Fast campus

Network

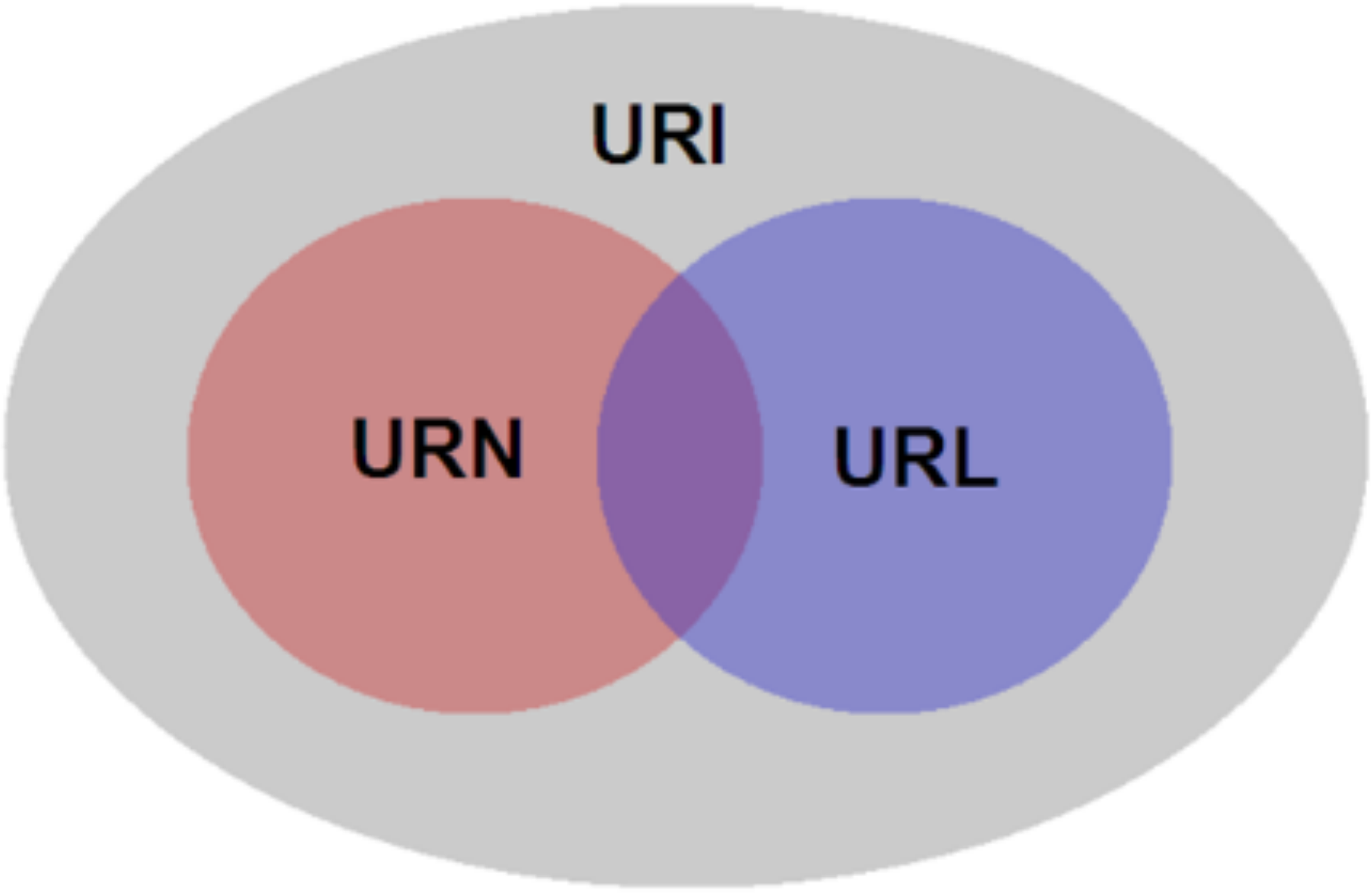
Giftbot

Fast campus

URI

Giftbot

URI, URL, URN

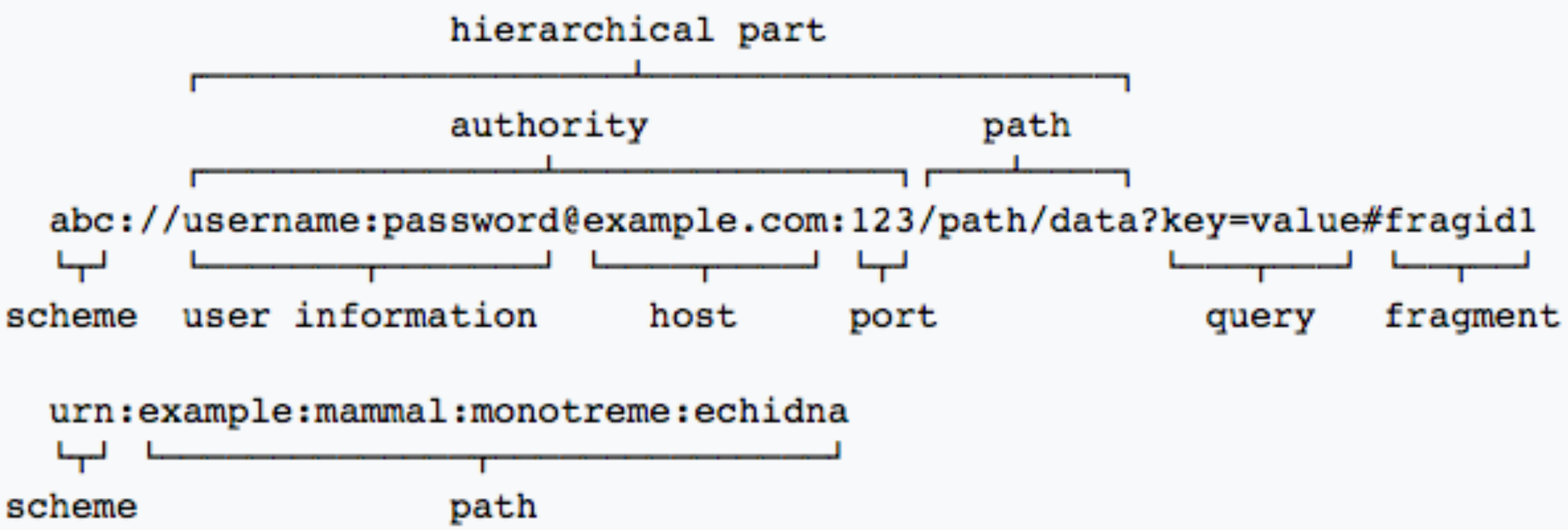


URI (Uniform Resource Identifier)

URI is a string of characters used to identify a name or abstract / physical resource

인터넷의 자원을 유일하게 식별하기 위한 통합 자원 식별자

절대적 경로와 상대적 경로를 모두 포함하는 URL 과 URN 의 SuperSet



URL (Uniform Resource Locator)

> URL is a useful but informal concept: a URL is a type of URI that identifies a resource via a representation of its primary access mechanism (e.g., its network "location"), rather than by some other attributes it may have.[1]

URL is a specific character string that constitutes a reference to a resource.

URL은 어떤 자원의 위치에 대한 절대경로값을 지닌 문자열

어떤 특정 주소 또는 파일 리소스에 접근하기 위한 주소값으로 흔히 HTTP 로 시작하는 웹 주소로 인식하는 경우가 많지만, URL 의 한 형태에 해당하는 것으로 컴퓨터 네트워크상의 자원을 모두 나타낼 수 있음

```
scheme:[//[user[:password]@]host[:port]][/path][?query][#fragment]
```

- http://www.someDomain.kr/index.php
- https://search.naver.com/search.naver?where=post&ie=utf8&query=url
- ftp://id:pw@someHost.com:21

URN (Uniform Resource Name)

URN is persistent, location-independent identifiers for resources

위치에 독립적이고, 지속되는 형태의 자원을 가르키기 위한 유일한 식별자

IETF 에서 표준 규격을 업데이트

```
urn:<namespace identifier>:<namespace-specific string>
```

- **urn:isbn:9788971991060 (국제표준도서번호)**
- **urn:ietf:rfc:2648 (RFC 문서번호)**
- **urn:uuid:6e8bc430-9c3a-11d9-9669-0800200c9a66 (UUID)**

URL vs URN

URL - 주소 / URN - 주민등록증

URL - 서울시 강남구 강남동 강남아파트 1층
URN - 철수(900101-1xxxxxx)
- 영희(900101-2xxxxxx)

강남아파트 1층에서 철수가 살다가 이사가고 영희가 이사를 왔을 때
주소(URL)는 동일하지만 사는 사람이 바뀌었고
철수와 영희의 경우는 사는 주소가 바뀌었지만 주민등록번호(URN)은 동일

영희가 유일하게 결정지어지는 식별자(URI)는
900101-2xxxxxx 라는 주민등록번호를 가진 사람(URN) 또는
현재 강남동 강남아파트 1층에 살고 있는 영희(URL) 가 된다.
(URL 단축주소를 사용하는 경우 지번 주소가 도로명 주소로 바뀐다 해도 최종 식별 대상이 바뀌지 않는다는 점을 생각하면 된다.)

