Applications REMINDER

2024/25 Q2

Jaime Delgado

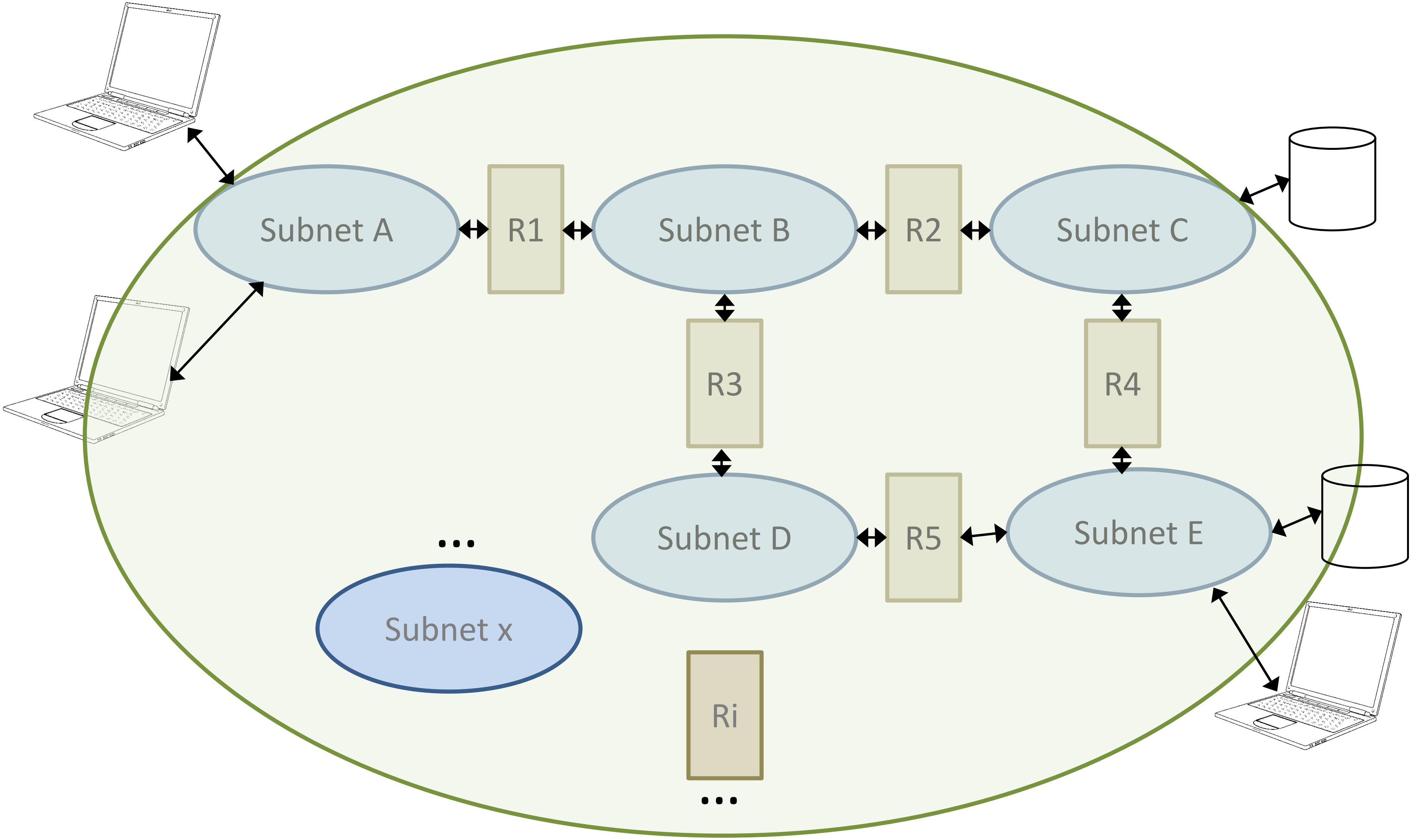
DAC – UPC

Contents

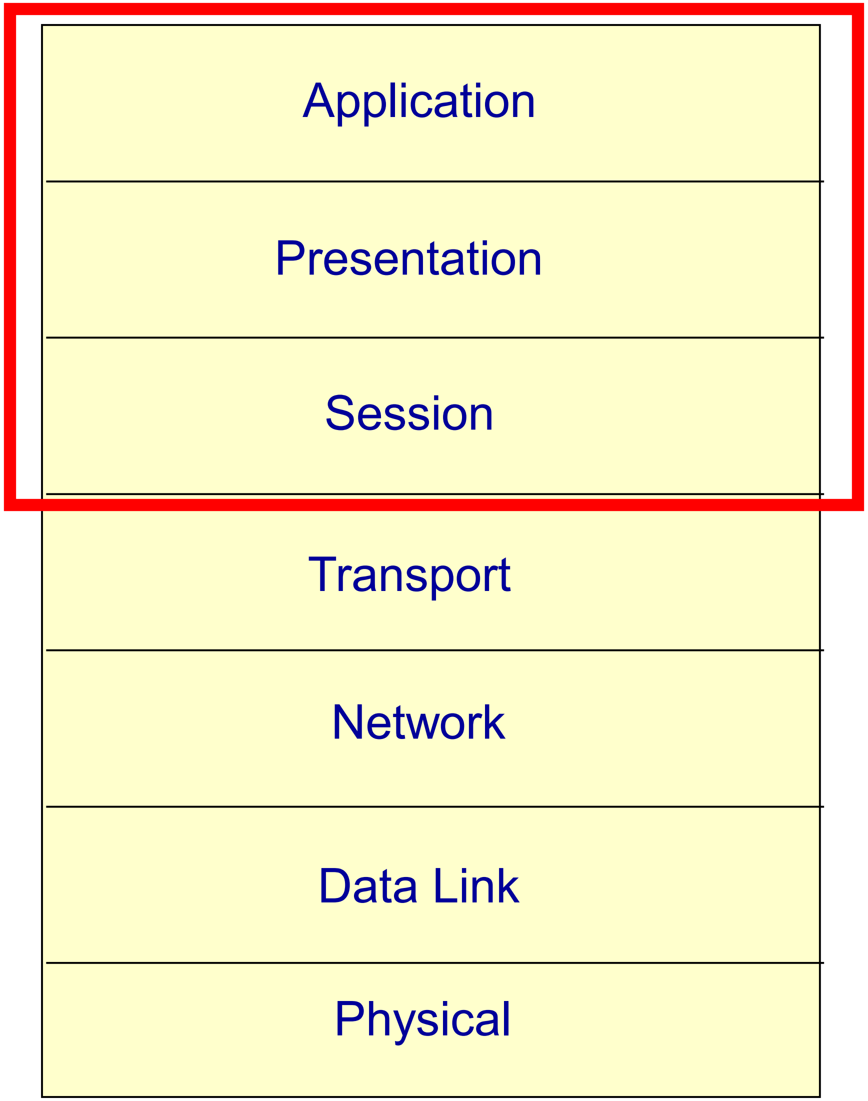
- The Internet
- Models, communication
- MIME
- Web (WWW)
- HTTP

