Information Network

Lecture 4 : Email

Holger Thies

KYOTO UNIVERSITY

京都大学

Today's lecture

2.2 HTTP infrastructure

2.3 electronic mail

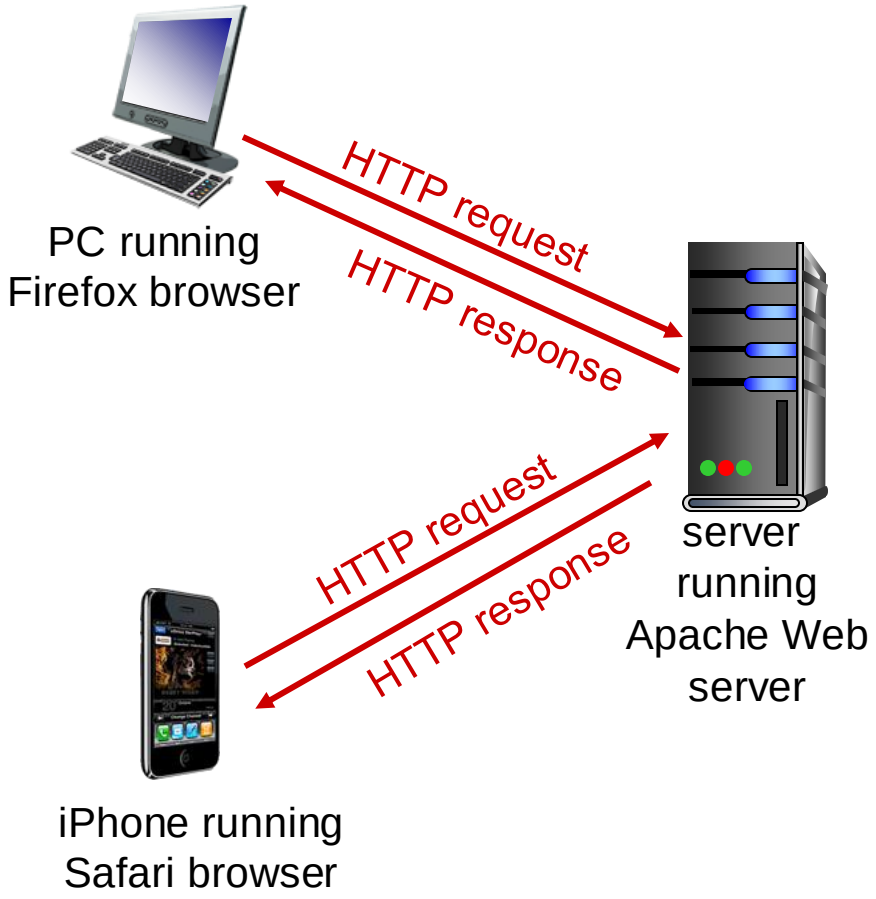
- SMTP, POP3, IMAP

2.4 DNS

HTTP overview

HTTP: hypertext transfer protocol

- Web's application layer protocol
- client/server model
  - *client:* browser that requests, receives, (using HTTP protocol) and "displays" Web objects
  - *server:* Web server sends (using HTTP protocol) objects in response to requests



京都大学

HTTP infrastructure

- Several technologies exist to enhance HTTP and to facilitate communication over the internet.
- Cookies are used to introduce state, e.g., to keep track of users etc.
- Web caches or proxy servers are used to store copies of frequently used content, reducing the load on servers and improving

Cookies

- HTTP is stateless.
  - Simplifies server design which is necessary to support thousands of simultaneous TCP connections.
- However, it is often desirable for a web site to identify users.
- HTTP uses **cookies** to allow sites to keep track of users.

What is a cookie?

- Cookies are used by web sites to remember information about the user.
- A cookie is a small piece of data that the web server sends to the user's web browser.
- The browser stores the data and sends it back to the server each time when the user accesses the same domain.
- Two types of cookies
  - Session Cookies: These types of browser cookies delete once the session ends.
  - Permanent Cookies: These types of browser cookies remain on the system and communicate with the server every time the website opens.

User-server state: cookies

many Web sites use cookies

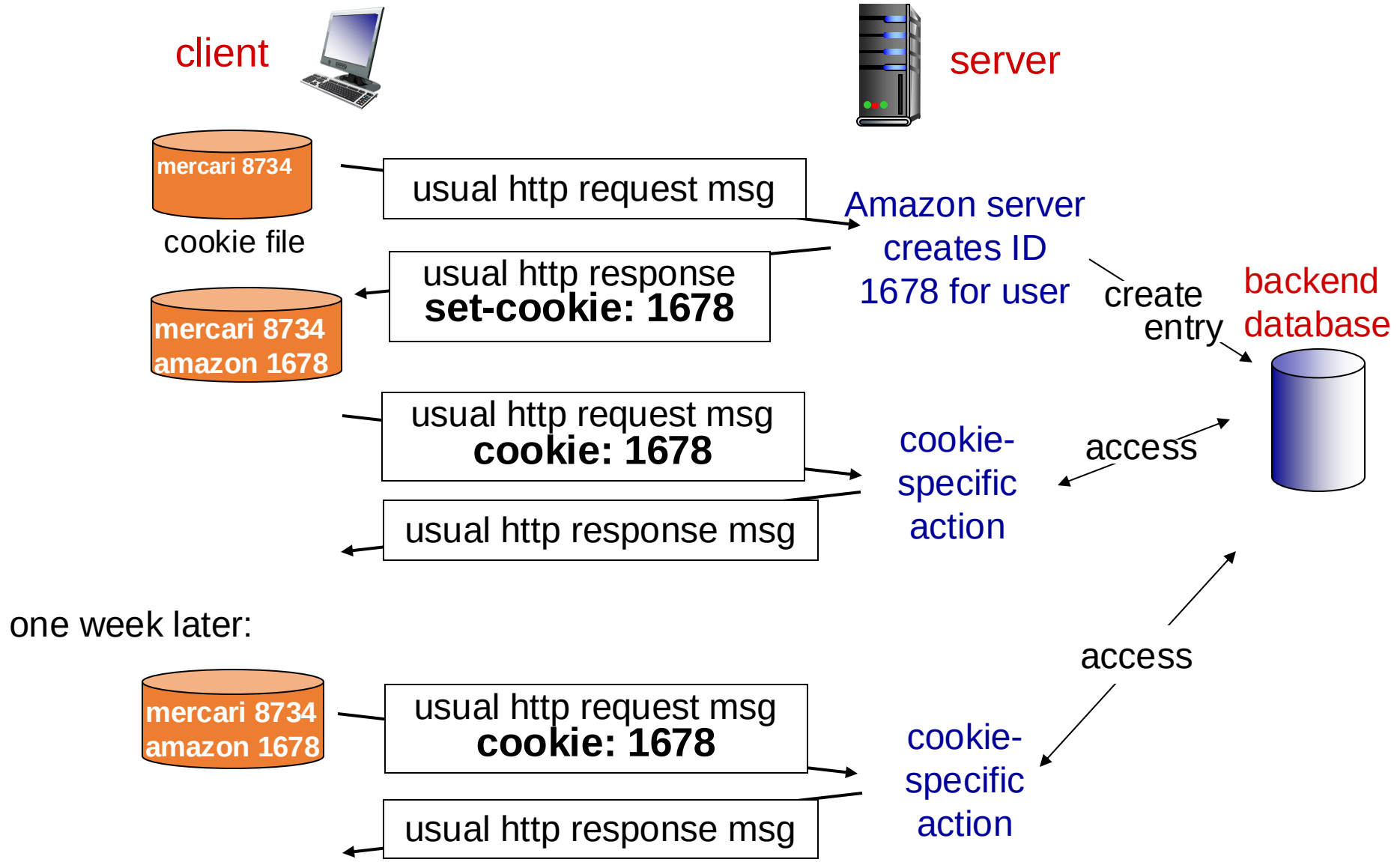
four components:

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

example:

- User always access Internet from PC
- visits specific e-commerce site for first time
- when initial HTTP requests arrives at site, site creates:
  - unique ID
  - entry in backend database for ID

Cookies: keeping "state" (cont.)



京都大学

Cookies (continued)

what cookies can be used for:

- authorization
- shopping carts
- recommendations
- user session state (Web e-mail)
- etc.

cookies and privacy:

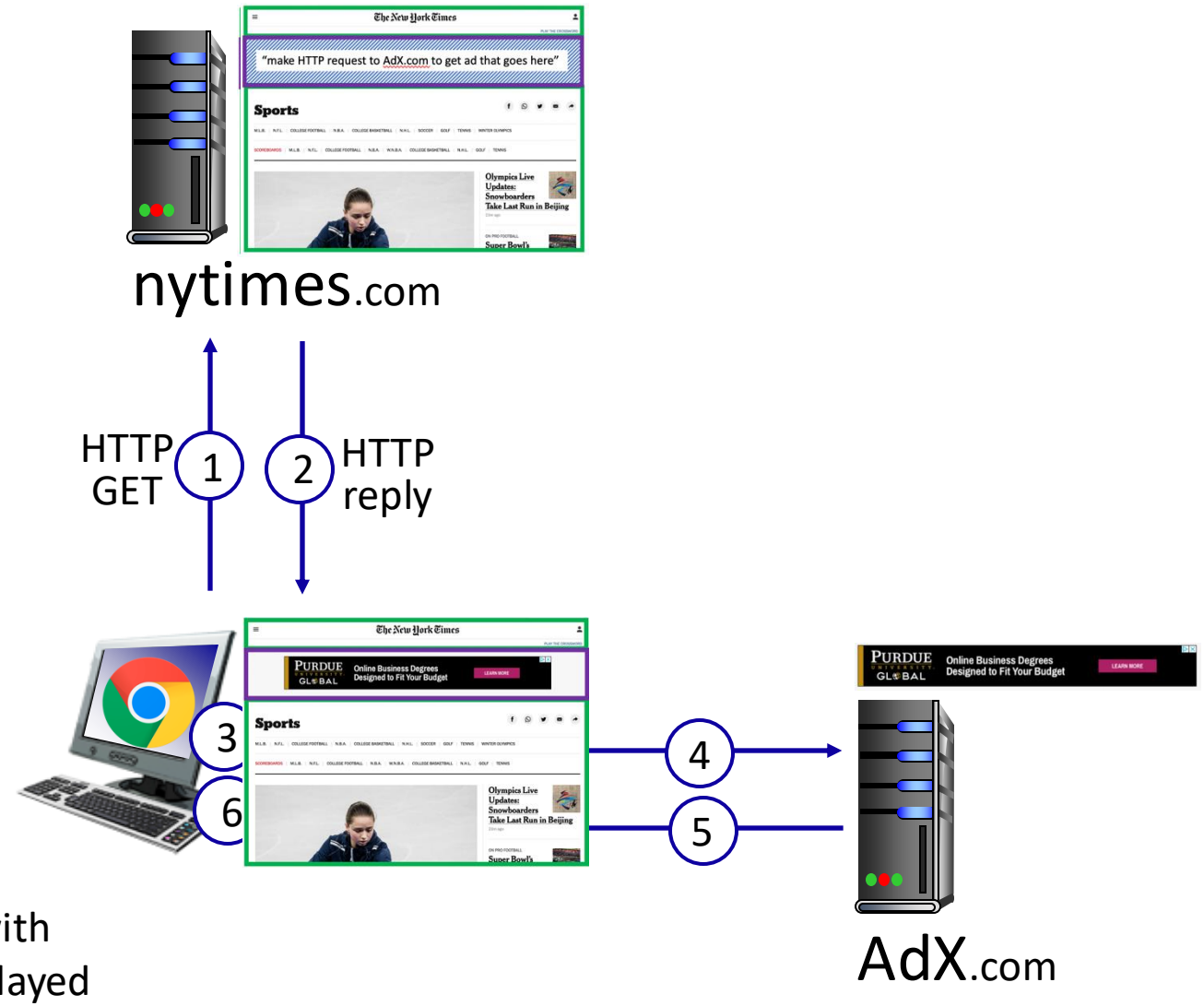
- Cookies are integral to web functionality, but they also raise privacy concerns.
- Cookies permit sites to learn a lot about you.
- They can track user behavior and collect personal information which is sometimes shared with third parties.
- They are used for profiling and targeted advertising.
- Regulations like GDPR have been introduced.

