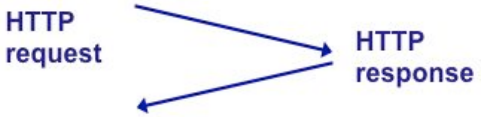

```
GET /tag/js/gpt.js HTTP/1.1
Host: www.googletagservices.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_5) AppleWebKit/537.71 (KHTML, like Gecko)
Version/6.1 Safari/537.71
Accept: */*
Referer: http://apcmag.com/why-using-google-dns-opendns-is-a-bad-idea.htm
If-None-Match: 5676676589794080235
Accept-Language: nb-no
Accept-Encoding: gzip, deflate
Connection: keep-alive

HTTP/1.1 304 Not Modified
Date: Fri, 10 Jan 2014 19:15:24 GMT
Expires: Fri, 10 Jan 2014 20:15:24 GMT
ETag: 5676676589794080235
Age: 3061
Server: GFE/2.0
Alternate-Protocol: 80:quic
```

COOKIES are used to store user state, as HTTP is stateless

- **Cookies** are simple data, and can bring
  - Authorization
  - Shopping carts
  - Recommendations
  - User session state (Web e-mail)

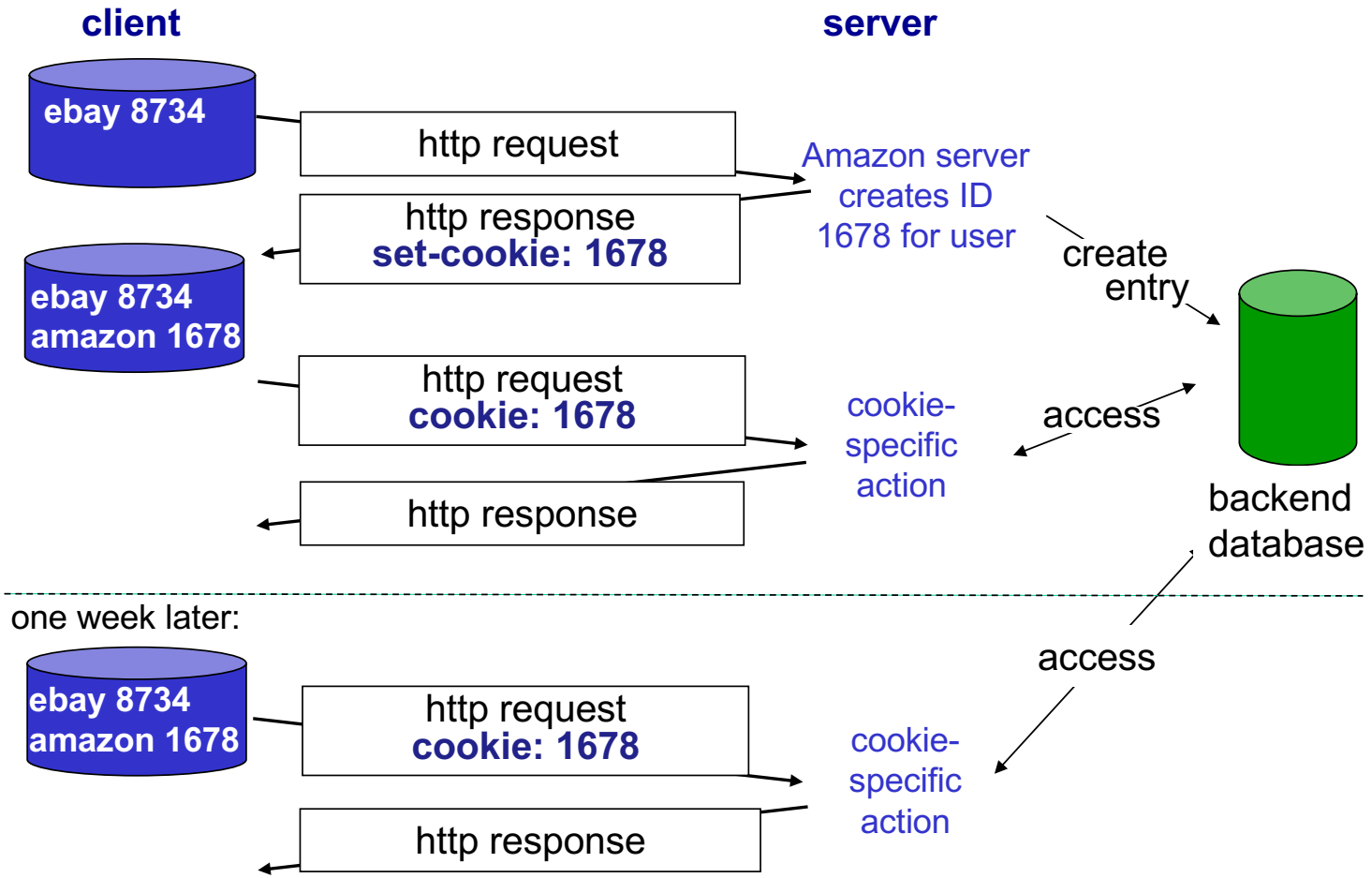
Cookies and privacy:
- cookies permit sites to learn a lot about you
- you may supply name and e-mail to sites

- Four cookie components
  1. **cookie header line** of HTTP **response** message
  2. **cookie header line** in HTTP **request** message
  3. **cookie file** kept on **user's host**, managed by user's browser
  4. back-end **database** at Web site



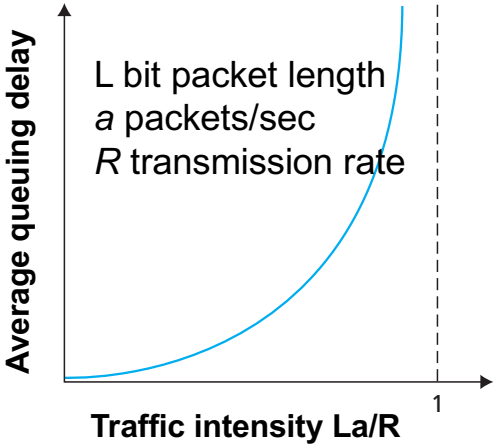
How to keep "state":
1. protocol endpoints maintain state at sender/receiver over multiple transactions
2. **cookies**: http messages carry state

Cookies keep “state” on user's host

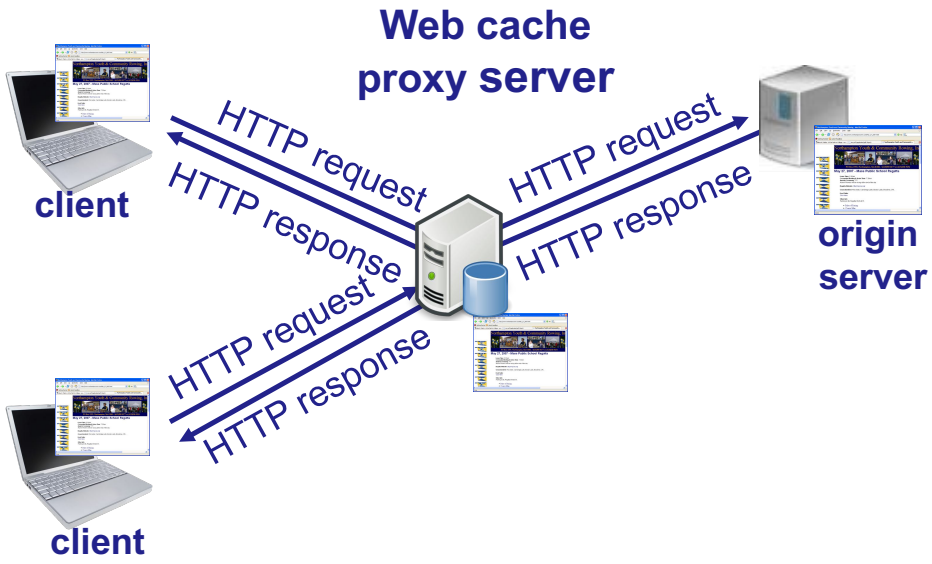


WEB CACHING
reduces HTTP response time for client request

- Why? Performance, cost
- Users set browser web access via cache
- Client requests satisfied without involving origin server

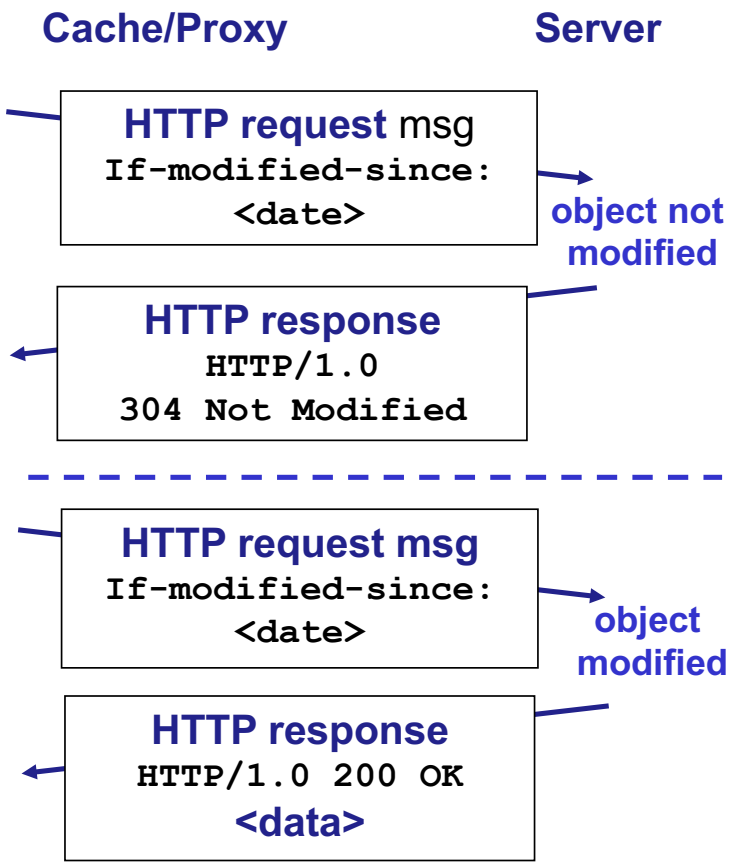


- Browser sends all HTTP requests to cache
  - object not in cache
    cache requests object
    from origin server,
    then returns object to client
  - else object in cache:
    cache returns object



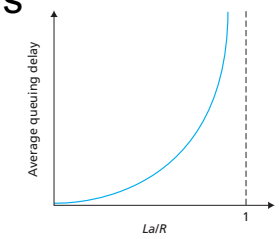
Conditional GET: don't send object if cache has up-to-date cached version

- **Cache/Proxy**:
  specify date of cached
  copy in HTTP request
  - `If-modified-since: <date>`
- **Original server**:
  response contains
  no object if cached copy
  is up-to-date:
  - `HTTP/1.0 304 Not Modified`

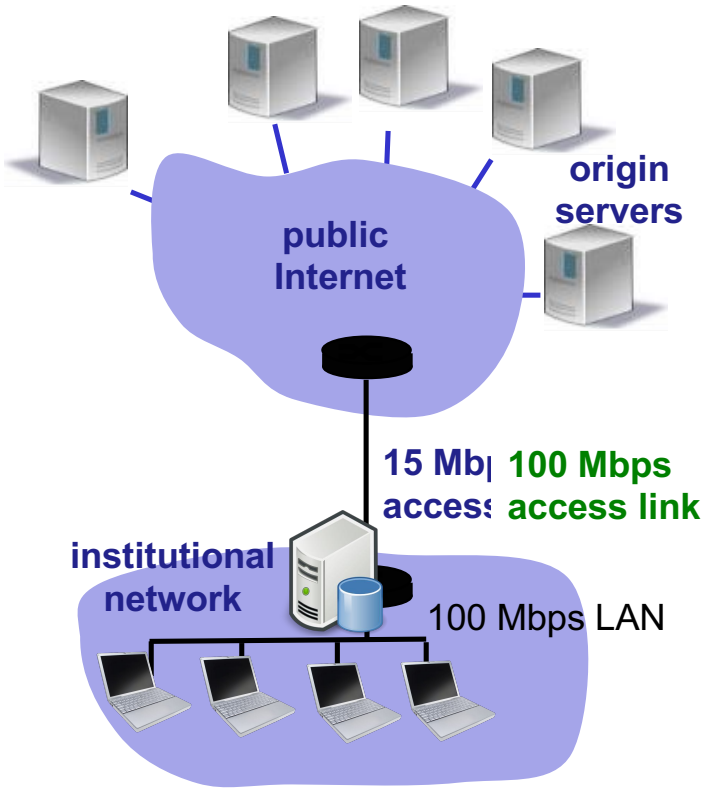


Caching example

- Assumptions
  - 15 requests per seconds of 1 Mbit objects
  - Internet delay (round-trip) from institutional router to any origin server = 2 s
  - Access link delay depends on traffic intensity (eg. < 0,01 s when < 60% loaded)
  - LAN delay ms



- **15 Mbps access link**
  - utilization LAN = 15%
  - utilization access link = 100%
  - total average delay = Internet delay + access link delay + LAN delay = 2 s + upto 30 s+ ms
- **100 Mbps access link**
  - total average delay = 2 s + 0,01 s + ms



- **Cache with hit rate 0.4** (within 10 ms)
  - Utilization of access link reduced to 60%, resulting in negligible delays (say 0,01 s)
  - Total average delay = .4 * 0,001 s + .6 * (2 s + 0,01 s + ms)

Securing TCP at the application layer

- TCP & UDP
  - no encryption: clear-text password sent into socket traverse the Internet in clear text
- **Secure Socket Layer (SSL)**
  - provides encrypted TCP connection
  - data integrity
  - end-point authentication

SSL technically resides in the application layer, from the developer's perspective it is a transport-layer protocol.
We will do 8.6 as part of transport layer

- SSL is at application layer
  - applications use SSL libraries, which "talk" to TCP
- SSL socket API
  - clear-text passwords sent into socket traverse Internet encrypted

Application layer: Roadmap

2.1 Principles of network applications

2.2 Web and HTTP

2.3 FTP

2.4 Electronic mail

- **SMTP, POP3, IMAP**
- **8.2-3 + 8.5.1 Secure email**

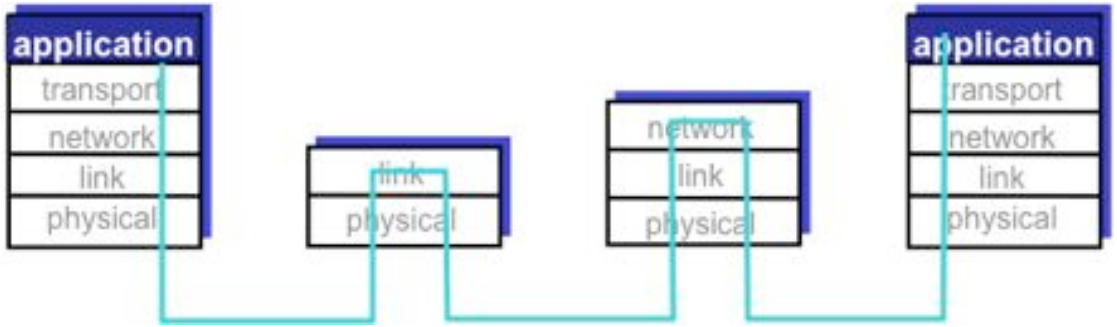
2.5 DNS

2.6 P2P applications

2.7 Socket programming with TCP

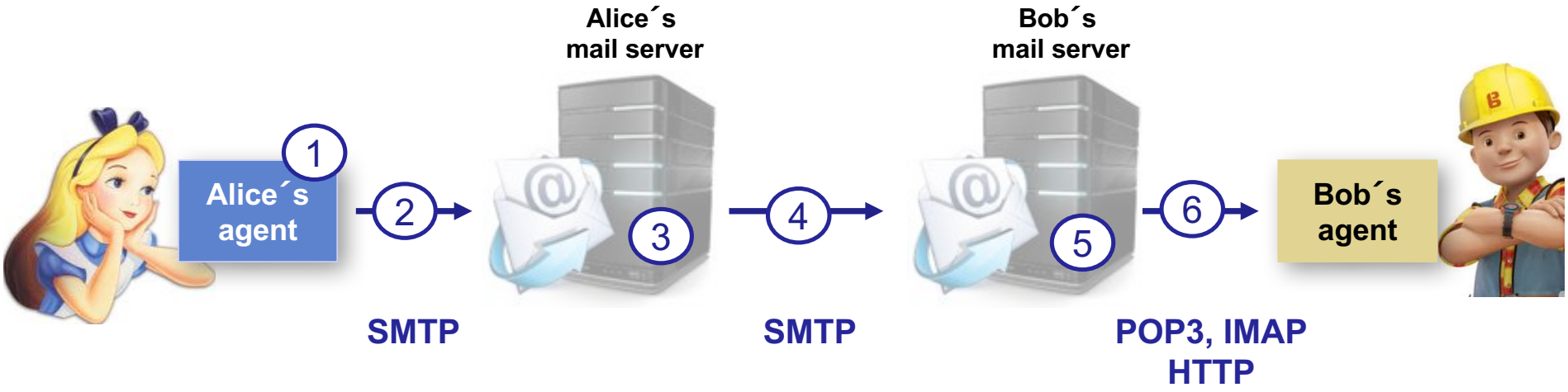
2.8 Socket programming with UDP

5-layer internet protocol stack



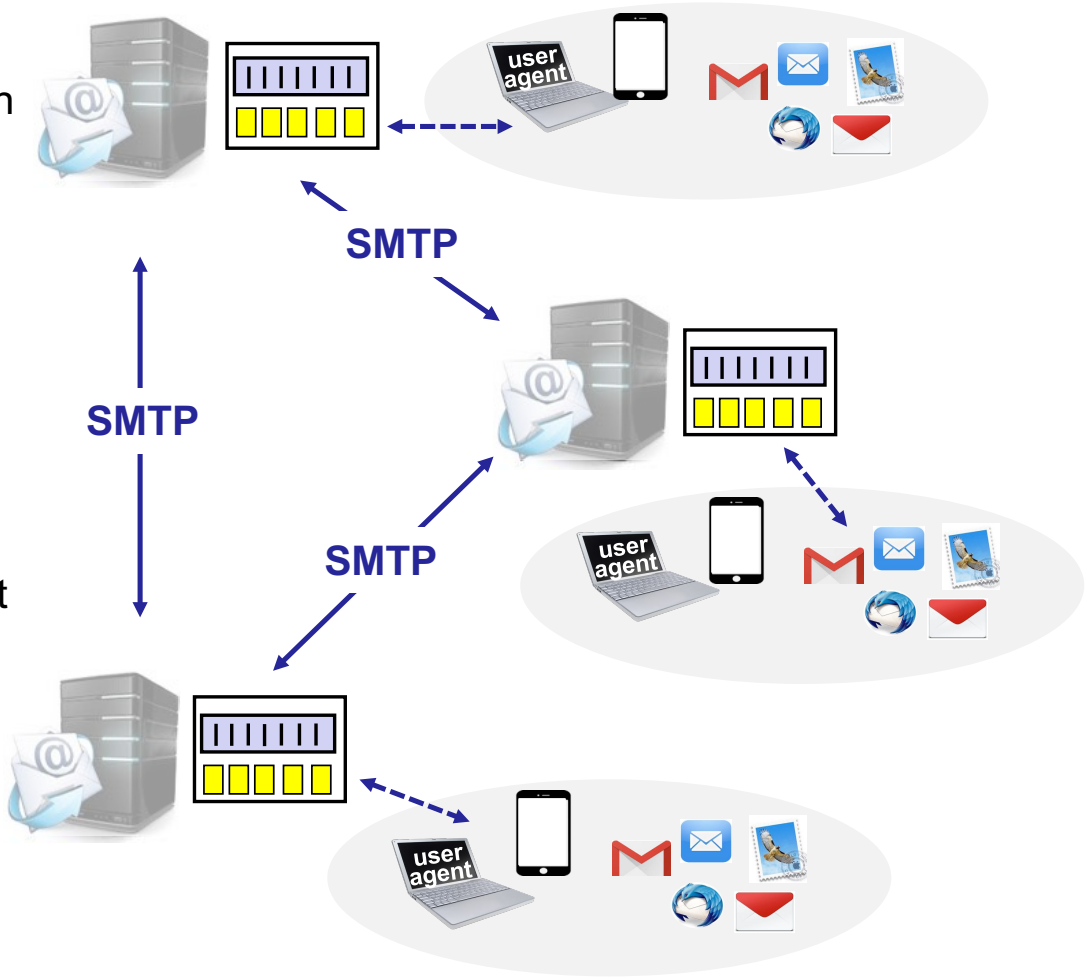
Scenario: Alice sends message to Bob

1. Alice compose message to bob@someschool.edu
2. Alice's UA sends message to her mail server; message placed in message queue
3. Client side of SMTP opens TCP connection with Bob's mail server
4. SMTP client sends Alice's message over the TCP connection
5. Bob's mail server places the message in Bob's mailbox
6. Bob invokes his user agent to read message



