Computer Networks (ComNet) 2/5 : **Application**

O. Fourmaux - T. Friedman

Version 8.3

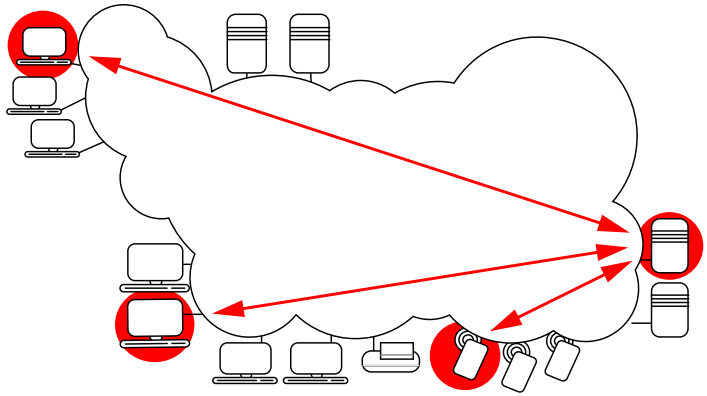
ComNet: course 2/5 outline

1. Legacy applications
   - Introduction
   - Remote login
   - File transfer
2. Contemporary applications
   - Electronic mail
   - World Wide Web
   - multimedia
3. Support applications
   - Directory (DNS)
   - Network management

ComNet: course 2/5 outline

1. Legacy applications
   - Introduction
   - Remote login
   - File transfer
2. Contemporary applications
   - Electronic mail
   - World Wide Web
   - multimedia
3. Support applications
   - Directory (DNS)
   - Network management

Applications



Application layer

Definition

The application layer: All high-level programs and protocols that allow users to communicate

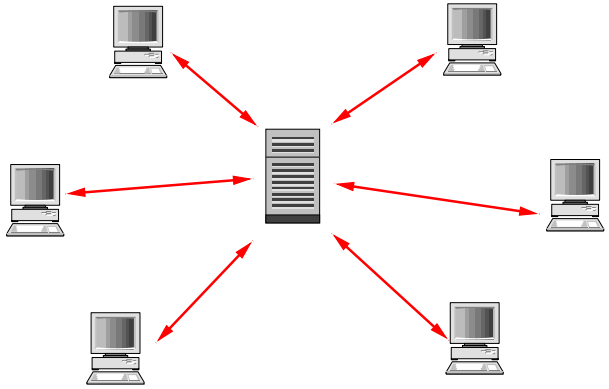
Remarks:

- standardizes the exchanges of the most common applications
  - web access (HTTP), e-mail (SMTP, POP, IMAP) ...

  application ≠ application layer protocol
- defines the user-network interface
  - relies on the end-to-end services defined in the lower layers
- supports heterogeneous environments

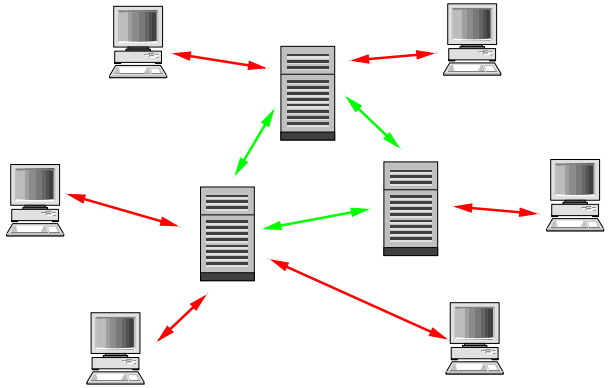
Architectures (1)

Classical centralised client/server



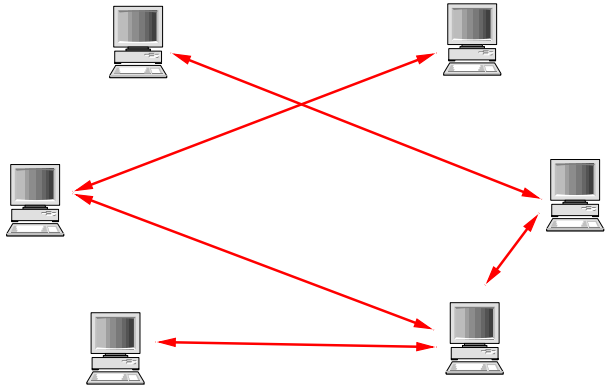
Architectures (2)

Client/server with server replication

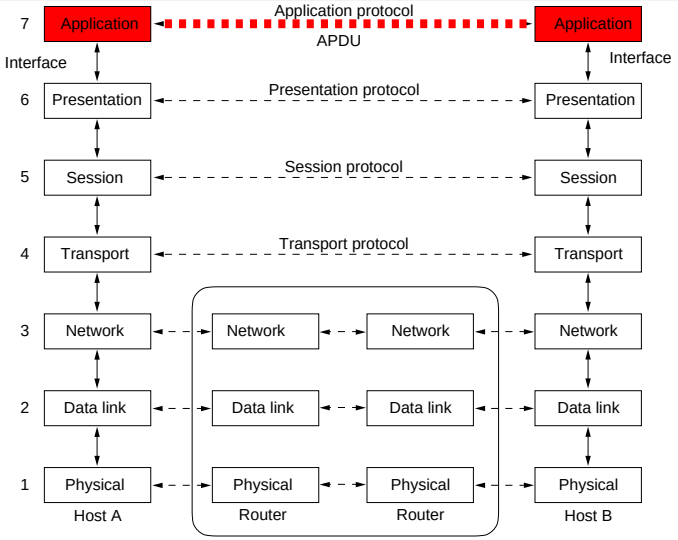


Architectures (3)

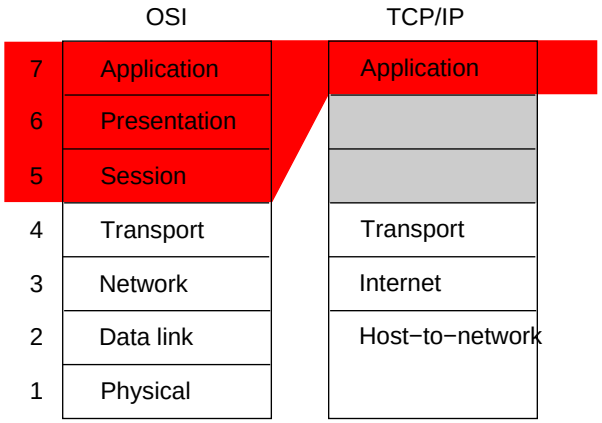
Classical peer-to-peer



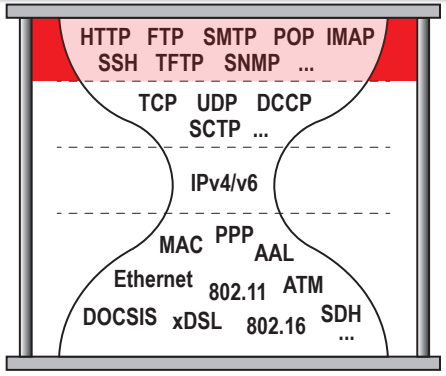
Application layer: OSI model



Application layer: TCP/IP model (1)



Application layer: TCP/IP model (2)



In the Internet, there are hundreds of application layer protocols!

- HTTP to surf the web
- FTP to transfer files
- SMTP to exchange e-mail...

ComNet: course 2/5 outline

1. Legacy applications
   - Introduction
   - Remote login
   - File transfer
2. Contemporary applications
   - Electronic mail
   - World Wide Web
   - multimedia
3. Support applications
   - Directory (DNS)
   - Network management

Remote login applications

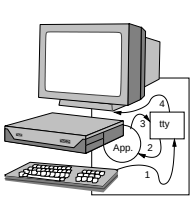
From a terminal open on local host, connect to a remote host

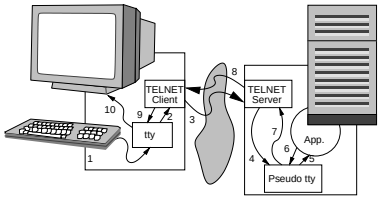
- several protocols:
  - TELNET
  - SSH...
- client/server applications
  - **client**: interacts with the user and the network protocols
  - **server**: interacts with the network protocols and an always-running process
- must be interactive
  - everything typed on the local keyboard is **quickly** sent over the network connection
  - everything received from the connection is **quickly** displayed on the local screen

TELNET (*TELecommunication NETwork protocol*)

Application present since 1969 (RFC 15) and standardized by the IETF in 1983 (RFC 854 and Internet Standard STD 8)

- based on a **TCP** connection (server port n° **23**)
- optional negotiation mechanisms
- virtual terminal service
- no confidentiality (password in the **clear**...)





TELNET: options

Several setup exchanges dedicated to options (RFC 855):

- the client sends **requests** (`WILL WON'T DO DON'T`)

```
Command: Do Suppress Go Ahead
Command: Will Terminal Type
Command: Will Negotiate About Window Size
Command: Will Terminal Speed...
```

- the server answers with **replies** (`DO DON'T WILL WON'T`)

```
Command: Do Terminal Type
Command: Will Suppress Go Ahead
Command: Dont Negotiate About Window Size
Command: Do Terminal Speed...
```

- each end-point implements basic NVT functionality
  - option negotiation for more sophisticated hosts

TELNET: NVT

Network Virtual Terminal (NVT)

- describes a standard way to encode data
  - all end-hosts can encode/decode

    local real terminal ⇔ network virtual terminal
  - Example:
    - *local:* `cc maa<bs>x.c`
    - *NVT:* `c` `.` `x` `IAC EC` `a` `a` `m` ␣ `c` `c` ⏎
      IAC = *Interpret As Command (byte value 255)*
    - it is not necessary to know how to convert for all types of machines
  - enables communication in **heterogeneous environments**
  - **in-band** control

TELNET: accessing various servers

Example of web server access via TELNET

```
Unix> telnet hobbes.lip6.fr 80
          Trying 137.86.111.77...
          Connected to hobbes.lip6.fr.
          Escape character is '^]'.
GET /index.html HTTP/1.0
          HTTP/1.1 200 OK
          Date: Tue, 24 Sep 2002 15:33:07 GMT
          Server: Apache/1.3.9 (Unix) Debian/GNU
          Connection: close
          Content-Type: text/html; charset=iso-8859-1

          <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
          <HTML>
          ...
          Connection closed by foreign host.
```

➠ raw TCP connection (NVT limitation)

SSH (*Secure SHell*)

- **encrypted communication**, ensuring:
  - authentification
  - privacy
  - integrity
- **TCP-based** connection (server port n° **22**)
  - adds an intermediate transport layer
  - encrypted authentification
  - algorithms are negotiated
  - (sessions, tunnels muxed: X11, port forwarding, SOCKS...)
- recently standardized (Jan. 2006): RFCs 4251-4254
- multiple implementations
  - OpenSSH (native on BSDs, GNU/Linux, MacOSX, Cygwin...)
  - PuTTY (Windows and Unixes)...

ComNet: course 2/5 outline

1. Legacy applications
   - Introduction
   - Remote login
   - File transfer
2. Contemporary applications
   - Electronic mail
   - World Wide Web
   - multimedia
3. Support applications
   - Directory (DNS)
   - Network management

Classical file transfer applications

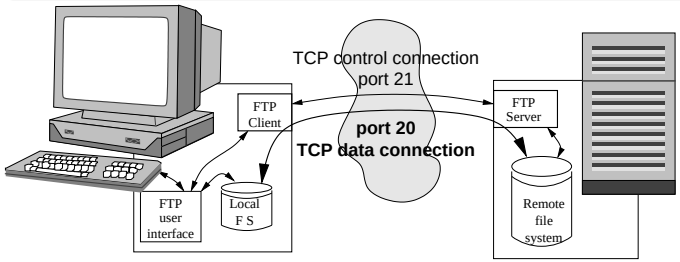
Copy a file from one system to another in a heterogeneous environment

- several protocols:
  - FTP
  - TFTP
  - RCP, SCP, SFTP,...
- client/server style of application
  - **client**: interacts with the user, the local filesystem, and the network protocols
  - **server**: interacts with the network protocols and the remote filesystem
- not to be confused with network filesystems
  - NFS (Sun, TCP/IP), SMB (Microsoft),...

FTP (*File Transfer Protocol*)

The standard TCP/IP file transfer application (RFC 959)

- **out-of-band** signaling, two **TCP** connections
- **interactive access**
- access control (but password **in clear type**)



FTP: connections

Two parallel **TCP** connections:

- **control** connection
  - **permanent** (created at the start of the FTP session)
  - full duplex, initiated by the client (server port n° **21**)
  - purely for exchanging **requests** and **responses**
  - requires **interactivity** (and reliability)
- **data transfer** connection
  - **temporary** (created for each file transfer)
  - full duplex, initiated by the **server**
    - **client port number** communicated ahead of time
  - send both files and lists of files/directories
  - requires **bandwidth** (and reliability)
  - released at the end of each file transfer

FTP: data

Many ways to represent data (heterogeneous hosts):

- file structures:
  - **unstructured**
  - record structure
  - page structure
- data types:
  - **ASCII** (*American Standard Code for Information Interchange*)
  - EBCDIC (*Extended Binary-Coded Decimal Interchange Code*)
  - **binary**
- transmission modes:
  - **stream**
  - block
  - compressed

➠ **verify the type of data being transferred**

FTP: requests

ASCII NVT encoding ➠ printable, so interactive is possible

```
Unix> telnet galion.ufr-info-p6.jussieu.fr 21
  Trying 197.18.176.12...
  Connected to localhost.
  Escape character is '^]'.
  220 ProFTPD 1.2.0pre10 Server (Debian) [galion.ufr-info-p6.jussieu.fr]
help
  214-The following commands are recognized (* =>'s unimplemented).
  214-USER    PASS    ACCT*   CWD     XCWD    CDUP    XCUP    SMNT*
  214-QUIT    REIN*   PORT    PASV    TYPE    STRU*   MODE*   RETR
  214-STOR    STOU*   APPE    ALLO*   REST    RNFR    RNTO    ABOR
  214-DELE    MDTM    RMD     XRMD    MKD     XMKD    PWD     XPWD
  214-SIZE    LIST    NLST    SITE    SYST    STAT    HELP    NOOP
  214 Direct comments to root@galion.ufr-info-p6.jussieu.fr.
```

Do not confuse the control commands with the `ftp` user interface ones

ftp software user commands

```
Unix> ftp
ftp> help
  Commands may be abbreviated.  Commands are:
  !             debug         mdir          sendport      site
  $             dir           mget          put           size
  account       disconnect    mkdir         pwd           status
  append        exit          mls           quit          struct
  ascii         form          mode          quote         system
  bell          get           modtime       recv          sunique
  binary        glob          mput          reget         tenex
  bye           hash          newer         rstatus       tick
  case          help          nmap          rhelp         trace
  cd            idle          nlist         rename        type
  cdup          image         ntrans        reset         user
  chmod         lcd           open          restart       umask
  close         ls            prompt        rmdir         verbose
  cr            macdef        passive       runique       ?
  delete        mdelete       proxy         send
```


FTP: replies

Standard encoding: *status code + text*

status	*description*	*status*	*description*
		x0z	Syntax
1yz	Positive Preliminary reply	x1z	Information
2yz	Positive Completion reply	x2z	Connections
3yz	Positive Intermediate reply	x3z	Authentication
4yz	Transient Negative Completion		
5yz	Permanent Negative Completion	x5z	File system

- `150 Opening BINARY mode data connection`
- `200 Command successful`
- `220 ProFTPD 1.2.0pre10 Server (Debian)`
- `226 Transfer complete`
- `230 User toto logged in`
- `331 Username OK, Password required`
- `425 Can't open data connection`
- `500 Syntax error (Unknown command)...`

FTP: example

`ftp` software (user interface)

```
[toto@hobbes]$ ftp calvin.lip6.fr
   Connected to calvin.lip6.fr.
   220 FTPD 1.2pre8 Server (Debian)
Name (calvin.lip6.fr):toto
   331 Password required for toto.
Password:
   230 User toto logged in.
ftp> get toinst.txt
   local: toinst.txt remote: toinst.txt

   200 PORT command successful.

   150 Opening BINARY mode data connection
       for toinst.txt (1 bytes).
   226 Transfer complete.
   1 bytes received in 0.377s (0.0026 KB/s)
ftp> quit
   221 Goodbye.
[toto@hobbes]$
```

FTP protocol (control connection)

```
   220 FTPD 1.2pre8 Server (Debian)
USER toto
   331 Password required for toto.
PASS AB]Ga!9F
   230 User toto logged in.


PORT 192,33,82,12,4,15
   200 PORT command successful.
RETR toinst.txt
   150 Opening BINARY mode data connection
       for toinst.txt (1 bytes).
   226 Transfer complete.

QUIT
   221 Goodbye.
```


FTP: miscellaneous

Anonymous: guest account on certain FTP servers:

- `username: anonymous`
- `password:` *e-mail@address.org*

Passive mode: reverse direction data connection

- if it is impossible to open a normal **data connection**
  - address filtering (*firewall*)
  - address translation (NAT)
- the `PASV` command (RFC 1579)
  - the client sends the `PASV` command to the server `a.b.c.d`
  - the server allocates port 256x+y, opens this port in passive mode, and informs the client via a reply
    ➠ `227 Entering passive mode (a,b,c,d,x,y)`
  - the client opens an active connection towards port 256x+y

TFTP (*Trivial File Transfer Protocol*)

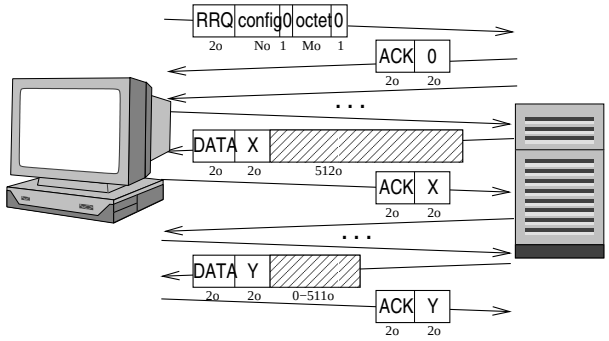
- lightweight file transfer protocol (version 2: RFC 1350)
- **UDP** datagrams sent to port 69
- 5 messages:

opcode	name	description
1	RRQ	Read request
2	WRQ	Write request
3	DATA	Data
4	ACK	Acknowledgement
5	ERREUR	Error

- DATA messages each contain 512 bytes (except the last one, which is smaller or empty)
- *stop-and-wait* protocol
  - numbered DATA messages
  - immediate acknowledgement with ACK
- no access control (in Unix, often limited to `/tftpboot`)

TFTP: example

```
[toto@hobbes]$ tftp calvin.lip6.fr
tftp> get config
   Received 5220 bytes in 0.377 secs
tftp> quit
[toto@hobbes]$
```



RCP, SCP, SFTP

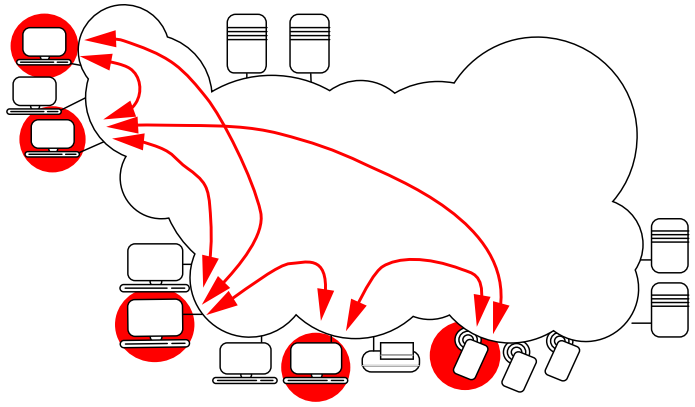
RCP: `rcp`

- A **Berkeley "r" command** (along with `rlogin`, `rsh`,...)
  - the `rcp` client works with the `rshd` server
  - idem `rlogin`: obsolete, security problems,...

