Computer Networks - Applications

2017/18 Q2

Jaime Delgado

DAC – UPC

Contents

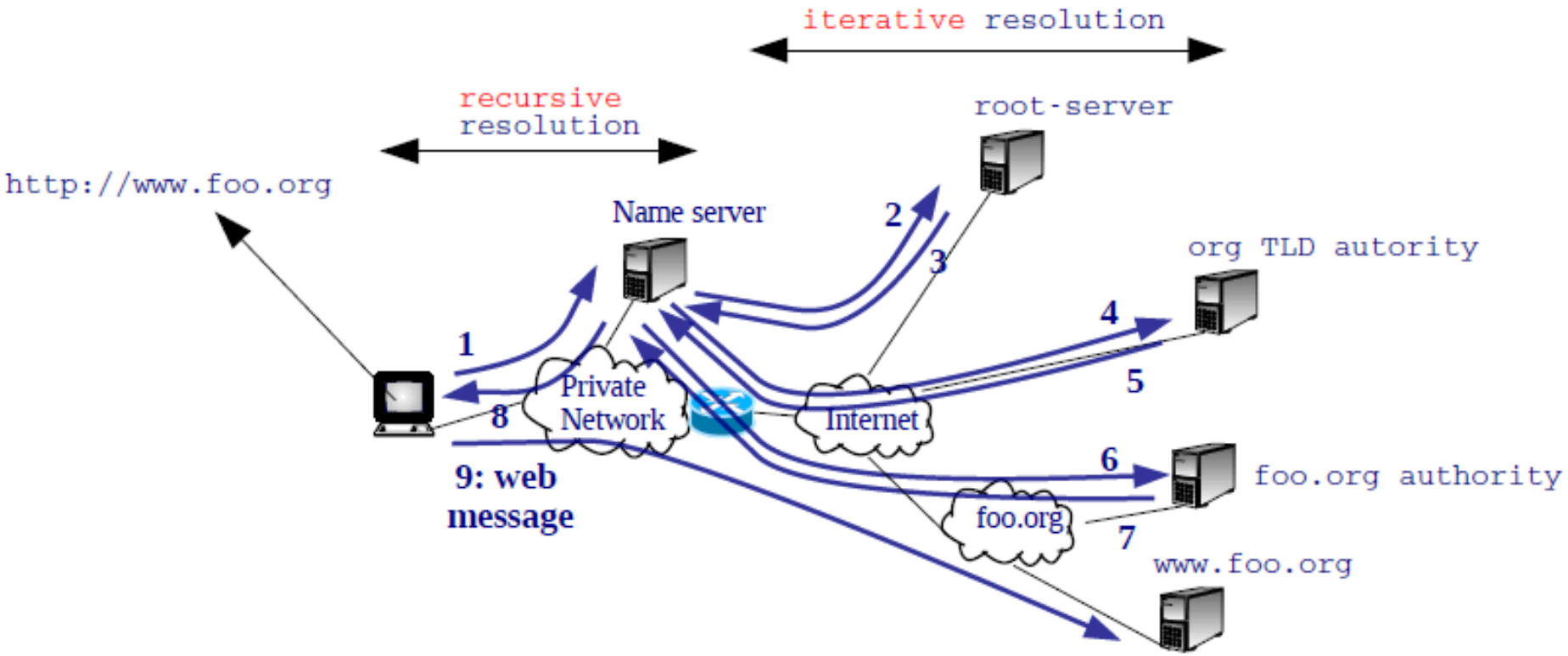
- DNS (Domain Name System).
- e-mail: Protocols and formats.
- Web elements & HTTP.
- HTML.
- XML.
- Characters in communications.

DNS (Domain Name System)

- Application protocol needed for IP:
  - Obtain IP addresses from “names”.
- Domain/sub-domain/host name:
  - Hierarchical structure: “*myhost.ac.upc.edu*”
  - *.edu* is a TLD (Top Level Domain).
- IP of *myhost.ac.upc.edu* (node/host name) know by local **Name Server** of *ac.upc.edu*
- DNS format & protocol needed.

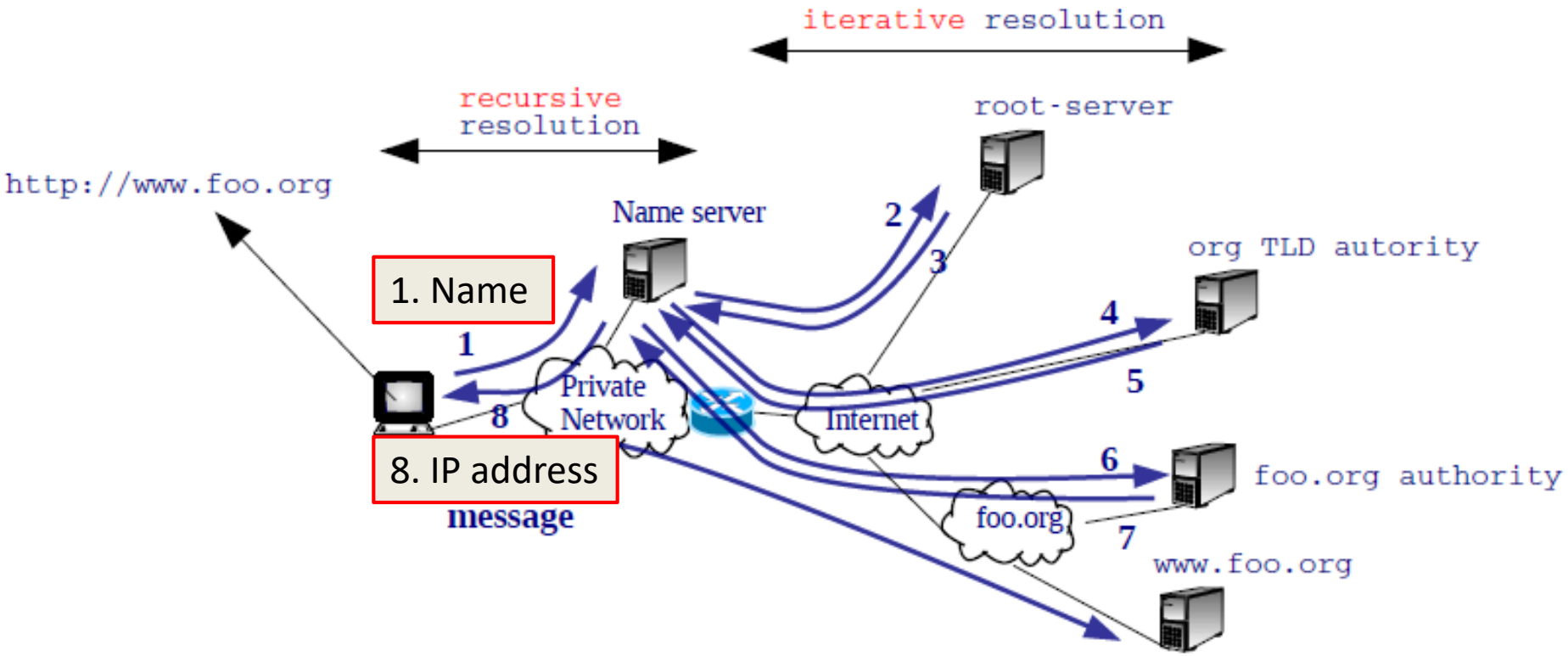
DNS (Domain Name System)

- Application protocol:



DNS (Domain Name System)

- Application protocol:

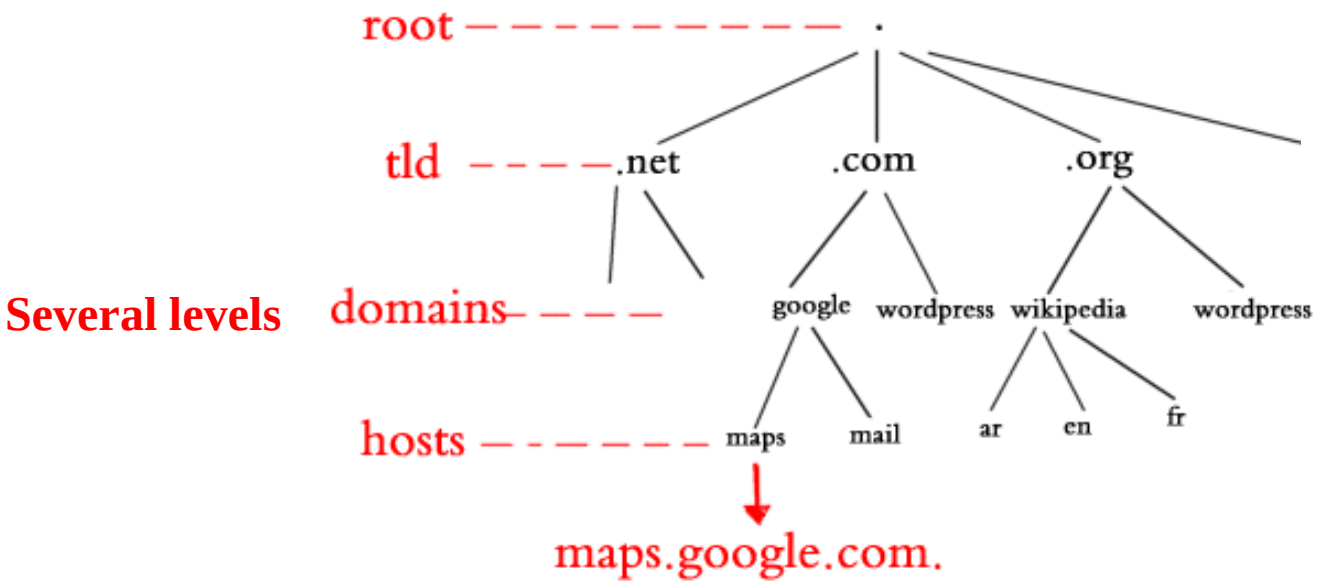


DNS

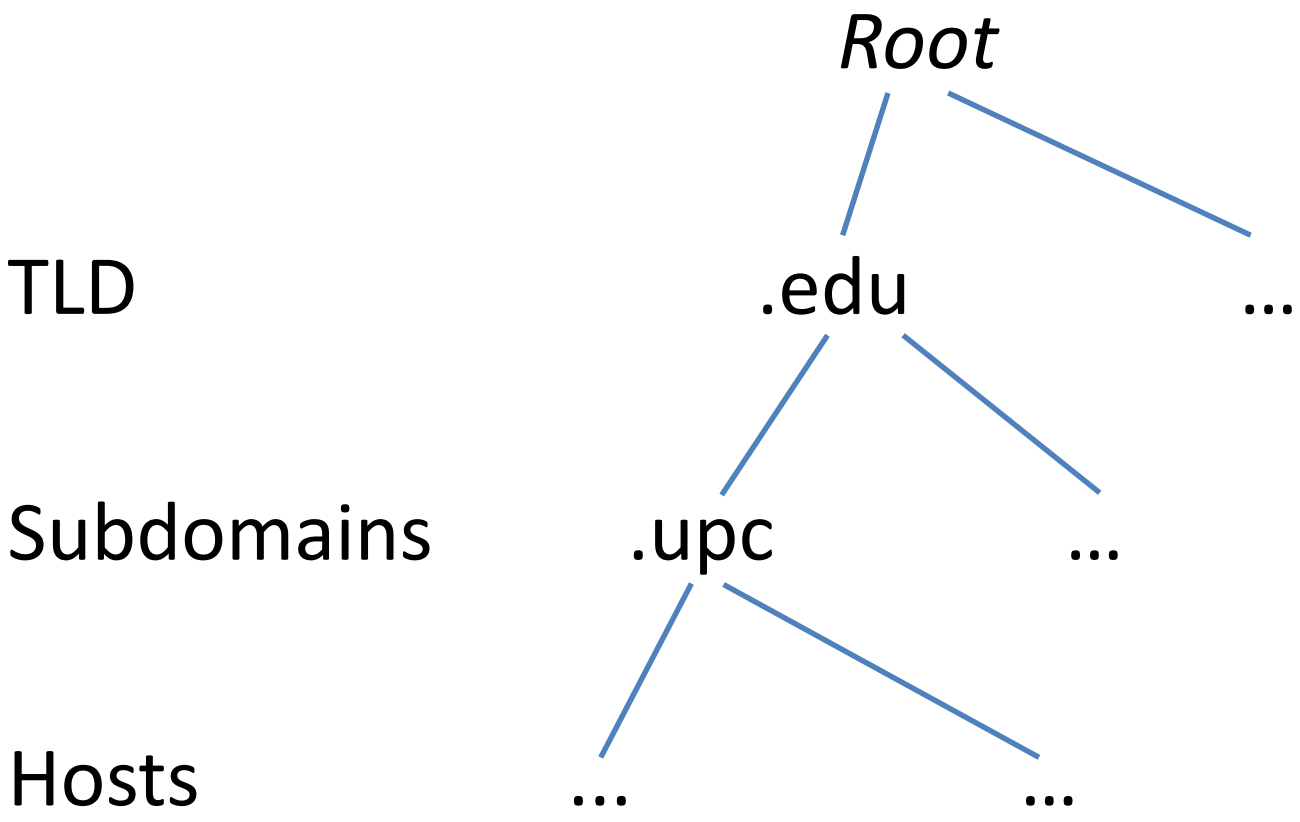
- **Domain Name System (RFC 1034, 1035)**
- Allows using names instead of IP addresses: e.g. *www.ac.upc.edu*
- Names: domain and subdomain names.
- The DNS is like a worldwide distributed data base.
- Authoritative name servers.
- DNS data base entries: *Resource Records* (RR).
- Information on a name: 1 or more RRs.
- *Protocol:* *Local name servers.*
  Iterative vs. Recursive access.

DNS

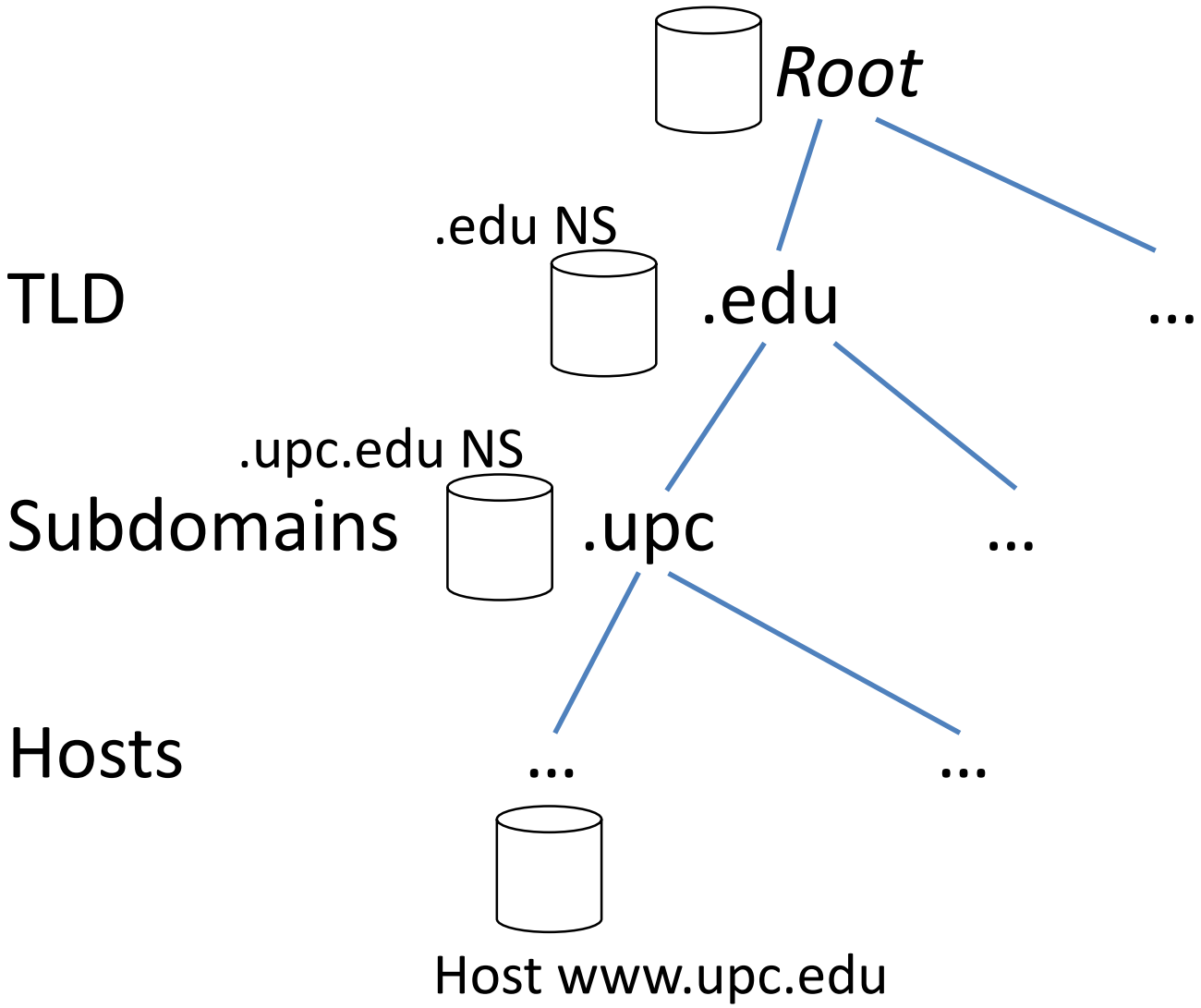
- Domain/sub-domain/host name:
  - Hierarchical structure: "*myhost.ac.upc.edu*"
  - *.edu* is a TLD (Top Level Domain).
- 13 root servers: *a.* to *m.root-servers.org*



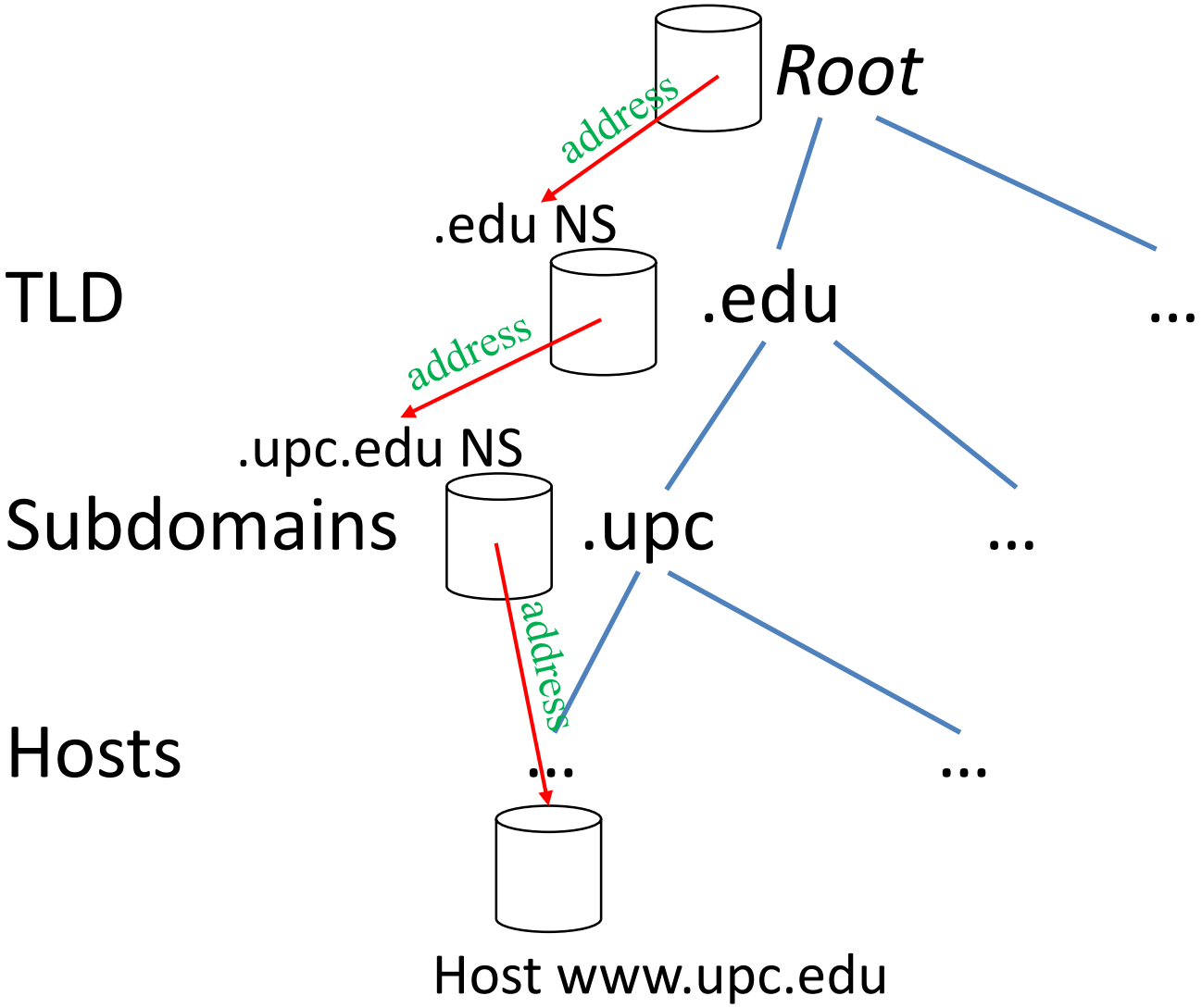
DNS (information hierarchy)



DNS (information hierarchy)

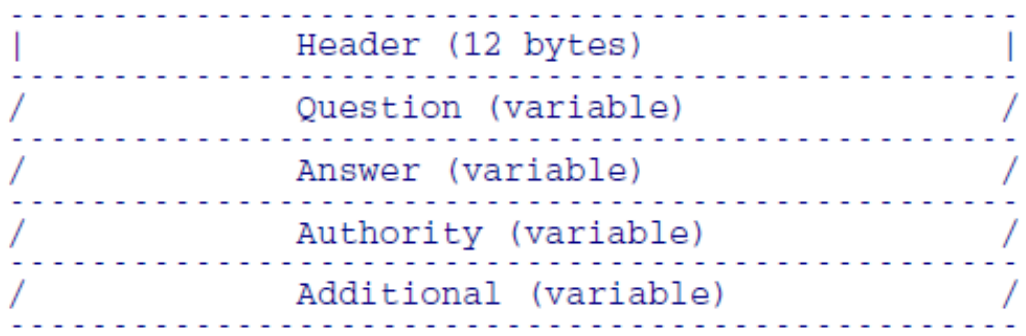


DNS (information hierarchy)



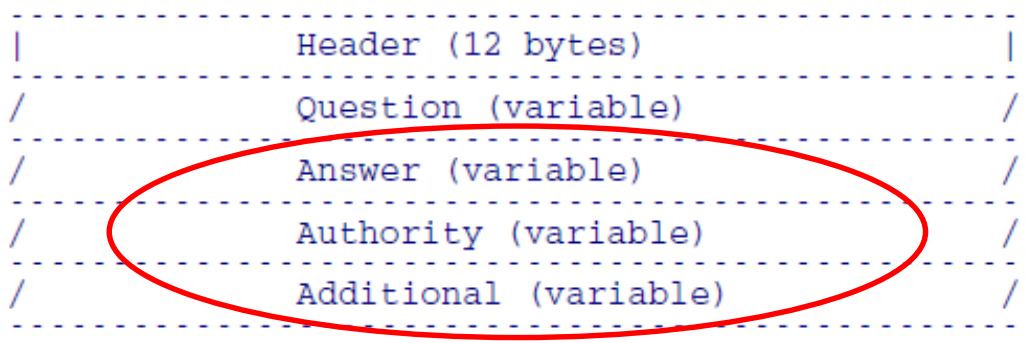
DNS – data structures

- Message format:
  - Header: type of message.
  - Question: What is to be resolved.
  - Answer: Answer to question.
  - Authority: Domain authority names.
  - Additional: Typically, the authority name's addresses.