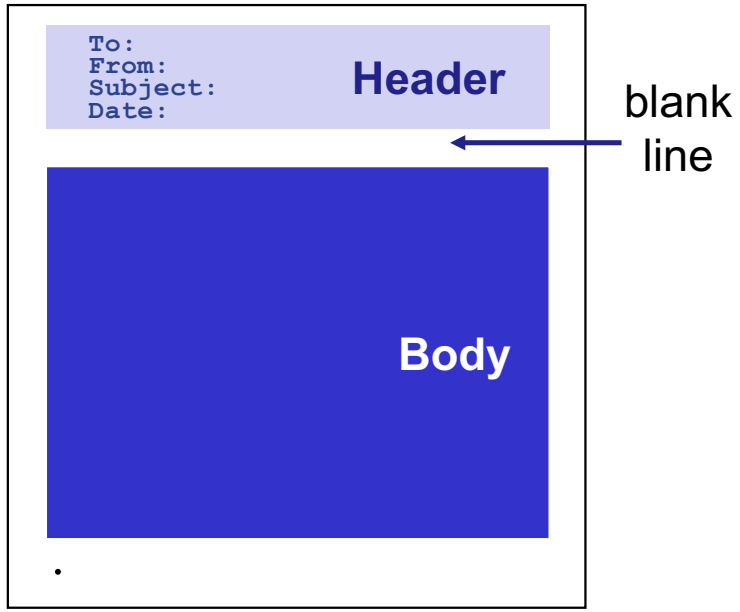
SMTP transfers mail, user agents compose, edit, send, receives and read mail messages

- SMTP (**Simple Mail Transfer Protocol**) between mail servers and user agent to server
- Incoming and outgoing messages stored on server
  - Queue of outgoing (to be sent) mail messages
  - User mailbox for incoming messages
- SMTP: port 25, port 465 SSL/TLS encrypted
- To retrieve messages from server
  - **POP**: Post Office Protocol: authorization (agent <--> server) and download
  - **IMAP**: Internet Mail Access Protocol
    - more features, e.g. manipulation of stored messages on server
  - **HTTP**: gmail, outlook.com, Yahoo! Mail etc.



Mail data message format: header + body

- Mail message header lines **different from SMTP commands!**
  E.g.
  - `To:`
  - `From:`
  - `Subject:`
  - `Date:`
- Mail message body
  - the “message”
  - 7-bit ASCII characters only
- SMTP server uses `CRLF.CRLF` to determine end of message



Manual SMTP over telnet port 25

kjersti — bash — 80×27

```
221 2.7.0 Error: I can break rules, too. Goodbye.
Connection closed by foreign host.
KM:~ kjersti$ telnet samson.item.ntnu.no 25
Trying 129.241.200.24...
Connected to samson.item.ntnu.no.
Escape character is '^]'.
220 samson.item.ntnu.no ESMTP Postfix (Ubuntu)
HELO pop.spacecentre.no
250 samson.item.ntnu.no
MAIL FROM: kjersti@spacecentre.no
250 2.1.0 Ok
RCPT TO: kjmoldek@item.ntnu.no
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
TO: Meg
FROM: Deg
Subject: TEST
Date: Wed, 20 Jan 2016 14:39 +0100
Dette er en SMTP test
HAdet paa badet
.
250 2.0.0 Ok: queued as BFD57480220
quit
221 2.0.0 Bye
Connection closed by foreign host.
KM:~ kjersti$
```

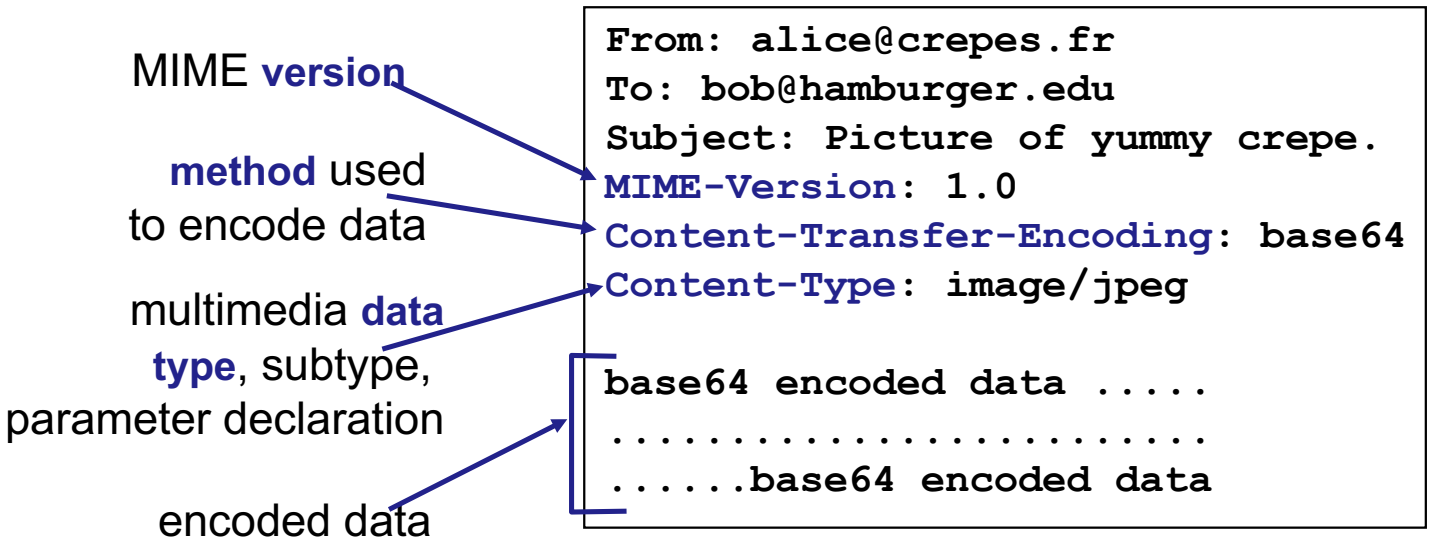
2016-01-20 SMTP samson 25-med date.pcapng

smtp

No.	Time	Source	Destination	Protocol	Length	Info
19	2.993770	129.241.200.24	172.16.3.196	SMTP	114	S: 220 samson.item.ntnu.no ESMTP Postfix (Ubuntu)
87	12.1286…	172.16.3.196	129.241.200.24	SMTP	91	C: HELO pop.spacecentre.no
89	12.1368…	129.241.200.24	172.16.3.196	SMTP	91	S: 250 samson.item.ntnu.no
130	20.9602…	172.16.3.196	129.241.200.24	SMTP	101	C: MAIL FROM: kjersti@spacecentre.no
131	20.9692…	129.241.200.24	172.16.3.196	SMTP	80	S: 250 2.1.0 Ok
159	29.3435…	172.16.3.196	129.241.200.24	SMTP	98	C: RCPT TO: kjmoldek@item.ntnu.no
160	29.3541…	129.241.200.24	172.16.3.196	SMTP	80	S: 250 2.1.5 Ok
166	31.6476…	172.16.3.196	129.241.200.24	SMTP	72	C: DATA
167	31.6559…	129.241.200.24	172.16.3.196	SMTP	103	S: 354 End data with <CR><LF>.<CR><LF>
179	36.1507…	172.16.3.196	129.241.200.24	SMTP	75	C: DATA fragment, 9 bytes
197	38.7666…	172.16.3.196	129.241.200.24	SMTP	77	C: DATA fragment, 11 bytes
209	44.2307…	172.16.3.196	129.241.200.24	SMTP	81	C: DATA fragment, 15 bytes
304	65.2458…	172.16.3.196	129.241.200.24	SMTP	102	C: DATA fragment, 36 bytes
323	70.4775…	172.16.3.196	129.241.200.24	SMTP	89	C: DATA fragment, 23 bytes
339	75.7172…	172.16.3.196	129.241.200.24	SMTP	83	C: DATA fragment, 17 bytes
350	76.6932…	172.16.3.196	129.241.200.24	IMF	69	from: Deg, subject: TEST, , TO: Meg , FROM: Deg ,
352	76.7031…	129.241.200.24	172.16.3.196	SMTP	103	S: 250 2.0.0 Ok: queued as BFD57480220
445	79.3725…	172.16.3.196	129.241.200.24	SMTP	72	C: quit
446	79.3813…	129.241.200.24	172.16.3.196	SMTP	81	S: 221 2.0.0 Bye

MIME - multipurpose mail extension

- Additional lines in message header declare MIME content types – multi media and message bodies with multiple parts



SMTP & HTTP: ASCII (text) command/response interaction, status codes

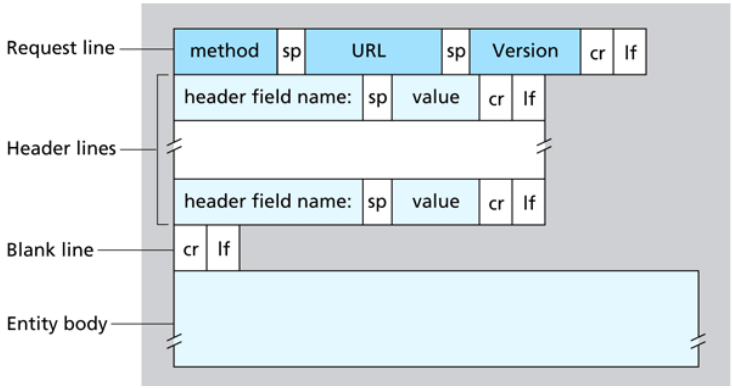
SMTP

- SMTP: **push**
- persistent connections
- multiple objects sent in multipart msg
- SMTP server uses `CRLF.CRLF` to determine end of message

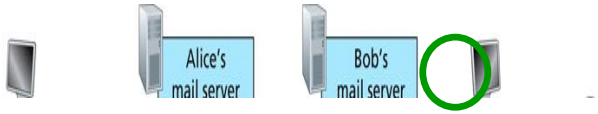
```
HELO edda
250 samson.item.ntnu.no
MAIL FROM: kjmoldek@online.no
250 2.1.0 Ok
DATA
554 5.5.1 Error: no valid recipients
RCPT TO: kjmoldek@item.ntnu.no
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
```

HTTP

- HTTP: **pull**
- (non) persistent connections
- each object encapsulated in its own response msg
- End-of HTTP request: **extra CRLF**
- End-of HTTP response: **Content-Length**



Internet mail access protocols transfer mail from mail server to user client



POP3 (Post office protocol)

- Limited functionality
- Authorization, transaction, update
- "Download-and-delete"
- "Download-and-keep": copies of messages on different clients
- **Stateless** across sessions

- Port 110, port 995 for POP3-over-SSL

IMAP (Internet mail access protocol)

- More features, more complex
- IMAP keeps **messages on server in folders**
- IMAP keeps user **state across sessions**:
  - names of folders and mappings between message IDs and folder name

- Port 143, port 993 for IMAP-over-SSL

POP3 protocol: download-and-delete

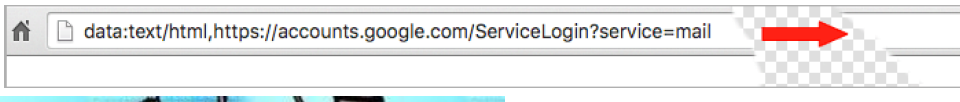
1. **Authorization phase,** client
   - `user` <username>
   - `pass` <password>
   - server responses
     - `+OK`
     - `-ERR`
2. **Transaction phase,** client
   - `list`: list message numbers
   - `retr`: retrieve message by number
   - `dele`: delete
   - `quit`
3. **Update phase**

```
S: +OK POP3 server ready
C: user bob
S: +OK
C: pass hungry
S: +OK user successfully logged on
```

S: Server
C: Client

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

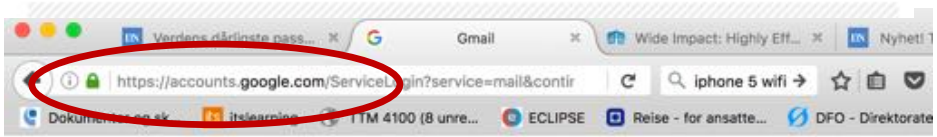
Email scam and spam



NEVER CONFIRM YOUR PASSWORD IN AN E-MAIL OR THROUGH A LINK IN AN E-MAIL

Wide Impact: Highly Effective Gmail Phishing Technique Being Exploited

This entry was posted in General Security, Miscellaneous on January 12, 2017 by Mark Maunder 135 Replies



Google

Én konto, for alt fra Google.

Logg på for å fortsette til Gmail

Betaling mottatt 2.800 NOK – Innboks

Skatteetaten
Til: Kjersti Moldeklev
Content-Type: text/html; charset="iso-8859-1"
X-Spam-Level: **
Mime-Version: 1.0
X-Spam-Status: No, score=2.991 tagged_above=2 required=20 tests=[BAYES_00=-1.9, HTML_MESSAGE=0.001, MIME_HTML_ONLY=0.723, MISSING_MID=0.497, RCVD_IN_BRBL_LASTEXT=1.4
RDNS_NONE=0.793, SPF_SOFTFAIL=0.665, TO_NO_BRKTS_NORDNS_HTML=0.763] autolearn=no autolearn_force=no
X-Spam-Score: 2.991
Return-Path: <skatteetaten@skatt.no>
X-Mailer: Mail::Sendmail version 0.79
X-Spam-Flag: NO
Content-Transfer-Encoding: quoted-printable
Received: from localhost (localhost [127.0.0.1]) by pop.spacecentre.no (Postfix) with ESMTP id 378841569FBD6 for <kjersti.moldeklev@spacecentre.no>; Tue, 30 Aug 2016 17:37:32 +0200 (CE
Received: from pop.spacecentre.no ([127.0.0.1]) by localhost (pop.spacecentre.no [127.0.0.1]) (amavisd-new, port 10024) with ESMTP id j_M0mVyt2xWm for <kjersti.moldeklev@spacecentre.no
30 Aug 2016 17:37:30 +0200 (CEST)
Received: from zbang.com (unknown [138.91.57.162]) by pop.spacecentre.no (Postfix) with SMTP id A3C1D156FBD0 for <kjersti.moldeklev@spacecentre.no>; Tue, 30 Aug 2016 17:37:30 +020
(CEST)
Delivered-To: kjersti@pop.spacecentre.no
Betaling mottatt 2.800 NOK

Kjære kunde,

Vi har funnet ut at du er berettiget til å motta en skatterefusjon 2.800 NOK.
Klikk her for å motta.

Cyberangrep mot Norge øker sterkt

I fjor avslørte norske myndigheter mer enn 22 000 dataangrep mot norske bedrifter og offentlige etater. Urovekkende økning, sier Nasjonal sikkerhetsmyndighet.

OVERVÅKER: Risikoen for at kritiske funksjoner i landet blir rammet av spionasje, sabotasje, terror og andre alvorlige handlinger er økende. Dette er fra rommet hvor Nasjonal sikkerhetsmyndighet avslører angrepene.
FOTO: SIRI VÅLBERG SAUGSTAD/NRK

Signe Karin Hotvedt
Bøgrekarm
Journalist

Siri Vålberg Saugstad
Videofotograf

MER OM DATASIKKERHET
MER OM NORGE
MER OM DATASIKKERHET I NORGE

Publisert i går, kl. 23:56

Dette kan DU gjøre for å hindre cyberangrep

Hver enkelt datamaskin inngår i et stort nettverk, og du kan bidra til datasikkerhet, også for andre, ved å:

- Hold din egen datamaskinen oppdatert
- Vær varsom med å klikke på lenker
- Vær varsom med å åpne vedlegg
- Bruk unike passord som du lagrer et trygt sted
- Bruk to-faktorautentisering der det er tilgjengelig
- Ikke videresend svindel-epost til vennene dine

Dette kan sikre at din datamaskin ikke blir misbrukt i angrep mot norske myndigheter eller interesser.

Kilde: NorCERT

Securing applications – What is network/information security?