Secure protocols: **SCP, SFTP**

- `scp`: copies files like `rcp`, but encapsulated in SSH
- `sftp`: like FTP, but easily encapsulated
  - SFTP is a new protocol (IETF IPSEC working group)
  - SFTP can work over SSH (the default for many `sftp` clients)
  - SFTP is not the same as FTPS, which secures connections with SSL/TLS *(Secure Socket Layer/Transport Layer Security)*

Peer-to-peer file sharing applications



Numerous peer-to-peer applications...

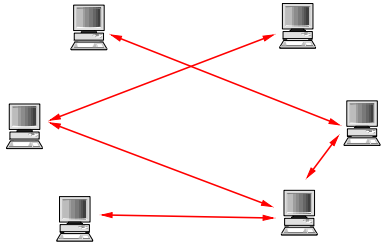
... but few standards

- **P2P file sharing**
  - Napster, eDonkey, BitTorrent...
  - FastTrack (KaZaA, Grokster et Imesh), Gnutella2...
  - Gnutella...
  - BitTorrent
- **audio/video streaming**
  - VoD : Peercast, Joost...
  - P2PTV : Coolstreaming, TVants, PPLive...
- **anonymous storage**
  - Freenet, Entropy...
- **distributed computing**
- ...

P2P features

Characteristics of *peer-to-peer* systems

- no clear separation of roles
  - avoid bottlenecks or defective nodes
  - need for distributed algorithms
    - service discovery (naming, addressing, metric calculation)
    - maintaining neighbor state
    - application-layer routing (linked to content)
    - robustness, management of link or node failure...



Napster

Program for sharing MP3 files

- not the first, but the best-known
  - highly informative regarding the significance of peer-to-peer systems...
    ...along technical, legal, political, and economic lines
- a disruptive technology?
  - timeline
    - end 98: Shawn Fanning (19 years old) starts developing the system
    - 05/99: creation of the *Napster Online Music Service*
    - 06/99: initial testing of the application
    - 12/99: first lawsuits (Metallica, RIAA...)
    - mid 00: more than **60M** users
      a significant portion of university traffic (30% to 50%)
    - 02/01: US 9th Circuit Cout of Appeals ruling
    - mid 01: 160K users...

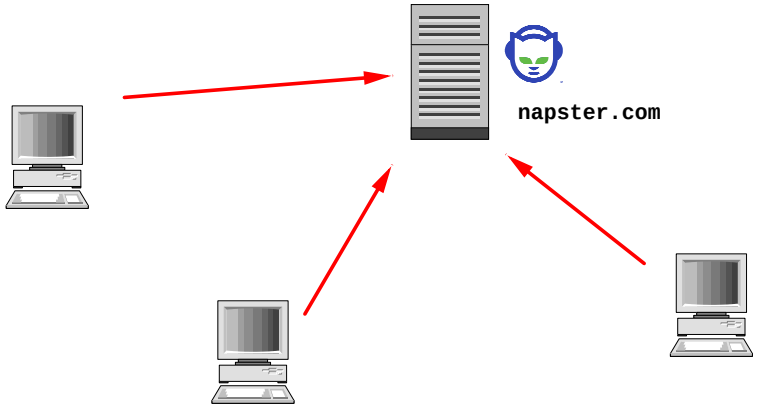
Napster: basic design

Combined approach

- client/server based search with a central list
- direct exchange of files between peers
- point-to-point TCP connections (port 7777 or 8888)
- 4 steps:
  - Connection to the Napster server
  - Peer sends its file list to the server (*push*)
  - Peer sends keywords and retrieves a list of peers
  - Selection of the best peer (*pings*)

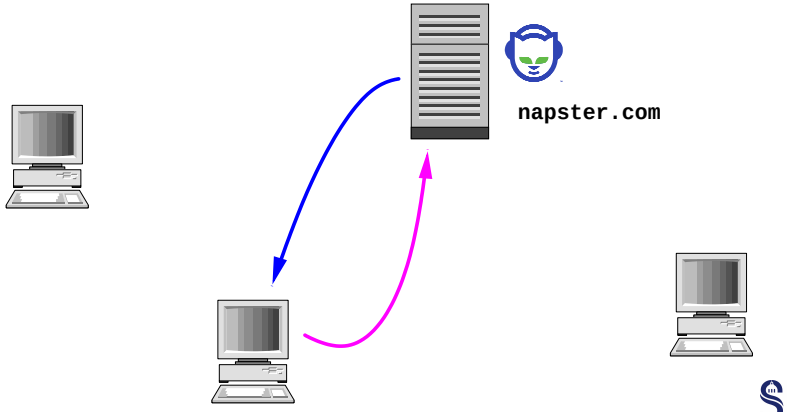
Napster: *Upload*

Peers send their lists of files to share



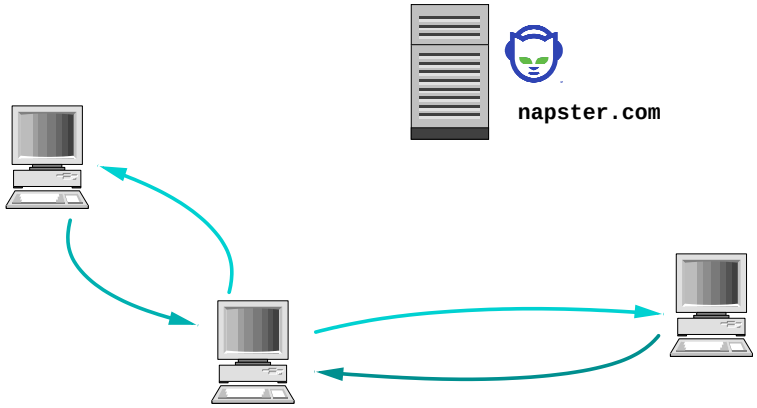
Napster: *Search*

A peer sends a search request
The server replies with a list of possible locations



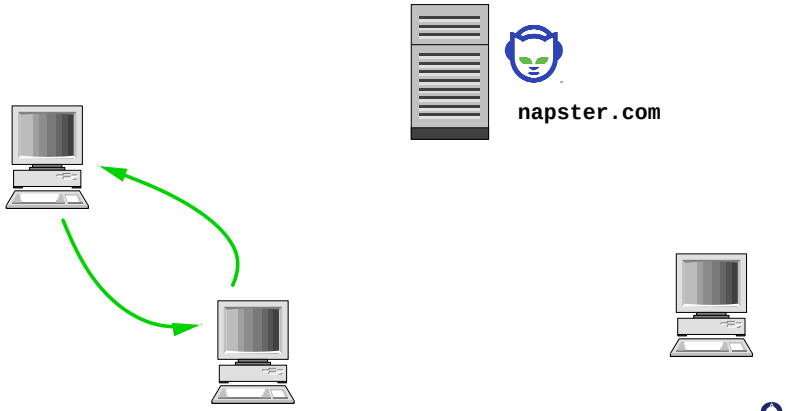
Napster: *Pings*

Pinging possible peers (searching for the best bandwidth)



Napster: *Download*

The peer obtains the file directly from its chosen peer



Napster: remarks

- **centralized** server
  - single point of failure
  - congestion risk
  - load balancing using round-robin DNS
  - enterprise-controlled
- no **security**
  - clear-type password
  - no authentication
  - no anonymity
  - proprietary codebase
  - automatic download of updates

Gnutella: background

Completely distributed file sharing

- corrects the perceived defects of Napster
  - *Open source*
  - no servers; no global index
  - purely local knowledge
- but still the same legal and economic issues...
  - no direct responsibility for the service
  - lacks anonymity
    - RIAA goes after the users directly!

Gnutella: background (2)

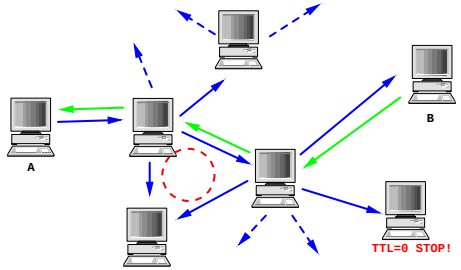
Peer-to-peer networking

- direct connection between peer applications
- problem:
  - **decentralized** file search
- each application:
  - stocks a variety of files
  - routes queries to and from its peers
  - serves file transfer requests
- timeline
  - 03/00 *freelance* project abandoned after just a few days by AOL (Nullsoft)
  - too late: already more than 20K users...
  - nombreux correctifs facilités par l'approche *open source*

Gnutella: principle

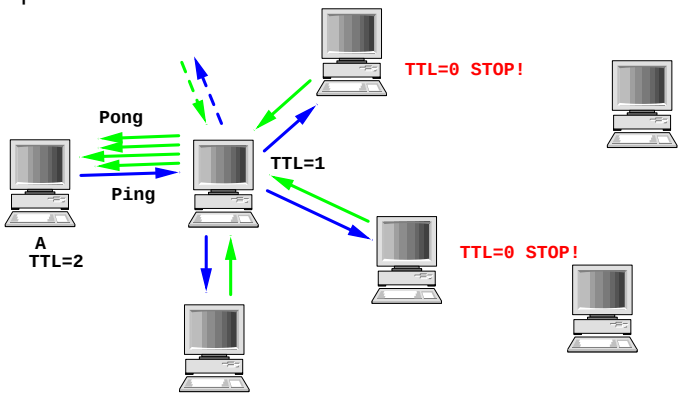
Search by **flooding**

- if I do not have the requested file:
  - I ask 7 peers if they have the file
- if they don't have it, they each contact 7 of their neighboring peers
  - recursive search limited to N hops
- loop detection through temporary memorization of requests
  - messages can be received twice



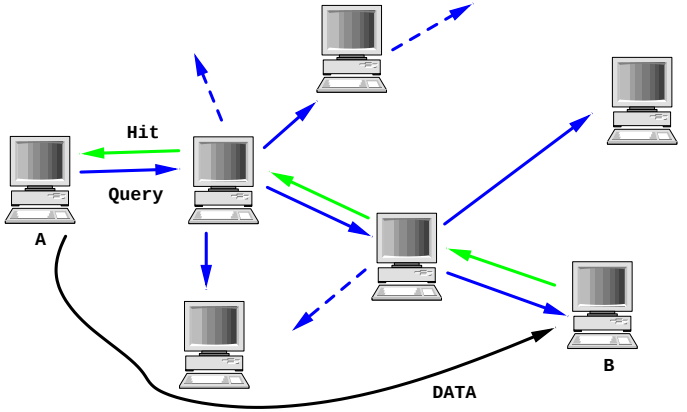
Gnutella: identifying peers

Active peer detection



Gnutella: file search

Request to obtain information



Gnutella: remarks

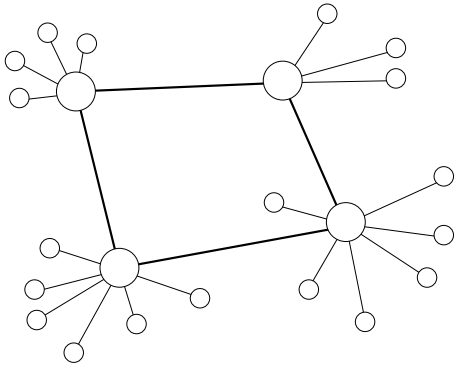
Lessons learned:

- saturation of low bandwidth peers (modems)
  - ability to indicate that one has a file but that one is saturated
- reachable network size is limited (connections broken due to modems)
  - creation of a hierarchy of peers
- **anonymity?**
  - the peer from which you retrieve the file knows your identity
    - $\exists$ protocols in which it is possible not to know the receiver

P2P evolution

Gnutella2, **KaZaA** (FastTrack network)...

- heterogeneous hosts
- hierarchical topology
  - *Super-Nodes*



BitTorrent (1)

File sharing:

- file broken into blocks of 64KB to 1MB *(chunks)*
- creation of a `.torrent`
  - metadata
  - signature for each *chunk*
- establishment of a *tracker*
  - machine that oversees the distribution
- data exchange between all requesters *(leechers)*
  - the source *(seed)* only gets the exchange going

Characteristics:

- no search feature
- no direct downloading (HTTP style)
- avantages:
  - inexpensive
  - redundancy
  - supports *flash-crowds*

BitTorrent (2)

Strategies:

- peer selection
  - *tit-for-tat* + *choking*
    - encourage cooperation and reduce *free-riding*
    - choose the best contributors and suppress the others
    - periodic mechanism (10 secs)
  - *optimistic unchoke*
    - discover new peers
    - supply a new peer at random
    - periodic mechanism (30 secs)
- *chunk* selection:
  - *rarest first*
    - provide the rarest *chunk* first
    - maximize the entropy of each *chunk*
  - *random first*
    - speed up the start phase for new peers

BitTorrent (3)

Evolutions :

- indexing/search
  - initially specialized search engines (web)
  - distributed *tracker* (distributed hash table)
    - based on Kademlia
- *multitracker*
  - redundancy
  - signaling overhead
- encrypted exchanges
  - *Protocol header encrypt (PHE)*
  - *Message stream encryption/Protocol encryption (MSE/PE)*
- content distribution *(A/V streaming)*
  - many projects...

P2P: others

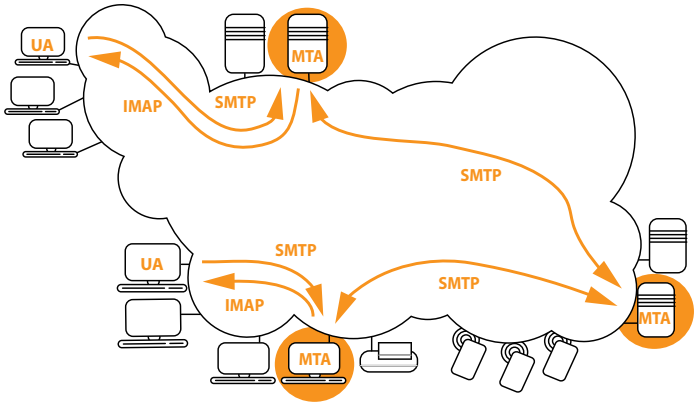
Structured *peer-to-peer* systems for content-based search:

- **Chord**
  - key-based identity (SHA-1 on a string)
  - key-based location (SHA-1 on the node address)
    - storage on the closest successor node
- **Tapestry**
  - route identifiers (hash) according to the node suffix
- **CAN** (*Content Addressable Network*)
  - virtual Cartesian space

ComNet: course 2/5 outline

1. Legacy applications
   - Introduction
   - Remote login
   - File transfer
2. Contemporary applications
   - Electronic mail
   - World Wide Web
   - multimedia
3. Support applications
   - Directory (DNS)
   - Network management

E-mail applications



SMTP: introduction

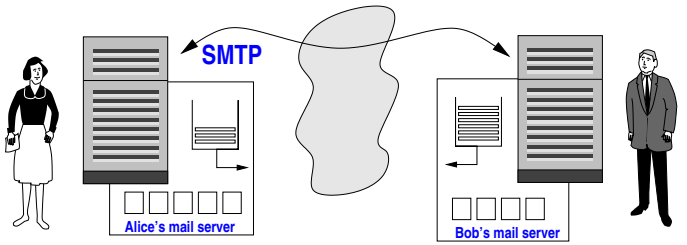
Asynchronous message exchange across the Internet

- the first "*killer app.*"
- three basic elements
  - UA (*User Agent*)
    - `mail`, `elm`, `pine`, `mutt`,...
    - Eudora, Outlook and MS Mail, Mail.app, Mozilla Thunderbird,ldots
  - mail server, or MTA (*Mail Transfer Agent*)
    - `sendmail`...
    - makes up the **infrastructure** of the mail distribution system
    - local user **mailboxes**
    - queue of outgoing messages or messages in transit
    - timeout and **retry** if the destination is unreachable
  - a protocol: **SMTP**

SMTP: overview

Simple Mail Transfer Protocol (RFC 821 - STD 10, updated by RFC 5321)

- client/server application
- based on reliable **TCP** connections
- old
  - ✔ widely used
  - ✘ messages coded in ASCII NVT
- connection to mail servers on port **25**



SMTP: example

```
   220 hobbes.lip6.fr SMTP Sendmail 8.9.3; Wed, 22 Sep 2008 00:59:49 +0200
HELO calvin.lip6.fr
   250 hobbes.lip6.fr Hello calvin.lip6.fr, pleased to meet you
MAIL FROM: pere-noel@hobbes.lip6.fr
   250 pere-noel@hobbes.lip6.fr... Sender ok
RCPT TO: totu@hobbes.lip6.fr
   550 totu@hobbes.lip6.fr... User unknown
RCPT TO: toto@hobbes.lip6.fr
   250 toto@hobbes.lip6.fr... Recipient ok
DATA
   354 Enter mail, end with "." on a line by itself
Cher Toto,
N'oubliez pas de m'envoyer votre liste de cadeaux
                  Le Pere Noel.
.
   250 BAA01090 Message accepted for delivery
QUIT
   221 hobbes.lip6.fr closing connection
```


SMTP: commands (1)

SMTP server interactive mode

```
Unix> telnet galion.ufr-info-p6.jussieu.fr 25
   Trying 192.133.82.123...
   Connected to galion.ufr-info-p6.jussieu.fr
   Escape character is '^]'.
   220 galion.ufr-info-p6.jussieu.fr SMTP Sendmail 8.9.3; Wed, 25 Sep 2002 00:
help
   214-This is Sendmail version 8.9.3
   214-Topics:
   214-    HELO    MAIL    RCPT    DATA
   214-    QUIT    VRFY    NOOP    RSET
   214-    HELP
   214-For more info use "HELP <topic>".
   214-To report bugs in the implementation send email to
   214-    sendmail-bugs@sendmail.org.
   214-For local information send email to Postmaster at your site.
   214 End of HELP info
```


SMTP: commands (2)

Basic SMTP commands (RFC 821), minimum implementation:

HELO	Communicates the client's domain name
MAIL	Specifies the e-mail address of the sender
RCPT	Specifies the e-mail address of the receiver
DATA	Starts content transfer, until a line with only "."
QUIT	Closes the connection
VRFY	Confirms that the receiver address is valid
NOOP	No operation, checks that the server is alive
RSET	Cancels the current transaction

SMTP: replies

Standard text-based encoding

- *status* + *description*:
  - 220 SMTP Sendmail 8.9.3
  - 221 Closing connection
  - 250 Command successful
  - 354 Enter mail, end with "." on a line by itself
  - 550 User Unknown

SMTP: legacy message structure

Messages encoded via ASCII NVT (RFC 822, updated RFC 2822)

- the **envelope**
  - updated by each successive SMTP entity
    - `MAIL FROM:` and `RCPT TO:` comands
- the **message**
  - mainly inserted by the user agent
    - `DATA` command
  - **header**
    - each field on one line ➠ *name*: *value*

```
From: Toto at Paris 13 <toto@galere.univ-paris13.fr>
Date: Mon, 22 Sep 2003 01:13:20 +0200
To: Titi at Paris 6 <titi@hypnos.lip6.fr>
Subject: rapport TER
X-Scanned-By: isis.lip6.fr
```

  - a blank line
  - **body**
    - end with a line containing only "."