URL vs URN

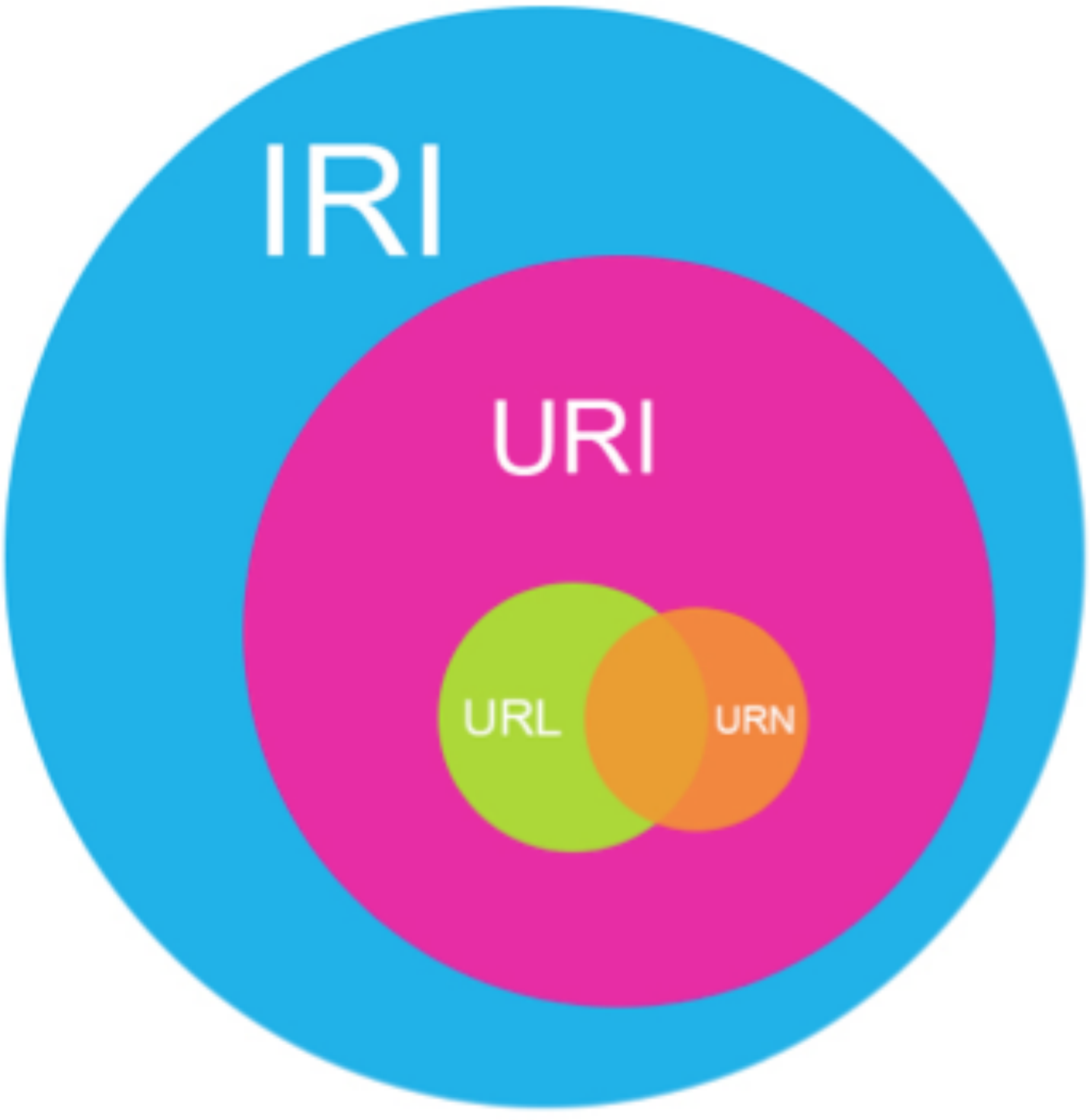
- URN
  - resource identification by name
  - teacher: “Student Jim!”

- URL
  - resource identification as location
  - teacher: “Student who seats in the 2nd row and 3rd column

in a class

	1	2	3
1	John	Grace	Jin
2	Sam	Tom	**Jim**
3	Kevin	Dan	Scott

IRI, URI



IRI (International Resource Identifier)

Defined by the IETF in 2005 as a new internet standard to extend upon the existing URI

URI - ASCII character set

IRI - Fully supports international characters, (mostly UTF-8)

즉 IRI 는 URI 의 상위 개념으로서 확장된 버전

- http://www.päypal.com

- http://www.paypal.com

위 예와 같이 URI 에서는 사용할 수 없는 문자를 이용해

비슷해 보이는 문자를 이용한 악의적인 공격이 있을 수 있음

ASCII Character Set

Fast campus

ASCII control characters

00	NULL	(Null character)
01	SOH	(Start of Header)
02	STX	(Start of Text)
03	ETX	(End of Text)
04	EOT	(End of Trans.)
05	ENQ	(Enquiry)
06	ACK	(Acknowledgement)
07	BEL	(Bell)
08	BS	(Backspace)
09	HT	(Horizontal Tab)
10	LF	(Line feed)
11	VT	(Vertical Tab)
12	FF	(Form feed)
13	CR	(Carriage return)
14	SO	(Shift Out)
15	SI	(Shift In)
16	DLE	(Data link escape)
17	DC1	(Device control 1)
18	DC2	(Device control 2)
19	DC3	(Device control 3)
20	DC4	(Device control 4)
21	NAK	(Negative acknowl.)
22	SYN	(Synchronous idle)
23	ETB	(End of trans. block)
24	CAN	(Cancel)
25	EM	(End of medium)
26	SUB	(Substitute)
27	ESC	(Escape)
28	FS	(File separator)
29	GS	(Group separator)
30	RS	(Record separator)
31	US	(Unit separator)
127	DEL	(Delete)

ASCII printable characters

32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

Extended ASCII characters

128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	░	208	ð	240	≡
145	æ	177	▒	209	Ð	241	±
146	Æ	178	▓	210	Ê	242	‗
147	ô	179	│	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	█	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	■
159	ƒ	191	┐	223	▀	255	nbsp

Fast campus

Byte Ordering

Giftbot

Byte Ordering

시스템이 내부적으로 데이터를 저장하고 표현하는 순서

CPU 벤더에 따라 바이트 단위로 데이터를 받아들이고 메모리에 저장하는 순서가 다른 데서 기인

Big Endian / Little Endian

그 외 Mixed, Middle, Bi 등의 Endian 이 있으나 많이 사용되지 않음

- Big Endian : IBM, Motorola, Sun Sparc, RISC, PowerPC 계열의 CPU
- Little Endian : Intel x86, x64, AMD 계열의 CPU

