

제4장 데이터 전송 프로토콜

데이터를 전송할때 사용하는 필수 프로토콜

1. TCP, UDP, IP, Ethernet
2. ICMP, ARP

1. TCP(Transmission Control Protocol) 20byte + option

- Transmission (전송)
- Control (제어)
- 데이터를 전송하는 프로토콜인데 제어하는 기능이 들어가 있다.
- 20byte
 - 더 크게 나오는 경우가 있다. (옵션추가)
- **Layer 4 계층 프로토콜**
- **다른 시스템과 통신 수립 연결을 실시한 이후, 데이터 요청 및 응답을 실시하는 연결 지향성 특징을 가지고 있다.**

헤더 크기는 20byte 이며, 옵션이 있을 경우 더 크게 나오는 경우도 있다. Layer 4 계층 프로토콜이며, 다른 시스템과 통신 수립 연결을 실시한 이후, 데이터 요청 및 응답을 실시하는 연결 지향성 특징을 갖고 있다.

29	0.587991	62.248.74.55	61.42.166.26	TCP	62	1942-445	[SYN] Seq=
39	0.825001	120.50.139.44	172.16.6.13	TCP	60	80-55235	[RST, ACK]
88	2.193351	172.16.5.254	121.78.58.15	TCP	70	1320-30060	[PSH, ACK]
89	2.235651	121.78.58.15	172.16.5.254	TCP	60	30060-1320	[ACK] Seq=
92	2.251731	121.78.58.15	172.16.5.254	TCP	70	30060-1320	[PSH, ACK]
94	2.265431	172.16.5.254	114.111.46.227	TCP	62	1980-80	[SYN] Seq=0
95	2.271041	114.111.46.227	172.16.5.254	TCP	60	80-1980	[SYN, ACK] :
96	2.271101	172.16.5.254	114.111.46.227	TCP	54	1980-80	[ACK] Seq=1
97	2.271361	172.16.5.254	114.111.46.227	TCP	1314		[TCP segment of a r
98	2.271701	172.16.5.254	114.111.46.227	HTTP	983		GET /addAndList.nhn
99	2.277601	114.111.46.227	172.16.5.254	TCP	60	80-1980	[ACK] Seq=1

Frame 94: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface 0
Ethernet II, Src: RealtekS_14:62:ba (00:e0:4c:14:62:ba), Dst: Cisco_00:0c:2c:00:00:00
Internet Protocol Version 4, Src: 172.16.5.254 (172.16.5.254), Dst: 114.111.46.227 (114.111.46.227)
Transmission Control Protocol, Src Port: 1980 (1980), Dst Port: 80 (80)

Source Port: 1980 (1980)

Destination Port: 80 (80)

[Stream index: 3]

[TCP Segment Len: 0]

Sequence number: 0 (relative sequence number)

Acknowledgment number: 0

Header Length: 28 bytes

+ 0000 0000 0010 = Flags: 0x002 (SYN)

Window size value: 65535

[Calculated window size: 65535]

+ Checksum: 0xcc2a [validation disabled]

Urgent pointer: 0

+ Options: (8 bytes), Maximum segment size, No-Operation (NOP), No-Op

1) 3-Way 핸드 셰이킹 동작 (3번 데이터가 옮겨진다)

1) '3-Way 핸드 셰이킹' 동작 실시

클라이언트와 서버는 다음과 같이 '3-Way 핸드 셰이킹' 동작을 실시하여 TCP 연결을 성립하고 데이터 요청 및 응답을 실시한다.

94	2.265	172.16.5.254	114.111.46.227	TCP	62	1980-80	[SYN]	Seq=0	Win=65535	
95	2.271	114.111.46.227	172.16.5.254	TCP	60	80-1980	[SYN, ACK]	Seq=0	Ack=1	
96	2.271	172.16.5.254	114.111.46.227	TCP	54	1980-80	[ACK]	Seq=1	Ack=1	Win=65535
97	2.271	172.16.5.254	114.111.46.227	TCP	1314		[TCP segment of a reassembled PDU]			
98	2.271	172.16.5.254	114.111.46.227	HTTP	983		GET /addAndList.nhn?r=linkedMei			
99	2.277	114.111.46.227	172.16.5.254	TCP	60	80-1980	[ACK]	Seq=1	Ack=1261	Win=65535
100	2.277	114.111.46.227	172.16.5.254	TCP	60	80-1980	[ACK]	Seq=1	Ack=2190	Win=65535
101	2.279	114.111.46.227	172.16.5.254	HTTP	902		HTTP/1.1 200 OK (text/plain)			

		클라이언트		서버	
		172.16.5.254:1980		114.111.46.227:80	

94	① Syn(Seq=0) ->	
95		<- Syn, Ack(Seq=0, Ack=1) ②
96	③ Ack(Seq=1, Ack=1) ->	
	----- 통신 수립 완료(ESTABLISHED) -----	
	데이터/서비스 요청 ->	
98	④ HTTP GET	
101		<- 데이터/서비스 응답
		HTTP/1.1 200 OK ⑤

[클라이언트]

[서버]

TCP를 통해서 연결을 하려면 TCP 성립을 해야한다.