Example: displaying a NY Times web page

1 2 GET base html file from nytimes.com

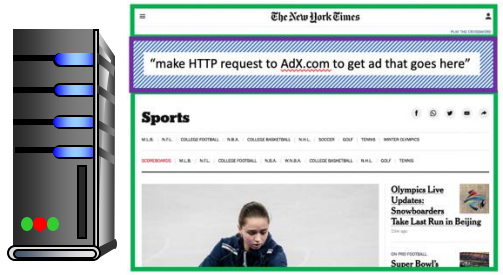
4 5 fetch ad from AdX.com

7 display composed page



NY times page with embedded ad displayed

Cookies: tracking a user's browsing behavior



nytimes.com (sports)

1634: sports, 2/15/22

"first party" cookie – from website you chose to visit (provides base html file)

HTTP GET

HTTP reply
Set cookie: 1634

"third party" cookie – from website you did not choose to visit

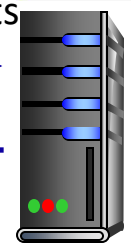
NY Times: 1634
AdX: 7493

HTTP GET
Referrer: NY Times Sports

4

5

HTTP reply
Set cookie: 7493

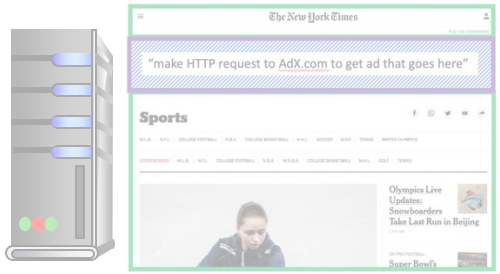


AdX.com

7493: NY Times sports, 2/15/22

Cookies: tracking a user's browsing behavior

socks.com



nytimes.com

HTTP reply

2

HTTP GET

1

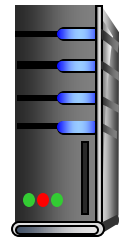
NY Times: 1634
AdX: 7493

HTTP GET
Referrer: socks.com, cookie: 7493

4

5

HTTP reply
Set cookie: 7493

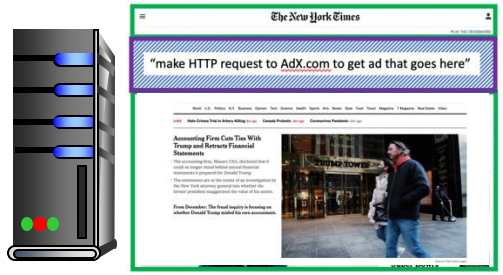


AdX.com

AdX:

- *tracks my web browsing* over sites with AdX ads
- can return targeted ads based on browsing history

Cookies: tracking a user's browsing behavior (one day later)



1634: sports, 2/15/22
1634: arts, 2/17/22

HTTP GET
cookie: 1634

HTTP reply
Set cookie: 1634

HTTP GET
Referrer: nytimes.com, cookie: 7493

4

5

HTTP reply
Set cookie: 7493
Returned ad for socks!

NY Times: 1634
AdX: 7493

AdX.com

7493: NY Times sports, 2/15/22
7493: socks.com, 2/16/22
7493: NY Times arts, 2/15/22

Cookies: tracking a user's browsing behavior

Cookies can be used to:

- track user behavior on a given website (first party cookies)
- track user behavior across multiple websites (third party cookies) without user ever choosing to visit tracker site (!)
- tracking may be *invisible* to user:
  - rather than displayed ad triggering HTTP GET to tracker, could be an invisible link

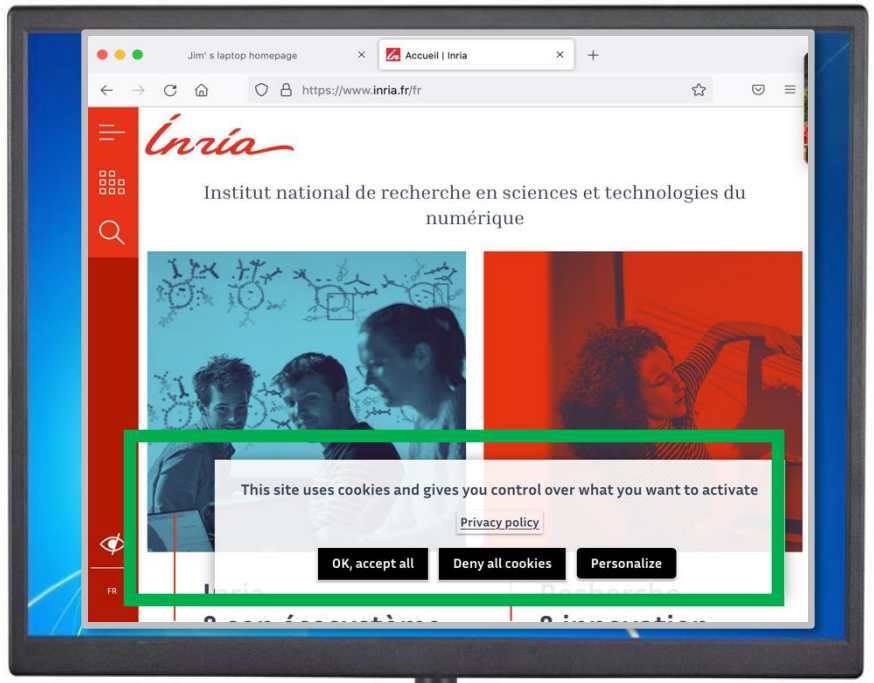
GDPR (EU General Data Protection Regulation) and cookies

"Natural persons may be associated with online identifiers [⋯] such as internet protocol addresses, cookie identifiers or other identifiers [⋯].

This may leave traces which, in particular when combined with unique identifiers and other information received by the servers, may be used to create profiles of the natural persons and identify them."

GDPR, recital 30 (May 2018)

when cookies can identify an individual, cookies are considered personal data, subject to GDPR personal data regulations



User has explicit control over whether or not cookies are allowed

- HTTP infrastructure
  - Cookies
  - Web caches

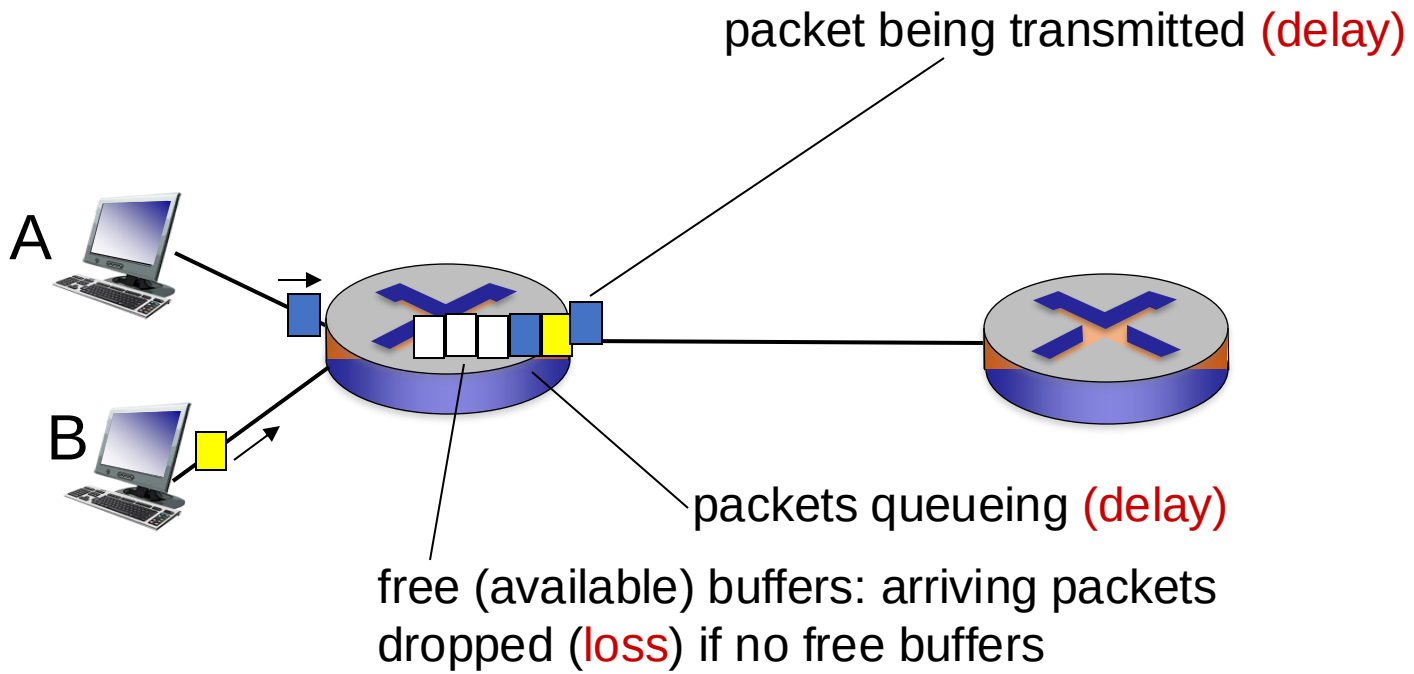
Delay and Loss

- Ideally, we would like internet services to move as much data as we want between any two end systems, instantaneously, without any loss of data.
- In reality, this can not be achieved.
- **Throughput** is the amount of data per second that can be transferred.
- In computer networks, throughput is constrained, there is delay between end systems and loss can occur.

