Chapter 2: Application layer

2.1 Principles of network applications

2.2 Web and HTTP

2.3 FTP

- http://www.ietf.org/rfc/rfc959.txt

2.4 Electronic Mail

- SMTP, POP3, IMAP

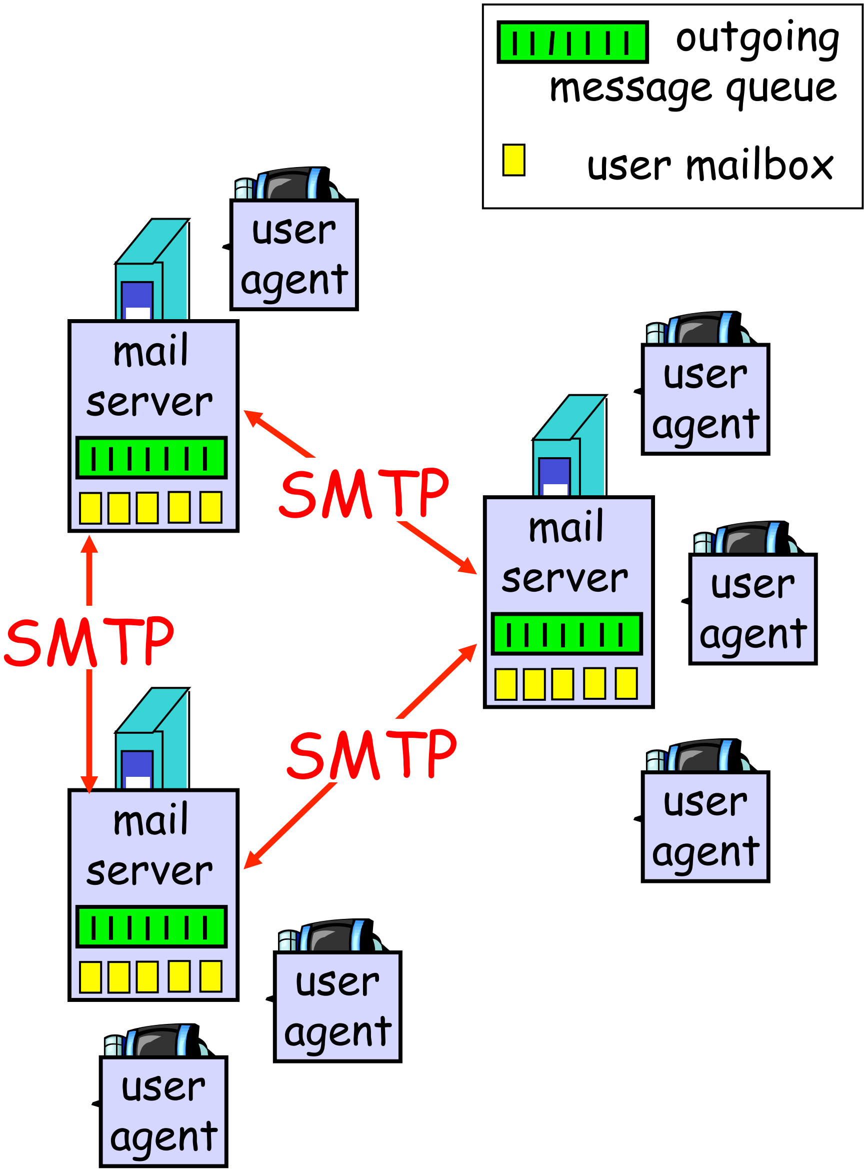
2.5 DNS

2.6 P2P applications

2.7 Socket programming with TCP

2.8 Socket programming with UDP

Electronic Mail



Three major components:

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

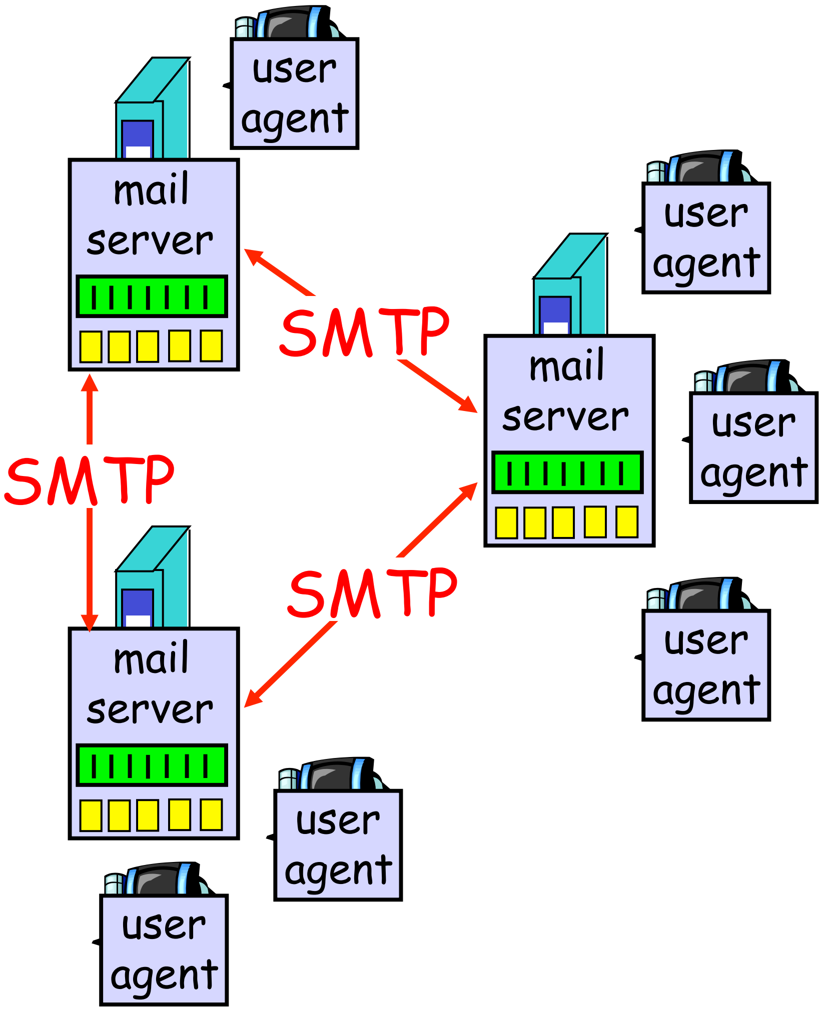
User Agent

- a.k.a. “mail reader/client”
- composing, editing, reading mail messages
- e.g., Outlook, elm/pine, Mozilla Thunderbird, iPhone mail client
- outgoing, incoming messages stored on server

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 (push) email messages
  - client: sending mail server
  - "server": receiving mail server



Electronic Mail: SMTP [RFC 5321]

- uses TCP to reliably transfer email message from client to server, port 25
- direct transfer: sending server to receiving server
  - no intermediate hops
- three phases of transfer
  - handshaking (greeting, introduction)
  - transfer of messages (reliable thanks to TCP)
  - closure
- command/response interaction
  - commands: ASCII text
  - response: status code and phrase
  - messages must be in 7-bit ASCII