DNS – data structures

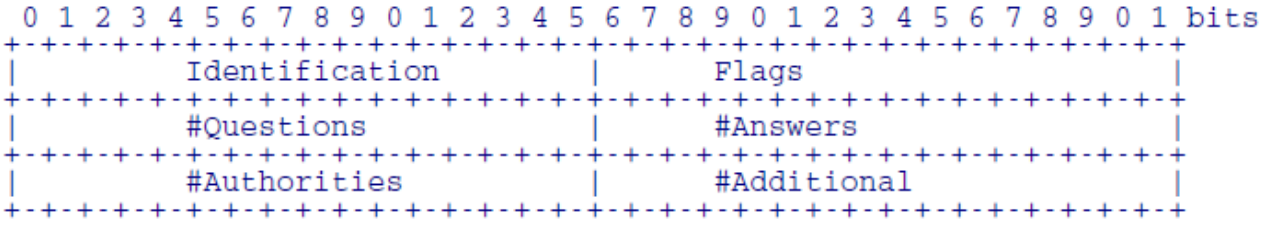
- Message format:
  - Header: type of message.
  - Question: What is to be resolved.
  - Answer: Answer to question.
  - Authority: Domain authority names.
  - Additional: Typically, the authority name's addresses.



DNS – data structures

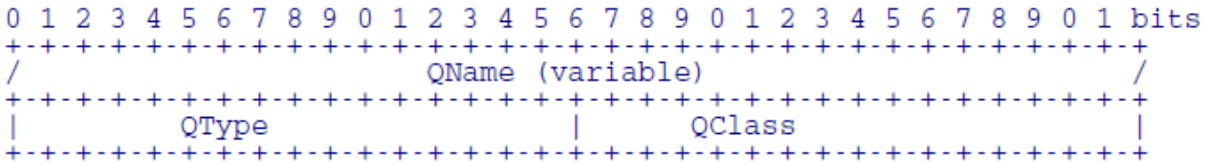
- Message format - Header:

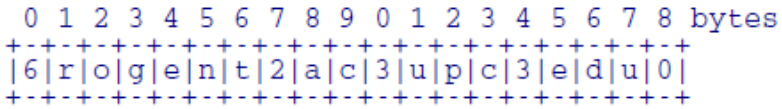
- Identification: 16 random bits used to match query/response
- Flags. Some of them:
  - Query-Response, QR: 0 for query, 1 for response.
  - Authoritative Answer, AA: When set, indicates an authoritative answer.
  - Recursion Desired, RD: When set, indicates that recursion is desired.
- The other fields indicate the number of Questions, Answer, Authority and Additional fields of the message.



DNS – data structures

- Message format - Question:
  - QName: Indicates the name to be resolved.
  - QType: Indicates the question type:
    - Address, A.
    - Name Server, NS.
    - Pointer, PTR: For an inverse resolution.
    - Mail Exchange, MX: Domain Mail Server
  - Qclass: For Internet addresses is 1.



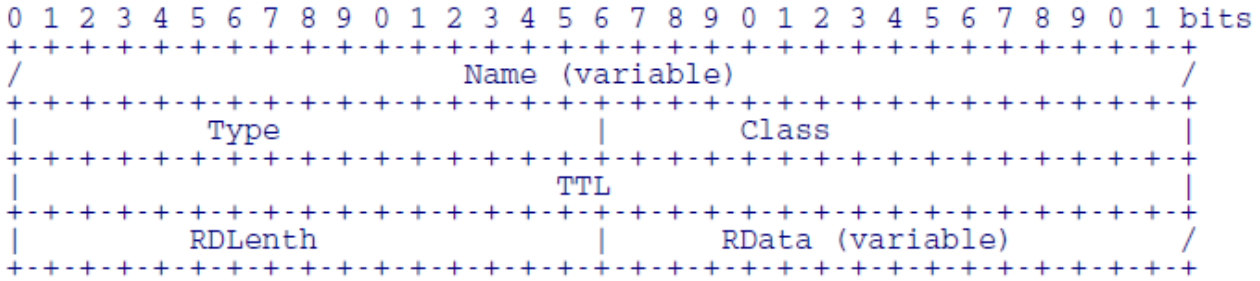


Codification example of `rogent.ac.upc.edu`

DNS – data structures

- Message format – Resource Records (RRs):

- The fields Answer, Authority and Additional are composed of RRs:
  - Name, Type, Class: The same as in the Question field.
  - TTL (Time To Live): Number of seconds the RR can be cached.
  - RDLenth: RR size in bytes.
  - Rdata: E.g. An IP address if the Type is 'A', or a name if the Type is 'NS', 'MX' or 'CNAME'.



DNS – data structures

- Message format – Example:

Query message:

- 36388: Identifier.
- +: Recursion-Desired is set.
- A?: Qtype = A.
- ns.uu.net.: Name to resolve.

Response message:

- 36388: Identifier.
- q: A? ns.uu.net.: Repeat the Question field.
- 1/2/2: 1 Answers, 2 Authorities, 2 Additional follows.
- ns.uu.net. A 137.39.1.3: The answer (RR of type A, address: 137.39.1.3).
- ns: ns.uu.net. NS auth00.ns.uu.net., ns.uu.net. NS auth60.ns.uu.net.: 2 Authorities (RRs of type NS: the domain ns.uu.net. authorities are auth00.ns.uu.net. and auth60.ns.uu.net).
- ar: auth00.ns.uu.net. A 198.6.1.65, auth60.ns.uu.net. A 198.6.1.181: 2 Additional (RRs of type A: authorities IP addresses).

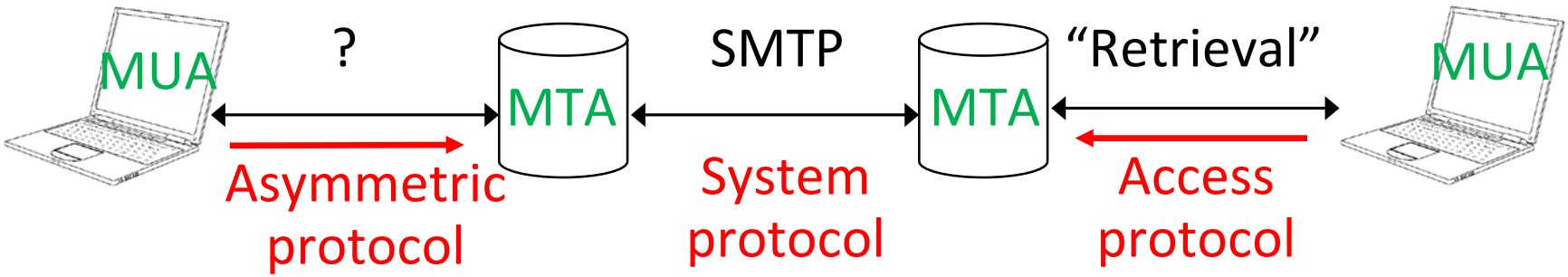
Contents

- DNS (Domain Name System).
- e-mail: Protocols and formats.
- Web elements & HTTP.
- HTML.
- XML.
- Characters in communications.

E-mail

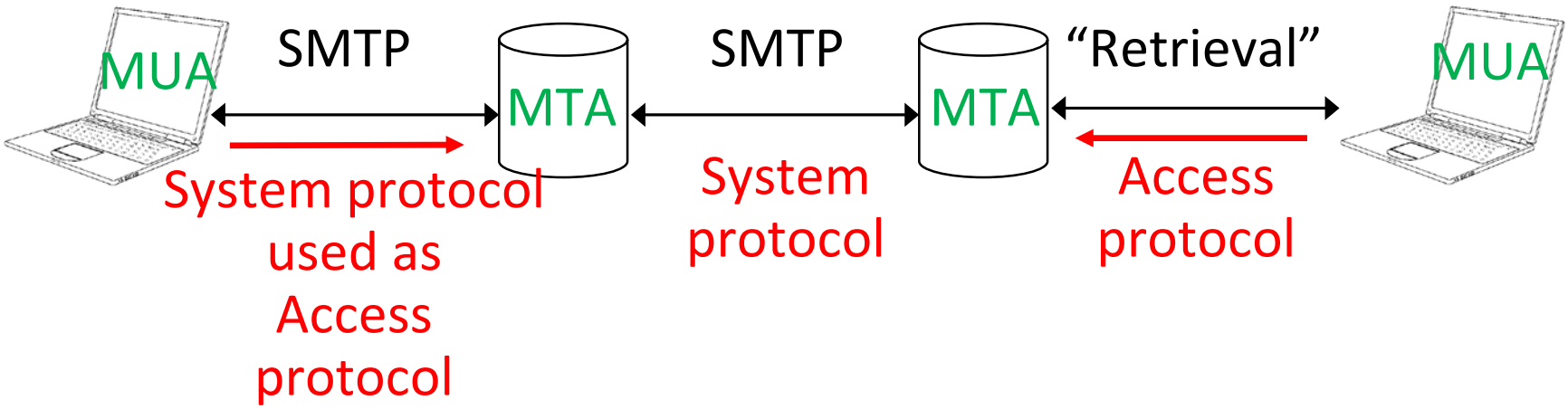
- Protocols (Dialogue) and Formats (Information).
- Protocol → Architecture.

E-mail architecture



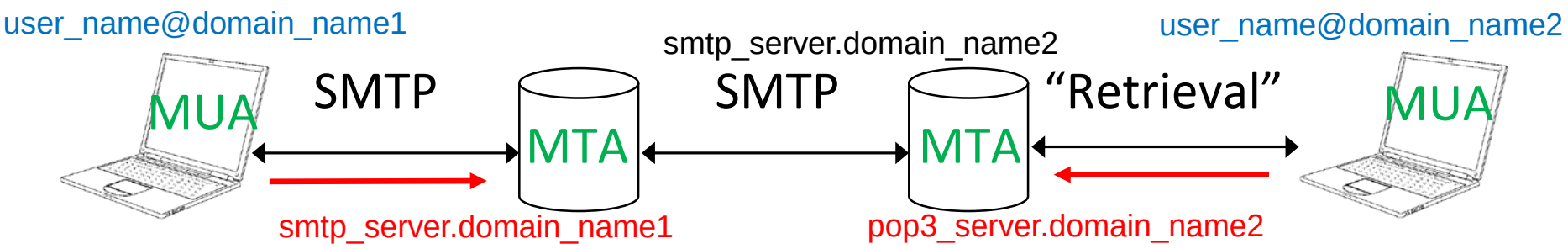
- MUA: Mail User Agent
- MTA: Mail Transfer Agent
- *SMTP: Simple Mail Transfer Protocol*

E-mail protocols



- "Retrieval" protocols (mailbox access):
  - POP3 *(Post Office Protocol)*
  - IMAP *(Internet Message Access Protocol)*
- SMTP: Simple Mail Transfer Protocol

E-mail protocols

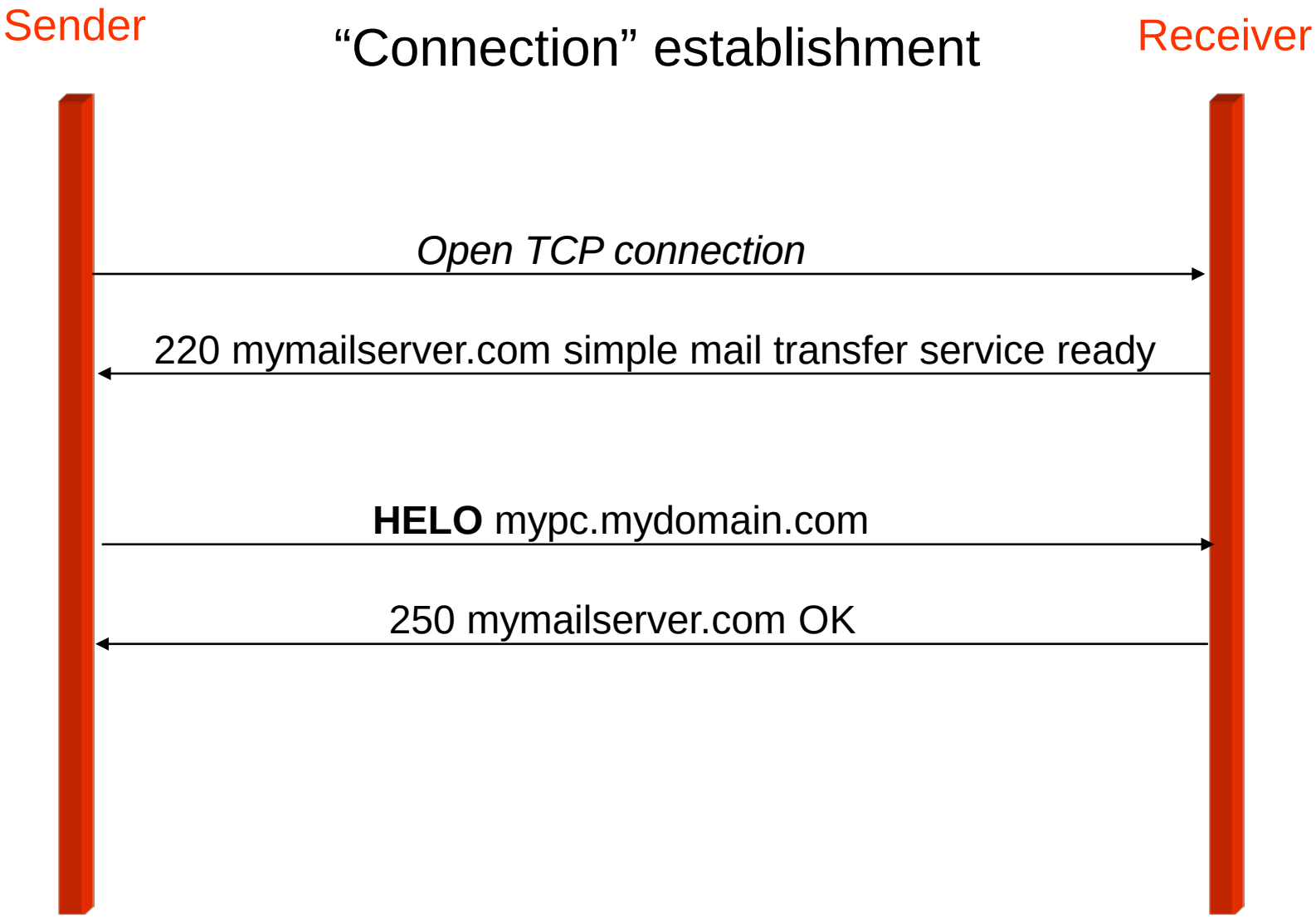


- “Retrieval” protocols (mailbox access):
  - POP3 *(Post Office Protocol)*
  - IMAP *(Internet Message Access Protocol)*
- SMTP: Simple Mail Transfer Protocol

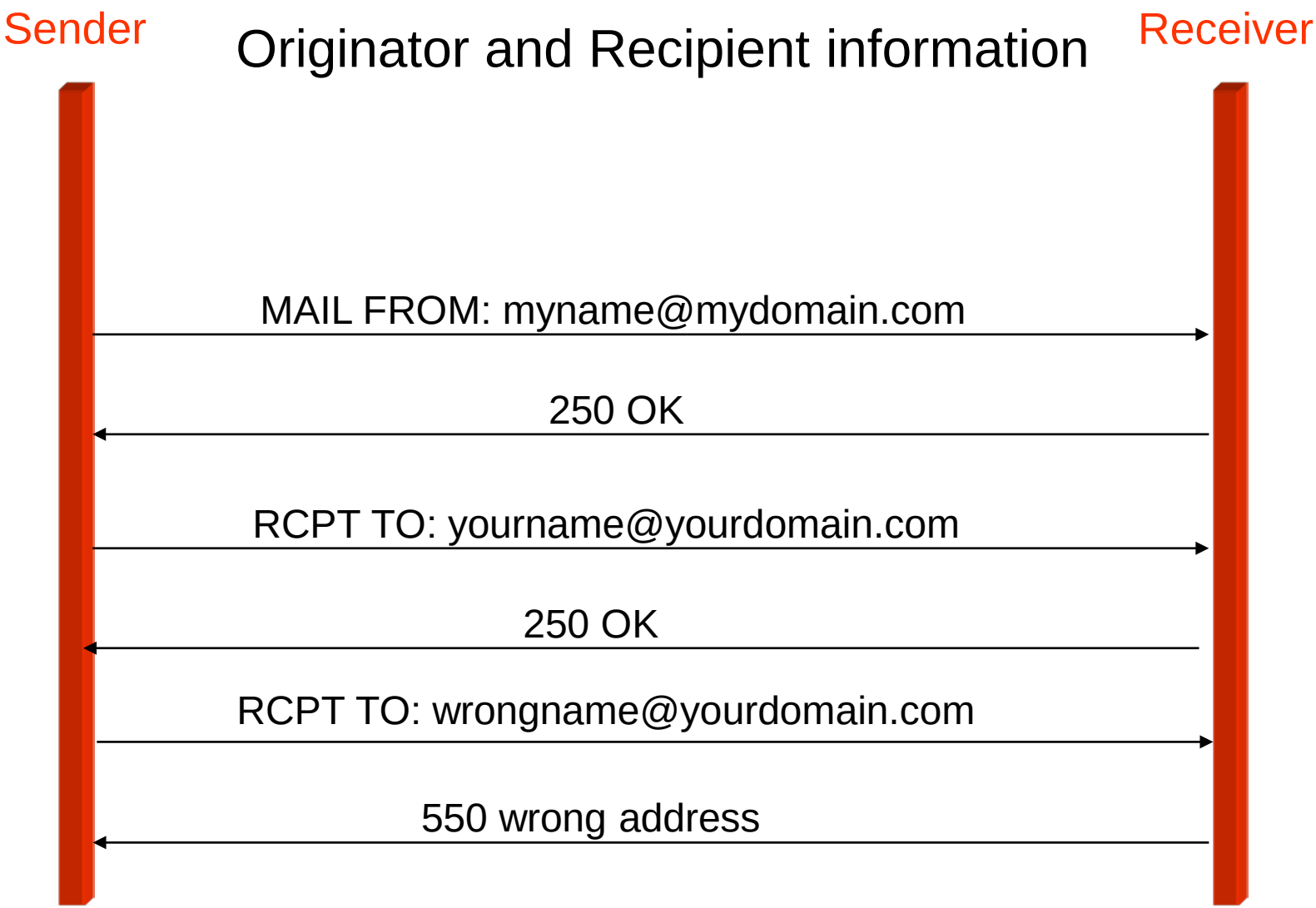
SMTP

- *Simple* Mail Transfer Protocol
- RFC 821 (the transfer protocol), 1982
- Related standards:
  - RFC 974 (DNS-MX, “Mail routing and the domain system”)
  - RFC 822 (Message format)
- RFC 2821 obsoletes RFC 821 & RFC 974, 2001
- Simple 7 bits ASCII commands:
  HELO/EHLO, MAIL, RCPT, DATA, QUIT, …
- No “Store-and-Forward”. (MTAs yes!).