How do loss and delay occur?

packets *queue* in router buffers

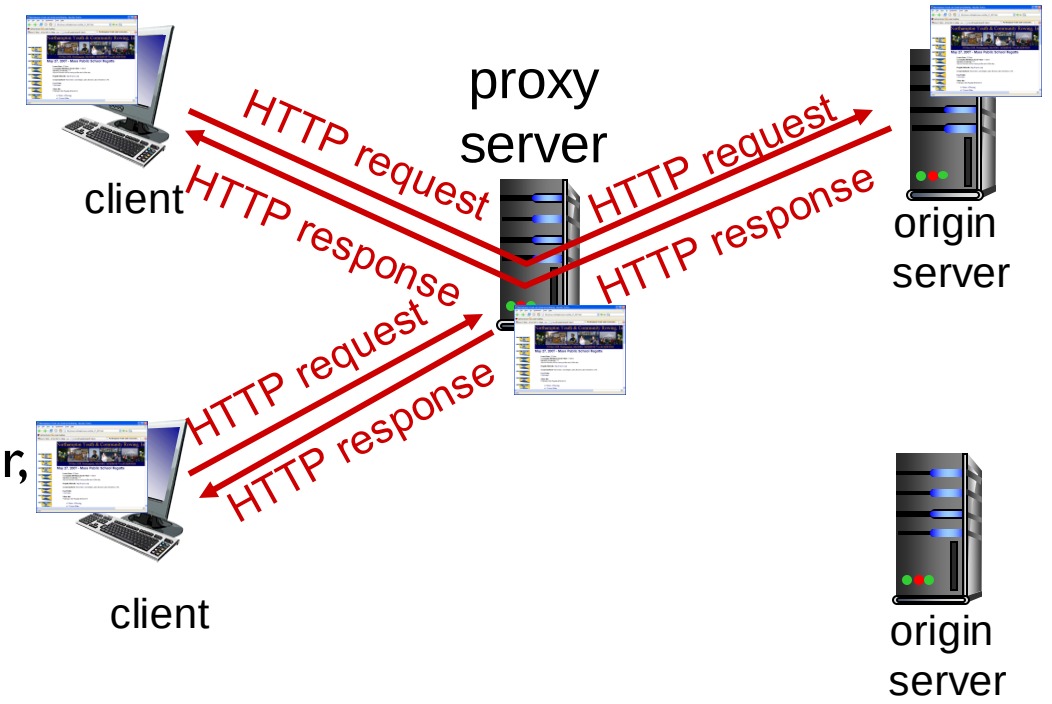
- packet arrival rate to link (temporarily) exceeds output link capacity
- packets queue, wait for turn



Web caches (proxy server)

goal: satisfy client request without involving origin server

- user sets browser: Web accesses via cache
- browser sends all HTTP requests to cache
  - object in cache: cache returns object
  - else cache requests object from origin server, then returns object to client



京都大学

Web cache example

Suppose browser is requesting http://www.somepage.jp/image.gif

1. The browser establishes a TCP connection to the Web cache.
2. The browser sends a HTTP request for the object to the web cache.
3. The web cache checks to see if a copy of the object is stored locally.
4. If the web cache does have the object, it returns the object in a HTTP response message.
5. If the web cache does not have the object, it opens a TCP connection to the origin server http://www.somepage.jp.
6. The web cache receives the object from the origin server.
7. The web cache stores a copy of the object in its local storage and sends a copy of the response message to the client.

More about Web caching

- cache acts as both client and server
  - server for original requesting client
  - client to origin server
- typically cache is installed by university, company, etc.

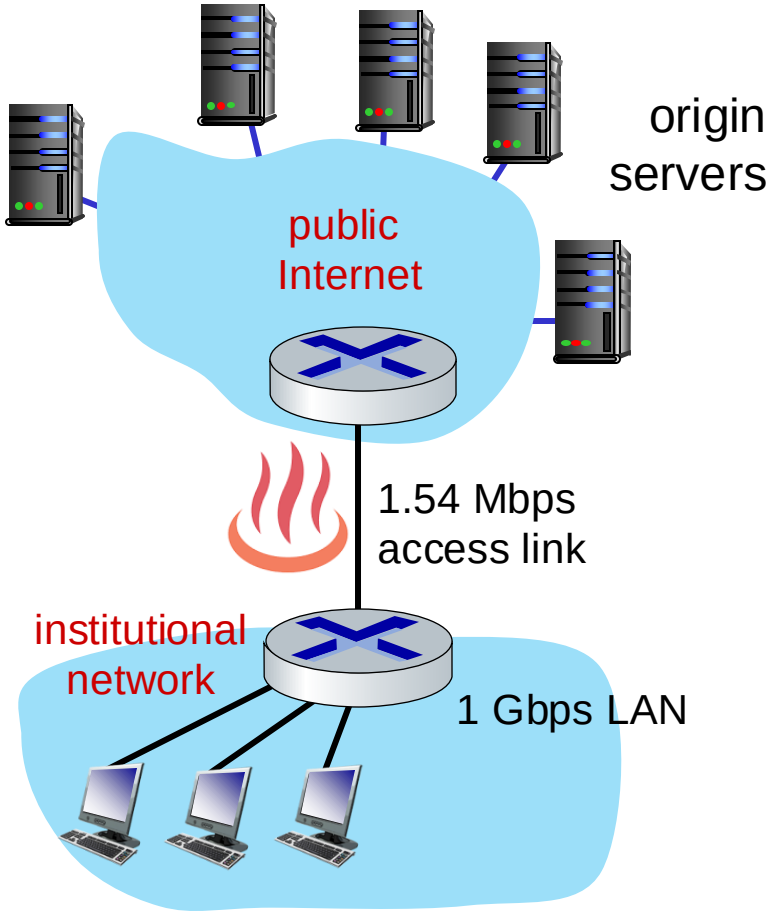
why Web caching?

- reduce response time for client request
- reduce traffic on an institution's access link
- Internet dense with caches: enables "poor" content providers to effectively deliver content

Caching example

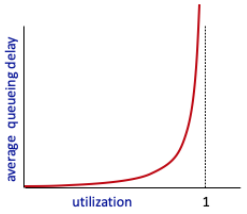
Scenario:

- access link rate: 1.54 Mbps
- RTT from institutional router to server: 2 sec
- web object size: 100K bits
- average request rate from browsers to origin servers: 15/sec
  - avg data rate to browsers: 1.50 Mbps



Performance:

- access link utilization = 0.97
- LAN utilization: 0.0015
- end-end delay = Internet delay + access link delay + LAN delay
  = 2 sec + minutes + usecs



problem: large queueing delays at high utilization!

京都大学

Option 1: buy a faster access link

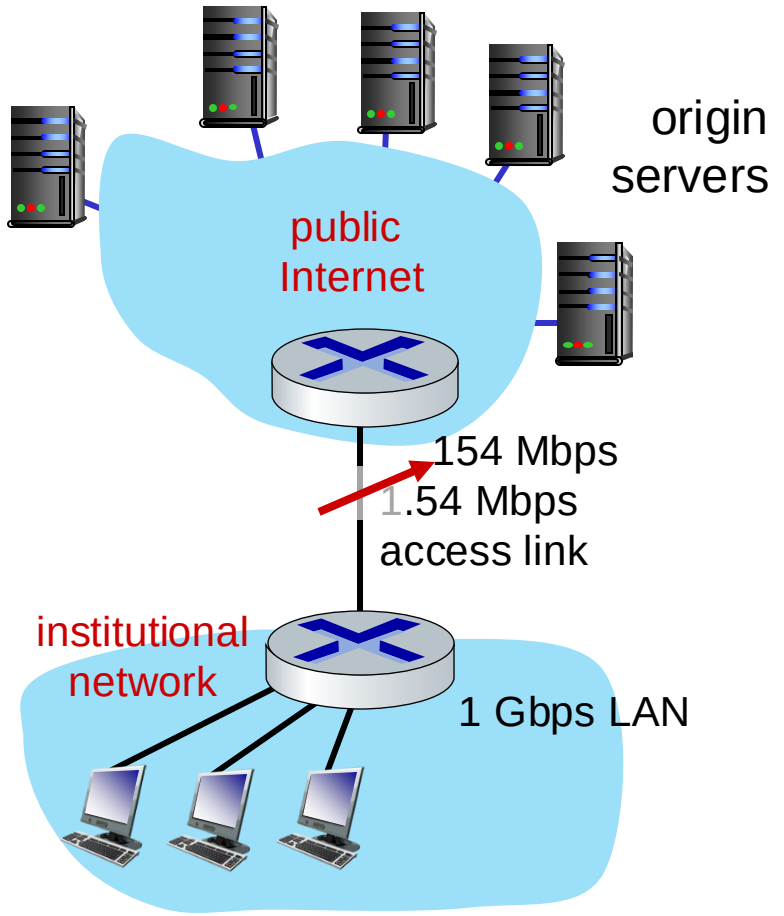
Scenario:

- access link rate: 1.54 Mbps → 154 Mbps
- RTT from institutional router to server: 2 sec
- web object size: 100K bits
- average request rate from browsers to origin servers: 15/sec
  - avg data rate to browsers: 1.50 Mbps

Performance:

- access link utilization = .97 → .0097
- LAN utilization: .0015
- end-end delay = Internet delay + access link delay + LAN delay
  = 2 sec + minutes + usecs → msecs

Cost: faster access link (expensive!)



京都大学

Option 2: install a web cache

Scenario:

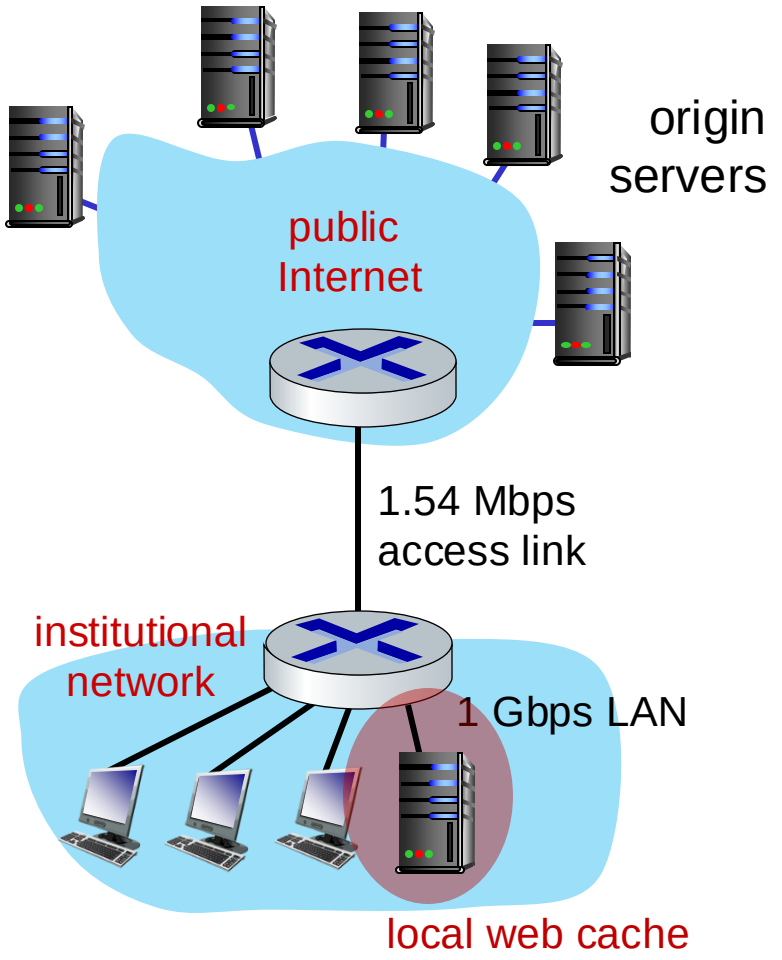
- access link rate: 1.54 Mbps
- RTT from institutional router to server: 2 sec
- web object size: 100K bits
- average request rate from browsers to origin servers: 15/sec
  - avg data rate to browsers: 1.50 Mbps

Cost: web cache (cheap!)