Confidentiality: Information kept private and secure between sender and receivers

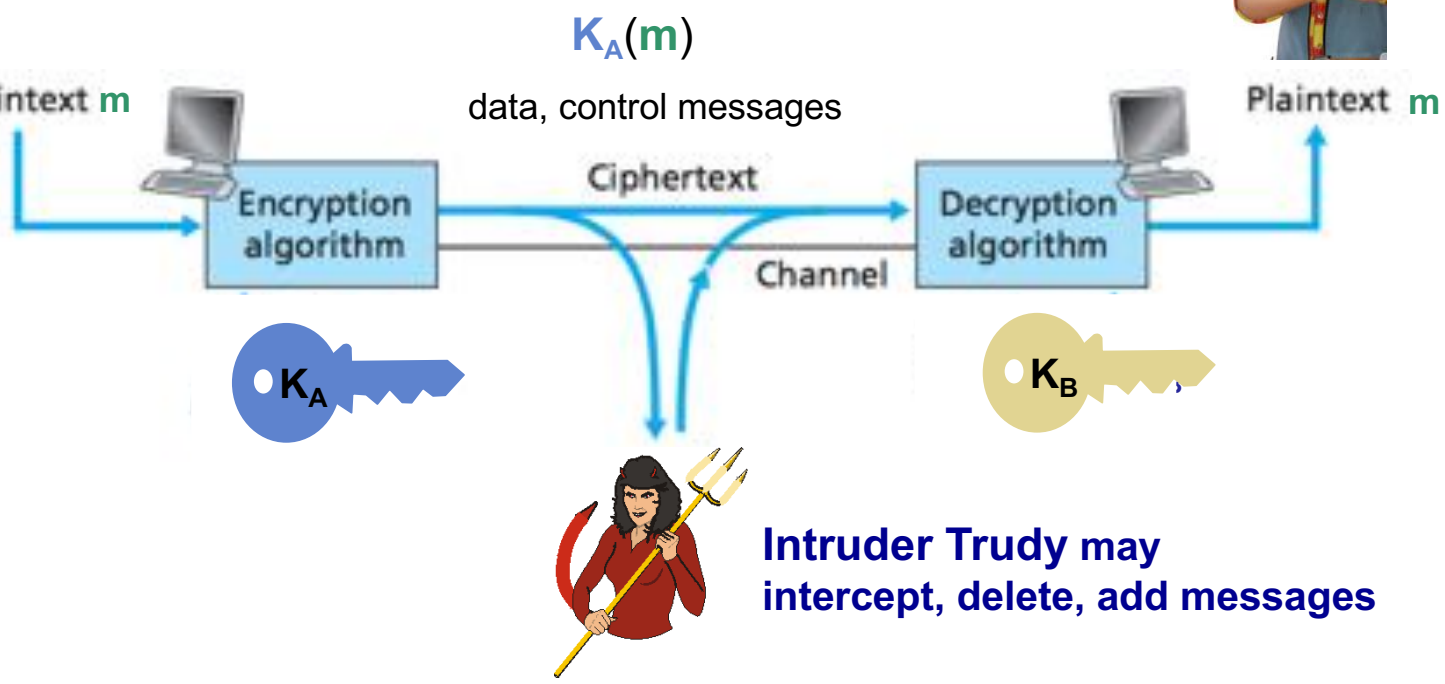
Authentication: sender and receiver want to confirm identity of each other

Availability: services must be accessible and available to whom requires them

Message integrity: sender and receiver want to ensure message not altered (modified, deleted or added) in transit, or afterwards without detection

Friends and enemies: Applications need secure communication

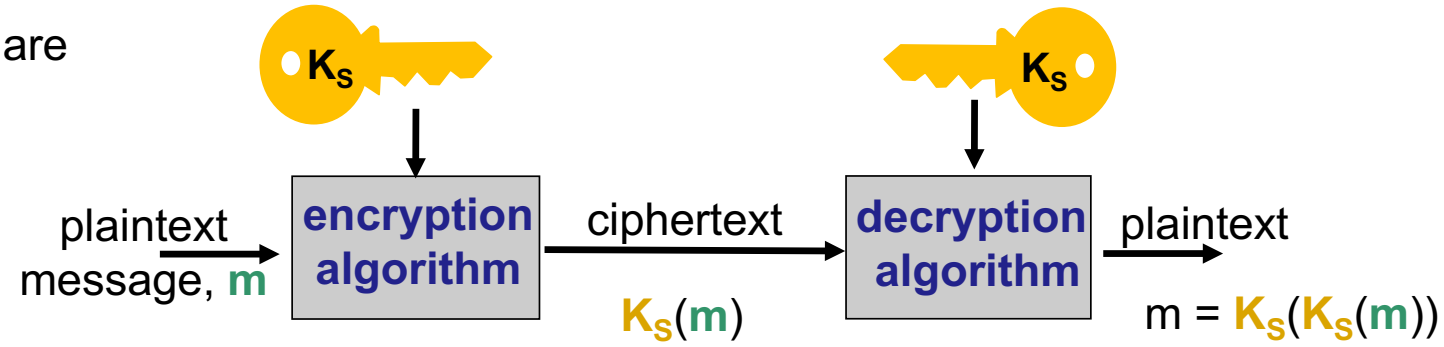
- **m** plaintext message
- $K_A(m)$ ciphertext, encrypted with key Alice´s key K_A
- Bob decrypts with key K_B, $m = K_B(K_A(m))$
- Fundamental problem: **key distribution**



Intruder Trudy may intercept, delete, add messages

Symmetric key cryptography is based on a common secret key

Bob and Alice share same key: **K**



Mono alphabetic substitution cipher

Plaintext letter: `abcdefghijklmnopqrstuvwxyz`
Ciphertext letter: `mnbvcxzasdfghjklpoiuytrewq`

K_S

plaintext: `bob. i love you. alice`
ciphertext: `nkn. s gktc wky. mgsbc`

Polyalphabetic substitution cipher

n substitution ciphers, **M1,M2,…,Mn** and **cycling pattern**: n=4: **M1,M3,M4,M3,M2**; repeating

For each new plaintext symbol, use subsequent substitution pattern in cyclic pattern

dog: **d** from M1, **o** from M3, **g** from M4

NTNU

Breaking an encryption scheme

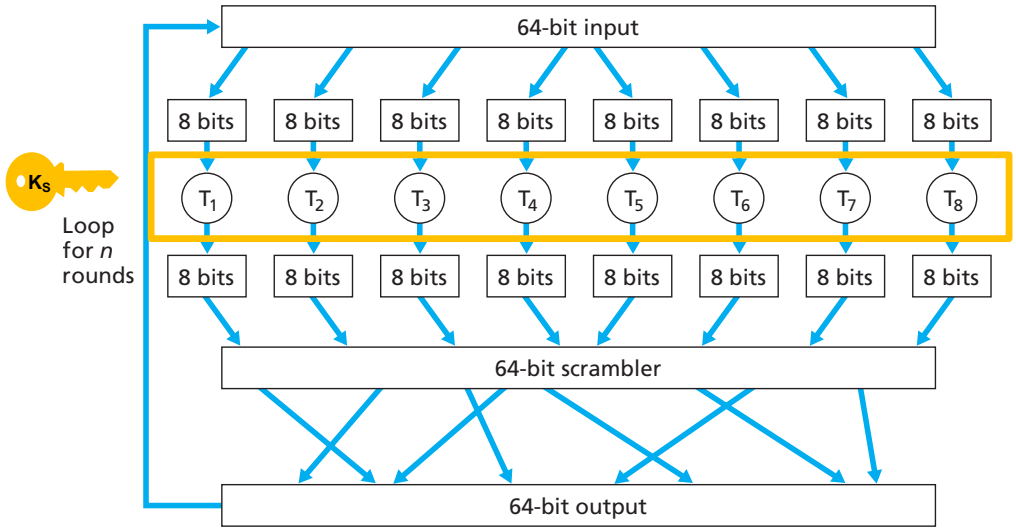
- **Cipher-text only attack**: Trudy has only ciphertext she can analyze
- Two approaches:
  - brute force: search through all keys
  - statistical analysis

- **Known-plaintext attack**: Trudy has plaintext corresponding to ciphertext
  - e.g. in monoalphabetic cipher, Trudy determines pairings for a,l,i,c,e,b,o,

- **Chosen-plaintext attack**: Trudy can get ciphertext for chosen plaintext

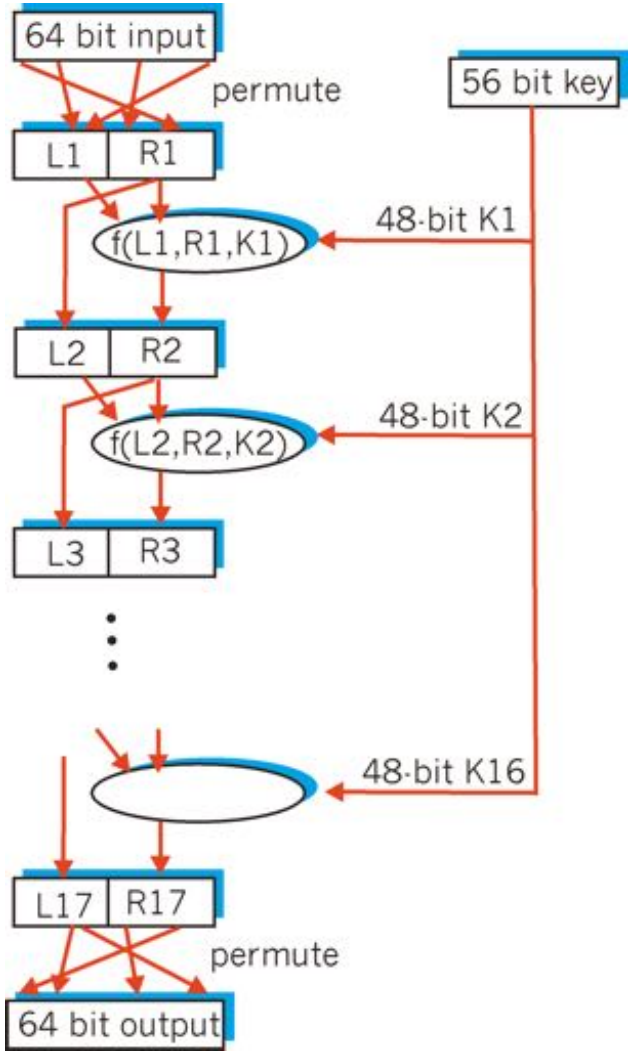
A block cipher encrypts messages in blocks of bits

- Typically use functions that simulate randomly permuted tables
- After n cycles, 64-bit cipher text
- Cipher block chaining to avoid the identification of identical cipher text blocks
- DES (Data Encryption Standard) – 64-bit blocks with a 56-bit key
- 3DES: encrypt 3 times with 3 different keys
- AES (Advanced Encryption Standard) are block ciphers – 128-bit blocks and keys of 128, 192, or 256 bits long
  - Brute force decryption (try each key) taking 1 sec on DES, takes 149 trillion years for AES



DES (Data Encryption Standard)

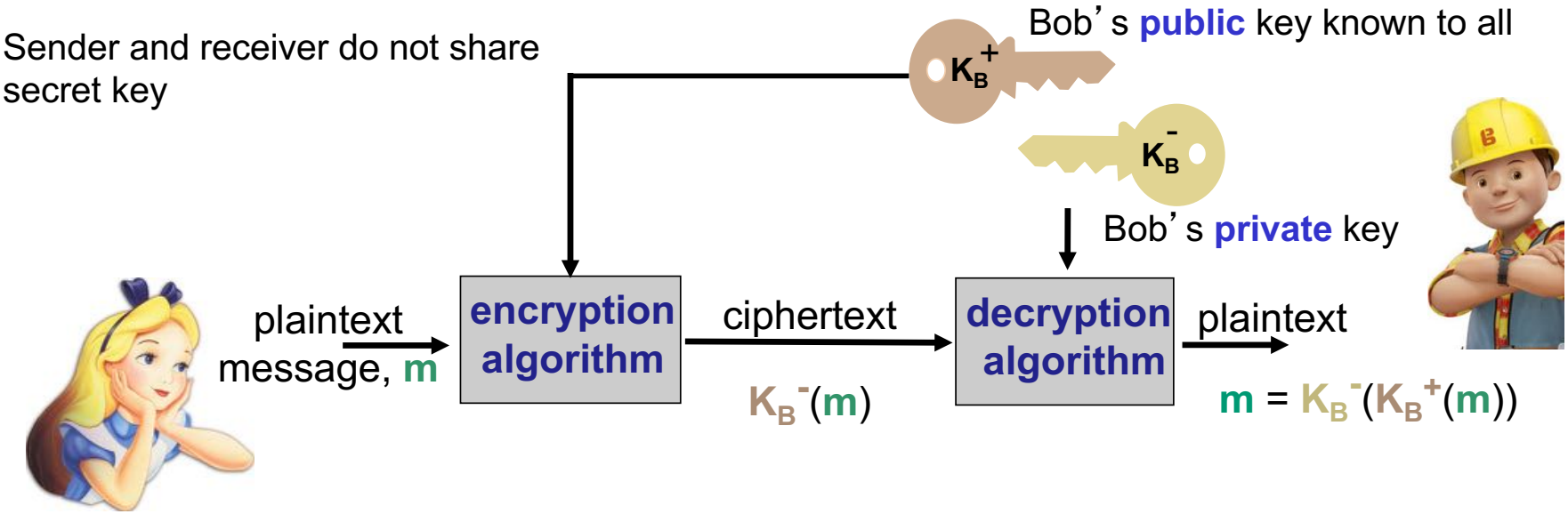
- Block cipher with cipher block chaining
- 56-bit symmetric key, 64-bit plaintext input
  - initial permutation
  - 16 identical "rounds" of function application, each using different 48 bits of key
  - final permutation
- How secure is DES?
  - DES Challenge: 56-bit-key-encrypted phrase decrypted (brute force try each key) in less than a day
  - no known good analytic attack
- Making DES more secure:
  - 3DES: encrypt 3 times with 3 different keys
- US encryption standard [NIST 1993]



NIST National Institute of Standards and Technology (US)

Public key cryptography: public and private key pair

- Sender and receiver do not share secret key

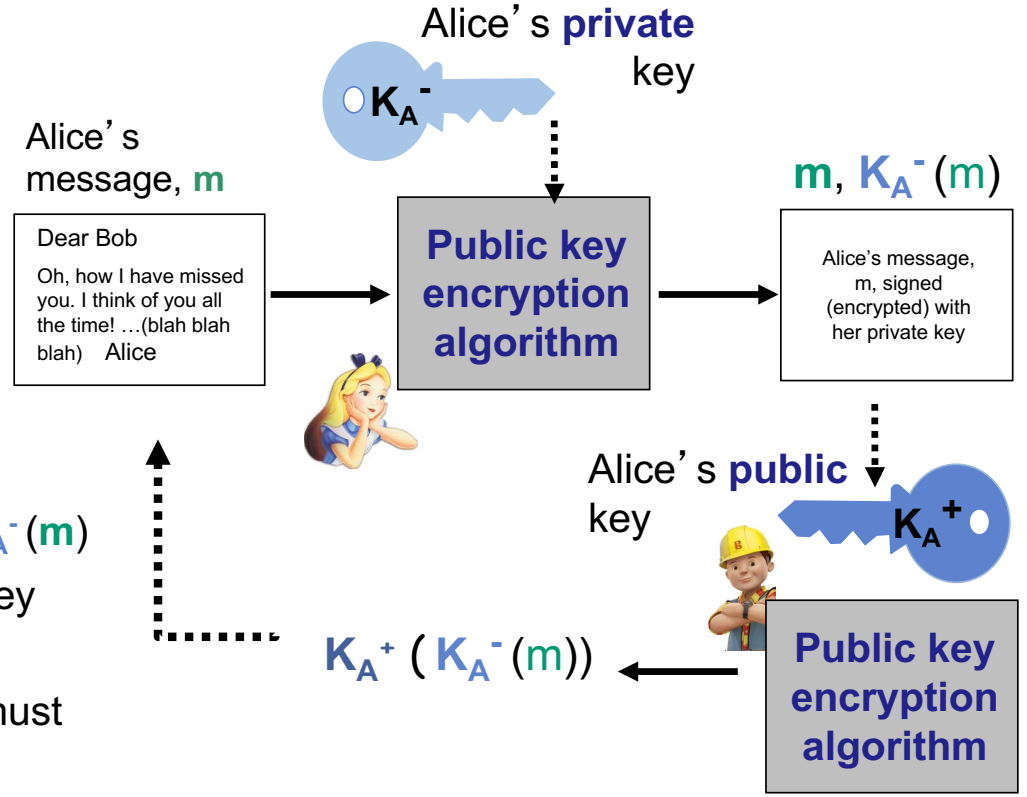


- Need $K_B^+( )$ and $K_B^-( )$ such that $K_B^-(K_B^+(m)) = m$
- Given public key K_B^+ it should be impossible to compute private key K_B^-
- The following property holds: $K_B^-(K_B^+(m)) = m = K_B^+(K_B^-(m))$

- **Works as factoring a big number is hard**!
- RSA: Rivest, Shamir, Adelson algorithm
- Use RSA to exchange a symmetric key, K_S, once both have K_S, use symmetric key cryptography

Message integrity: Digital signatures using public key cryptography

- Sender (Alice) digitally signs document
- Verifiable, non-forgeable: recipient (Bob) can prove that Alice must have signed document
- Alice signs message m by encrypting with private key K_A^-, creating "signed" message, $K_A^-(m)$
- Eg. MD5, SHA-1

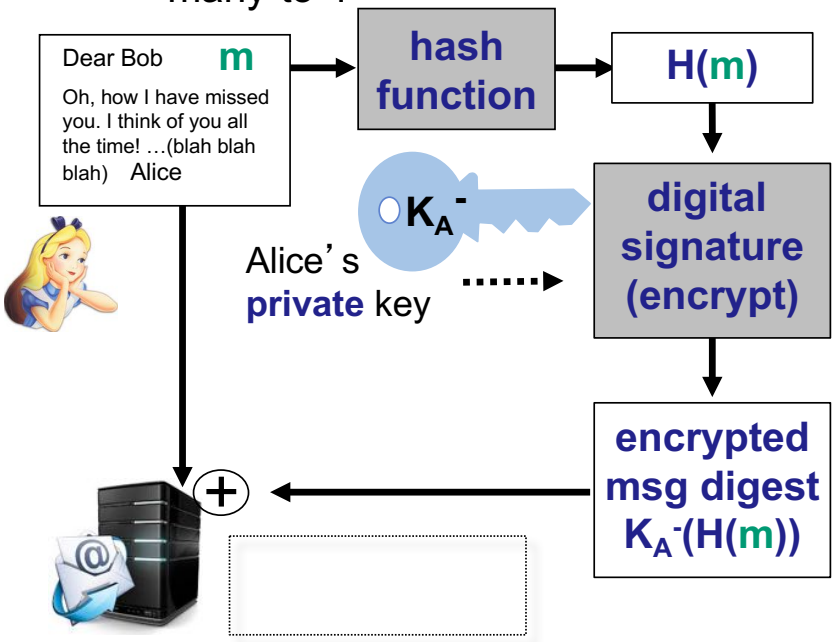


- Bob receives **m**, with signature: **m**, $K_A^-(\mathbf{m})$
- Verifies m by applying Alice's public key K_A^+ to $K_A^-(\mathbf{m})$ and checks
- If $K_A^+(K_A^-(\mathbf{m})) = \mathbf{m}$ whoever signed must have used Alice's private key.