Evolution of the envelope: ESMTP

Some ESMTP (RFC 1425, updated RFC 5321) commands:

EHLO	Signals use of ESMTP and introduces the client
SIZE	Declares the maximum message size
8BITMIME	Signals possibility of 8 bit coding
X???	Local SMTP extensions

Negotiation of ESMTP extensions:

```
EHLO hobbes.lip6.fr.
250-hobbes.lip6.fr Hello [62.62.169.227], pleased to meet you
250-PIPELINING
250-SIZE 35000000
250-VRFY
250-STARTTLS
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
```


Evolution of the header structure

Non-ASCII characters in the header:

=?*charset*?*encode*?*encoded-text*?=

- *charset*: `us-ascii`, `iso-8859-`*x*, ...
- *encode*: le coded form of the text must still be ASCII NVT
  - **Quoted-printable** (`Q`) for Latin character sets:
    - characters > 128 ➠ encoded with 3 characters (= hex coding)
    - space character ➠ always =20
  - **Base64** (`B`):
    - three bytes of text (24 bits) ➠ encoded with 4 ASCII chars.
    - 6 bit values (0, 1, 2... 63) ➠ `ABC...YZab...yz01...9+/`
    - padded with "=" to complete 4 characters
- *encoded-text*:

```
=?iso-8859-2?Q?Igen,=20k=F6sz=F6n=F6m?=
=?iso-8859-1?B?QnJhdm8sIHZvdXMgYXZleiBy6XVzc2kgIQo=?=
```


MIME *(Multipurpose Internet Mail Extensions)*

New MIME headers (RFC 2045 and RFC 2046)

- `Mime-Version: 1.0`
- `Content-Type:` *type*/*sous-type*;*parametres*
  - **simple**: `text/plain; charset="ISO-8859-1"`
    - `text/html`, `image/jpeg`,...
  - **structured**: `multipart/mixed; Boundary=hjfdskjhfdshf`
    - each message block begins with: `hjfdskjhfdshf`
    - nesting is possible
- `Content-Disposition:` presents a block (RFC 2183)
- `Content-Transfer-Encoding:` coding of the block
  - 7 bits is compatible with legacy MTAs (RFC 821)
    - `7bit` (ASCII NVT)
    - `quoted-printable` (recommended for all text)
    - `base64` (recommended for byte streams)
  - 8 bits if the `8BITMIME` command is accepted
    - `8bit` et `Binary` (lines or blocks of 8 bit data)

MIME: types and subtypes

/etc/mime.types

```
application/mac-binhex40
application/msword
application/octet-stream
application/postscript
application/vnd.hp-PCL
application/vnd.ms-excel
application/x-debian-package
application/x-doom
application/x-gnumeric
application/x-java-applet
application/x-javascript
application/x-msdos-program
application/x-tar

audio/basic
audio/midi
audio/mpeg
audio/x-wav

image/jpeg
image/png
image/tiff

message/delivery-status
message/external-body
message/http
message/partial
message/rfc822

multipart/alternative
multipart/digest
multipart/encrypted
multipart/mixed
multipart/parallel
multipart/signed

text/html
text/plain
text/richtext
text/rtf
text/xml
text/x-java
text/x-tex
text/x-vcard

video/mpeg
video/quicktime
video/x-msvideo
```

SCIENCES SORBONNE UNIVERSITÉ

ESMTP: MIME message example

```
From: Olivier Fourmaux <olivier.fourmaux@lip6.fr>
Date: Wed, 20 Feb 2002 01:21:01 +0100
To: Toto <toto@free.fr>
Subject: Document no 3.02
Mime-Version: 1.0
Content-Type: multipart/mixed; boundary="/9DWx/yDrRhgMJTb"
Content-Disposition: inline
Content-Transfer-Encoding: 8bit
User-Agent: Mutt/1.2.5i

--/9DWx/yDrRhgMJTb
Content-Type: text/plain; charset=iso-8859-1
Content-Disposition: inline
Content-Transfer-Encoding: 8bit

Voici le document secret que vous m'avez demandé...

--/9DWx/yDrRhgMJTb
Content-Type: application/pdf
Content-Disposition: attachment; filename="sujet-exam-RES.pdf"
Content-Transfer-Encoding: base64

JVBERi0xLjIKJcfsj6IKNSAwIG9iago8PC9MZW5ndGggNiAwIFIvRmlsdGVyIC9GbGF0ZURl
Y29kZT4+CnN0cmVhbQp4nO1dy7YdtRGd3684Mx6L07T63ZkBdghgXvYlJF1MHNsYm+sHhkCS...
```

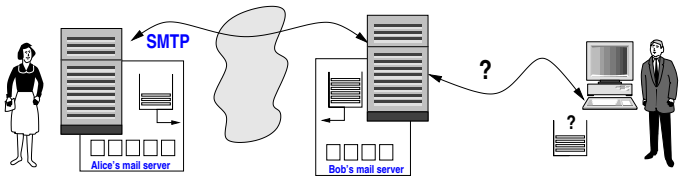
SCIENCES SORBONNE UNIVERSITÉ

Completing message delivery

Machine only connects to the network sporadically?

➠ Messages stored on the last MTA (e.g., that of the ISP)

- several possibilities:
  - direct server access (mount an NFS or SMB volume)
  - POP
  - IMAP
  - HTTP

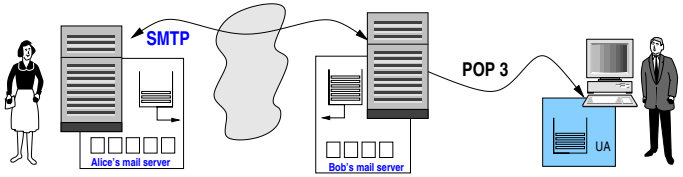


SCIENCES SORBONNE UNIVERSITÉ

POP3

Post Office Protocol – Version 3 (RFC 1939)

- simple
- TCP connection on port 110
- three states:
  - AUTHORIZATION (client identification)
  - TRANSACTION (client requests and server actions)
  - UPDATE (server releases resources)

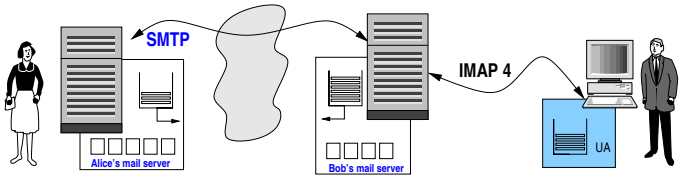


SCIENCES SORBONNE UNIVERSITÉ

IMAP4

Internet Mail Access Protocol – version 4rev1 (RFC 3501)

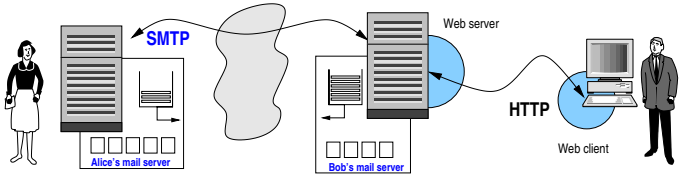
- complex
- TCP connection on port 143
- same functionality as POP, with:
  - attribute-based access (12^{th} e-mail from Alice)
  - partial message retrieval (3^{rd} attachment)
  - **synchronization** of mailboxes



Web-mail

Accessing the SMTP server via a web interface

- various web-based services:
  - *Hotmail*, *Yahoo!*, *GMail*...
- an alternative means to access corporate or ISP e-mail:
  - *horde/IMP*, *Squirrelmail*, *Zimbra*...



E-mail and security

The underlying protocols are not secure

- text-based exchanges lacking confidentiality (control or data)
- no authentication under SMTP
- username and password in clear type over POP and IMAP

Some solutions:

- PGP *(Pretty good privacy)* in unfriendly environments:
  - authentication, integrity, and confidentiality (signed and/or encrypted data)
  - **OpenPGP** (RFC 2440): GPG *(Gnu Privacy Guard)*
- securing the connection to a trusted remote site:
  - if the site is accessible via SSH
    - remote access via SSH (text-based), **SSH tunnels**
- if the client and **server** support TLS
  - SMTP+STARTLS (RFC 3207), POP3S and IMAPS (RFC 2595)
  - HTTPS for securing web-mail...

ComNet: course 2/5 outline

1. Legacy applications
   - Introduction
   - Remote login
   - File transfer
2. Contemporary applications
   - Electronic mail
   - World Wide Web
   - multimedia
3. Support applications
   - Directory (DNS)
   - Network management

World Wide Web

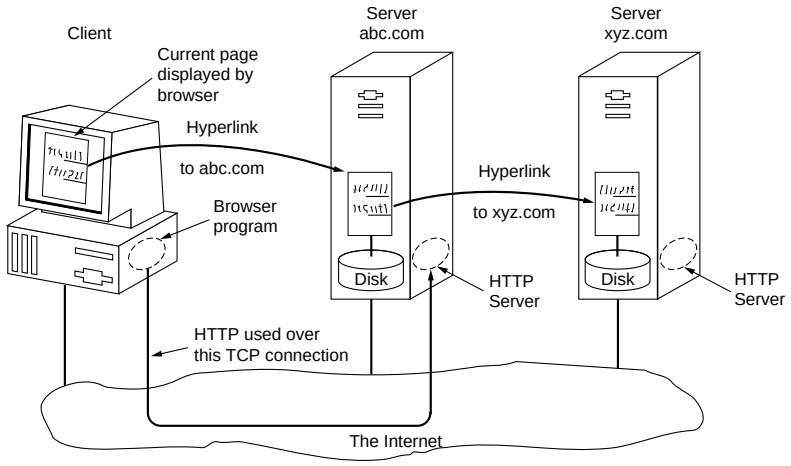
→ 90's: Internet = pure academic network

90's →: **World Wide Web**

- simple and intuitive (graphic) data access system
- developed at CERN by Tim Berners-Lee, starting in 1990
- browser:
  - NCSA Mosaic in 1993 (U. Illinois Urbana-Champagne)
    - only 200 sites on the WWW
    - pioneered integrated images
    - exponential popularity growth!
  - Netscape Navigator in 1994 (➠ Mozilla **Firefox** in 1998)
  - Microsoft Internet Explorer in 1995 (start of the *browser wars*)
  - and many more (see the W3C website)
- server (*web server*):
  - NCSA `httpd` Web Server (➠ **Apache** in 1998)
  - Microsoft IIS *(Internet Information Service)* in 1995

➠ one protocol: **HTTP**

HTTP: overview



pictures from Tanenbaum A. S. *Computer Networks 3rd edition*

HTTP: terminology

- a **web page** or **document** is made up of objects
  - text files in HTML or XHTML
  - GIF, JPEG,... images
  - Java applets
  - ...
- a document is generally an **HTML file** that refers to other objects using URLs
  - **HTML** (*HyperText Markup Language*) is a structured markup language for describing documents that contain hyperlinks that are identified by URLs
  - a **URL** (*Uniform Resource Locator*) identifies a protocol to be used in order to retrieve an object, and a location at which to retrieve it
    - `http://www.lip6.fr/info/linux.html`
    - `ftp://ftp.lip6.fr/pub/linux/disrib/debian/ls-lR.txt`
    - `file:/public/image/penguin.jpeg`
    - `mailto:timur.friedman@upmc.fr`

HTTP: protocol

HyperText Transfer Protocol

- TCP connection on port 80
- defined exchanges:
  - **requests** for objects (client ⇒ server)
  - **answers** including requested objects (server ⇒ client)
  - no stored state in the server (**stateless protocol**)
- HTTP versions:
  - → '97 **HTTP/1.0** (RFC 1945)
    - **non-persistent** connections, one connection per object, high overhead and latency (TCP *3-way handshake* and *slow start*)
  - '98 → **HTTP/1.1** (**RFC 7230 to 7237**, ex-RFC 2616)
    - greater compatibility, **persistent** connections, **pipelined** requests possible, chunked transfer...
  - 2015 → **HTTP/2** (RFC 7540)
  - 2022 → **HTTP/3** (RFC 9114)

HTTP: example

Browser:

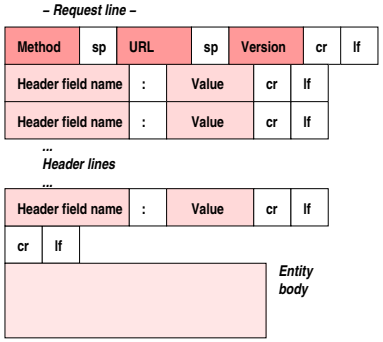
```
GET /index.html HTTP/1.1
Connection: Keep-Alive
User-Agent: Mozilla/4 [en] (X11; I; Linux 0.99 i486)
Host: calvin.lip6.fr
Accept: image/gif, image/jpeg, image/png, */*
Accept-Encoding: gzip
Accept-Language: fr-FR, fr, en
Accept-Charset: iso-8859-1,*,utf-8
```

Web server:

```
HTTP/1.1 200 OK
Date: Tue, 24 Sep 2002 12:59:28 GMT
Server: Apache/1.3.9 (Unix) Debian/GNU
Last-Modified: Sat, 29 Apr 2000 07:07:45 GMT
ETag: "1382c-ffe-390a8a41"
Accept-Ranges: bytes
Content-Length: 4094
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
   <META HTTP-EQUIV="Content-Type" CONTENT="tex...
   <META NAME="GENERATOR" CONTENT="Mozilla/4.05...
   <META NAME="Author" CONTENT="johnie@debian.o...
   <META NAME="Description" CONTENT="The initia...
   <TITLE>Welcome to Your New Home Page!</TITLE>
</HEAD>
<BODY TEXT="#000000" BGCOLOR="#FFFFFF" LINK="#0...
<BR>
<H1>Welcome to Your New Home in Cyberspace!</H1>
<BR>
<IMG SRC="icons/openlogo-25.jpg" ALT="Debian">
<IMG SRC="icons/apache_pb.gif" ALT="Apache"></P>

<P>This is a placeholder page installed by the
<A HREF="http://www.debian.org/">Debian</A>
release of the
<A HREF="http://www.apache.org/">Apache</A> Web
server package, because no home page was installed
on this host. You may want to replace this as soon
as possible with your own web pages, of course....

<BLOCKQUOTE>
This computer has installed the Debian GNU/Linux
operating system but has nothing to do with the
Debian GNU/Linux project.  If you want to report
something about this hosts behavour or domain,
please contact the ISPs involved directly,
<strong>not</strong> the Debian Project. <P>
</BLOCKQUOTE>
.............................
</HTML>
```


HTTP: request structure

Classical message structure



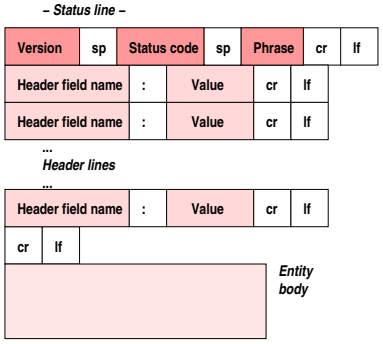
Example

```
GET /index.html HTTP/1.1
Connection: Keep-Alive
User-Agent: Mozilla/4 [en] (X11;...)
Host: calvin.lip6.fr
Accept: image/gif, image/jpeg,  */*
Accept-Encoding: gzip
Accept-Language: fr-FR, fr, en
Accept-Charset: iso-8859-1,*,utf-8
```

- **Method**
  - GET
  - POST (forms)
  - HEAD (page test)...
- **Connection**
  - Close
  - Keep-Alive

HTTP: answer structure

Classical message structure



Example

```
HTTP/1.1 200 OK
Date: Tue, 24 Sep 2002 12:59:28 GMT
Server: Apache/1.3.9 (Unix) Debian/GNU
Last-Modified: Sat, 29 Apr 2000 07:07:45 GMT
Content-Length: 4094
...
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
.............................
</HTML>
```

- *status + description*:
  - 200 OK
  - 301 Move permanently
  - 400 Bad Request
  - 404 Not Found
  - 505 HTTP Version Not Supported ...