Performance:

- LAN utilization: .?
- access link utilization = ?
- average end-end delay = ?

How to compute link utilization, delay?



京都大学

Calculating access link utilization, end-end delay with cache:

suppose cache hit rate is 0.4:

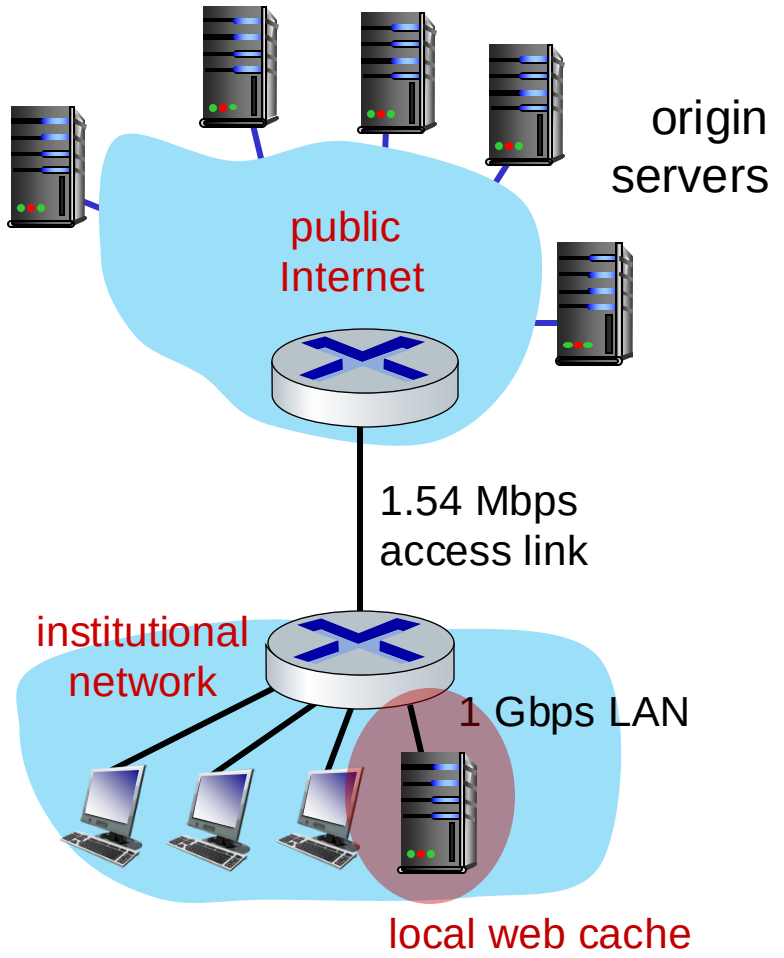
- 40% requests served by cache, with low (msec) delay
- 60% requests satisfied at origin
  - rate to browsers over access link

    = 0.6 * 1.50 Mbps = .9 Mbps
  - access link utilization = 0.9/1.54 = .58 means low (msec) queueing delay at access link
- average end-end delay:

  = 0.6 * (delay from origin servers)

  + 0.4 * (delay when satisfied at cache)

  = 0.6 (2.01) + 0.4 (~msecs) = ~ 1.2 secs



lower average end-end delay than with 154 Mbps link (and cheaper too!)

京都大学

Browser caching: Conditional GET

Goal: don't send object if browser has up-to-date cached version

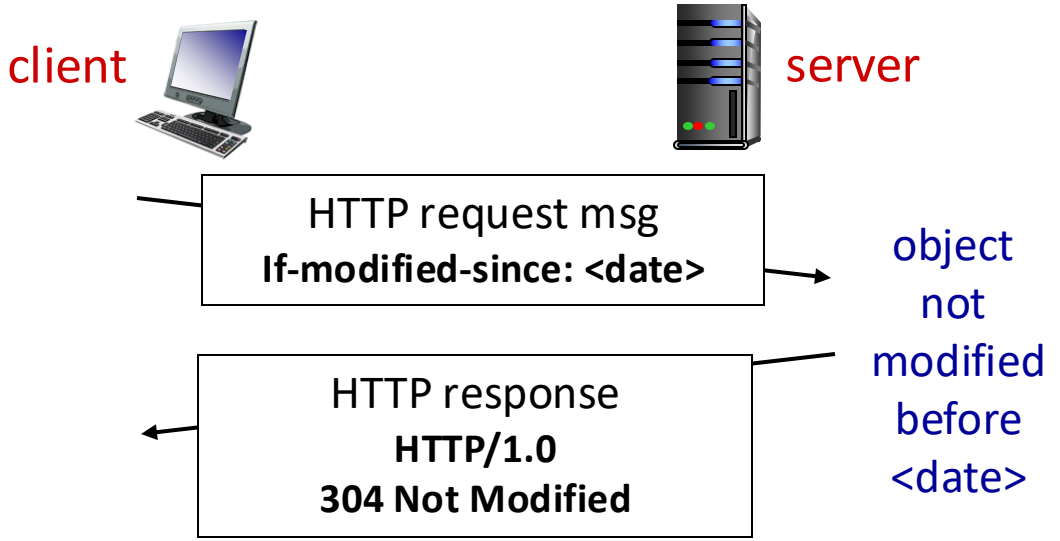
- no object transmission delay (or use of network resources)

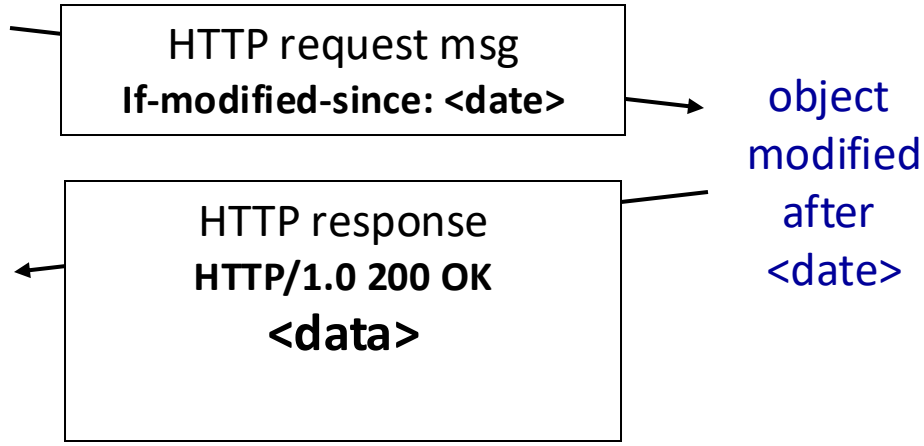
- *client:* specify date of browser-cached copy in HTTP request

  If-modified-since: <date>

- *server:* response contains no object if browser-cached copy is up-to-date:

  HTTP/1.0 304 Not Modified





Today's lecture

2.2 Web and HTTP

2.3 electronic mail

- SMTP, POP3, IMAP

2.4 DNS

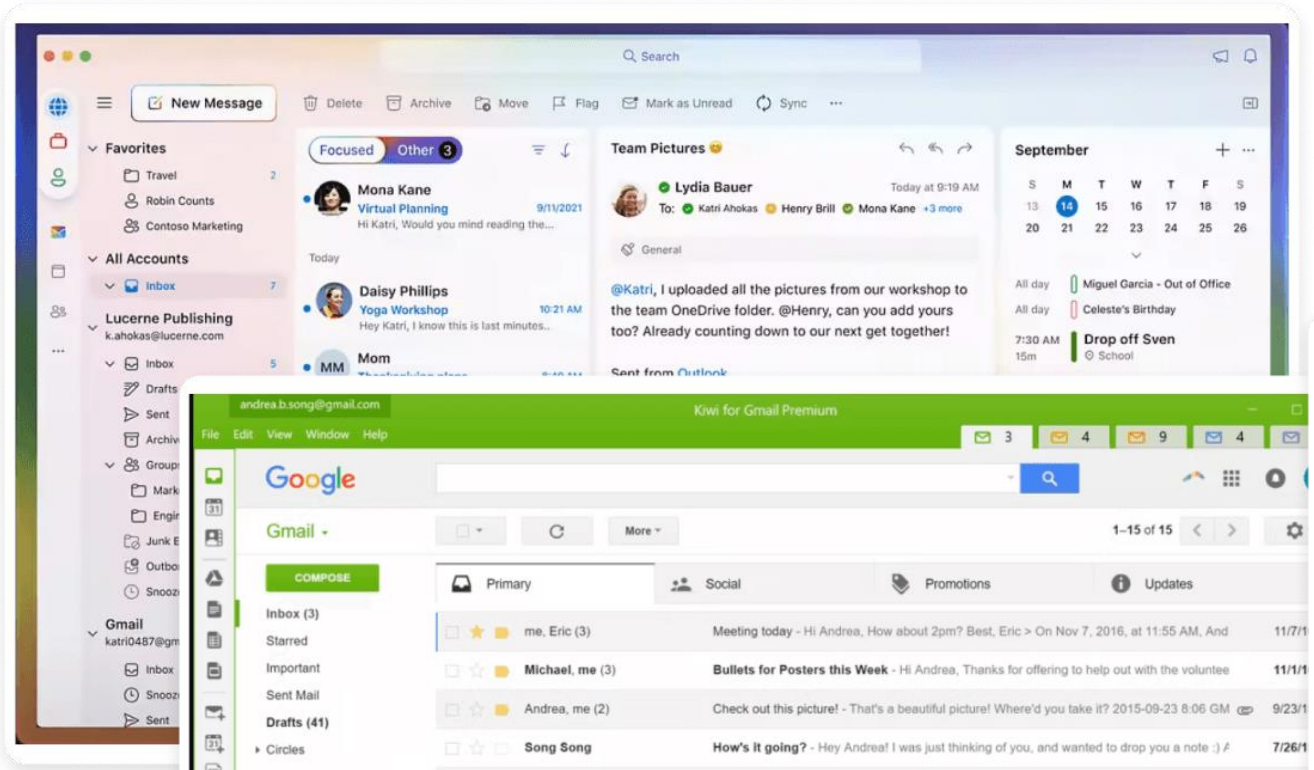
Electronic mail

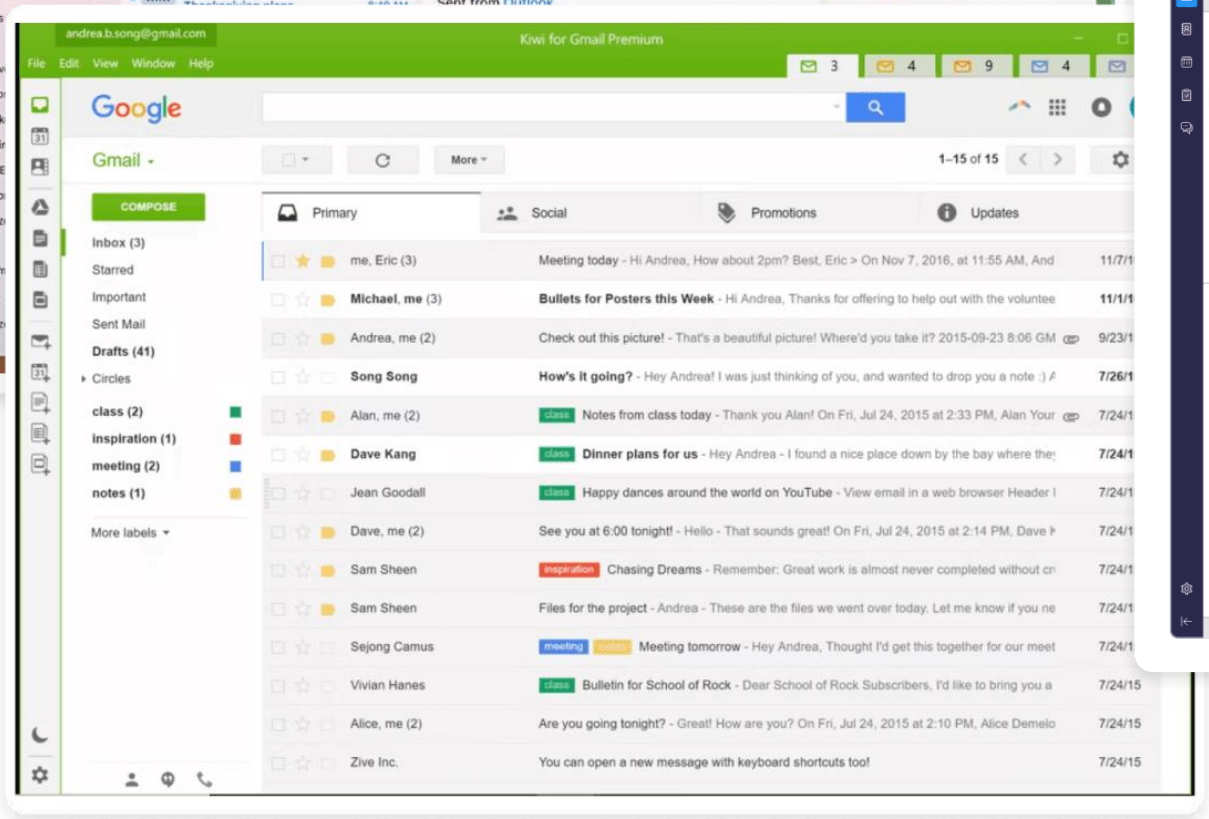
- Electronic mail (email) is used to send and receive messages over the Internet.
- One of the earliest and most important applications of the Internet (first email sent in 1971).
- Asynchronous: People send and read emails when it is convenient for them.
- Users create emails using email client software such as Outlook, Gmail, or Apple Mail.
- Emails are then sent to the sender's outgoing mailserver and forwarded to the recipient's mail server.
- When the recipient wants to access the email, the recipient's email client software fetches it from the mail server.

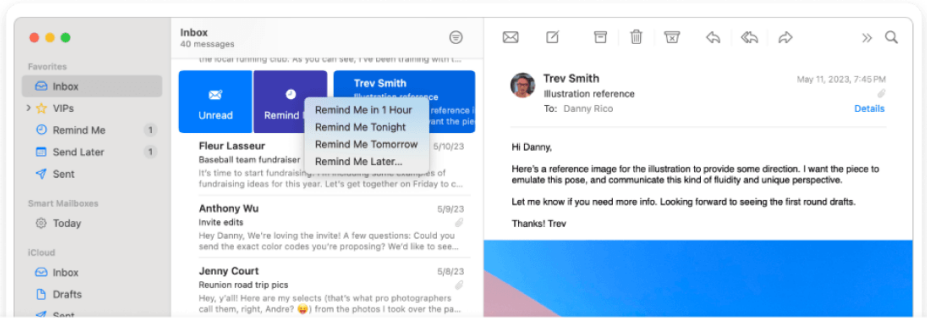
Mail servers and Providers

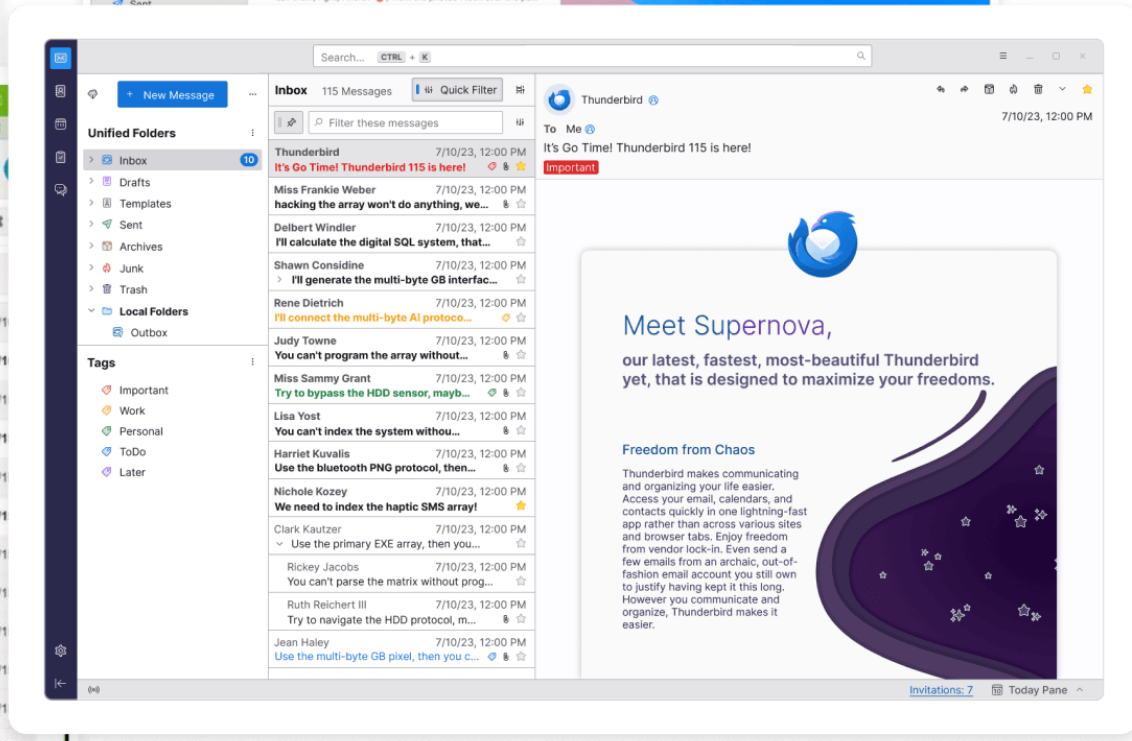
- Email uses a client-server architecture.
- Mail Servers are typically operated by an email provider.
  - The provider offers an email address and email services like sending, receiving and storing emails.
- In the past, email services were typically provided by the internet service provider (AOL, NTT, Softbank, etc.).
  - Example email addresses: username@aol.com, username@softbank.ne.jp, username@ocn.ne.jp
- Today, most users choose their email provider and set up their email accounts by themselves.
- Popular email providers include Gmail, Outlook, Yahoo Mail, etc.
- Institutions like universities and companies often host their own email servers.
  - name@st.kyoto-u.ac.jp, etc.

Mail clients (User agent)