Scenario: Alice sends message to Bob

1) Alice uses UA to compose message and "to" `bob@someschool.edu`

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

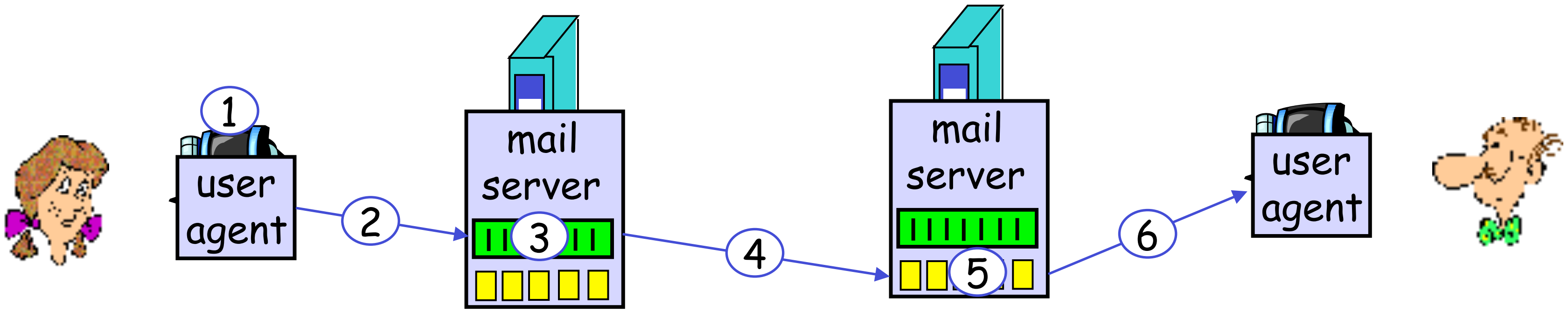
3) Client side of SMTP opens a persistent TCP connection with Bob's mail server

4) SMTP client sends Alice's message over the TCP connection

- no intermediate servers!
- if Bob's server is down, Alice's server will retry later
- Alice's server may send multiple messages (persistent connection)

5) Bob's mail server places the message in Bob's mailbox

6) Bob invokes his user agent to read message



Try SMTP interaction for yourself

Send email without using email client!

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

Check your email.

Example (from your book)

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

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Test

[Message Body in ASCII]
.
```

Example (on our own)

Telnet from my laptop (C) to odysseas.calit2.uci.edu (S)
Send email from athina@uci.edu to athina@eecs.uci.edu

```
telnet vivian.eecs.uci.edu 25
220 vivian.eecs.uci.edu
HELO uci.edu
250 Hello athina@odysseas.calit2.uci.edu [128.195.185.112],
pleased to meet you
MAIL FROM: athina@uci.edu
250 <athina@uci.edu>... Sender ok
RCPT TO: athina@eecs.uci.edu
250 <athina@odysseas.calit2.uci.edu>... Recipient ok
DATA
354 Please start mail input.
This is a test email from my uci to my eecs account
.
250 Mail queued for delivery.
QUIT
221 vivian.eecs.uci.edu closing connection
```

Check email, see source, spelling errors…

Spoofing Example

Telnet from my laptop to odysseas.eecs.uci.edu (S)
Send email from obama@whitehouse.gov to athina@uci.edu