HTTP: user authentication

Authentication (**RFC 7235**, ex-RFC 2617)

- 5 methods: simple *(Basic)*, via MD5 signature *(Digest)*...
- example: client page request with basic authentication
  - server responds with an empty page, with header:
    - `401 Authorisation Required`
    - `WWW-Authenticate:` *authorization method details*
  - client request for the same page, with header:
    - `Authorization:` *username password*
  - server response:
    - if OK ➠ the requested page
    - else `401 Authorisation Required`...

all the RFC methods have limitations ➠ non-HTTP mechanism use

HTTP: Conditional GET

1st HTTP request:

```
GET /carte/france.jpg HTTP/1.1
Host: www.atlas.org
```

1st HTTP answer:

```
HTTP/1.1 200 OK
Date: Mon, 2 Oct 2005 23:56:18
Server: Apache/1.3.9 (Unix)
Last-Modified: Sat, 29 Apr 2005 ...
Content-Type: image/jpeg

Données.....................
.....................
.....................
.....................
```

2nd HTTP request:

```
GET /carte/france.jpg HTTP/1.1
Host: www.atlas.org
If-modified-since: Sat, 29 Apr 2005 ...
```

2nd HTTP answer:

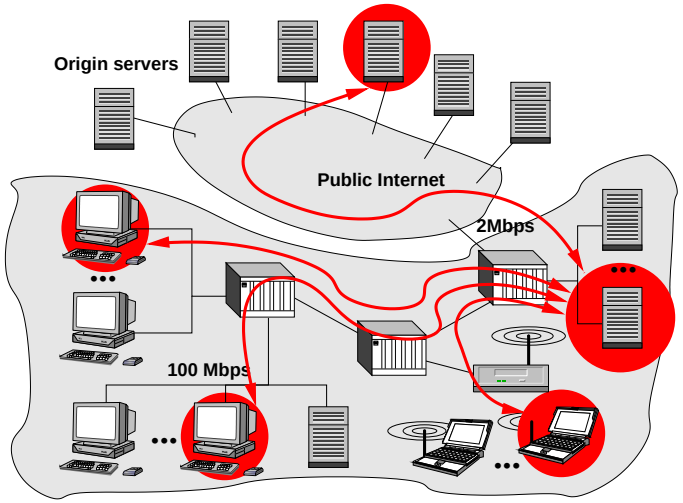
```
HTTP/1.1 304 Not Modified
Date: Mon, 3 Oct 2005 00:06:43
Server: Apache/1.3.9 (Unix) Debian/GNU
```


HTTP: state management mechanism

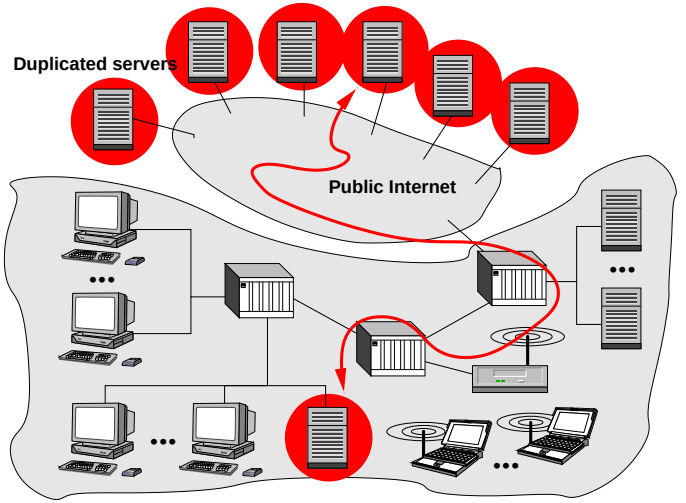
Cookies (**RFC 6265**, ex-RFC 2109/2965)

- identifier associated with a user at a particular machine
- the server identifies the cookie with a header:
  - `Set-cookie:` *id_number*
- the cookie is stored by the client, who, when asking for the same page from the same server, sends it via the header:
  - `Cookie:` *id_number*

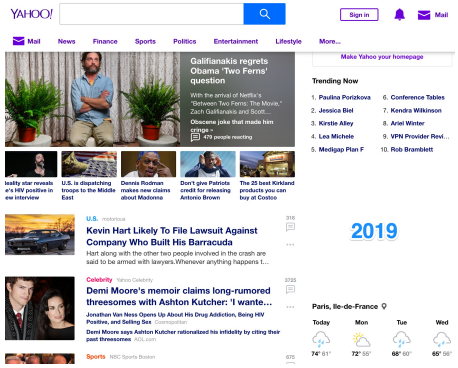
HTTP: cache and proxy



HTTP: CDN



HTTP: evolution du contenu



1995: 17KB, 3 req. ➠ **2019**: 8.8MB (1,2MB JS), 198 req.

HTTP: evolution

HTTP/2 (RFC 7540) and **HTTP/3** (RFC 9114)

- HTTP/1.1 problems:
  - intricate standard
  - pipelining not used
  - to many parallel TCP connections
  - client initiated transfer
  - unefficiant header encoding
- HTTP/2 introduce:
  - multiplexing of several transfer in one TCP connection
  - head of line blocking (HOL) partial avoidance
  - potential server side sending (PUSH)
  - binary protocol
- HTTP/3 introduce:
  - replacing TCP with QUIC/UDP : less connection openning
  - efficient multiplexing of several transfer managed by QUIC

HTTP/2: Example

```
User-Agent: curl/7.31.2
Host: test-http2.lip6.fr
Accept: */*
Connection: Upgrade, HTTP2-Settings
Upgrade: h2c
HTTP2-Settings: AAMAAABkAAQAAP__

   HTTP/1.1 101 Switching Protocols
   Connection: Upgrade
   Upgrade: h2c
   ..............d......

PRI * HTTP/2.0

SM

   .............z.%.P.........f..i................a..Y>...a,j..m@..-..jb.._.I|...M..
   .q.;l..z....a,j..m@.q..6%1h.....[h..Yy‘.I..m........r.(a....r!......
   .-...j
   .!>.*’)SR...J.v..i....T..5.........|.
   .R..Jk..........
   <!DOCTYPE html>
   <head>
      <meta charset="utf-8">
      ...
```


ComNet: course 2/5 outline

1. Legacy applications
   - Introduction
   - Remote login
   - File transfer
2. Contemporary applications
   - Electronic mail
   - World Wide Web
   - multimedia
3. Support applications
   - Directory (DNS)
   - Network management

Multimedia applications

Multimedia applications, such as video-conferencing, involve several types of mechanism:

- session set-up
  - preparing (media, codecs, protocols...)
  - announcing and inviting (news, web, e-mail...)
  - initiation (signaling or connection)
- participation
  - **sending** : encoding/compression, packetizing, sending...
  - **receiving** : decoding/decompression, audio/video readout...

➠ Particular protocols are required for:

- encapsulating multimedia data
- managing participants
- ...

RTP

Real-Time Transport Protocol (version 2, RFC 3550)

- end-to-end transmission of multimedia data
- handles functions required by many applications
  - identification (type of data, participants)
  - packet sequence numbers and timestamps
  - synchronization between flows
  - multimedia quality tracking
  - reaching one or multiple receivers

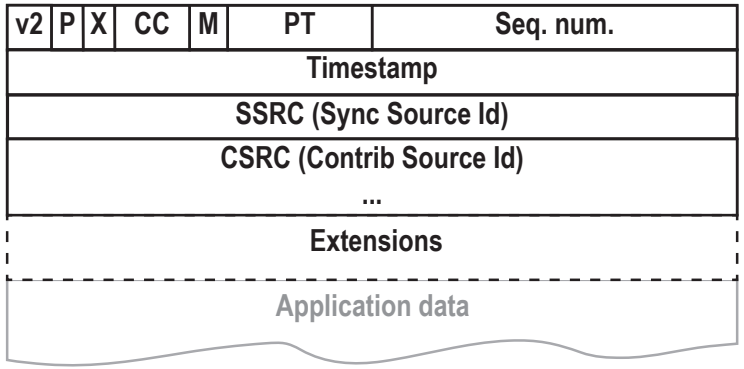
encoding, synchronizing, and reading of individual flows is left up to the application layer

Transport et RTP

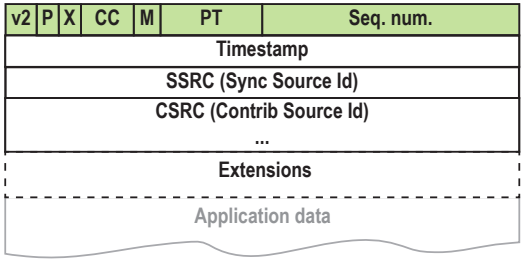
Witch transport layer for RTP/RTCP

- Is RTP a transport protocol ?
  - no mux/démux
- must be associated to a classical transport protocol
  - usually UDP
  - TCP, DCCP, SCTP...
- classical configuration with UDP :
  - even port number for RTP
  - next port number for RTCP
  - RTP and RTCP port number must be différent

RTP framing

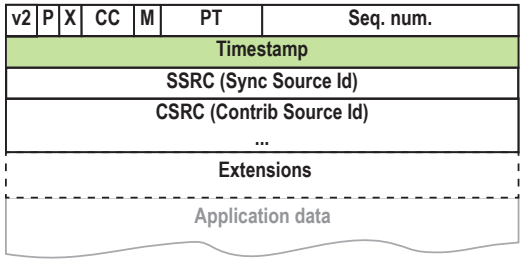


RTP Header



- P=1 indicates padding, X=1 indicates extension
- CC: number of CSRC IDs following the header
- M: profile-specific marker
- PT: packet type (profile-defined)
- Seq. num.: random initial value, increases by one for each packet

RTP Header: Timestamp



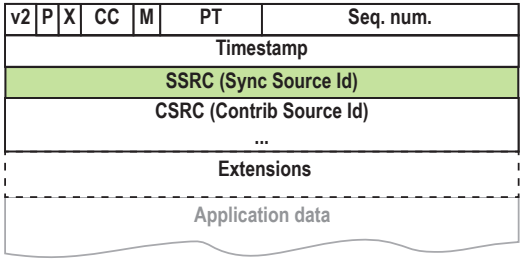
- timestamp of the first byte contained in the RTP packet
  - random starting value
  - clock frequency is application-dependent
  - to permit synchronization and jitter calculations, clock must increment monotonically and linearly in time
- translation to universal "wall clock" time
  - equivalence defined in RTCP

RTP Header : Timestamp

The timestamp allows playback at the right pace of multimedia data:

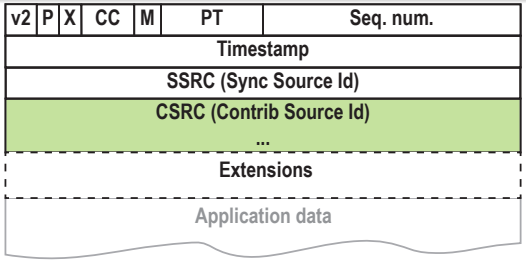
- **sufficient resolution** to allow calculation of the jitter
  - (unit = a video frame) is not sufficient
  - if the messages are generated regularly (eg 160 audio samples), then the clock can be the sampling clock (and incremented by 160 for each message)
- **identical clock** for all messages generated at the same time
  - ex:all messages associated with the same video frame will have the same timestamp (but not the same sequence number)
- according to the coding, the growth of the stamp can be non-monotonous
  - ex: messages carrying interpolated MPEG frames (IPBBBPBBB...)

RTP Header: SSRC



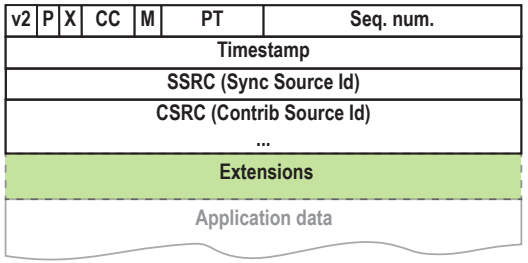
- data source identifier ("synchronization source")
- one sequence number space per SSRC
- value generated at random to ensure uniqueness (RTP implementation must be capable of handling ID collisions)
- an application sending multiple data streams will have multiple SSRCs

RTP Header: CSRC



- a "mixer" can stream several data streams in a single RTP session; each stream being identified by a CSRC ("contributing source" ID)
- there can be zero CSRCs, up to as many as fifteen
- example of mixing:
  - the mixer of a videoconference provides its own SSRC and identifies each participant's stream by a separate CSRC

RTP Header : Extensions



- allows proprietary implementation
  - in an experimental setting for example (to test new data formats)
  - pextension parameters defined by the implementation

Basic Audio and Video RTP Profiles

RFC 3551 defines some basic profiles with predefined parameters

PT	encoding	A/V	Clock (Hz)
0	PCMU	A	8000
2	G721	A	8000
3	GSM	A	8000
5	DVI4	A	8000
7	LPC	A	8000
8	PCMA	A	8000
9	G722	A	8000
15	G728	A	8000
26	JPEG	V	90000
31	H261	V	90000...

RTP Encoding Rules

- controlled sending at the **source**
  - choice of sending with different PT
  - segmentation of data at the encoder level
  - use of the M bit if necessary
- **multiplexing** of several types of data
  - forbidden in a RTP session
  - use multiple sessions in parallele
- very different audio and video mechanisms...

RTP Audio Encoding Rules (1)

Basic rules for audio encoding with RTP:

- silence suppression (thanks to timestamps)
  - but possible addition of a silence descriptor (Confort Noise)
  - positioning the marker to the first audio message after a silence (to distinguish silence from a network problem)
- RTP clock independent of the number of channels used and encoding
  - if N channels then N samples during a period
- initial sampling frequencies
  - 8000, 11025, 16000, 22050, 24000, 32000, 44100 et 48000 Hz

RTP Audio Encoding Rules (2)

Multi-channel sessions: Samples at the same time must be in the same RTP message following this order

number	channel						
	1	2	3	4	5	6	
stereo	l	r					
3	l	r	c				
4	l	c	r	S			
5	Fl	Fr	Fc	Sl	Sr		
6	l	lc	c	r	rc	S	

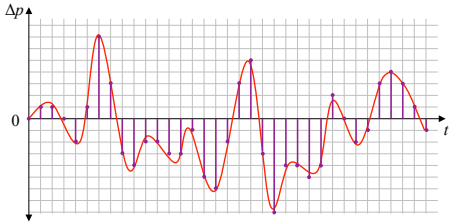
l	left
r	right
c	center
S	surround
F	front

RTP Audio Operating Recommandation

Size of transported audio data:

- **20 ms** or **an encoded frame** par message
  - the time interval in a message ➠ end-to-end delay
  - longer interval ➠ increased delay and loss rate
  - acceptable values: 0 to 200 ms
- encodings based on **samples**
  - each sample coded on S fixed bits
  - the sample bits are written as a result
    - if multichannel, samples of the same time are written in sequence in the defined order
  - duration of a message depends on the number N of samples
  - the data (S*N bits) must be a multiple of the byte
- encodings based on **frames**
  - encodes a predefined fixed size audio interval
  - resulting coding of variable or fixed size
    - several frames can be embedded in a message
  - multichannel: channels written in sequence in the defined order

Sampled Audio Example : L16 profile



16-bit PCM coding

- data = sequence of 16-bit words
  - signed (two's complement)
  - transmitted in network byte order (MSB first)
- additional information transmitted out-of-band (SDP ...)
  - sample rate
  - number of channels (and order if needed)
  - upstream analog processing (amplification etc.)

Framed Audio Example : GSM profile

Blocks of 160 samples (20 ms) coded in 33 bytes

Octet	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
0	1	1	0	1	LARc0.0	LARc0.1	LARc0.2	LARc0.3
1	LARc0.4	LARc0.5	LARc1.0	LARc1.1	LARc1.2	LARc1.3	LARc1.4	LARc1.5
2	LARc2.0	LARc2.1	LARc2.2	LARc2.3	LARc2.4	LARc3.0	LARc3.1	LARc3.2
3	LARc3.3	LARc3.4	LARc4.0	LARc4.1	LARc4.2	LARc4.3	LARc5.0	LARc5.1
4	LARc5.2	LARc5.3	LARc6.0	LARc6.1	LARc6.2	LARc7.0	LARc7.1	LARc7.2
5	Nc0.0	Nc0.1	Nc0.2	Nc0.3	Nc0.4	Nc0.5	Nc0.6	bc0.0
6	bc0.1	Mc0.0	Mc0.1	xmaxc00	xmaxc01	xmaxc02	xmaxc03	xmaxc04
7	xmaxc05	xmc0.0	xmc0.1	xmc0.2	xmc1.0	xmc1.1	xmc1.2	xmc2.0
8	xmc2.1	xmc2.2	xmc3.0	xmc3.1	xmc3.2	xmc4.0	xmc4.1	xmc4.2
9	xmc5.0	xmc5.1	xmc5.2	xmc6.0	xmc6.1	xmc6.2	xmc7.0	xmc7.1
10	xmc7.2	xmc8.0	xmc8.1	xmc8.2	xmc9.0	xmc9.1	xmc9.2	xmc10.0
11	xmc10.1	xmc10.2	xmc11.0	xmc11.1	xmc11.2	xmc12.0	xmc12.1	xcm12.2
12	Nc1.0	Nc1.1	Nc1.2	Nc1.3	Nc1.4	Nc1.5	Nc1.6	bc1.0
13	bc1.1	Mc1.0	Mc1.1	xmaxc10	xmaxc11	xmaxc12	xmaxc13	xmaxc14
14	xmax15	xmc13.0	xmc13.1	xmc13.2	xmc14.0	xmc14.1	xmc14.2	xmc15.0
15	xmc15.1	xmc15.2	xmc16.0	xmc16.1	xmc16.2	xmc17.0	xmc17.1	xmc17.2
16	xmc18.0	xmc18.1	xmc18.2	xmc19.0	xmc19.1	xmc19.2	xmc20.0	xmc20.1
17	xmc20.2	xmc21.0	xmc21.1	xmc21.2	xmc22.0	xmc22.1	xmc22.2	xmc23.0
18	xmc23.1	xmc23.2	xmc24.0	xmc24.1	xmc24.2	xmc25.0	xmc25.1	xmc25.2
19	Nc2.0	Nc2.1	Nc2.2	Nc2.3	Nc2.4	Nc2.5	Nc2.6	bc2.0
20	bc2.1	Mc2.0	Mc2.1	xmaxc20	xmaxc21	xmaxc22	xmaxc23	xmaxc24
21	xmaxc25	xmc26.0	xmc26.1	xmc26.2	xmc27.0	xmc27.1	xmc27.2	xmc28.0
22	xmc28.1	xmc28.2	xmc29.0	xmc29.1	xmc29.2	xmc30.0	xmc30.1	xmc30.2
23	xmc31.0	xmc31.1	xmc31.2	xmc32.0	xmc32.1	xmc32.2	xmc33.0	xmc33.1
24	xmc33.2	xmc34.0	xmc34.1	xmc34.2	xmc35.0	xmc35.1	xmc35.2	xmc36.0
25	Xmc36.1	xmc36.2	xmc37.0	xmc37.1	xmc37.2	xmc38.0	xmc38.1	xmc38.2
28	xmaxc35	xmc39.0	xmc39.1	xmc39.2	xmc40.0	xmc40.1	xmc40.2	xmc41.0
29	xmc41.1	xmc41.2	xmc42.0	xmc42.1	xmc42.2	xmc43.0	xmc43.1	xmc43.2
30	xmc44.0	xmc44.1	xmc44.2	xmc45.0	xmc45.1	xmc45.2	xmc46.0	xmc46.1