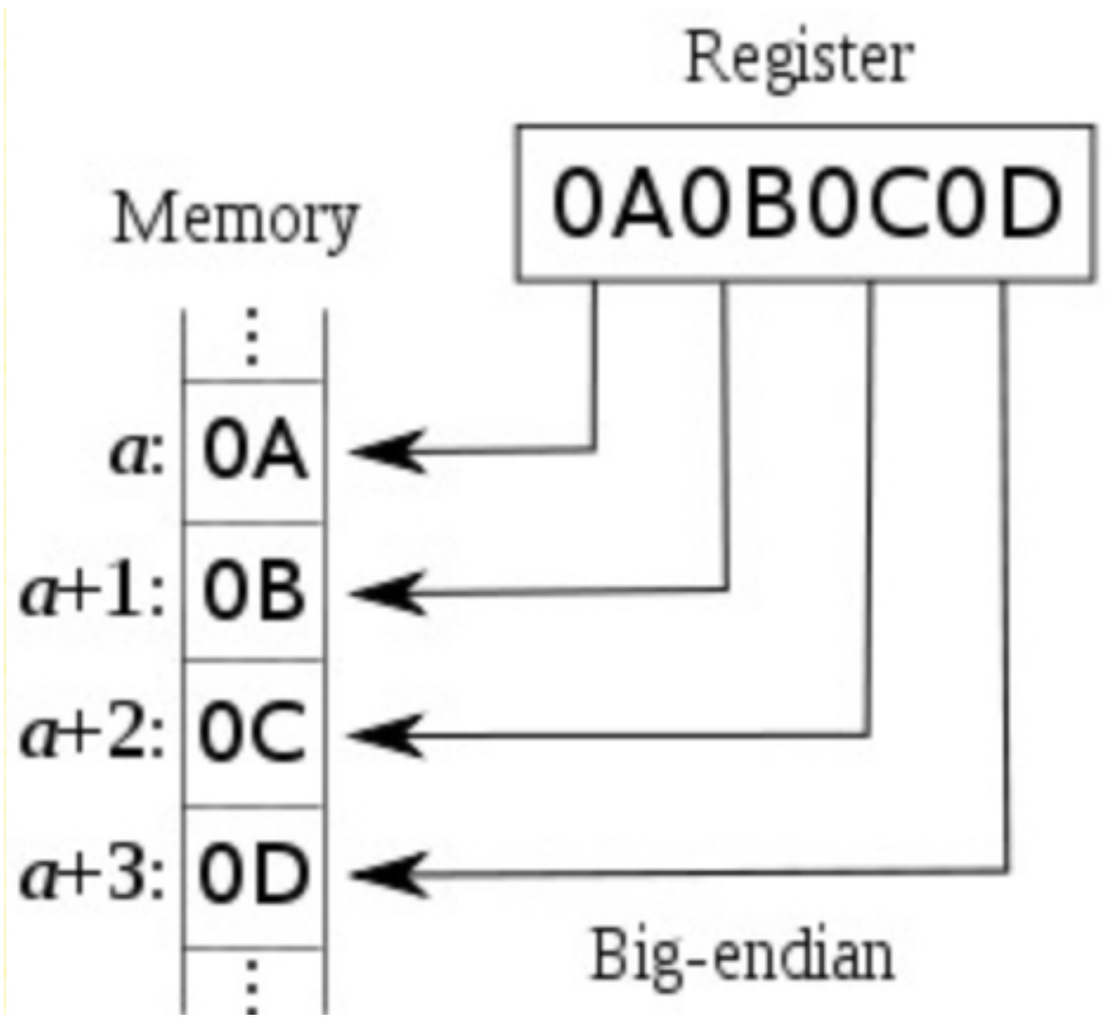
Big Endian

Network Ordering 이라고도 함

데이터를 상위 바이트부터 낮은 메모리 주소에 저장하는 형태

e.g. 0x12345678 => 0x12 0x34 0x56 0x78

낮은 주소 --> 높은주소



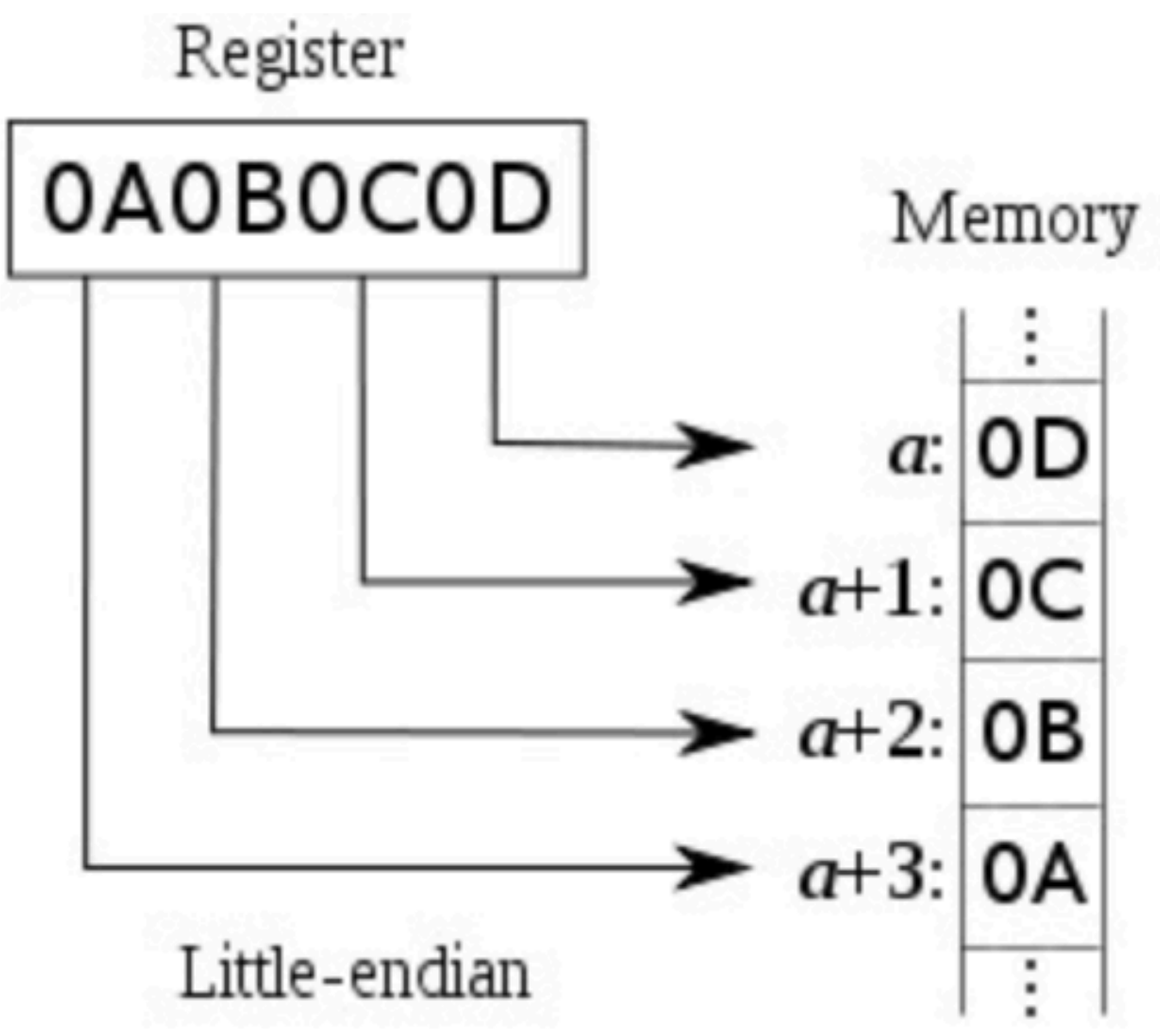
Little Endian

Host Ordering 이라고도 하며, iPhone 도 여기에 해당

데이터를 하위 바이트부터 낮은 메모리 주소에 저장하는 형태

e.g. 0x12345678 => 0x78 0x56 0x34 0x12

낮은 주소 --> 높은주소



Resolution

Endian 통일 (Network Byte Order)

- 모든 시스템이 저장하는 방식을 통일시키기 어려운 상황이므로

모든 프로그램이 네트워크 전송 시, 약속된 공통의 Endian 을 사용하고 수신 측에서 변환

- 네트워크 바이트 오더 표준 : Big Endian

Byte 단위 전송

- Endian 문제는 Byte 단위로 저장할 때 순서의 차이에 의해 발생하므로,

애초에 1 Byte 단위로 데이터를 보내면 바이트 순서에 구애받지 않고 통신 가능

Check Endian (Objective-C)

```objc
#import <Foundation/Foundation.h>
int main(int argc, const char * argv[]) {
  int a = 0x12345678;
  unsigned char *c = (unsigned char *)(&a);

  if (*c == 0x78) {
    NSLog(@"Little Endian");
  } else {
    NSLog(@"Big Endian");
  }

  return 0;
}
```

Check Endian (Swift)

Fast campus

```swift
let number: UInt32 = 0x12345678
let convertedToBig = number.bigEndian

if number == convertedToBig {
  print("BigEndian \(number)")
} else {
  print("LittleEndian \(number)")
}
```

Internet Protocol Suite

Giftbot

TCP/IP Protocol Suite

인터넷 프로토콜 스위트(Protocol Suite)는 인터넷 네트워크에 쓰이는 프로토콜의 모음

TCP/IP 가 가장 많이 사용되기에 TCP/IP 프로토콜 스위트 또는 프로토콜 스택이라고도 함

HTTP, FTP, Telnet 등이 TCP/IP 기반

- TCP (Transmission Control Protocol) : 패킷 전송 여부와 순서 보장의 특징

- IP (Internet Protocol) : 비신뢰성(패킷 전송 보장 X)과 비연결성의 특징 (IPv4, IPv6)

Kirk Bater
@KirkBater

Follow

This image is a TCP/IP Joke. This tweet is a UDP joke. I don't care if you get it.

Thread
iamkirkbater and jkjustjoshing

iamkirkbater Aug 23rd, 2017 at 9:37 AM
in #www

Do you want to hear a joke about TCP/IP?

7

7 replies

jkjustjoshing 5 months ago
Yes, I'd like to hear a joke about TCP/IP

iamkirkbater 5 months ago
Are you ready to hear the joke about TCP/IP?

jkjustjoshing 5 months ago
I am ready to hear the joke about TCP/IP

iamkirkbater 5 months ago
Here is a joke about TCP/IP.

iamkirkbater 5 months ago
Did you receive the joke about TCP/IP?

jkjustjoshing 5 months ago
I have received the joke about TCP/IP.