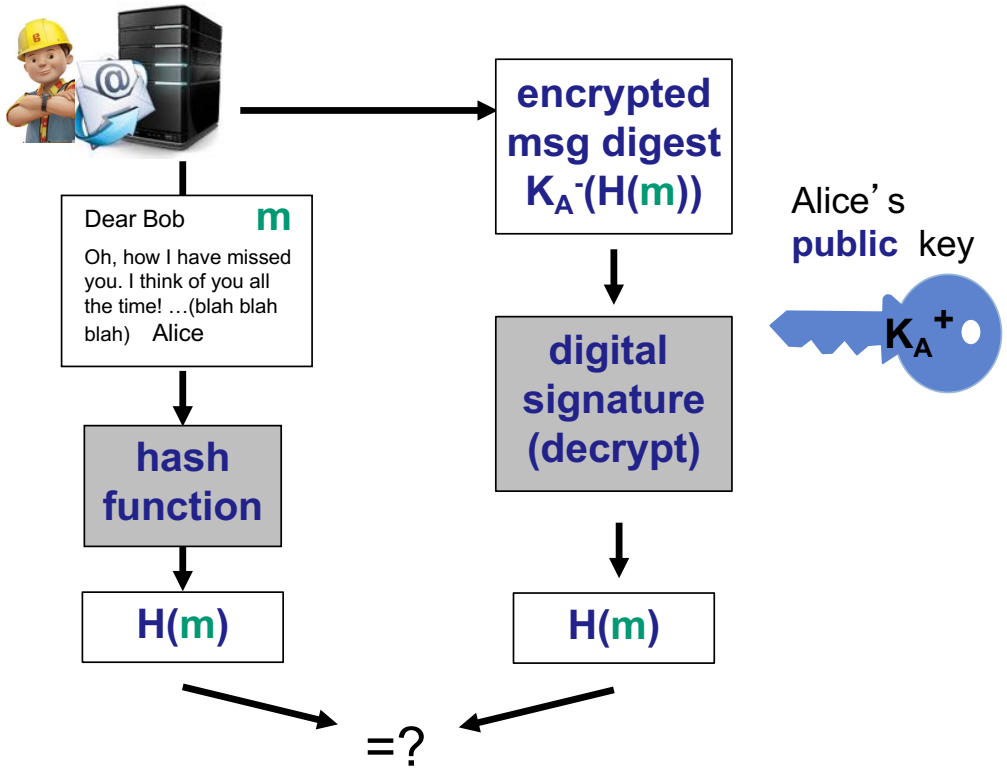
Computationally expensive to public-key-encrypt long messages

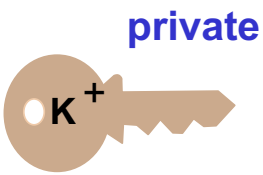
Message digest: fixed size fingerprint

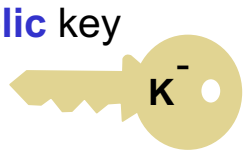
- Apply hash function H to m H(m):
  - produces fixed-size msg digest (fingerprint)
  - given message digest x, computationally infeasible to find **m** such that x = H(**m**)
  - many-to-1



- Receiver verifies message integrity by computing H(**m**) and comparing to received H(**m**)





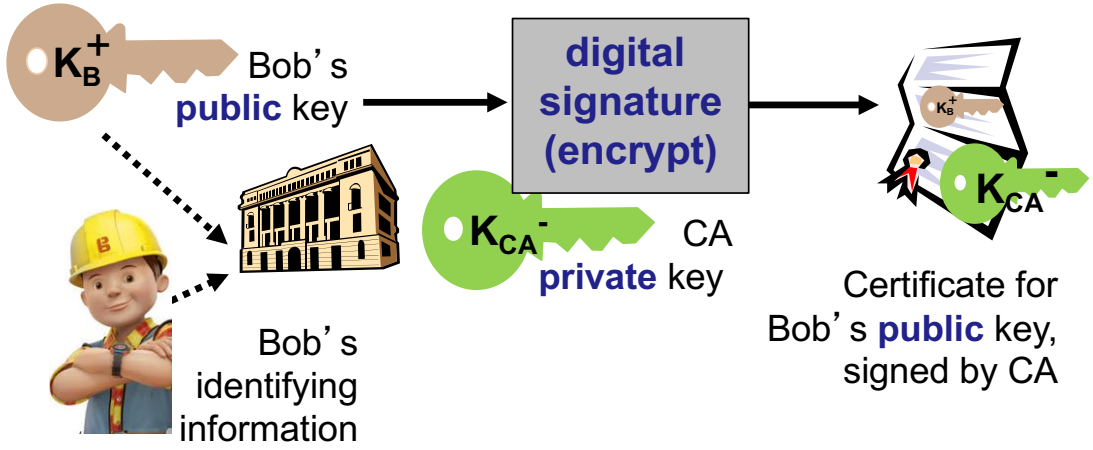


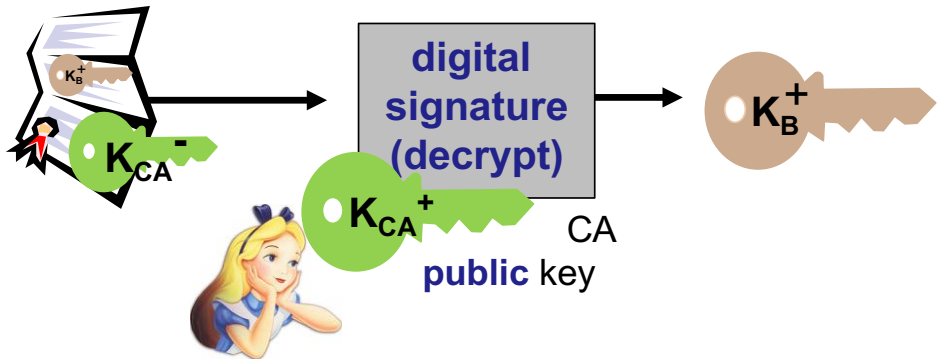
How to distribute the public key?

plaintext message, **m** → **encryption algorithm** → ciphertext **K⁻(m)** → **decryption algorithm** → plaintext **m = K⁻(K⁺(m))**

Certification authorities (CA) binds public key to particular entity, E

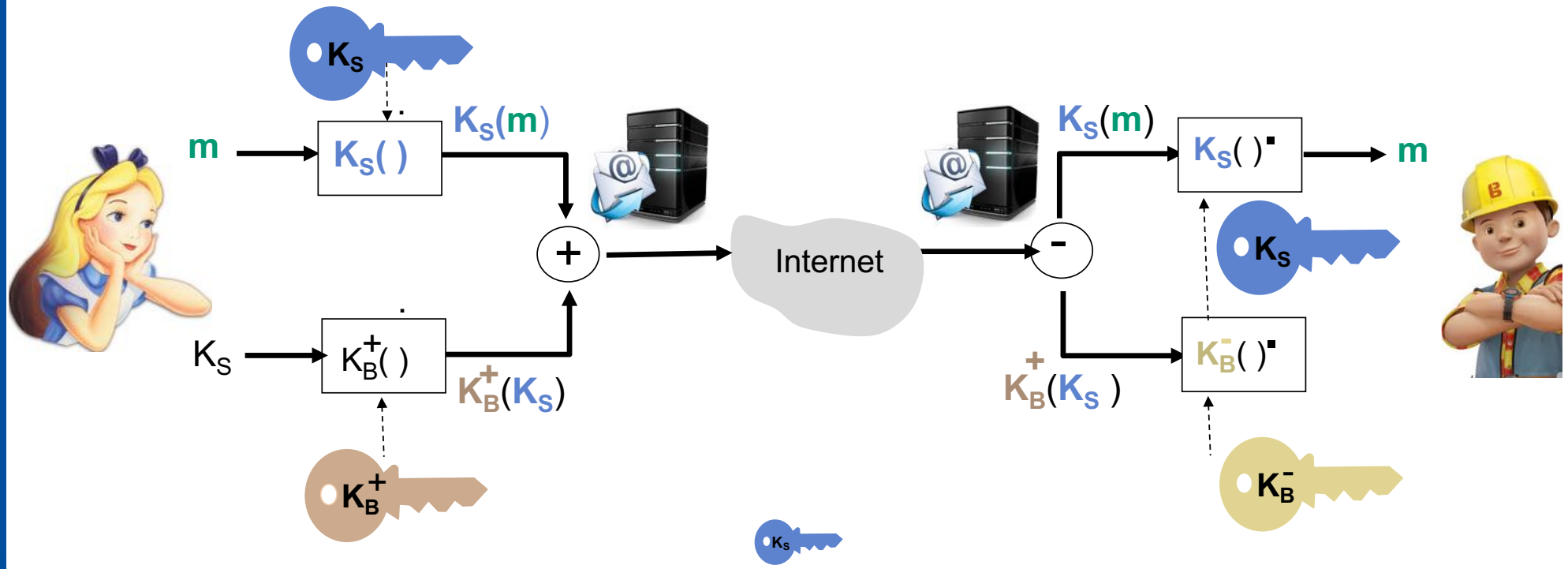
- To avoid signing with false key
- E (person, router) registers its public key with CA
  - E provides "proof of identity" to CA
  - CA creates certificate binding E to its public key
  - certificate containing E's public key digitally signed by CA – CA says "this is E's public key"
- When Alice wants Bob's public key
  - gets Bob's certificate (Bob or elsewhere)
  - apply CA's public key to Bob's certificate, get Bob's public key





Secure e-mail from Alice to Bob – confidentiality

Alice wants to send confidential e-mail, m, to Bob.

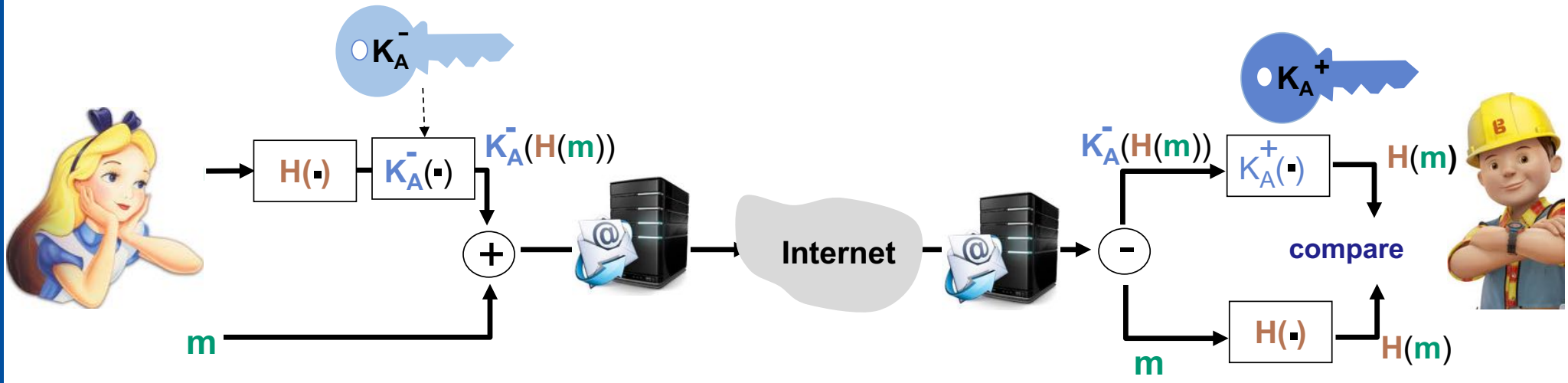


1. Alice generates random *symmetric* private key, K_S
2. encrypts message with K_S (for efficiency)
3. also encrypts K_S with Bob's public key
4. sends both $K_S(m)$ and $K_B(K_S)$

5. Bob uses his private key to decrypt and recover K_S
6. uses K_S to decrypt $K_S(m)$ to recover m

Secure e-mail – Alice provides sender authentication message integrity

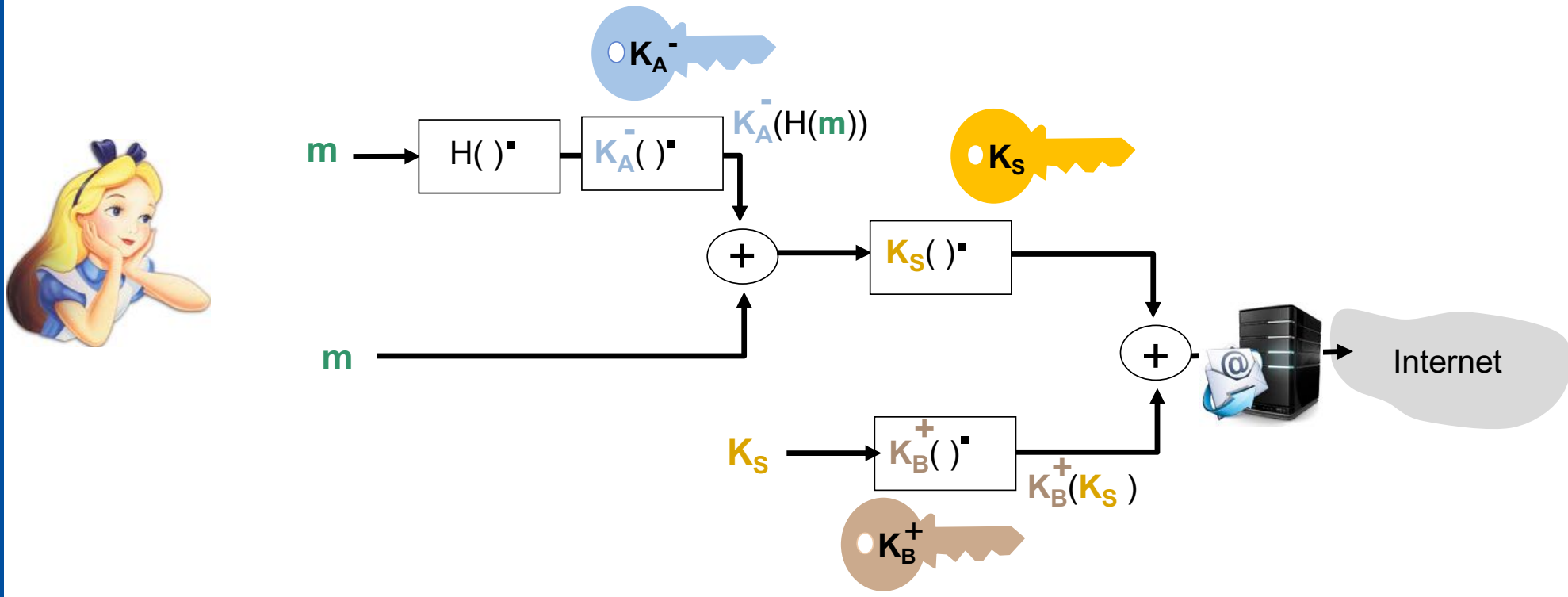
Alice wants to provide sender authentication message integrity



- Alice digitally signs message and sends both message (in the clear) **m** and digital signature $K_A(H(m))$

Secure e-mail (confidential, authenticated)

Alice wants to provide secrecy (confidentiality) and sender authentication message integrity.

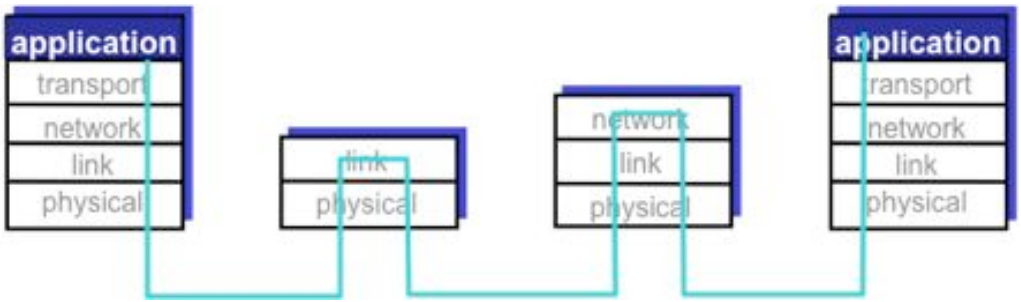


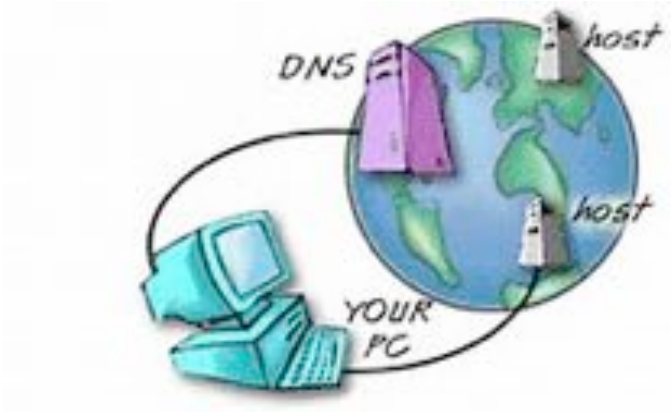
Alice uses three keys: her private key, Bob's public key, newly created symmetric key

Application layer: Roadmap

- 2.1 Principles of network applications
- 2.2 Web and HTTP
- ~~2.3 FTP~~
- 2.4 Electronic mail
  - SMTP, POP3, IMAP
  - 8.5.1 Secure email
- **2.5 DNS – Domain Name System**
- 2.6 P2P applications
- 2.7 Socket programming with UDP and TCP

5-layer internet protocol stack



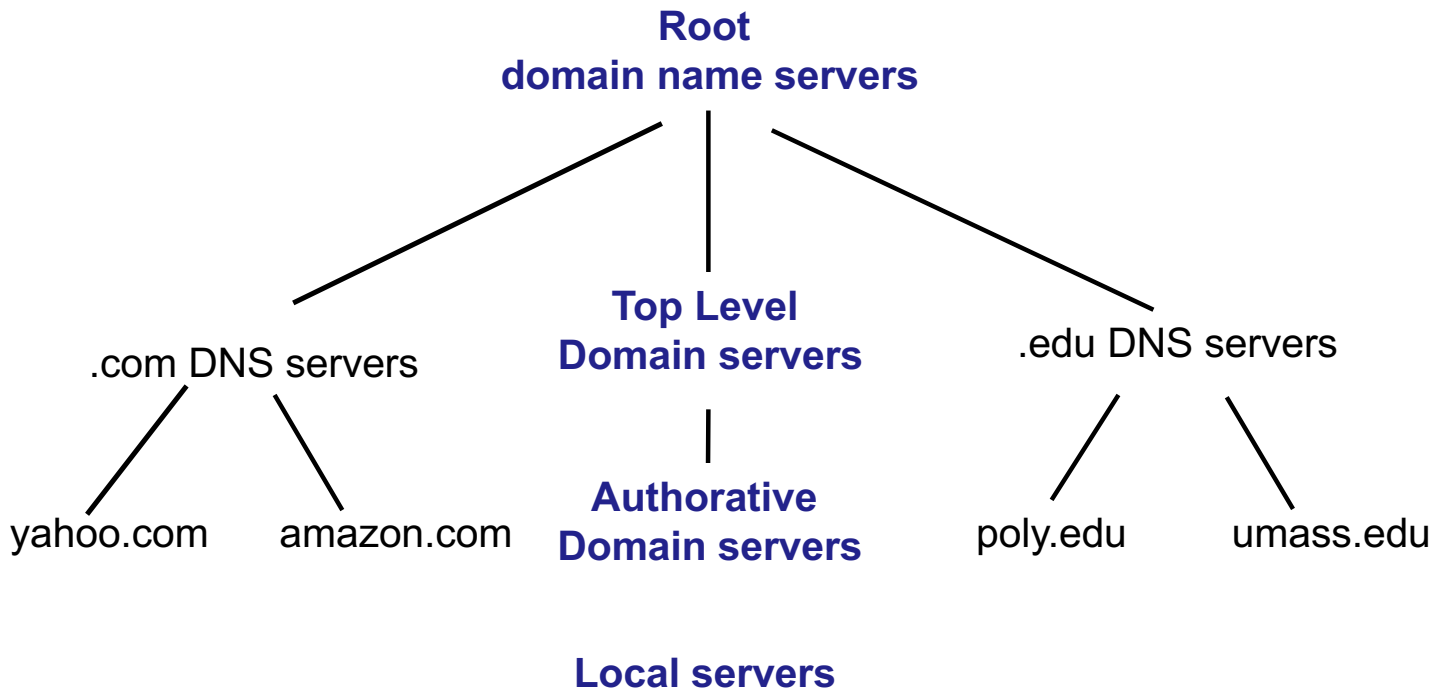


DNS: Domain Name System maps between name and IP address

- People: many identifiers
  - social security #, name, passport #
- Application-layer in internet hosts uses names
  - "**name**", e.g. www.yahoo.com used by humans
- Internet hosts & routers use
  - **IP address** (32 bit) - used for addressing packets
- Need to resolve names (name/address translation)