1. Sync 라는 데이터를 보낸다 (서버한테 통신한다고 알려줌) ->
2. 서버가 Sync를 받았다면 서버는 Sync + Ack 를 보낸다 <-
3. Sync + Ack를 Client 가 받은 후 서버가 잘 연결된걸 확인한다
4. Client는 Ack를 다시 서버로 보낸다. ->

이 과정을 거치면 TCP 연결 성립이 된다.

데이터 요청 ->

<-데이터 응답

ex) 3-way hand shaking

92	2.25173:121.78.58.15	172.16.5.254	TCP	70	30060-1320	[PSH, ACK]	Seq=1	Ack=17	
94	2.26543(172.16.5.254	114.111.46.227	TCP	62	1980-80	[SYN]	Seq=0	Win=65535	Len=0
95	2.27104(114.111.46.227	172.16.5.254	TCP	60	80-1980	[SYN, ACK]	Seq=0	Ack=1	Win=65535
96	2.27110:172.16.5.254	114.111.46.227	TCP	54	1980-80	[ACK]	Seq=1	Ack=1	Win=65535
97	2.27136:172.16.5.254	114.111.46.227	TCP	1314		[TCP segment of a reassembled PDU]			

```

Frame 94: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface 0
Ethernet II, Src: Realtek_14:62:ba (00:e0:4c:14:62:ba), Dst: Cisco_31:81:b1 (00:13:02:31:81:b1)
Internet Protocol Version 4, Src: 172.16.5.254 (172.16.5.254), Dst: 114.111.46.227 (114.111.46.227)
Transmission Control Protocol, Src Port: 1980 (1980), Dst Port: 80 (80), Seq: 0, Len: 0
Source Port: 1980 (1980)
Destination Port: 80 (80)
[Stream index: 3]
[TCP Segment Len: 0]
Sequence number: 0 (relative sequence number)
Acknowledgment number: 0
Header Length: 28 bytes
... 0000 0000 0010 = Flags: 0x002 (SYN)
000. .... = Reserved: Not set
...0 .... = Nonce: Not set
... 0... = Congestion Window Reduced (CWR): Not set
... .0.. = ECN-Echo: Not set
... ..0. = Urgent: Not set
... ...0 = Acknowledgment: Not set
... ....0... = Push: Not set
... .....0.. = Reset: Not set
... ..1. = Syn: Set
... ...0 = Fin: Not set
Window size value: 65535
[Calculated window size: 65535]

```

Sync 설정됨 (연결됨)

```

... 0000 0001 0010 = Flags: 0x012 (SYN, ACK)
000. .... = Reserved: Not set
...0 .... = Nonce: Not set
... 0... = Congestion Window Reduced (CWR): Not set
... .0.. = ECN-Echo: Not set
... ..0. = Urgent: Not set
... ...1 = Acknowledgment: Set
... ....0... = Push: Not set
... .....0.. = Reset: Not set
... ..1. = Syn: Set
... ...0 = Fin: Not set

```

Sync, Ack (잘 받았다는 의미로 보낸다) 설정

```

... 0000 0001 0000 = Flags: 0x010 (ACK)
000. .... = Reserved: Not set
...0 .... = Nonce: Not set
... 0... = Congestion Window Reduced (CWR): Not set
... .0.. = ECN-Echo: Not set
... ..0. = Urgent: Not set
... ...1 = Acknowledgment: Set
... ....0... = Push: Not set
... .....0.. = Reset: Not set
... ..0. = Syn: Not set
... ...0 = Fin: Not set

```

다시 Ack만 설정

2) TCP Control Flag(6bit)

다음은 TCP 플래그 상태에 따른 TCP 주요 동작이다.

