

# Communication Networks

Prof. Laurent Vanbever

## Communication Networks

Spring 2019



Laurent Vanbever  
[nsg.ee.ethz.ch](mailto:nsg.ee.ethz.ch)

ETH Zürich (D-ITET)  
May 6 2019

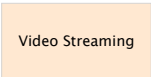
Materials inspired from Scott Shenker and Jennifer Rexford

### Last Monday on Communication Networks

2



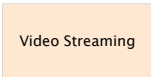
<http://www.google.ch>



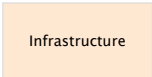
HTTP-based



<http://www.google.ch>



### The WWW is made of three key components



Clients/Browser  
Servers  
Proxies



Objects  
files, pictures, videos, ...  
*organized in*  
Web sites  
a collection of objects

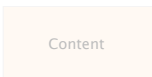


URL: name content  
HTTP: transport content

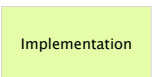
### We'll focus on its implementation



Clients/Browser  
Servers  
Proxies



Objects  
files, pictures, videos, ...  
*organized in*  
Web sites  
a collection of objects



URL: name content  
HTTP: transport content

### A Uniform Resource Locator (URL) refers to an Internet resource

`protocol://hostname[:port]/directory_path/resource`

HTTP is a rather simple  
synchronous request/reply protocol

- HTTP is layered over a bidirectional byte stream  
almost always TCP
- HTTP is text-based (ASCII)  
human readable, easy to reason about
- HTTP is stateless  
it maintains *no info* about past client requests



HTTP clients make request to the server

HTTP request

|                    |      |       |      |         |          |
|--------------------|------|-------|------|---------|----------|
| method             | <sp> | URL   | <sp> | version | <cr><lf> |
| header field name: |      | value |      |         | <cr><lf> |
| ...                |      |       |      |         |          |
| header field name: |      | value |      |         | <cr><lf> |
| <cr><lf>           |      |       |      |         |          |
| body               |      |       |      |         |          |

|         |                                        |                             |
|---------|----------------------------------------|-----------------------------|
| method  | GET                                    | return resource             |
|         | HEAD                                   | return headers only         |
|         | POST                                   | send data to server (forms) |
| URL     | relative to server (e.g., /index.html) |                             |
| version | 1.0, 1.1, 2.0                          |                             |

HTTP servers answers to clients' requests

HTTP response

|                    |      |        |      |        |          |
|--------------------|------|--------|------|--------|----------|
| version            | <sp> | status | <sp> | phrase | <cr><lf> |
| header field name: |      | value  |      |        | <cr><lf> |
| ...                |      |        |      |        |          |
| header field name: |      | value  |      |        | <cr><lf> |
| <cr><lf>           |      |        |      |        |          |
| body               |      |        |      |        |          |

|        | 3 digit response code | reason phrase     |
|--------|-----------------------|-------------------|
| Status | 1XX                   | informational     |
|        | 2XX                   | success           |
|        | 3XX                   | redirection       |
|        | 4XX                   | client error      |
|        | 5XX                   | server error      |
|        | 200                   | OK                |
|        | 301                   | Moved Permanently |
|        | 303                   | Moved Temporarily |
|        | 304                   | Not Modified      |
|        | 404                   | Not Found         |
|        | 505                   | Not Supported     |

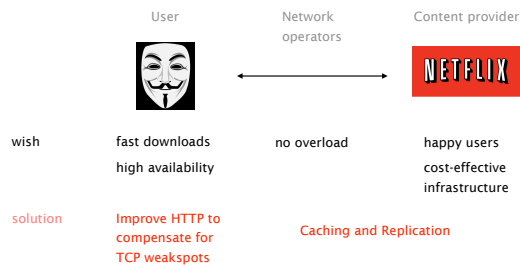
HTTP makes the client maintain the state.  
This is what the so-called **cookies** are for!