- **Domain Name System**
  - DNS is core Internet function, implemented as application-layer protocol
  - complexity at network's "edge"

DNS – a distributed, hierarchical database



http://en.wikipedia.org/wiki/List_of_Internet_top-level_domains

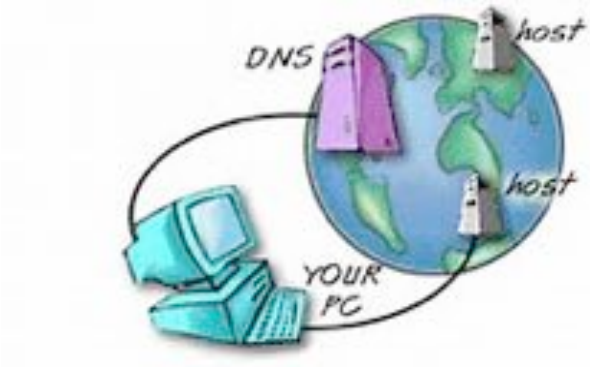
Distributed on many servers

DNS services

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

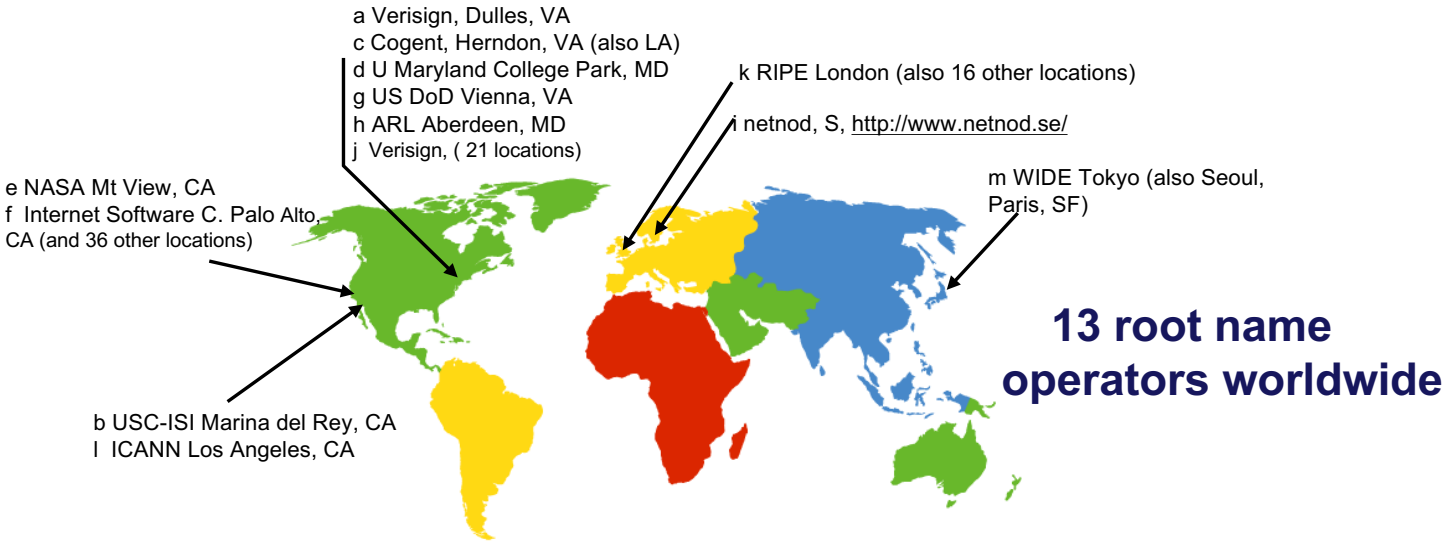
Why not centralize DNS?

- Single point of failure
- Traffic volume
- Distant centralized database
- Maintenance
- Doesn't scale!



Root domain name servers answer requests for the root zone

- Contacted by local name server that can not resolve name
- Root name server
  - contacts authoritative name server if name mapping not known
  - gets mapping
  - returns mapping to local name server
  - http://www.root-servers.org/



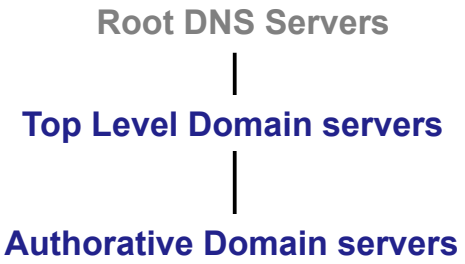
13 root name operators worldwide

Root DNS Servers

Top Level Domain servers

Authorative Domain servers

TLD (Top-level domain) and Authoritative servers



TLD servers

responsible for top-level domains such as

- generic TLDs (gTDL), eg. .com, .org, .net, .edu, .etc
- country-code TLD (ccTDL), eg .uk, .fr, .ca, .no

```
> no.
Server:         192.168.10.1
Address:        192.168.10.1#53

Non-authoritative answer:
no      nameserver = i.nic.no.
no      nameserver = njet.norid.no.
no      nameserver = x.nic.no.
no      nameserver = z.nic.no.
no      nameserver = not.norid.no.
no      nameserver = y.nic.no.

Authoritative answers can be found from:
i.nic.no        internet address = 194.146.106.6
x.nic.no        internet address = 128.39.8.40
y.nic.no        internet address = 193.75.4.22
z.nic.no        internet address = 158.38.8.133
y.nic.no        has AAAA address 2001:8c0:8200:1::2
z.nic.no        has AAAA address 2001:700::52d:158:38:8:133
```

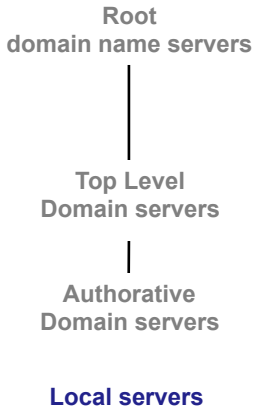
- http://www.norid.no/statistikk/

Authoritative DNS servers

- organization's DNS servers, providing authoritative hostname to IP mappings for organization's servers (e.g. web, mail)
- maintained by organization or service provider
  - eg. ntnu.no.

```
> ntnu.no
Server:         192.168.10.1
Address:        192.168.10.1#53

Non-authoritative answer:
ntnu.no nameserver = ns2.ntnu.no.
ntnu.no nameserver = ns1.ntnu.no.

Authoritative answers can be found from:
ns1.ntnu.no     internet address = 129.241.0.208
ns1.ntnu.no     has AAAA address 2001:700:300::208
ns2.ntnu.no     internet address = 129.241.0.209
ns2.ntnu.no     has AAAA address 2001:700:300::209
```

A local name server is part of the end system configuration



- When host makes DNS query, query is sent to its **local DNS server**
- Does not strictly belong to the hierarchy
- Each ISP (residential ISP, company, university) has one
  - also called “default name server”
  - You find your local name server by ipconfig /all, ipconfig getpacket *if*

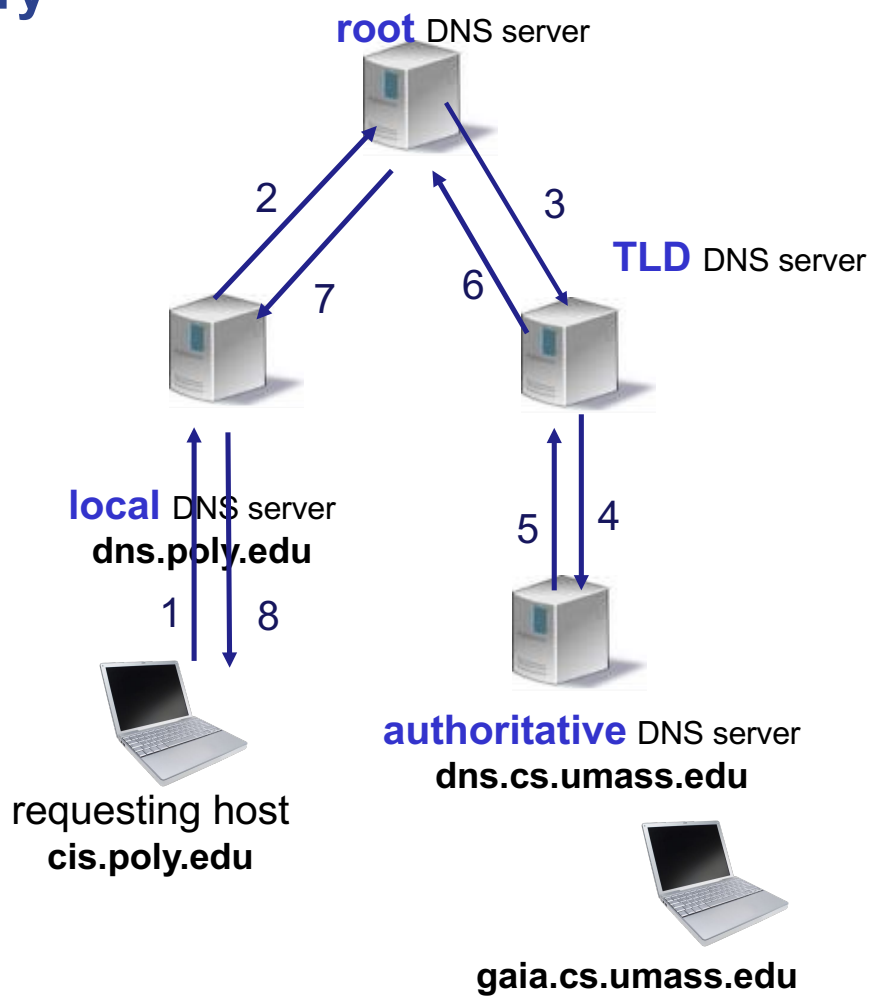
```
yiaddr = 10.0.0.37
siaddr = 10.0.0.138
subnet_mask (ip): 255.255.255.0
broadcast_address (ip): 10.0.0.255
domain_name_server (ip_mult): {130.67.15.198, 193.213.112.4, 10.0.0.138}
router (ip_mult): {10.0.0.138}
```

- The local DNS server acts as proxy, forwards query into hierarchy

Name resolution: Recursive query

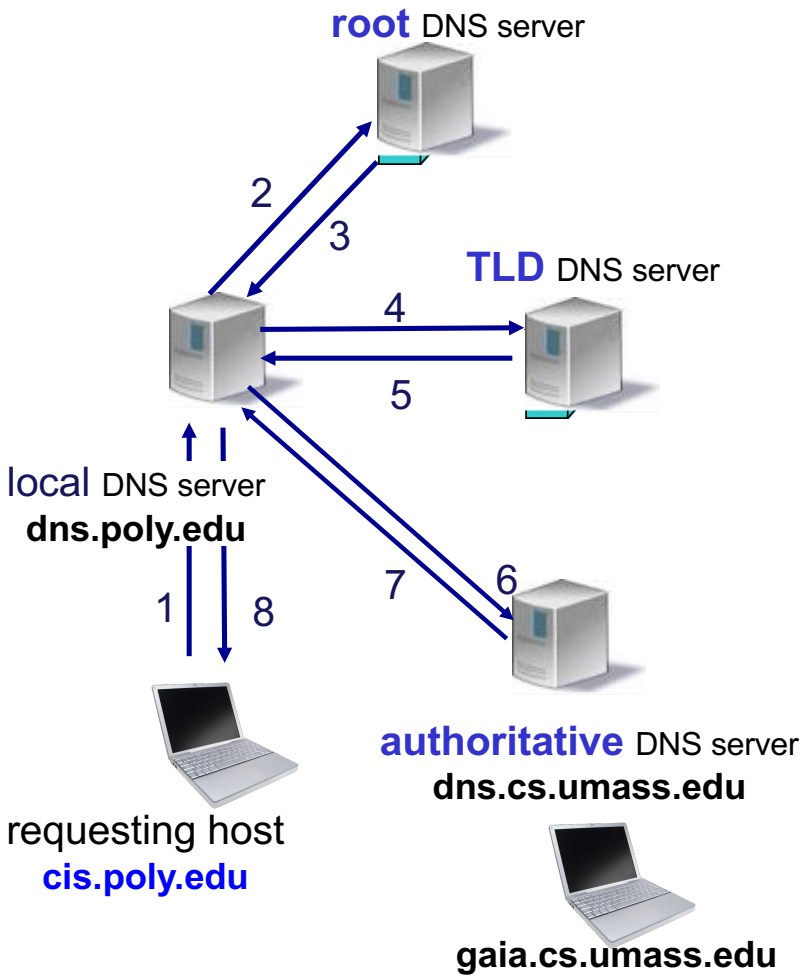
- Host at cis.poly.edu wants IP address for **gaia.cs.umass.edu**
- Puts burden of name resolution on contacted name server

Any performance challenges?



Name resolution: Iterated query

- Host at **cis.poly.edu** wants IP address for **gaia.cs.umass.edu**
- Contacted server replies with name of server to contact
  - "I don't know this name, but ask this server"



Once (any) name server learns mapping, it caches mapping

- Cache entries timeout (disappear) after some time
- TLD (top level domain) servers typically cached in local name servers
  - thus root name servers not often visited
  - http://www.internic.net/zones/root.zone

- DNS servers are configured
  - statically from a configuration file
  - dynamically via DNS messages with the UPDATE option
  - RFC2136

A distributed data base storing resource records (RR) of various types, eg

- **type=A**
  - **name** is hostname
  - **value** is IP address
- **type=CNAME**
  - **name** is alias name for some "canonical" (the real) name
  - **value**

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

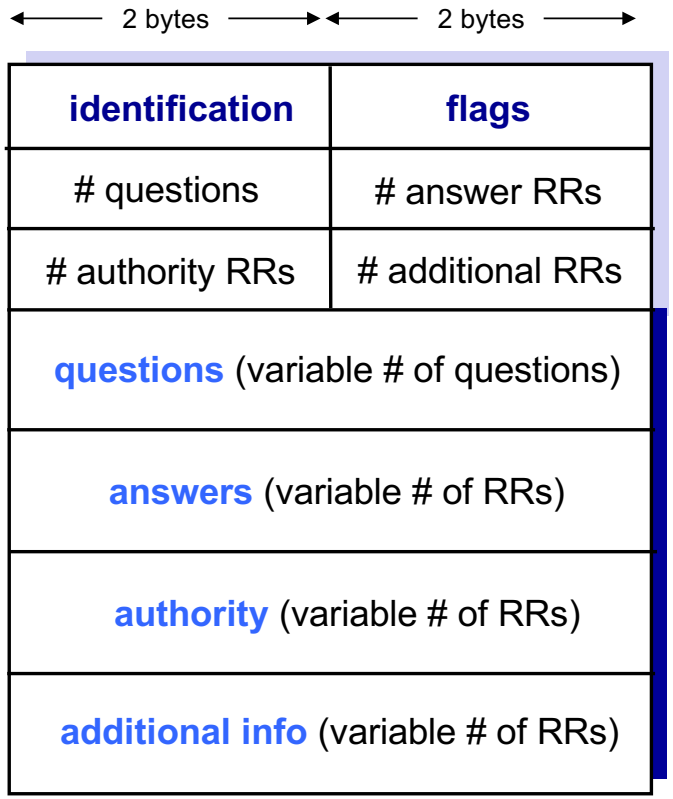
time to live

- **type=NS**
  - **name** is domain (e.g. foo.com)
  - **value** is hostname of authoritative name server for this domain
- **type=MX**
  - **value** is name of mailserver associated with **name**

Protocol and messages

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

- **Message header**
  - **identification**: 16 bit for query, reply to query uses same id
  - **flags**:
    - query or reply
    - recursion desired
    - recursion available
    - reply is authoritative



Query and reply message format

DNS


```
KM:~ kjerstiⓢ nslookup www.google.com
Server:         92.220.228.70
Address:        92.220.228.70#53

Non-authoritative answer:
Name:   www.google.com
Address: 81.167.35.106
Name:   www.google.com
Address: 81.167.35.113
Name:   www.google.com
Address: 81.167.35.102
Name:   www.google.com
Address: 81.167.35.110
Name:   www.google.com
Address: 81.167.35.99
Name:   www.google.com
Address: 81.167.35.90
Name:   www.google.com
Address: 81.167.35.91
Name:   www.google.com
Address: 81.167.35.95
Name:   www.google.com
Address: 81.167.35.88
Name:   www.google.com
Address: 81.167.35.117
Name:   www.google.com
Address: 81.167.35.101
Name:   www.google.com
Address: 81.167.35.123
Name:   www.google.com
Address: 81.167.35.80
Name:   www.google.com
Address: 81.167.35.84
Name:   www.google.com
Address: 81.167.35.121
Name:   www.google.com
Address: 81.167.35.112

KM:~ kjerstiⓢ
```

Rotates each query

```
KM:~ kjerstiⓢ nslookup
> set type = MX
*** Invalid option: type
> set type=MX
> ntnu.no
Server:         92.220.228.70
Address:        92.220.228.70#53

Non-authoritative answer:
ntnu.no mail exchanger = 10 mx.ntnu.no.

Authoritative answers can be found from:
ntnu.no nameserver = ns2.ntnu.no.
ntnu.no nameserver = ns1.ntnu.no.
mx.ntnu.no      internet address = 129.241.56.67
mx.ntnu.no      has AAAA address 2001:700:300:3::234
mx.ntnu.no      has AAAA address 2001:700:300:3::226
ns1.ntnu.no     internet address = 129.241.0.208
ns1.ntnu.no     has AAAA address 2001:700:300::208
ns2.ntnu.no     internet address = 129.241.0.209
ns2.ntnu.no     has AAAA address 2001:700:300::209
> set type=A
> ntnu.no
Server:         92.220.228.70
Address:        92.220.228.70#53

Non-authoritative answer:
Name:   ntnu.no
Address: 129.241.56.116
>
```