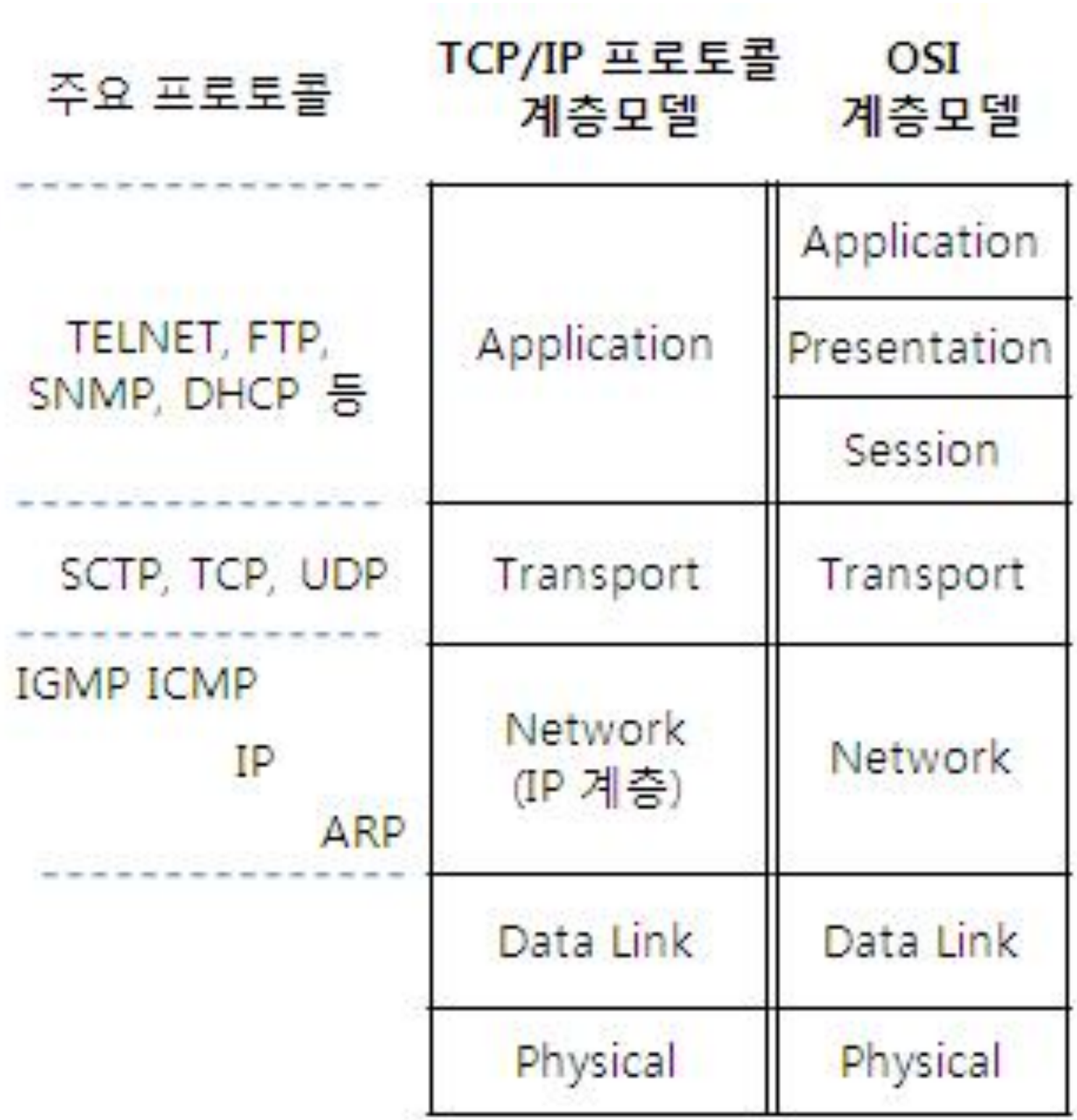
iamkirkbater 5 months ago
Excellent. You have received the joke about TCP/IP. Goodbye.

TCP/IP Protocol Suite

4개의 계층 구조로 구성 (Link 계층을 Physical 과 구분할 경우 5단계)

1. Application (응용) 계층 - HTTP, FTP, SMTP, POP, Telnet 등
2. Transport (전송) 계층 - TCP, UDP, SCTP 등
3. Network 계층 - IP, ARP 등
4. Link 계층 - 이더넷, 토큰링, PPP , Wi-Fi 등
5. Physical (물리) 계층

TCP/IP Protocol Suite



TCP/IP Protocol Suite

[4계층 Application]

데이터를 어떤 서비스를 활용해 주고 받을 것인지 정의한다. (HTTP, FTP 등)

[3계층 Transport]

데이터 송수신 방식을 결정한다.

※ TCP - 연결지향형 서비스, 메시지 전달 보장 및 순서 보장

※ UDP - 비연결형 서비스, 메시지 전달 여부나 순서 보장이 되지 않는 대신 TCP 보다 빠른 속도

[2계층 Network]

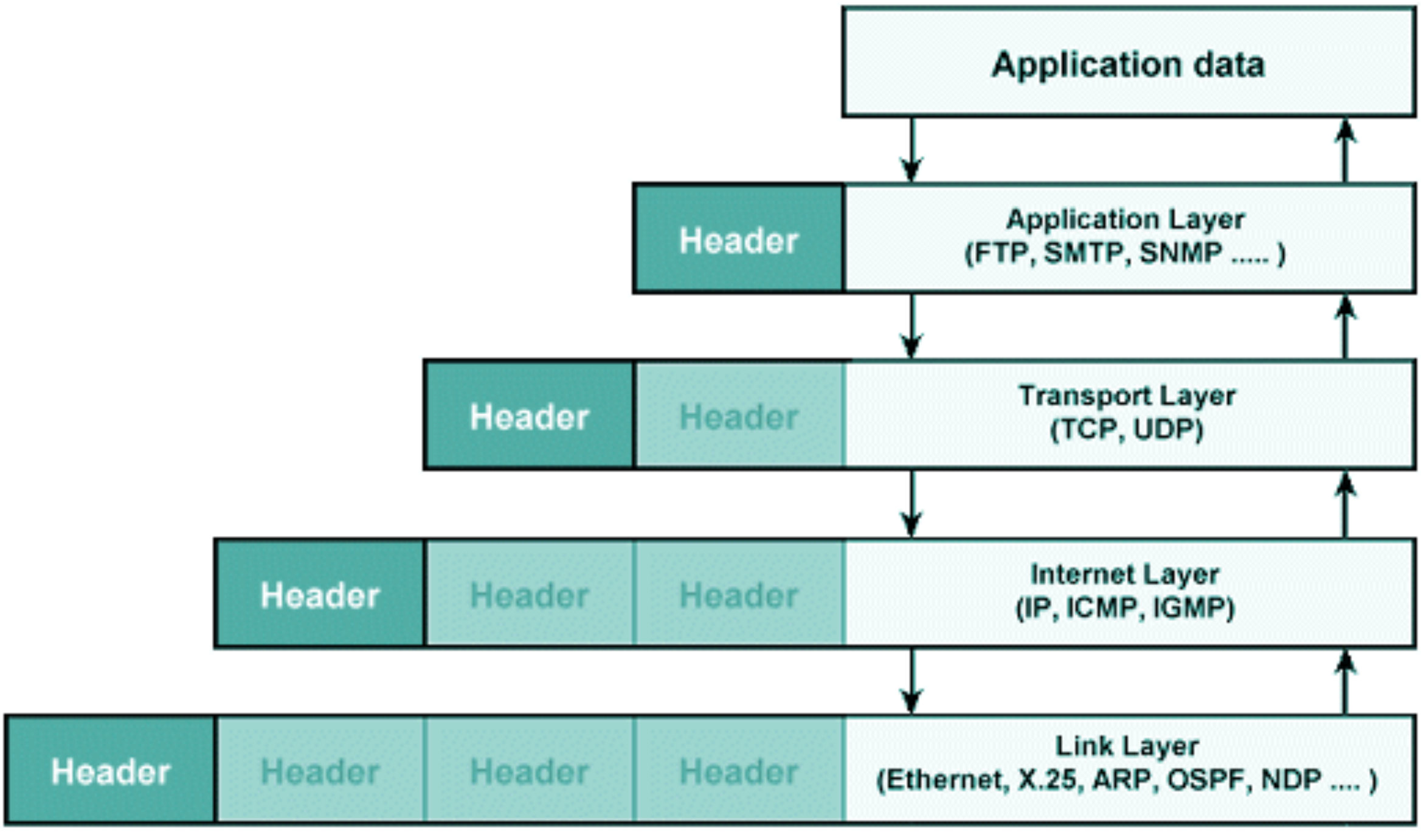
데이터가 어떤 라우터들을 거쳐 목적지로 전달될 지에 대한 전달 경로를 정의한다.

※ IP 프로토콜은 비연결지향적이며, 신뢰할 수 없는 특성을 가지므로 신뢰성 확보를 위해 TCP 가 필요

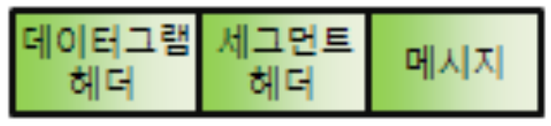
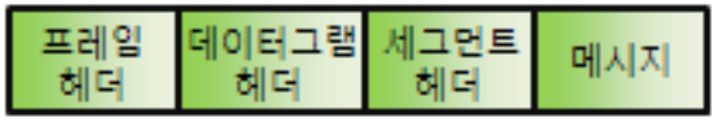
[1계층 Link (Physical 포함)]

실제 데이터 패킷 전송을 책임진다.

