Last Monday on
Communication Networks
google.ch ← 172.217.16.131 → http://www.google.ch
The DNS system is a distributed database which enables to resolve a name into an IP address.

```
name          DNS          IP address
www.ethz.ch   129.132.19.216
```
To scale, DNS adopt **three** intertwined hierarchies:

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>naming structure</td>
<td>addresses are hierarchical</td>
</tr>
<tr>
<td></td>
<td><a href="http://www.ee.ethz.ch">www.ee.ethz.ch</a></td>
</tr>
<tr>
<td>management</td>
<td>hierarchy of authority</td>
</tr>
<tr>
<td></td>
<td>over names</td>
</tr>
<tr>
<td>infrastructure</td>
<td>hierarchy of DNS servers</td>
</tr>
</tbody>
</table>
naming structure addresses are hierarchical
www.ee.ethz.ch
Top Level Domain (TLDs) sit at the top

- com
- org
- net
- edu
- gov
- mil
- be
- ch
- de
- fr
- + many more
Domains are subtrees
A name, *e.g.* ee.ethz.ch, represents a leaf-to-root path in the hierarchy
management hierarchy of authority over names
The DNS system is hierarchically administered.
infrastructure  hierarchy of DNS servers
13 root servers (managed professionally) serve as root (*)

(*) see http://www.root-servers.org/
The bottom (and bulk) of the hierarchy is managed by Internet Service Provider or locally.
Every server knows the address of the root servers (*) required for bootstrapping the systems

(*) see https://www.internic.net/domain/named.root
Each server knows the address of all children
Using DNS relies on two components

get_host_by_name()
<table>
<thead>
<tr>
<th>Records</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>hostname</td>
<td>IP address</td>
</tr>
<tr>
<td>NS</td>
<td>domain</td>
<td>DNS server name</td>
</tr>
<tr>
<td>MX</td>
<td>domain</td>
<td>Mail server name</td>
</tr>
<tr>
<td>CNAME</td>
<td>alias</td>
<td>canonical name</td>
</tr>
<tr>
<td>PTR</td>
<td>IP address</td>
<td>corresponding hostname</td>
</tr>
</tbody>
</table>
DNS resolution can either be recursive or iterative
Where is .edu?

Where is nyu.edu?

Where is www.nyu.edu?
The WWW is made of three key components.

- **Infrastructure**
  - Clients/Browser
  - Servers
  - Proxies

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

- **Implementation**
  - URL: name content
  - HTTP: transport content
Performance

Protocol

http://

SPEED LIMIT 80

Performance
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
Today on

Communication Networks
Video Streaming
HTTP-based

E-mail
MX, SMTP, POP, IMAP
Video Streaming

E-mail

HTTP-based
We want the highest video quality
Without seeing this …
Why should you care? Just look at this: video's share of global internet traffic

Mark Cuban: Only a 'Moron' Would Buy YouTube

Data: Cisco VNI, 2016 to 2020 forecasted.
A naive approach: one-size-fits-all

[bitmovin.com]
In practice, things are slightly more complex

[Adapted from: Adaptive Streaming of Traditional and Omnidirectional Media, Begen & Timmerer, ACM SIGCOMM Tutorial, 2017]
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
Encoding

Replication

Adaptation
Video size: 1920 x 1080 px
Screen size: 1920 x 1080 px

Video size: 1280x 720 px
Screen size: 1280x 720 px

Video size: 854x 480 px
Screen size: 854x 480 px

Video size: 426x 240 px
Screen size: 426x 240 px
1920 x 1080 px

Fast Internet

Screen size: 1920 x 1080 px
With fast internet.

Video plays at high quality
1920 x 1080 px with no buffering

1280x720 px

Slow Internet

Screen size: 1920 x 1080 px
With slower internet.