0000 0000 0010 = Flags: 0x002 (SYN)	urg	(1....)	: 긴급한 데이터 표기
000. = Reserved: Not set			
...0 = Nonce: Not set	ack	(.1....)	: 확인 응답/승인
...0 = Congestion Window Reduced (CWR): Not set			
...0 = ECN-Echo: Not set	psh	(..1..)	: 상위 프로세스 처리
...0 = Urgent: Not set	rst	(...1.)	: 강제 종료
...0 = Acknowledgment: Not set	syn	(....1.)	: 통신 개시
...0 = Push: Not set			
...0 = Reset: Not set	fin	(.....1)	: 정상적인 종료
...1 = Syn: Set			
...0 = Fin: Not set			

Fin, Reset은 종료이다.

Fin: 상대방의 상태를 확인 후 종료 (정상종료)

Reset: 상대방의 상태를 확인 안하고 강제종료

- TCP 연결 된 상태에서, 연결이 없는 경우에도 가능
- ex)

클라이언트

웹-서버(TCP 80포트 오픈)

SA 50001

SA 22

DA 22

DA 50000001

Syn ----->

<-----RST(Reset) + Ack

리셋을 받은 경우 포트번호 22번이 닫혀 있기 때문이다.

공격자 입장에서는 22번이 닫혀있다는걸 알 수 있다.

클라이언트

웹-서버(TCP 80포트 오픈)

SA 50001

SA 80

DA 80

DA 50000001

Syn ----->

<-----Syn + Ack

Syn를 받은경우 80 번 포트가 열려있다는걸 알 수 있다.

Push

- 데이터를 모아놨다가 출력한다
- 모아놨다가 출력하는걸 버퍼링이라고 한다.
- **1로 설정되어있으면 버퍼링하지 말고 바로 출력시켜라 라는 뜻이다.**
- **바로바로 처리해라 라는 뜻으로 해석**

```

000. .... = Reserved: Not set
...0 .... = Nonce: Not set
.... 0... = Congestion Window Reduced (CWR): Not set
.... .0.. = ECN-Echo: Not set
.... ..0. = Urgent: Not set
.... ...1 = Acknowledgment: Set
.... ....0... = Push: Not set
.... .... .0.. = Reset: Not set
.... .... ..0. = Syn: Not set
.... .... ...0 = Fin: Not set

```

Ack

- 뭔가를 잘 받았다는 의미로 보낸다.

```

.... 0000 0001 0000 = Flags: 0x010 (ACK)
000. .... = Reserved: Not set
...0 .... = Nonce: Not set
.... 0... = Congestion Window Reduced (CWR): Not set
.... .0.. = ECN-Echo: Not set
.... ..0. = Urgent: Not set
.... ...1 = Acknowledgment: Set
.... ....0... = Push: Not set
.... .... .0.. = Reset: Not set
.... .... ..0. = Syn: Not set
.... .... ...0 = Fin: Not set

```

Urgent

- 서비스를 먼저 출력해주는 서비스
- 긴급한 데이터
- 하지만 설정하려면 소프트웨어 상으로 설정해야한다 하지만 그냥 놔두면 아무것도 안한다.

```

000. .... = Reserved: Not set
...0 .... = Nonce: Not set
.... 0... = Congestion Window Reduced (CWR): Not set
.... .0.. = ECN-Echo: Not set
.... ..0. = Urgent: Not set
.... ...1 = Acknowledgment: Set
.... ....0... = Push: Not set
.... .... .0.. = Reset: Not set
.... .... ..0. = Syn: Not set
.... .... ...0 = Fin: Not set

```

다음은 '3-Way 핸드 셰이킹' 과정에서 사용하는 TCP 플래그의 10 진수, 16 진수 값이다.

URG	ACK	PSH	RST	SYN	FIN	10 진수	16 진수	tcp.flag
2^5	2^4	2^3	2^2	2^1	2^0			
32	16	8	4	2	1			
0	0	0	0	1	0	2	0x02	syn
0	1	0	0	1	0	18	0x12	syn+ack
0	1	0	0	0	0	16	0x10	ack