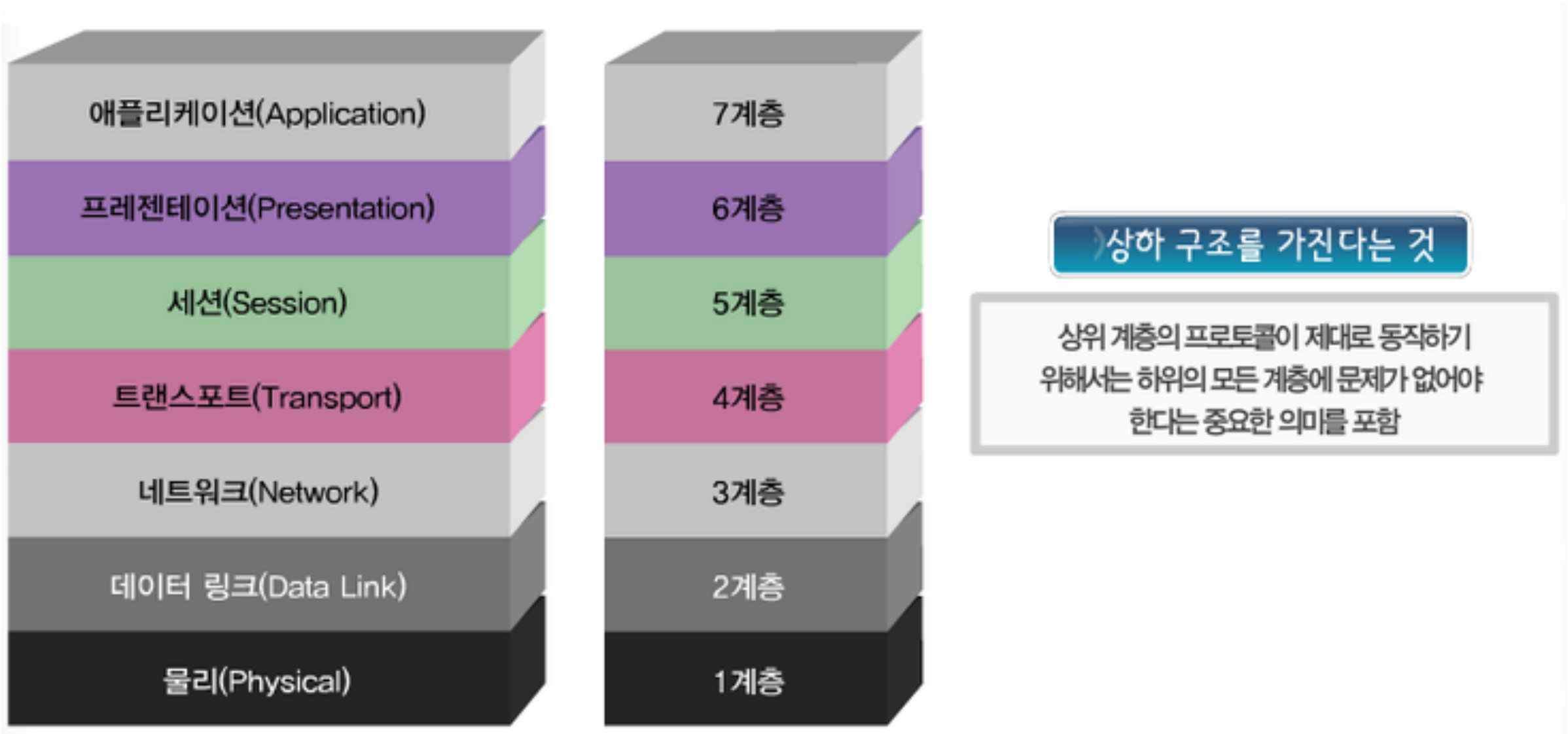
Protocol Interface



Data Packet

계층	데이터 종류	설명	형태
어플리케이션	메시지	어플리케이션 계층이 보낼 데이터입니다. 파일 전송을 목적으로 하는 FTP의 경우 파일의 일부가 메시지에 포함됩니다.	
트랜스포트	세그먼트	메시지에 세그먼트 헤더를 결합하여 세그먼트가 생성됩니다. 세그먼트 헤더는 이 계층들 사이의 통신에 필요한 부가적인 데이터들을 담습니다.	
네트워크	데이터그램	세그먼트에 데이터그램 헤더를 결합하여 데이터그램이 생성됩니다. 데이터그램 헤더도 이 계층들의 통신에 필요한 부가적인 데이터들을 담습니다. IP 프로토콜의 경우 IP 주소가 데이터그램의 헤더에 포함됩니다.	
링크	프레임	데이터그램에 프레임 헤더를 결합하여 프레임이 생성됩니다. 프레임 헤더 역시 링크 계층들의 통신에 필요한 데이터들을 갖습니다.	
물리적	비트	프레임을 구성하는 비트들입니다. 이들이 유선/무선으로 전송하기 위해 신호로 바뀝니다. 수신측에서는 이들을 다시 비트로 원복합니다.	

OSI 7 Layer


애플리케이션(Application)
프레젠테이션(Presentation)
세션(Session)
트랜스포트(Transport)
네트워크(Network)
데이터 링크(Data Link)
물리(Physical)
7계층
6계층
5계층
4계층
3계층
2계층
1계층
상하 구조를 가진다는 것
상위 계층의 프로토콜이 제대로 동작하기
위해서는 하위의 모든 계층에 문제가 없어야
한다는 중요한 의미를 포함

OSI 7 Layer


사용자 지원계층
네트워크 지원계층
전송층
7
Application
(응용층)
6
Presentation
(표현층)
5
Session
(세션층)
4
Transport
(전송층)
3
Network
(네트워크층)
2
Data link
(데이터링크층)
1
Physical
(물리층)

Fast campus

HTTP

Giftbot

Protocols

- HTTP : Hyper Text Transfer Protocol
- HTTPS : Secure Hyper Text Transfer Protocol
- FTP : File Transfer Protocol
- SFTP : Secure File Transfer Protocol
- Telnet : TEminaL NETwork
- POP3 : Post Office Protocol version 3
- SMTP : Simple Mail Transfer Protocol
- SSH : Secure Shell
- SSL : Secure Socket Layer
- SOAP : Simple Object Access Protocol

…

TCP / UDP Common Ports

21 FTP
22 SSH
23 TELNET
25 SMTP
53 DNS
80 HTTP
110 POP3
115 SFTP
135 RPC
139 NetBIOS
143 IMAP
194 IRC
443 SSL
445 SMB
1433 MSSQL
3306 MySQL
3389 Remote Desktop

List of Port

국제 인터넷 주소 관리 기구(ICANN) 에서 포트번호를 다음과 같이 세 구간으로 구분

잘 알려진 포트 (Well-Known Port)

- 0번 ~ 1023번
- ICANN 에 의해 통제되는 포트

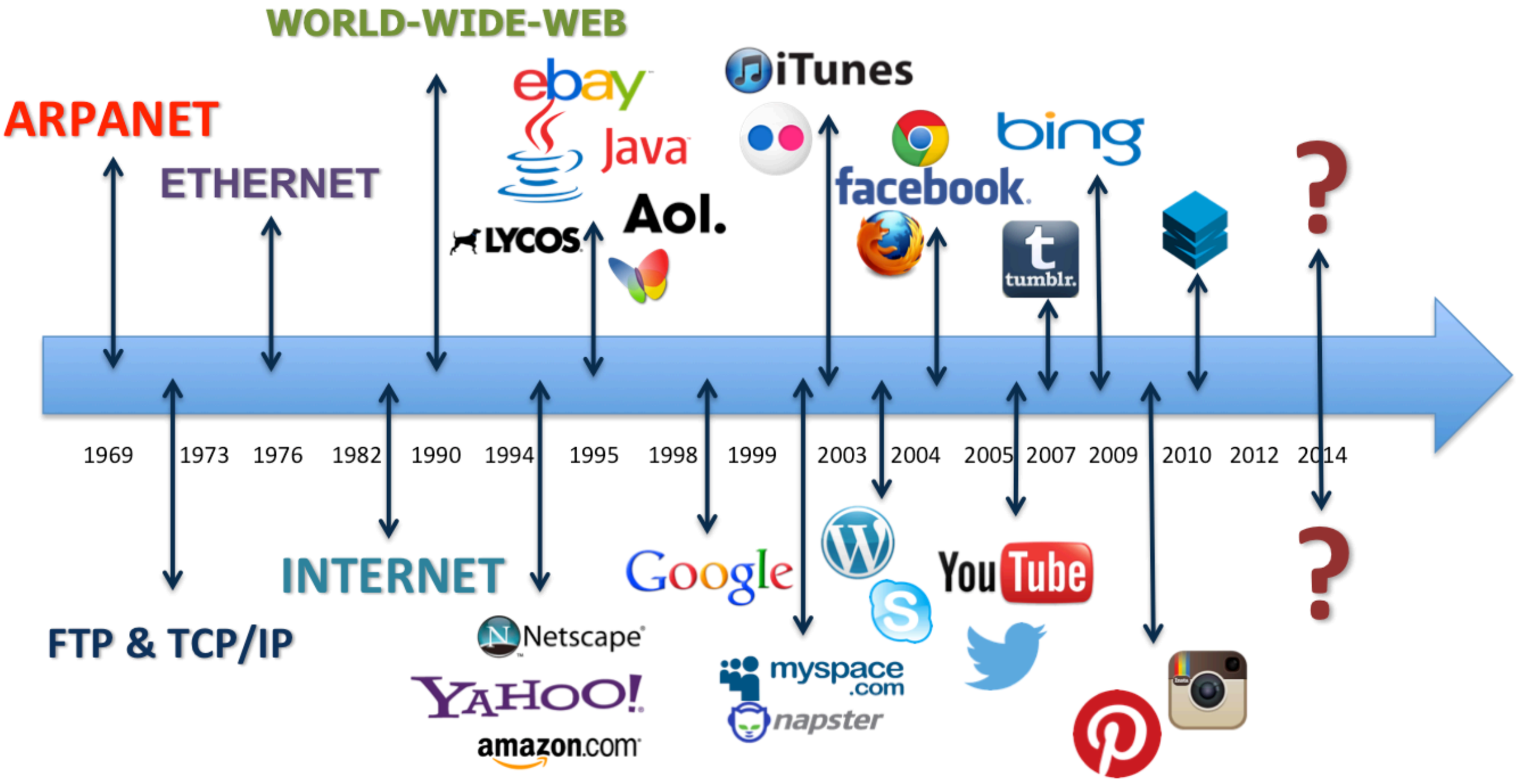
등록된 포트 (Registered Port)