SMTP protocol



SMTP protocol

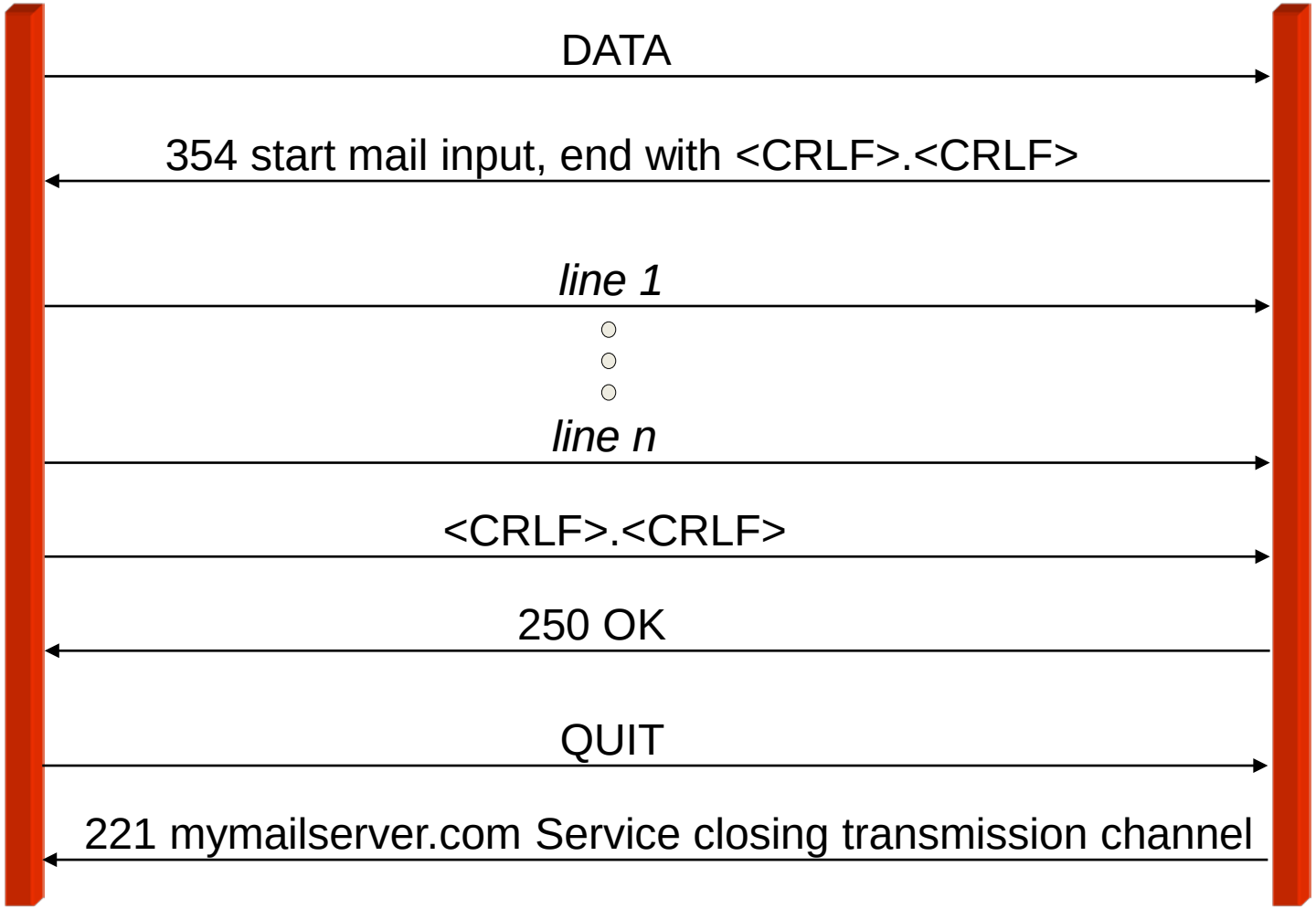


SMTP protocol

Sender

Message transmission and Close

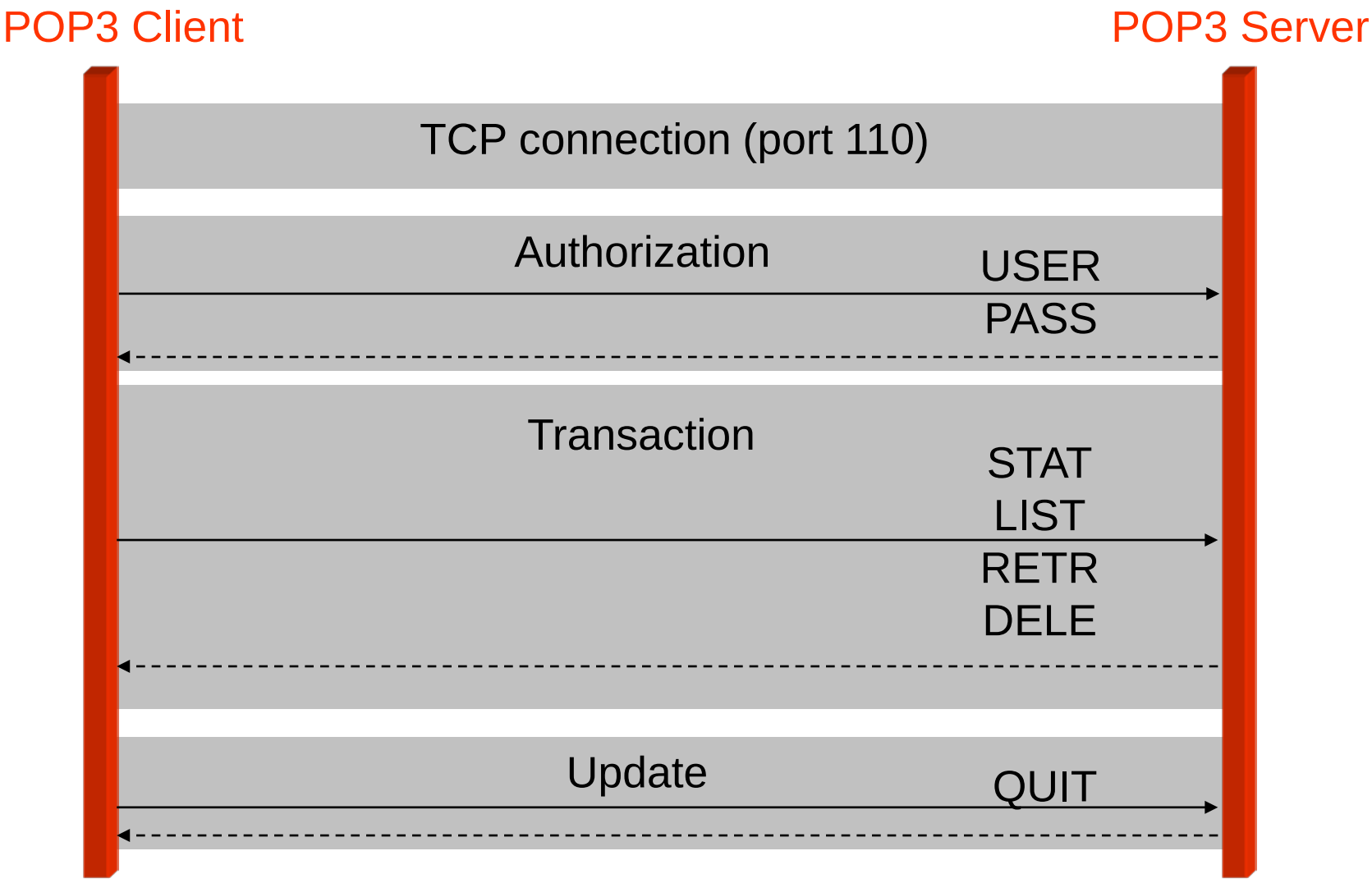
Receiver



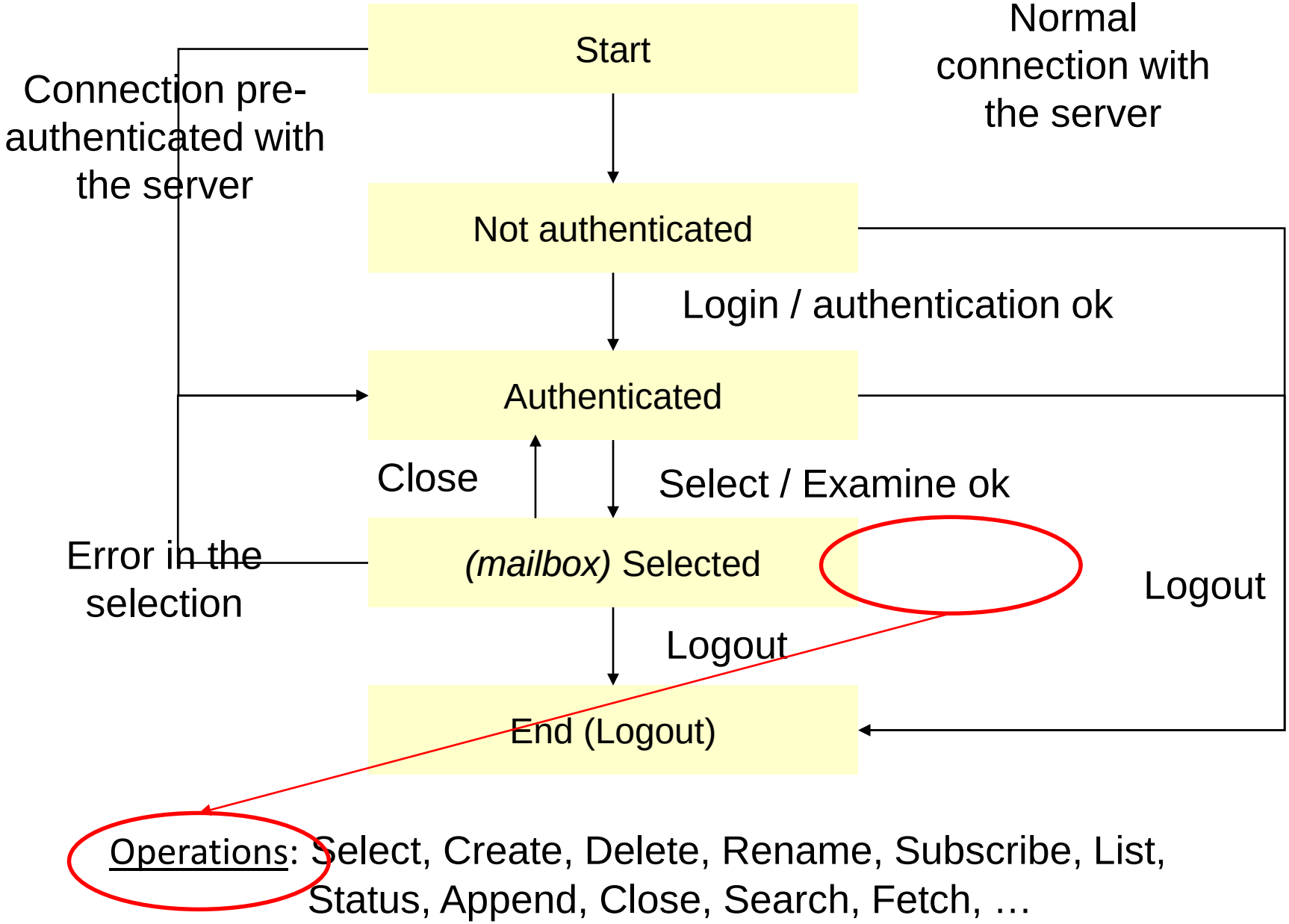
Mailbox Access protocols

- Post Office Protocol (POP) version 3 (POP3)
  - RFC 1939 (1996)
  - Client-server protocol (Asymmetric)
  - Messages retrieved from the mail server (copied locally).
- Internet Message Access Protocol (IMAP)
  - RFC 3501 (2003). 1st version 4 in RFC 1730 (1994). 1st RFC (version 2) in 1988 (RFC 1064).
  - Client-server protocol (Asymmetric)
  - Messages accessed and managed (folders, …) at the server

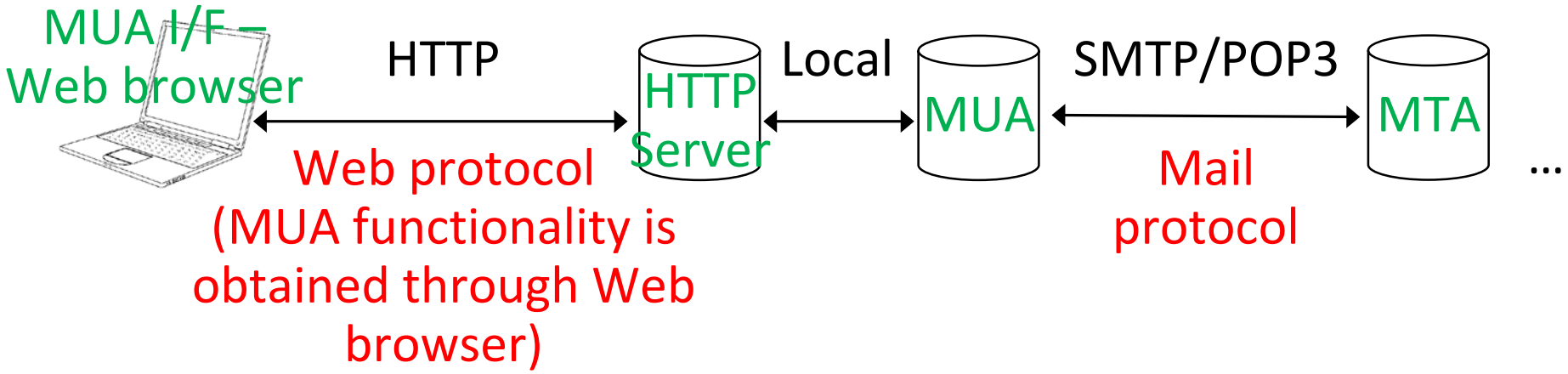
Post Office Protocol 3 (POP3)



IMAP


Start
Normal connection with the server
Connection pre-authenticated with the server
Not authenticated
Login / authentication ok
Authenticated
Close
Select / Examine ok
Error in the selection
(mailbox) Selected
Logout
Logout
End (Logout)
Operations: Select, Create, Delete, Rename, Subscribe, List, Status, Append, Close, Search, Fetch, …

Webmail



- Web front-end for mail services. The MUA is a web browser.
- Real protocol to access the services: HTTP (web).
- The HTTP server machine uses SMTP or POP3, as required.

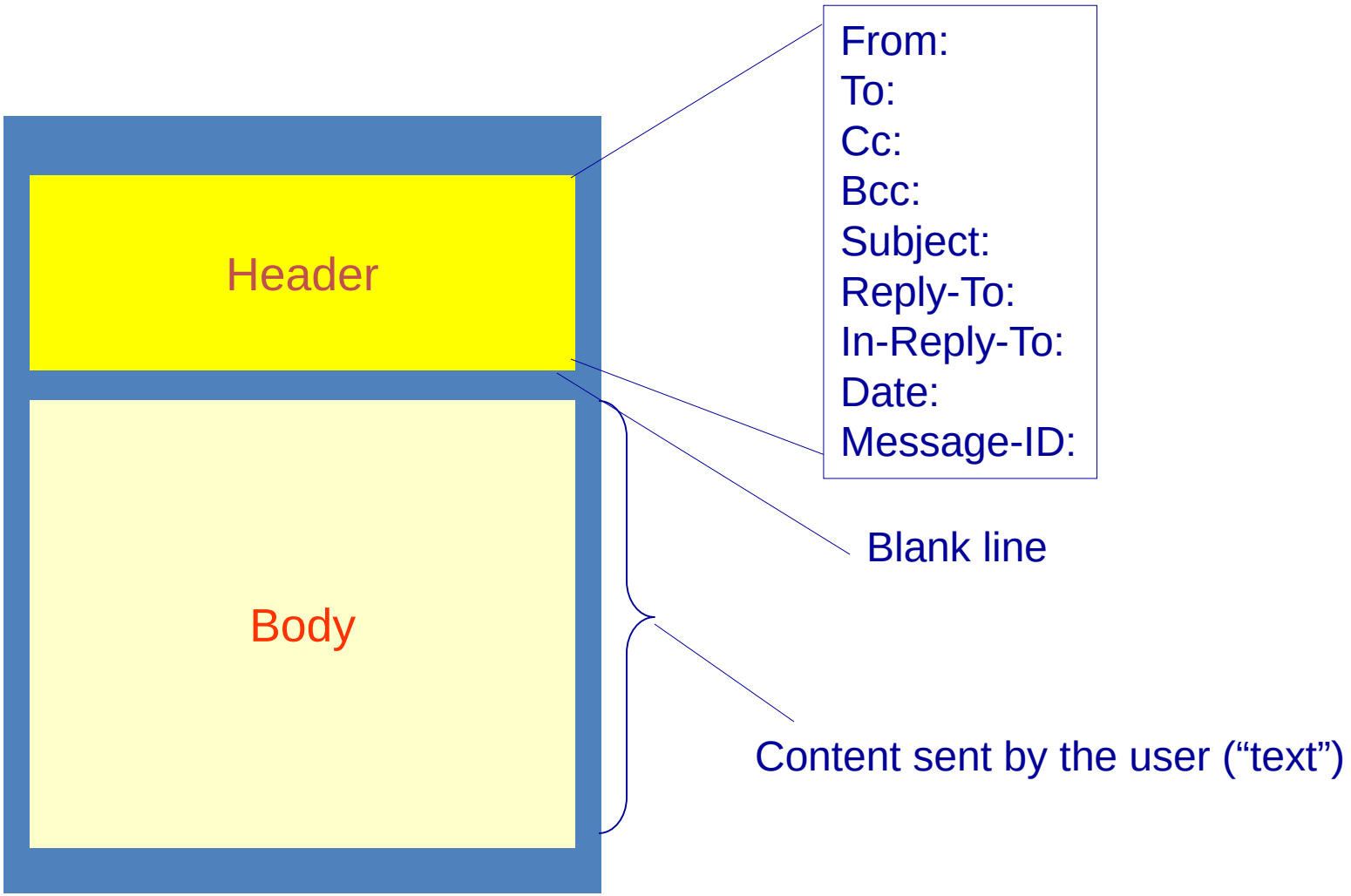
E-mail

- Protocols (Dialogue) and Formats (Information).
- Protocol → Architecture.