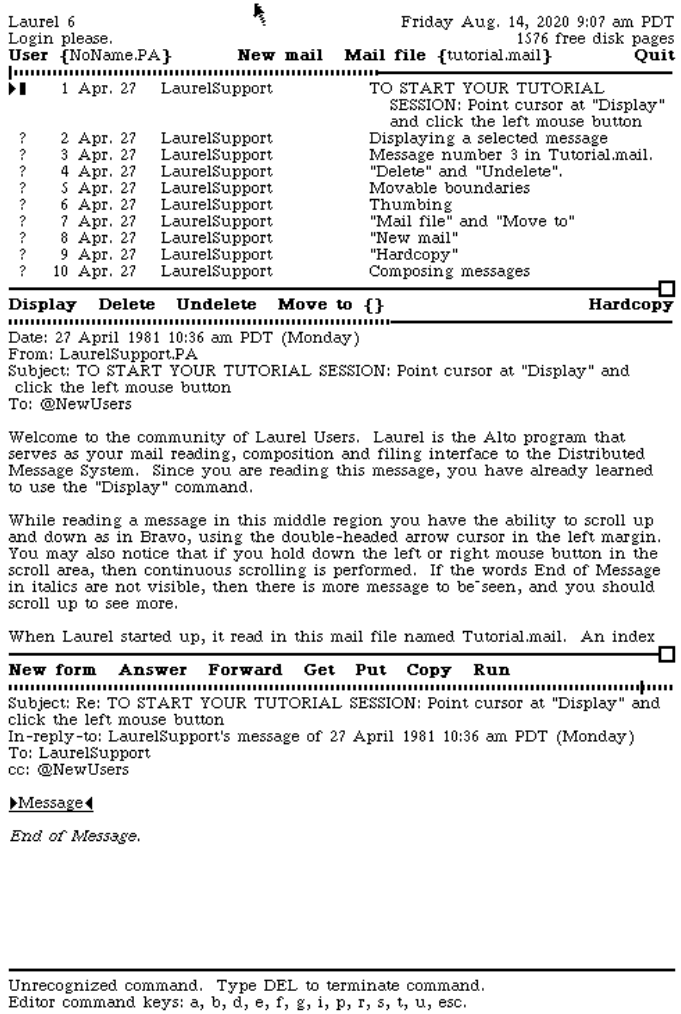


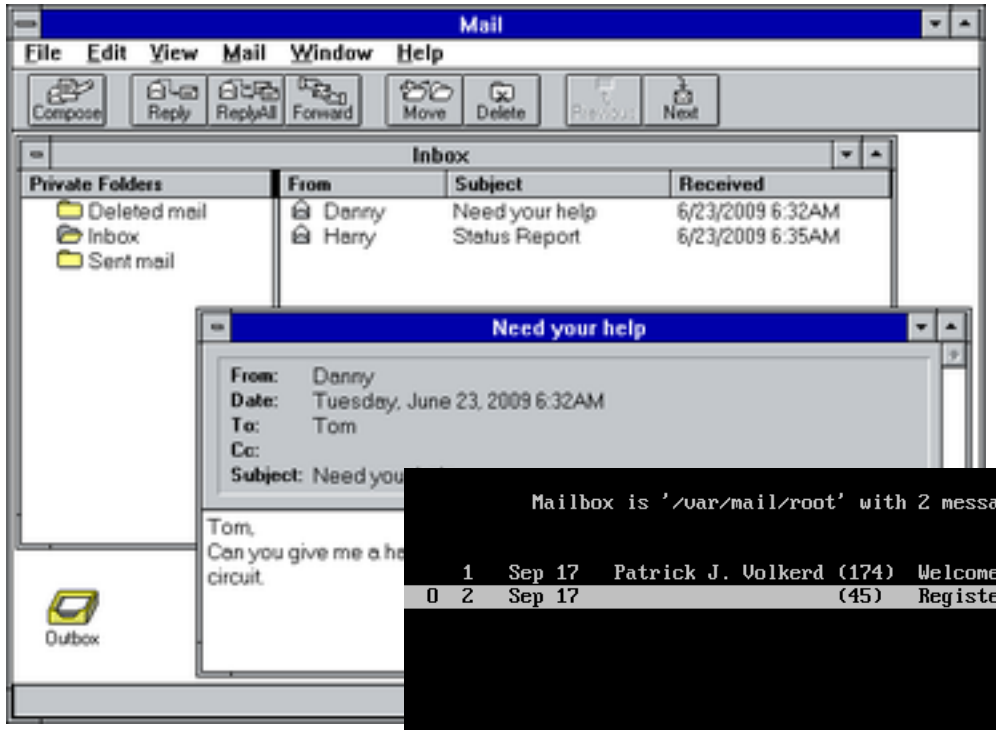


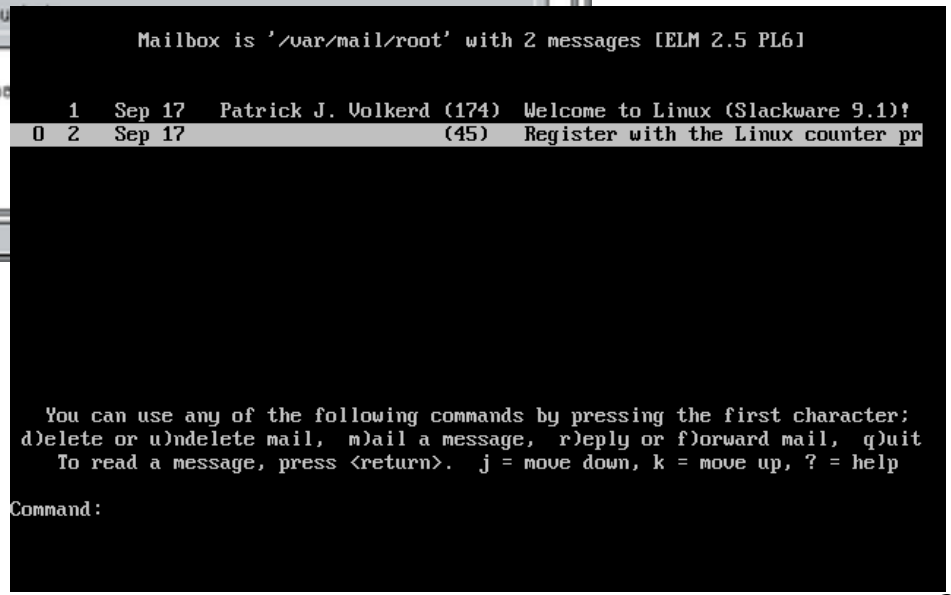


京都大学

Mail clients (User agent)







京都大学

Email protocols

Different protocols are used for sending and receiving email.

Protocols for Sending Emails

- SMTP (Simple Mail Transfer Protocol)
  - The primary protocol used to send emails.
  - Transfers email from the sender's client to the outgoing mail server.
  - Also used to transfer emails between mail servers.

Protocols for Receiving Emails

- IMAP (Internet Message Access Protocol)
  - Allows email access from multiple devices.
  - Emails stay on the server, syncing across devices.
- POP3 (Post Office Protocol 3)
  - Downloads emails to a local device.
  - Typically removes them from the server after download.

Use in email software

We recommend using a web browser for student emails, but you can also read emails with general email software POP/IMAP/Exchange.

In your email software, please make the following settings for receiving and sending emails. If you can't connect with POP/IMAP OAuth2 (advanced authentication), please use the Exchange connection described at the bottom of this page.
note: You cannot connect to and send/receive from Gmail because it does not support these authentication methods.

OAuth2 (advanced authentication)

	Incoming	Outgoing
Protocol	IMAP or POP	SMTP
Server	outlook.office365.com	smtp.office365.com
Port	993(IMAP)/995(POP)	587
SSL	SSL/TLS	STARTTLS
Authentification	OAuth2	
User Name	your.name.a12@st.kyoto-u.ac.jp Your mail address	

Email address format

An Email address identifies an email box to which messages are delivered.

Examples:

thies.holger.5c@kyoto-u.ac.jp
example@gmail.com
long.email-address-with-hyphens@and.subdomains.example.com

"very.(),:;<>[]¥".VERY.¥"very@¥¥ ¥"very¥".unusual"@strange.example.com

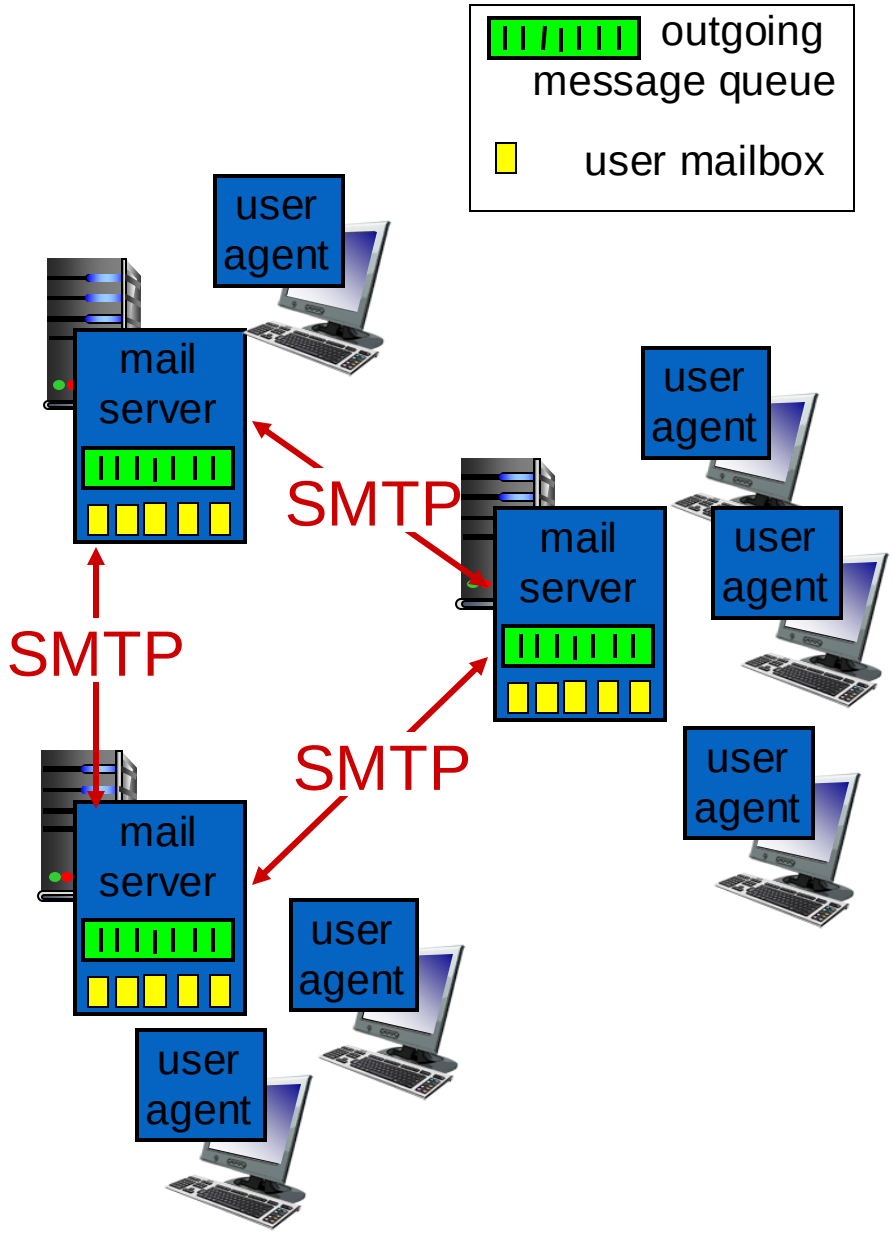
Electronic mail

Three major components:

- user agents
- mail servers
- simple mail transfer protocol: SMTP

User Agent

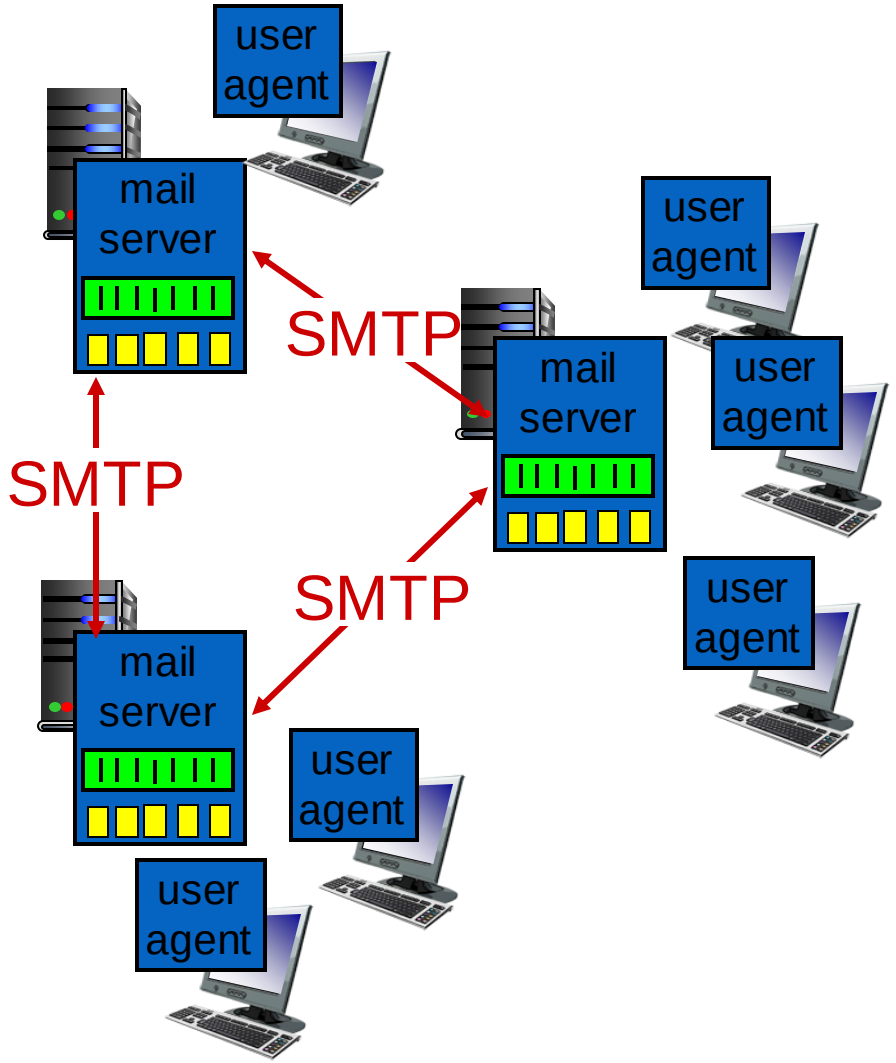
- "mail reader"
- composing, editing, reading mail messages
- e.g., Outlook, Thunderbird, iPhone mail client
- outgoing, incoming messages stored on server
- Mail access protocols: POP3, IMAP