- 1024번 ~ 49151번
- 중복 방지를 위해 ICANN 에 등록은 되어 있으나 미통제

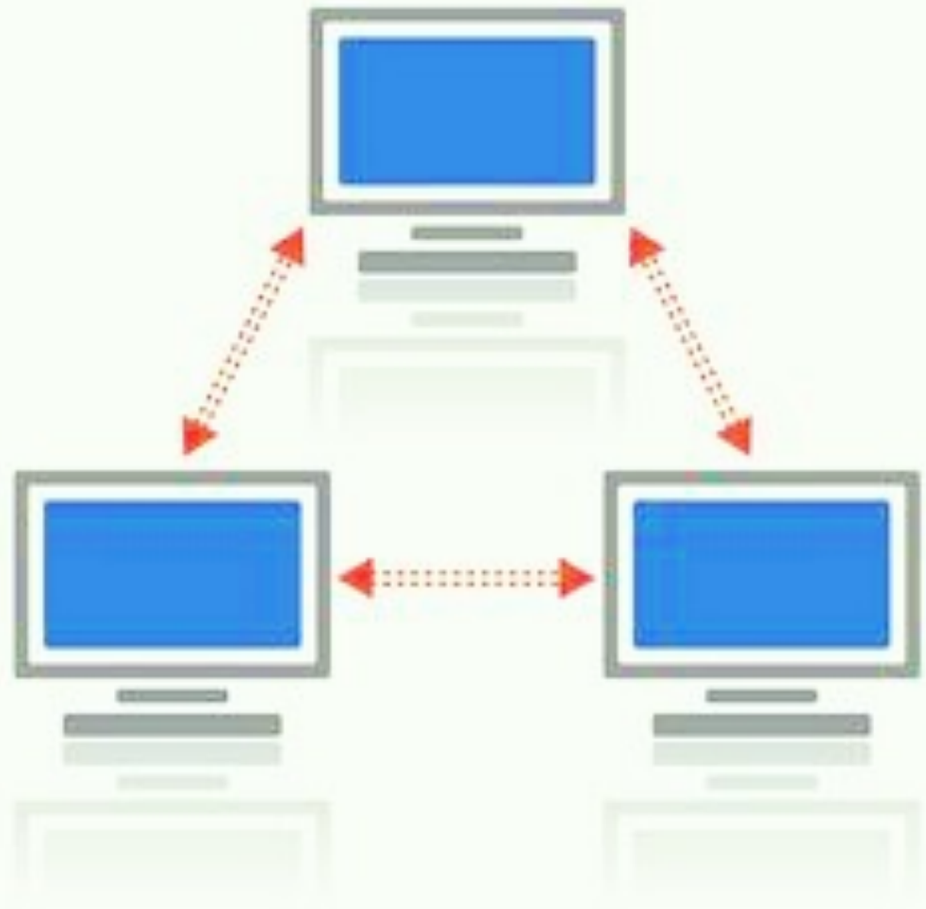
동적 포트 (Dynamic / Private Port)

- 49152번 ~ 65535번
- 어떤 프로세스에서도 사용 가능

History of Internet



Internet vs Web

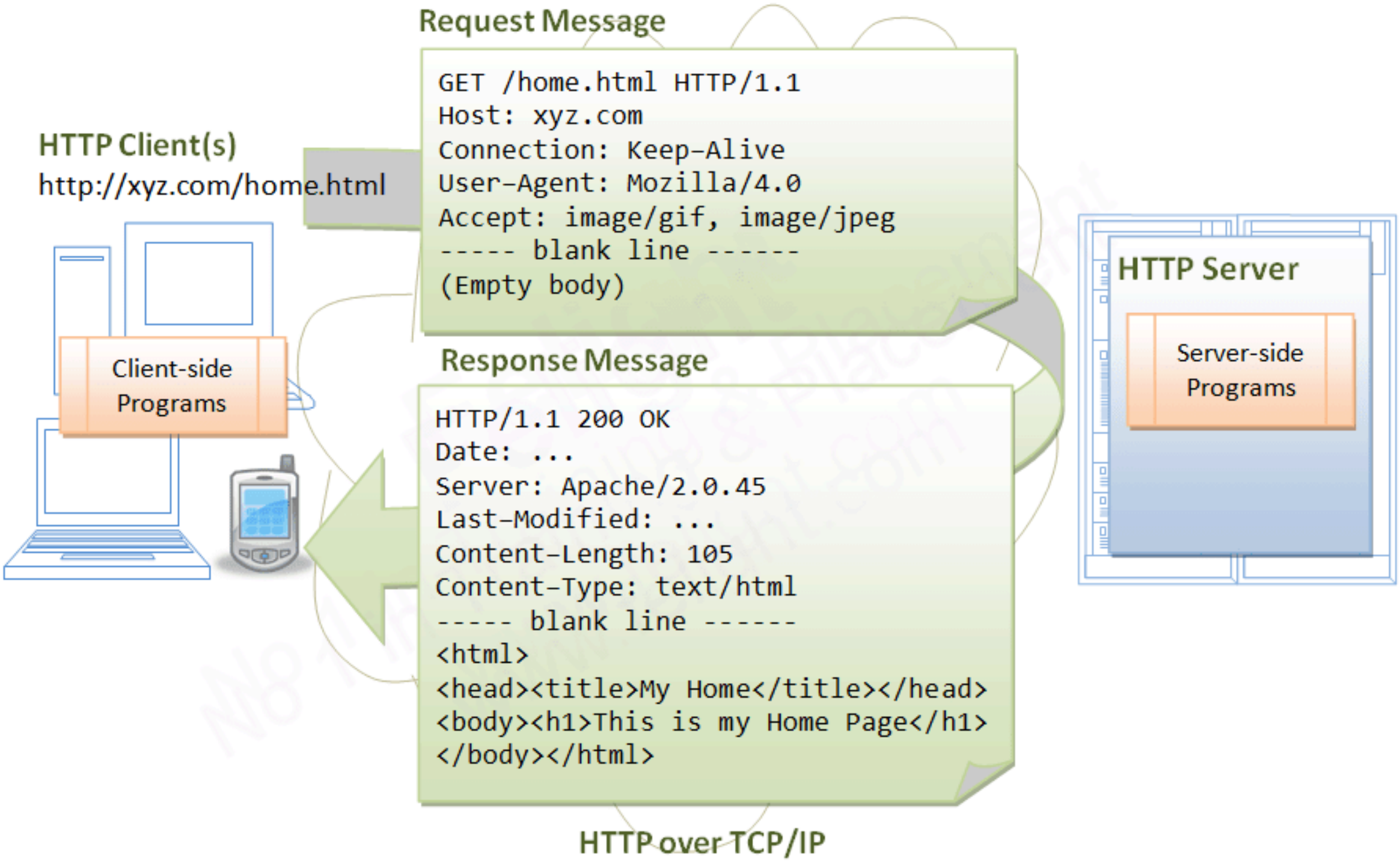


The Internet: Connecting **Computers**

The Web: Connecting **People**

HTTP (HyperText Transfer Protocol)

HTTP Request



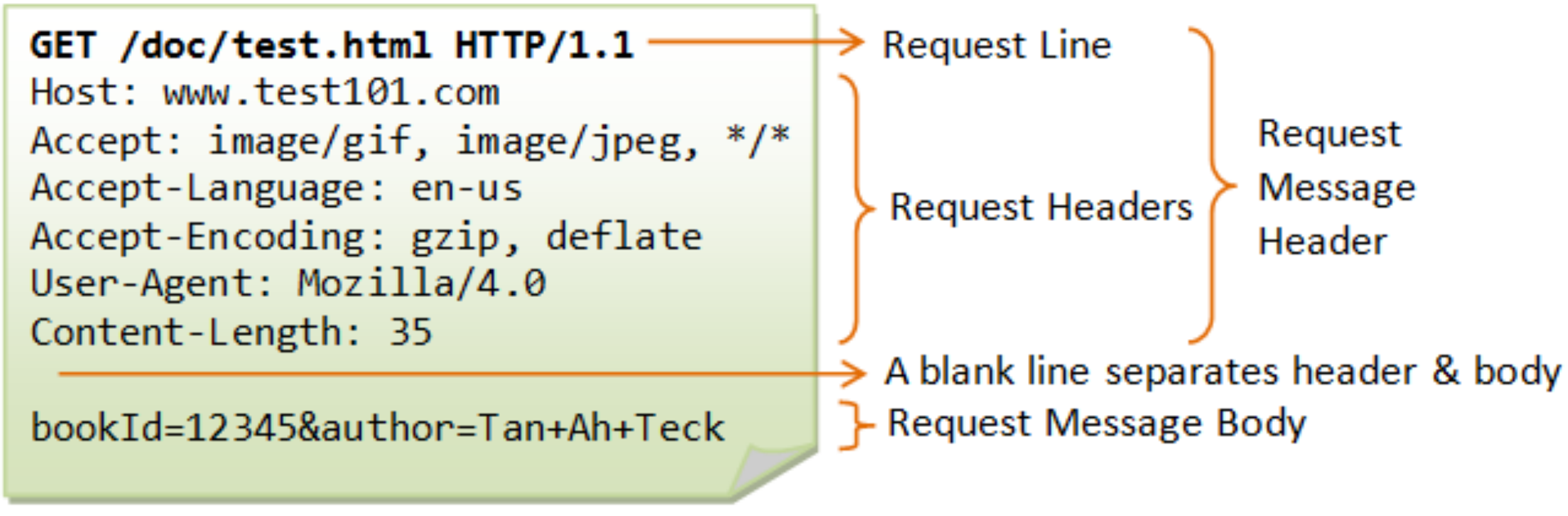
HTTP Request

Request Line

- Method : Get, Post, Delete 등

- URI : 일반적으로 URL 주소 형태 (https://www.google.com)