Other RTP Payload Audio Format

RFC 2658 RTP PureVoice(tm) Audio
RFC 3389 RTP Comfort Noise (CN)
RFC 3558 RTP for Enhanced Variable Rate Codecs (EVRC)
RFC 4184 RTP for AC-3 Audio
RFC 4348 RTP for VMR-WB Audio Codec
RFC 4352 RTP for AMR-WB+ Audio Codec
RFC 4867 RTP for AMR (Adaptive Multi-Rate) and AMR-WB Audio Codecs
RFC 5188 RTP for the Enhanced Variable Rate Wideband Codec (EVRC-WB)
RFC 5215 RTP for Vorbis Encoded Audio
RFC 5219 A More Loss-Tolerant RTP Payload Format for MP3 Audio
RFC 5391 RTP for ITU-T Recommendation G.711.1
RFC 5404 RTP for G.719
RFC 5574 RTP for the Speex Codec
RFC 5577 RTP for ITU-T Recommendation G.722.1 (wideband)
RFC 5584 RTP for the Adaptive TRansform Acoustic Coding (ATRAC) Family
RFC 5993 RTP for GSM Communications Half Rate (GSM-HR)
RFC 6884 RTP for the Enh. Var. Rate Narrow-Wideband Codec (EVRC-NW)
RFC 7587 RTP for the Opus Speech and Audio Codec

RTP Video Encoding Rules

Basic rules for video encoding with RTP:

- most video profiles use a **90 kHz** clock
  - identical to MPEG clock
  - integer multiple of 24 Hz (cinema), 25 Hz (PAL/SECAM), 29.97 Hz (NTSC) et 30 Hz (HDTV)
  - high enough to compute jitter
- timestamp = **image snapshot time**
  - if image on multiple messages: same timestamp
- the marking bit is used on the last message of an image

Video Example: H.264/MPEG4 AVC

NAL : Network Abstraction Layer

- packetization controlled at the encoder
  - splitting images into standalone slices (relative to image encoding)
  - then partition in NAL units (NALU, relative to the transport encoding)
- standalone NALU
  - NALU header (1 byte)
  - NALU data (raw byte stream: image or H264 parameters)
- intégration in RTP
  - usually one NALU ~ 1.4 Ko

Other RTP video profiles

RFC 2435 RTP for JPEG-compressed Video
RFC 2250 RTP for MPEG1/MPEG2
RFC 3640 RTP for Transport of MPEG-4 Elementary Streams
RFC 4425 RTP for Video Codec 1 (VC-1)
RFC 4587 RTP for H.261 Video Streams
RFC 4628 RTP for H.263 Video Streams
RFC 4629 RTP for the 1998 Version of ITU-T Rec. H.263 Video
RFC 5371 RTP for JPEG 2000 Video Streams
RFC 6184 RTP for H.264 Video
RFC 6190 RTP for Scalable Video Coding (extension H264 AVC)
RFC 6416 RTP for MPEG-4 Audio/Visual Streams
RFC 6469 RTP for DV (IEC 61834) Video
RFC 7741 RTP for VP8 Video
RFC 7798 RTP for High Efficiency Video Coding (HEVC)
RFC 8450 RTP for VC-2 HQ Profile
draft IETF RTP for VP9 Video
working draft RTP for AV1 Video

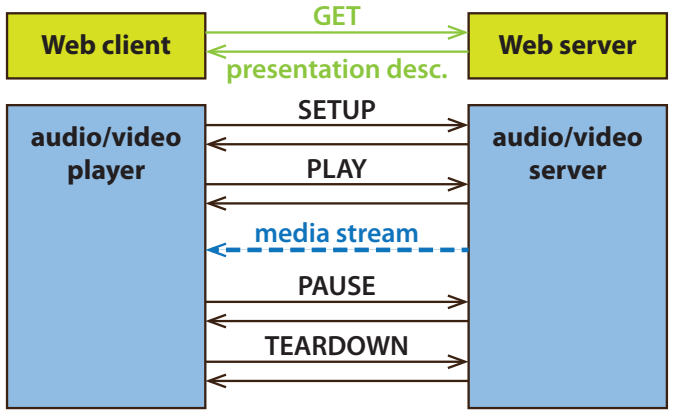
RTSP

RTSP

Real Time Streaming Protocol (RFC 2326)

- **out-of-band** streaming control
  (usually TCP port 554)
- identifies the resource via a URL
  (e.g., `rtsp://media.upmc.fr:554/videofile`)
- typical video player functions:
  - play/pause
  - fast-forward
  - skip to a specific timestamp...
- doesn't define any video or audio encodings
- doesn't define how data should be encapsulated
- doesn't specify a data buffering method

RTSP : Operation



RTSP: Methods

- main requests (mandatory)
  - **OPTIONS**: indicates the available commands
  - **SETUP**: start of an RTSP session and server-side resource allocation
  - **PLAY**: start the transmission of a stream
  - **TEARDOWN**: permanent shutdown with release of resources
- additional requests (optional)
  - **DESCRIBE**: technical description of a media (recom.)
  - **PAUSE**: temporarily stop the transm. of a stream (recom.)
  - **ANNOUNCE**: change in the description of a media (SDP)
  - **RECORD**: request to record a stream
  - **REDIRECT**: client redirection to a new server
  - **GET_PARAMETER**: retrieve transmission parameters
  - **SET_PARAMETER**: control of encoding or device

SCIENCES SORBONNE UNIVERSITÉ

RTSP: Responses

- 1xx *(Informational)*: request received, continuing process
  - `100 Continue`
- 2xx *(Success)*: action received, understood, and accepted
  - `200 OK`
  - `201 Created`
- 3xx *(Redirection)*: further action must be taken to complete
  - `300 Multiple Choices`
  - `302 Moved Temporarily`
  - `305 Use Proxy`
- 4xx *(Client Error)*: bad syntax or cannot be fulfilled
  - `400 Bad Request`
  - `415 Unsupported Media Type`
  - `453 Not Enough Bandwidth`
- 5xx: *(Server Error)*: server failed valid request
  - `500 Internal Server Error`
  - `503 Service Unavailable`

SCIENCES SORBONNE UNIVERSITÉ

RTSP : Example

```
C->W: GET /concert.sdp HTTP/1.1
      Host: www.upmc.fr

W->C: HTTP/1.1 200 OK
      Content-Type: application/x-rtsl

      <session>
        <track src="rtsp://live.upmc.fr/concert/audio">
      </session>


C->M: DESCRIBE rtsp://live.upmc.fr/concert/audio RTSP/1.0
      CSeq: 1

M->C: RTSP/1.0 200 OK
      CSeq: 1
      Content-Type: application/sdp
      Content-Length: 44

      v=0
      o=- 2890844526 2890842807 IN IP4 132.227.24.202
      s=RTSP Session
      m=audio 3456 RTP/AVP 0
      a=control:rtsp://live.upmc.fr/concert/audio
      c=IN IP4 224.2.0.1/16
```

```
C->M: SETUP rtsp://live.upmc.fr/concert/audio RTSP/1.0
      CSeq: 2
      Transport: RTP/AVP;multicast

M->C: RTSP/1.0 200 OK
      CSeq: 2
      Transport: RTP/AVP;multicast;destination=224.2.0.1;
          port=3456-3457;ttl=16
      Session: 0456804596

C->M: PLAY rtsp://live.upmc.fr/concert/audio RTSP/1.0
      CSeq: 3
      Session: 0456804596

M->C: RTSP/1.0 200 OK
      CSeq: 3
      Session: 0456804596
```

SCIENCES SORBONNE UNIVERSITÉ

SIP

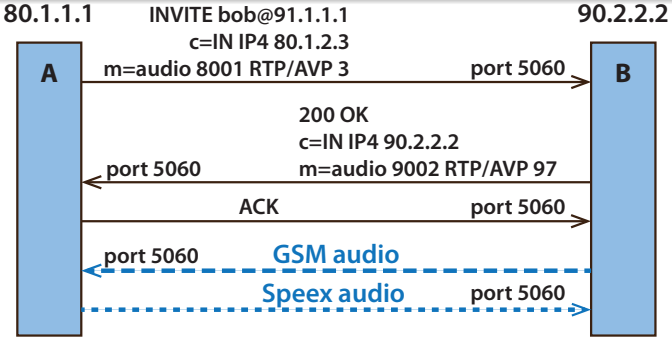
SCIENCES SORBONNE UNIVERSITÉ

SIP

Session Initiation Protocol (RFC 3261)

- call lifecycle
  - **notification** (call establishment)
  - **negotiation** (encodings, media...)
  - **termination** (ending a call)
- users identified by names and/or e-mail addresses (not by telephone numbers)
- handles correspondence between IDs and IP addresses
- call management
  - **adding a new media** during the call
  - **changing encodings** during the call
  - **inviting** participants
- the basis for IMS (IP Multimedia Subsystem of the 3GPP)

SIP: Operation



- **negotiation** of the codec (reply `606 not acceptable`, giving the list of supported codecs)
- call **rejection** (replies `busy`, `gone`, `payment`,`forbidden`...)
- **transmission** (multimedia data sent over RTP or another protocol)

SIP : Example

```
INVITE sip:auto@localhost SIP/2.0
Via: SIP/2.0/UDP 127.0.0.1:13764;rport
Max-Forwards: 70
Contact: <sip:matthew@127.0.0.1:13764>
To: "sip:auto@localhost"<sip:auto@localhost>
From: "Olivier"<sip:olivier@upmc.fr>;tag=5c7cdb68
Call-ID: NmNhYWNhMjY0Y2M0OT.
CSeq: 1 INVITE
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY,
   MESSAGE, SUBSCRIBE, INFO
Content-Type: application/sdp
User-Agent: eyeBeam release 1014g stamp 44944
Content-Length: 389

v=0
o=- 7 2 IN IP4 132.227.61.199
...
```

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP 127.0.0.1:13764;rport=13764;received=12
To: "sip:auto@localhost" <sip:auto@localhost>
From: "Olivier"<sip:olivier@upmc.fr>;tag=5c7cdb68
Call-ID: NmNhYWNhMjY0Y2M0OTc4YTI2MzgzZTN1YTRhZTMxNTE.
CSeq: 1 INVITE
Content-Length: 0
```

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 127.0.0.1:13764;;rport=13764;received=127.0.0.1
To: "sip:auto@localhost" <sip:auto@localhost>;tag=tCAED3F5A
From: "Olivier"<sip:olivier@upmc.fr>;tag=5c7cdb68
Call-ID: NmNhYWNhMjY0Y2M0OTc4YTI2MzgzZTN1YTRhZTMxNTE.
CSeq: 1 INVITE
Contact: <sip:127.0.0.1:5060>
Content-Type: application/sdp
Content-Length: 271

v=0
o=olivier 1208261984604 1208261984604 IN IP4 127.0.0.1
...
```

```
ACK sip:127.0.0.1:5060 SIP/2.0
Via: SIP/2.0/UDP 127.0.0.1:13764;rport
Max-Forwards: 70
Contact: <sip:matthew@127.0.0.1:13764>
To: "sip:auto@localhost"<sip:auto@localhost>;tag=tCAED3F5A
From: "Olivier"<sip:olivier.@upmc.fr>;tag=5c7cdb68
Call-ID: NmNhYWNhMjY0Y2M0OTc4YTI2MzgzZTN1YTRhZTMxNTE.
CSeq: 1 ACK
Content-Length: 0
```


SIP: managing identities and locations

Matching the SIP identifier (name/e-mail address) to the callee's IP address

- the callee can be mobile
- he/she can have a dynamic or private IP address
- he/she can have several devices (computer, tablet, smartphone, etc.)

Parameters to take into account

- the time and the location
- the state of the call and/or the callee (call transfer, etc.)

➠ SIP servers manage this service:

- the **SIP Registrar**
- the **SIP Proxy**

SIP: Registrar

Upon launch of the user's SIP client:

➠ a `REGISTER` is sent to the user's **SIP Registrar server**.

```
REGISTER sip:upmc.fr SIP/2.0
Via: SIP/2.0/UDP 132.227.61.205:5061;rport;branch=z9hG4bKDC8595CD770E4317ACBC3B64CF5(
From: Olivier <sip:olivier@upmc.fr>;tag=1516659370
To: Olivier <sip:olivier@upmc.fr>
Contact: "Olivier" <sip:olivier@132.227.61.205:5061>
Call-ID: 46E1C3CB36304F84A020CF6DD3F96461@upmc.fr
CSeq: 37764 REGISTER
Expires: 1800
Max-Forwards: 70
User-Agent: LIP6-SIP-Phone v0.9
Content-Length: 0
```


SIP: Proxy

The user sends an`INVITE` message to his/her **SIP proxy**

- indicates the SIP address of the receiver (`sip:olivier@upmc.fr`)

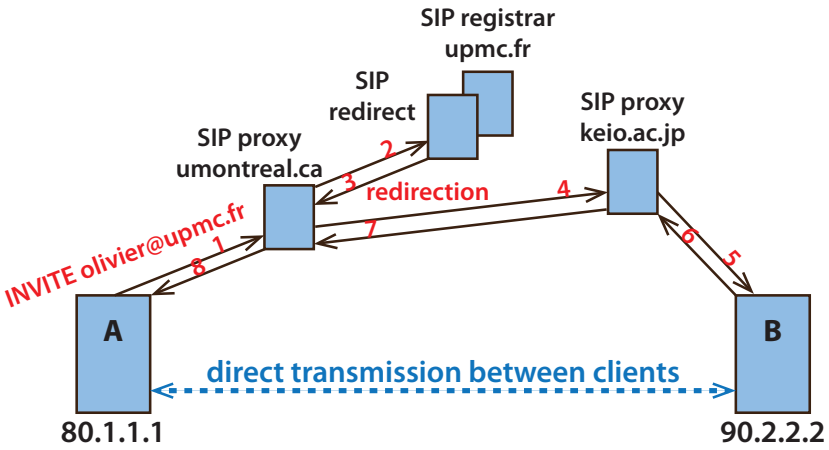
The SIP Proxy is responsible for forwarding SIP messages

- perhaps across several proxies
- the callee replies acruss the same proxies

➠ the callee's proxy sends a SIP reply containing its IP address

A SIP proxy functions much like a local DNS server

SIP: example



SDP

Session Description Protocol (RFC 4566)

- multimedia **session description** setup
- present the description of the media to be transmitted to the participants
  - addresses, ids, codecs, metadata, etc.
- not a transmission/transport protocol
  - only description
- can be used on any transport protocol
- can be integrated into any initialization/signaling protocols
  - RTSP, SIP, etc.
- **standard** presentation format
  - SDP description content indication with `Content-Type: application/sdp`
  - SDP description is a sequence of line of text `<type>=<value>`

SDP : example

```
v=0
o=fourmaux 2990844526 2990842807
    IN IP4 132.227.63.51
s=SDP Seminar
i=A Seminar on SDP
u=http://www.upmc.fr/sem/sdp.pdf
e=olivier.fourmaux@upmc.fr
c=IN IP4 224.2.17.12/127
t=2973397496 2973404696
a=recvonly
m=audio 49170 RTP/AVP 0
m=video 51372 RTP/AVP 99
a=rtpmap:99 h263-1998/90000
```

SDP Possible types:

```
v=  (protocol version)
o=  (originator and session identifier)
s=  (session name)
i=* (session information)
u=* (URI of description)
e=* (email address)
p=* (phone number)
c=* (connection information)
b=* (0 or more bandwidth information lines)
t=  (time the session is active)
r=* (0 or more repeat times)
z=* (time zone adjustments)
k=* (encryption key)
a=* (0 or more session attribute lines)
   0 or more media descriptions
     m=  (media name and transport address)
     i=* (media title)
     c=* (connection information)
     b=* (0 or more b.w. information lines)
     k=* (encryption key)
     a=* (0 or more media attribute lines)
```

SCIENCES SORBONNE UNIVERSITÉ

Vidéo

SCIENCES SORBONNE UNIVERSITÉ

Internet video evolution

avant 1995 : Academic Internet

- start of video on Internet research

1995 - 2005 : Web Internet

- 1st generation of commercial applications (progressive download via a dedicated reader)

2005 - 2010 : Beginning of the multimedia Internet

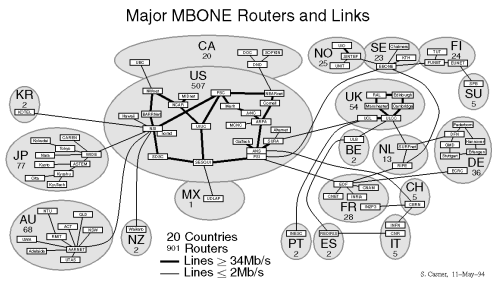
- 2nd generation of commercial applications with RTSP, and especially RTMP (de facto standard **flash**)

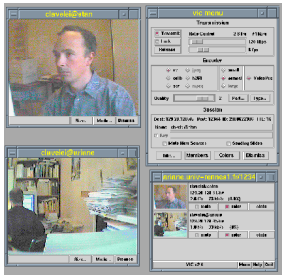
2010... : Internet massively for video

- 3rd generation of commercial applications with **HTTP Streaming**...