The Internet

Ri: Router



OSI model



Application

Presentation

Session

Transport

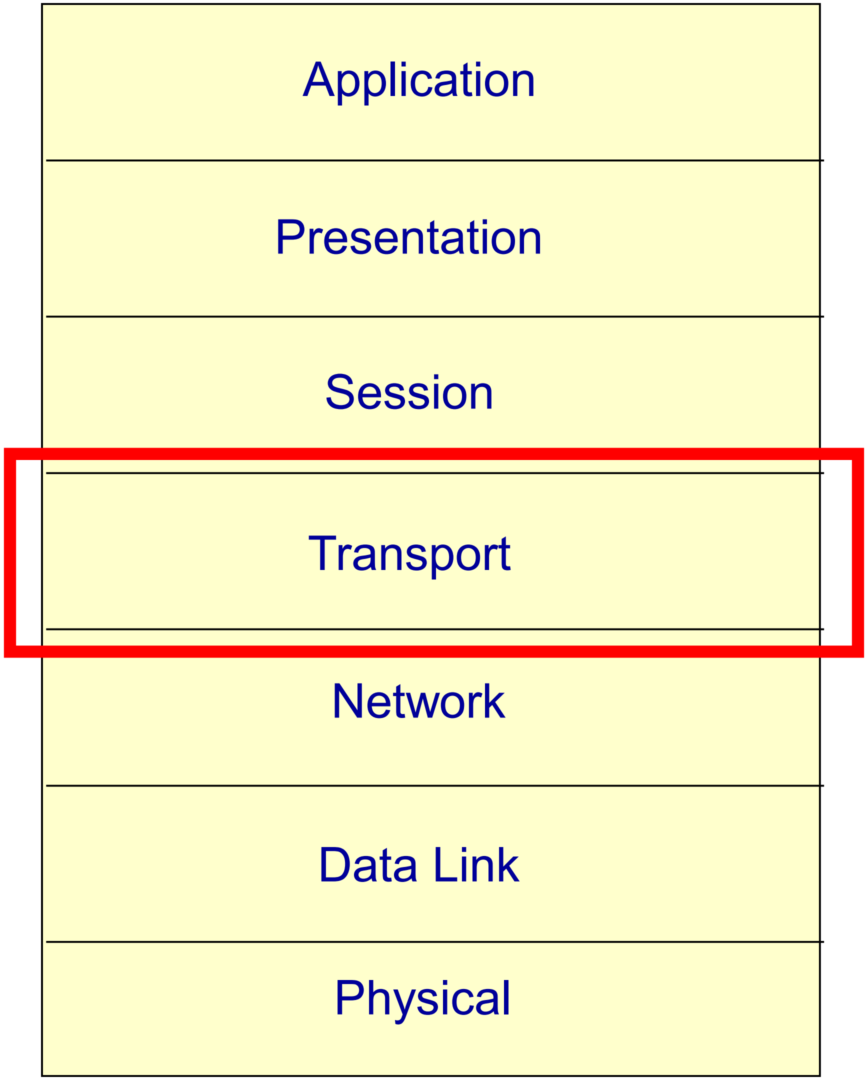
Network

Data Link

Physical

“Application”
layer

OSI model



Application

Presentation

Session

Transport

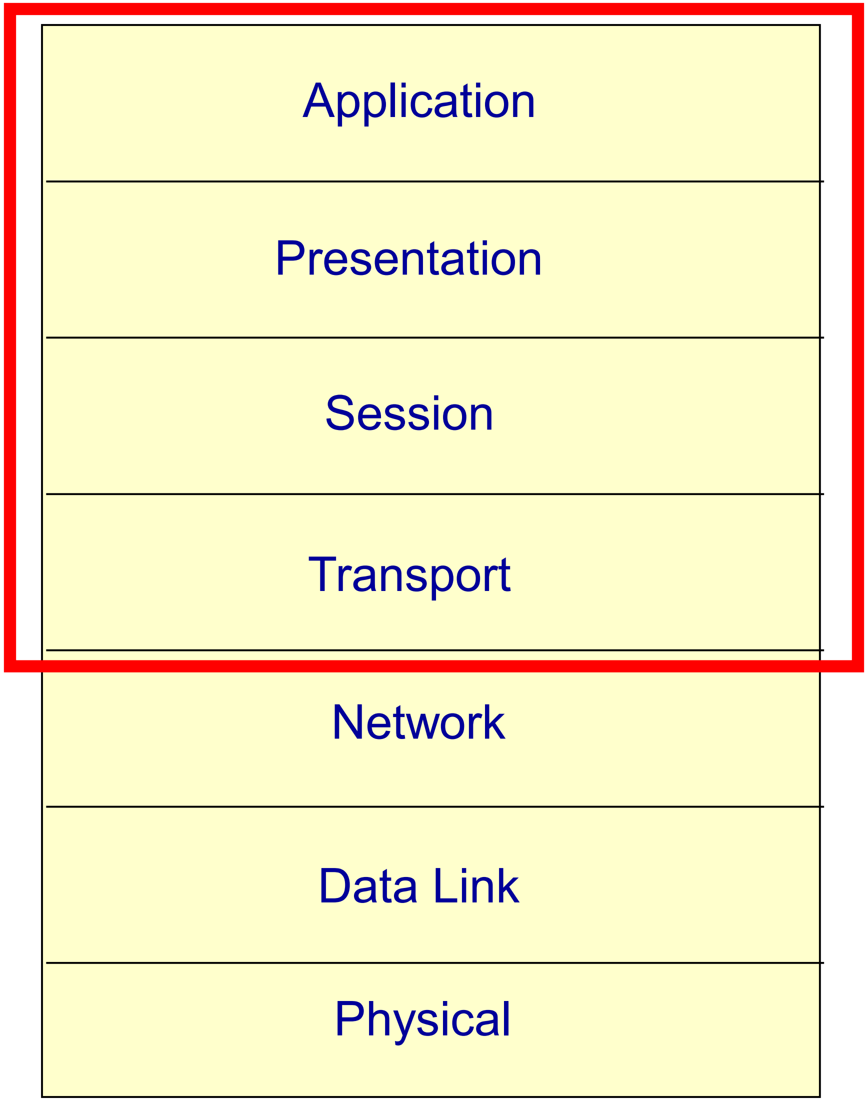
Network

Data Link

Physical

Application - Network

OSI model



Application

Presentation

Session

Transport

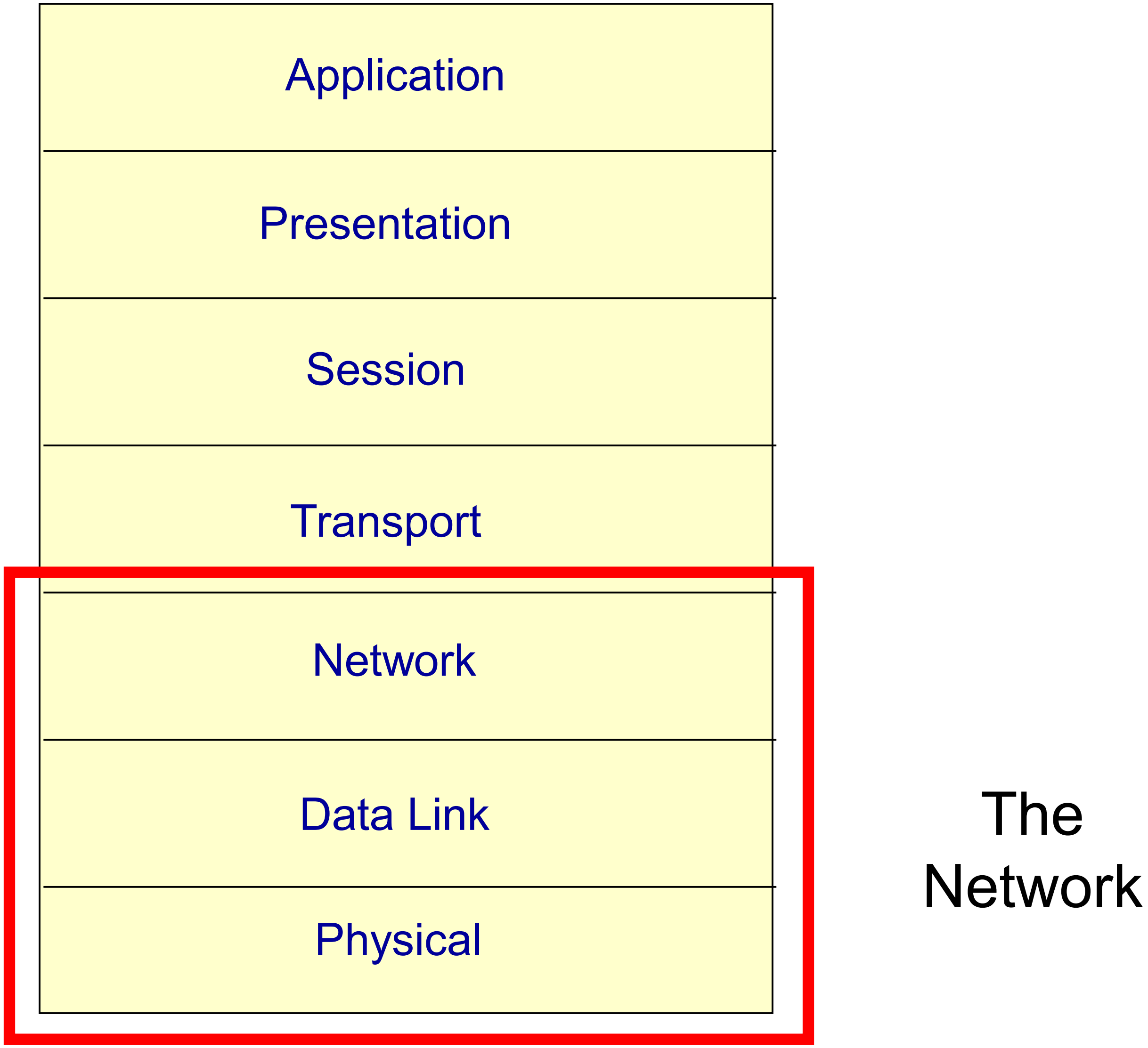
Network

Data Link

Physical

Only in the Computer (“Host”)