- HTTP 버전 (현재 대부분 1.1 또는 2.0)



HTTP Request

GET /index.html HTTP/1.1	**Request Line**
Date: Thu, 20 May 2004 21:12:55 GMT Connection: close	**General Headers**
Host: www.myfavoriteamazingsite.com From: joebloe@somewebsitesomewhere.com Accept: text/html, text/plain User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)	**Request Headers**
	Entity Headers
	Message Body

HTTP Request - Safari Network

Fast campus

56 0 8 0 검색

요소 | 네트워크 | 리소스 | 타임라인 | 디버거 | 저장 공간 | 콘솔

모든 리소스 | 도큐먼트 | bid

이름	도메인	유형	방식	체계	상태	캐시됨	크기	전송됨	시작 시간	대기 시간	실행 시간
ADTECH;v=2;cmd=bid;cors=yes;alias=168e1938...	adserver-us.adtech....	XHR	GET	HTTPS	200	아니요	1.11KB	1.23KB	52.02초	437.0밀리초	0.010밀리초
ADTECH;v=2;cmd=bid;cors=yes;alias=1586540f...	adserver-us.adtech....	XHR	GET	HTTPS	200	아니요	48B	86B	50.01초	301.8밀리초	0.459밀리초
ADTECH;v=2;cmd=bid;cors=yes;alias=14836e31...	adserver-us.adtech....	XHR	GET	HTTPS	200	아니요	1.11KB	1.22KB	2.02초	454.6밀리초	0.014밀리초
ADTECH;v=2;cmd=bid;cors=yes;alias=13768c56...	adserver-us.adtech....	XHR	GET	HTTPS	200	아니요	47B	85B	0밀리초	347.4밀리초	0.947밀리초
SetHbBidRequestProto	analytics.carambo.la	Fetch	POST	HTTPS	204	아니요	—	176B	54.02초	201.6밀리초	0.092밀리초
SetAdsRequest	analytics.carambo.la	Fetch	POST	HTTPS	204	아니요	—	176B	52.58초	217.7밀리초	0.054밀리초
SetAdsRequest	analytics.carambo.la	Fetch	POST	HTTPS	204	아니요	—	176B	52.27초	207.0밀리초	0.059밀리초
SetHbBidRequestProto	analytics.carambo.la	Fetch	POST	HTTPS	204	아니요	—	176B	52.01초	202.0밀리초	0.063밀리초
SetHbBidRequestProto	analytics.carambo.la	Fetch	POST	HTTPS	204	아니요	—	176B	4.01초	328.7밀리초	0.062밀리초
SetAdsRequest	analytics.carambo.la	Fetch	POST	HTTPS	204	아니요	—	176B	2.58초	324.8밀리초	0.064밀리초
SetAdsRequest	analytics.carambo.la	Fetch	POST	HTTPS	204	아니요	—	176B	2.28초	231.3밀리초	0.089밀리초
SetHbBidRequestProto	analytics.carambo.la	Fetch	POST	HTTPS	204	아니요	—	176B	2.01초	232.3밀리초	0.337밀리초
bid	ap.lijit.com	XHR	POST	HTTPS	200	아니요	25B	612B	52.02초	433.2밀리초	1.994밀리초
bid	ap.lijit.com	XHR	POST	HTTPS	200	아니요	26B	613B	50.01초	418.3밀리초	0.781밀리초
bid	ap.lijit.com	XHR	POST	HTTPS	200	아니요	26B	613B	2.01초	394.8밀리초	2.352밀리초
bid	ap.lijit.com	XHR	POST	HTTPS	200	아니요	26B	613B	1.719밀리초	408.0밀리초	0.888밀리초

20.00초 40.00초

리소스 목록 필터

주 프레임

리소스

▼ 유형
- MIME 유형 application/json
- 리소스 유형 XHR

▶ 위치

▼ 요청 및 응답
- 방식 POST
- 프로토콜 HTTP/1.1
- 우선 순위 중간
- 캐시됨 아니요
- 상태 OK
- 코드 200
- IP 주소 5.153.8.144:443
- 연결 ID 2
- 인코딩 52B
- 디코딩 26B
- 전송됨 613B
- 압축됨 예
- 압축 0.50×

▼ 쿼리 매개변수

이름	값
src	prebid_prebid_0.34.0

▼ 요청 데이터
- MIME 유형 text/plain
- 데이터 217B

▼ 요청 헤더

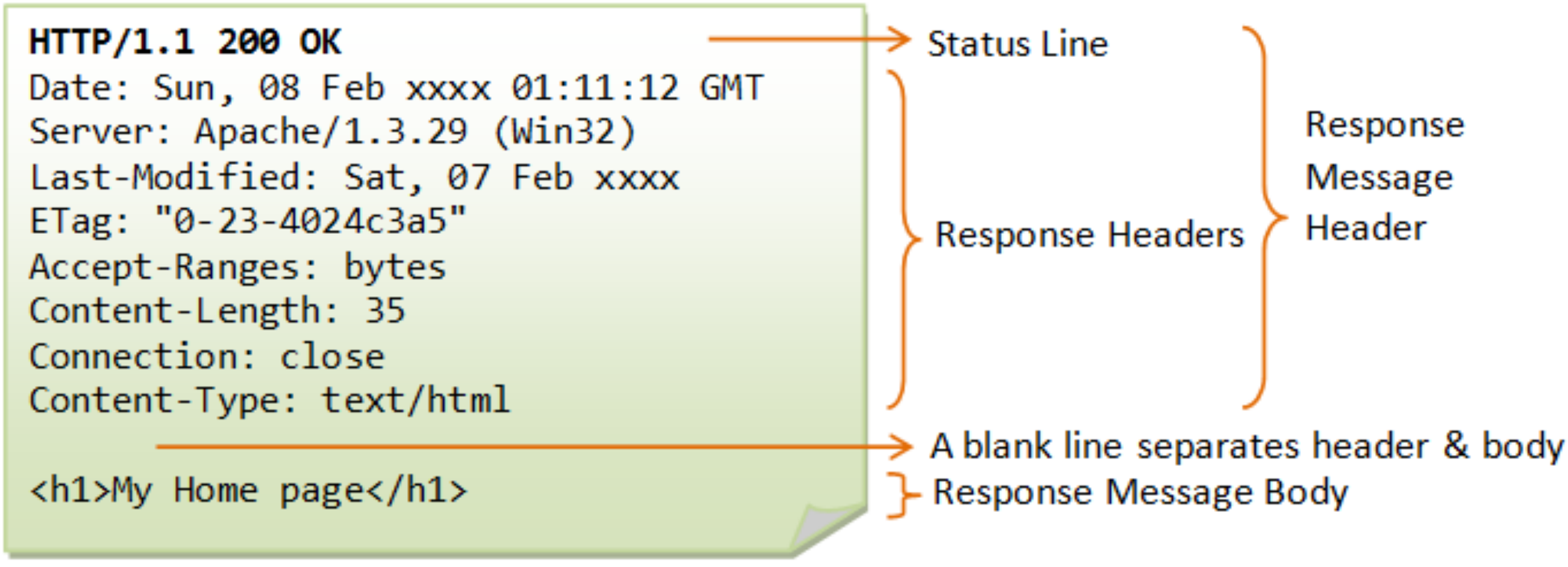
이름	값
Referer	https://www.mkyong.com/computer-tips/how-to-view-http-headers-in-google-chrome/
Content-Type	text/plain
Origin	https://www.mkyong.com
Host	ap.lijit.com
Accept	*/*

HTTP Response

Status Line

- Response Status Code

- HTTP 버전 (현재 대부분 1.1 또는 2.0)