- **Formats**:
- Protocol: Header + User data (from the application)
- User data = *Message*
- *Message* = Header (elements) + Data (from user)
  RFC 822 → RFC 5322 (2008) "Internet Message Format"

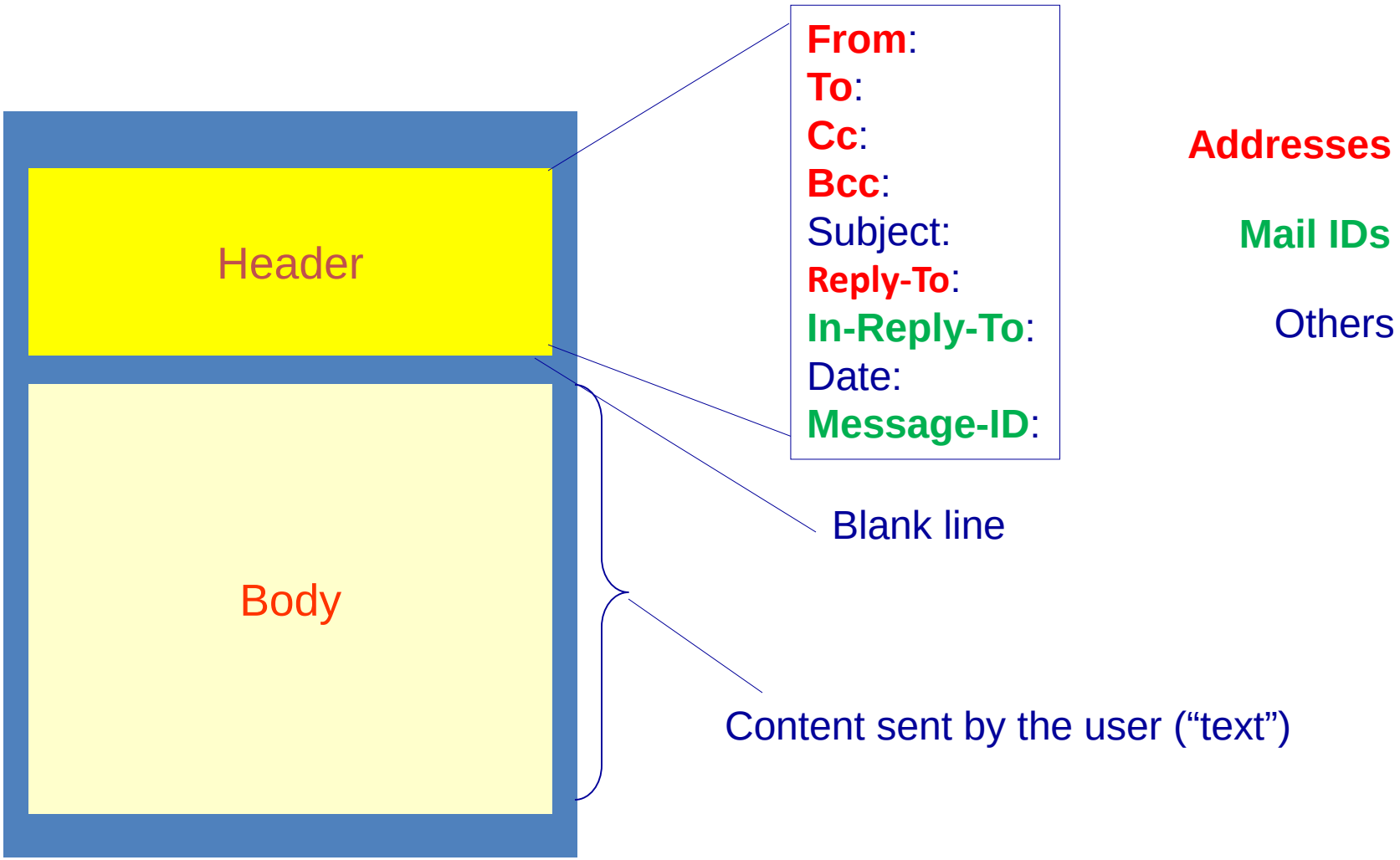
- Address: user_name@domain_name

SMTP message format



All lines separated by the <CR><LF> characters

SMTP message format



All lines separated by the <CR><LF> characters

MIME

Multipurpose Internet Mail Extensions

- **RFCs**: 1341+1342 (‘92), 1521+1522 (‘93)
  2045 (format), 2046 (media types), 2047/8/9 **(‘96)**
  + updates + compl. (registration 6838 (‘03), …)
- **Main new features (“extensions”):**
  - Inclusion of non-ASCII data (*all 8 bits used!*) → “types”
  - Multipart messages
- Approach: Adding new header elements →
  `Content-Type`, …
- **MIME goes further away than mail!**
  (HTTP, …, *when “8-bit” files are needed*)

MIME header elements

- **MIME-Version**
- **Content-Type**
- **Content-Transfer-Encoding**
- Content-ID
- Content-Description
- *Additional header fields*:
  - Content-Disposition (inline/attachment) *(RFC 2183)*
  - Content-Language *(RFC 3282)*,
  - …

MIME Content/Media types

- application
- audio
- example (RFC4735, ‘06)
- font (RFC8081, Feb’17!)
- image
- message
- model (RFC2077, ‘97)
- multipart
- text
- video

MIME Content/Media types

- application
- audio
- example
- font
- image
- message
- model
- multipart
- text
- video

MIME Content/Media types

- application
- **audio**
- example
- font
- **image**
- message
- model
- multipart
- **text**
- **video**

MIME Content/Media types

- **application**
- **audio**
- example
- font
- **image**
- message
- model
- multipart
- **text**
- **video**

MIME Content/Media types

- application
- audio
- (example)
- (font)
- image
- **message**
- (model)
- **multipart**
- text
- video

MIME content types

- `Content-Type` element structure:
  - **`type/subtype`**

- Examples of type/subtype:
  - application/pdf, application/msword, application/soap+xml, application/vnd.ms-powerpoint, application/vnd.nokia.radio-preset, ...
  - audio/GSM, audio/mpeg, audio/vnd.dolby.mps, …
  - image/gif, image/jpeg, image/png, image/vnd.adobe.photoshop, ...
  - text/plain, text/html, text/vnd.dvb.subtitle, …
  - message/rfc822, message/http, ...
  - model/iges, …
  - multipart/mixed, multipart/alternative, …
  - video/H264, video/mp4, video/vnd.nokia.videovoip, …

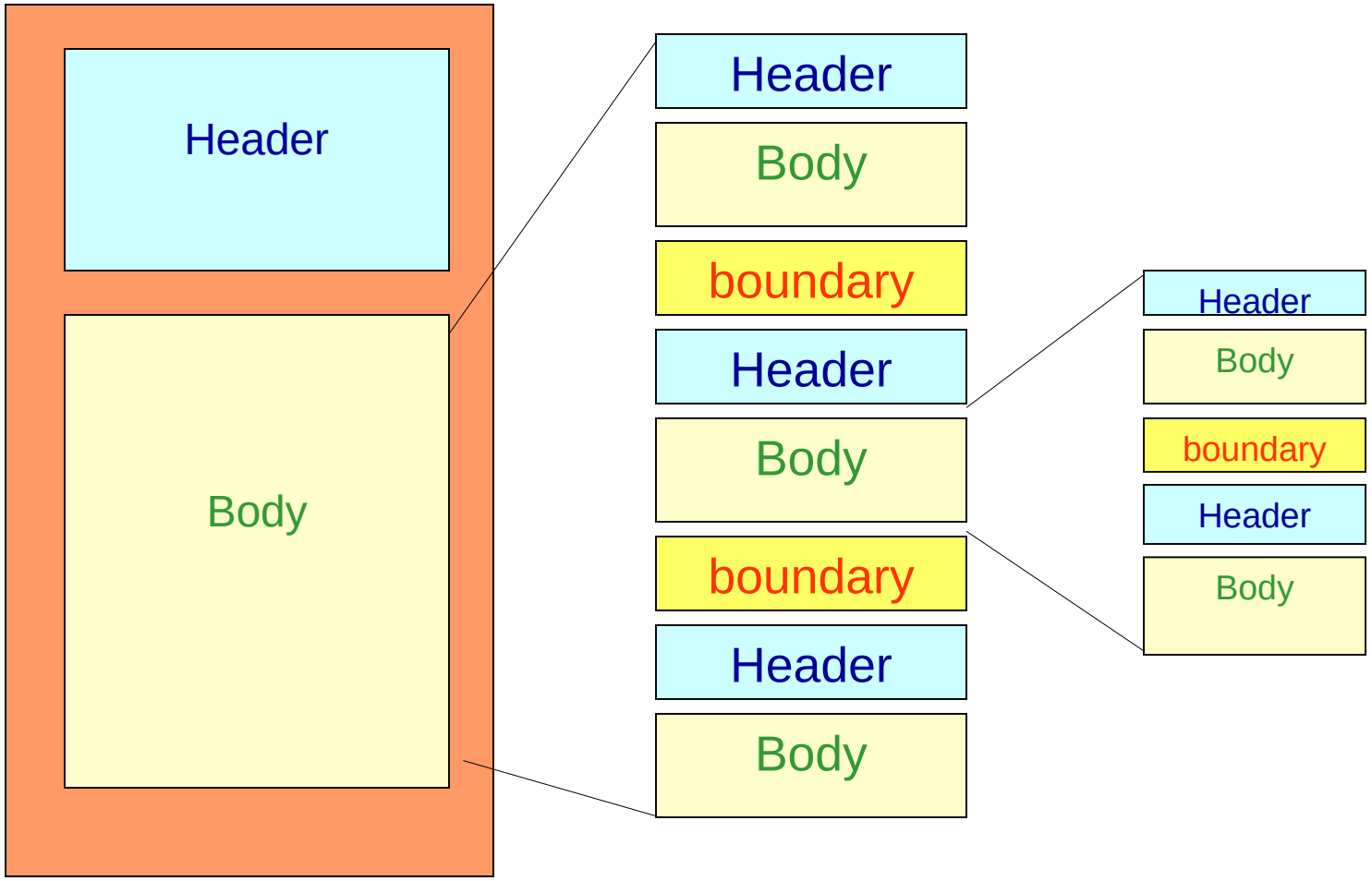
MIME Content/Media types

- MIME Content subtypes for every type *(September 2017)*:
  - application (386+860*=1246) *(standards+vendor)*
  - audio (109+39=148)
  - example (No subtypes)
  - font (6+0=6)
  - image (25+29=54)
  - message (19+1=20)
  - model (9+14=23)
  - multipart (15+1=16)
  - text (46+27=73)
  - video (44+34=78)
  - TOTAL: 659+1005=1664 (some repeated)

  http://www.iana.org/assignments/media-types

* 71 for vnd.openxmlformats-officedocument

MIME multipart message

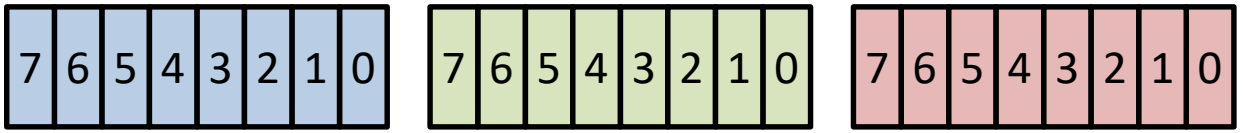


MIME Content-Transfer-Encoding

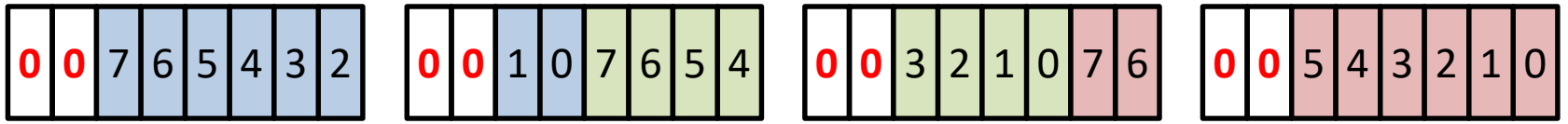
- With “normal” SMTP servers (only 7-bit support):
  - “7bit”, “quoted-printable”
  - “**base64**”
- 8 bit support:
  (Extended SMTP: RFC1869 (1995))
  (8BIT MIME: RFC 1653 (1994) → RFC 6152 (2011))
  (BINARY MIME: RFC 3030 (2000))
  - “8bit”, “binary”

Base64 encoding

Bytes to transmit (8 bits either 0 or 1):



Encoded bytes (sent as ASCII with 2 higher bits set to 0):



Only ASCII values from 0 to 63
(**64** posible values)

Inefficiency: 4 bytes transmitted for every 3!

Contents

- DNS (Domain Name System).
- e-mail: Protocols and formats.
- Web elements & HTTP.
- HTML.
- XML.
- Characters in communications.

Web elements

- Protocol
- Information (format)
- LINK to information

Web elements

- Protocol
  - **HTTP** (HyperText Transfer Protocol)
- Information (format)
  - **HTML** (HyperText Markup Language)
- LINK to information
  - **URI** (Uniform Resource Identifier):

    URN (Name), **URL** (Locator)

Web elements

- Protocol
  - **HTTP** (HyperText Transfer Protocol)
- Information (format)
  - **HTML** (HyperText Markup Language)
- LINK to information
  - **URI** (Uniform Resour

URN (Name), **URL** (Locator)

→ Internationalized Resource Identifier (IRI)

Web elements

- Others:
  - Web browsers (in HTTP Client)
  - Web servers (in HTTP Server)

- But WWW access is not the only use …

Links - URI (Uniform Resource Identifier)

- URI Generic Syntax: RFC 3986 (2005)

URI =
scheme ":" hier-part ["?" query] ["#" fragment]

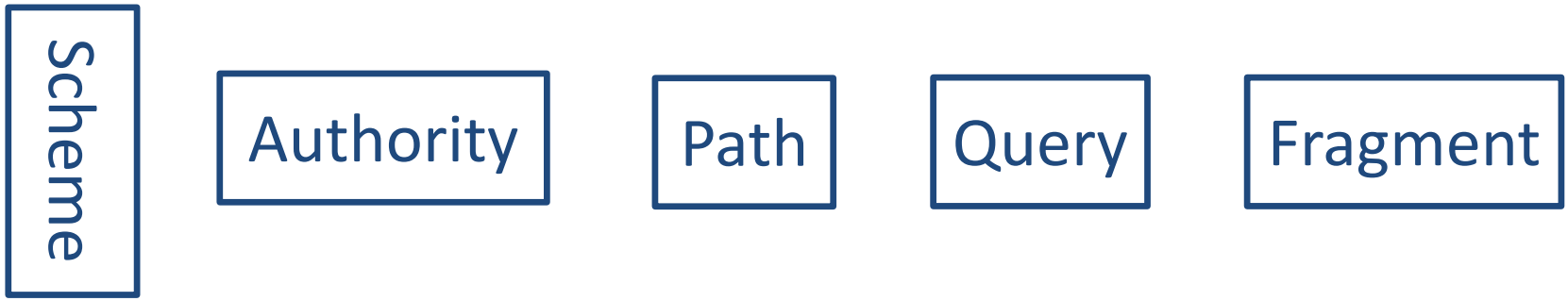
hier-part = "//" authority path-abempty
/ path-absolute
/ path-rootless
/ path-empty

Links - URI (Uniform Resource Identifier)

- EXAMPLES:

URL:

foo://example.com:8042/over/there?name=ferret#nose

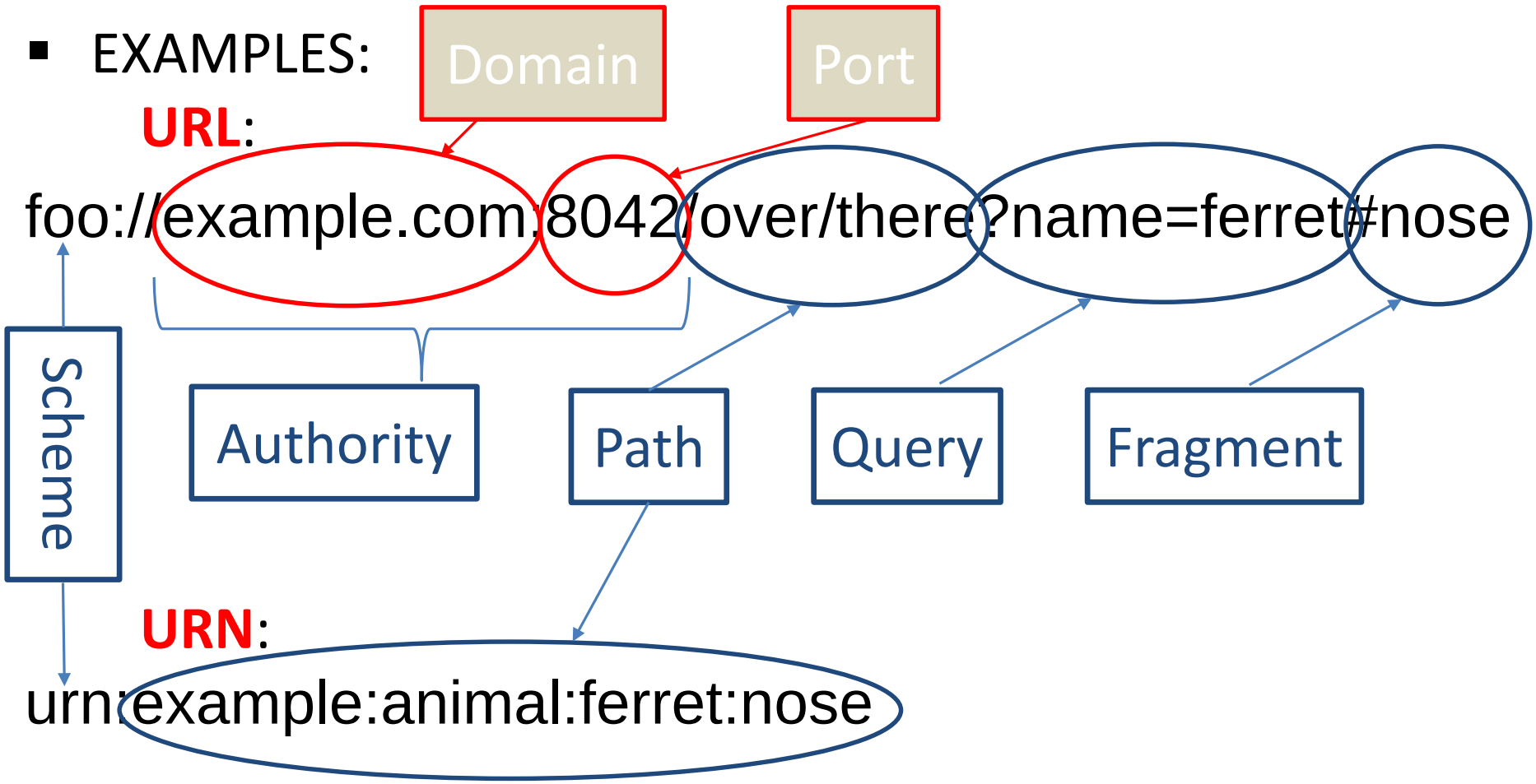


URN:

urn:example:animal:ferret:nose

Links - URI (Uniform Resource Identifier)

- EXAMPLES:

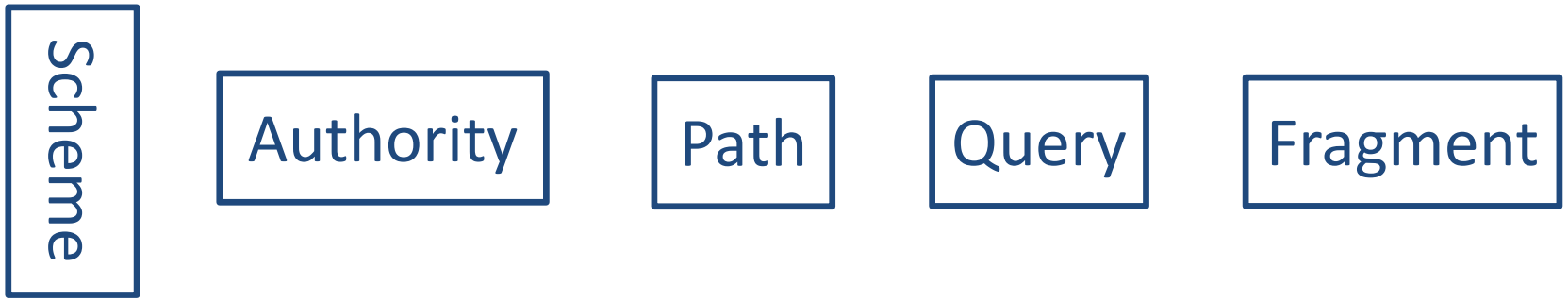


Links - URI (Uniform Resource Identifier)

- EXAMPLES:

URL:

http://www.ac.upc.edu/etsetb/pam?name=http#get



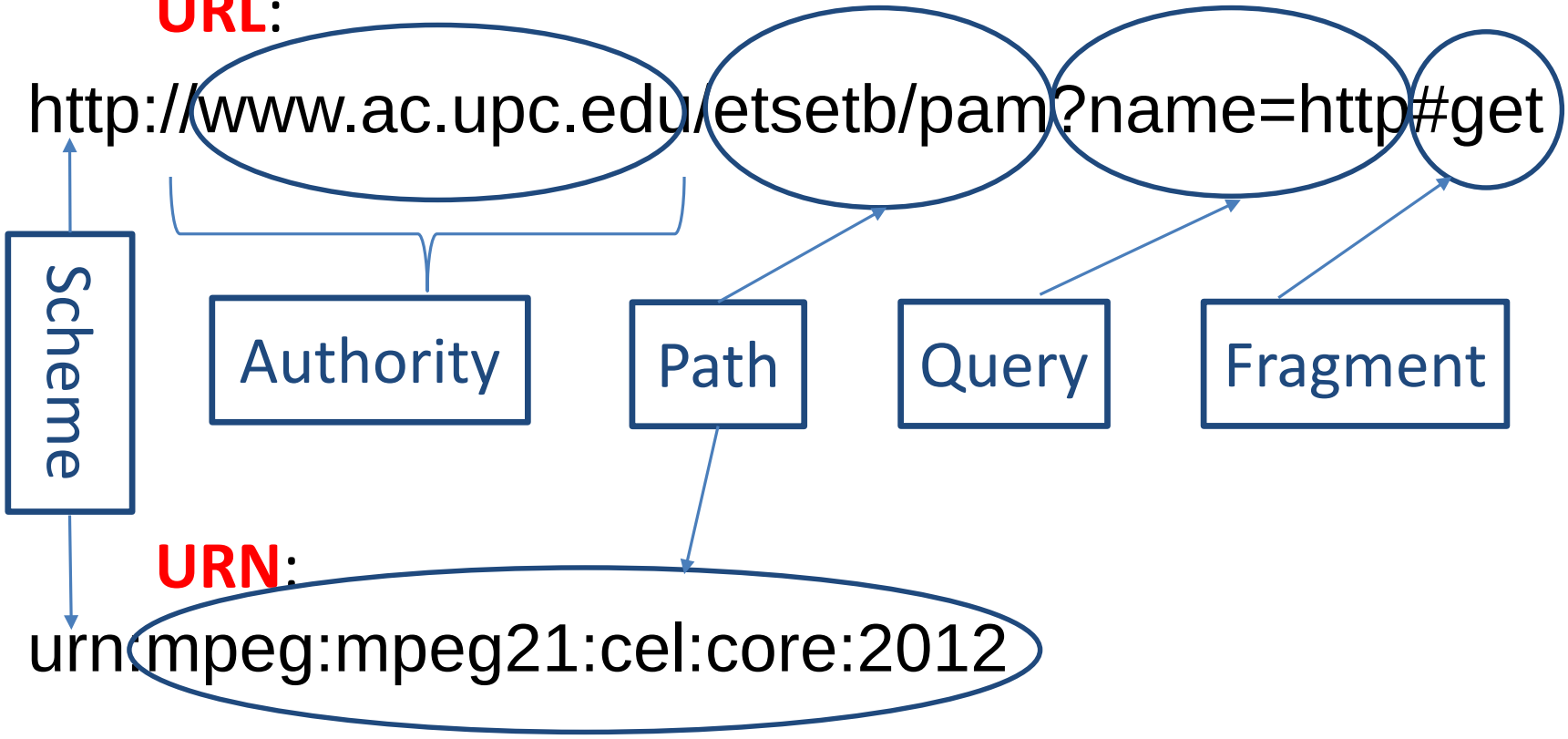
URN:

urn:mpeg:mpeg21:cel:core:2012

Links - URI (Uniform Resource Identifier)

- EXAMPLES:

URL:



Contents

- DNS (Domain Name System).
- e-mail: Protocols and formats.
- Web elements & HTTP.
  - **HTTP.**
- HTML.
- XML.
- Characters in communications.