```
telnet vivian.eecs.uci.edu 25
220 vivian.eecs.uci.edu
HELO whitehouse.gov
250 Hello athina@vivian.eecs.uci.edu [128.200.85.10], pleased
to meet you
MAIL FROM:obama@whitehouse.gov
250 <obama@whitehouse.gov>... Sender ok
RCPT TO: athina@uci.edu
250 <athina@uci.edu>... Recipient ok
DATA
354 Please start mail input.
Hello from Obama!
250 Mail queued for delivery.
QUIT
221 vivian.eecs.uci.edu closing connection
```

Server recognized me

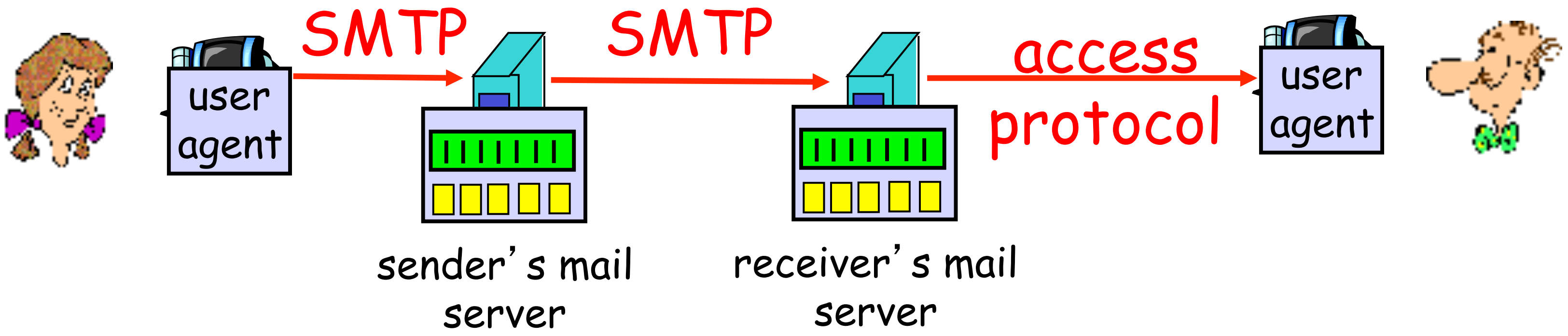
Would have not worked for non existing sender domain

1) Would have worked with RCPT To: uci.edu, gmail.com or other non-uci

2) Would have not worked if not “logged in” or relaying.

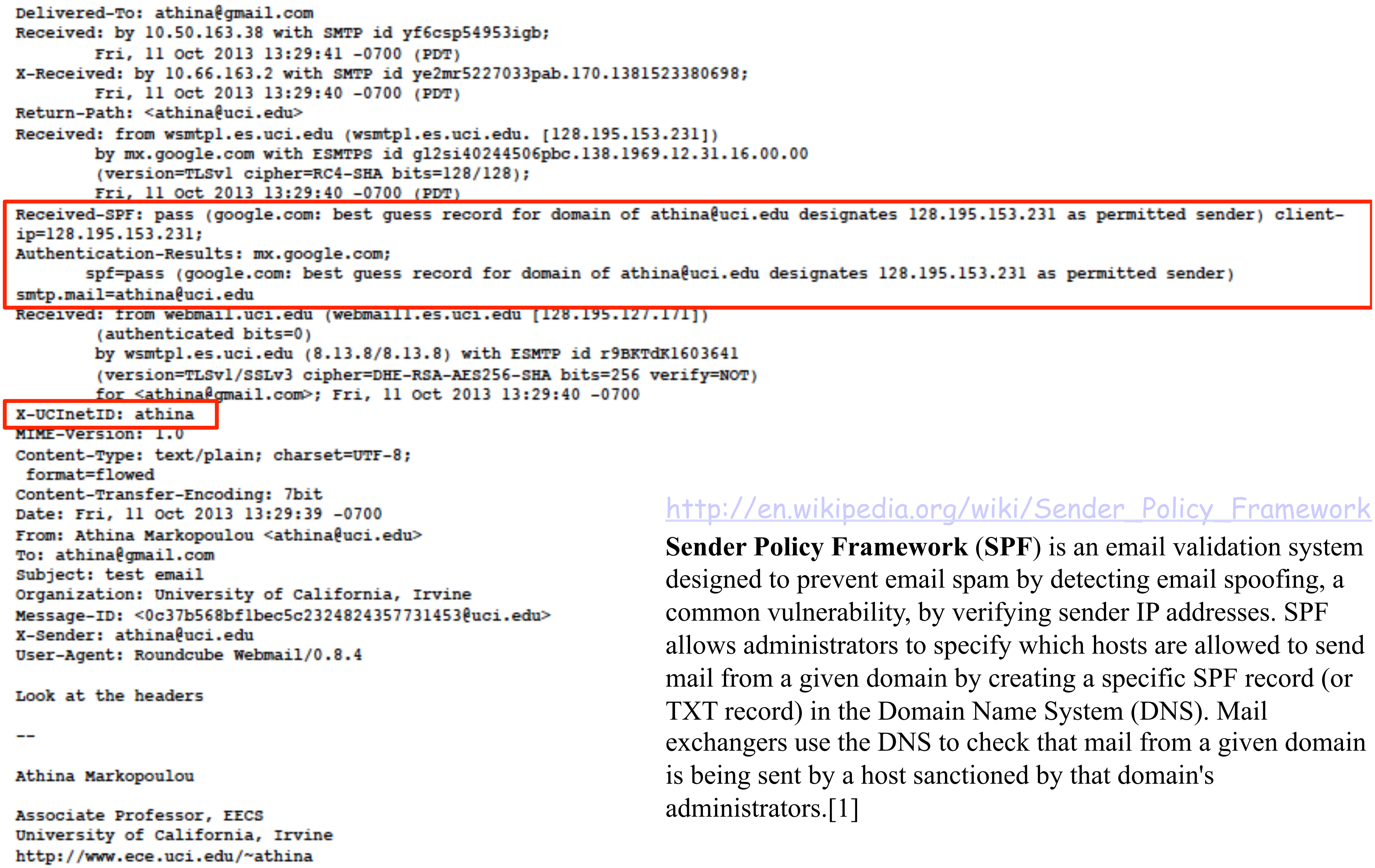
Let’s check my email… Settings, show source…
Read about MAIL FROM, RCPT TO at http://www.ietf.org/rfc/rfc2821.txt

FAQ: what can be done about spoofing/spam



- End-to-end mechanisms
  - Authentication end-to-end (PGP keys)
- Spam detection, machine learning
- User must login (authenticate) to the server, before it sends an email
- SMTP servers are more trustworthy than mail agents
  - Turn off relaying by SMTP server
  - SMTP servers also in DNS → receiving SMTP server can check IP of sending SMTP server

Sender Policy Framework (SPF): RFC 4408