- client stores small state  
on behalf of the server X
- client sends state  
in all future requests to X
- can provide authentication



Performance goals vary depending on who you ask



Considering the time to retrieve  $n$  small objects, pipelining wins

|               | # RTTS      |
|---------------|-------------|
| one-at-a-time | $\sim 2n$   |
| M concurrent  | $\sim 2n/M$ |
| persistent    | $\sim n+1$  |
| pipelined     | 2           |

Considering the time to retrieve  $n$  big objects, there is no clear winners as bandwidth matters more

$$\# \text{ RTTS} \sim \frac{n * \text{avg. file size}}{\text{bandwidth}}$$

To limit staleness of cached objects, HTTP enables a client to validate cached objects

- Server hints when an object expires (kind of TTL) as well as the last modified date of an object
- Client conditionally requests a resource using the "if-modified-since" header in the HTTP request
- Server compares this against "last modified" time of the resource and returns:
  - Not Modified if the resource has not changed
  - OK with the latest version

Caching can and is performed at different locations

|                          |                                                     |
|--------------------------|-----------------------------------------------------|
| client                   | browser cache                                       |
| close to the client      | forward proxy<br>Content Distribution Network (CDN) |
| close to the destination | reverse proxy                                       |



We want the highest video quality



(c) copyright 2008, Blender Foundation / [www.bigbuckbunny.org](http://www.bigbuckbunny.org), CC-BY-3.0

Without seeing this ...

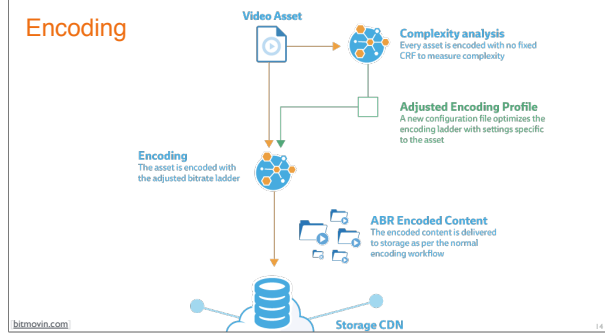


## The three steps behind most contemporary solutions

- Encode video in multiple bitrates
- Replicate using a content delivery network
- Video player picks bitrate adaptively
  - Estimate connection's available bandwidth
  - Pick a bitrate  $\leq$  available bandwidth

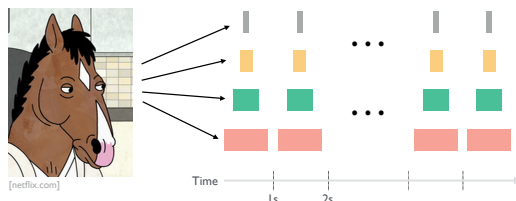
6

## Encoding



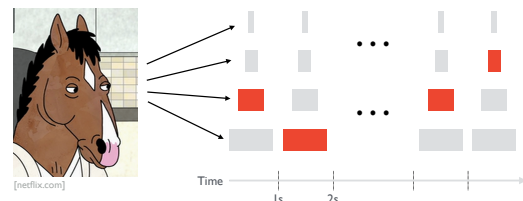
11

## Your player download “chunks” of video at different bitrates



16

## Depending on your network connectivity, your player fetches chunks of different qualities



17

## Your player gets metadata about chunks via “Manifest”

```
<?xml version="1.0" encoding="UTF-8"?>
<MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="urn:mpeg:DASH:schema:MPD:2011"
  xsi:schemaLocation="urn:mpeg:DASH:schema:MPD:2011"
  profiles="urn:mpeg:dash:profile:isoff-main:2011"
  type="static"
  mediaPresentationDuration="PT0H9M56.465S"
  minBufferTime="PT15.0S">
  <BaseURL>http://wilestlab.poly.edu/~ffund/video/2s_480p_only/</BaseURL>
  <Period start="PT0S">
    <AdaptationSet bitstreamSwitching="true">
      <Representation id="0" codecs="avc1" mimeType="video/mp4"
        width="480" height="360" startWithSAP="1" bandwidth="101492">
        <SegmentBase>
          <Initialization sourceURL="bunny_2s_100kbit/bunny_100kbit.mp4"/>
          <SegmentList duration="2">
            <SegmentURL media="bunny_2s_100kbit/bunny_2s1.m4s"/>
            <SegmentURL media="bunny_2s_100kbit/bunny_2s2.m4s"/>
            <SegmentURL media="bunny_2s_100kbit/bunny_2s3.m4s"/>
            <SegmentURL media="bunny_2s_100kbit/bunny_2s4.m4s"/>
            <SegmentURL media="bunny_2s_100kbit/bunny_2s5.m4s"/>
            <SegmentURL media="bunny_2s_100kbit/bunny_2s6.m4s"/>
          </SegmentList>
        </Representation>
      </AdaptationSet>
    </Period>
  </MPD>
```