OSI model



Application

Presentation

Session

Transport

Network

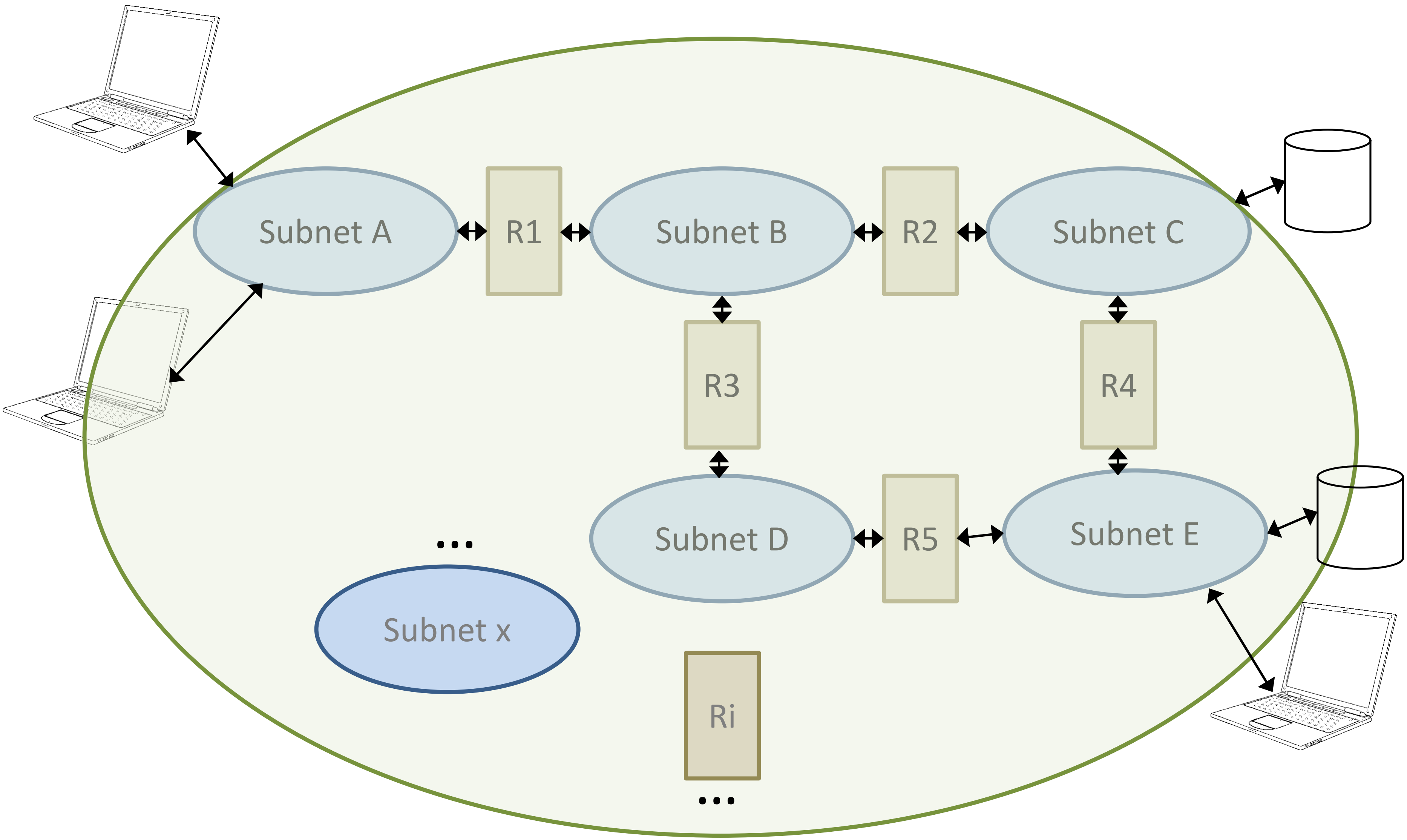
Data Link

Physical

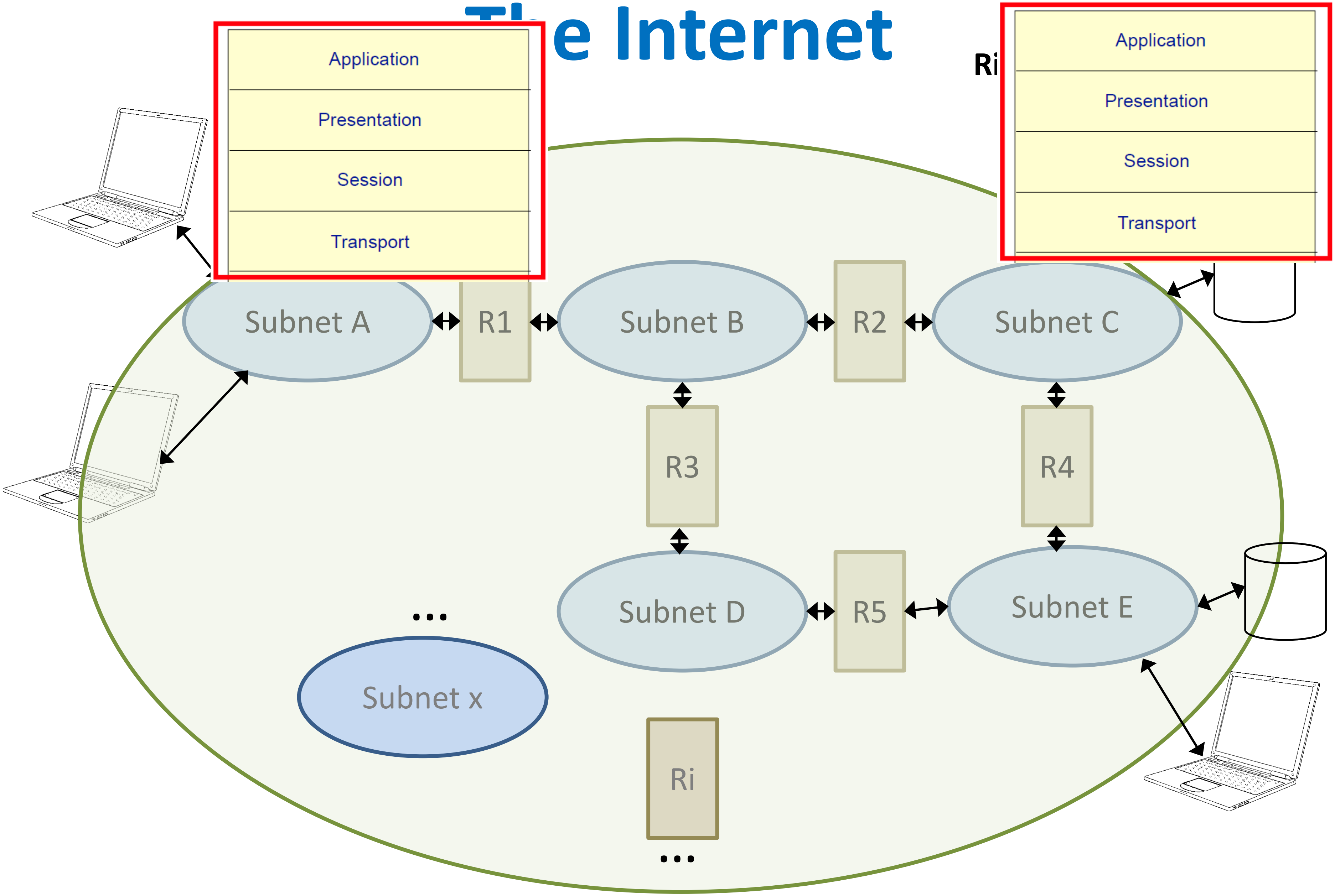
The Network

The Internet

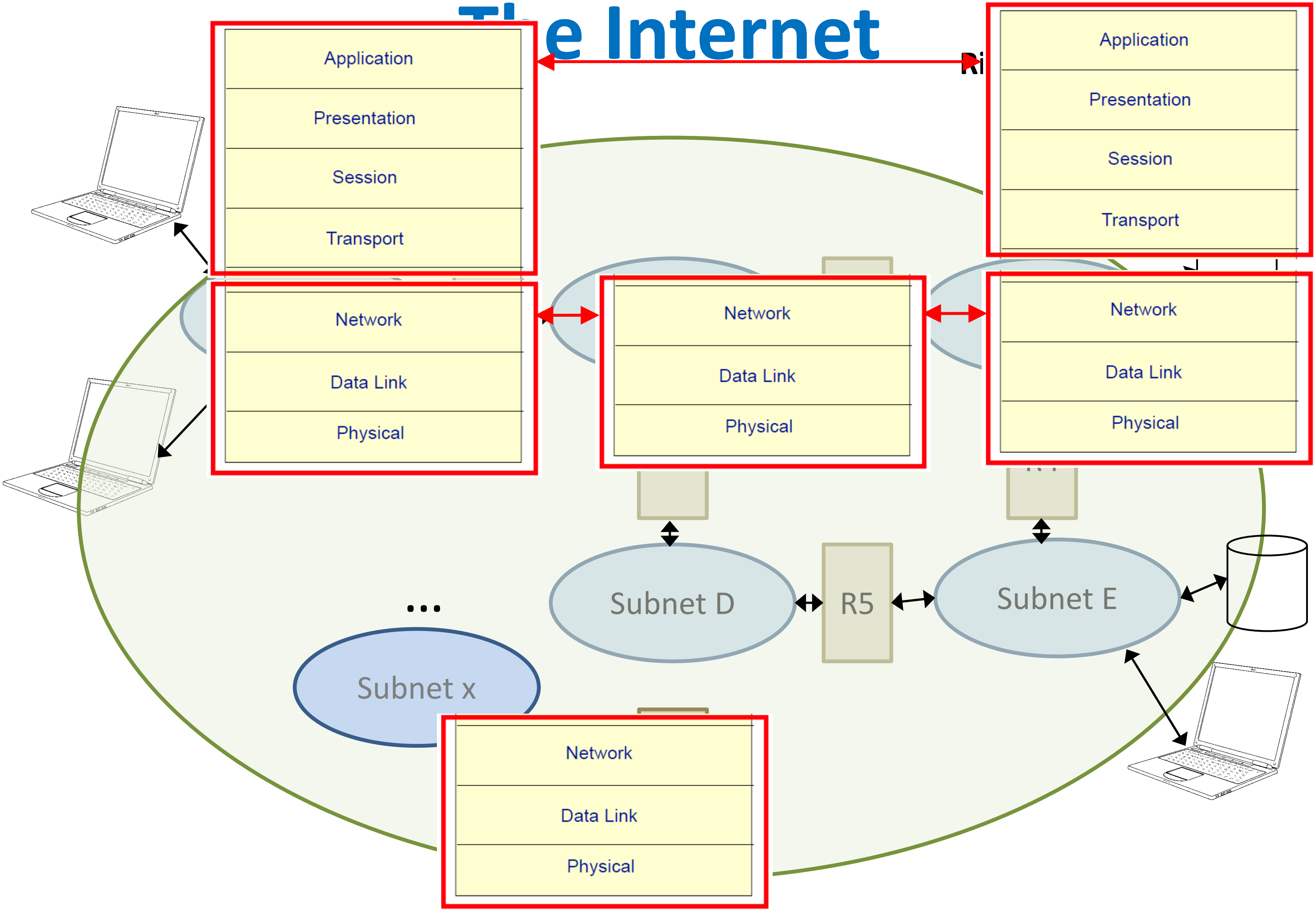
Ri: Router



The Internet



The Internet



Application

Presentation

Session

Transport

Network

Data Link

Physical

Application

Presentation

Session

Transport

Network

Data Link

Physical

Network

Data Link

Physical

Network

Data Link

Physical

...

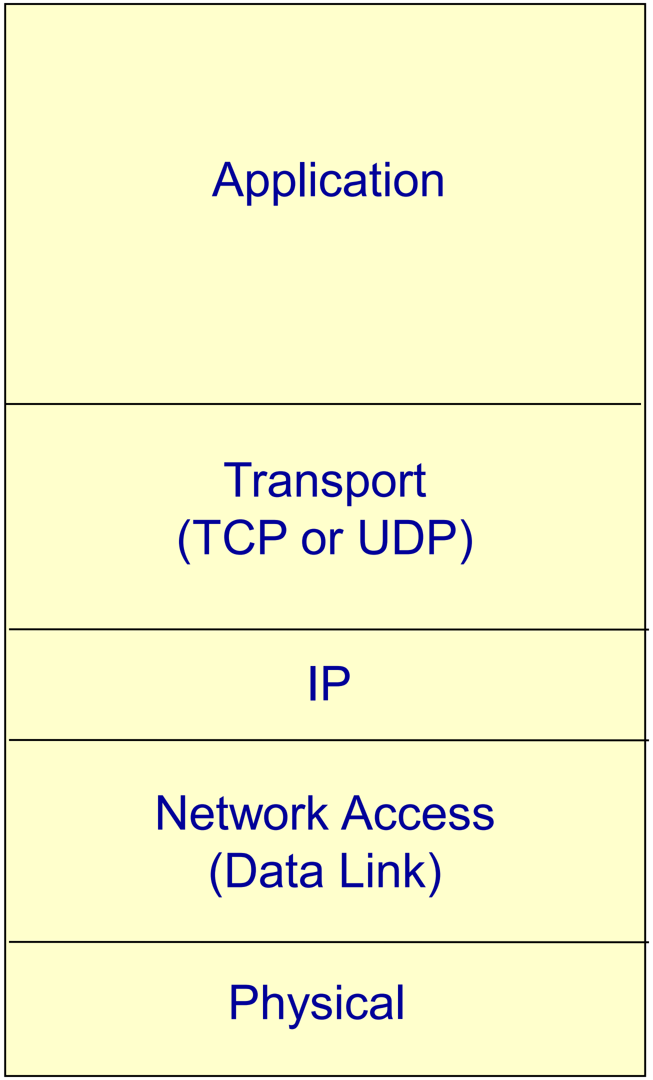
Subnet D

R5

Subnet E

Subnet x

Internet model



Application

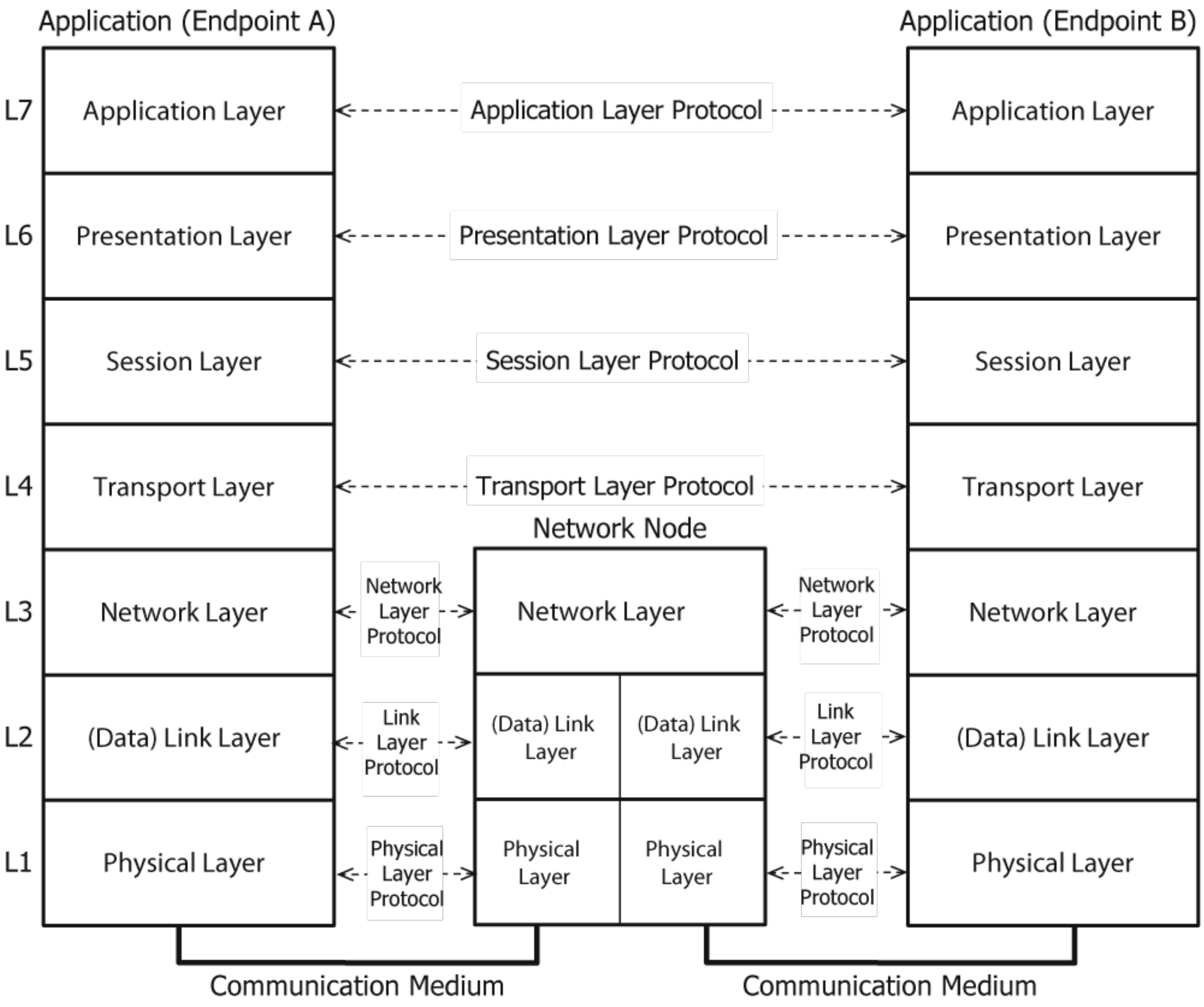
Transport
(TCP or UDP)