HTTP introduction

- HyperText Transfer Protocol.
- RFC 2616 (HTTP/1.1, 1999).
  First version (0.9) in 1991. Now 2.0 already available.
  HTTP/2 **RFC 7540**, May 2015. > 17 versions since 2012
- Stateless. Request/Response.
- Normally over TCP (Port 80 as default).

HTTP concepts

- **Proxy** server: Intermediary.
- **Caching**.
- **Persistency**: Not-closing / closing TCP connection.
- **Pipelining**: Requesting new objects before complete downloading of previous ones.

HTTP methods ("No modification")

- **GET**. Requests the specified resource.
  Should only retrieve data. No other effect.
- HEAD. Response identical to GET without the body.
- TRACE. Echoes back the received request.
- OPTIONS. Returns the HTTP methods that the server supports for the specified URL.

HTTP methods ("modification")

- **POST**. Submits data to be processed → update, creation. *Examples*: HTML form, annotation, message, item to add to a database, …
- PUT. Uploads the specified resource.
- DELETE. Deletes the specified resource.
- PATCH. Applies partial modifications to the resource.
- …

HTTP Request format

REQUEST LINE:

```
GET /index.html HTTP/1.1
```

HEADER LINES:

```
Host: www.example.com
```

BLANK LINE

BODY: ***Empty*** *for GET Request*

HTTP Response format

STATUS LINE:

`HTTP/1.1 200 OK`

HEADER LINES:

…

BLANK LINE

BODY: ***HTML document****, for example, for GET Response*

HTTP status codes

- 1xx Informational
- 2xx Success
  - 200 OK
- 3xx Redirection
  - 301 Moved Permanently
- 4xx Client Error
  - 401 Unauthorized
  - 403 Forbidden
  - 404 Not Found
- 5xx Server Error
  - 500 Internal Server Error
  - 503 Service Unavailable

HTTP GET Request example

```
GET /search?q=myBook HTTP/1.1
Host: www.google.com
User-Agent: Mozilla/5.0 ...
Accept:
text/xml,application/xml,text/html,
text/plain,image/png, …
Accept-Language: da,en-us, …
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8 …
Keep-Alive: 300
Connection: keep-alive
Referer: http://www.google.com/
```

MIME types

Time out (in s.)

Persistency

HTTP GET Response example

```
HTTP/1.1 200 OK
Date: Fri, 17 Sep 2009 07:59:01 GMT
Server: Apache/2.0.50 (Unix) …
Last-Modified: Tue, 24 Feb 2009
08:32:26 GMT
ETag: "ec002-afa-fd67ba80"
Accept-Ranges: bytes
Content-Length: 2810
Content-Type: text/html

… body content …
```

Entity Tag

More on HTTP GET header lines

REQUEST:

Conditional

```
If-Modified-Since: May 1, 2013 8:00 PM

Range: bytes = 387–
```

RESPONSE:

```
Connection: close
```

Persistency

More on HTTP GET functionality

GET Response:

```
HTTP/1.1 200 OK
…
Etag:"…"
…
```

Server assigned

GET Request:

```
…
If-None_Match: "Etag"
…
```

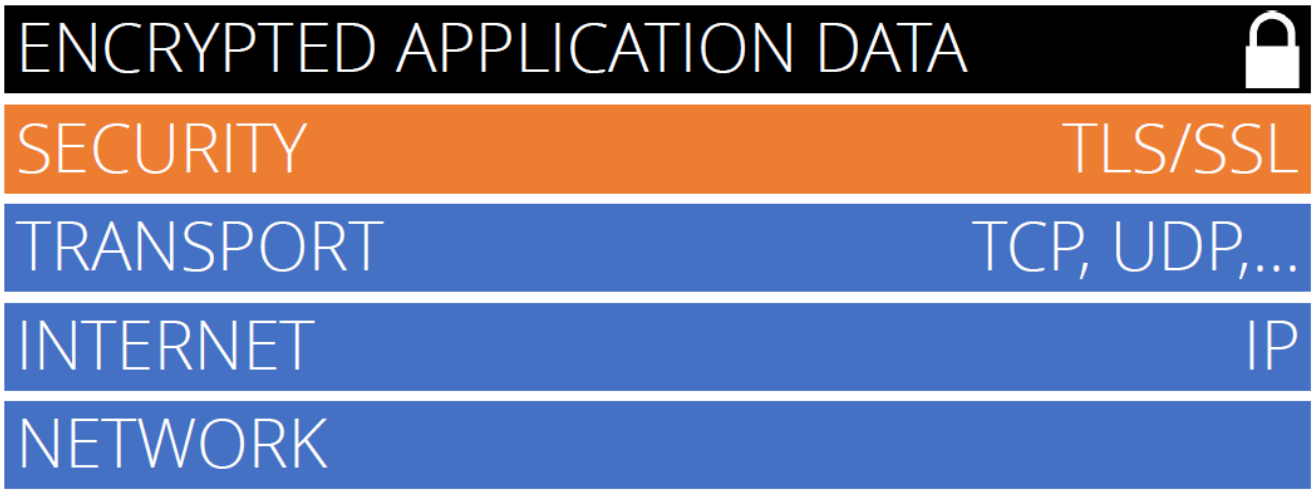
GET Response:

```
HTTP/1.1 304 Not Modified
…
```

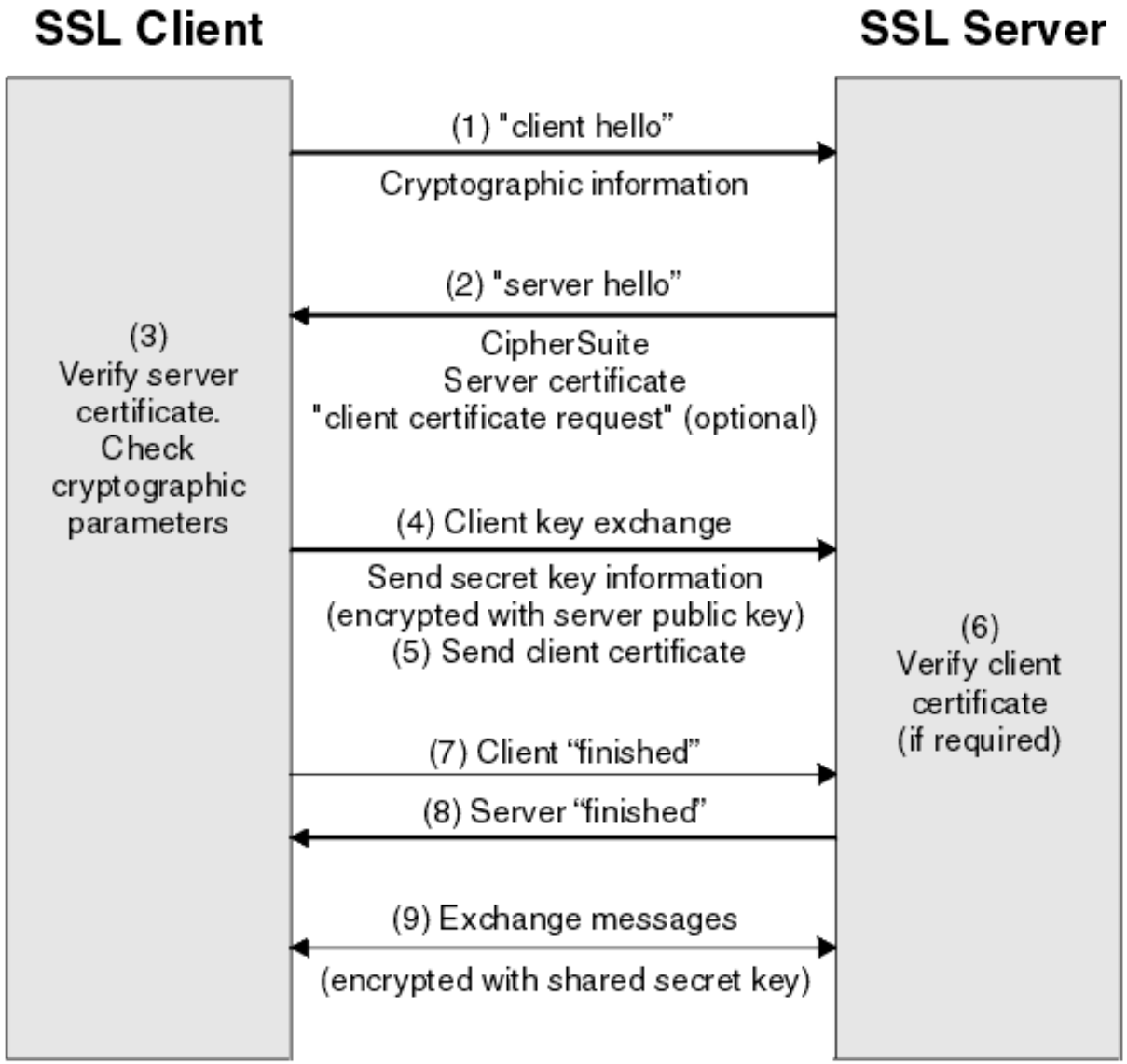
Secure HTTP

- Securing the communications channel.
- **HTTPS**: Secure transport (TCP) connection

  → Transport Layer Security (TLS) / Secure Sockets Layer (SSL)



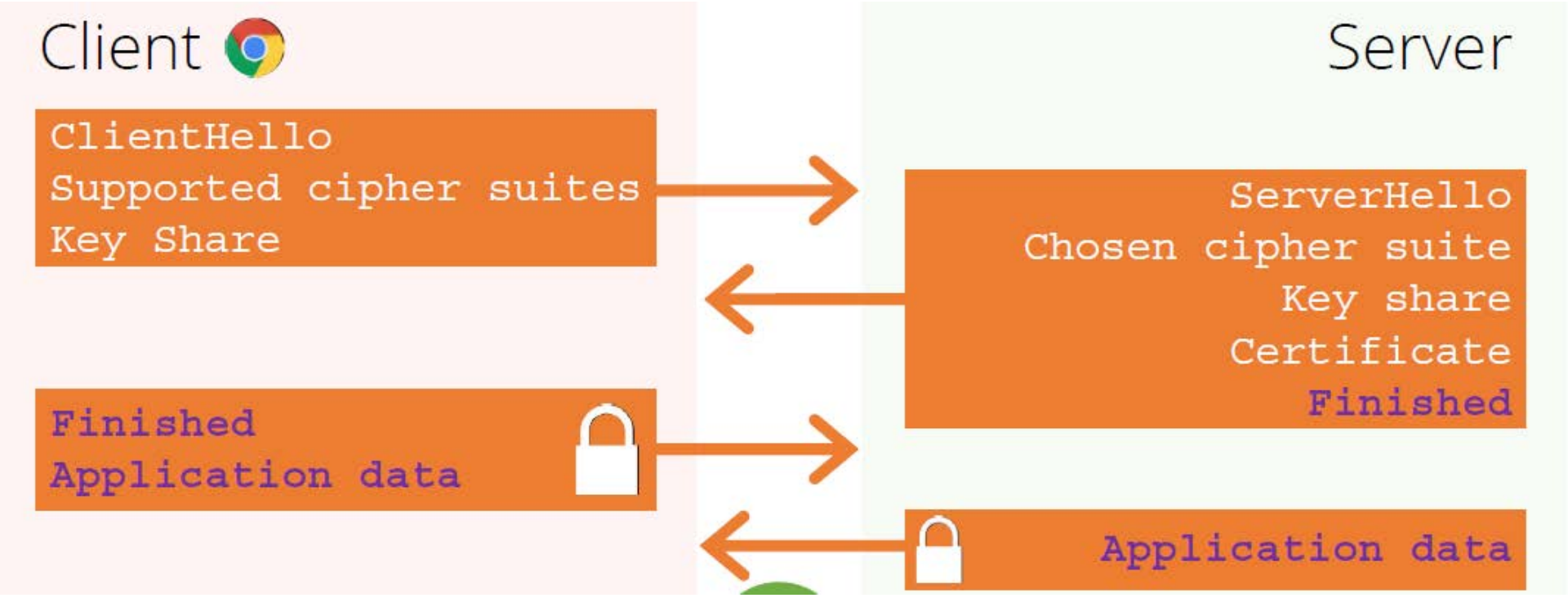
HTTPS: TLS/SSL handshake (old)



TLS (Transport Layer Security) Protocol

- Versions: Current TLSv1.2; New **TLSv1.3** (2017)
- *Handshake protocol phase*:
  - Authentication one or both sides (usually the server)
  - Negotiation: “cipher suites” (only Eliptic Curve *Diffie-Hellman* key exchange algorithms)
- *Record protocol phase*:
  - Carries and encapsulates data.
  - Adds a MAC *(Message Authentication Code)*, encrypts *application protocol data* and adds a TLS header (5 bytes).

HTTPS: TLSv1.3 handshake & record



Contents

- DNS (Domain Name System).
- e-mail: Protocols and formats.
- Web elements & HTTP.
- HTML.
- XML.
- Characters in communications.

HTML
(HyperText Markup Language)

- Language to express the WWW documents
  - "Markup" language
    (as SGML, *Standard Generalized Markup Language*)
  - World Wide Web Consortium (W3C) http://www.w3.org

- Characteristics:
  - Based on tags: <tag> … </tag>
  - **Logic** structure coding …
  - but also **presentation**!
  - *Logical structure vs. Layout/Physical structure*
  - Links to other objects
    (value or "inline" / URL reference)

Basic concepts

- Tags:
  - Separate text/data fragments
  - Provide separated text with “semantics”
  - In general, there is a *start* and an *end* (exceptions exist)

  - Start of area delimited by a tag: `<tag_name>`
  - End of area delimited by a tag: `</tag_name>`

  - Example: `<tag_name>` delimited text `</tag_name>`

- Attributes:
  - Complete the semantics of a tag

  - Form: `<tag attrib1=“value” attrib2=“value”>` text `</tag>`

HTML: structure

Tags for structuring documents:

- Start and End of an HTML document:
  - <html> and </html>
- Header:
  - <head> and </head>
- Body:
  - <body> and </body>

Example of a document basic structure:

```
<html>
   <head>
      <title>Basic document</title>
   </head>
   <body>
      A very simple document
   </body>
</html>
```

HTML: example

```
<html>
<head>
<title>Things To Do</title>
</head>
<body>
<ol>
<li>Feed the cat.
<li>Try out the shell command:
<pre>foreach x ( `ls` )
        cat $x | tr "aeiouy" "x" > $x
    end</pre>
<li>Buy ticket for Timbuktu.
</ol>
</body>
</html>
```

1. Feed the cat.
2. Try out the shell command:

```
foreach x ( `ls` )
  cat $x | tr "aeiouy" "x" > $x
end
```

3. Buy ticket for Timbuktu.

HTML (v.4): tags

<!--
<
<A>
<ABBREV>
<ACRONYM>
<ADDRESS>
<APPLET>
<AREA>
<AU>
<AUTHOR>
<B>
<BANNER>
<BASE>
<BASEFONT>
<BGSOUND>
<BIG>
<BLINK>
<BLOCKQUOTE>
<BQ>
<BODY>

<CAPTION>
<CENTER>
<CITE>
<CODE>
<COL>
<COLGROUP>
<CREDIT>
<DEL>
<DFN>
<DIR>
<DIV>
<DL>
<DT>
<DD>
<EM>
<EMBED>
<FIG>
<FN>
<FONT>
<FORM>
<FRAME>
<FRAMESET>
<H1>
<H2>
<H3>
<H4>
<H5>
<H6>
<HEAD>
<HR>
<HTML>
<I>
<IFRAME>
<IMG>
<INPUT>
<INS>
<ISINDEX>
<KBD>
<LANG>
<LH>
<LI>
<LINK>
<LISTING>
<MAP>
<MARQUEE>
<MATH>
<MENU>
<META>
<MULTICOL>
<NOBR>
<NOFRAMES>
<NOTE>
<OL>
<OVERLAY>
<P>
<PARAM>
<PERSON>
<PLAINTEXT>
<PRE>
<Q>
<RANGE>
<SAMP>
<SCRIPT>
<SELECT>
<SMALL>
<SPACER>
<SPOT>
<STRIKE>
<STRONG>
<SUB>
<SUP>
<TAB>
<TABLE>
<TBODY>
<TD>
<TEXTAREA>
<TEXTFLOW>
<TFOOT>
<TH>
<THEAD>
<TITLE>
<TR>
<TT>
<U>
<UL>
<VAR>
<WBR>
<XMP>

HTML (v.4): tags

<!--
<
<A> link
<ABBREV>
<ACRONYM>
<ADDRESS>
<APPLET>
<AREA>
<AU>
<AUTHOR>
<B>
<BANNER>
<BASE>
<BASEFONT>
<BGSOUND>
<BIG>
<BLINK>
<BLOCKQUOTE>
<BQ>
<BODY>

<CAPTION>
<CENTER>
<CITE>
<CODE>
<COL>
<COLGROUP>
<CREDIT>
<DEL>
<DFN>
<DIR>
<DIV>
<DL>
<DT>
<DD>
<EM>
<EMBED>
<FIG>
<FN>
<FONT>
<FORM>
<FRAME />
<FRAMESET>
<H1> header 1
<H2>
<H3>
<H4>
<H5>
<H6>
<HEAD>
<HR>
<HTML>
<I>
<IFRAME>
<IMG> image
<INPUT />
<INS>
<ISINDEX>
<KBD>
<LANG>
<LH>
<LI> element of a list
<LINK>
<LISTING>
<MAP>
<MARQUEE>
<MATH>
<MENU>
<META />
<MULTICOL>
<NOBR>
<NOFRAMES>
<NOTE>
<OL> ordered list
<OVERLAY>
<P>
<PARAM>
<PERSON>
<PLAINTEXT>
<PRE>
<Q>
<RANGE>
<SAMP>
<SCRIPT>
<SELECT>
<SMALL>
<SPACER>
<SPOT>
<STRIKE>
<STRONG>
<SUB>
<SUP>
<TAB>
<TABLE>
<TBODY>
<TD>
<TEXTAREA>
<TEXTFLOW>
<TFOOT>
<TH>
<THEAD>
<TITLE>
<TR>
<TT>
<U>
<UL> non-ordered list
<VAR>
<WBR>
<XMP>

HTML: client scripting

- Document processing by client (fields validation, value calculations, …)
- Javascript:
  - Client scripting language based in Java
  - Example:
    - “a” tag & “href” attribute:

```
<a href=“page.html”> Link text </a>
```

    - Script - by clicking a link, a warning is issued:

```
<script>
   <a href=“page.html” onClick=“alert(‘You clicked’);"> Link text </a>
</script>
```

HTML: client scripting

- Javascript:
  - Example:
    - By clicking in a button, a warning is issued:

```
<html>
  <head>
     <script type="text/javascript">
          function displaymessage() { alert("Hello World!");}
     </script>
  </head>
  <body>
     <form>
        <input type="button“ value="Click me!" onclick="displaymessage()" />
     </form>
  </body>
</html>
```

HTML: CSS

- HTML should not be used to provide style to the data it contains.
- It should use “style sheets”:
  - **Cascade Style Sheets (CSS)**
- Style sheets are created to control the layout of an HTML document:
  - Complement the structural information of HTML
  - Separate structure (logical) and style (layout)
- Why “cascade”?
  - Style information is appearing / overlapping
  - It is falling (cascading) and being applied over the document

HTML: CSS

- Example of rule (presentation of a tag)

- What to control?
  - Fonts (color, size, caps, font type, etc.)
  - Background (image, color, tiling properties)
  - Text (spacing, line-height, alignment, decoration, word-spacing)
  - Box properties (margin, border, float)
  - List properties (image for bullets)
  - Links (visited, hover, active, link)

HTML: CSS

- **CSS Syntax**

```
h1 { color:red; font-size:20px; }
```

- Example
  - Content of the file "mystyle.css":

```
h1 {color:red; font-size:20px;}
p {margin-left:20px; color:blue; font-size:18px;}
```

Presentation info

```
<html>
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css" />
</head>
<body>
  <h1>First Heading</h1>
  <p>first paragraph.</p>
</body>
</html>
```

Logical structure

First Heading

first paragraph.