[wilestlab.poly.edu]

18

## NETFLIX Open Connect: Starting from a Greenfield (a mostly Layer 0 talk)

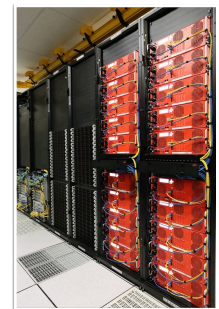
Dave Temkin  
06/01/2015

### Storage Appliance

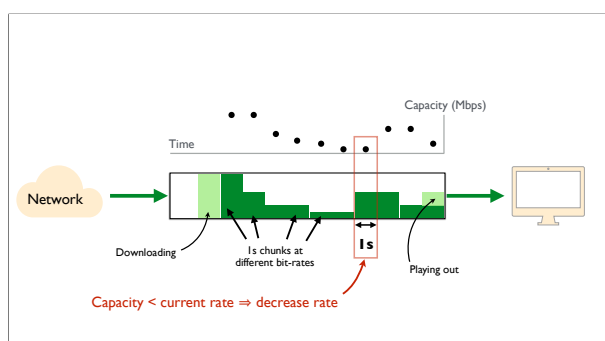
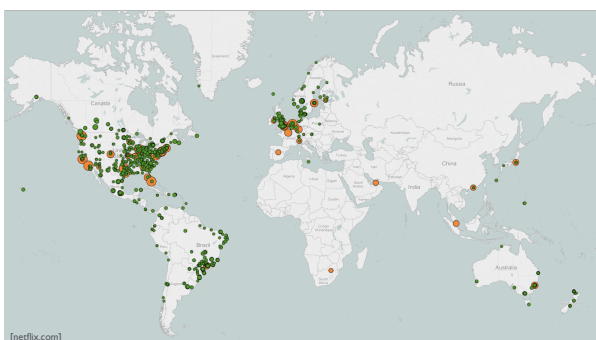
- Still 4U high
- ~550 watts
- 288 TB of storage
- 2x 10G ports
- 20Gbit/s delivery

### Flash Appliance

- 1U
- ~175 watts
- 24 TB of flash
- 2x 40G ports
- 40Gbit/s delivery

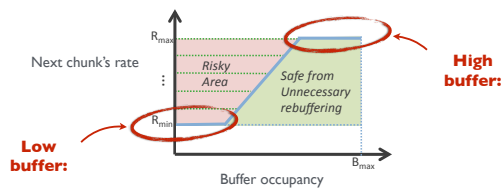


20





## Buffer-based adaptation

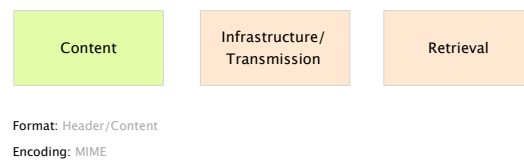
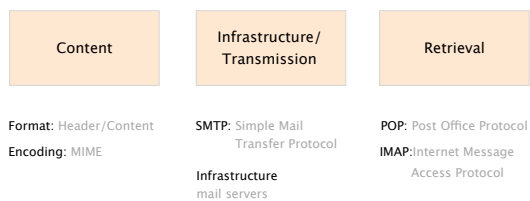


[A Buffer-Based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service, Huang et al., ACM SIGCOMM 2014]

## Today on Communication Networks



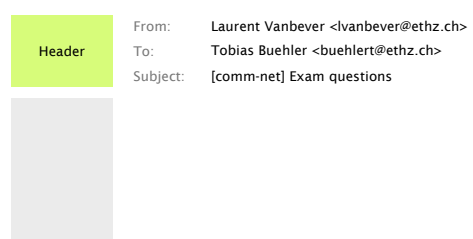
## We'll study e-mail from three different perspectives



## An e-mail is composed of two parts



## A header, in 7-bit U.S. ASCII text



A body, also in 7-bit U.S. ASCII text

**Header**

From: Laurent Vanbever <lvanbever@ethz.ch>  
To: Tobias Buehler <buehlert@ethz.ch>  
Subject: [comm-net] Exam questions

**Body**

Hi Tobias,  
  
Here are some interesting questions...