IP

Network Access
(Data Link)

Physical

Communication

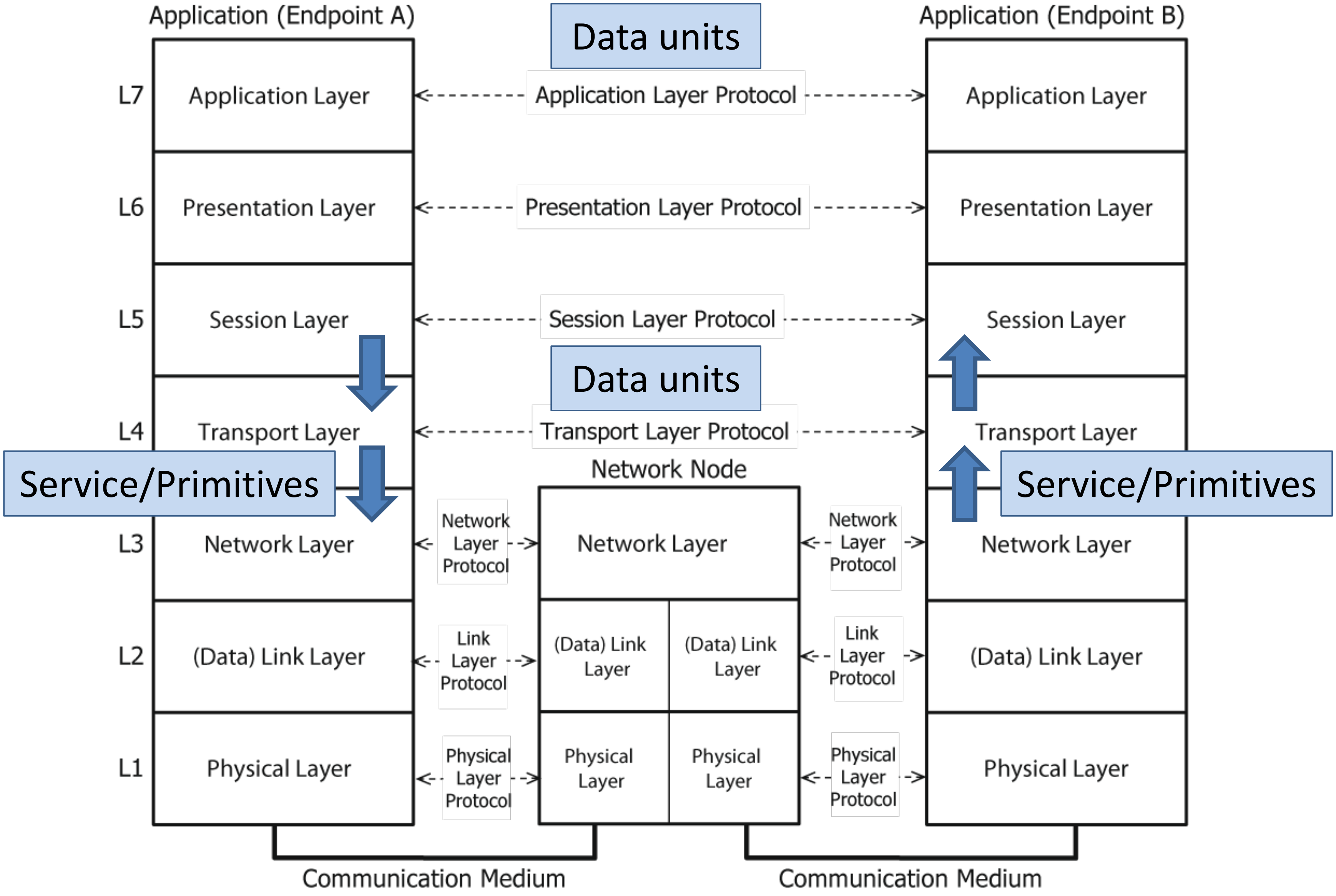


	Application (Endpoint A)		Network Node			Application (Endpoint B)
L7	Application Layer	Application Layer Protocol				Application Layer
L6	Presentation Layer	Presentation Layer Protocol				Presentation Layer
L5	Session Layer	Session Layer Protocol				Session Layer
L4	Transport Layer	Transport Layer Protocol				Transport Layer
L3	Network Layer	Network Layer Protocol	Network Layer		Network Layer Protocol	Network Layer
L2	(Data) Link Layer	Link Layer Protocol	(Data) Link Layer	(Data) Link Layer	Link Layer Protocol	(Data) Link Layer
L1	Physical Layer	Physical Layer Protocol	Physical Layer	Physical Layer	Physical Layer Protocol	Physical Layer