HTML5

- New version of HTML:
  World Wide Web Consortium (W3C) +
  Web Hypertext Application Technology Working Group (WHATWG)
  “This document covers the W3C HTML5 specification, W3C HTML5.1 specification, and the WHATWG HTML standard. For readability, these are referred to as if they were a single specification: "the HTML specification" or simply "HTML" when something applies equally to all of them; otherwise, they are called out explicitly.”
- W3C Rec. Oct. 2014, but still new developments!
- Browsers start to support some of its new features
- Basic ideas:
  - Based on HTML, CSS, DOM, Javascript
  - Reduce the need of external plug-ins
  - Improve error management
  - More markup to reduce scripting
  - Device independent

HTML5 new elements

- `<canvas>` to draw 2D elements
- Multimedia: `<video>`, `<audio>`, `<source>`, `<embed>`, `<track>`
- Content specific element:
  `<figure>`, `<footer>`, `<header>`, `<nav>`, `<section>`, …
- Other elements:
  `<output>`, new values for `<input>` (`date`, `email`, `url`, `search`), `<time>`, …
- Obsoleted elements:
  `<big>`, `<center>`, `<font>`, `<frame>`, ...
- Support for inline MathML and SVG.

HTML5

http://w3c.github.io/html-reference/elements.html

HTML5.x

https://www.w3.org/TR/html52/

Version 5.3 going on!
https://w3c.github.io/html/

Contents

- DNS (Domain Name System).
- e-mail: Protocols and formats.
- Web elements & HTTP.
- HTML.
- XML.
- Characters in communications.

XML

- XML: eXtensible Markup Language
- Designed to transport and store data (HTML to *display* data).
- XML
  - To describe information structures → Process them automatically with applications.
  - "Users" must define their own **tags**.
    - "**Users**": "Private" users and SDO ("Standards Developing Organizations").

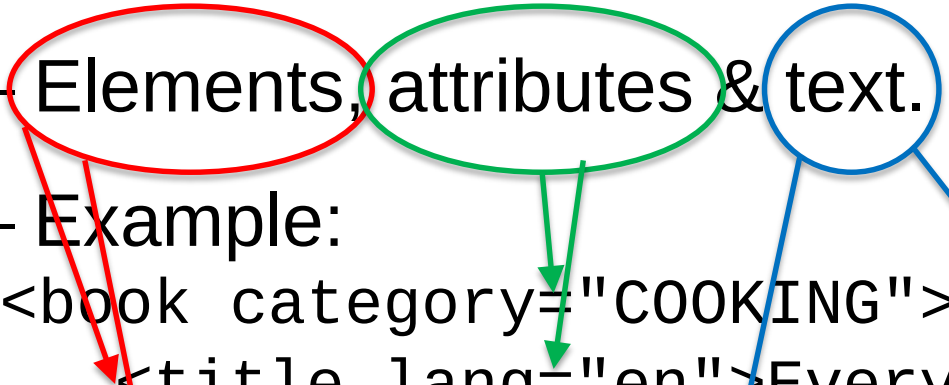
XML structure & syntax

- XML:
  - Tree structure.
  - Elements, attributes & text.
  - Example:

```
<book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    …
</book>
```

XML structure & syntax

- XML:
  - Tree structure.
  - Elements, attributes & text.
  - Example:



```
<book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    …
</book>
```

XML structure & syntax

- First line (example):
  `<?xml version="1.0" encoding="ISO-8859-1"?>`

- XML simple syntax:
  - Closing tag mandatory.
  - Tags are case sensitive.
  - Elements could be nested:

    `<a> <b>…</b> <c>…</c> </a>`

    (`a` parent, `b, c` childs , `b, c` siblings).

  - Root element needed.
  - Comments:

    `<!-- … -->`

XML other issues

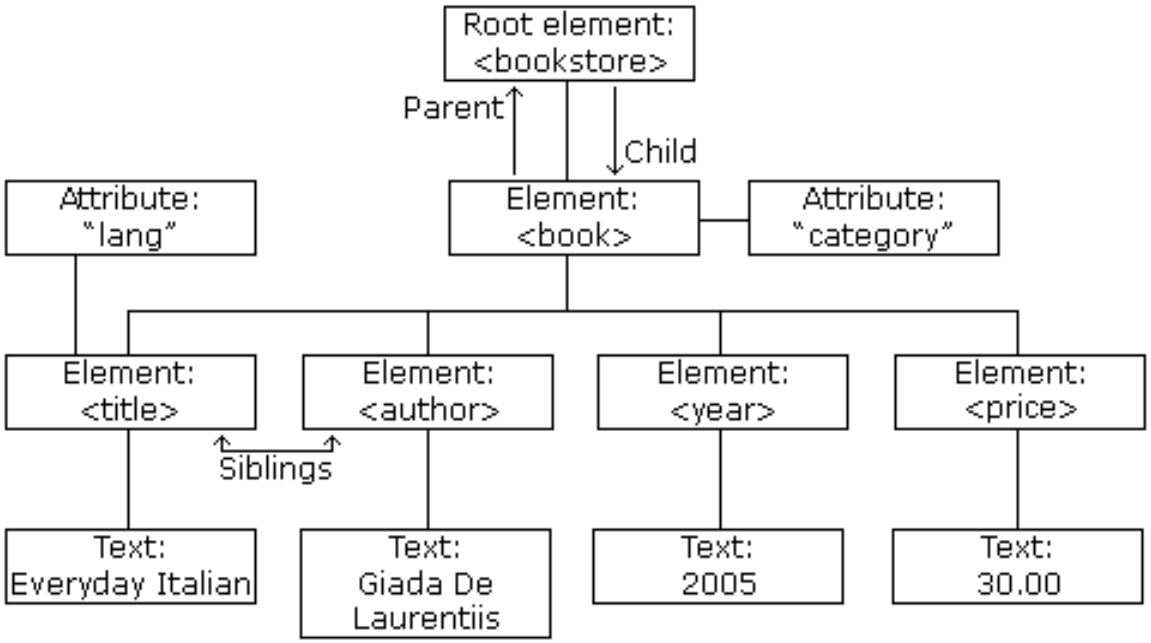
- Attributes vs. Elements: *Design decision*
- Name conflicts:
  - *Namespaces*
    - Allow differentiating element names defined by different developers/standards.
  - xmlns attribute (in the start tag of an element):

    `xmlns:prefix="URI"`

    - URLs often used as an easy way to define "unique" namespaces
- How to define tags and "structure": Schemas
  - Examples …

XML simple example

```
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="CHILDREN">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
...
</bookstore>
```



XML: Idea of Schema

- XML Schema Definition, XSD
  - Content of the file "note.xsd":

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="note">
    <xs:complexType>
      <xs:sequence>
            <xs:element name="to" type="xs:string"/>
            <xs:element name="from" type="xs:string"/>
            <xs:element name="heading" type="xs:string"/>
            <xs:element name="body" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

http://www.w3schools.com/xml/

namespace where the schema is defined, the namespace should be prefixed xs.

root element

complexType: contains other elements

sequence: child elements must appear in the same order

  - Reference to the XSD defined in "note.xsd":