계산기 >> 프로그래머 >> 10진수 변환

3) Fin 플래그를 이용한 TCP 정상 종료



- ① 클라이언트가 TCP 종료를 하기 위해서 Fin 세그먼트를 전송한다..
- ② Fin 세그먼트를 전송한 클라이언트의 TCP 연결 상태는 'FIN_WAIT_1'이 된다.
- ③ Fin 세그먼트를 수신한 서버는 TCP 연결 상태를 'CLOSE_WAIT'로 전환한다.
- ④ 그리고 서버는 클라이언트로 Ack 세그먼트를 전송한다.
- ⑤ Ack 세그먼트를 수신한 클라이언트는 TCP 연결 상태를 'FIN_WAIT_2'로 전환한다.
- ⑥ 서버는 클라이언트로 Fin 세그먼트를 전송한다.
- ⑦ Fin 세그먼트를 전송한 서버의 TCP 연결 상태는 'LAST_ACK'가 된다.
- ⑧ Fin 세그먼트를 수신한 클라이언트는 TCP 연결 상태를 'TIME_WAIT'로 전환한다.
- ⑨ 그리고 클라이언트는 서버로 Ack 세그먼트를 전송한다.
- ⑩ Ack 세그먼트를 수신한 서버는 TCP 연결 상태를 'CLOSED'로 전환한다.
- ⑪ 클라이언트가 전송한 Ack 를 서버가 수신할 수 있도록 일정 시간 대기한다.
(만약, 대기 시간이 없다면, 서버가 Ack 를 못받을 경우 FIN 를 재전송하기 때문이다.)
- ⑫ 대기 시간이 경과되면 클라이언트의 TCP 연결 상태가 'CLOSED'로 전환되면서 TCP 연결이 종료된다

- 서버는 Client가 Finish를 받을때까지 그리고 서버가 Ack를 받을때까지 Fin을 보낸다.
- 그리고 서버가 Ack를 받으면 바로 닫힌다.
- Client가 서버에서 또 정보를 받을수도 있기때문에 240초라는 대시시간을 받는다.
- 240초가 지난 후에 컴퓨터를 종료한다.

4) 데이터 스트림 서비스

데이터를 세그먼트 단위로 생성하여 전송 및 수신 처리하는 기능이다. 이를 통해서 전송률과 처리율을 효율적으로 운영할 수 있다. 다음과 같이 순서 번호와 확인 번호를 이용한 'Stop & Wait', 'Sliding Window'라는 흐름 제어 기능이 필요하며, 현재 TCP에서는 'Sliding Window' 기법을 사용하고 있다.

- 큰 데이터를 통째로 들고오는것보다 분해해서 들고오면 더 효율적이다.
- 분할된 데이터 단위를 Segment 라고 부른다.
- 수신측에서는 분할된 데이터를 받으면 다시 조립해야한다.
 - 다시 원래대로 조립하려면 번호가 필요하다
 - Sequence Number (순서 번호)
 - 데이터를 보낼때 쓰는 번호
 - Ark Number (아까랑 다른 에크)
 - Segment 몇번을 잘 받았음을 알리는 번호이다. (옆면 윗면 등)
 - 데이터를 받을때 쓰는 번호
- **흐름 제어 기능**
 1. **Stop & Wait**
 - a. 2번 segment를 받으려면 1번 segment를 받아야하다.
 - b. 순서대로 받아야한다.
 - c. 수신하는 양이 많아지는게 단점이다.

유형	내용
Stop & Wait	- 송신한 세그먼트에 대한 Ack 를 수신해야지만, 그 다음 세그먼트를 전송한다. - 다음 세그먼트 송신 처리에 대한 지연이 발생하고 수신하는 Ack 양이 많다.
	① 송신측에서 1 번 세그먼트 전송 [1] ->
	② 1 번 세그먼트에 대한 Ack 를 수신해야지만, 송신측에서 2 번 세그먼트 전송 <div style="text-align: right;"><- ack</div> [2] ->
	③ 2 번 세그먼트에 대한 Ack 를 수신해야지만, 송신측에서 3 번 세그먼트 전송 <div style="text-align: right;"><- ack</div> [3] ->

ex)

나 는 흥 길 동 입 니 다

나
는
흥
길
동
입
니
다

2. Sliding Window

- a. 데이터 수 통신을 하려면 window 크기를 협의한다.
- b. 윈도우를 계속 옆으로 밀어서 슬라이딩이다.
 - i. 내가 받을 수 있는 segment 크기이다.
 - ii. ex)

A: 20 B:100

B가 A한테 보낼려면 20으로 보내야한다.

수신측 기준으로 맞춘다.