SCIENCES SORBONNE UNIVERSITÉ

Internet Video Research - 1990s



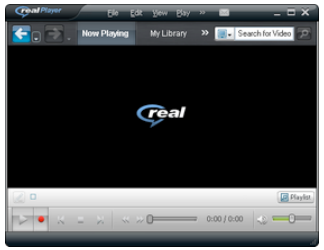


- Quality of Service support inside network
  - signaling, ressources reservation protocol and packet scheduling algorithms (RSVP/IntServ)
  - traffic prioritization (DiffServ)
- multicast communications

➠ target application: **multi-party vidéoconference**

SCIENCES SORBONNE UNIVERSITÉ

1st generation of commercial video app. over Internet (1)



- simple video playback, no support for rich app
- not well integrated with Web browser
- proprietary protocols, servers, and streamers
- no critical mass of compelling content over Internet
- not enough broadband penetration

➠ time of the Web Internet (1995-2005)

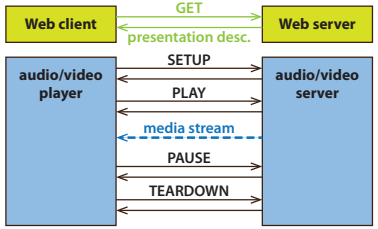
1st generation of commercial video app. over Internet (2)

- complete download
  - objects retrieved via HTTP then read via dedicated player
    - ✘ no *pipelining*
- **progressive** download
  - **metafile** retrieved and passed to the dedicated player
  - the player et up a direct communication to the web server
    - ✘ HTTP transfers data as quickly as possible
    - ✘ important transmission even if playback is stopped
    - ✘ unsuitability for audio/video use (stop, play, direct access)
    - ✘ TCP overhead
    - ✘ no adaptation to network conditions

2nd generation of commercial video app. over Internet (1)

Dedicated streaming protocols

- statefull protocolss
- separated control and data connections
  - control : play, pause, back, fast forward...
  - data : UDP or TCP (no HTTP)
- example : RTSP, RTMP...



2nd generation of commercial video app. over Internet (2)

Flash : main technology for video decoding

- player de facto on the client side (95% PC, all OSs, all browsers)
- programming environnement multi-plateforme (Action script)
- rich multimedia content (with decent browser integration)
- video FLV : codec Sorenson, VP6 puis H264
- RTMP protocol
  - ✘ security issues, non standard port filtering
  - ✘ web servers more profitable than dedicated servers

Now 2010

New screens : smart TV, smartphones, tablets, game consoles...

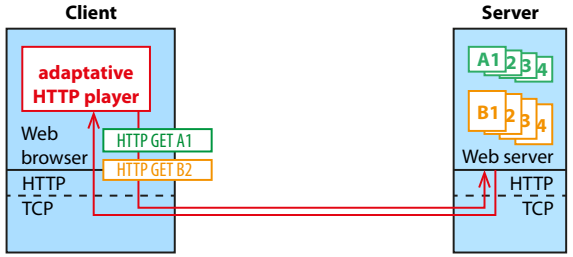
- new requirements (multiple encodings: + de 4 screen types)
- new plateformes where Flash is not available (iOS, Android, game consoles...)

3rd generation: HTTP Streaming

Adaptation of the content delivery to the Internet (rather than the opposite)

- **client oriented** architecture (stateful client, stateless server)
  - classical server: web servers
  - classical protocol: HTTP
  - state of the session and flow regulation on the client side
- video **split** in multiples **chunks**
  - a **chunk** start on a reference image (I)
  - progressive download of each **chunk** via HTTP
  - sequential play of **chunk** ➠ continuous video play
- adaptation to different stream rates
  ➠ **adaptative HTTP streaming**

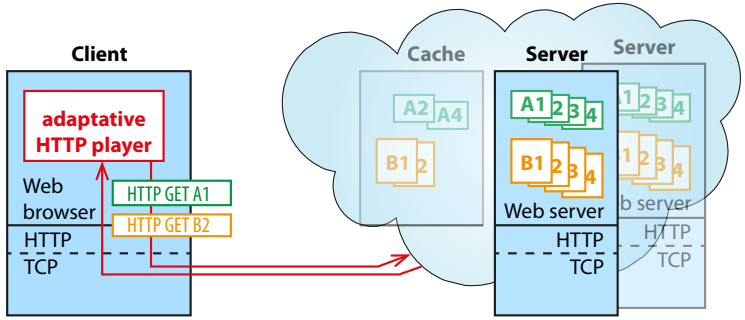
HTTP Chunking Protocol



Video stream segmentation mechanism for HTTP transport

- each segment is encoded at different levels of quality (**chunks**)
- adaptation by the client: request of the **chunk** with the appropriate encoding
  ➠ all this chunks encode the same video segments

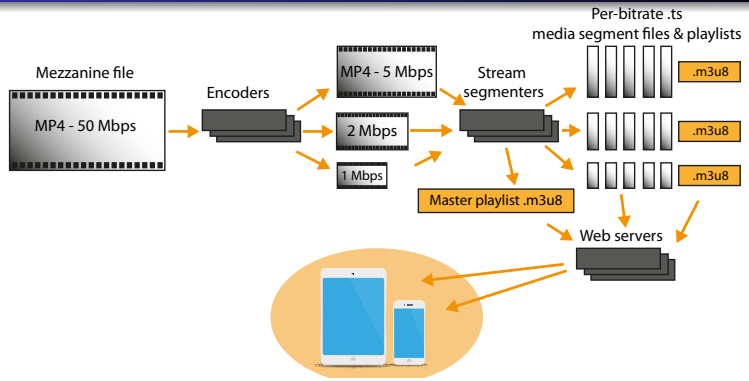
Adaptative HTTP streaming advantages



Reasons for wide adoption

- client driven control enable server/CDN switch
- reuse the CDN infrastructure
- middlebox/firewall penetration

Apple HTTP Live Streaming (2009)



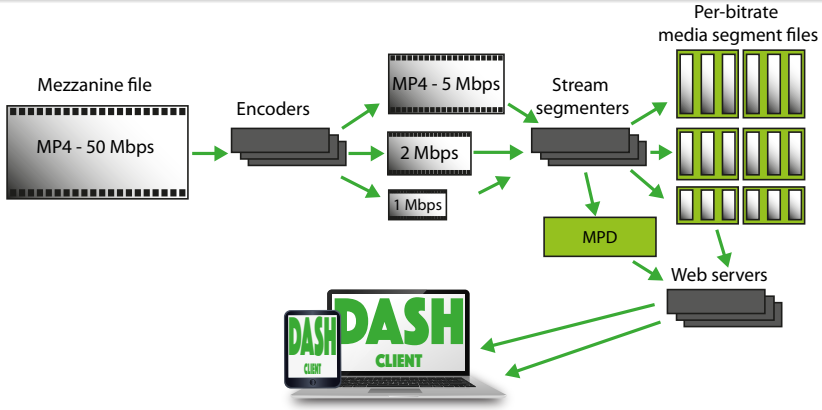
- content segmentation in small standard **HTTP object** `.ts` (10s)
- **broadcast servers**: classical HTTP server
- **clients**: initial download of playing list (**Extended M3U**) + HTTP GET on segments + re-assemble the flow
- **open format** (draft IETF informational RFC)

MPEG DASH ISO/IEC 23009-1

MPEG's Dynamic Adaptative Streaming over HTTP (2012)

- motivation
  - HTTP est mandatory (caches/CDN, NAT/Firewalls, network condition adapation...)
  - multiple kind of devices
  - multiple HTTP streaming solutions
- principes
  - sending of small video pieces via HTTP
  - no delivery infrastructures modification
  - client control and reassembly
  - network condition and client adaptation
- target : OTT on all devices

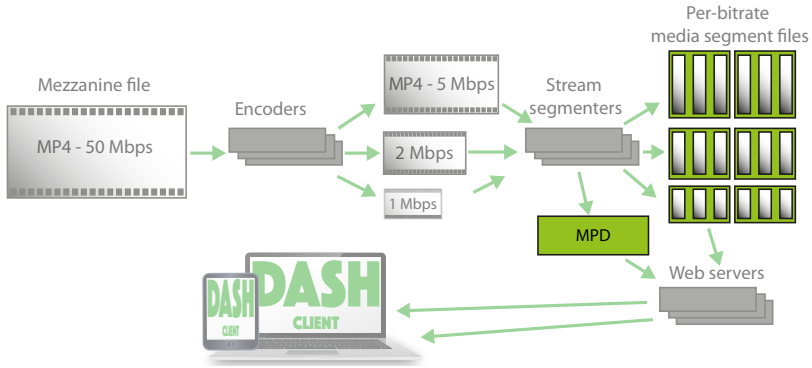
DASH: architecture



Basic mechanisms:

- "manifest" MPD file retrieval
- HTTP GET for all needed video segments

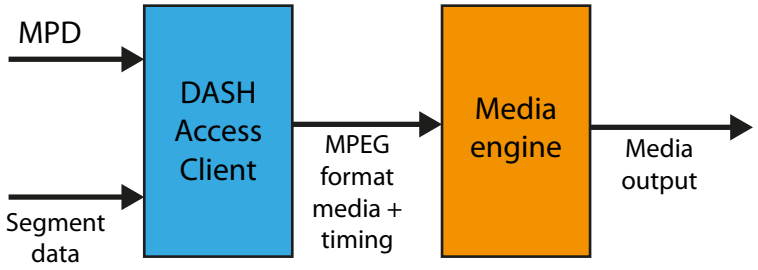
DASH: what is specified



- presenting data on the HTTP server
- MPD file (Media Presentation Description)
- structure des URL pour retrouver les segments via HTTP

DASH: information classification

- MPD and indexed information for the DASH client
  - heart of DASH specifications
- initialization and content flow for the reader
  - reuse of existing multimedia containers



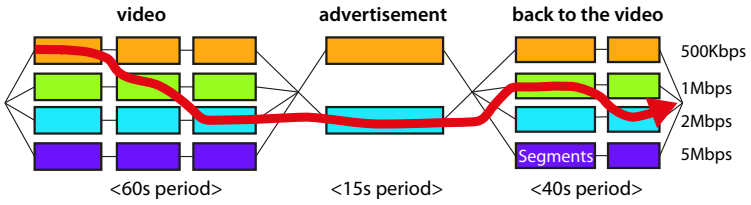
DASH: MPD

Media Presentation Description

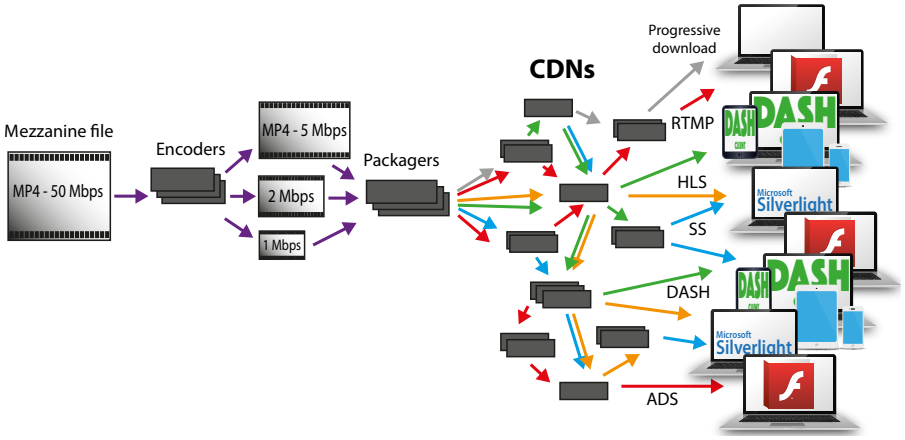
- information to select the elements of the different representations:
  - bitrate, resolution, codec, language, DRM...
- time and access informations:
  - URL and position in each accessible segment (byte-range)
  - availability of segments (start/end in absolute time)
  - initial presentation time and relative duration of segments
  - recommended reading time (live)
- information of cutting and switching between representations

DASH: client-side mechanisms

- client process:
  - download of **MPD file**
  - download **segment by segment** according to the reading
  - **determination of available throughput**
- choice factor for representation:
  - **memories** buffers status
  - **network** status
  - **device** status
  - change of user resolution



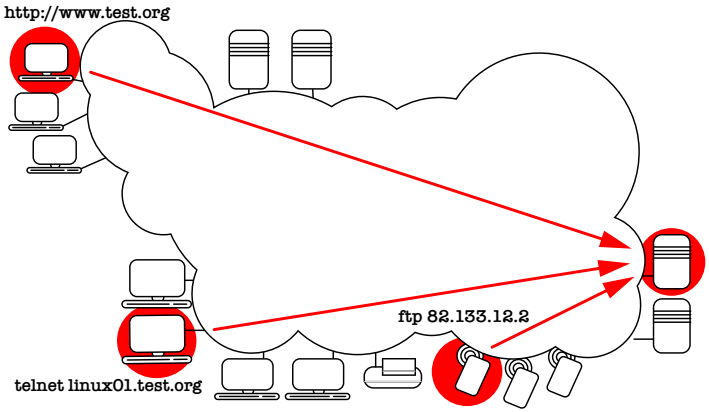
Current Video Delivery



ComNet: course 2/5 outline

1. Legacy applications
   - Introduction
   - Remote login
   - File transfer
2. Contemporary applications
   - Electronic mail
   - World Wide Web
   - multimedia
3. Support applications
   - Directory (DNS)
   - Network management

Associating names and addresses



Directory

Converting written Internet host names into numerical addresses

- initially
  - **one** file
  - flat "namespace"
  - administration centralized by a *NIC (Network Information Center)*
- today: **DNS**
  - **distributed** database
  - **hierarchical** namespace
    - decorrelated from the physical topology
  - system controlled by*InterNIC* (1992-1998), then ICANN *(Internet Corporation for Assigned Names and Numbers)* and its numerous subauthorities
    - hierarchical delegation (close to the namespace structure)
    - delegating reasonable-sized pieces of the namespace
  - protocol...

DNS (*Domain Name System*)

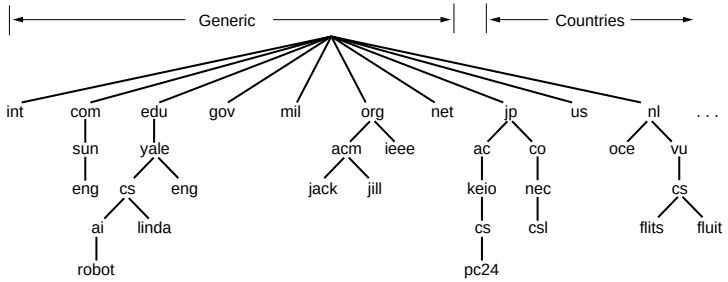
Standard directory for the Internet (RFC 1034 and RFC 1035)

- hierarchical namespace and system of delagation
- **name servers** (DNS servers)
  - physical components of the **hierarchy** that host the distributed database
  - handle DNS requests
  - transport via **UDP** or TCP, port **53**
  - applications access them through the **resolver** (UNIX):
    - `gethostbyname` (3), `gethostbyaddr` (3)
- services:
  - *name resolving*
  - *host aliasing*
  - *mail server aliasing*
  - *load disribution...*
- example:
  - BIND (*Berkeley Internet Name Domain*)
    - `named` (UNIX)

DNS: namespace

Hierarchical naming system

- tree structure (~ Unix filesystem)
- node label: 63 chars. max. (`A..Z0..9-` case insensitive)
- **domain name** = list of labels, traversing the tree from root to leaf (total 255 chars. max. and "." label separator):
  - absolute (**FQDN**): `pc24.CS.keio.ac.jp.`
  - relative names are handled locally (host)



DNS: ccTLD (*country code Top Level Domain*)

ccTLD (ISO 3166)	240 countries and external territories
`.ac`	Ascension Island
`.af`	Afghanistan
`.aq`	Antarctica (-60°S)
`.eu`	European Union
`.fr`	France
`.gf`	French Guiana
`.gp`	Guadeloupe
`.mq`	Martinique
`.pf`	French Polynesia + Clipperton
`.pm`	Saint-Pierre and Miquelon
`.re`	Réunion
`.tf`	TAAF
`.ru`	Russia (+`.su`)
`.tv`	Tuvalu
`.uk`	United Kingdom (+`.gb`)
`.us`	United States
`.zw`	Zimbabwe

DNS: gTLD (*generic Top Level Domain*)

`.aero`	2001	Air-transport industry *	SITA
`.asia`	2006	Asia-Pacific region *	Afilias
`.biz`	2001	Unrestricted	NeuLevel
`.cat`	2005	Catalan lingu. & cult.*	Asso. puntCAT
`.com/.net`	1985	Unrestricted	VeriSign
`.coop`	2001	Cooperative *	DotCooperation
`.edu`	1985	(US) educational inst. *	VeriSign
`.gov`	1985	US government *	US Admin.
`.info/.org`	01/85	Unrestricted	Afilias
`.int`	1988	Internat. organisations	ICANN
`.job`	2005	Human resrc. managment*	Employ Media
`.mil`	1985	US military *	US DoD NIC
`.mobi`	2005	Mobile device use *	Mobi JV
`.museum`	2001	Museums *	MuseDoma
`.name`	2001	Individuals	VeriSign
`.pro`	2001	Professionals	RegistryPro
`.tel`	2005	Internet Tel. serv.*	Telnic Limited
`.travel`	2005	Travel industry*	Tralliance Corp.

DNS: New TLD since 2012

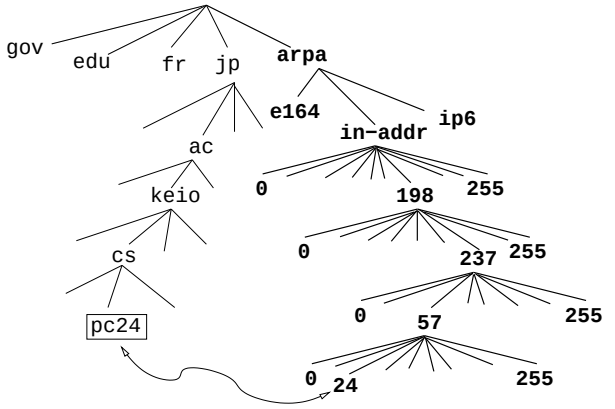
In 2012, ICANN, authorized ~2,000 new TLD

- large variety of TLD:
  - trademark (`.google`, `.danone`, etc.)
  - community based (`.archi`, `.immo`, etc.)
  - location (`.stockholm`, `.nyc`, etc.)
  - non ASCII charset recoded with Punycode/RFC3492 (`.`移动 encoded `.xn-6frz82g` : ideogram for "mobile", etc.)
  - general (`.music`, `.photo`, `.love`, etc.)

DNS: .arpa domain

Lookup: `pc24.cs.keio.ac.jp.` ➠ ?