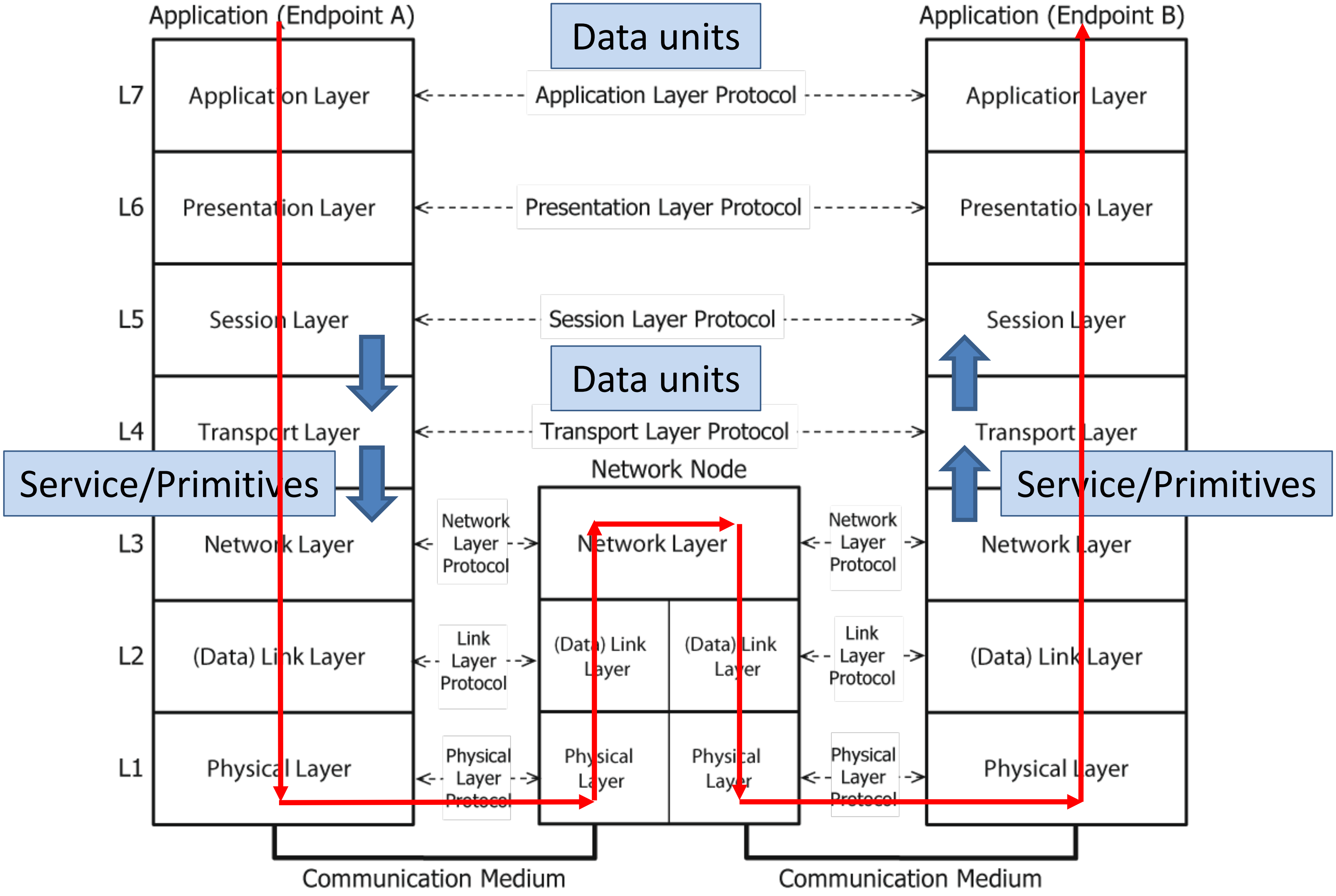
Communication Medium

Communication Medium

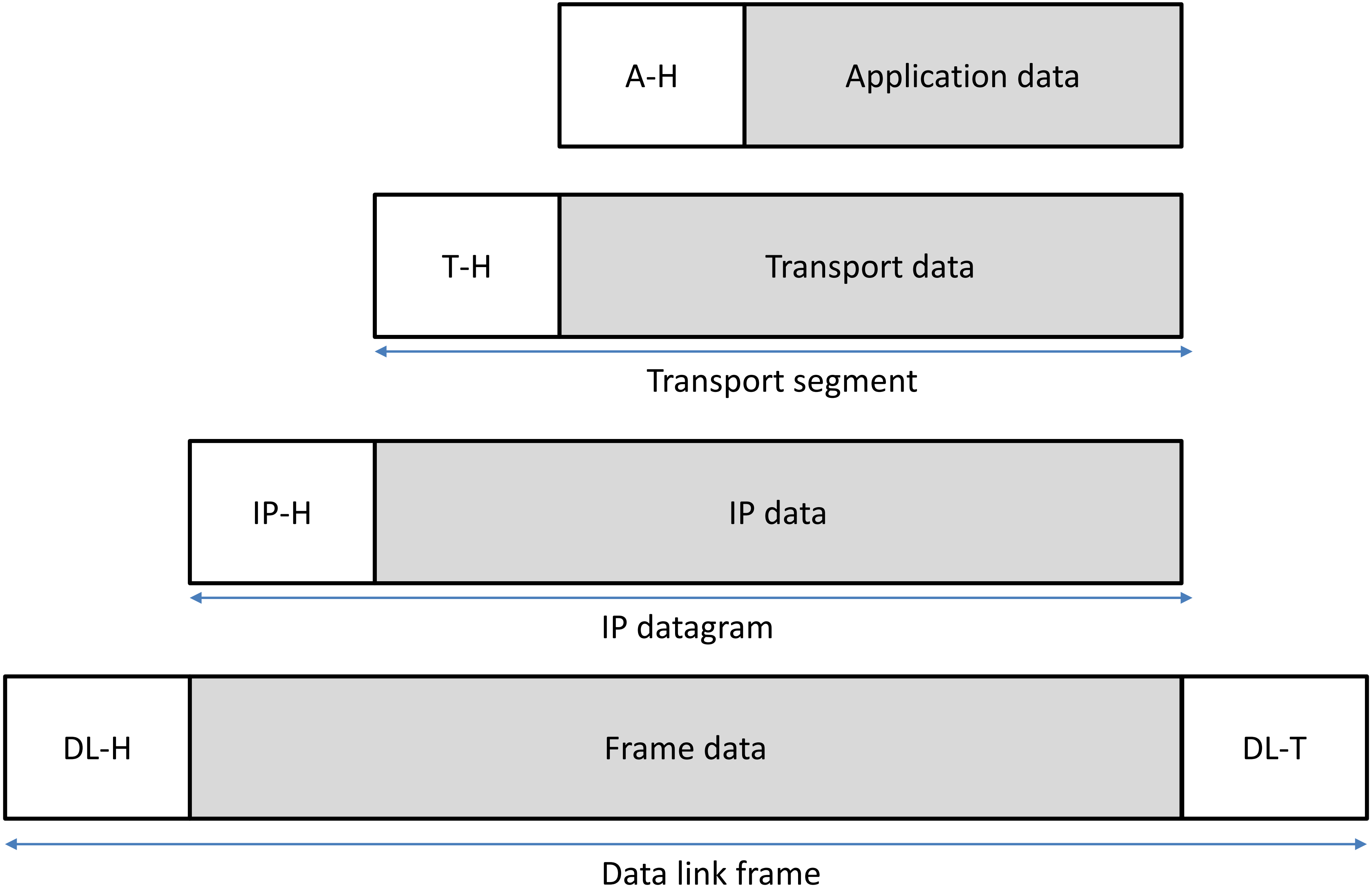
Communication



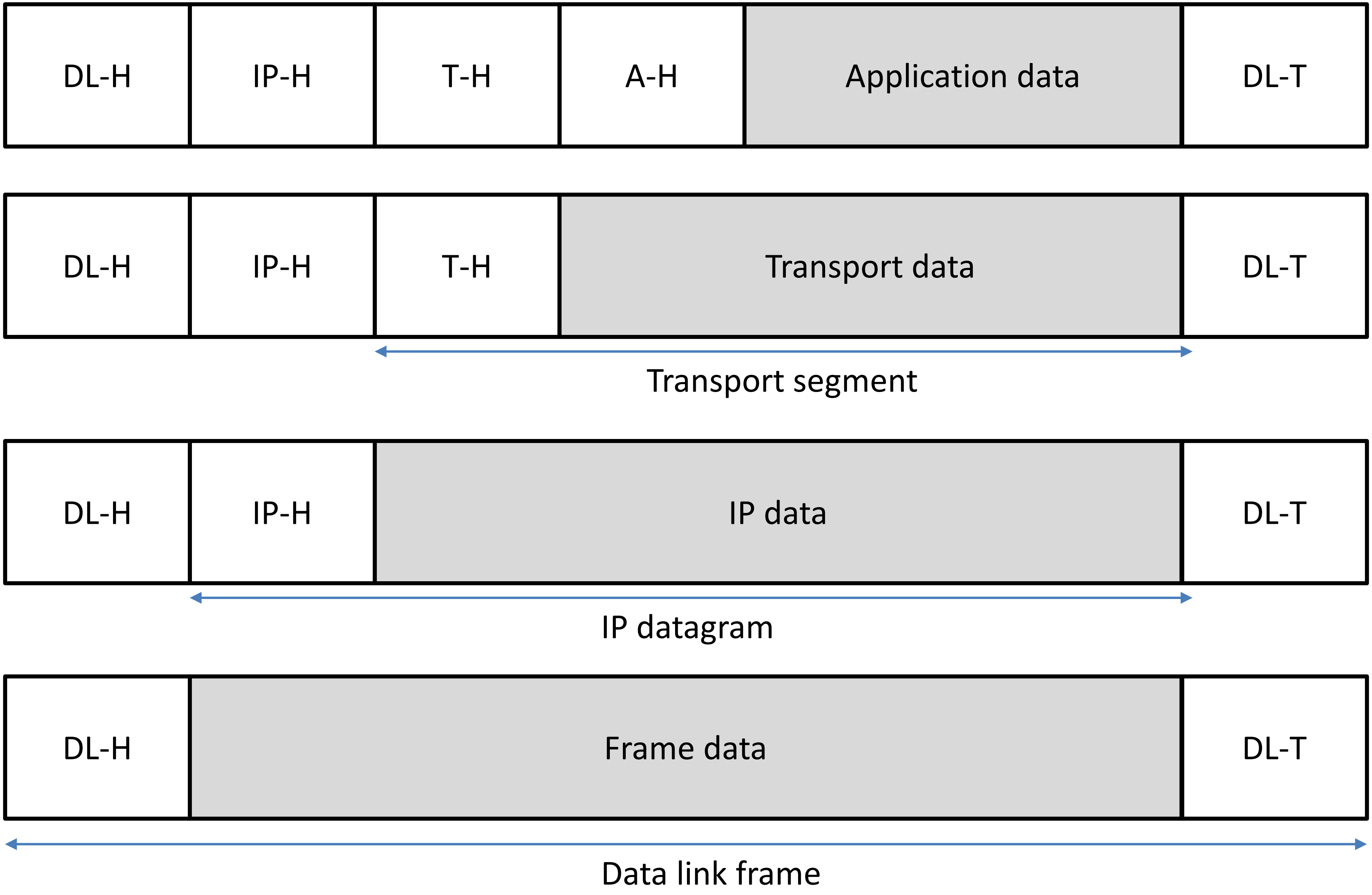
Communication



Protocol data units



Protocol data units



DL-H	IP-H	T-H	A-H	Application data	DL-T

DL-H	IP-H	T-H	Transport data	DL-T

Transport segment

DL-H	IP-H	IP data	DL-T

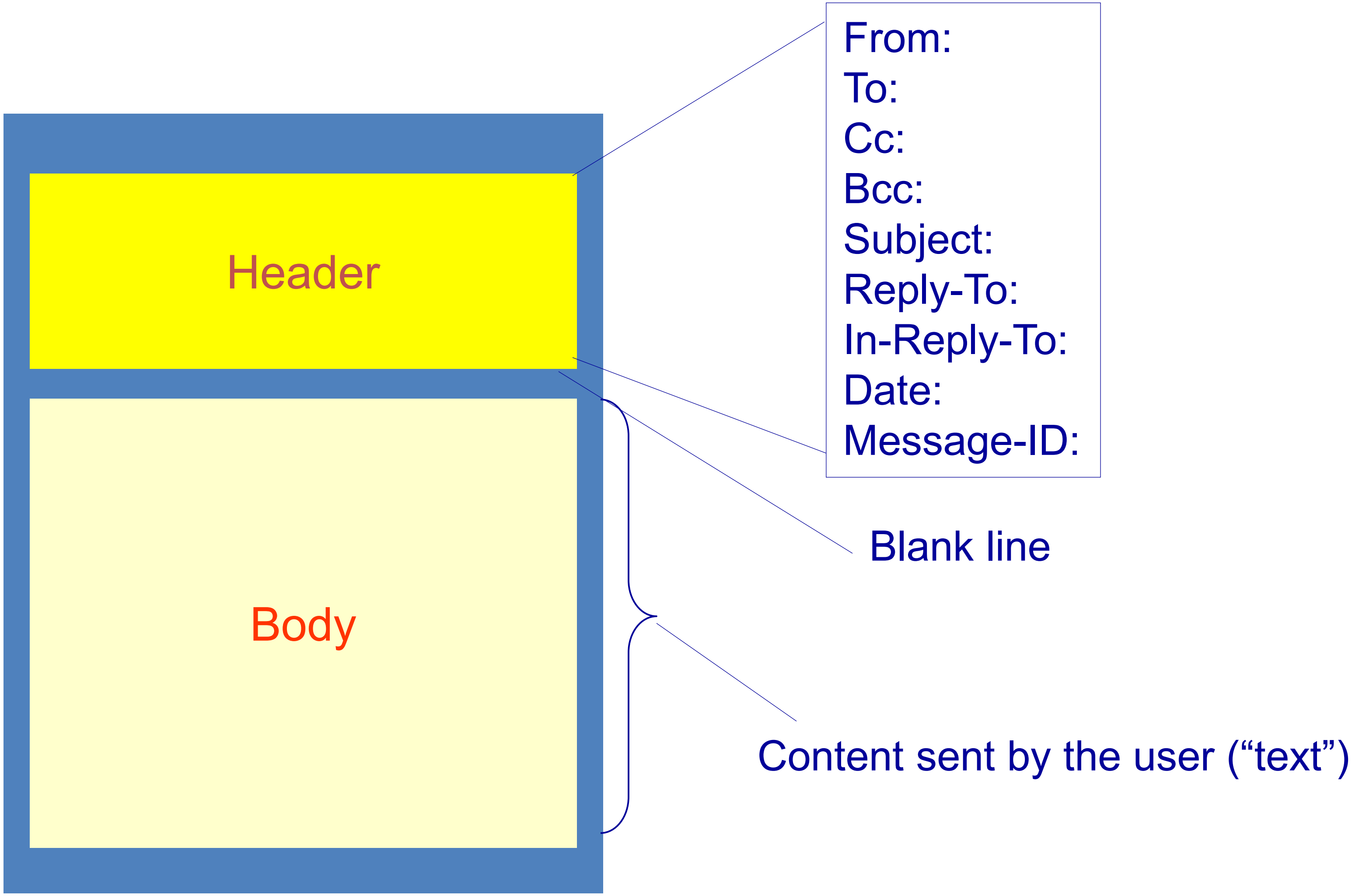
IP datagram

DL-H	Frame data	DL-T

Data link frame

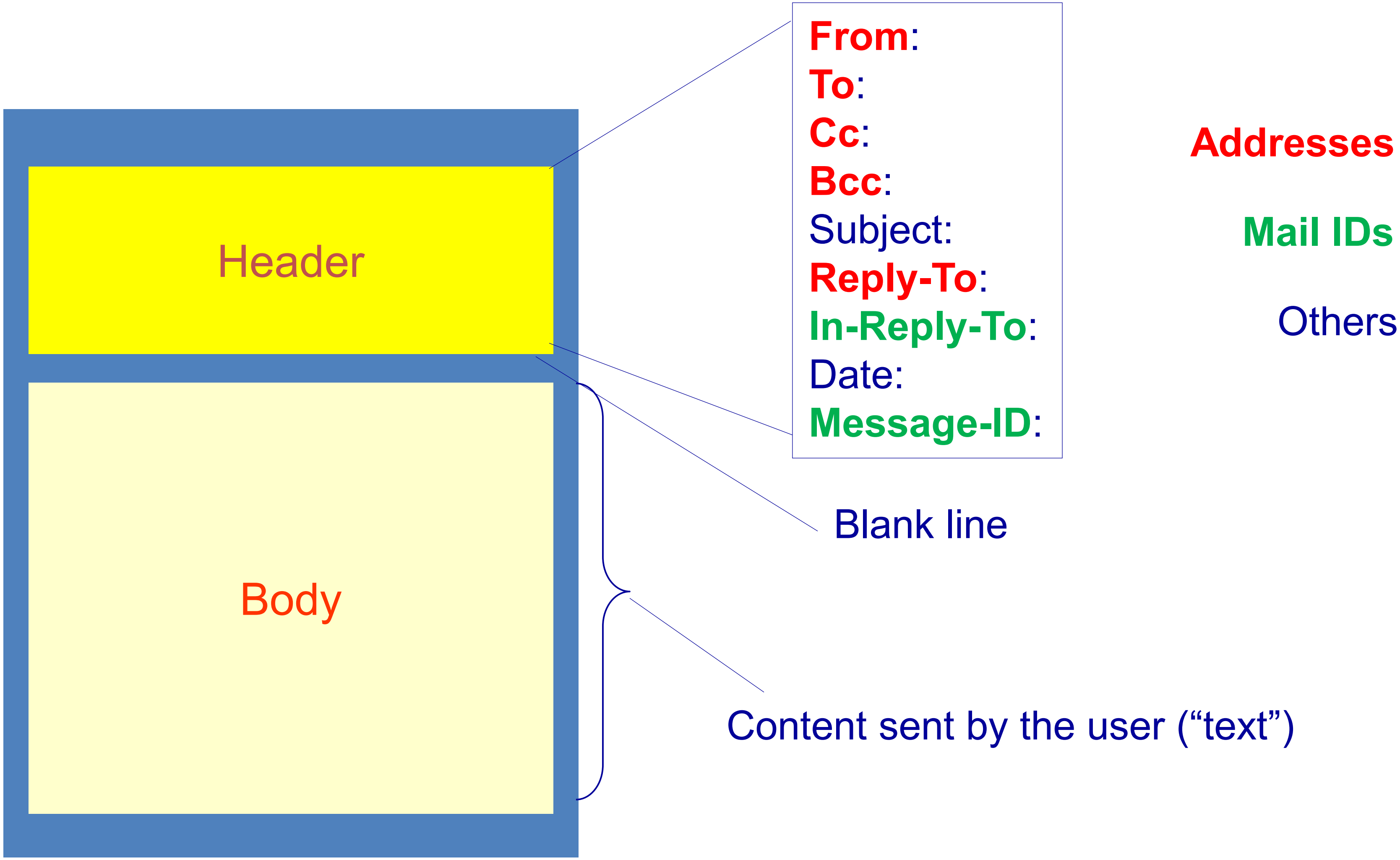
MIME

Original message format



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

Original message format



From:
To:
Cc:
Bcc:
Subject:
Reply-To:
In-Reply-To:
Date:
Message-ID:

Addresses

Mail IDs

Others

Blank line

Content sent by the user (“text”)

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

MIME

Multipurpose Internet Mail Extensions

Request For Comments: número de estándar es secuencial nos da una pista de la antigüedad. Algunos RFCs nacieron siendo directamente estándares.

- **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”
    ↳ usa 7 bits
  - Multipart messages
- Approach: Adding new header elements →

  `Content-Type`, …

- **MIME goes further away than mail!**
  (HTTP (Content types, …), *7 bits* environments)

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 media types

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

MIME media types

- application
- audio
- example *(RFC4735, ‘06)*
- font *(RFC8081, Feb’17)*
- haptics *(draft-ietf-mediaman-haptics-05, ’23!!)*
- image
- message
- model *(RFC2077, ‘97)*
- multipart
- text
- video

MIME media types