Video plays at medium quality
1280x720 px with no buffering
Normal connection:
The Player downloads the best quality video

Normal connection:
The Player returns to the maximum quality video file

Poor connection:
The Player changes to downloading a smaller, faster video file

Player adapts to slower connection

Player adapts to faster connection

854 x 480 pixels

426 x 240 pixels

426 x 240 pixels

854 x 480 pixels
Simple solution for encoding: use a “bitrate ladders”

<table>
<thead>
<tr>
<th>Bitrate (kbps)</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>235</td>
<td>320x240</td>
</tr>
<tr>
<td>375</td>
<td>384x288</td>
</tr>
<tr>
<td>560</td>
<td>512x384</td>
</tr>
<tr>
<td>750</td>
<td>512x384</td>
</tr>
<tr>
<td>1050</td>
<td>640x480</td>
</tr>
<tr>
<td>1750</td>
<td>720x480</td>
</tr>
<tr>
<td>2350</td>
<td>1280x720</td>
</tr>
<tr>
<td>3000</td>
<td>1280x720</td>
</tr>
<tr>
<td>4300</td>
<td>1920x1080</td>
</tr>
<tr>
<td>5800</td>
<td>1920x1080</td>
</tr>
</tbody>
</table>

[netflix.com]
Your player download \textit{“chunks”} of video at different bitrates
Depending on your network connectivity, your player fetches chunks of different qualities.
Your player gets metadata about chunks via “Manifest”

```xml
<?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="PT0H9M56.46S"
     minBufferTime="PT15.0S">
  <BaseUrl>http://wittestlab.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"/>
        </SegmentBase>
        <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>
```
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
Encoding

Replication

Adaptation
Network
Capacity < current rate ⇒ decrease rate
Common solution approach

- 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
Estimating available capacity

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

“A random sample of 300,000 Netflix sessions shows that roughly 10% of sessions experience a median throughput less than half of the 95th percentile throughput.”

“20–30% of rebuffers are unnecessary”

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

Decide based on the buffer alone?
Buffer-based adaptation

Nearly full buffer $\Rightarrow$ large rate
Buffer-based adaptation

Nearly empty buffer $\Rightarrow$ small rate
Buffer-based adaptation

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

Pick a rate based on immediate past throughput
Summary

- Encode video in multiple bitrates
- Replicate using a content delivery network
- Video player picks bitrate adaptively
- Problem of active research interest, many competing algorithms and objectives
We'll study e-mail from three different perspectives

- **Content**
  - Format: Header/Content
  - Encoding: MIME

- **Infrastructure/Transmission**
  - SMTP: Simple Mail Transfer Protocol
  - Infrastructure mail servers

- **Retrieval**
  - POP: Post Office Protocol
  - IMAP: Internet Message Access Protocol
Format: Header/Content
Encoding: MIME
An e-mail is composed of two parts
A header, 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
Hi Tobias,

Here are some interesting questions...

Best,
Laurent
Type, followed by ":"  

Header

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

Series of lines ending with Carriage Return and Line Feed
Hi Tobias,

Here are some interesting questions...

Best,
Laurent
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**

<table>
<thead>
<tr>
<th>Content-Type</th>
<th>indicates that the message contains</th>
</tr>
</thead>
<tbody>
<tr>
<td>multipart/mixed</td>
<td>multiple independent parts</td>
</tr>
<tr>
<td></td>
<td>e.g. plain text <em>and</em> a binary file</td>
</tr>
<tr>
<td>multipart/alternative</td>
<td>multiple representation of the same content</td>
</tr>
<tr>
<td></td>
<td>e.g. plain text <em>and</em> HTML</td>
</tr>
</tbody>
</table>
MIME defines

- additional headers for the email body
- a set of content types and subtypes
- base64 to encode binary data in ASCII
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 length of the sequence to be encoded is not a multiple of 3
<table>
<thead>
<tr>
<th>Binary input</th>
<th>0x14fb9c03d97e</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-bits</td>
<td>00010100 11111011 10011100 00000011 11011001 01111110</td>
</tr>
<tr>
<td>6-bits</td>
<td>000101 001111 101110 011100 000000 111101 100101 111110</td>
</tr>
<tr>
<td>Decimal</td>
<td>5 15 46 28 0 61 37 62</td>
</tr>
<tr>
<td>base64</td>
<td>F P u c A 9 1 +</td>
</tr>
<tr>
<td>Value</td>
<td>Char</td>
</tr>
<tr>
<td>-------</td>
<td>------</td>
</tr>
<tr>
<td>0</td>
<td>A</td>
</tr>
<tr>
<td>1</td>
<td>B</td>
</tr>
<tr>
<td>2</td>
<td>C</td>
</tr>
<tr>
<td>3</td>
<td>D</td>
</tr>
<tr>
<td>4</td>
<td>E</td>
</tr>
<tr>
<td>5</td>
<td>F</td>
</tr>
<tr>
<td>6</td>
<td>G</td>
</tr>
<tr>
<td>7</td>
<td>H</td>
</tr>
<tr>
<td>8</td>
<td>I</td>
</tr>
<tr>
<td>9</td>
<td>J</td>
</tr>
<tr>
<td>10</td>
<td>K</td>
</tr>
<tr>
<td>11</td>
<td>L</td>
</tr>
<tr>
<td>12</td>
<td>M</td>
</tr>
<tr>
<td>13</td>
<td>N</td>
</tr>
<tr>
<td>14</td>
<td>O</td>
</tr>
<tr>
<td>15</td>
<td>P</td>
</tr>
</tbody>
</table>
If the length of the input is not a multiple of three, Base64 uses "=" as padding character.