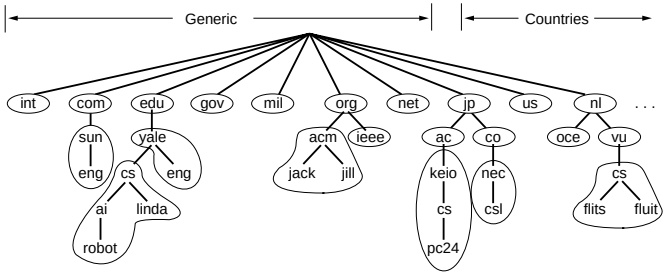
Reverse lookup: `24.57.237.198.in-addr.arpa.` ➠ ?



DNS: zones (1)

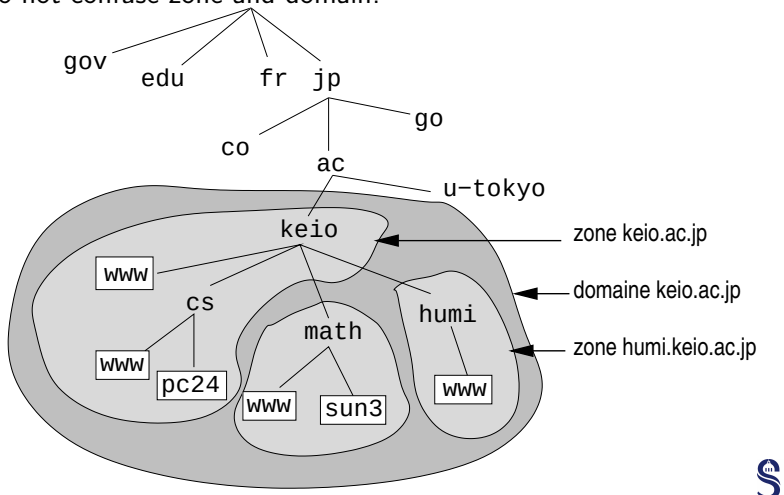
ICANN administers the root and delegates the TLDs to *domain name registries*

- separately administered zones (subtrees to the DNS tree)
  - (~ physical partitions in a Unix filesystem)
  - delegation of the names of the corresponding subdomains
  - example: `keio.ac.jp`.
- each with its own **name servers**



DNS: zones (2)

Do not confuse zone and domain!

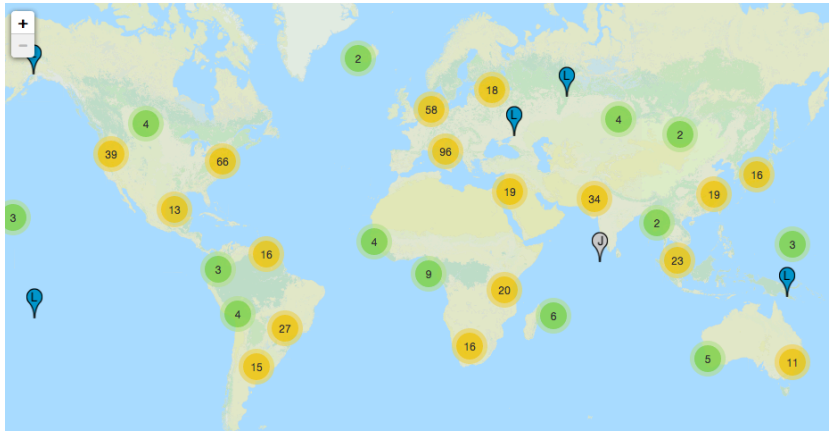


DNS: name servers

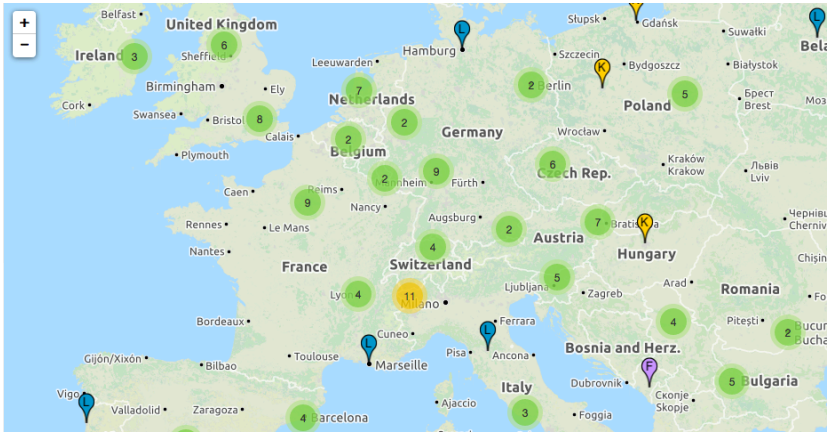
Different types of name server

- **authoritative servers** for a zone:
  - primary name server
    - authoritative records
    - aware of its descendants (delegations)
    - configured locally (from disk)
  - secondary name server
    - redundancy: completely separate from the primary
    - configured and updated by the primary (**zone transfer**)
  - physically independent from the zone
- **local servers** (providing access to the service)
  - top-down resolving (from TLD to sub-domain)
  - knowledge of root name servers
    - 1 primary and 12 secondaries, high availability (anycast)
    - hardwired (`ftp.rs.internic.net/domain/named.root`)
  - **recursive** or **iterative** queries

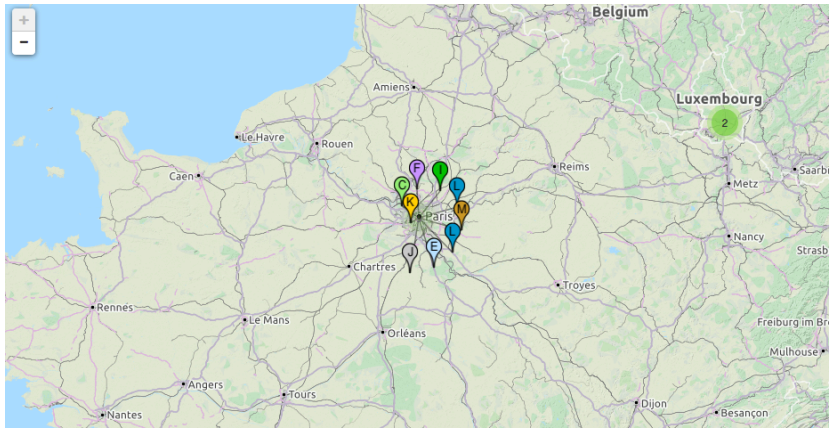
Root-servers (1)



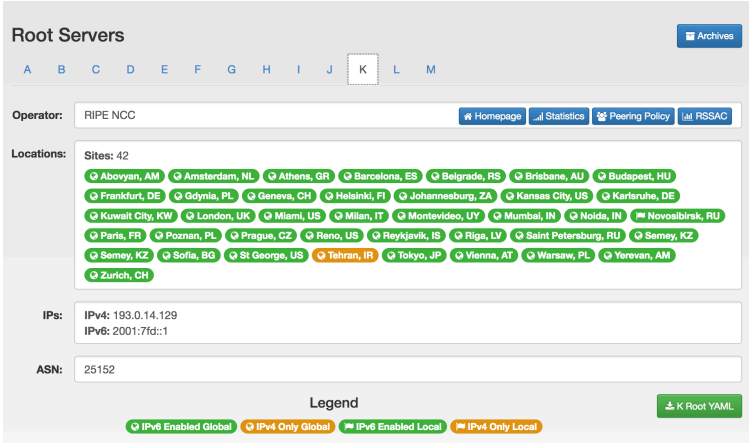
Root-servers (2)



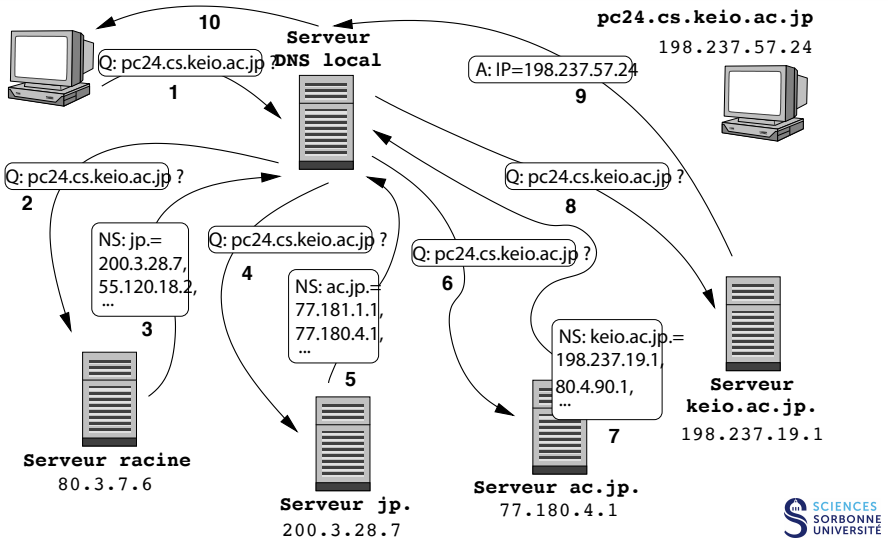
Root-servers (3)



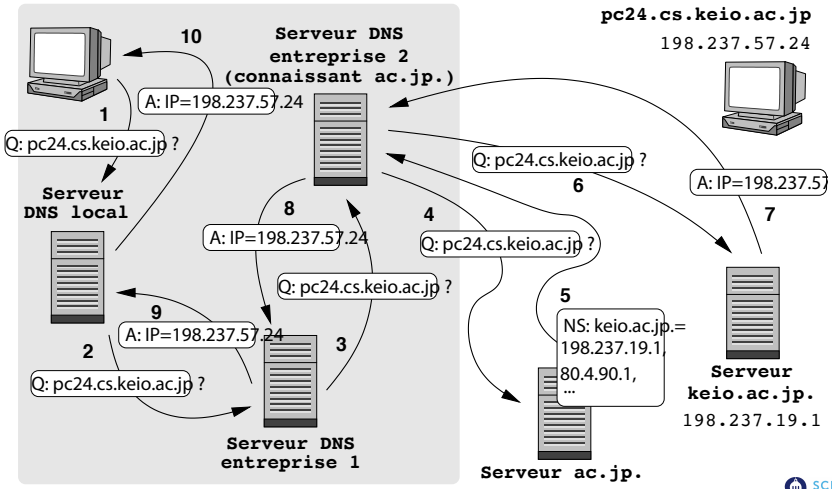
Root-server K



DNS: iterative query



DNS: recursive query

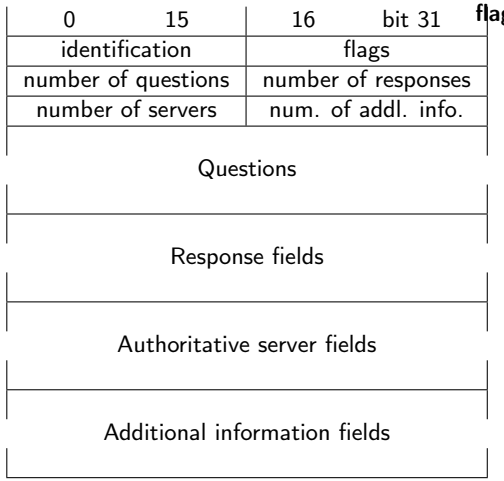


DNS: performance

Can the DNS system support the load?

- problems from systematic consultation of the root
  - does not take into accoun the locality of requests
    - local server generally distinct from the authoritative server
  - load on the root servers
    - how many requests from the Internet as a whole?
  - availability of the root servers
    - all requests pass through these
- the use of **caches**
  - second-hand information (*non-authoritative records*)
  - response from an authoritative server includes a time limit on its validity (TTL)
    - responses from the TLDs regarding the root servers are valid 48 hours
      - ➠ 100,000 requests per second (2005)

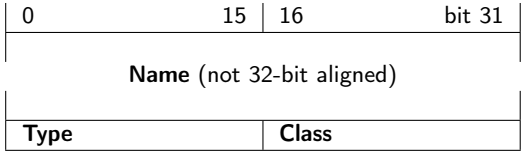
DNS: basic message structure



flags:

- **QR** (1 bit): 0 = question, 1 = response
- **opcode** (4 bits) 0 = standard ...
- **AA** (1 bit): 1 = authoritative response
- **TC** (1 bit): 1 = truncated (UDP datagram < 512B)
- **RD** (1 bit): 1 = recursion desired (indicated by the client)
- **RA** (1 bit): 1 = recursion available (indicated by the server)
- **reserved** (3 bits): 000
- **rcode** (4 bits): 0 = no error... 3 = name error...

DNS: structure of **one** query field



- **Name**: N bytes, each label is preceded by a byte indicating its length (or, if >0x3F, 0xC0ZZ is a pointer to ZZ bytes from the start of the message). Terminated by 0x00.

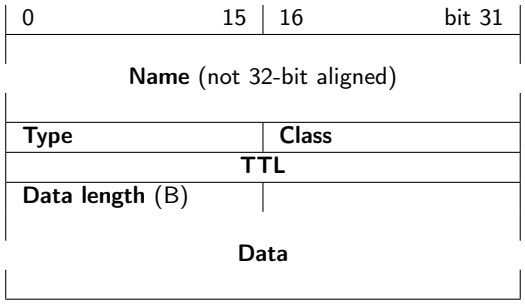
  4, 'p', 'c', '2', '4', 2, 'c', 's', 4, 'k', 'e', 'i', 'o', 2, 'a', 'c', 2, 'j', 'p', 0

- **Type** (16 bits):

val	name	description	val	name	description
1	**A**	IPv4 addr	13	**HINFO**	eqpt. info.
2	**NS**	serv. name	15	**MX**	server msg.
5	**CNAME**	alias	28	**AAAA**	IPv6 address
6	**SOA**	zone		...	
12	**PTR**	name ptr.	255	*	tt types (quest.)

- **Class** (16 bits): 1 = Internet

DNS: structure of **one** answer field



- **Name**, **Type**, **Class**: idem
- **TTL** (32 bits): validity in seconds
- **Data length** (16 bits): in bytes
- **Data** (N bytes without padding):
  - Name (string encoded as for a question) NS, CNAME...
  - Addresses (numerical value) A 4 bytes, AAAA 16 bytes...

DNS: inverse directory

Converting numerical addresses into literal names

- queries of type **name pointer** (PTR)
  - IPv4 address
    - `198.237.57.24`
  - conversion into the `in-addr.arpa` domain
    - `24.57.237.198.in-addr.arpa`

    ➠ often used to verify access rights

DNS: obtaining a delegation

Becoming the authoritative reference for an official subdomain:

- reserve the name with a *domain name registrar*
- put in place servers that conform to the DNS standard
  - authoritative zone information
    - replication in at least one secondary server
  - if there are sub-delegations:
    - knowledge of the server's descendants
  - if managing the corresponding IP addresses:
    - authoritative name pointers

DNS: dynamic updates

Dynamic DNS (RFC 2136)

- host autoconfig compatibility (local DNS):
  - *update*
  - *notification*
- security problems...

Dynamic DNS service (third-party provider)

- to operate with a dynamic address (residential access):
  - server: `dyndns.org`, `no-ip.org`...
  - client indicates the address update *(host/setupbox)*
  - virtual delegation (3rd level subdomain)
    - `toto123.myftp.biz`
    - `toto123.blogsite.org`
    - `toto123.homelinux.org`
    - `toto123.dyn-o-saur.com`
    - `toto123.endofinternet.net`...

DNS: security

No security in the basic protocol (RFC 3833)

- interception/modification of DNS messages
- false messages *(DNS cache poisoning)*
- denial of service...

DNSSEC (RFC 4033 à 4035 + RFC 4310 + RFC 4641)

- extension of the DNS system, enabling:
  - authentication of the origin of data
  - authentication of existence denial
  - data integrity
- obligatory for securing DNS updates
- limitations:
  - DNSSEC does not provide confidentiality of data
  - DNSSEC does not protect against DDoS Attacks

DNS: example

```
Unix> dig www.math.keio.ac.jp

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 11895
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 3, ADDITIONAL: 4

;; QUESTION SECTION:
;www.math.keio.ac.jp.           IN    A

;; ANSWER SECTION:
www.math.keio.ac.jp.    3600    IN    CNAME   sun3.math.keio.ac.jp.
sun3.math.keio.ac.jp.   3600    IN    A       131.113.70.3

;; AUTHORITY SECTION:
math.keio.ac.jp.        3600    IN    NS      relay.math.keio.ac.jp.
math.keio.ac.jp.        3600    IN    NS      ns.st.keio.ac.jp.
math.keio.ac.jp.        3600    IN    NS      ns0.sfc.keio.ac.jp.

;; ADDITIONAL SECTION:
relay.math.keio.ac.jp.  3600    IN    A       131.113.70.1
ns.st.keio.ac.jp.       127     IN    A       131.113.1.8
ns0.sfc.keio.ac.jp.     1199    IN    AAAA    3ffe:501:1085:8001::121
ns0.sfc.keio.ac.jp.     2358    IN    A       133.27.4.121

;; Query time: 577 msec MSG SIZE  rcvd: 206
```


ComNet: course 2/5 outline

1. Legacy applications
   - Introduction
   - Remote login
   - File transfer
2. Contemporary applications
   - Electronic mail
   - World Wide Web
   - multimedia
3. Support applications
   - Directory (DNS)
   - Network management

Network management

Network evolution (many machines and devices to administer)

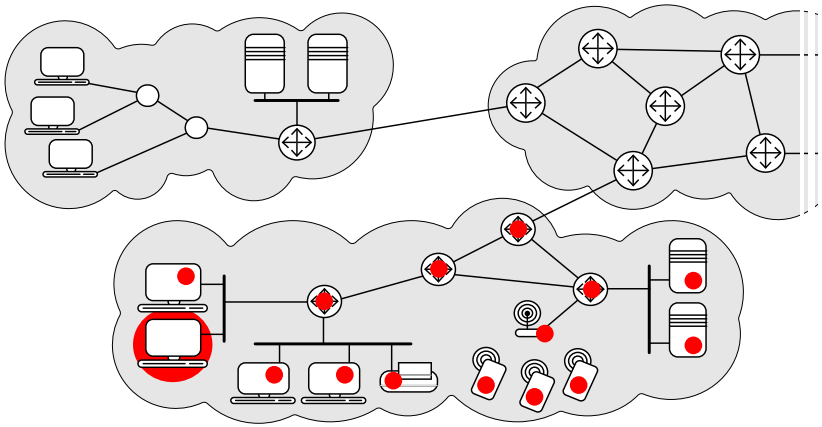
Needs:

- network supervision
  - fault detection
  - performance measurement
- equipment control
  - activating (interfaces...)
  - configuring (routing tables...)
- centralized control

Constraints:

- heterogeneous devices
  - routers, hubs, switches...
  - computers, printers, probes...
- multiple constructors
- remotely deployed devices

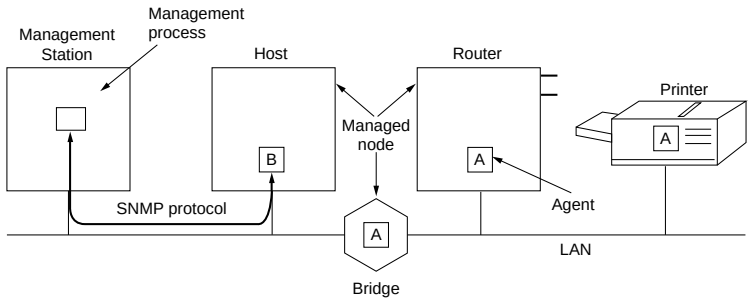
Manageable equipement



TCP/IP management

Managing machines in a TCP/IP environment

- instrumenting the devices (**agents**)
- supervision software (HP Openview, Cisco Works, Nagios...)
- management protocol ➠ SNMP



pictures from Tanenbaum A. S. *Computer Networks 3rd edition*

SNMP: design

Information is stored in two types of databases:

- **agent bases** (in the devices): The values are directly linked to the internal records
- **centralized base** (supervision platform): most recent values received and history (statistics)

Standardisation (for heterogeneous environments)

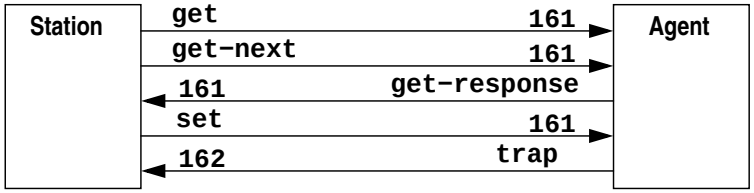
- **MIB** standards designate the types of information stored
- **SMI** standards describe common structures and nomenclature
- data format is **ASN.1**
- the **SNMP** protocol for controlling agents enables:
  - **reading/writing** of variables at governed entities
  - communication of **alarms**
  - **listing** the variables of the governed entities

➠ **combined global view**

SNMP: commands

The richness the MIBs!