Electronic mail: mail servers

mail servers:

- *mailbox* contains incoming messages for user
- *message queue* of outgoing (to be sent) mail messages
- *SMTP protocol* between mail servers to send email messages
  - client: sending mail server
  - "server": receiving mail server



京都大学

The SMTP protocol

- SMTP stands for Simple Mail Transfer Protocol.
- It is used to send emails from an email client to a mail server and to send emails between mail servers.
- It uses TCP as transport layer protocol.
- Three phases of transfer
  - opening the connection
  - transfer of messages
  - closure
- command/response interaction (like HTTP)
  - commands: human readable text
  - response: status code and phrase

Scenario: Alice sends message to Bob

1) Alice uses user agent to compose message "to" `bob@someschool.edu`

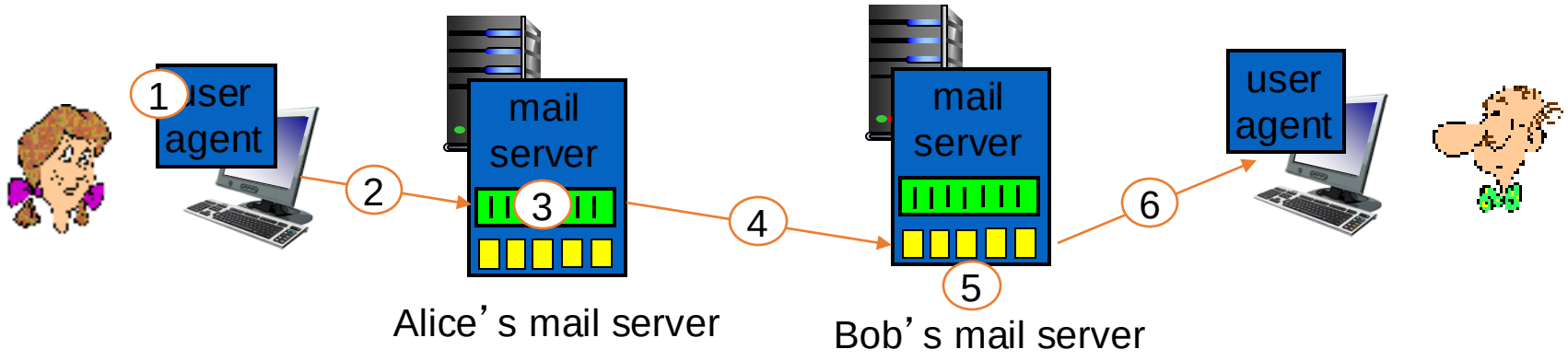
2) Alice's user agent sends message to her mail server; message placed in message queue

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



京都大学

Sample SMTP interaction

```
S: 220 sample.com  SMTP server ready
C: HELO mydomain.com
S: 250 Hello mydomain.com
C: MAIL FROM:<sender@mydomain.com>
S: 250 Ok
C: RCPT TO:<recipient@anotherdomain.com>
S: 250 Accepted
C: DATA
S: 354 Enter message, ending with "." on a line by itself
C: Subject: sample message
C: From: sender@mydomain.com
C: To: recipient@anotherdomain.com
C:
C: Greetings,
C: Typed message (content)
C: Goodbye.
C: .
S: 250 OK
C: QUIT
S: 221 www.sample.com closing connection
```

Sample SMTP Session (Initialization)

1. Connection Establishment

- Server: 220 sample.com SMTP Server Ready

2. HELO Command

- Client: HELO mydomain.com
- Server: 250 Hello mydomain.com

Sample SMTP Session (Sending Email)

3. MAIL FROM Command

- Client: MAIL FROM:<sender@mydomain.com>
- Server: 250 OK

4. RCPT TO Command

- Client: RCPT TO:<recipient@anotherdomain.com>
- Server: 250 OK

Sample SMTP Session (Sending Email)

5. DATA Command

- Client: DATA
- Server: 354 Start mail input; end with .
- Client: Sends email content, ends with '.' on a new line
- Server: 250 Message accepted for delivery

Sample SMTP Session (Ending Connection)

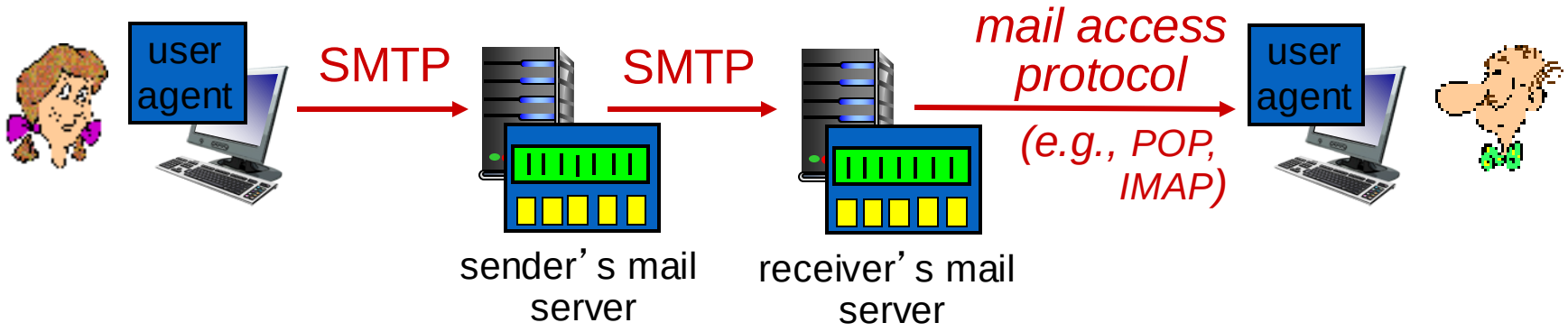
6. QUIT Command

- Client: QUIT
- Server: 221 www.sample.com closing connection

SMTP comparison with HTTP

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

Mail access protocols



- SMTP: delivery/storage to receiver's server
- mail access protocol: retrieval from server
  - POP: Post Office Protocol [RFC 1939]: authorization, download
  - IMAP: Internet Mail Access Protocol [RFC 1730]: more features, including manipulation of stored messages on server
  - HTTP: gmail, Hotmail, Yahoo! Mail, etc.

POP3

- Simple mail protocol
- Client opens TCP connection to mail server (port 110)
- Three phases
  - Authorization: Send username and password
  - Transaction: Retrieve messages, mark messages for deletion, etc.
  - Update: Mail server deletes messages marked for deletion
- Client issues commands and server responds with reply

POP3 protocol

authorization phase

- client commands:
  - `user:` declare username
  - `pass:` password
- server responses
  - `+OK`
  - `-ERR`

transaction phase, client:

- `list:` list message numbers
- `retr:` retrieve message by number
- `dele:` delete
- `quit`