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

MIME media types

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

MIME media types

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

MIME media 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 media types

- MIME media subtypes for every type *(updated January 2025)*:
  - application (554+1073*=1627) *(standards+vendor)*
  - audio (122+39=161) *(“vendor” includes 3GPP, DVB,*
  - example (No subtypes) *ETSI, OASIS, OMA, …)*
  - font (6+0=6)
  - image (52+30=82)
  - haptics (3+0=3)
  - message (22+2=22)
  - model (21+20=41)
  - multipart (16+1=17)
  - text (59+37=96)
  - video (56+37=93)
  - TOTAL: 911+1239=2150 (a few repeated or obsoleted)

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

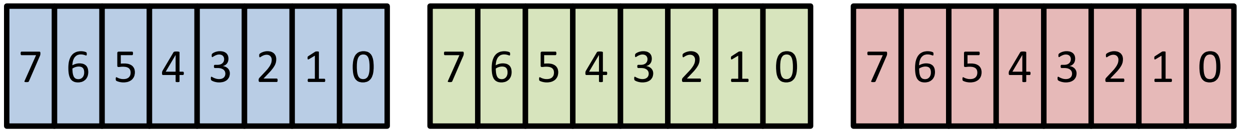
* 71 for vnd.openxmlformats-officedocument

MIME Content-Transfer-Encoding

- With “normal” SMTP servers (only 7-bit support):
  - “7bit”, “quoted-printable”
  - “**base64**”
- *SMTP Service Extensions* (for 8-bit support):
  Binary MIME: RFC 3030 (2000))
  8-bit MIME: RFC 1652 (1994) → **RFC 6152** (2011))
  - “8bit”, “binary” (no line length restriction to 1000 bytes)

Base64 encoding

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



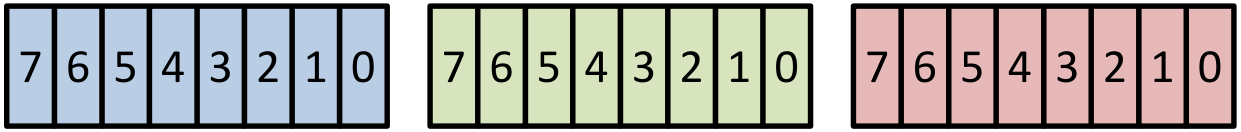
7 6 5 4 3 2 1 0

7 6 5 4 3 2 1 0

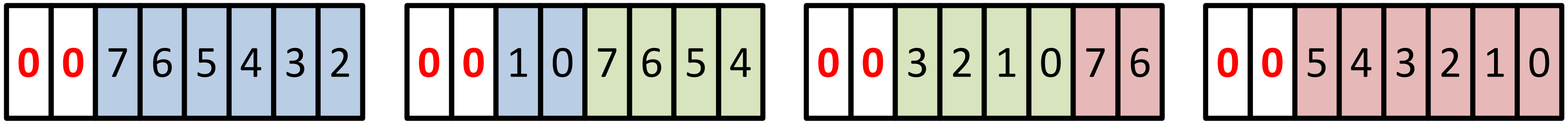
7 6 5 4 3 2 1 0

Base64 encoding

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

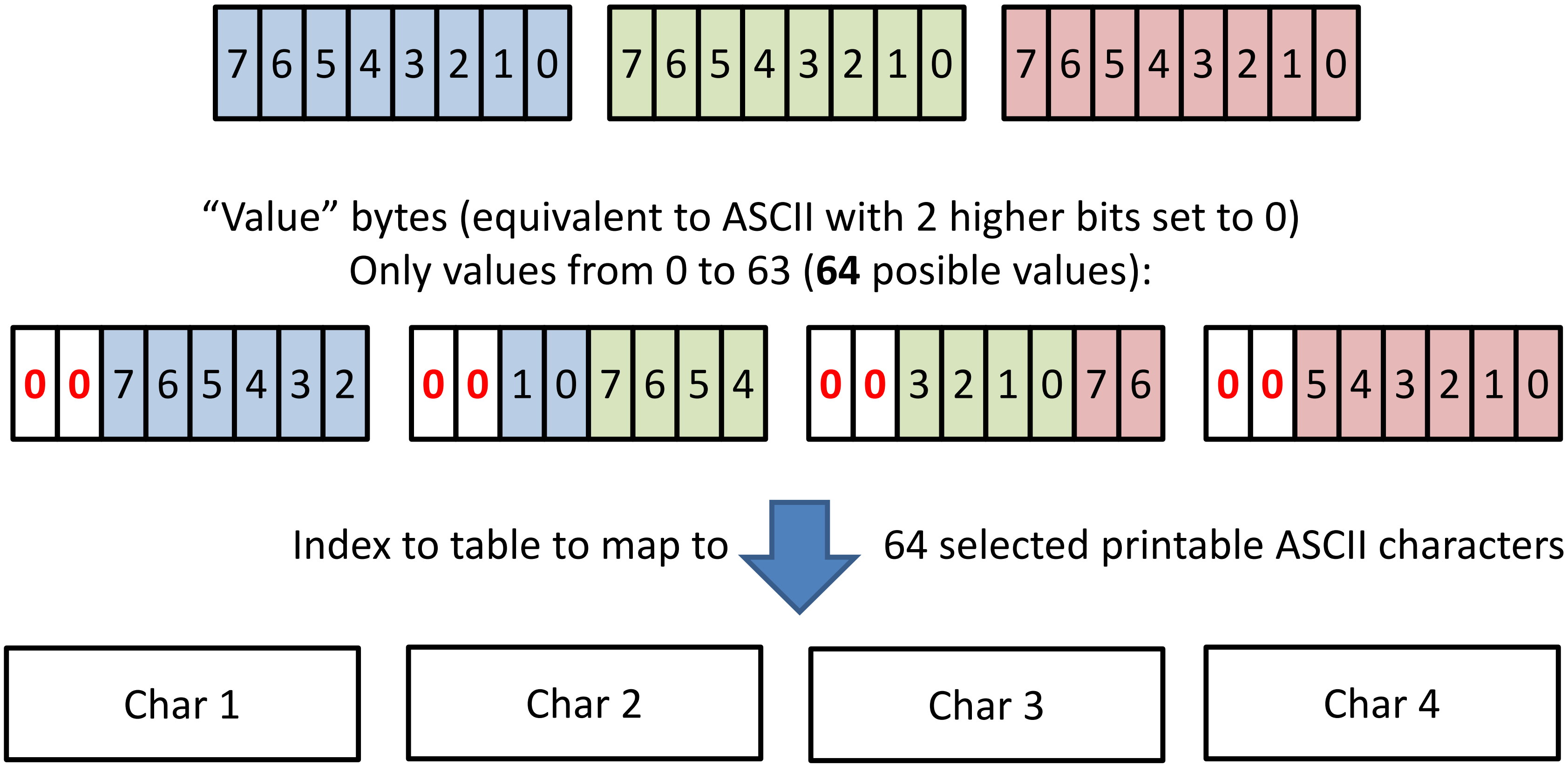


"Value" bytes (equivalent to ASCII with 2 higher bits set to 0)
Only values from 0 to 63 (**64** posible values):