```
Delivered-To: athina@gmail.com
Received: by 10.50.163.38 with SMTP id yf6csp54953igb;
        Fri, 11 Oct 2013 13:29:41 -0700 (PDT)
X-Received: by 10.66.163.2 with SMTP id ye2mr5227033pab.170.1381523380698;
        Fri, 11 Oct 2013 13:29:40 -0700 (PDT)
Return-Path: <athina@uci.edu>
Received: from wsmtpl.es.uci.edu (wsmtpl.es.uci.edu. [128.195.153.231])
        by mx.google.com with ESMTPS id gl2si40244506pbc.138.1969.12.31.16.00.00
        (version=TLSvl cipher=RC4-SHA bits=128/128);
        Fri, 11 Oct 2013 13:29:40 -0700 (PDT)
Received-SPF: pass (google.com: best guess record for domain of athina@uci.edu designates 128.195.153.231 as permitted sender) client-
ip=128.195.153.231;
Authentication-Results: mx.google.com;
       spf=pass (google.com: best guess record for domain of athina@uci.edu designates 128.195.153.231 as permitted sender)
smtp.mail=athina@uci.edu
Received: from webmail.uci.edu (webmaill.es.uci.edu [128.195.127.171])
        (authenticated bits=0)
        by wsmtpl.es.uci.edu (8.13.8/8.13.8) with ESMTP id r9BKTdK1603641
        (version=TLSvl/SSLv3 cipher=DHE-RSA-AES256-SHA bits=256 verify=NOT)
        for <athina@gmail.com>; Fri, 11 Oct 2013 13:29:40 -0700
X-UCInetID: athina
MIME-Version: 1.0
Content-Type: text/plain; charset=UTF-8;
 format=flowed
Content-Transfer-Encoding: 7bit
Date: Fri, 11 Oct 2013 13:29:39 -0700
From: Athina Markopoulou <athina@uci.edu>
To: athina@gmail.com
Subject: test email
Organization: University of California, Irvine
Message-ID: <0c37b568bf1bec5c2324824357731453@uci.edu>
X-Sender: athina@uci.edu
User-Agent: Roundcube Webmail/0.8.4