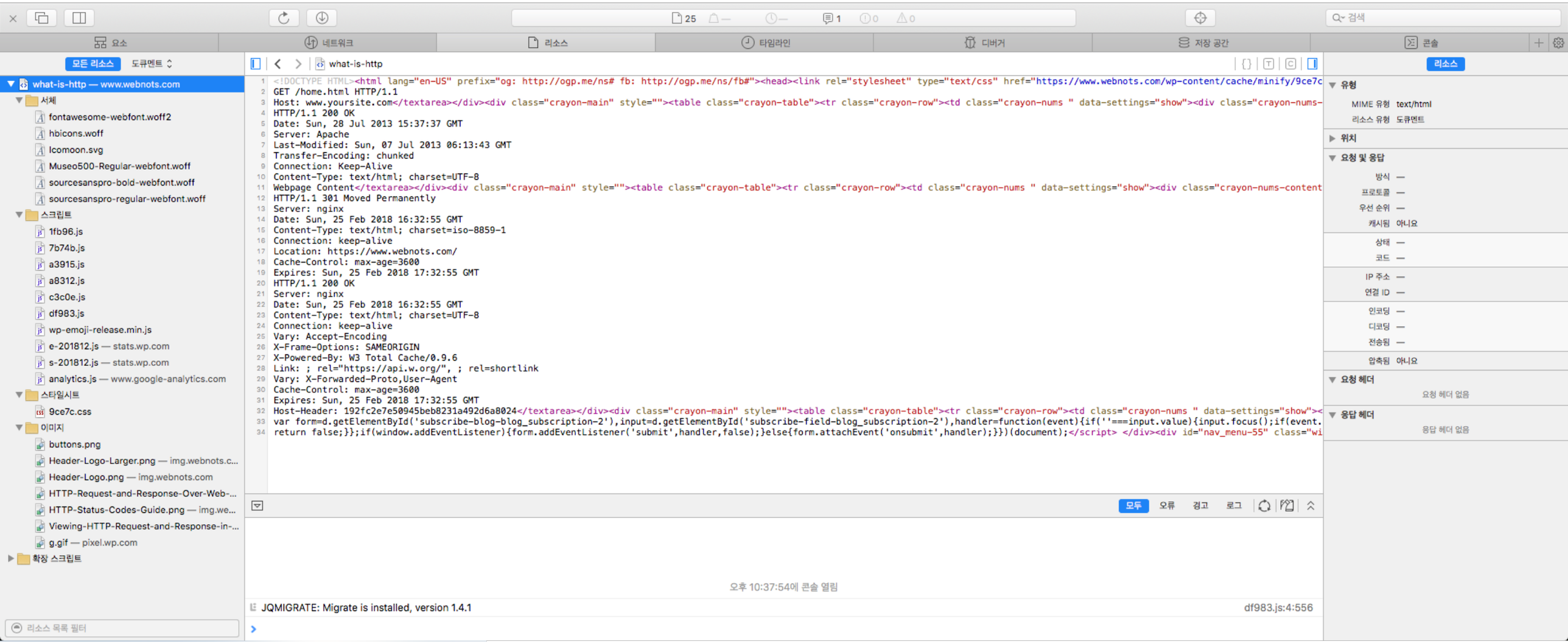


HTTP Response

HTTP/1.1 200 OK	**Status Line**
Date: Thu, 20 May 2004 21:12:58 GMT Connection: close	**General Headers**
Server: Apache/1.3.27 Accept-Ranges: bytes	**Response Headers**
Content-Type: text/html Content-Length: 170 Last-Modified: Tue, 18 May 2004 10:14:49 GMT	**Entity Headers**
<html> <head> <title>Welcome to the Amazing Site!</title> </head> <body> This site is under construction. Please come back later. Sorry! </body> </html>	**Message Body**

HTTP Response - Safari Resource

Fast campus



Response Header Example (Naver)


```
curl -I https://www.naver.com  (헤더만 보기)
```

```
HTTP/2 200
server: NWS
date: Wed, 21 Mar 2018 13:20:00 GMT
content-type: text/html; charset=UTF-8
cache-control: no-cache, no-store, must-revalidate
pragma: no-cache
p3p: CP="CAO DSP CURa ADMa TAIa PSAa OUR LAW STP PHY ONL UNI PUR FIN COM NAV INT DEM STA PRE"
x-frame-options: SAMEORIGIN
strict-transport-security: max-age=31536000; preload
referrer-policy: unsafe-url
```

Response Header Example (Naver)


```
curl -i https://www.naver.com
```

```
HTTP/2 200
server: NWS
date: Wed, 21 Mar 2018 13:20:00 GMT
content-type: text/html; charset=UTF-8
cache-control: no-cache, no-store, must-revalidate
pragma: no-cache
p3p: CP="CAO DSP CURa ADMa TAIa PSAa OUR LAW STP PHY ONL UNI PUR FIN COM NAV INT DEM STA PRE"
x-frame-options: SAMEORIGIN
strict-transport-security: max-age=31536000; preload
referrer-policy: unsafe-url
```

Response Body Example (Naver)

```
<html lang="ko" class="svgless">
<head>
<meta charset="utf-8">
<meta name="Referrer" content="origin">
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=1100">
<meta name="apple-mobile-web-app-title" content="NAVER" />
<meta name="robots" content="index,nofollow"/>
<meta name="description" content="네이버 메인에서 다양한 정보와 유용한 컨텐츠를 만나 보세요"/>
<meta property="og:title" content="네이버">
<meta property="og:url" content="http://www.naver.com/">
<meta property="og:image" content="https://s.pstatic.net/static/www/mobile/edit/2016/0705/mobile_212852414260.png">
<meta property="og:description" content="네이버 메인에서 다양한 정보와 유용한 컨텐츠를 만나 보세요"/>
<meta name="twitter:card" content="summary">
<meta name="twitter:title" content="">
<meta name="twitter:url" content="http://www.naver.com/">
<meta name="twitter:image" content="https://s.pstatic.net/static/www/mobile/edit/2016/0705/mobile_212852414260.png">
<meta name="twitter:description" content="네이버 메인에서 다양한 정보와 유용한 컨텐츠를 만나 보세요"/>


<link rel="shortcut icon" type="image/x-icon" href="/favicon.ico" />

<link rel="stylesheet" type="text/css" href="https://pm.pstatic.net/css/main_v180214a.css"/>
<link rel="stylesheet" type="text/css" href="https://pm.pstatic.net/css/webfont_v170623.css"/>
<link rel="stylesheet" type="text/css" href="https://ssl.pstatic.net/sstatic/search/pc/css/api_atcmp_170914.css"/>


<script type="text/javascript" src="https://pm.pstatic.net/js/c/nlog_v180212.js"></script>

<script type="text/javascript">
var nsc = "navertop.v3";
```

HTTP Response Status Code

- 1xx - 정보성 (Informational)
- 2xx - 성공 (Success)
- 3xx - 전환 (Redirection)
- 4xx - 클라이언트 에러 (Client errors)
- 5xx - 서버 에러 (Server erros)

Fast campus

REST API

Giftbot

REST API

REST - REpresentational State Transfer

자원을 정의하고 자원에 대한 주소를 지정하는 방법 등을 의미하는 네트워크 아키텍처 원리의 모음

웹 창시자 중 한 명인 Roy Fielding 이 기존 아키텍처의 문제점을 보완하여 웹의 장점을 최대한 활용할 수 있는

네트워크 기반의 아키텍처로 설계하여 2000년 논문에서 발표

REST 원리는 잘 따르는 시스템에 대하여 RESTFul 하다고 표현

Main Components

Rest 구성 요소

- 메서드 : 자원에 대한 행위 정의 (GET, POST 등) - 아래 예제에서 POST
- 리소스 : 자원을 정의 (URI) - 아래 예제에서 http://myweb/users/
- 메시지 : 자원에 대한 행위의 내용을 정의 (일반적으로 JSON 을 이용한 데이터)

```
HTTP POST , http://myweb/users/
{
    "users":{
        "name":"someName"
    }
}
```

Method

HTTP 의 여러 메서드 중 CRUD 에 해당하는 4가지 메서드만 사용

- POST (Create)
- GET (Read)
- PUT / PATCH (Update)
- DELETE (Delete)

POST

POST (Create)

- URI 와 관련된 자원 생성 / 작업 수행

- 다른 메서드들과 달리 URI 에 특정 자원을 지정하는 ID 가 없음

```
HTTP Post, http://myweb/users/
{
    "name":"someName",
    "address":"myAddress"
}
```

GET

GET (Read)

- 지정 URI 에 해당하는 자원을 조회하고 가져오기 위함

```
HTTP Get, http://myweb/users/someName
```

PUT / PATCH

PUT / PATCH (Update)

- 지정 URI 에 해당하는 자원 수정

- PUT : 전체 내용 수정

PATCH : 일부 항목 수정

```
HTTP PUT, http://myweb/users/someName
{
    "name":"someName",
    "address":"changedAddress"
}
```

DELETE

DELETE (Delete)

- 지정 URI 에 해당하는 자원 삭제

```
HTTP DELETE, http://myweb/users/someName
```

REST API Example

기능	HTTP 메서드	HTTP URL & BODY
모든 회원 정보 조회	HTTP GET	http://www.javastudy.co.kr/users
특정 회원 정보 조회	HTTP GET	http://www.javastudy.co.kr/users/terry
회원 정보 검색	HTTP GET	http://www.javastudy.co.kr/users?query=xxx

회원 등록	HTTP POST	http://www.javastudy.co.kr/users
		{
		"name":"terry",
		:
		}

회원 삭제	HTTP DELETE	http://www.javastudy.co.kr/users/terry

해당 회원 정보 변경	HTTP PUT	http://www.javastudy.co.kr/users/terry
		{
		"name":"terry",
		"address":"seoul"
		}