Sliding Window	<ul style="list-style-type: none"> - 수신측 윈도우 크기에 맞게 송신측에서 세그먼트양 조정하여 전송한다. - 세그먼트 송신 지연 발생과 Ack 양을 최소화한다. <p>① 송신측 윈도우 크기가 '5'라면, 1~5 번 세그먼트 전송 가능</p> <p>[1][2][3][4][5] 6 7 8 9</p> <p>② 그러나 수신측 윈도우 크기가 '2'라면, 송신측은 1~2 번 세그먼트만 전송 실시</p> <p>1 2 [3][4][5] -></p> <p>③ 수신측으로부터 1~2 번 세그먼트에 대한 Ack 를 받으면, 송신측은 자신의 윈도우 크기만큼 슬라이딩 윈도우 실시</p> <p style="text-align: right;"><- ack</p> <p>1 2 [3][4][5][6][7]</p>
----------------	---

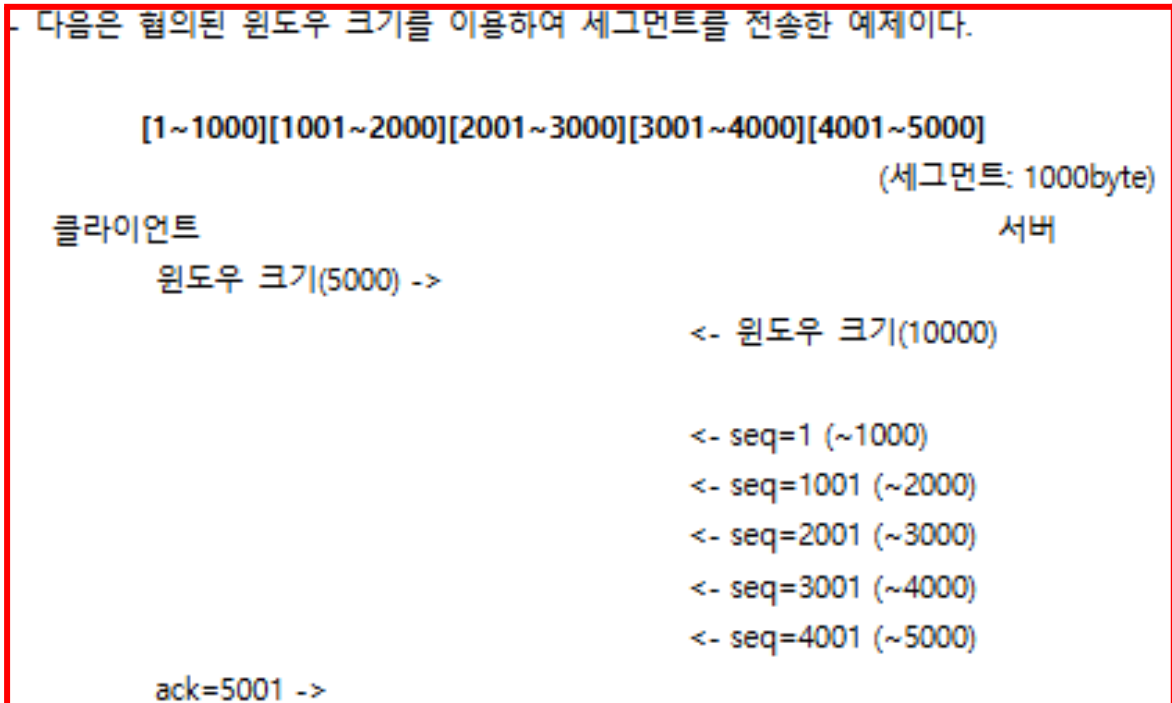
- c. segment의 크기를 조절하여 한번에 여러개를 보내고 Ack를 받을 수 있다.
 - i. 수신측의 window를 고려하여 보낸다.
 - ii. 조금 더 효율적이다.
 - iii. ex)

나 는 흥 길 동 입 니 다

나는
홍길동

입니다

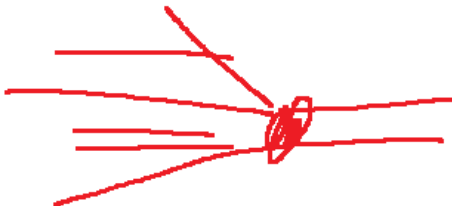
ex) Sliding Window method:



5) 혼잡 제어 기능

혼잡 발생시 전송률을 최소화하여 혼잡을 줄이는 기능이다. TCP에만 있는 필드이기 때문에 잘 사용하지 않으며, 라우터에서 QoS 정책을 별도로 구성하여 적용하고 있다.

- 들어오는 속도보다 나가는 속도가 느린경우 (트래픽 혼잡)



- TCP에만 있는 필드이다. (잘 사용하지 않는다)
- 라우터에서 QoS 정책을 별도로 구성 (모든 패킷들을 대상)

6) 오류 검사

- 수신한 세그먼트에 대한 손상 여부 판단하여, 세그먼트를 드랍하는 기능이다.