Base64 encoding

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



7	6	5	4	3	2	1	0

7	6	5	4	3	2	1	0

7	6	5	4	3	2	1	0

“Value” bytes (equivalent to ASCII with 2 higher bits set to 0)
Only values from 0 to 63 (**64** posible values):

0	**0**	7	6	5	4	3	2

0	**0**	1	0	7	6	5	4

0	**0**	3	2	1	0	7	6

0	**0**	5	4	3	2	1	0

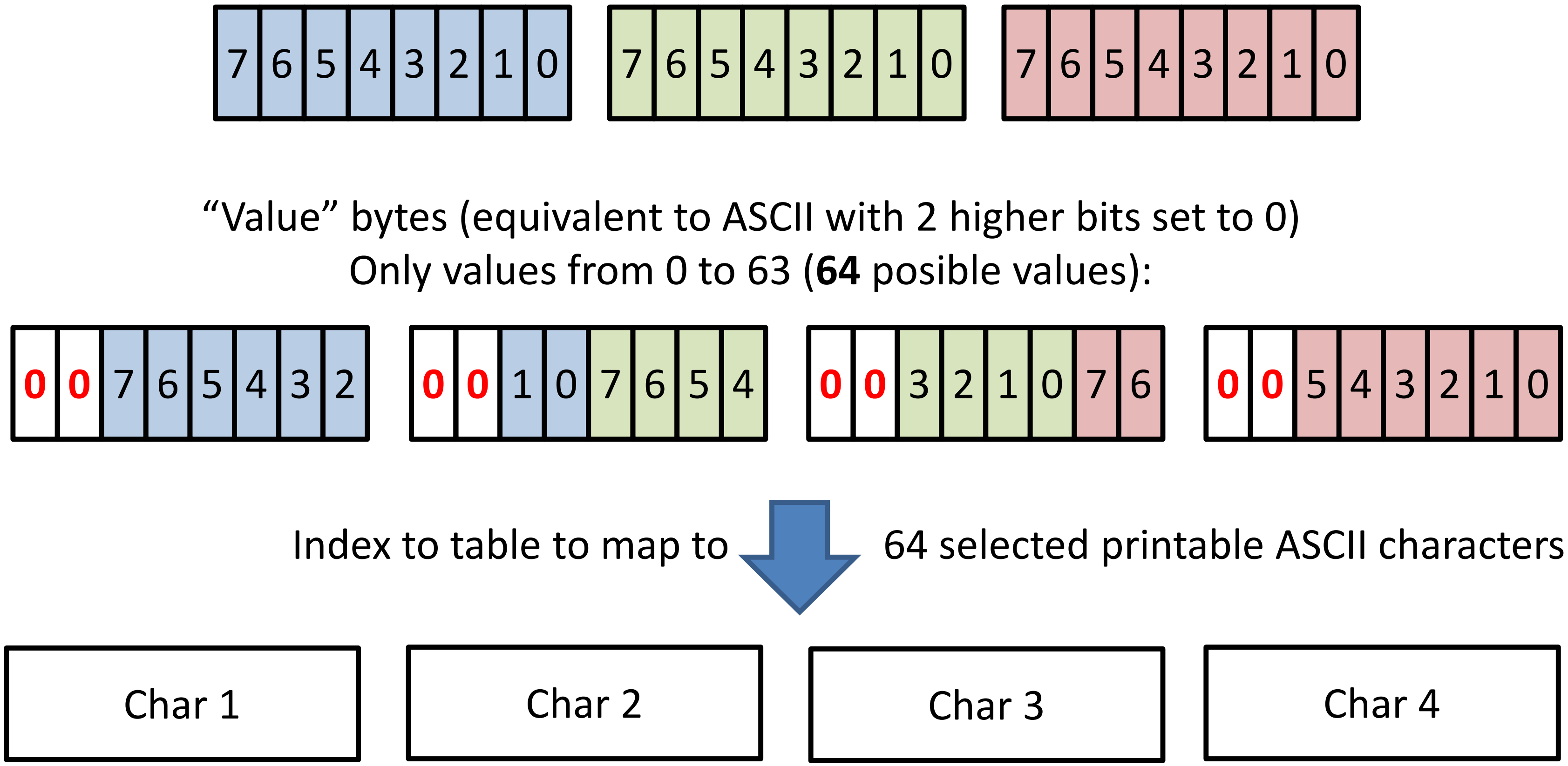
Index to table to map to 64 selected printable ASCII characters

Char 1	Char 2	Char 3	Char 4

“Encoded” bytes (sent as selected ASCII characters with higher bits set to 0)

Base64 encoding

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



7 6 5 4 3 2 1 0 | 7 6 5 4 3 2 1 0 | 7 6 5 4 3 2 1 0

“Value” bytes (equivalent to ASCII with 2 higher bits set to 0)
Only values from 0 to 63 (**64** posible values):

0 0 7 6 5 4 3 2 | 0 0 1 0 7 6 5 4 | 0 0 3 2 1 0 7 6 | 0 0 5 4 3 2 1 0

Index to table to map to 64 selected printable ASCII characters

Char 1 | Char 2 | Char 3 | Char 4

“Encoded” bytes (sent as selected ASCII characters with higher bits set to 0)

Inefficiency: 4 bytes transmitted for every 3!

Base64 encoding

- *RFC2045 (1996) Multipurpose Internet Mail Extensions (MIME).* ***Part One: Format of Internet Message Bodies***
- Encoding: 24-bit groups → strings of 4 encoded chars
- 24-bit formed by concatenating 3 8-bit input groups
- 24-bit treated as 4 concatenated 6-bit groups → each translated into a single *digit* in the **base64 alphabet**
- 6-bit group → index to an array of 64 printable chars
- Character referenced by the index placed in the output string
- Characters (*Table 1*) universally representable

Base64 encoding

Table 1: The Base64 Alphabet

Value	*Encoding*	*Value*	*Encoding*	*Value*	*Encoding*	*Value*	*Encoding*
0	A (65d 41h)	17	R	34	i	51	z (122d 7Ah)
1	B	18	S	35	j	52	0 (48d 30h)
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z (90d 5Ah)	42	q	59	7
9	J	26	a (97d 61h)	43	r	60	8
10	K	27	b	44	s	61	9 (57d 39h)
11	L	28	c	45	t	62	+ (43d 2Bh)
12	M	29	d	46	u	63	/ (47d 2Fh)
13	N	30	e	47	v		
14	O	31	f	48	w	(pad)	= (61d 3Dh)
15	P	32	g	49	x		
16	Q	33	h	50	y		

Web (WWW)

Web elements

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

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

Web elements

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

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

→ Internationalized Resource Identifier (IRI)

Links - URI (Uniform Resource Identifier)

- URI Generic Syntax: RFC 3986 (2005)