Look at the headers

--

Athina Markopoulou

Associate Professor, EECS
University of California, Irvine
http://www.ece.uci.edu/~athina
```

http://en.wikipedia.org/wiki/Sender_Policy_Framework

Sender Policy Framework (**SPF**) is an email validation system designed to prevent email spam by detecting email spoofing, a common vulnerability, by verifying sender IP addresses. SPF allows administrators to specify which hosts are allowed to send mail from a given domain by creating a specific SPF record (or TXT record) in the Domain Name System (DNS). Mail exchangers use the DNS to check that mail from a given domain is being sent by a host sanctioned by that domain's administrators.[1]

SMTP vs HTTP

SMTP	HTTP
Transfer messages (files)	Transfer webpages (files)
Push-based	Pull-based
persistent connections	persistent or non-persistent
SMTP requires message (header & body) to be in 7-bit ASCII	also have ASCII command/ response interaction, status codes
SMTP: multiple objects sent in multipart msg	HTTP: each object encapsulated in its own response msg
SMTP server uses `CRLF.CRLF` to determine end of message	HTTP terminates with `CRLF CRLF`

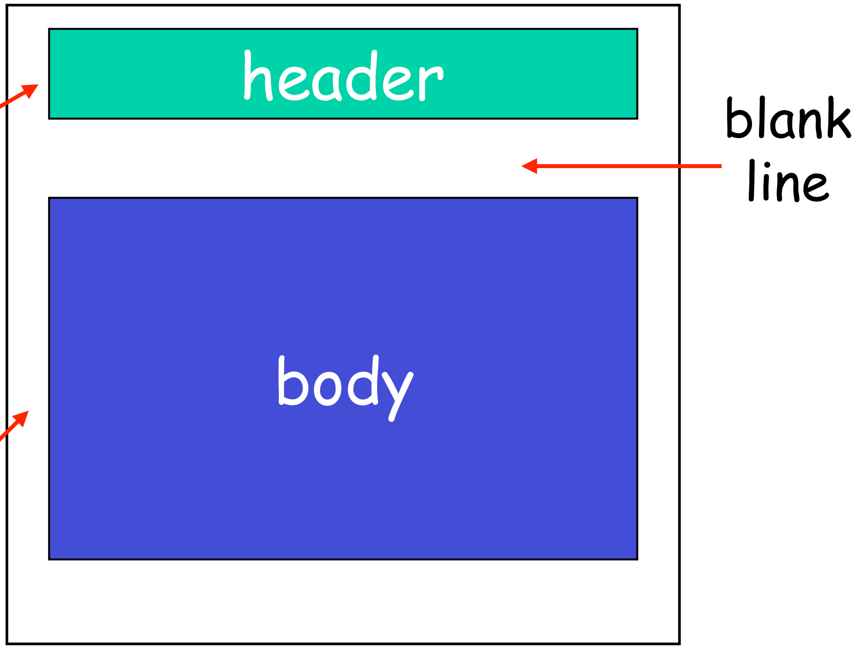
Mail message format

RFC 2821: SMTP protocol for exchanging email msgs

RFC 5322: standard for text message format:

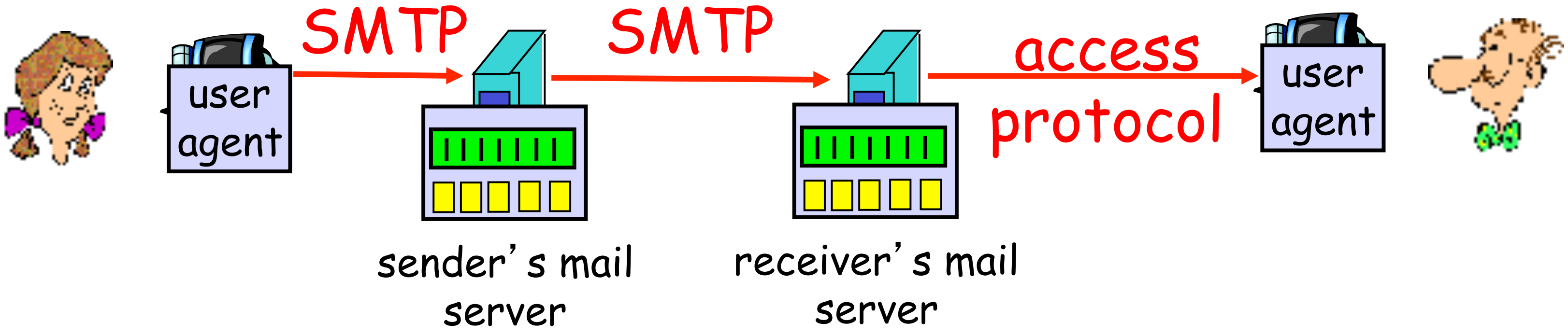
- header lines, e.g.,
  - To:
  - From:
  - Subject:
  - other optional fields

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



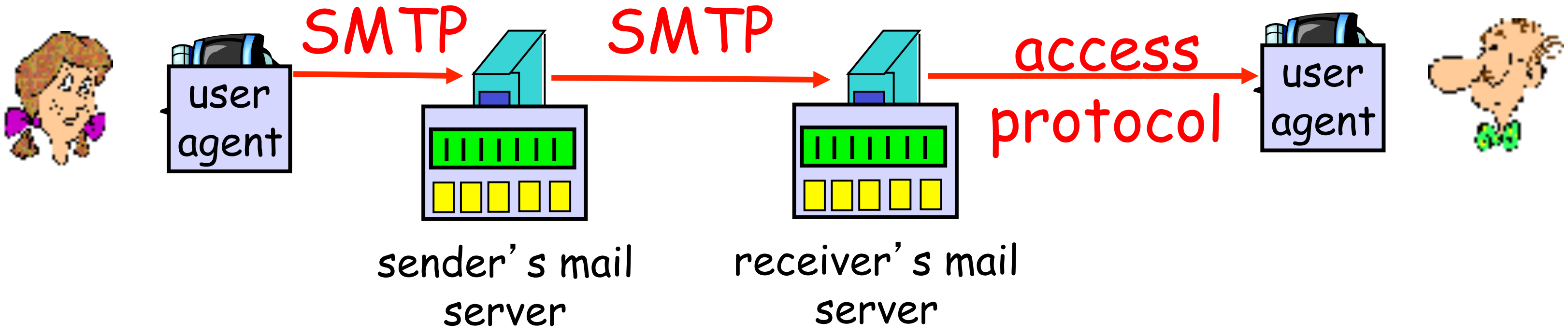
```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Test