7) 재전송 기능 (Ack를 못받으면 재전송한다)

송신한 세그먼트에 대한 Ack를 재전송 시간 초과 타이머(RTO) 안에 수신하지 못하면, 해당 세그먼트가 손상되었거나, 손실된 것으로 간주하여 세그먼트를 재전송한다. RTO 타이머는 가변적인 시간을 갖고 있다.

8) Window Size

처리할 수 있는 세그먼트양을 표기하는 필드이다. 송신측에게 자신의 윈도우 사이즈를 알려주면, 송신측에서 그만큼의 세그먼트를 한번에 전송하고 수신측이 다 처리했는지 확인 후에 다음 세그먼트를 전송한다.

윈도우 사이즈는 가변적이기 때문에 상황에 따라서 증가되거나 감소된다.

- 크기는 가변적이다 (상황에 따라 바뀐다)

9) TCP를 사용하는 서비스

- HTTP(80), HTTPS/SSL(443), Telnet(23), SSH(22), FTP(21), FTP-Data(20), SMTP(25), POP3(110), Mysql(3306)
- Tomcat(8080), NFS(2049), RPC(111), SMB(139,445)

클라이언트

서버

Syn(seq 0) ----->

<-----Syn + Ack(seq 0, ack 1)

Ack(seq 1, ack 1) ----->

|

2. UDP (User Datagram Protocol)

헤더 크기는 8byte 이다. Layer 4 계층 프로토콜이며, 상대방과 연결 과정 없이 데이터 요청 및 응답을 바로 실시하는 비연결 지향성 특징을 갖고 있다.

- TCP와 마찬가지로 Layer 4 계층 프로토콜이다.
- 바로 데이터 요청과 응답을 실시한다. (비연결 지향성 특징)

1) 비연결 지향성 프로토콜

UDP 는 연결성이 없기 때문에 데이터 요청 및 응답을 바로 실시한다.

55	1.374	172.16.5.254	168.126.63.1	DNS	73 Standard query 0x3968
60	1.415	168.126.63.1	172.16.5.254	DNS	221 Standard query response
클라이언트			서버		
172.16.5.254:53258			168.126.63.1:53		
데이터/서비스 요청 ->					
55	① DNS(query)				
				<- 데이터/서비스 응답	
60	DNS(response) ②				

TCP 에서 제공하는 다음과 같은 기능은 지원하지 않는다. 단, 오류 검사 기능은 지원한다.

'3-Way 핸드 셰이킹' 동작
 데이터 스트림 서비스
 흐름 제어 기능
 혼잡 제어 기능
 재전송 기능
 Window Size

2) UDP를 사용하는 서비스

- DNS(53), TFTP(69), DHCP Server(67), DHCP Client(68), SNMP(161), NTP(123), NMB(137,138), Syslog(514)

3) UDP 헤더 내용 (8byte)

User Datagram Protocol, Src Port: 60669 (60669), Dst Port: 53 (53)
Source Port: 60669 (60669)
Destination Port: 53 (53)
Length: 39
*Checksum: 0x6760 [validation disabled]
[Stream index: 22]

TCP vs UDP

TCP	UDP
신뢰성	>
신속성	<

3. IP (Internet Protocol)

헤더 크기는 20byte 이다. Layer 3 계층 프로토콜이며, 비연결 지향성 특징을 갖고 있다. IP 프로토콜은 로컬 환경에서 리모트 환경으로 데이터 전송을 하기 위해서 사용한다. IP 헤더 내용은 다음과 같다.

- 20byte
- Layer 3 계층 프로토콜
- 비연결성
- 로컬 환경에서 리모트 환경으로 데이터 전송을 하기 위해서 사용한다.

```
Internet Protocol Version 4, Src: 172.16.5.254 (172.16.5.254), Dst: 168.126.63.1 (168.126.63.1)
Version: 4
Header Length: 20 bytes
▣ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable T
  0000 00.. = Differentiated Services Codepoint: Default (0x00)
  .... 00 = Explicit Congestion Notification: Not-ECT (Not ECN-Capable Transport) (0x00)
Total Length: 59
Identification: 0xc9a5 ($1621)
▣ Flags: 0x00
  0... .... = Reserved bit: Not set
  .0.. .... = Don't fragment: Not set
  ..0. .... = More fragments: Not set
Fragment offset: 0
Time to live: 128
Protocol: UDP (17)
▣ Header checksum: 0xd77e [validation disabled]
Source: 172.16.5.254 (172.16.5.254)
Destination: 168.126.63.1 (168.126.63.1)
```