Standard IPv4 query – response

QUERY

```
▷ Ethernet II, Src: Apple_ee:45:5a (c8:bc:c8:ee:45:5a), Dst: ZyxelCom_11:88:e3 (cc:5d:4e:11:88:e3)
▷ Internet Protocol Version 4, Src: 10.0.0.37 (10.0.0.37), Dst: 193.213.112.4 (193.213.112.4)
▷ User Datagram Protocol, Src Port: 54937 (54937), Dst Port: domain (53)
▽ Domain Name System (query)
    [Response In: 22]
    Transaction ID: 0x441a
  ▷ Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ▽ Queries
    ▷ sslvpn.ntnu.no: type A, class IN
```

```
▽ Flags: 0x0100 Standard query
    0... .... .... .... = Response: Message is a query
    .000 0... .... .... = Opcode: Standard query (0)
    .... ..0. .... .... = Truncated: Message is not truncated
    .... ...1 .... .... = Recursion desired: Do query recursively
    .... .... .0.. .... = Z: reserved (0)
    .... .... ...0 .... = Non-authenticated data: Unacceptable
```

name, type fields for a query

RESPONSE

```
▷ Ethernet II, Src: ZyxelCom_11:88:e3 (cc:5d:4e:11:88:e3), Dst: Apple_ee:45:5a (c8:bc:c8:ee:45:5a)
▷ Internet Protocol Version 4, Src: 193.213.112.4 (193.213.112.4), Dst: 10.0.0.37 (10.0.0.37)
▷ User Datagram Protocol, Src Port: domain (53), Dst Port: 54937 (54937)
▽ Domain Name System (response)
    [Request In: 21]
    [Time: 0.011411000 seconds]
    Transaction ID: 0x441a
  ▷ Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 2
    Additional RRs: 4
  ▽ Queries
    ▷ sslvpn.ntnu.no: type A, class IN
  ▽ Answers
    ▷ sslvpn.ntnu.no: type A, class IN, addr 129.241.77.150
  ▽ Authoritative nameservers
    ▷ ntnu.no: type NS, class IN, ns ns1.ntnu.no
    ▷ ntnu.no: type NS, class IN, ns ns2.ntnu.no
  ▽ Additional records
    ▷ ns1.ntnu.no: type A, class IN, addr 129.241.0.208
    ▷ ns1.ntnu.no: type AAAA, class IN, addr 2001:700:300::208
    ▷ ns2.ntnu.no: type A, class IN, addr 129.241.0.209
    ▷ ns2.ntnu.no: type AAAA, class IN, addr 2001:700:300::209
```

name, type fields for a query

RRs in response to query

records for authoritative servers

additional “helpful” info that may be used

Standard IPv4 resource records in response

RESPONSE

```
▷ Ethernet II, Src: ZyxelCom_11:88:e3 (cc:5d:4e:11:88:e3), Dst: Apple_ee:45:5a (c8:bc:c8:ee:45:5a)
▷ Internet Protocol Version 4, Src: 193.213.112.4 (193.213.112.4), Dst: 10.0.0.37 (10.0.0.37)
▷ User Datagram Protocol, Src Port: domain (53), Dst Port: 54937 (54937)
▽ Domain Name System (response)
    [Request In: 21]
    [Time: 0.011411000 seconds]
    Transaction ID: 0x441a
  ▷ Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 2
    Additional RRs: 4
  ▽ Queries
    ▷ sslvpn.ntnu.no: type A, class IN
  ▽ Answers
    ▷ sslvpn.ntnu.no: type A, class IN, addr 129.241.77.150
  ▽ Authoritative nameservers
    ▷ ntnu.no: type NS, class IN, ns ns1.ntnu.no
    ▷ ntnu.no: type NS, class IN, ns ns2.ntnu.no
  ▽ Additional records
    ▷ ns1.ntnu.no: type A, class IN, addr 129.241.0.208
    ▷ ns1.ntnu.no: type AAAA, class IN, addr 2001:700:300::208
    ▷ ns2.ntnu.no: type A, class IN, addr 129.241.0.209
    ▷ ns2.ntnu.no: type AAAA, class IN, addr 2001:700:300::209
```

```
▽ Flags: 0x8180 Standard query response, No error
    1... .... .... .... = Response: Message is a response
    .000 0... .... .... = Opcode: Standard query (0)
    .... .0.. .... .... = Authoritative: Server is not an authority for domain
    .... ..0. .... .... = Truncated: Message is not truncated
    .... ...1 .... .... = Recursion desired: Do query recursively
    .... .... 1... .... = Recursion available: Server can do recursive queries
    .... .... .0.. .... = Z: reserved (0)
    .... .... ..0. .... = Answer authenticated: Answer/authority portion was not authenticated by the server
    .... .... ...0 .... = Non-authenticated data: Unacceptable
    .... .... .... 0000 = Reply code: No error (0)
```

E.g. resource records for slottet.no

```
▾ Domain Name System (response)
    [Request In: 8429]
    [Time: 0.011487000 seconds]
    Transaction ID: 0x3bf0
  ▸ Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 3
    Authority RRs: 0
    Additional RRs: 0
  ▾ Queries
    ▸ www.slottet.no: type A, class IN
  ▾ Answers
    ▸ www.slottet.no: type CNAME, class IN, cname www.kongehuset.no
    ▸ www.kongehuset.no: type CNAME, class IN, cname kongehuset.kunder.ravn.no
    ▸ kongehuset.kunder.ravn.no: type A, class IN, addr 195.159.126.222
```

```
▾ Domain Name System (response)
    [Request In: 8795]
    [Time: 0.009265000 seconds]
    Transaction ID: 0x20b1
  ▸ Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 2
    Authority RRs: 0
    Additional RRs: 0
  ▾ Queries
    ▸ slottet.no: type MX, class IN
  ▾ Answers
    ▸ slottet.no: type MX, class IN, preference 5, mx mail2.slottet.no
    ▸ slottet.no: type MX, class IN, preference 20, mx mail3.slottet.no
```

Inserting resource records, e.g. new startup “Network Utopia”

- At **DNS registrar** register name networkutopia.com
  - provide names, IP addresses of authoritative name server (primary and secondary)
  - registrar inserts two RRs into com TLD server:
    - **(networkutopia.com, dns1.networkutopia.com, NS)**
    - **(dns1.networkutopia.com, 212.212.212.1, A)**

- In the Network Utopia´s authority name server register **authority web server** and **mail server**, both run by the server 212.212.7.4, ignore `ttl`

  - type **A** record for authoritative server www.networkutopia.com
    (www.networkutopia.com, 212.212.71.4, A)
  - type **MX** record for networkutopia.com
    (networkutopia.com, 212.212.71.4, MX)

Security attacks

DDoS attacks

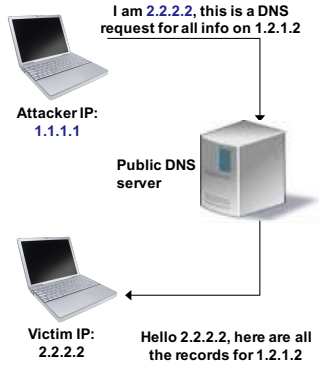
- Bombard root servers with traffic
  - Traffic filtering
- Bombard TLD servers
  - Local DNS servers cache IPs of TLD servers, allowing root server bypass
  - Potentially more dangerous

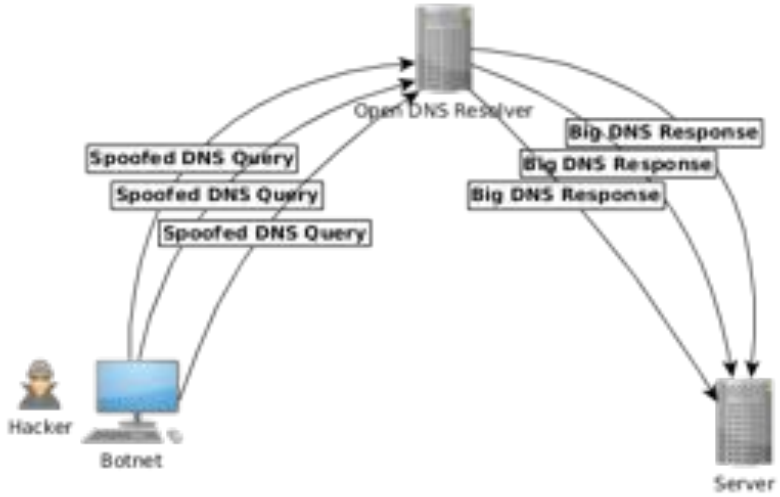
Redirect attacks

- Man-in-middle
  - Intercept queries
- DNS poisoning
  - Send bogus replies to DNS server, which caches

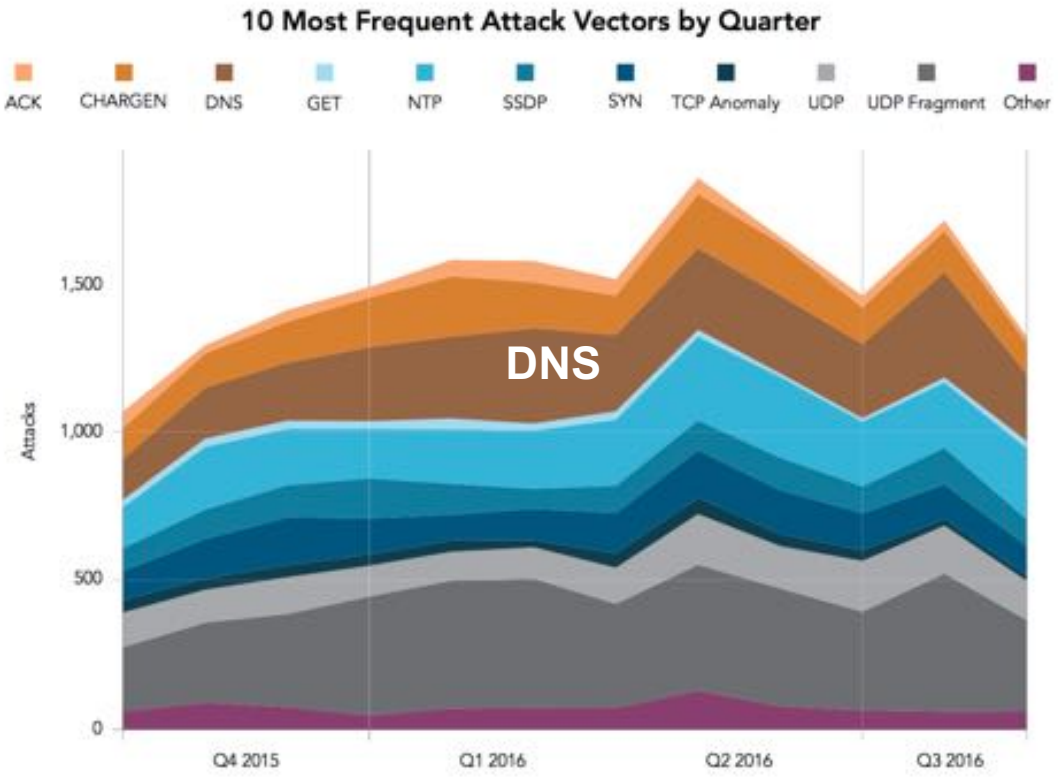
Exploit DNS for DDoS attacks

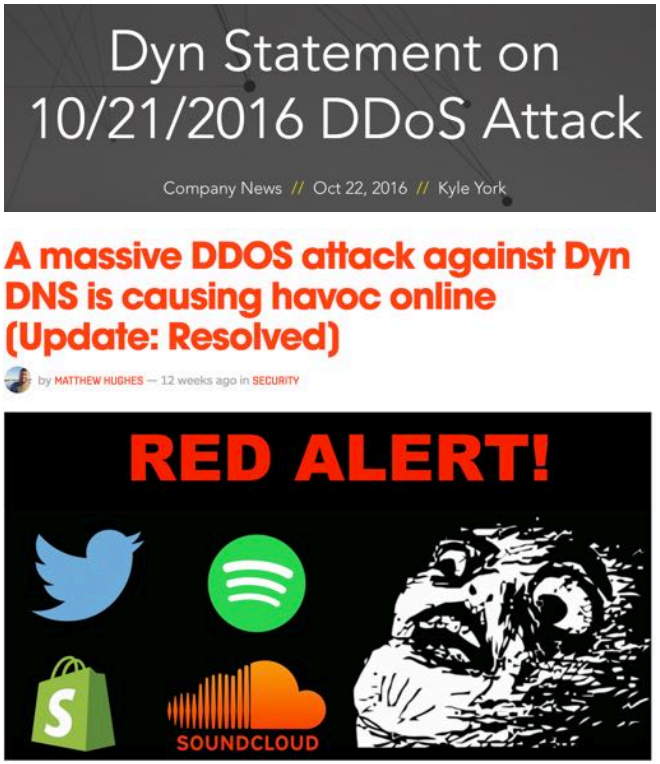
- Send queries with spoofed source address: victim IP
- Requires amplification
- DNS amplification attacks





Distributed denial of service attack vectors





Source: akamai's [state of the internet] / security Q3 2016 report

Application layer: Roadmap

2.1 Principles of network applications

2.2 Web and HTTP

2.3 FTP

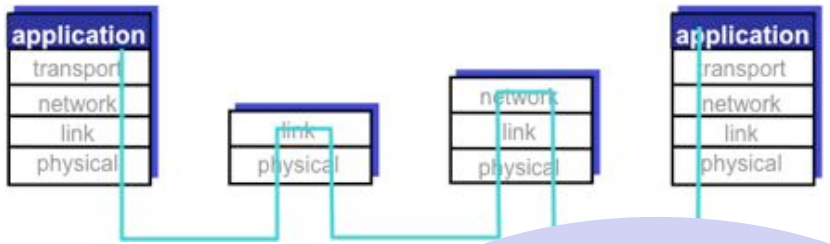
2.4 Electronic mail

- SMTP, POP3, IMAP
- 8.5.1 Secure email

2.5 DNS

2.6 P2P applications

2.7 Socket programming with UDP and TCP

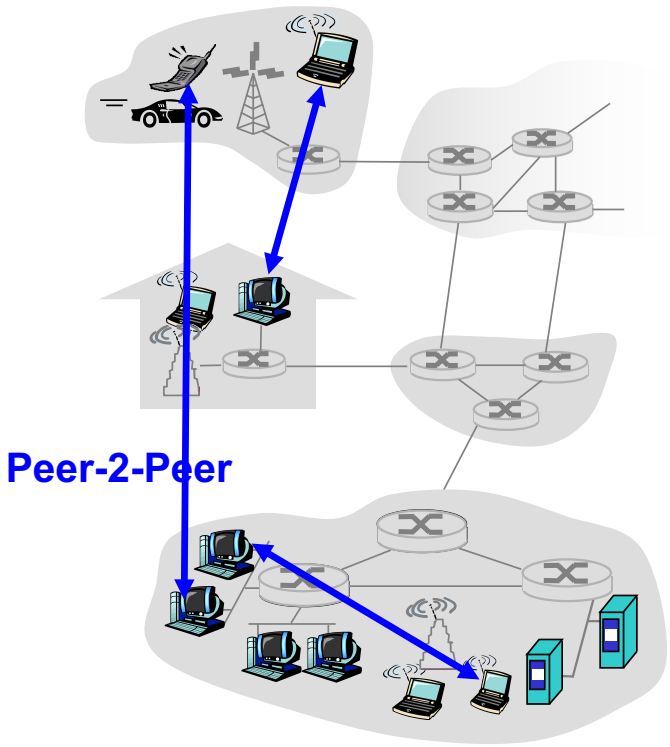


Nov 11 2013

"For the first time ever, peer-to-peer filesharing has fallen below 10% of total traffic in North America, which is a stark difference from the 60% share it consumed 11 years ago," said Dave Caputo, CEO, Sandvine.

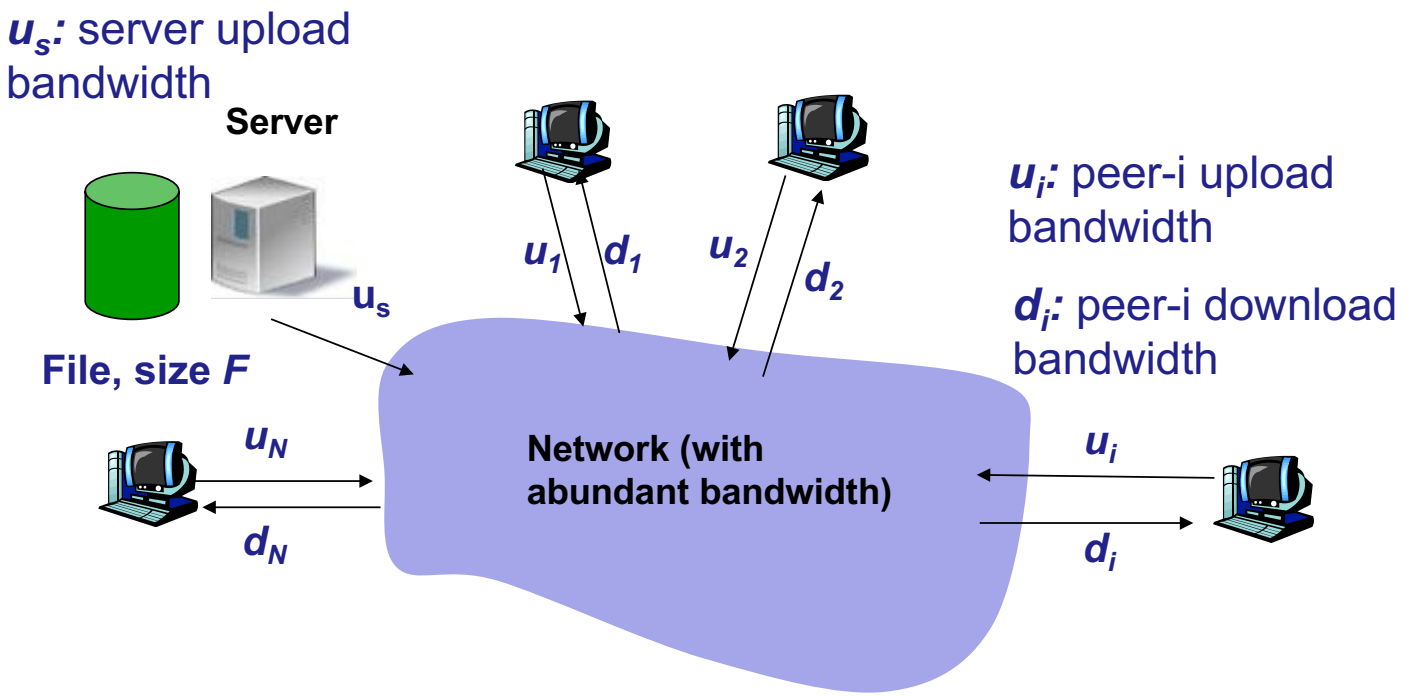
Pure peer-2-peer architecture

- Arbitrary end systems communicate directly
- No always-on server
- Peers are intermittently connected and change IP addresses
- Peers: both client and server
- Highly scalable but difficult to manage
- Examples
  1. **File distribution (Bit Torrent)**
  2. **Searching for information (Distributed Hash Tables)**



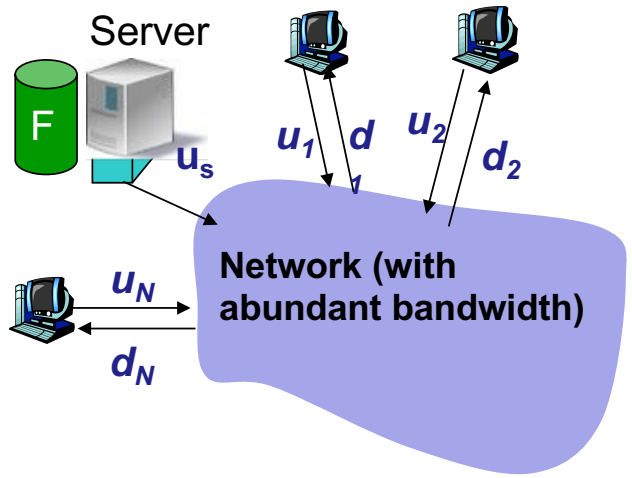
File distribution: client-server vs. P2P

- How much time to distribute file from one server to N peers?