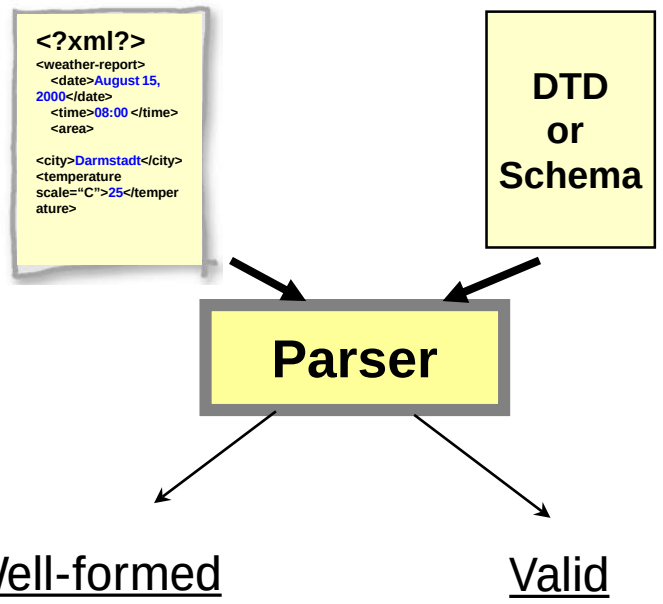
```
<?xml version="1.0"?>
<note xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="note.xsd">
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

XSD schema defined in location "note.xsd"

XML: validity

– Document XML “well-formed”:
  - A document that **satisfies syntax rules** of XML

– Document XML “valid”:
  - A well-formed document that also **conforms to a set of rules specified in a restrictions document**

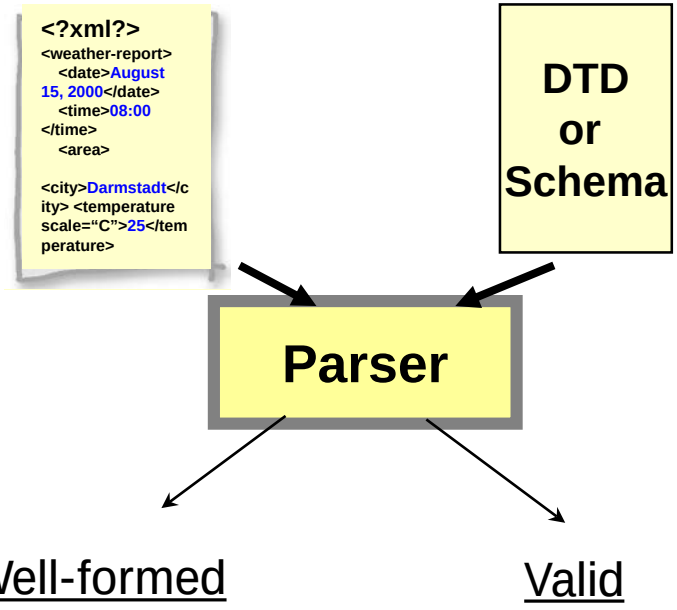


XML: validity

- Different formats for restrictions/rules documents/files:

1. DTD (Document Type Definition):
   - 1st generation: based on SGML,
   - syntax not XML, few data types, …

2. **XML Schema**:
   - XML format,
   - More data types,
   - more restrictions



Schema languages

- Define the valid structure (grammar) of a set of XML documents (a XML application)
- Initially DTD (Document Type Definition):
  - Simple but limited
- Now **XML Schema**
  - Higher expressiveness, but very complex

Contents

- DNS (Domain Name System).
- e-mail: Protocols and formats.
- Web elements & HTTP.
- HTML.
- XML.
- Characters in communications.

Characters

- Coding / representation
- Visualization: Fonts, …
- Character Sets:
  - ASCII → ISO 646.
  - ISO 2022:
    Variable width encoding (7-8 bits bytes).
    Multiple char sets. Escape chars.
  - ISO/IEC 8859 (8-bit printable chars encodings).
  - UCS (Universal Character Set)
    - ISO/IEC 10646
    - Aligned to **UNICODE**
    - **UTF-x** concept

Unicode Characters

- A **character** is a symbol that appears in a text
  - letters of the alphabet
  - pictograms (like ©)
  - accents
- Unicode characters are abstract entities:
  - `LATIN CAPITAL LETTER A`
  - `LATIN CAPITAL LETTER A WITH RING ABOVE`
  - `HIRAGANA LETTER SA`
  - `RUNIC LETTER THURISAZ THURS THORN`

Hiragana letter SA

Runic letter Thurisaz Thurs Thorn

Unicode Glyphs

- A **glyph** is a graphical presentation
- A typical example is: Å
- This may represent several characters:
  - `LATIN CAPITAL LETTER A WITH RING ABOVE`
  - `ANGSTROM SIGN`
- Or even a sequence of characters:
  - `LATIN CAPITAL LETTER A`
    `COMBINING RING ABOVE`
- Some characters even result in several glyphs

Unicode Code Points

- A **code point** is a unique number assigned to every Unicode character
- Code points are between 0 and 1,114,112
- Only around 100,000 are used today
- The character `HIRAGANA LETTER SA` is assigned the code point 12,373
- Code point 0 through 127 coincide with ASCII
- Some code point are never assigned

Unicode Character Encoding

- A **character encoding** interprets a sequence of bytes as a sequence of code points
- The bytes are first parsed into **code units**
- Code units have a fixed length
- One or more code units may be required to denote a code point
- Examples are UTF-8, UTF-16, UTF-32

UTF-8

- A code unit is a single byte
- A code point is from 1 to 4 code units
- Code units between 0 and 127 directly represent the corresponding code points
- `110XXXXX` indicates that 2 code units are used
- `1110XXXX` indicates that 3 code units are used