항목	내용
Version	IP 버전 표기(IPv4, IPv6)
Header Length	IP 헤더 크기
Differentiated Services Field	QoS 정책 구현시 사용하는 필드
Identification	패킷 식별자
Flags	IP Fragments 가 실시된 패킷을 알리는 필드(More fragments 가 '1'로 설정됨)
Fragment offset	IP Fragments 가 실시된 누적된 패킷 크기
Time to live(0~255)	패킷이 네트워크상에 전송될 수 있는 시간(시간 단위는 라우터이다.)
Protocol	상위 프로토콜 정보
Header checksum	헤더 오류 검사(불필요한 필드이기 때문에 IPv6 프로토콜에서는 삭제함)
Source	출발지 IP 주소
Destination	목적지 IP 주소

3bit

IP Precedence

000	0	
001	1	
010	2	
011	3	<-A
100	4	
101	5	<-B
110	6	
111	7	

6bit (DSCP)

```
Frame 55: 73 bytes on wire (584 bits), 73 bytes captured (584 bits) on interface 0
Ethernet II, Src: RealtekS_14:62:ba (00:e0:4c:14:62:ba), Dst: Cisco_31:81:b1 (00:13:60:31:81:b1)
  Destination: Cisco_31:81:b1 (00:13:60:31:81:b1)
  Source: RealtekS_14:62:ba (00:e0:4c:14:62:ba)
  Type: IP (0x0800)
Internet Protocol Version 4, Src: 172.16.5.254 (172.16.5.254), Dst: 168.126.63.1 (168.126.63.1)
  Version: 4
  Header Length: 20 bytes
  Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    0000 00.. = Differentiated Services Codepoint: Default (0x00)
    .... ..00 = Explicit Congestion Notification: Not-ECT (Not ECN-Capable Transport)
```

MTU 크기 (기본값 1500byte)

4000 byte

	id	more fragment	fragment offset
1500byte	17	1 (처음분할)	0
1500byte	17	1	1500byte (누적)
1000byte	1y	0 (분할x)	3000byte (누적)

Flags: 0x00

```
0... .... = Reserved bit: Not set
.0.. .... = Don't fragment: Not set
..0. .... = More fragments: Not set
```

IP의 Time to live (TTL) (0~255) 기능

▣ Flags: 0x00

0... .. = Reserved bit: Not set

.0... .. = Don't fragment: Not set

..0. = More fragments: Not set

Fragment offset: 0

Time to live: 128

- 네트워크가 바뀔때마다 차감된다.
- Windows(162) → R1(161) → R2(160) → R3(159) → R4 → Linux
- 패킷이 네트워크상에 전송될 수 있는 시간(시간 단위는 라우터이다)
- 원래 목적은 네트워크 상에서 루프가 발생했을때 방지하기 위해서이다.

```
Packet Tracer PC Command Line 1.0
PC>ping 10.1.1.1

Pinging 10.1.1.1 with 32 bytes of data:

Request timed out.
Reply from 10.1.1.1: bytes=32 time=8ms TTL=126
Reply from 10.1.1.1: bytes=32 time=1ms TTL=126
Reply from 10.1.1.1: bytes=32 time=6ms TTL=126

Ping statistics for 10.1.1.1:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 8ms, Average = 5ms

PC>
```

TTL = 126

Apc to Bpc 에 라우터가 2개있다.

```
PC>ping 13.13.12.2

Pinging 13.13.12.2 with 32 bytes of data:

Reply from 13.13.12.2: bytes=32 time=1ms TTL=254
Reply from 13.13.12.2: bytes=32 time=3ms TTL=254
Reply from 13.13.12.2: bytes=32 time=3ms TTL=254
Reply from 13.13.12.2: bytes=32 time=4ms TTL=254

Ping statistics for 13.13.12.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 4ms, Average = 2ms

PC>
```

TTL = 254

중간에 라우터가 1대있다.

```

PC>ping 192.168.1.254

Pinging 192.168.1.254 with 32 bytes of data:

Reply from 192.168.1.254: bytes=32 time=0ms TTL=255
Reply from 192.168.1.254: bytes=32 time=1ms TTL=255
Reply from 192.168.1.254: bytes=32 time=0ms TTL=255
Reply from 192.168.1.254: bytes=32 time=0ms TTL=255

Ping statistics for 192.168.1.254:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