File distribution time: client-server

- Server sequentially sends N copies in
  - **NF/u_s** time
- Client-i takes
  F/d_i time to download
- Time to distribute *F* to *N* clients using client/server approach

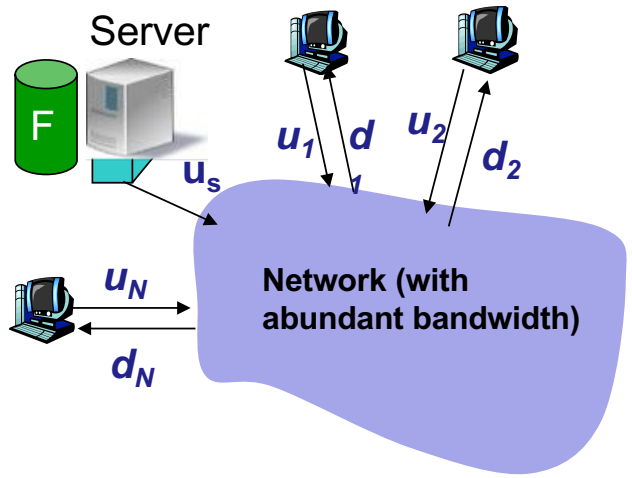


increases linearly in N (for large N)

$$= D_{cs} >= \max \{ NF/u_s, F/d_{min} \}$$

File distribution time: P2P

- Server must send at least one copy (F/u_s time)
- Client i takes F/d_i time to download
- NF bits must be downloaded (aggregate)

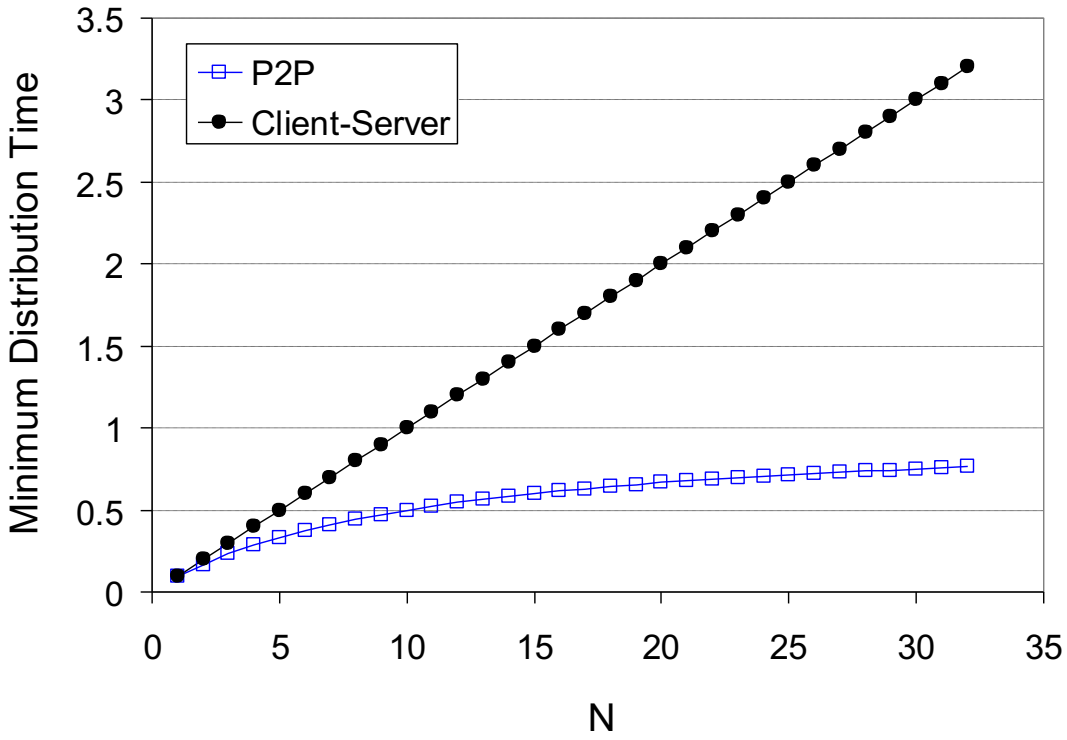


- Fastest possible upload rate: $u_s + \Sigma u_i$

$$D_{P2P} >= \max \{ F/u_s, F/d_{min} , NF/(u_s + \Sigma u_i) \}$$

Client-server vs. P2P: example

Client **upload rate** = u, F/u = 1 hour, u_s = 10u
Smallest **download rate** $d_{min} \geq u_s$ (so no bottleneck)



max { NF/u_s, ~~F/d_{min}~~}

$NF/u_s = N/10$

max {~~F/u_s~~, ~~F/d_{min}~~, $NF/(u_s + \Sigma u_i)$}

$NF/(u_s + \Sigma u_i) = NF/(10+N)u = N/(10+N)$

Application layer: Roadmap

2.1 Principles of network applications

2.2 Web and HTTP

2.3 FTP

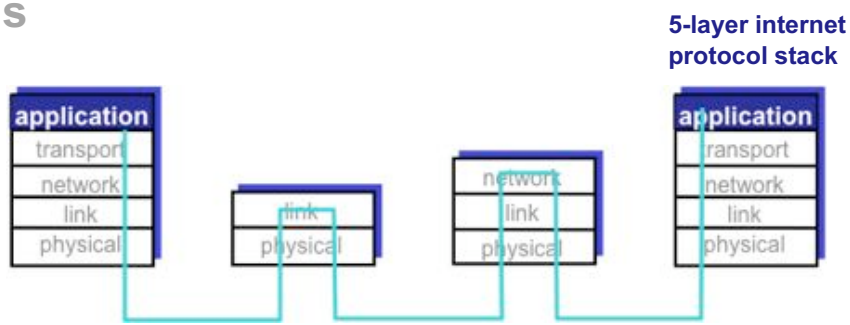
2.4 Electronic mail

- SMTP, POP3, IMAP
- 8.2-3 + 8.5.1 Secure email

2.5 DNS

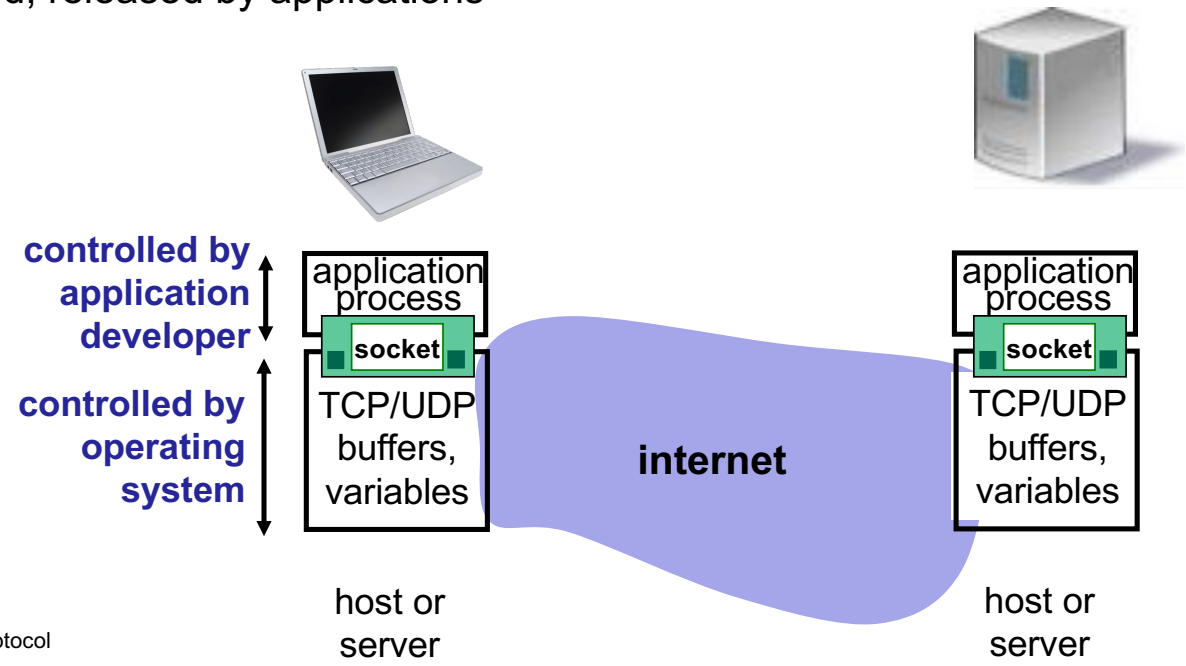
2.6 P2P applications

2.7 Socket programming with UDP and TCP



Network programming using the socket application programming interface (API)

- **Goal:** learn how to build client/server application that communicate using sockets
- **Socket**: a door between application process and end-end-transport protocol (UDP or TCP)
  - explicitly created, used, released by applications



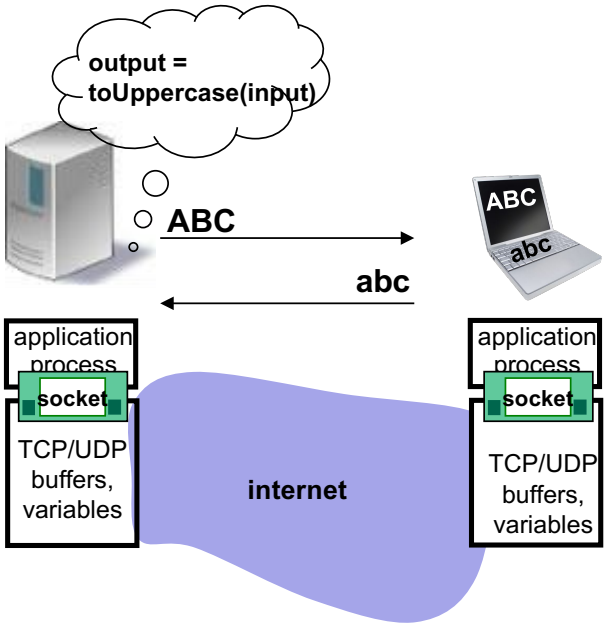
UDP User Datagram Protocol
TCP Transmission Control Protocol

Socket programming to utilize the communication services of the operating system

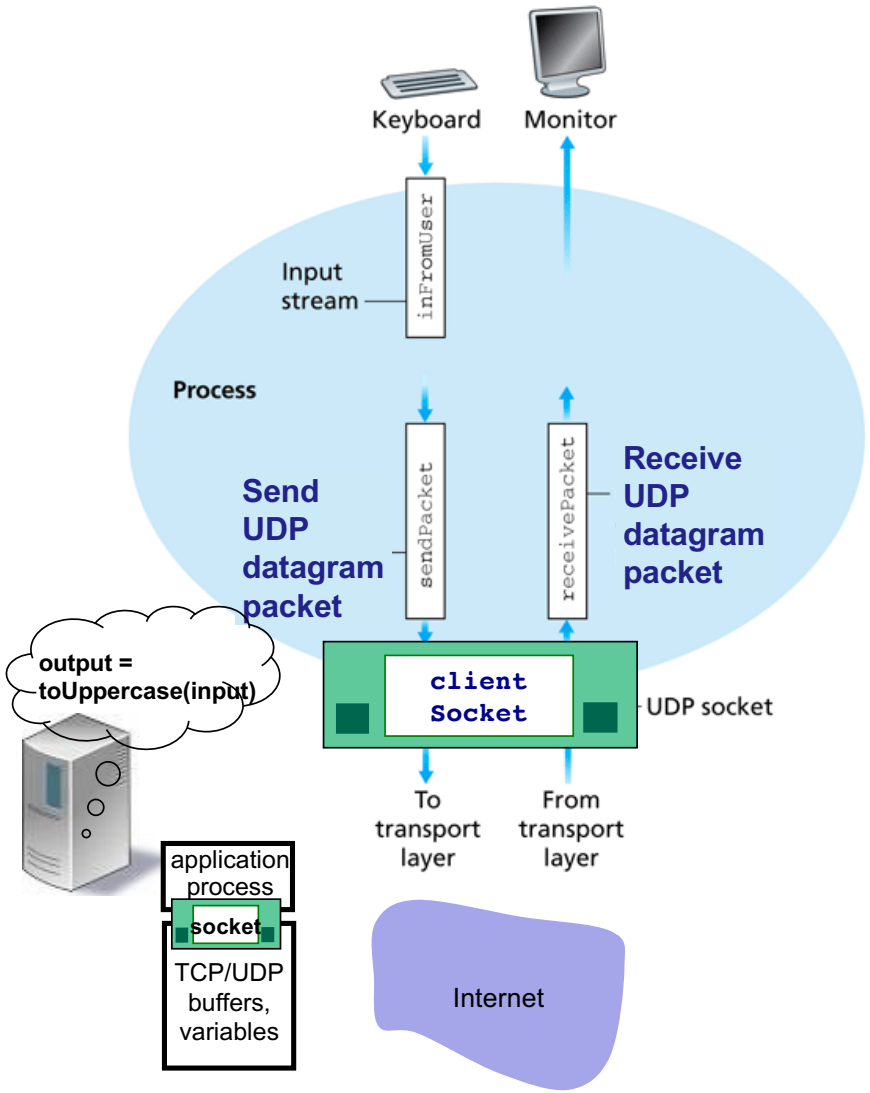
- Two socket types for two transport communication services
  - **UDP**: unreliable **datagrams**
  - **TCP**: reliable, byte **stream**-oriented

Application example

1. Client reads a line of characters (data) from its keyboard and sends the data to the server.
2. The server receives the data and converts characters to uppercase.
3. The server sends the modified data to the client.
4. The client receives the modified data and displays the line on its screen.



CLIENT sends UDP datagram to be received by SERVER

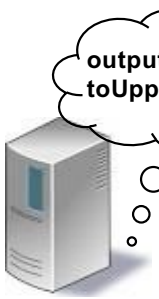


- **UDP: no "connection" between client and server**
- No handshaking up front
- Sender explicitly attaches IP address and port of destination to each packet
- **Server must extract IP address, port of sender from received packet**
- UDP: transmitted data may be received out of order, or lost
- **Same server socket receives from different clients**

Client/server socket interaction: UDP

SERVER

create socket,
`port= x`
`serverSocket = DatagramSocket()`



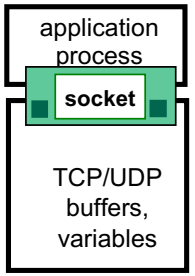
read datagram from `serverSocket`

write reply to `serverSocket` specifying client address, port number

CLIENT

create socket,
`clientSocket = DatagramSocket()`

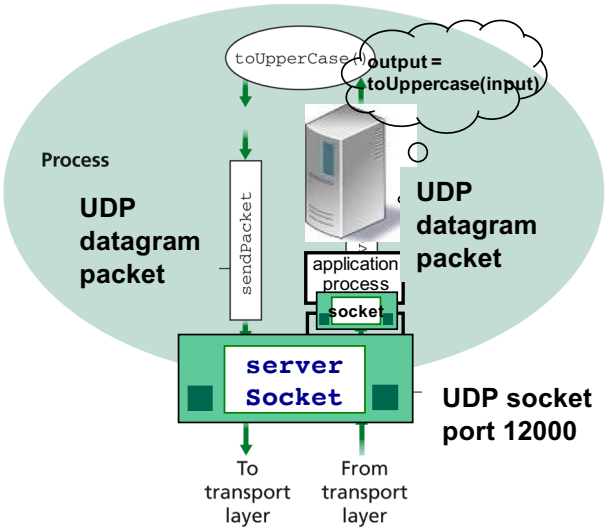
Create datagram with server IP addr and `port=x`; **send** datagram via `clientSocket`



read datagram from `clientSocket`

close
`clientSocket`

Example: Python UDP server



include Python's socket library

create UDP socket

bind socket to local port number 12000

loop forever

read from UDP socket into message, **getting client's address (client IP and port)**

send upper case string back to this client

```
from socket import *
serverPort = 12000

serverSocket = socket(AF_INET, SOCK_DGRAM)

serverSocket.bind(('', serverPort))

print "The server is ready to receive"

while 1:
    message, clientAddress = serverSocket.recvfrom(2048)
    modifiedMessage = message.upper()
    serverSocket.sendto(modifiedMessage, clientAddress)
```

http://docs.python.org/2/library/socket.html

Example: Python UDP client

include Python's socket library

```
from socket import  *
serverName = 'hostname'
serverPort = 12000
```

create UDP socket (SOCK_DGRAM) for server

```
clientSocket = socket(socket.AF_INET,
                      socket.SOCK_DGRAM)
```

get user keyboard input

```
message = raw_input('Input lowercase sentence:')
```

attach (server name, port) to message; send into socket

```
clientSocket.sendto(message,(serverName, serverPort))
```

read reply characters from socket into string

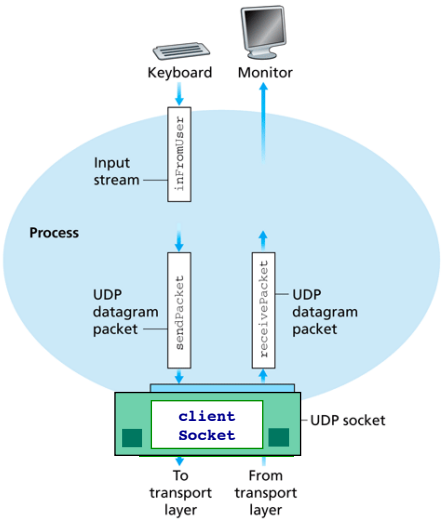
```
modifiedMessage, serverAddress = clientSocket.recvfrom(204
```

print out received string

```
print modifiedMessage
```

close socket

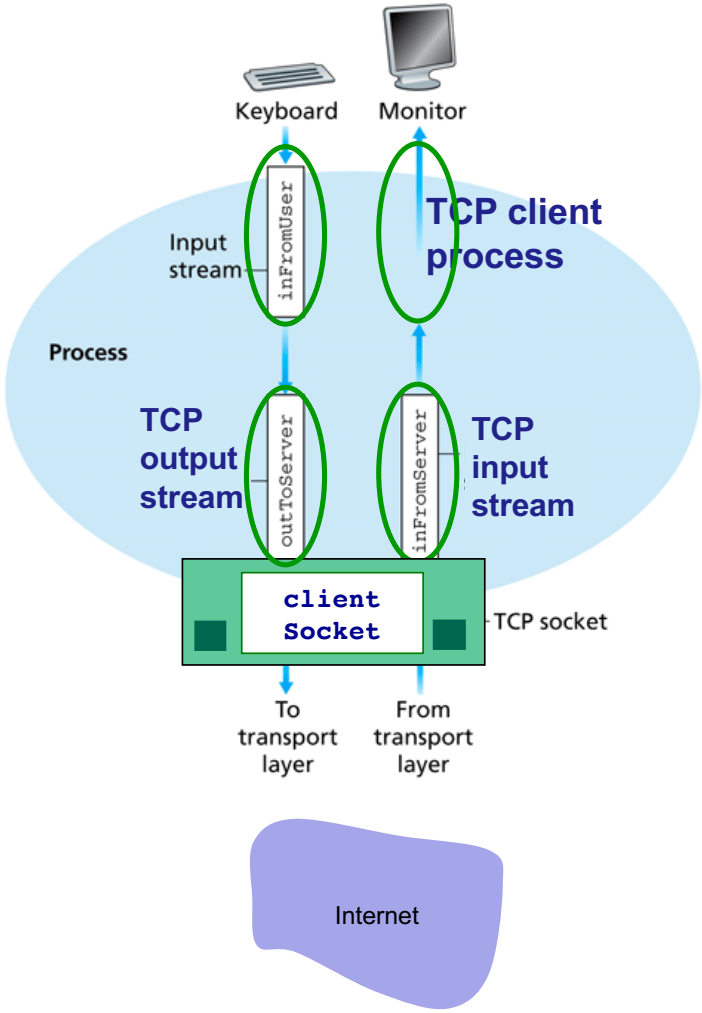
```
clientSocket.close()
```



http://docs.python.org/2/library/socket.html

CLIENT TCP

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

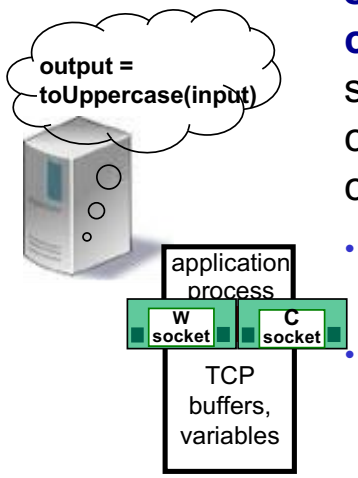


Socket programming with TCP

application viewpoint

TCP provides reliable, in-order transfer of bytes ("pipe") between client and server