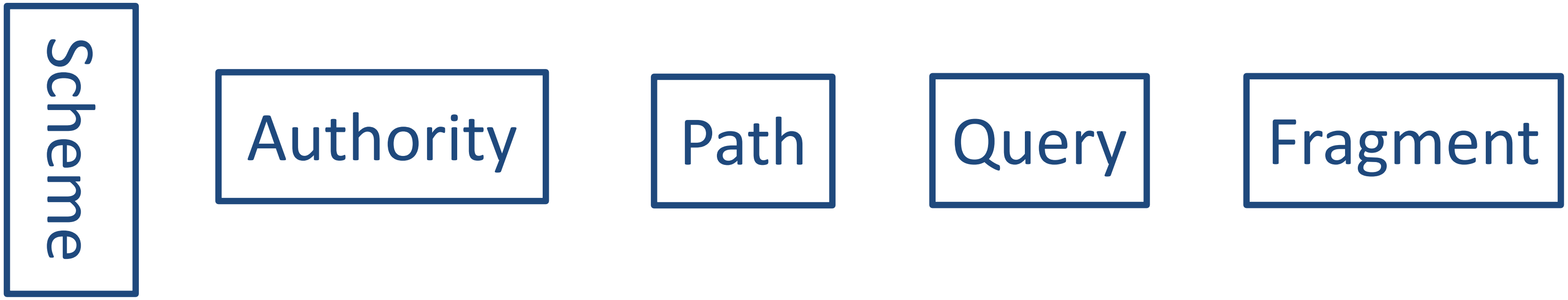
```
URI =
scheme ":" hier-part [ "?" query ] [ "#" fragment ]

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



Scheme

Authority

Path

Query

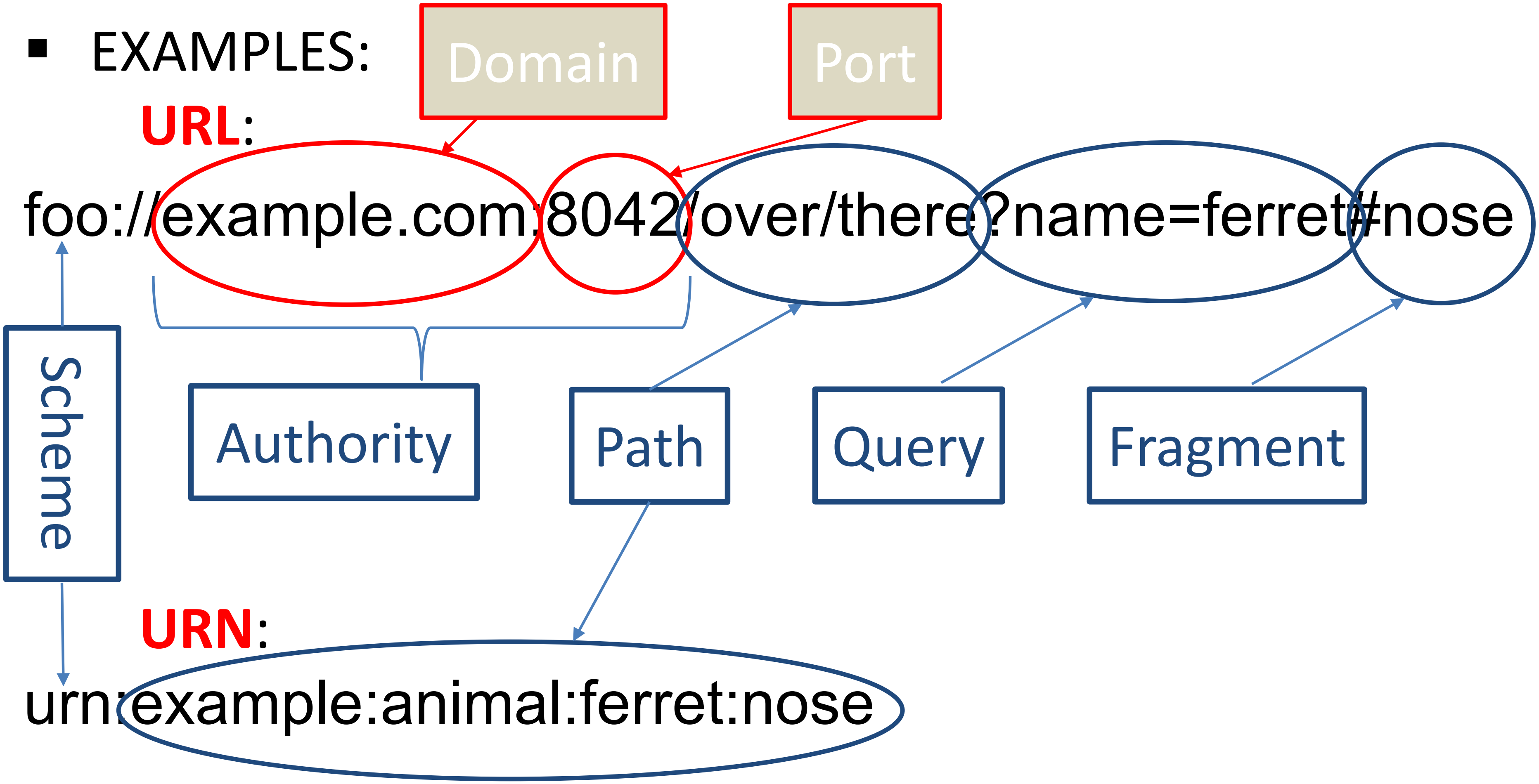
Fragment

URN:

urn:example:animal:ferret:nose

Links - URI (Uniform Resource Identifier)

- EXAMPLES:



URL:

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

Domain

Port

Scheme

Authority

Path

Query

Fragment

URN:

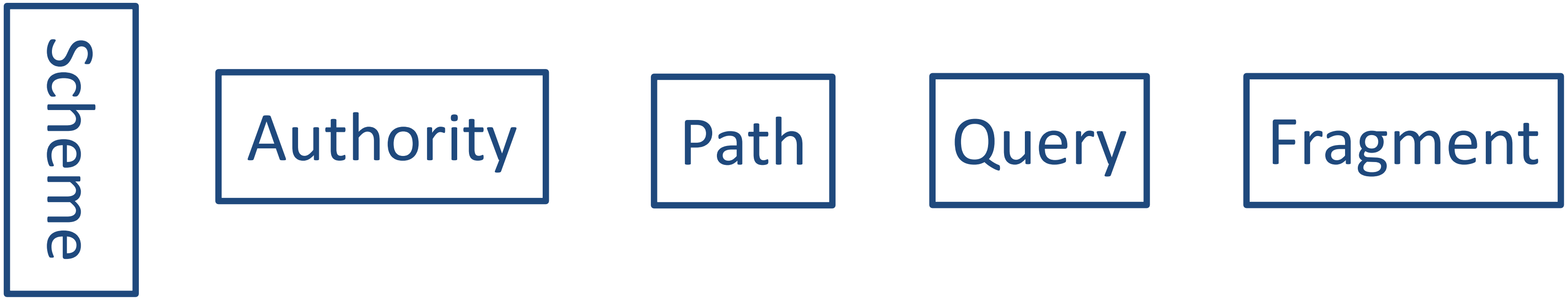
urn:example:animal:ferret:nose

Links - URI (Uniform Resource Identifier)

- EXAMPLES:

  URL:

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



Scheme

Authority

Path

Query

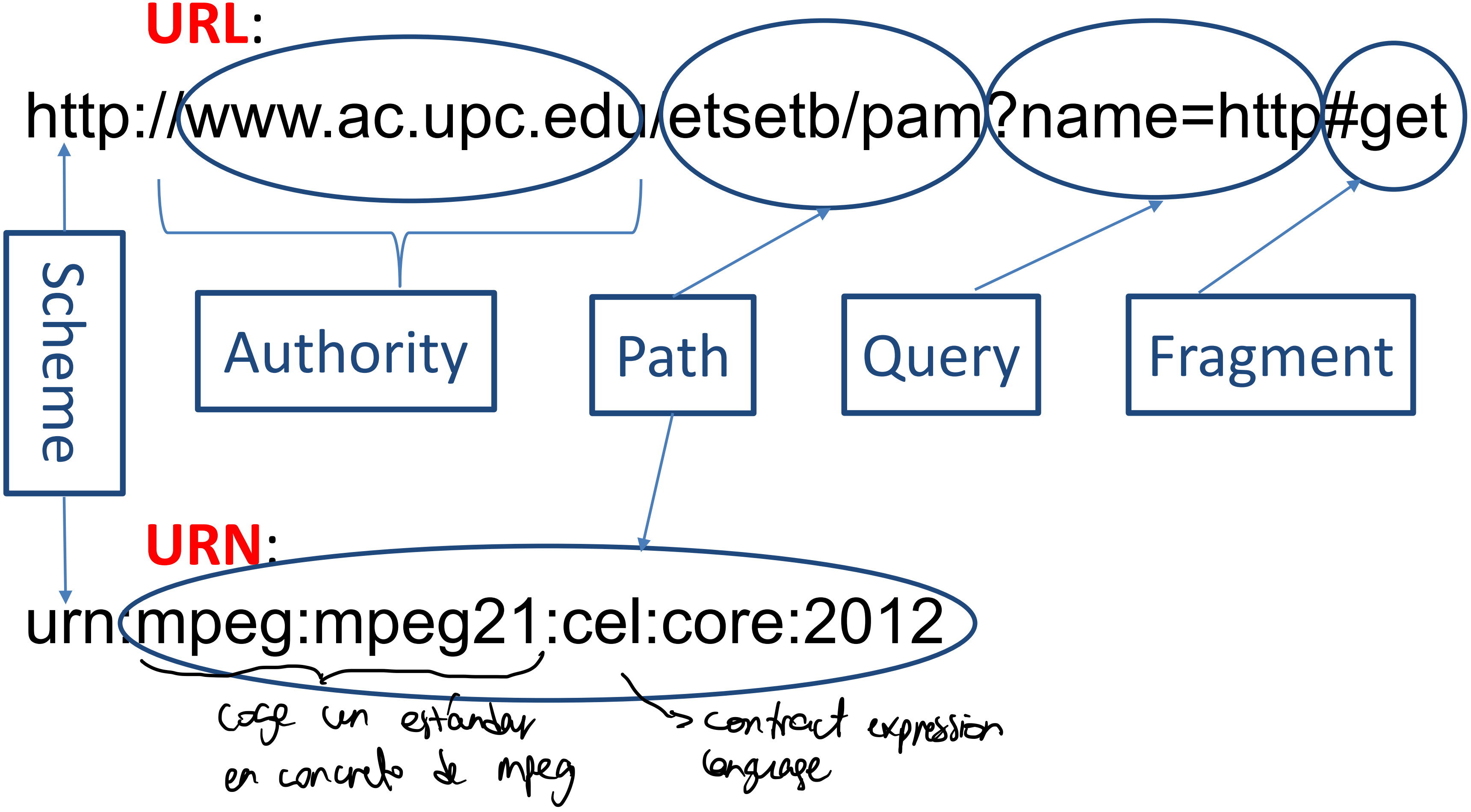
Fragment

URN:

urn:mpeg:mpeg21:cel:core:2012

Links - URI (Uniform Resource Identifier)

- EXAMPLES:



URL:

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

URN:

urn:mpeg:mpeg21:cel:core:2012

coge un estandar en concreto de mpeg

→ contract expression language

HTTP

HTTP introduction

- HyperText Transfer Protocol
- RFC 2616 (HTTP/1.1, 1999)
  Obsoleted by RFCs 7230 to 7235 (2014)
  --> **RFC 9110 (STD97) to 9112 (STD99)** June 2022!
  First version (0.9) in 1991. Now 2.0 already available.
  HTTP/2 **RFC 7540**, May 2015 → **RFC 9113**, June 2022
  > 17 versions 2012-2015!
  HTTP/3?!
- Stateless. Request/Response
- Normally over TCP (Port 80 as default)

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
- CONNECT. Requests to establish a tunnel (f.e., to create an end-to-end virtual connection)

Extensions:

- PATCH. Applies partial modifications to the resource (RFC 5789, 2010)

HTTP Request format

REQUEST LINE:

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

HEADER LINES:

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

BLANK LINE

BODY: ***Empty*** *for GET Request* (unless directly agreed)

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
  - 100, 101
- 2xx Successful
  - 200 OK, 201 Created, … 206
- 3xx Redirection
  - 300, 301 Moved Permanently, … 304 Not Modified, … 308
- 4xx Client Error
  - 400 Bad Request, 401 Unauthorized, 402
  - 403 Forbidden, 404 Not Found, … 418, 421, 422, 426
- 5xx Server Error
  - 500 Internal Server Error, 501 Not Implemented, 502
  - 503 Service Unavailable, … 505

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 …
Referer: http://www.google.com/
```

MIME types

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 …
```

Entity Tag

especie de hash para ver si en contenido es el mismo que otra web.

More on HTTP GET header lines

REQUEST:

Conditional

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

```
Range: bytes = 387-
```

del byte 387 hasta el final

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
…
```

HTTP/2

- Specifies how HTTP is expressed “on the wire”
- Same Methods, Status codes and semantics
- Focus on performance:
  - end-user perceived latency
  - network and server resource usage
  - allow the use of a single connection from browsers to a Web site

HTTP/2 structure

- Two layers of protocol:
  - Lower: general purpose used atop a reliable transport (likely TCP) for multiplexed, prioritized, & compressed data communication of concurrent streams
  - Upper: provides HTTP-like semantics for compatibility with existing HTTP application servers

HTTP/2 frame format

- Basic protocol unit: **frame** (different purposes)
- Frames: fixed 9-octet header followed by a variable-length payload

```
+-----------------------------------------------+
|                 Length (24)                   |
+---------------+---------------+---------------+
|   Type (8)    |   Flags (8)   |
+-+-------------+---------------+-------------------------------+
|R|                 Stream Identifier (31)                      |
+=+=============================================================+
|                   Frame Payload (0...)                      ...
+---------------------------------------------------------------+
```

`Length`: Of the frame payload.

`Type`: Of the frame.

`R`: Reserved.

HTTP/2 features

Frame types:

- DATA
- HEADER (to open a stream)
- PRIORITY (of a stream)
- RST_STREAM (immediate termination)
- PING
- WINDOW_UPDATE (flow control)
- CONTINUATION
- *New ones could be registered with IANA*

- **Server Push**
- **Flow control**
- **Prioritization**
- Redundant data → relevant frames **compressed**
- **Multiplexing** of requests: own streams for each request/response exchange → a blocked or stalled request or response progress on ot*does not preven*ther streams *("head of line blocking" problem solved)*