- only 5 **simple** commands
- uses **UDP** ports **161** and **162**



SNMP: message structure

version	communauté	type PDU	ident req.	erreur status	erreur index	nom	valeur	nom	valeur	...

- `version`: current version 1 (0 ~ SNMPv1)
- `community`: string defining access rights
  - generally "`public`"
- `PDU type`: 0 (`get`), 1 (`get-next`), 2 (`get-response`), 3 (`set`)
  - we will explain type 4 (`trap`) messages later...
- `ident. req.`: link queries and responses
- `error status` and `error index`: error type relating to the variable being referenced (0 ~ no error)
- `name` and `value`: variables being described

Field sizes are not pre-set, as the structure is described in ASN.1 with BER coding.

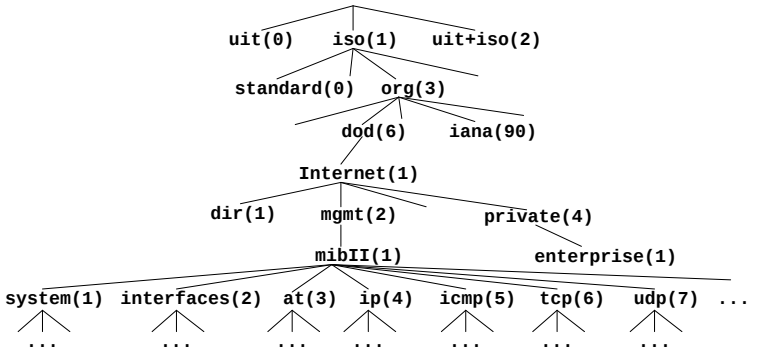
SNMP: SMI (*Structure for Management Information*)

- types defined by SMIv1 (RFC 1155 et 1212)

`NULL`	no value
`INTEGER`	signed integer (no limit)
`Counter`	positive integer (0 to $2^{32}-1$) wrapping
`Gauge`	positive integer (0 to $2^{32}-1$) bounded
`TimeTicks`	time in hundredths of a second
`OCTET STRING`	unrestricted byte string
`DisplayString`	string coded in NVT, 255 chars. max
`IpAddress`	4-byte string
`PhyAddress`	6 byte string
`OBJECT ID.`	numeric identifier...
`SEQUENCE`	structure of named elements
`SEQUENCE OF`	vector of identical elements

OID (*Object IDentifier*)

- namespace tree (**unique** name for each object)
  - Internet objects start 1.3.6.1.



SNMP: MIB *(Management Information Base)*

- object groups defined by MIB II (RFC 1213):

```
1.3.6.1.2.1.1    system
1.3.6.1.2.1.2    interfaces
1.3.6.1.2.1.3    at
1.3.6.1.2.1.4    ip
1.3.6.1.2.1.5    icmp
1.3.6.1.2.1.6    tcp
1.3.6.1.2.1.7    udp
1.3.6.1.2.1.8    egp
1.3.6.1.2.1.10   transmission
1.3.6.1.2.1.11   snmp
```

- other groups and subgroups are defined (other RFCs):

```
1.3.6.1.2.1.17   bridge
1.3.6.1.2.1.43   printer ...
```

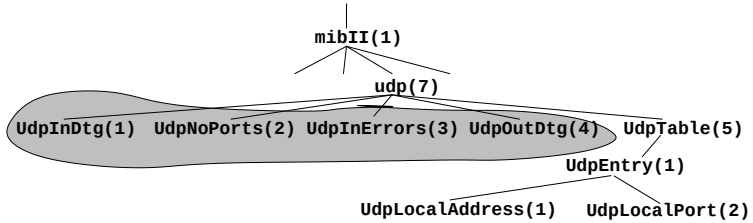
These groups contain **simple** variables or **tables**

MIB: simple variable

Four simple variables in the UDP group:

- the MIB II relates SMI-defined types

udpInDatagrams	Counter	ro	no. datagrams delivered to applications
udpNoPorts	Counter	ro	no. datagrams for which no application
udpInErrors	Counter	ro	no. datagrams undelivered
udpOutDatagrams	Counter	ro	no. datagrams sent

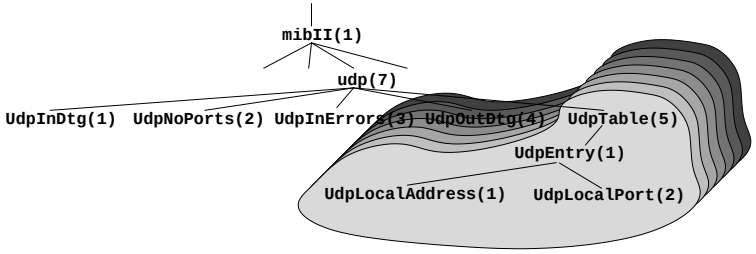


MIB: table variable

One table in the UDP group:

- udpTable indicates the ports being supervised
- udpTable is a **vector** of udpEntry structures

udpLocalAddress	IpAddress	ro	adresse IP locale
udpLocalPorts	[0..65535]	ro	port correspondant



- the table **index** is: udpLocalAddress.udpLocalPorts
  - this index is specified at MIB creation

SNMP: referring to variables

Referring to variables:

- simple: add ".0" to the end
- table: add values to the index field
  - traverse the table's OIDs in **lexicographic** order

short name	OID	value
udpInDatagrams.0	1.3.6.1.2.1.7.1.0	17625
udpLocalAddress.0.0.0.0.53	1.3.6.1.2.1.7.5.1.1.0.0.0.0.53	0.0.0.0
udpLocalAddress.0.0.0.0.161	1.3.6.1.2.1.7.5.1.1.0.0.0.0.161	0.0.0.0
udpLocalPort.0.0.0.0.53	1.3.6.1.2.1.7.5.1.2.0.0.0.0.53	53
udpLocalPort.0.0.0.0.161	1.3.6.1.2.1.7.5.1.2.0.0.0.0.161	161

- this naming scheme allows objects to be specified in UDP messages
  - only the OIDs and the values are transported

SNMP: get-next command

Operator for **lexicographic** traversal of the OIDs:

- returns the next terminal reference
  - `get-next udp ➠ udpInDatagrams.0 = 17625`
- allows one to iterate through variables...
  - `get-next udpInDatagrams.0 ➠ udpNoPorts.0 = 0`
- ... and tables
  - `get-next udpTable`
    `➠ udpLocalAddress.0.0.0.0.53 = 0.0.0.0`
    `get-next udpLocalAddress.0.0.0.0.53`
    `➠ udpLocalAddress.0.0.0.0.161 = 0.0.0.0`
    `get-next udpLocalAddress.0.0.0.0.161`
    `➠ udpLocalPort.0.0.0.0.53 = 53 ...`
- end of table when the name changes:
  - `get-next udpLocalPort.0.0.0.0.161`
    `➠ snmpInPkts.0 = 12`

SNMP: Trap

Agent sends message to admin. **port 162**

version	communauté	type = 4	entreprise	adr. agent	type trap	code entr.	estamp. temp.	nom	valeur	...

- `enterprise`: identifies the agent's creator
  - OID starting with `1.3.6.1.4.1.`
- `agent addr.`: IP address of the agent
- `trap type`:

0	`coldStart`	agent initialised
1	`warmStart`	agent reinitialised
2	`linkDown`	interface disactivated
3	`linkUp`	interface activated
	...	
6	`entr. specific`	see the field `code entr.`

- `code entr.`: sub-code specific to the enterprise
- `timestamp`: number of hundredths of a second since the agent started

Abstract Syntax Notation ASN.1

OSI layer 6 (recommendation X.680 of the ITU)

- properties:
  - universal information representation
  - type associated with data
  - designation by a unique identifier (OID)
  - BNF notation
- description of information exchanged over SNMP:

```
RFC1157-SNMP DEFINITIONS ::= BEGIN
    Message ::= SEQUENCE {
                    version      INTEGER {version-1(0)},
                    community    OCTET STRING,
                    data         ANY
                }
    PDUs ::= CHOICE {
                get-request      GetRequest-PDU,
                get-next-request GetNextRequest-PDU,
                get-response     GetResponse-PDU,
                set-request      SetRequest-PDU,
                trap             Trap-PDU
            }...
```


ASN.1: PDU

get message written in ASN.1:

```
getRequest-PDU ::=  [0]
                    IMPLICIT SEQUENCE {
                        request-id   INTEGER,
                        error-status INTEGER {
                            noError(0), tooBig(1),
                            noSuchName(2), badValue(3),
                            readOnly(4), genErr(5),      -- always 0
                        }
                        error-index  INTEGER,            -- always 0
                        variable-bindings SEQUENCE OF
                            SEQUENCE {
                                name   ObjectName,
                                value  ObjectSyntax
                            }
                    }
```

SNMP: BER encoding

TLV (type, length, value) encoding

- types (1B): the two high bits determine the category
  - UNIVERSAL (00)

0x02	INTEGER
0x04	OCTET STRING
0x05	NULL
0x06	OBJECT IDENTIFIER
0x30	SEQUENCE

  - APPLICATION (01)

0x40	IpAddress
0x41	Counter
0x42	Gauge
0x43	TimeTicks

  - CONTEXT (10)
  - PRIVATE (11)
- data length (1 byte if si < 0x80, else see X.208)
  - length 49 ➠ 0x31, length 242 ➠ 0x8200F2...
- data (value)
  - the OIDs (and successive integers A.B.C.D...) are coded in bytes combining the first two: A*40+B, C, D...

SNMP: example

```
0020                                 30 82 00 f2 02 01   J...D... ..0.....
0030  00 04 06 70 75 62 6c 69  63 a2 82 00 e3 02 01 01   ...publi c.......
0040  02 01 00 02 01 00 30 82  00 d6 30 82 00 0d 06 08   ......0. ..0.....
0050  2b 06 01 02 01 02 01 00  02 01 03 30 82 00 0f 06   +....... ...0....
0060  0a 2b 06 01 02 01 02 02  01 08 01 02 01 01 30 82   .+...... ......0.
0070  00 0f 06 0a 2b 06 01 02  01 02 02 01 08 02 02 01   ....+... ........
0080  02 .. ..
0100                                 .. .. 30 82 00 10   ........ C..,0...
0110  06 0a 2b 06 01 02 01 02  02 01 09 01 43 02 01 2c   ..+..... ....C...
```


MIB RMON

Remote MONitoring (RFC 2819 - STD 59)

Probe for obtaining **statistics** about an administered network

- 9 groups:
  - Ethernet statistics (21 attribute table)
  - network equipment (observed addresses...)
  - statistics matrix (between two stations)
  - frame capture
  - ...
- many extensions
  - RMON protocol identification (RFC 2895, 2896)
  - RMON for switched networks (SMON : RFC 2613)
  - interface management for RMON (IFTOPN : RFC 3144)
  - RMON for DiffServ (DSMON: RFC 3287) ...

Other IETF MIBs (1)

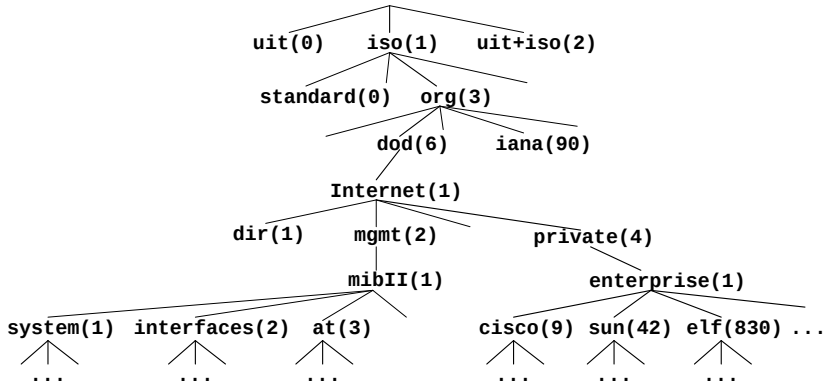
Printer MIB (**RFC 1759** - RFC 3805)

- 274 objects (228 OIDs, including 16 tables)
  - 20 groups:
    - general group
    - input media group
    - output group
    - output dimensions group
    - cover table
    - marker supplies group
    - marker colorant group...

Other IETF MIBs (2)

RFC1230: IEEE 802.4 Token Bus MIB
RFC1381: MIB Extension for X.25 LAPB
RFC1559: DECnet Phase IV MIB Extensions
RFC1593: SNA APPN Node MIB
RFC1611: DNS Server MIB Extensions
RFC1612: DNS Resolver MIB Extensions
RFC1696: Modem MIB
RFC1697: Relational DB Mngmnt System MIB
RFC1724: RIP Version 2 MIB
RFC1748: IEEE 802.5 MIB
RFC2020: IEEE 802.12 Interface MIB
RFC2320: Classical IP and ARP Over ATM MIB
RFC2564: Application Management MIB
RFC1792: TCP/IPX Connection MIB
RFC2605: Directory Server Monitoring MIB
RFC2707: Job Monitoring MIB
RFC2720: Traffic Flow Measurement: Meter MIB
RFC2788: Network Services Monitoring MIB
RFC2789: Mail Monitoring MIB
RFC2790: Host Resources MIB
RFC2863: The Interfaces Group MIB
RFC2922: Physical Topology MIB
RFC2932: IPv4 Multicast Routing MIB
RFC2933: IGMP MIB
RFC2934: PIM MIB for IPv4
RFC2981: Event MIB
RFC2982: Distributed Management Expression MIB
RFC3014: Notification Log MIB
RFC3144: RMon MIB Extensions for Interface
RFC3287: RMon MIB Extensions for DiffServ...
RFC6765: xDSL Multi-Pair Bonding (G.Bond) MIB
RFC6766: xDSL Multi-Pair Bonding TDIM MIB
RFC6767: Ethernet-Based xDSL Multi-Pair Bonding MIB
RFC6768: ATM-Based xDSL Bonded Interfaces MIB
RFC6779: Definition of Managed Objects for the NDP
RFC6825: TE Database MIB for MPLS-TE/GMPLS
RFC6933: Entity MIB (Version 4)
RFC7052: Locator/ID Separation Protocol (LISP) MIB
RFC7124: Ethernet in the 1st Mile Copper (EFMCu) MIB
RFC7147: Definitions of Managed Objects for iSCSI
RFC7184: Def. of Managed Objects for the OLSRv2
RFC7330: Def. of TCs for BFD
RFC7331: BFD MIB
RFC7367: Def. for the MANET Simplified Multicast
RFC7388: Def. for 6LoWPANs
RFC7420: PCEP MIB Module
RFC7453: MPLS-TP TE MIB
RFC7460: Monitoring & Control MIB for Power & Energy
RFC7461: Energy Object Context MIB
RFC7577: Def. of Managed Objects for Battery Monit.
RFC7659: Def. of Managed Objects for NATs
RFC7666: MIB for VMs Controlled by a Hypervisor
RFC7697: MPLS-TP OAM Identifiers MIB
RFC7784: TRILL OAM MIB
RFC7856: Softwire Mesh MIB
RFC7870: Dual-Stack Lite MIB for AFTRs
RFC7939: Def. of Managed Objects for NDP
RFC8038: Exporting MIB Variables Using IPFIX Proto
RFC8150: MPLS Transport Profile Linear Protection MIB
...

MIB constructor



SNMP versions

Several versions have been standardized:

- **SNMPv1** defined in RFC 1157 (1990) simple and not secure
  ➠ still widely used
- **SNMPv2** defined in RFC 1901 to 1908 with extensions (`get-bulk` and `inform` queries, MIB SNMPv2 and SNMPv2-M2M) and security but no constructor concensus
  - **SNMPv2c** the new functionalities but without the security *(community-based)*
  - **SNMPv2u** simplified version of the new security mechanism *(user-based)*
- **SNMPv3** defined in RFC 3410 to 3418, again brings in security
  - the only IETF standard is v3 (STD-62)
  - multi-version use: RFC 3584

SNMP: limitations

- measurement should not perturb the network
- latency
- MIB owners
- security
  - *(packet sniffing)* to know the community
  - *(IP spoofing)* facilitated by UDP

➠ improvements with **SNMPv3**