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

```
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 2 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

POP3 (more)

more about POP3

- previous example uses POP3 "download and delete" mode
  - Bob cannot re-read e-mail if he changes client
- POP3 "download-and-keep": copies of messages on different clients
- POP3 is stateless across sessions

Limitations of POP3

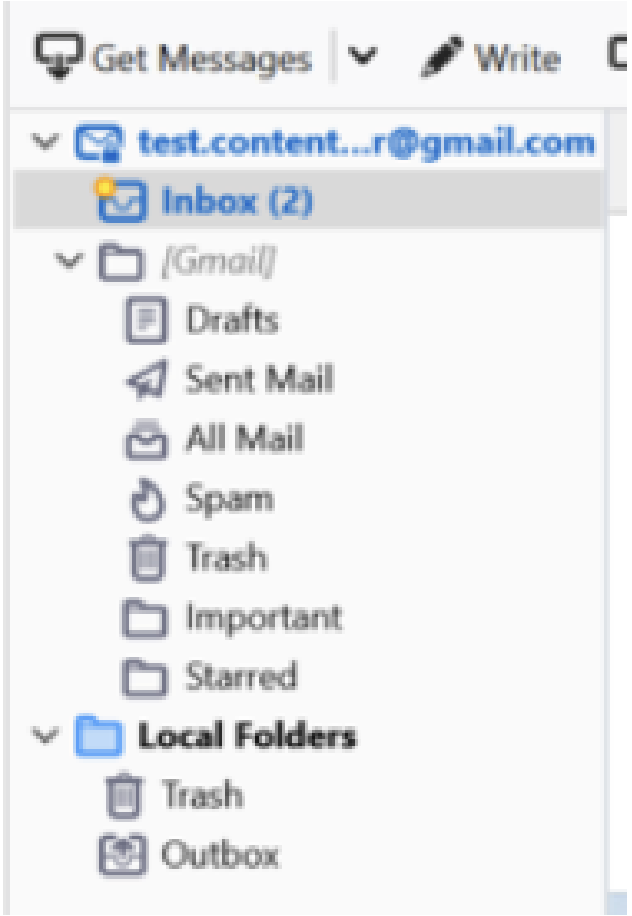
- Does not handle multiple mailboxes easily
  - Designed to put user's incoming e-mail in one folder
- Not designed to keep messages on the server
  - Instead, designed to download messages to the client
- Poor handling of multiple-client access to mailbox
  - Increasingly important as users have home PC, work PC, laptop, smartphone, etc.
- High network bandwidth overhead
  - Transfers all of the e-mail messages, often well before they are read (and they might not be read at all!)

Internet Message Access Protocol (IMAP)

- Main difference: IMAP stores the data in the remote server instead of downloading
  - Users can download message contents on demand
- Multiple clients can connect to mailbox at once
  - Detects changes made to the mailbox by other clients
  - Server keeps state about message (e.g., read, replied to)
- Access to parts of messages & partial fetch
  - Clients can retrieve individual parts separately
  - E.g., text of a message without downloading attachments
- Multiple mailboxes on the server
  - Client can create, rename, and delete mailboxes
  - Client can move messages from one folder to another
- Server-side searches
  - Search on server before downloading messages

IMAP Mailboxes

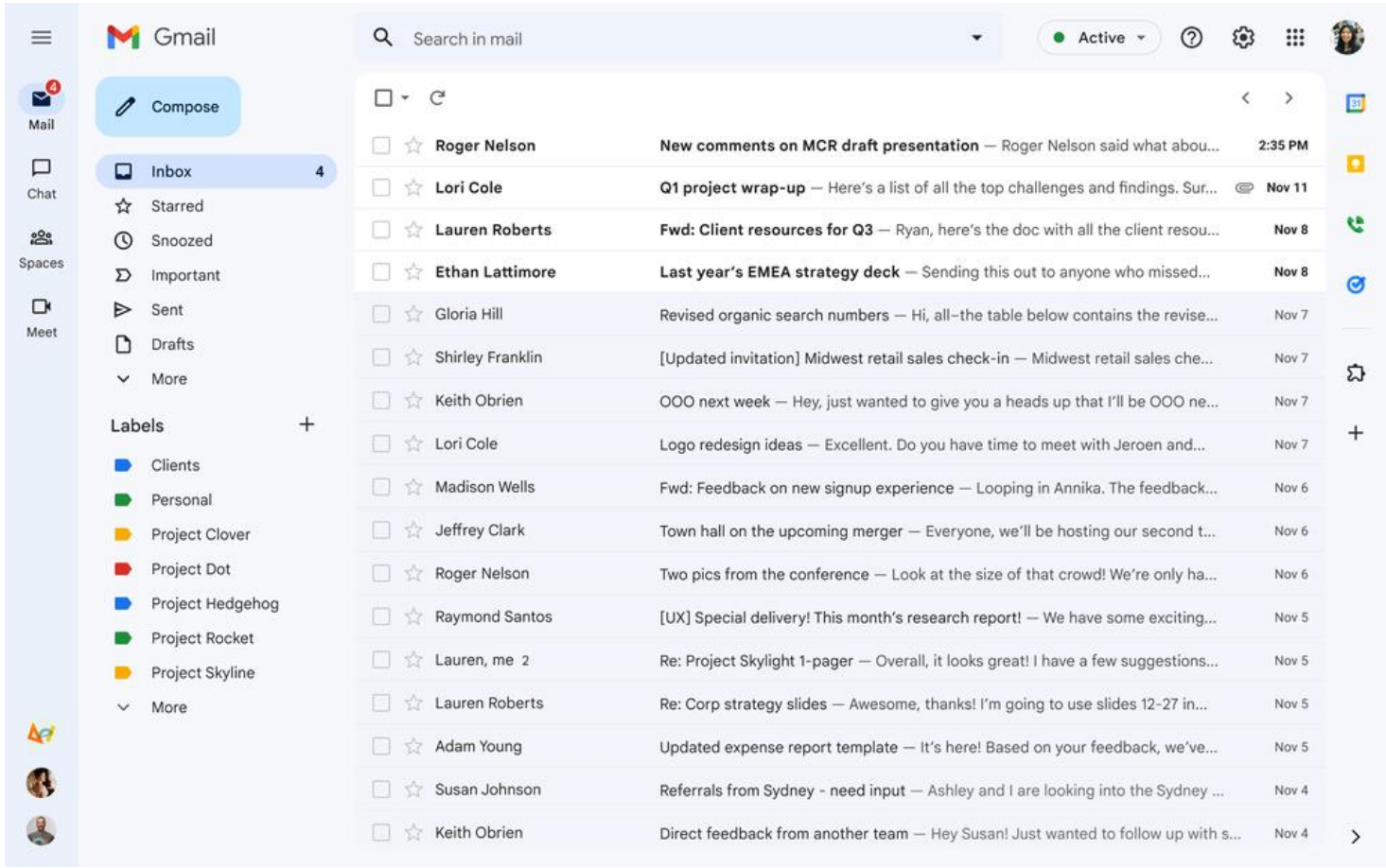
- IMAP allows to create, rename and delete mailboxes on the server and to copy messages between mailboxes.
- The message state (unread, read, replied to, etc.) and flags (important, urgent, etc.) is also stored on the server, so that clients accessing the same mailbox at different times can detect changes made by other clients.
- An Email client can manipulate emails stored on the server in the same way as using local folders



IMAP vs POP3

Feature	IMAP (Internet Message Access Protocol)	POP3 (Post Office Protocol 3)
Message Storage	Server-based; emails are stored on the email server.	Client-based; emails are downloaded to the local device.
Multiple Devices	Supports synchronization across multiple devices.	Limited support for accessing emails from multiple devices.
Message Management	Allows folder creation, organization, and server-based operations (e.g., marking emails as read/unread).	Limited folder support; typically only Inbox.
Email Download	Downloads email headers first and provides options to download full messages.	Downloads entire messages but doesn't sync with the server.
Storage Space	Emails are stored on the server, so storage space is managed by the email provider.	Local device storage must be managed by the user.
Use Cases	Ideal for users who need access to email from multiple devices and want server-based email management.	Suitable for users who want to download emails to a single device and prefer local email management.

Web-Based E-Mail



京都大学

Web-Based E-Mail

- User agent is an ordinary Web browser
  - User communicates with server via HTTP
  - E.g., Gmail, Yahoo mail, Hotmail
- Reading e-mail
  - Web pages display the contents of folders
  - … and allow users to download and view messages
  - “GET” request to retrieve the various Web pages
- Sending e-mail
  - User types the text into a form and submits to the server
  - “POST” request to upload data to the server
  - Server uses SMTP to deliver message to other servers

Email message format

- There is also a standard [RFC2822] on how the email message itself should be formatted.
  - The message does not only contain the body text but also additional header lines.
  - Similar to how HTML defines the language on how web documents are written.
- Header lines provide essential information
- Common header lines include:

  From: Sender's email address.

  To: Recipient's email address.

  Subject: The topic of the email.

  Date: The timestamp when the email was sent.

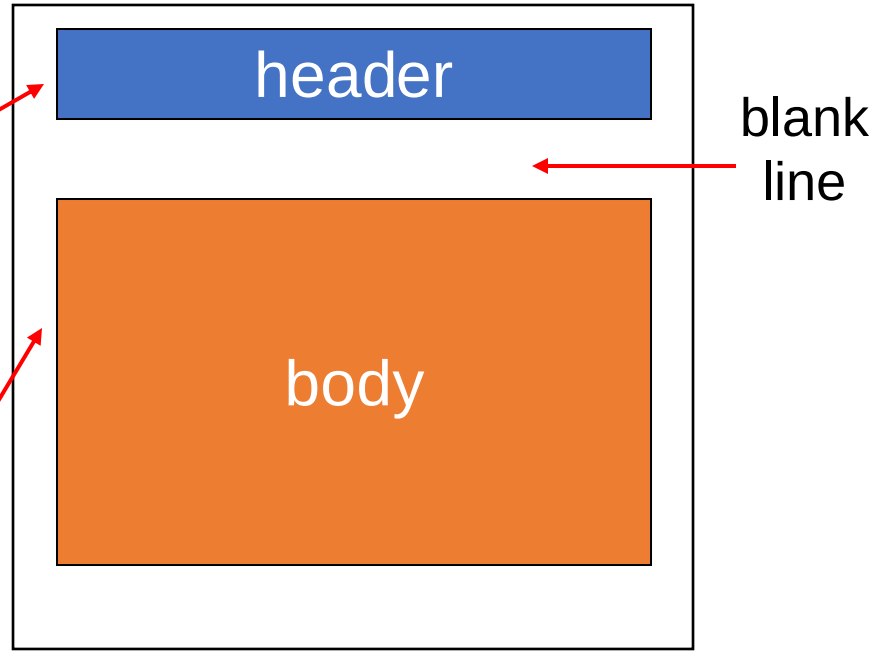
Email message format

SMTP: protocol for exchanging email messages

RFC 822: standard for text message format:

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

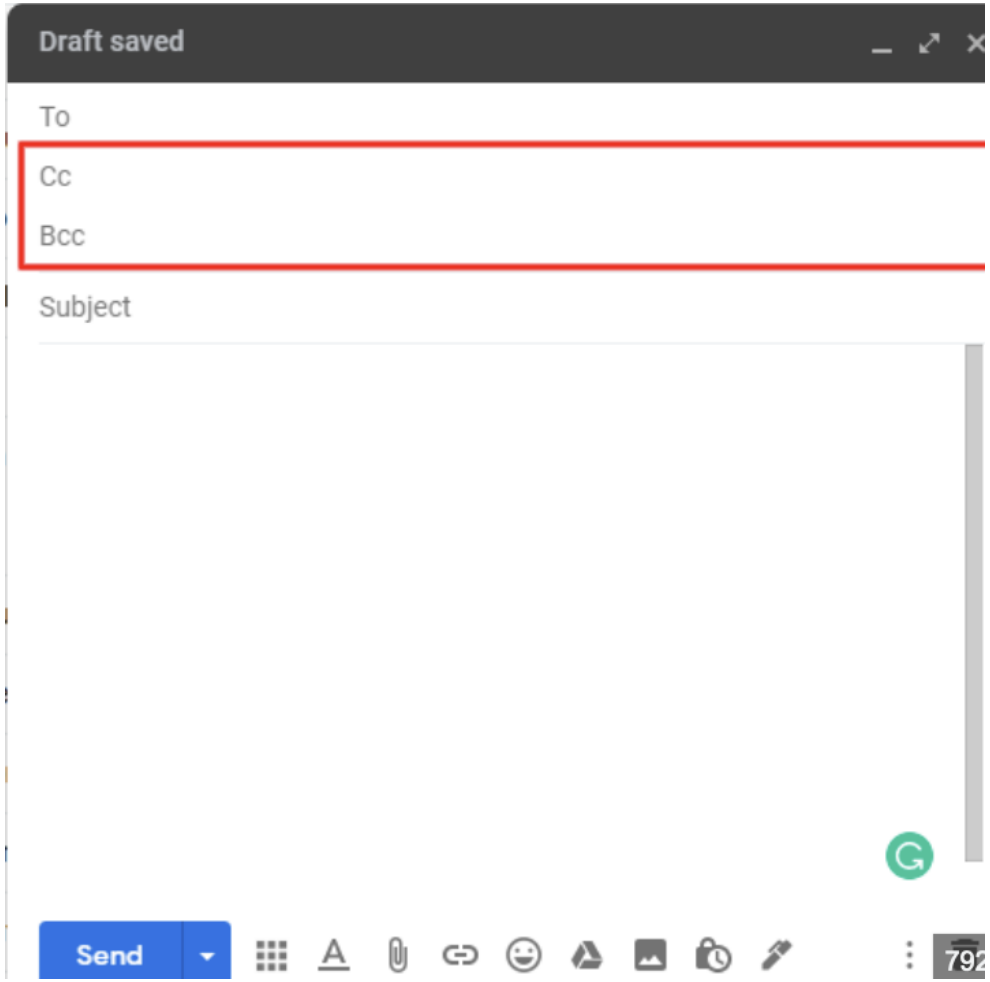
  different from SMTP MAIL FROM, RCPT TO: commands!
- Body: the "message"



京都大学

CC and BCC header lines

- E-mail clients usually have CC and BCC fields additional to the recipient address field.
- The CC (Carbon Copy) field allows to send copies of the email to other recipients.
- The CC header field is included in the sent message, so all recipients can view each other's addresses.
- The BCC (Blind Carbon Copy) field is used to include recipients without revealing their email addresses to others.
- The email client typically removes the BCC header field from the message, ensuring that recipients cannot see the addresses of others included in the BCC field.



京都大学