Best,  
Laurent

Type, followed by ":"      Value

**Header**

From: Laurent Vanbever <lvanbever@ethz.ch> CRLF  
To: Tobias Buehler <buehlert@ethz.ch> CRLF  
Subject: [comm-net] Exam questions CRLF

Series of lines ending with Carriage Return and Line Feed

Series of lines with no structure/meaning

**Body**

Hi Tobias,  
  
Here are some interesting questions...

Best,  
Laurent

**Header**

**Body**

A blank line separates the header from the body

Email relies on 7-bit U.S. ASCII...  
How do you send non-English text? Binary files?

**Solution**      **Multipurpose Internet Mail Extensions**  
commonly known as MIME, standardized in RFC 822

**MIME defines**

- additional headers for the email body
- a set of content types and subtypes
- base64 to encode binary data in ASCII

**MIME defines**

- additional headers for the email body

MIME-Version: the version of MIME being used  
Content-Type: the type of data contained in the message  
Content-Transfer-Encoding: how the data are encoded

**MIME defines**

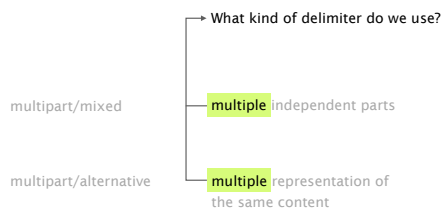
- additional headers for the email body
  - a set of content types and subtypes
- e.g. image with subtypes gif or jpeg  
text with subtypes plain, html, and rich text  
application with subtypes postscript or msword  
multipart with subtypes mixed or alternative

The two most common types/subtypes for MIME are:  
*multipart/mixed* and *multipart/alternative*

|                       |                                                                                |
|-----------------------|--------------------------------------------------------------------------------|
| Content-Type          | indicates that the message contains                                            |
| multipart/mixed       | multiple independent parts<br>e.g. plain text <i>and</i> a binary file         |
| multipart/alternative | multiple representation of the same content<br>e.g. plain text <i>and</i> HTML |

MIME defines

- additional headers for the email body
- a set of content types and subtypes
- **base64** to encode binary data in ASCII



Content-Type contains a parameter that specifies a string delimiter (usually chosen randomly by the client)

MIME relies on Base64 as binary-to-text encoding scheme

Relies on 64 characters out of the 128 ASCII characters the most common *and* printable ones, i.e. A-Z, a-z, 0-9, +, /

Divides the bytes to be encoded into sequences of 3 bytes each group of 3 bytes is then encoded using 4 characters

Uses padding if the last sequence is partially filled  
 i.e. if the |sequence| to be encoded is not a multiple of 3

|              |                                                            |
|--------------|------------------------------------------------------------|
| Binary input | 0x14fb9c03d97e                                             |
| 8-bits       | 00010100 11111011 10011100<br>00000011 11011001 01111110   |
| 6-bits       | 000101 001111 101110 011100<br>000000 111101 100101 111110 |
| Decimal      | 5 15 46 28 0 61 37 62                                      |
| base64       | F P u c A 9 l +                                            |

| Value | Char | Value | Char | Value | Char | Value | Char |
|-------|------|-------|------|-------|------|-------|------|
| 0     | A    | 16    | Q    | 32    | g    | 48    | w    |
| 1     | B    | 17    | R    | 33    | h    | 49    | x    |
| 2     | C    | 18    | S    | 34    | i    | 50    | y    |
| 3     | D    | 19    | T    | 35    | j    | 51    | z    |
| 4     | E    | 20    | U    | 36    | k    | 52    | 0    |
| 5     | F    | 21    | V    | 37    | l    | 53    | 1    |
| 6     | G    | 22    | W    | 38    | m    | 54    | 2    |
| 7     | H    | 23    | X    | 39    | n    | 55    | 3    |
| 8     | I    | 24    | Y    | 40    | o    | 56    | 4    |
| 9     | J    | 25    | Z    | 41    | p    | 57    | 5    |
| 10    | K    | 26    | a    | 42    | q    | 58    | 6    |
| 11    | L    | 27    | b    | 43    | r    | 59    | 7    |
| 12    | M    | 28    | c    | 44    | s    | 60    | 8    |
| 13    | N    | 29    | d    | 45    | t    | 61    | 9    |
| 14    | O    | 30    | e    | 46    | u    | 62    | +    |
| 15    | P    | 31    | f    | 47    | v    | 63    | /    |