- `11110XXX` indicates that 4 code units are used
- The remaining code units looks like `10XXXXXX`

UTF-8 Example

- 11100011 10000001 10010101
- **111**00011 **10**000001 **10**010101
- 1100001010101
- 12,373
- HIRAGANA LETTER SA

3055 H

UTF-16

- A code unit consists of 2 bytes
- Code points below 65,536 are in a single code unit
- Higher code points are represented as:
  - `110110XXXXXXXXXX 110111XXXXXXXXXX`

  (after subtracting 65,536)
- This makes sense because Unicode assign no code points between the numbers:

  `1101100000000000 (55,296)`

  and

  `1101111111111111 (57,343)`

UTF-16 Example

“Big-endian byte order” / Character

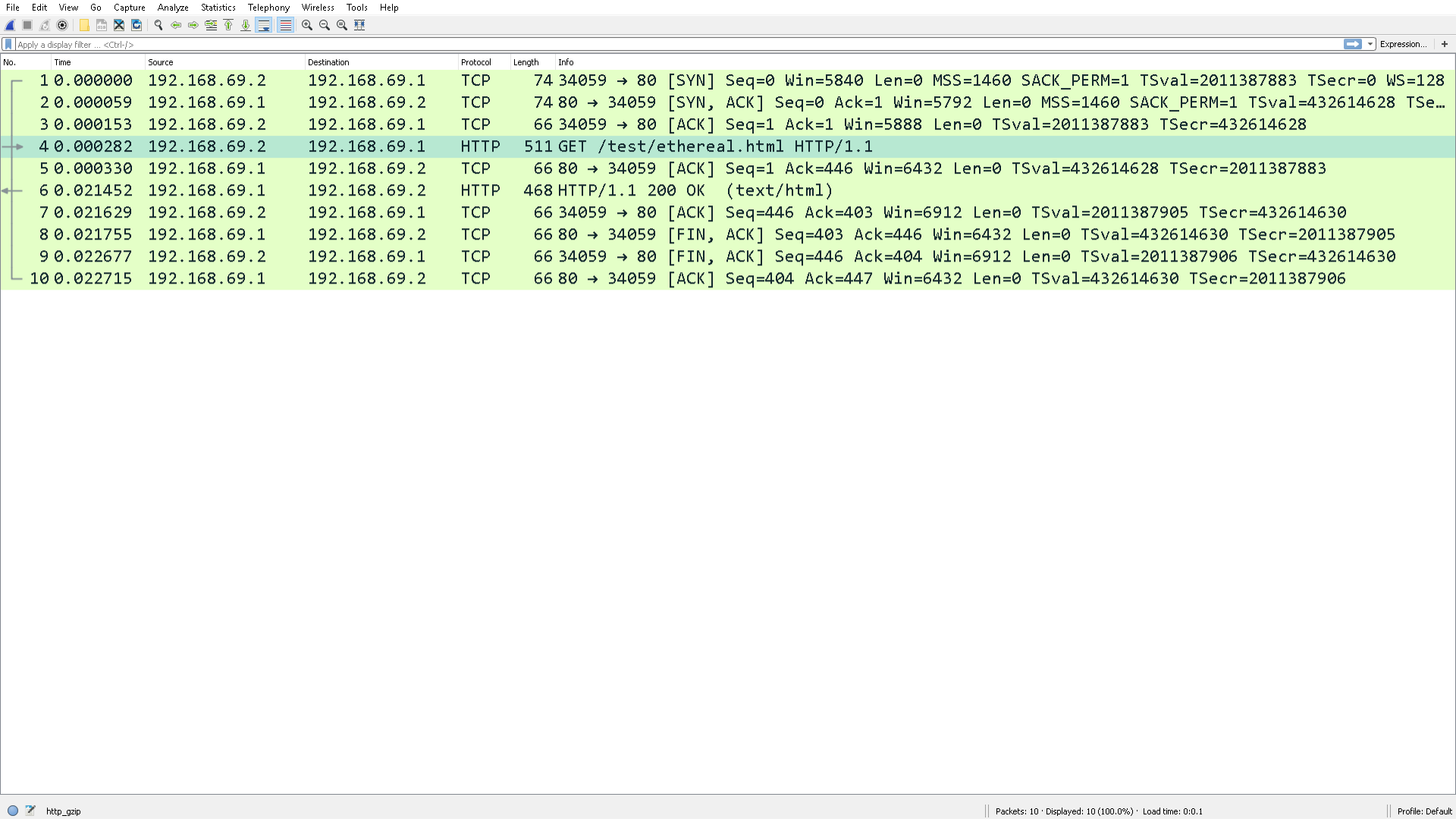
- 11111110 11111111 00110000 01010101
- 00110000 01010101
- 12,373
- HIRAGANA LETTER SA

3055 H

Contents

- DNS (Domain Name System).
- e-mail: Protocols and formats.
- Web elements & HTTP.
- HTML.
- XML.
- Characters in communications.
- WireShark screens (*re-visited*)

Web protocol

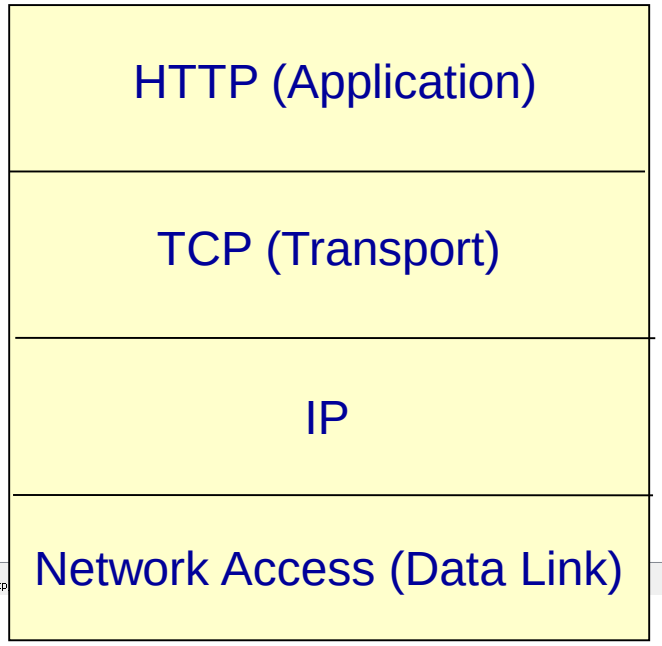


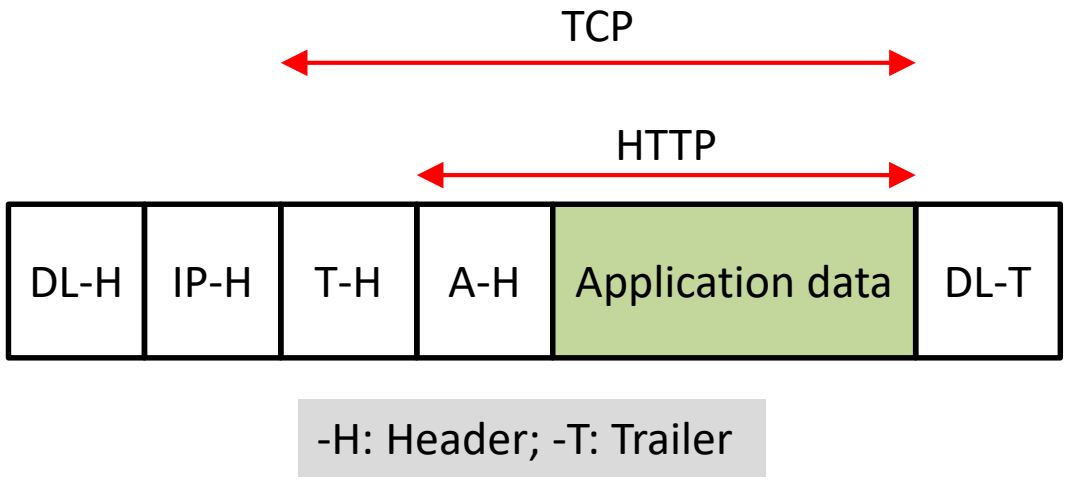
Web protocol - HTTP

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/> Expression...

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.69.2	192.168.69.1	TCP	74	34059 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=2011387883 TSecr=0 WS=128
2	0.000059	192.168.69.1	192.168.69.2	TCP	74	80 → 34059 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=432614628 TSe…
3	0.000153	192.168.69.2	192.168.69.1	TCP	66	34059 → 80 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSval=2011387883 TSecr=432614628
4	0.000282	192.168.69.2	192.168.69.1	HTTP	511	GET /test/ethereal.html HTTP/1.1
5	0.000330	192.168.69.1	192.168.69.2	TCP	66	80 → 34059 [ACK] Seq=1 Ack=446 Win=6432 Len=0 TSval=432614628 TSecr=2011387883
6	0.021452	192.168.69.1	192.168.69.2	HTTP	468	HTTP/1.1 200 OK (text/html)
7	0.021629	192.168.69.2	192.168.69.1	TCP	66	34059 → 80 [ACK] Seq=446 Ack=403 Win=6912 Len=0 TSval=2011387905 TSecr=432614630
8	0.021755	192.168.69.1	192.168.69.2	TCP	66	80 → 34059 [FIN, ACK] Seq=403 Ack=446 Win=6432 Len=0 TSval=432614630 TSecr=2011387905
9	0.022677	192.168.69.2	192.168.69.1	TCP	66	34059 → 80 [FIN, ACK] Seq=446 Ack=404 Win=6912 Len=0 TSval=2011387906 TSecr=432614630
10	0.022715	192.168.69.1	192.168.69.2	TCP	66	80 → 34059 [ACK] Seq=404 Ack=447 Win=6432 Len=0 TSval=432614630 TSecr=2011387906

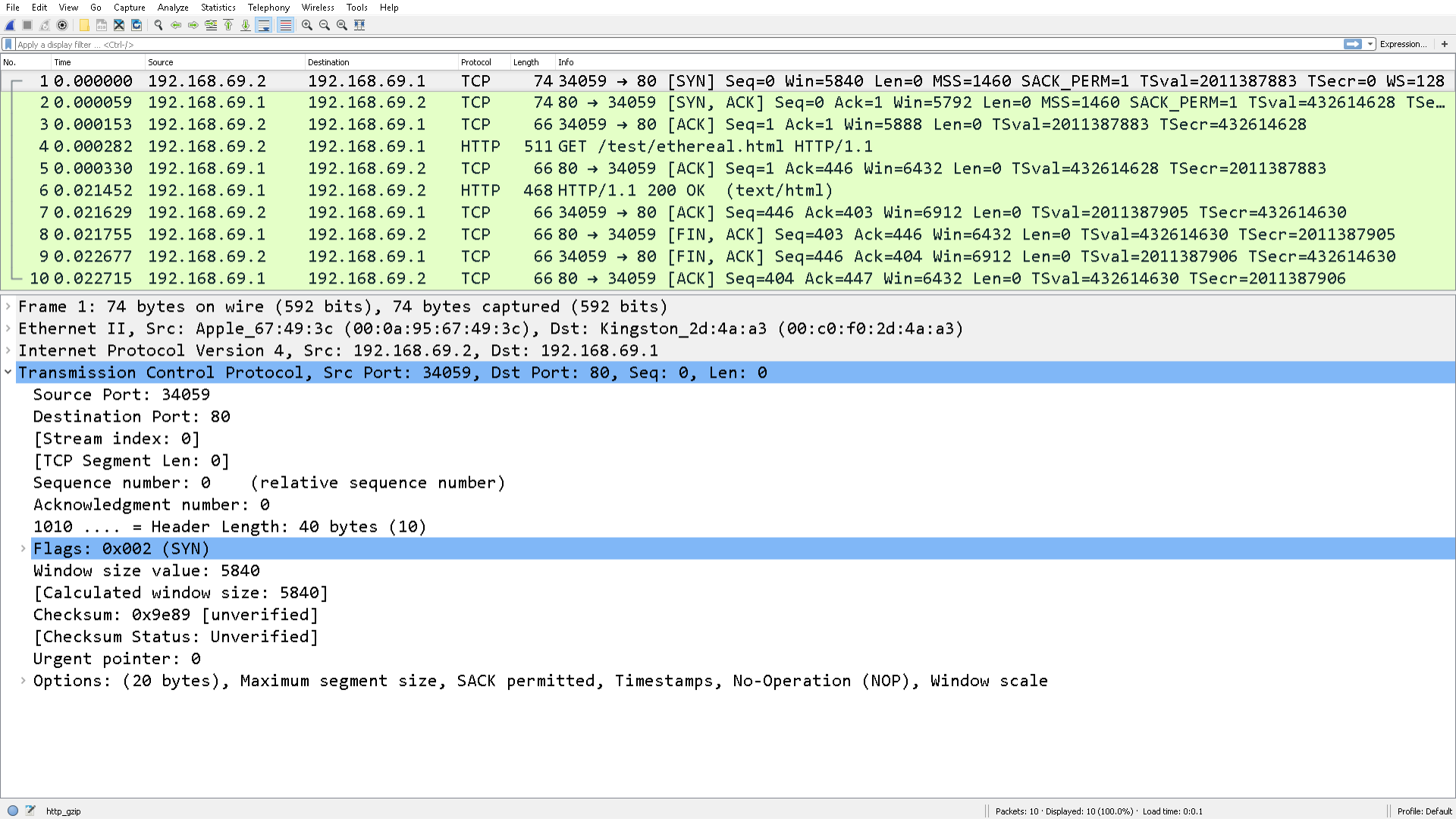




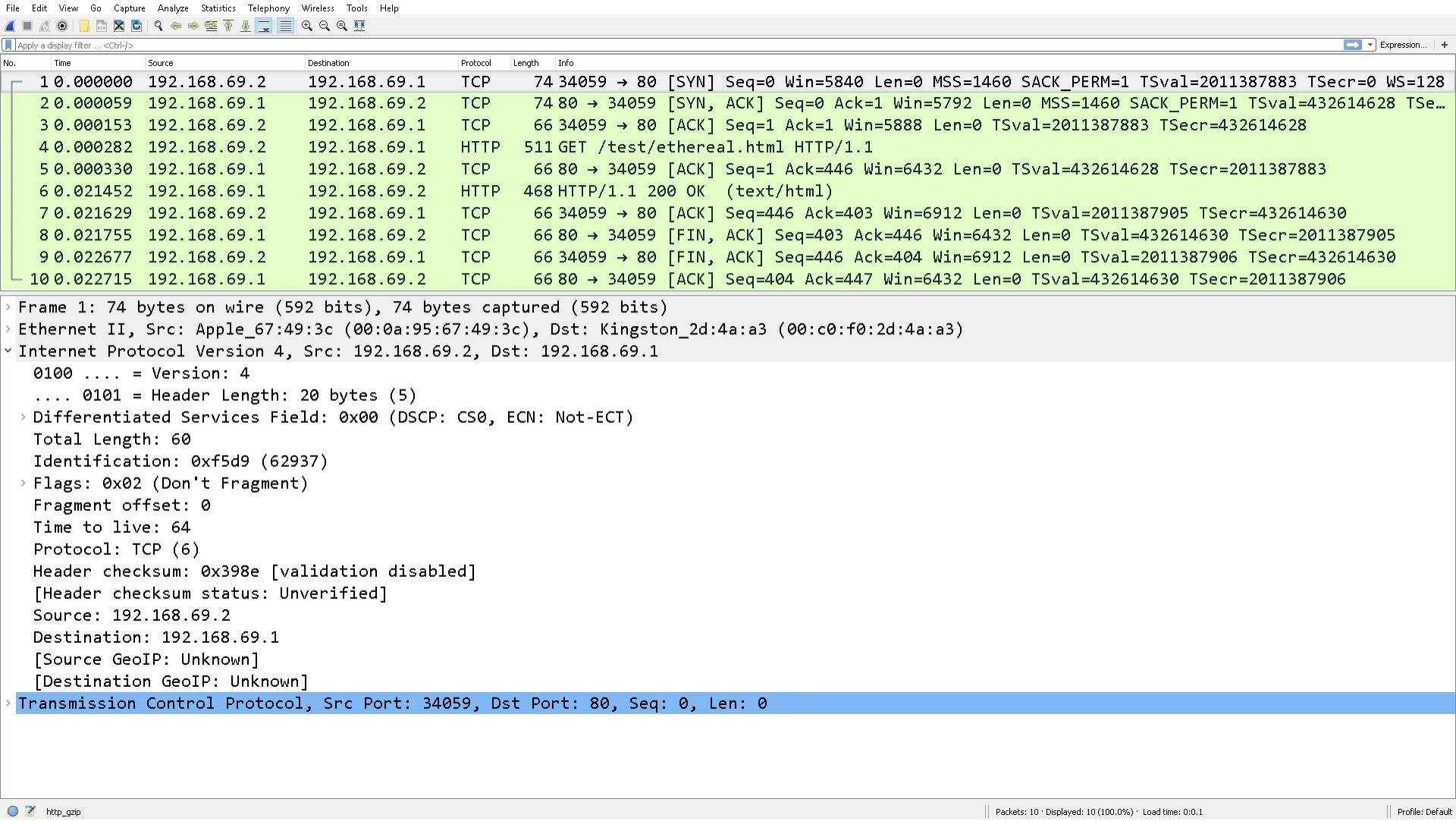
-H: Header; -T: Trailer

Packets: 10 · Displayed: 10 (100.0%) · Load time: 0:0.1 Profile: Default

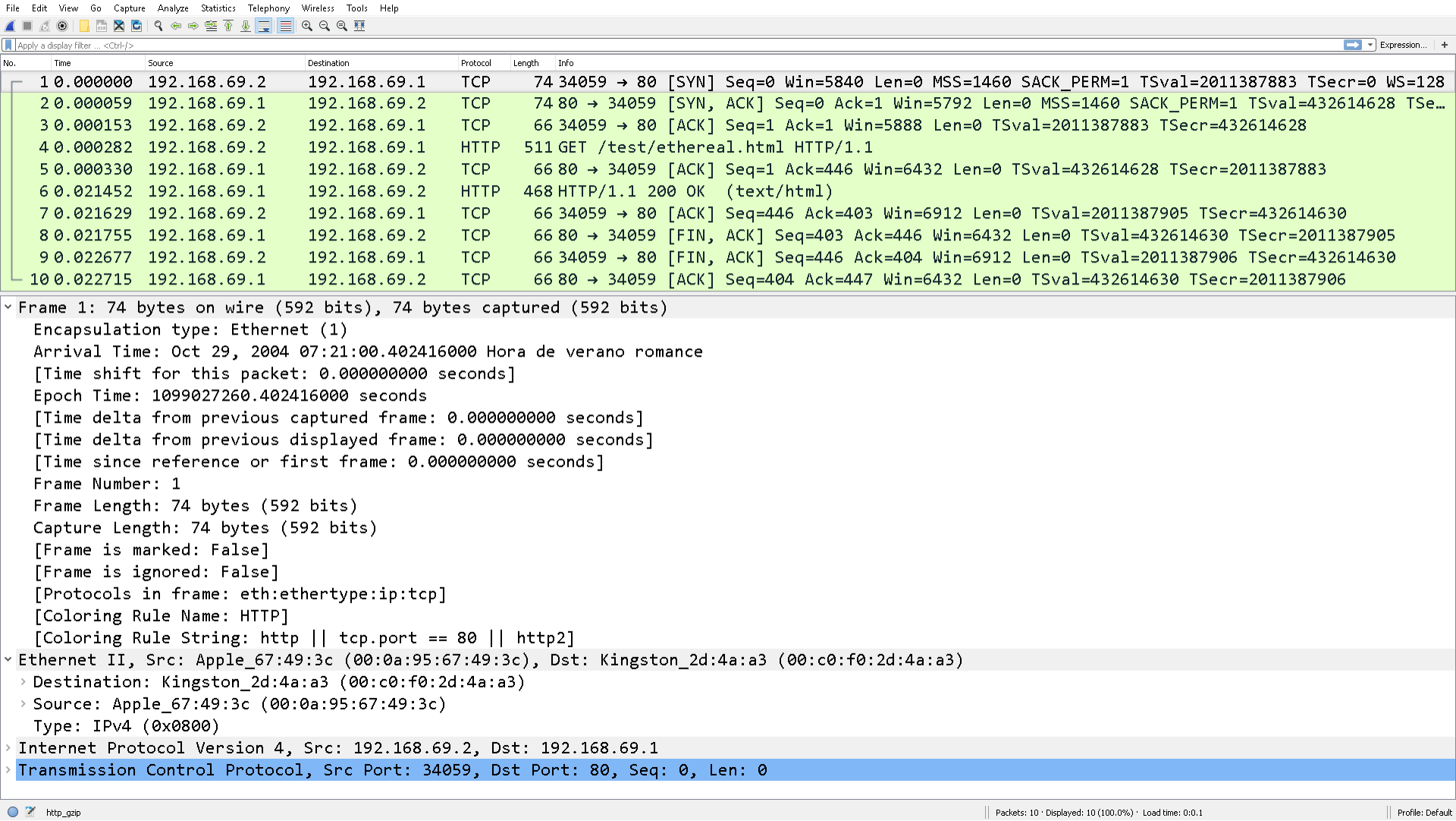
TCP connection



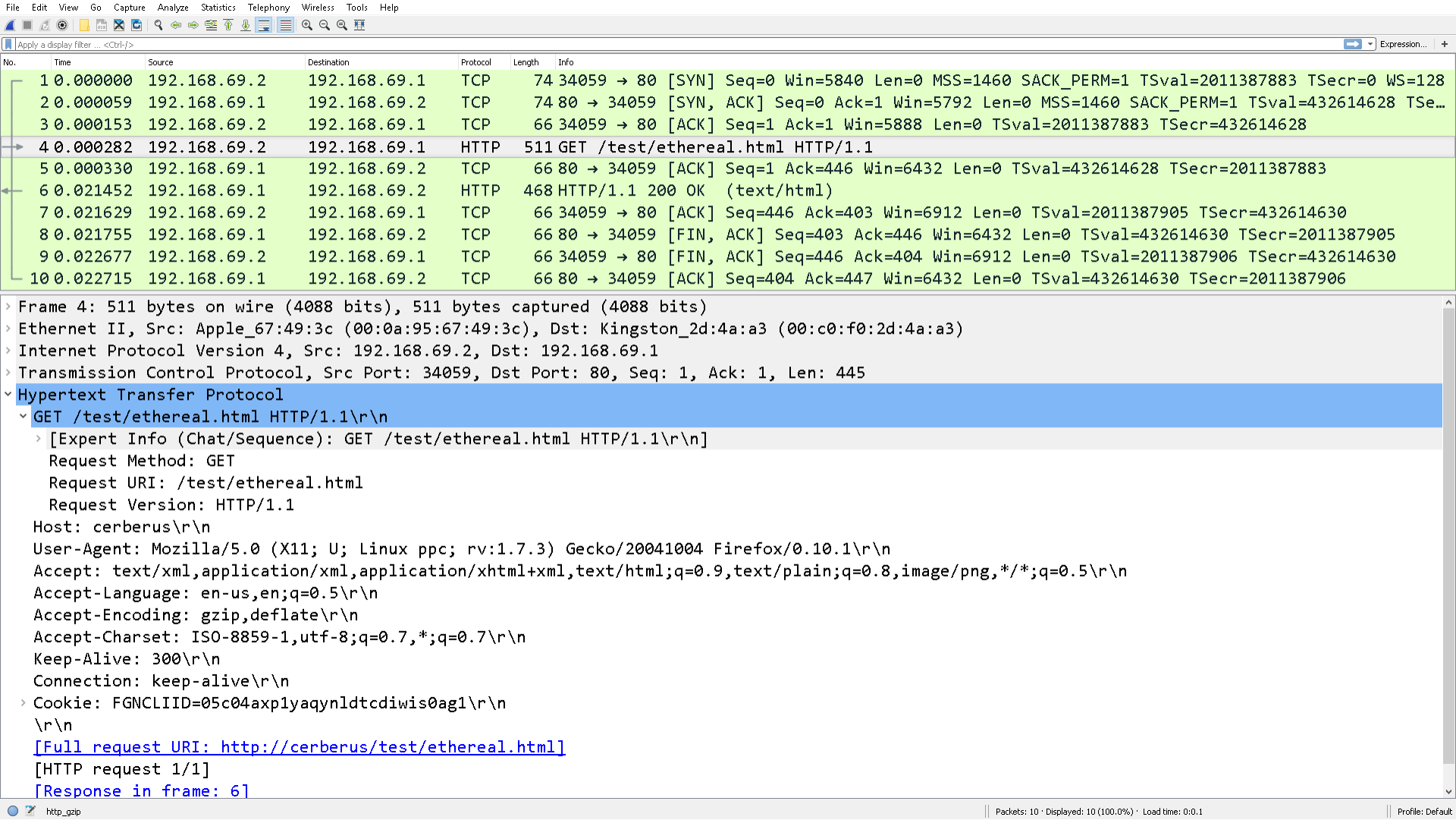
TCP connection - IP



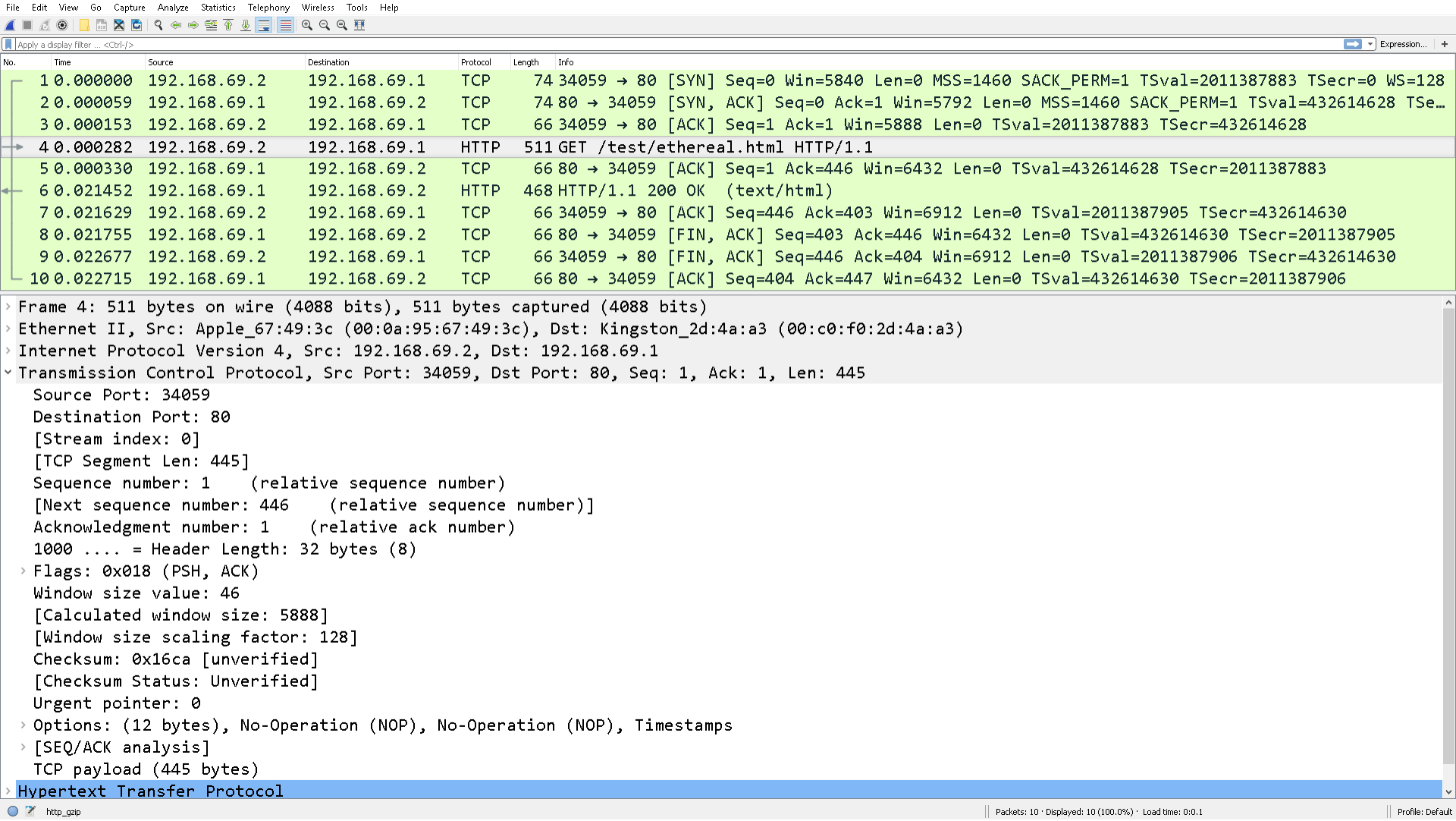
TCP connection – Data link



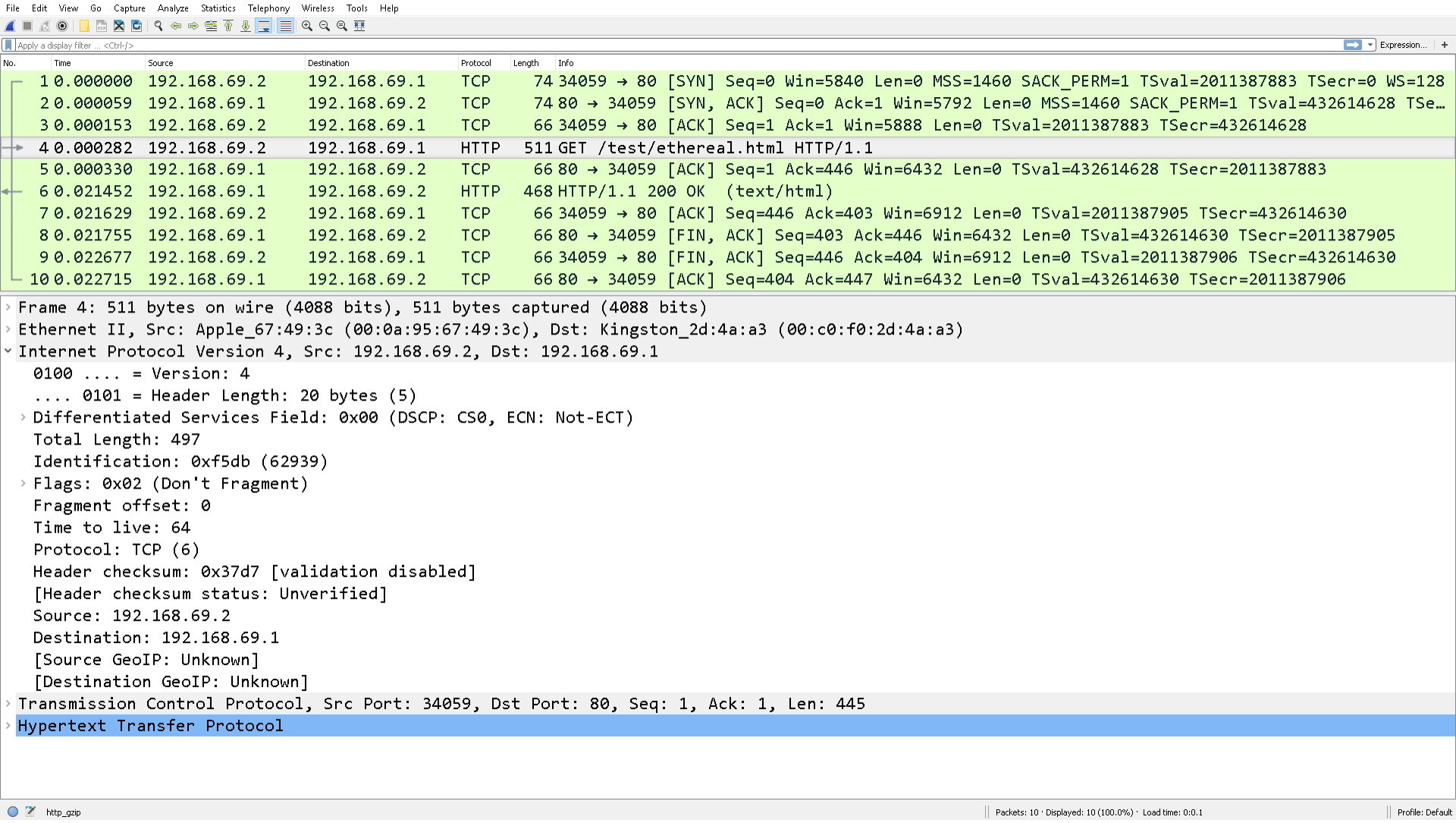
HTTP Request



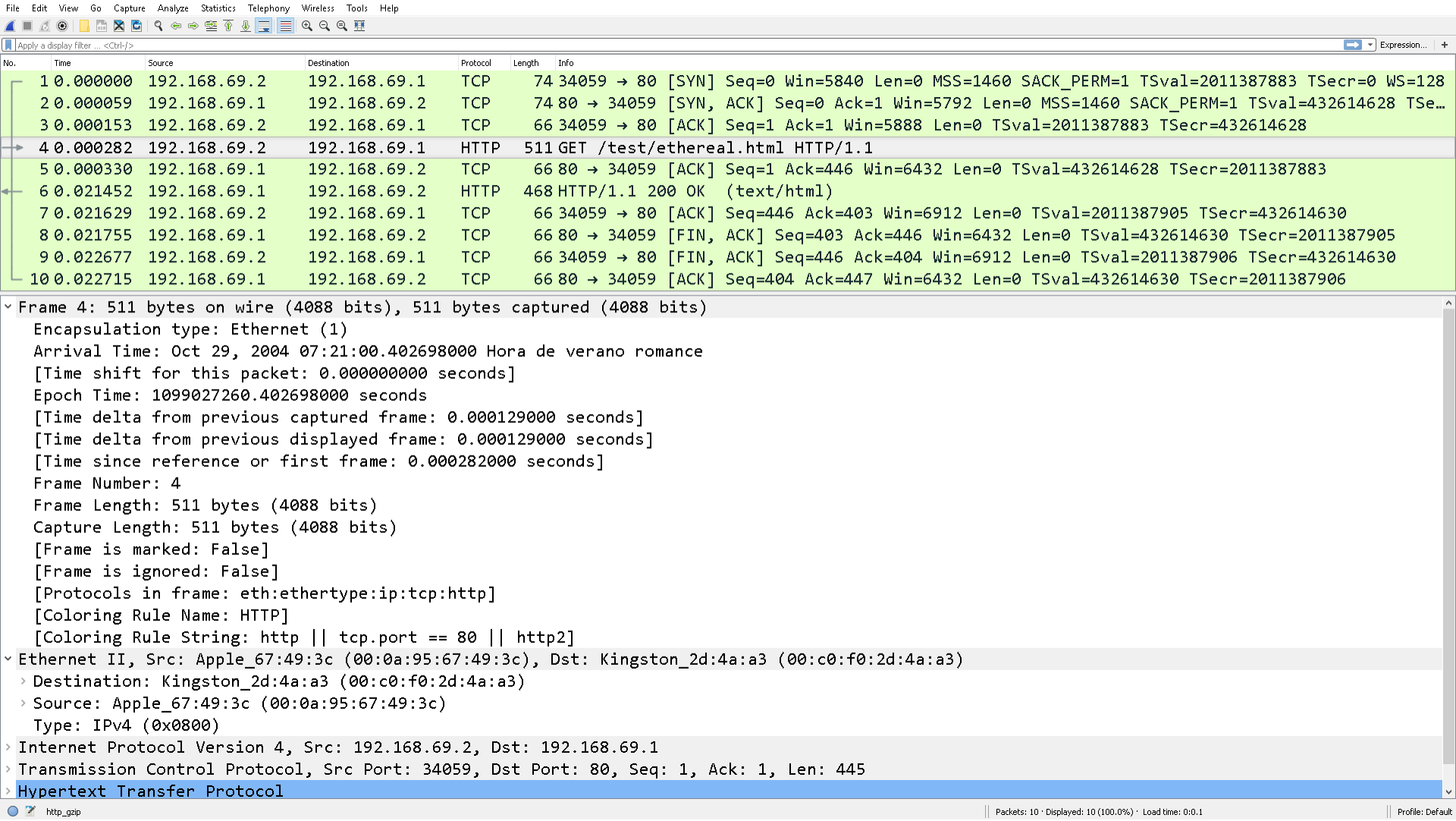
HTTP Request - TCP



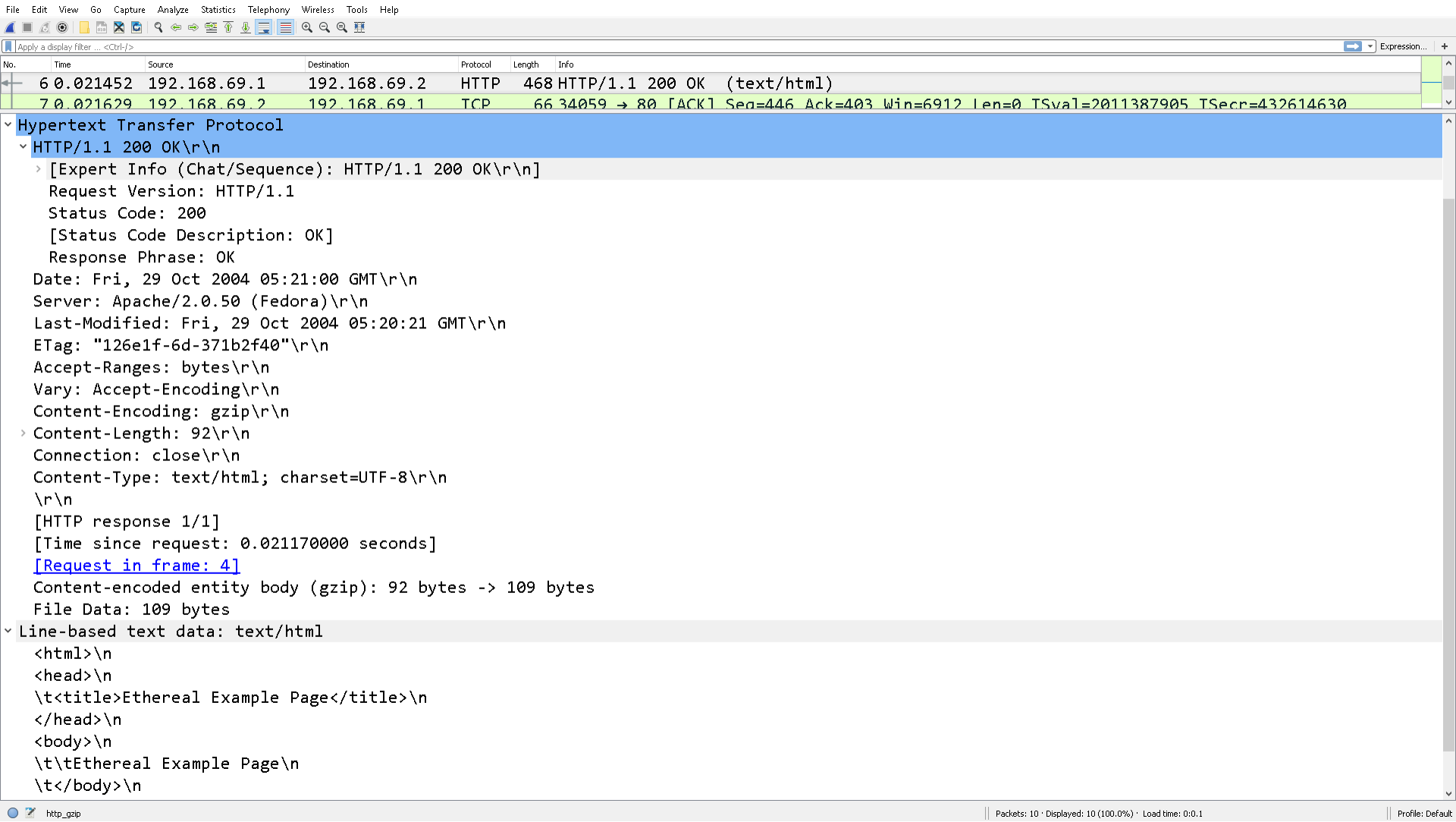
HTTP Request - IP



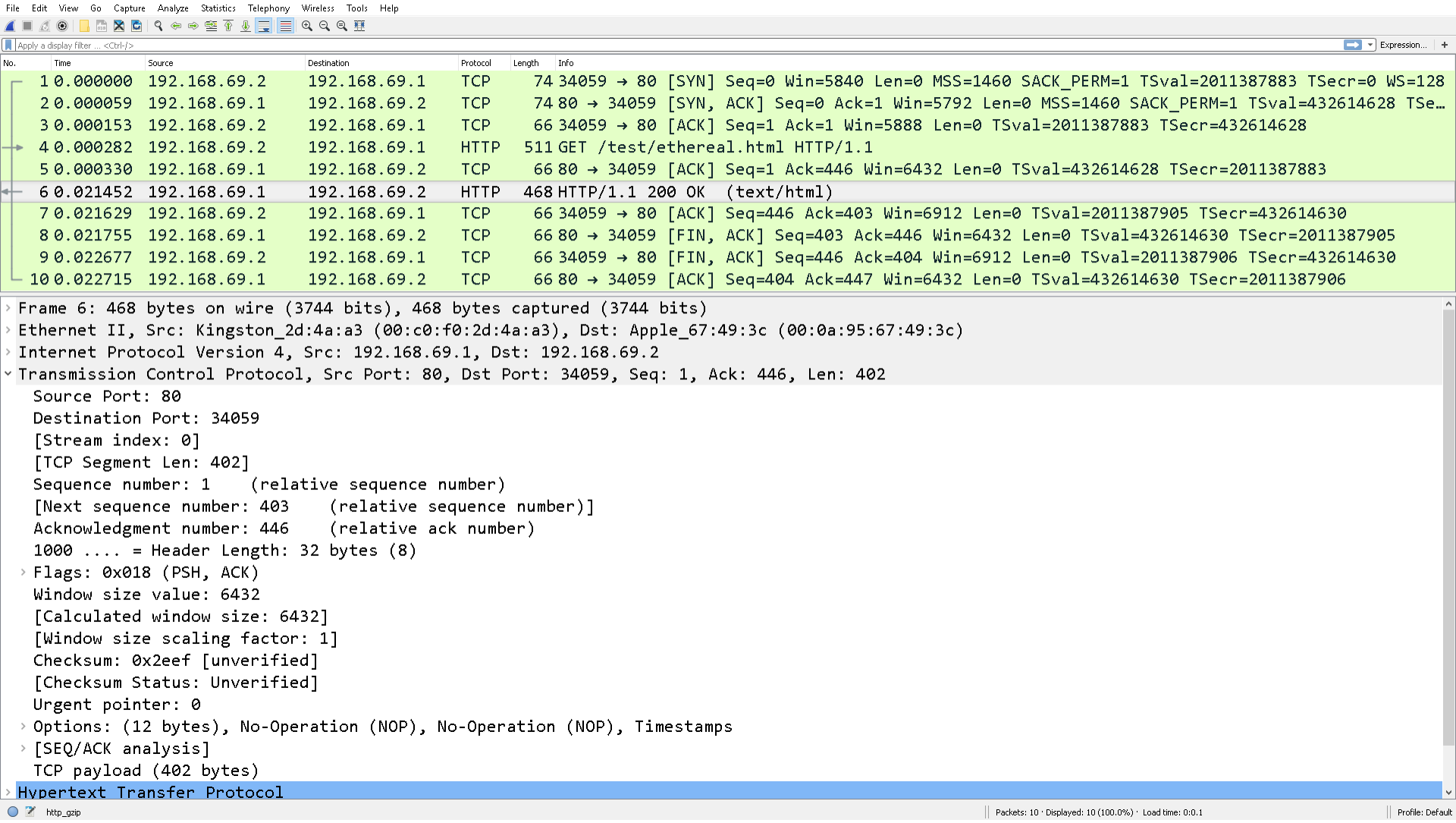
HTTP Request – Data Link



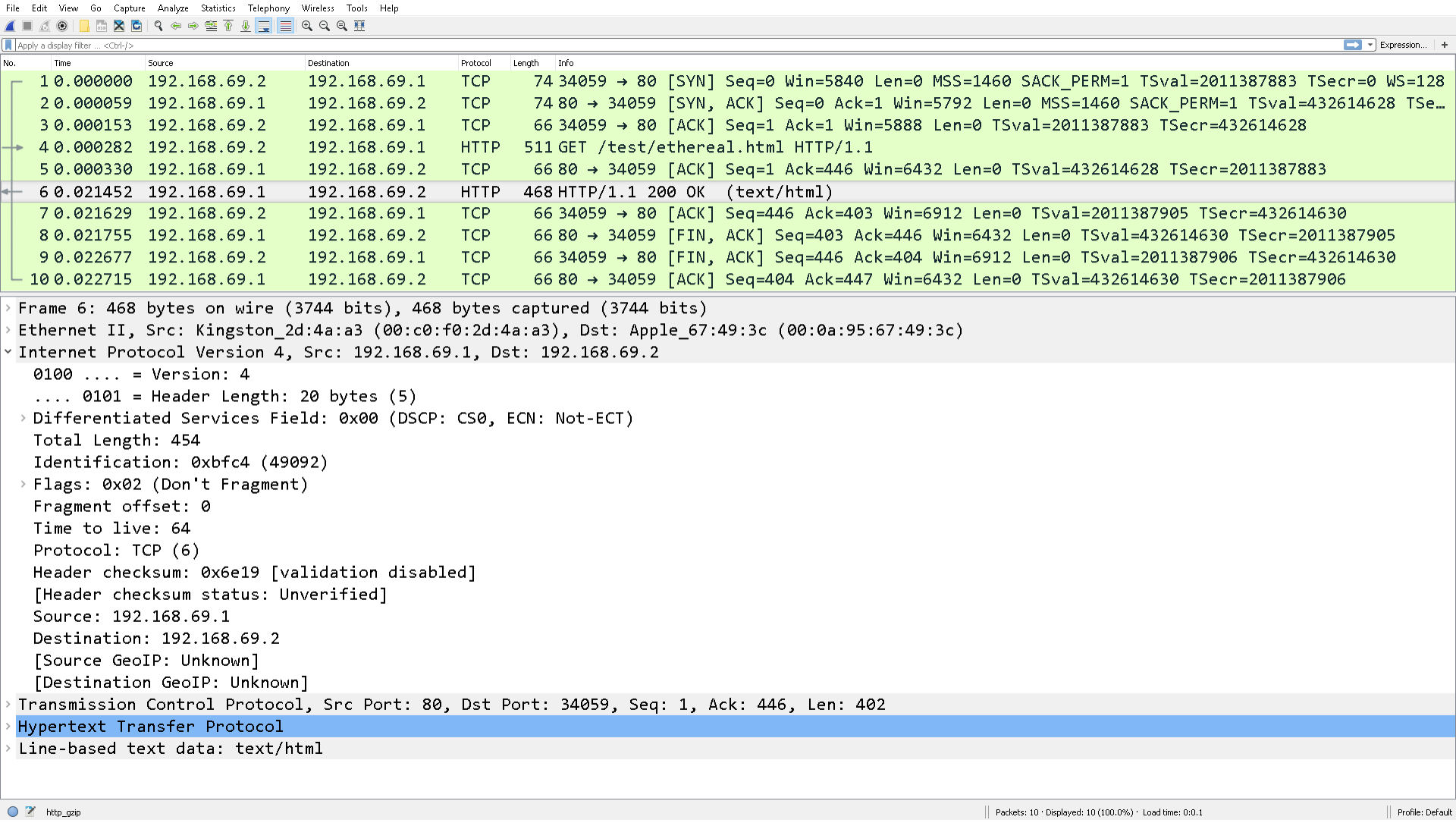
HTTP Response - HTML



HTTP Response - TCP



HTTP Response - IP



HTTP Response – Data Link