- server listens on **welcoming socket**
- **server TCP creates new connection socket** for server process to communicate with contacting client
  - allows server to talk with multiple clients
  - source port numbers used to distinguish clients

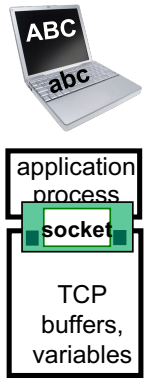


client must contact server

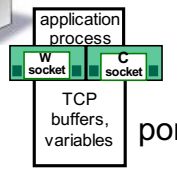
- server process must be running
- server must have created socket to welcome client's contact

client contacts server by

- creating TCP socket
- specifying IP address & port number of server process
- establishing TCP connection to server – connect



Client/server socket interaction: TCP



Server (running on `serverIP`)

Create socket, port=`X`, for incoming request:
WelcomeServerSocket = socket()

Wait for incoming connection request
connectionSocket = WelcomeServerSocket.accept()

Read request from **connectionSocket**

Write reply to **connectionSocket**

Close **connectionSocket**

TCP connection setup

TCP data

TCP data

client

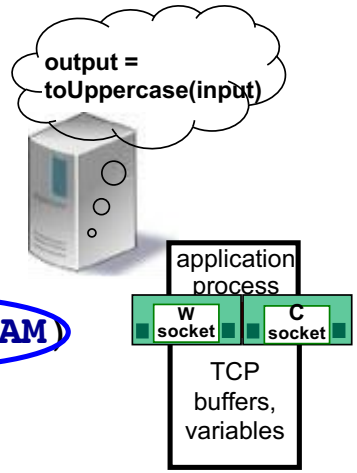
Create socket, connect to `serverIP`, port=`X`
clientSocket = socket()

Send request using **clientSocket**

Read reply from **clientSocket**

Close **clientSocket**

Example: Python TCP server



create TCP **welcoming** socket

server begins listening for incoming TCP requests

loop forever

server waits on accept() for incoming requests, new **socket created** on return

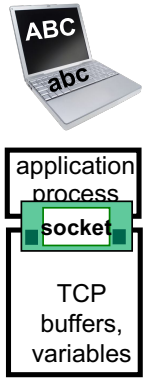
read bytes from socket (but not address as in UDP)

close connection to this client (but *not* welcoming socket)

```
from socket import *
serverPort = 1030
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('', serverPort))

serverSocket.listen(1)

print 'The server is ready to receive'

while 1:
    connectionSocket, addr = serverSocket.accept()

    sentence = connectionSocket.recv(1024)
    capitalizedSentence = sentence.upper()

    connectionSocket.send(capitalizedSentence)

    connectionSocket.close()
```

http://docs.python.org/2/library/socket.html

Example: Python TCP client



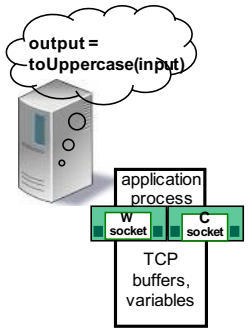
create TCP socket (SOCK_STREAM) for server, remote port 1030

No need to attach server name, port

```
from socket import *
serverName = 'servername'
serverPort = 1030
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = raw_input('Input lowercase sentence:')
clientSocket.send(sentence)
modifiedSentence = clientSocket.recv(1024)
print 'From Server:', modifiedSentence
clientSocket.close()
```

http://docs.python.org/2/library/socket.html

Socket methods to initialize connection



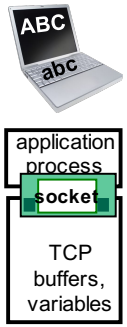
Server (running on 10.0.0.76)

```
serverPort = 1030
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('',serverPort))

serverSocket.listen(1)
```

```
connectionSocket, addr = serverSocket.accept()
```

TCP connection setup

Client (on 10.0.0.37)

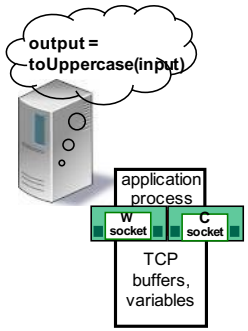



```
serverName = '10.0.0.76'
serverPort = 1030
clientSocket = socket(AF_INET, SOCK_STREAM)

clientSocket.connect((serverName,serverPort))
```

TCP connection setup

```
34 9.981273000  10.0.0.37  10.0.0.76   TCP   78 51211 > iad1 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=16 TSval=57082033 TSecr=0 SACK_PERM=1
35 9.984233000  10.0.0.76  10.0.0.37   TCP   78 iad1 > 51211 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=16 TSval=622449672 TSecr=57082033 SACK_PERM=1
36 9.984495000  10.0.0.37  10.0.0.76   TCP   66 51211 > iad1 [ACK] Seq=1 Ack=1 Win=131760 Len=0 TSval=57082062 TSecr=622449672
```

Socket methods to send and receive data

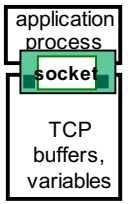


Server (running on 10.0.0.76)

```
Melding fra klient: Kan du oversette: En jeger gikk i skogen
Svar sendt til klient: KAN DU OVERSETTE: EN JEGER GIKK I SKOGEN
```

```
sentence = connectionSocket.recv(1024)

capitalizedSentence = sentence.upper()

connectionSocket.send(capitalizedSentence)
```

Client (on 10.0.0.37)




```
Client: Skriv din melding
Kan du oversette: En jeger gikk i skogen
Klient: Avventer svar fra server
```

```
sentence = raw_input('Input lowercase sentence:')
clientSocket.send(sentence)
```

TCP data

```
Svar mottatt fra server: KAN DU OVERSETTE: EN JEGER GIKK I SKOGEN
```

	From	To	Protocol	Length	PortS	PortD (iad1 = 1030)
42	22.126653000	10.0.0.37	10.0.0.76	TCP	67	51211 > iad1 [PSH, ACK] Seq=1 Ack=1 Win=131760 Len=1 TSval=57094164 TSecr=622449674
43	22.273157000	10.0.0.76	10.0.0.37	TCP	66	iad1 > 51211 [ACK] Seq=1 Ack=2 Win=131760 Len=0 TSval=622461927 TSecr=57094164
44	22.273391000	10.0.0.37	10.0.0.76	TCP	106	51211 > iad1 [PSH, ACK] Seq=2 Ack=1 Win=131760 Len=40 TSval=57094310 TSecr=622461927
45	22.275080000	10.0.0.76	10.0.0.37	TCP	66	iad1 > 51211 [ACK] Seq=1 Ack=42 Win=131712 Len=0 TSval=622461929 TSecr=57094310
46	22.275759000	10.0.0.76	10.0.0.37	TCP	67	iad1 > 51211 [PSH, ACK] Seq=1 Ack=42 Win=131712 Len=1 TSval=622461929 TSecr=57094310
47	22.276009000	10.0.0.37	10.0.0.76	TCP	66	51211 > iad1 [ACK] Seq=42 Ack=2 Win=131760 Len=0 TSval=57094311 TSecr=622461929
48	22.277655000	10.0.0.76	10.0.0.37	TCP	106	iad1 > 51211 [PSH, ACK] Seq=2 Ack=42 Win=131712 Len=40 TSval=622461931 TSecr=57094311
49	22.277710000	10.0.0.37	10.0.0.76	TCP	66	51211 > iad1 [ACK] Seq=42 Ack=42 Win=131712 Len=0 TSval=57094313 TSecr=622461931

Summary: Learned about protocols

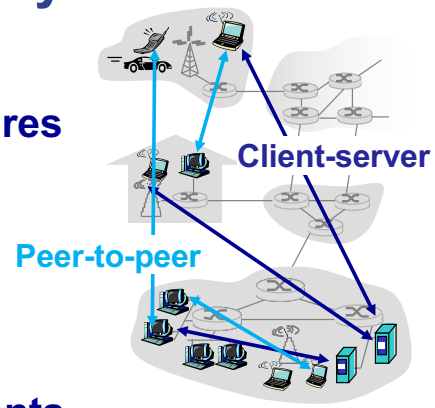
- Typical **request/reply** message exchange
  - client requests info or service
  - server responds with data, status code
- Message formats
  - **headers**: fields giving info about data
  - **data**: info being communicated

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”

Chapter 2: Summary

- **Application architectures**
  - client-server: web
  - peer-to-peer



- **Application requirements**
  - bandwidth, delay, loss rate

- **Socket programming**
  - UDP: unreliable, **datagram socket**
  - TCP: connection-oriented, reliable **stream socket**

- **Specific application protocols**:
  - Web - **HTTP**
  - Email - **SMTP**, POP, IMAP
  - Name service – **DNS**
  - **Security challenges**

Protocol	Length	Info
HTTP	872	GET /why-using-google-dns-opendns-is-a-bad-
HTTP	497	GET /css/future_branding.min.css HTTP/1.1
HTTP	271	HTTP/1.1 304 Not Modified
HTTP	459	GET /tag/js/gpt.js HTTP/1.1
HTTP	525	GET /j.php?a=29941&u=http%3A%2F%2Fapcmag.co
HTTP	256	HTTP/1.1 304 Not Modified
HTTP	326	HTTP/1.1 200 OK (application/x-javascript)
HTTP	1136	GET /gampad/ads?gdfp_req=1&correlator=20595

```
GET /dc.js HTTP/1.1
Host: stats.g.doubleclick.net
Accept: */*
Connection: keep-alive
Cookie: id=22d6327cd501001d||t=1389079506|et
User-Agent: Mozilla/5.0 (Macintosh; Intel Ma
Gecko) Version/7.1.3 Safari/537.85.12
Accept-Language: nb-no
Referer: http://tv.nrk.no/
Accept-Encoding: gzip, deflate

HTTP/1.1 200 OK
Date: Tue, 10 Mar 2015 18:37:05 GMT
Expires: Tue, 10 Mar 2015 20:37:05 GMT
Last-Modified: Thu, 05 Feb 2015 17:35:24 GMT
X-Content-Type-Options: nosniff
Content-Type: text/javascript
Vary: Accept-Encoding
Content-Encoding: gzip
Server: Golfe2
Content-Length: 15844
Cache-Control: public, max-age=7200
Age: 3390
Alternate-Protocol: 80:quic,p=0.08
```

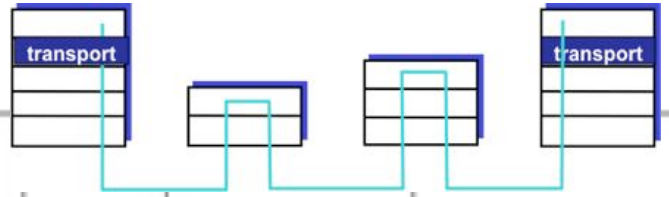
Protocol	Length	Info
SMTP	114	S: 220 samson.item.ntnu.no ESMTP Postfix (…
SMTP	77	C: HELO edda
SMTP	91	S: 250 samson.item.ntnu.no
SMTP	97	C: MAIL FROM: kjmoldek@online.no
SMTP	80	S: 250 2.1.0 Ok
SMTP	72	C: DATA
SMTP	104	S: 554 5.5.1 Error: no valid recipients
SMTP	98	C: DATA fragment, 32 bytes
SMTP	80	S: 250 2.1.5 Ok
SMTP	72	C: DATA fragment, 6 bytes
SMTP	103	S: 354 End data with <CR><LF>.<CR><LF>
SMTP	89	C: DATA fragment, 23 bytes
SMTP	84	C: DATA fragment, 18 bytes
SMTP	106	C: DATA fragment, 40 bytes
SMTP	77	C: DATA fragment, 11 bytes
IMF	69	from: Gjett hvem, subject: Dette er en SMT…
SMTP	103	S: 250 2.0.0 Ok: queued as A33602C00B7
SMTP	71	C: DATA fragment, 5 bytes
SMTP	107	S: 502 5.5.2 Error: command not recognized
SMTP	72	C: quit
SMTP	81	S: 221 2.0.0 Bye

```
▶ Ethernet II, Src: Sonicwal_10:51:f7 (00:17:c5:10:51:f7), Dst: Apple_e
▶ Internet Protocol Version 4, Src: 129.241.200.24, Dst: 10.0.0.103
▶ Transmission Control Protocol, Src Port: smtp (25), Dst Port: 49422 (
▶ Simple Mail Transfer Protocol
```

DNS	72	Standard query 0xe9f7 A www.tuaw.com
DNS	75	Standard query 0x0c43 A code.google.com
DNS	83	Standard query 0x0a18 A googleblog.blogspot.com
DNS	71	Standard query 0x41de A web.mac.com
DNS	352	Standard query response 0x0c43 A code.google.com CNAME cod
DNS	174	Standard query response 0x0a18 A googleblog.blogspot.com C
DNS	191	Standard query response 0xe9f7 A www.tuaw.com CNAME v6v4.w
DNS	167	Standard query response 0x41de No such name A web.mac.com

```
▼ Queries
  ▶ code.google.com: type A, class IN
▼ Answers
  ▶ code.google.com: type CNAME, class IN, cname code.l.google.com
  ▶ code.l.google.com: type A, class IN, addr 148.123.29.34
  ▶ code.l.google.com: type A, class IN, addr 148.123.29.49
  ▶ code.l.google.com: type A, class IN, addr 148.123.29.39
  ▶ code.l.google.com: type A, class IN, addr 148.123.29.55
  ▶ code.l.google.com: type A, class IN, addr 148.123.29.59
  ▶ code.l.google.com: type A, class IN, addr 148.123.29.40
  ▶ code.l.google.com: type A, class IN, addr 148.123.29.35
  ▶ code.l.google.com: type A, class IN, addr 148.123.29.20
  ▶ code.l.google.com: type A, class IN, addr 148.123.29.30
  ▶ code.l.google.com: type A, class IN, addr 148.123.29.44
  ▶ code.l.google.com: type A, class IN, addr 148.123.29.50
  ▶ code.l.google.com: type A, class IN, addr 148.123.29.24
  ▶ code.l.google.com: type A, class IN, addr 148.123.29.54
  ▶ code.l.google.com: type A, class IN, addr 148.123.29.45
  ▶ code.l.google.com: type A, class IN, addr 148.123.29.29
  ▶ code.l.google.com: type A, class IN, addr 148.123.29.25
```

Next week, 26-27 January



3	Thursday 12:15 – 14:00 Thursday 14:15 – 15:00	Application Layer Theory Assignment 1: *Overview of Computer Networks and the Internet* Wireshark Lab 1: *Intro (optional but highly recommended!)*	R1	Assistants/ Ida/Norvald	One must deliver and pass at least 5 of the 8 theory assignments.
3	Friday 09:15 – 11:00	Application Layer (cont)	R1	Kjersti	Chapter 2 Chapter 8.2-3 and 8.5.1
4	Thursday 12:15 – 14:00 Thursday 14:15 – 15:00	Transport Layer Theory Assignment 2: *Application Layer*	R1 R1	Kjersti Assistants/ Ida/Norvald	Chapter 3 One must deliver and pass at least 5 of the 8 theory assignments.
4	Friday 09:15 – 11:00	Transport Layer (cont)	R1	Kjersti	Chapter 3
5	Thursday 12:15 – 14:00	Transport Layer (cont)	R1	Kjersti	Chapter 3 Chapter 8.6
5	Friday 09:15 – 11:00	Network Layer	R1	Kjersti	Chapter 4

NTNU

From http://docs.python.org/2/library/socket.html

`socket.socket([family[,type[,proto]]])`

Create a new socket using the given address family, socket type and protocol number. The address family should be AF_INET (the default), AF_INET6 or AF_UNIX. The socket type should be SOCK_STREAM (the default), SOCK_DGRAM or perhaps one of the other SOCK_ constants. The protocol number is usually zero and may be omitted in that case.

`socket.sendto(string,address) socket.sendto(string,flags,address)`

Send data to the socket. The socket should not be connected to a remote socket, since the destination socket is specified by *address*. The optional *flags* argument has the same meaning as for recv() above. Return the number of bytes sent. (The format of *address* depends on the address family — see above.)

`socket.recvfrom(bufsize[,flags])`

Receive data from the socket. The return value is a pair (string, address) where *string* is a string representing the data received and *address* is the address of the socket sending the data. See the Unix manual page *recv(2)* for the meaning of the optional argument *flags*; it defaults to zero. (The format of *address* depends on the address family — see above.)

`socket.close()`

Close the socket. All future operations on the socket object will fail. The remote end will receive no more data (after queued data is flushed). Sockets are automatically closed when they are garbage-collected.

Note close() releases the resource associated with a connection but does not necessarily close the connection immediately. If you want to close the connection in a timely fashion, call shutdown() before close().