If the length of the input is not a multiple of three, Base64 uses "=" as padding character

|              |               |
|--------------|---------------|
| Binary input | 0x14          |
| 8-bits       | 00010100      |
| 6-bits       | 000101 000000 |
| Decimal      | 5 0           |
| base64       | F A = =       |

```

From: Laurent Vanbever <lvanbever@ethz.ch>
To: Tobias Buehler <buehlert@ethz.ch>
Subject: [comm-net] Final exam
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: multipart/mixed;
        boundary="--123boundary"

```

This is a multipart message in MIME format.

```

--123boundary
Content-Type: text/plain

```

Hi Tobias, Please find the exam enclosed. Laurent

```

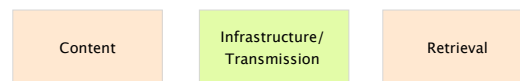
--123boundary
Content-Type: application/pdf;
Content-Disposition: attachment;
        filename="exam_2018.pdf"

```

```

base64 encoded data .....
.....base64 encoded data

```



SMTP: Simple Mail Transfer Protocol

Infrastructure mail servers

An e-mail address is composed of two parts identifying the local mailbox and the domain

lvanbever @ ethz.ch

local mailbox

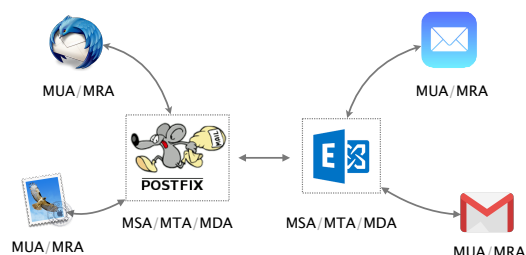
domain name

actual mail server is identified using a DNS query asking for MX records

We can divide the e-mail infrastructure into five functions

|      |              |       |                                            |
|------|--------------|-------|--------------------------------------------|
| Mail | User         | Agent | Use to read/write emails (mail client)     |
| Mail | Submission   | Agent | Process email and forward to local MTA     |
| Mail | Transmission | Agent | Queues, receives, sends mail to other MTAs |
| Mail | Delivery     | Agent | Deliver email to user mailbox              |
| Mail | Retrieval    | Agent | Fetches email from user mailbox            |

MSA/MTA/MDA and MRA/MUA are often packaged together leading to simpler workflows



Simple Mail Transfer Protocol (SMTP) is the current standard for transmitting e-mails

SMTP is a text-based, client-server protocol  
client sends the e-mail, server receives it

SMTP uses reliable data transfer  
built on top of TCP (port 25 and 465 for SSL/TLS)

SMTP is a push-like protocol  
sender pushes the file to the receiving server (no pull)

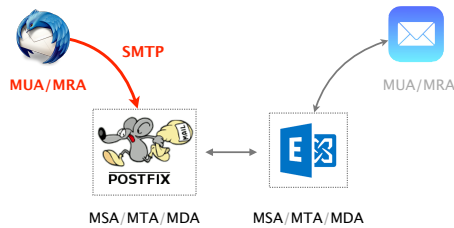
|        | SMTP 3 digit response code |                 |     | comment                         |
|--------|----------------------------|-----------------|-----|---------------------------------|
| Status | 2XX                        | success         | 220 | Service ready                   |
|        |                            |                 | 250 | Requested mail action completed |
|        | 3XX                        | input needed    | 354 | Start mail input                |
|        |                            |                 | 421 | Service not available           |
|        | 4XX                        | transient error | 450 | Mailbox unavailable             |
|        |                            |                 | 452 | Insufficient space              |
|        |                            |                 | 500 | Syntax error                    |
|        | 5XX                        | permanent error | 502 | Unknown command                 |
|        |                            |                 | 503 | Bad sequence                    |

```

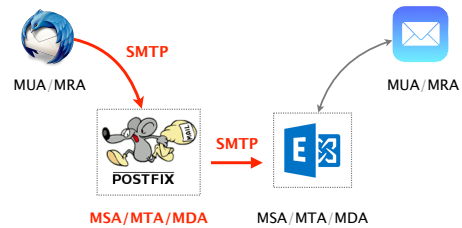
server — 220 hamburger.edu
          EHLO crepes.fr
          250 Hello crepes.fr, pleased to meet you
client — MAIL FROM: <alice@crepes.fr>
          250 alice@crepes.fr... Sender ok
          RCPT TO: <bob@hamburger.edu>
          250 bob@hamburger.edu ... Recipient ok
          DATA
          354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