<table>
<thead>
<tr>
<th>Binary input</th>
<th>0x14</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-bits</td>
<td>00010100</td>
</tr>
<tr>
<td>6-bits</td>
<td>000101 000000</td>
</tr>
<tr>
<td>Decimal</td>
<td>5 0</td>
</tr>
<tr>
<td>base64</td>
<td>F A = =</td>
</tr>
</tbody>
</table>
Hi Tobias, Please find the exam enclosed. Laurent

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

Infrastructure/Transmission

Retrieval

SMTP: Simple Mail Transfer Protocol

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

Ivanbever @ 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

<table>
<thead>
<tr>
<th>Mail</th>
<th>User</th>
<th>Agent</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Use to read/write emails (mail client)</td>
</tr>
<tr>
<td>Mail</td>
<td>Submission</td>
<td>Agent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Process email and forward to local MTA</td>
</tr>
<tr>
<td>Mail</td>
<td>Transmission</td>
<td>Agent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Queues, receives, sends mail to other MTAs</td>
</tr>
<tr>
<td>Mail</td>
<td>Delivery</td>
<td>Agent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Deliver email to user mailbox</td>
</tr>
<tr>
<td>Mail</td>
<td>Retrieval</td>
<td>Agent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fetches email from user mailbox</td>
</tr>
</tbody>
</table>
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)
<table>
<thead>
<tr>
<th>Status</th>
<th>SMTP 3 digit response code</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>2XX</td>
<td>success</td>
<td>220</td>
</tr>
<tr>
<td></td>
<td></td>
<td>250</td>
</tr>
<tr>
<td>3XX</td>
<td>input needed</td>
<td>354</td>
</tr>
<tr>
<td>4XX</td>
<td>transient error</td>
<td>421</td>
</tr>
<tr>
<td></td>
<td></td>
<td>450</td>
</tr>
<tr>
<td></td>
<td></td>
<td>452</td>
</tr>
<tr>
<td>5XX</td>
<td>permanent error</td>
<td>500</td>
</tr>
<tr>
<td></td>
<td></td>
<td>502</td>
</tr>
<tr>
<td></td>
<td></td>
<td>503</td>
</tr>
</tbody>
</table>
server — 220 hamburger.edu
   HELO 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.
Each SMTP server/MTA hop adds its identity to the e-mail header by prepending a "Received" entry
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!

<table>
<thead>
<tr>
<th><strong>SMTP-MTA</strong></th>
<th>telnet server_name 25</th>
</tr>
</thead>
<tbody>
<tr>
<td>plaintext (!), hard to find</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>SMTP-MSA</strong></th>
<th>openssl s_client -starttls smtp -connect mail.ethz.ch:587 -crlf -ign_eof (*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>rely on TLS encryption</td>
<td></td>
</tr>
<tr>
<td>authentication required</td>
<td>perl -MMIME::Base64 -e 'print encode_base64(&quot;username&quot;);' perl -MMIME::Base64 -e 'print encode_base64(&quot;password&quot;);'</td>
</tr>
</tbody>
</table>

(*) 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
Content

Infrastructure/Transmission

Retrieval

POP: Post Office Protocol
IMAP: Internet Message Access Protocol
Content

Infrastructure/Transmission

Retrieval

**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
+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
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
Content

Infrastructure/Transmission

Retrieval

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).