[Message Body in ASCII]
.
```

Mail access protocols



- Why not have Alice directly contact Bob's PC?
  - Asynchronous communication
  - Bob's server: always on, shared by users and maintained by ISP
- Why not have Alice contact Bob's server directly?
  - Bob's server may be down, Alice's server can retry
- How does Bob get his message?
  - Pull operation, not SMTP

Mail access protocols



- SMTP: delivery/storage (push) to receiver's server
- mail access protocol: retrieval (pull) from server
  - POP: Post Office Protocol [RFC 1939]
    - authorization (agent <-->server) and download
  - IMAP: Internet Mail Access Protocol [RFC 1730]
    - more features (more complex)
    - manipulation of stored msgs on server
  - HTTP: gmail, Hotmail, Yahoo! Mail, etc.

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`

```
C: telnet mailserver 110
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) and IMAP

more about POP3

- previous example uses "download and delete" mode.
- Bob cannot re-read e-mail if he changes client
- "download-and-keep": copies of messages on different clients
- POP3 is stateless across sessions
- Folders and messages on local machine(s) are not ideal for the "nomadic" user

IMAP

- keeps all messages in one place: at server
- allows user to organize messages in remote folders
- keeps user state across sessions:
  - names of folders and mappings between message IDs and folder name
- Other: search, download parts

Web-based email

- Server-browser via HTTP
- E.g.: yahoo, gmail, webmail