```

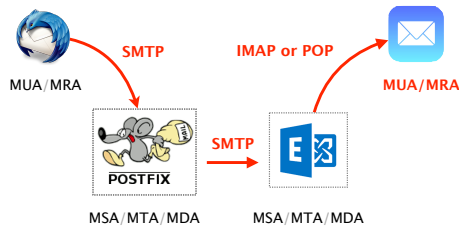
The sender MUA uses SMTP to transmit the e-mail first to a local MTA (e.g. mail.ethz.ch, gmail.com, hotmail.com)



The local MTA then looks up the MTA of the recipient domain (DNS MX) and transmits the e-mail further



Once the e-mail is stored at the recipient domain, IMAP or POP is used to retrieve it by the recipient MUA



E-mails typically go through **at least 2 SMTP servers**, but often way more

sending and receiving sides

Each SMTP server/MTA hop adds its identity to the e-mail header by prepending a "Received" entry

```

8 Received: from edge20.ethz.ch (82.130.99.26) by CAS10.d.ethz.ch
  (172.31.38.210) with Microsoft SMTP Server (TLS) id 14.3.361.1; Fri, 23 Feb
  2018 01:48:56 +0100
7 Received: from phil4.ethz.ch (129.132.183.133) by edge20.ethz.ch
  (82.130.99.26) with Microsoft SMTP Server id 14.3.361.1; Fri, 23 Feb 2018
  01:48:57 +0100
6 Received: from outprodmail02.cc.columbia.edu ([128.59.72.51]) by phil4.ethz.ch
  with esmtps (TLSv1:AES256-SHA:256) (Exim 4.69) (envelope-from
  <ethan@ee.columbia.edu>) id 1ep1Xg-0002s3-FH for lvanbever@ethz.ch; Fri, 23
  Feb 2018 01:48:55 +0100
5 Received: from hazelnut (hazelnut.cc.columbia.edu [128.59.213.250]) by
  outprodmail02.cc.columbia.edu (8.14.4/8.14.4) with ESMTP id w1N0iAu4026008
  for <lvanbever@ethz.ch>; Thu, 22 Feb 2018 19:48:51 -0500
4 Received: from hazelnut (localhost.localdomain [127.0.0.1]) by hazelnut
  (Postfix) with ESMTP id 421126D for <lvanbever@ethz.ch>; Thu, 22 Feb 2018
  19:48:52 -0500 (EST)
3 Received: from sendprodmail01.cc.columbia.edu (sendprodmail01.cc.columbia.edu
  [128.59.72.13]) by hazelnut (Postfix) with ESMTP id 211526D for
  <lvanbever@ethz.ch>; Thu, 22 Feb 2018 19:48:52 -0500 (EST)
2 Received: from mail-pl0-f43.google.com (mail-pl0-f43.google.com
  [209.85.160.43]) (user=ebk2141 mech=PLAIN bits=0) by
  sendprodmail01.cc.columbia.edu (8.14.4/8.14.4) with ESMTP id w1N0mnlx052337
  (version=TLSv1/SSLv3 cipher=AES128-GCM-SHA256 bits=128 verify=NOT) for
  <lvanbever@ethz.ch>; Thu, 22 Feb 2018 19:48:50 -0500
1 Received: by mail-pl0-f43.google.com with SMTP id u13so3927207plq.1 for
  <lvanbever@ethz.ch>; Thu, 22 Feb 2018 16:48:50 -0800 (PST)
  
```

E-mails typically go through at least 2 SMTP servers, but often way more

Separate SMTP servers for separate functions  
SPAM filtering, virus scanning, data leak prevention, etc.

Separate SMTP servers that redirect messages  
e.g. from lvanbever@tik.ee.ethz.ch to lvanbever@ethz.ch

Separate SMTP servers to handle mailing-list  
mail is delivered to the list server and then expanded

Try it out yourself!

```

SMTP-MTA      telnet server_name 25
plaintext (!),
hard to find

SMTP-MSA      openssl s_client -starttls smtp
              -connect mail.ethz.ch:587
              -crlf -ign_eof (*)

authentication required
per1 -MMIME::Base64 -e 'print encode_base64("username");'
per1 -MMIME::Base64 -e 'print encode_base64("password");'
  
```

(\*) <https://www.ndchost.com/wiki/mail/test-smtp-auth-telnet>

As with most of the key Internet protocols, security is an afterthought

#### SMTP Headers

**MAIL FROM:** no checks are done to verify that the sending MTA is authorized to send e-mails on behalf of that address

#### Email content (DATA)

**From:** no checks are done to verify that the sending system is authorized to send e-mail on behalf of that address

**Reply-to:** ditto

In short, *none* of the addresses in an email are typically reliable

Let's spoof some e-mails!

And, as usual, multiple countermeasures have been proposed with various level of deployment success

#### Example\*

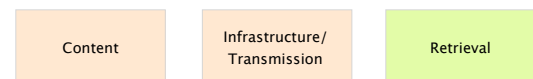
##### Sender Policy Framework (SPF)

Enables a domain to explicitly authorize a set of hosts that are allowed to send emails using their domain names in "MAIL FROM".

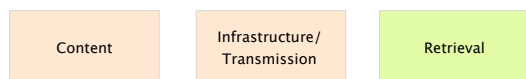
##### How? using a DNS TXT resource record

look for "v=spf1" in the results of "dig TXT google.com"

\* if you are interested, also check out Sender ID, DKIM, and DMARC



POP: Post Office Protocol  
IMAP: Internet Message Access Protocol



POP: Post Office Protocol  
IMAP: Internet Message Access Protocol

POP is a simple protocol which was designed to support users with intermittent network connectivity

POP enables e-mail users to

- retrieve e-mails locally when connected
- view/manipulate e-mails when disconnected

and that's pretty much it...

#### Example

```
POP server — +OK POP3 server ready
              user bob
              +OK
client — pass hungry
         +OK user successfully logged on

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

#### Authorization phase

Clients declares username  
password  
Server answers +OK/-ERR

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

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

### Transaction phase

**list** get message numbers  
**retr** retrieve message X  
**dele** delete message X  
**quit** exit session

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

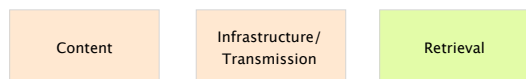
list
1 498
2 912
.
retr 1
<message 1 contents>
.
dele 1
retr 2
<message 1 contents>
.
dele 2
quit
+OK POP3 server signing off
```

POP is heavily limited. Among others, it does not go well with multiple clients or always-on connectivity

**Cannot deal with multiple mailboxes**  
designed to put incoming emails in one folder

**Not designed to keep messages on the server**  
designed to download messages to the client

**Poor handling of multiple-client access**  
while many (most?) users have now multiple devices



POP: Post Office Protocol  
IMAP: Internet Message Access Protocol

Unlike POP, Internet Message Access Protocol (IMAP) was designed with multiple clients in mind

**Support multiple mailboxes and searches on the server**  
client can create, rename, move mailboxes & search on server

**Access to individual MIME parts and partial fetch**  
client can download only the text content of an e-mail

**Support multiple clients connected to one mailbox**  
server keep state about each message (e.g. read, replied to)



MX, SMTP, POP, IMAP

Introduction

## Communication Networks Spring 2019



Laurent Vanbever  
[nsg.ee.ethz.ch](mailto:nsg.ee.ethz.ch)

ETH Zürich (D-ITET)  
May 6 2019