```

중간에 라우터가 없다 (TTL이 줄어들지 않았다)
같은 네트워크라는 의미다. 255

```

PC>ping 192.168.1.2

Pinging 192.168.1.2 with 32 bytes of data:

Reply from 192.168.1.2: bytes=32 time=0ms TTL=128
Reply from 192.168.1.2: bytes=32 time=0ms TTL=128
Reply from 192.168.1.2: bytes=32 time=0ms TTL=128
Reply from 192.168.1.2: bytes=32 time=0ms TTL=128

Ping statistics for 192.168.1.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

PC>

```

줄어든게 없다 128

```

C:\Users\Administrator>ping 168.126.63.1

Ping 168.126.63.1 32바이트 데이터 사용:
168.126.63.1의 응답: 바이트=32 시간=3ms TTL=53
168.126.63.1의 응답: 바이트=32 시간=3ms TTL=53
168.126.63.1의 응답: 바이트=32 시간=3ms TTL=53
168.126.63.1의 응답: 바이트=32 시간=3ms TTL=53

168.126.63.1에 대한 Ping 통계:
    패킷: 보냄 = 4, 받음 = 4, 손실 = 0 (0% 손실),
왕복 시간(밀리초):
    최소 = 3ms, 최대 = 3ms, 평균 = 3ms

C:\Users\Administrator>

```

63 -> 53
중간에 라우터가 10대가 있다.

```

C:\Users\Administrator>ping www.google.com

Ping www.google.com [142.250.196.132] 32바이트 데이터 사용:
142.250.196.132의 응답: 바이트=32 시간=85ms TTL=109
142.250.196.132의 응답: 바이트=32 시간=85ms TTL=109
142.250.196.132의 응답: 바이트=32 시간=85ms TTL=109
142.250.196.132의 응답: 바이트=32 시간=85ms TTL=109

142.250.196.132에 대한 Ping 통계:
    패킷: 보냄 = 4, 받음 = 4, 손실 = 0 (0% 손실),
    왕복 시간(밀리초):
        최소 = 85ms, 최대 = 85ms, 평균 = 85ms

```

라우터 10대

```

C:\Users\Administrator>ping 192.168.11.30

Ping 192.168.11.30 32바이트 데이터 사용:
192.168.11.30의 응답: 바이트=32 시간<1ms TTL=128
192.168.11.30의 응답: 바이트=32 시간<1ms TTL=128
192.168.11.30의 응답: 바이트=32 시간<1ms TTL=128
192.168.11.30의 응답: 바이트=32 시간<1ms TTL=128

192.168.11.30에 대한 Ping 통계:
    패킷: 보냄 = 4, 받음 = 4, 손실 = 0 (0% 손실),
    왕복 시간(밀리초):
        최소 = 0ms, 최대 = 0ms, 평균 = 0ms

```

학원 PC 같은 네트워크 이기때문에 TTL이 줄어들지 않는다.

4. 네트워크 계층 모델

1) TCP/IP 5 Layer

계층	프로토콜		주소 유형
상위계층 (Application)	http, https(ssl), telnet ssh, ftp, ftp-data, smtp, pop3	dns, tftp, snmp, ntp dhcp server/client, syslog	-
Layer 4 (Transport)	TCP 통신 연결 O '3-Way 핸드 셰이킹' 동작 O 데이터 스트림 서비스 O 흐잡 제어 기능 O 재전송 기능 O Window Size O 오류 검사 O	UDP 통신 연결 X '3-Way 핸드 셰이킹' 동작 X 데이터 스트림 서비스 X 흐잡 제어 기능 X 재전송 기능 X Window Size X 오류 검사 O	포트 번호
Layer 3 (Internet)	IP 로컬 환경에서 리모트 환경으로 데이터 전송 TTL를 이용하여 패킷 루프 방지 기능 및 거리 측정		IP 주소
Layer 2 (Network Interface)	Ethernet Ethernet 내부 환경에서 데이터 전송 최종적으로 오류 검사 실시		MAC 주소
Layer 1 (Physical)	전기 신호 변환 및 입출력		bit(0,1)

ex)

Ethernet | IP | UDP | DNS 요청

2) OSI 7 Layer

데이터 생성과 전송 과정을 7개 계층으로 제시한 모델이다. OSI 7 Layer를 이해하고 있다면, 네트워크 작업 및 장애 처리 접근을 손쉽게 할 수 있다.

Layer 7 애플리케이션	서비스가 구현되는 계층	
Layer 6 프레젠테이션	서비스를 어떤 방식으로 표현할 것인지를 결정	
Layer 5 세션	OS/서비스 간에 논리적인 연결	
----- 상위 계층(서비스 구현, OS/응용 프로그램 담당)		
Layer 4 트랜스포트	TCP	UDP
Layer 3 네트워크	IP	
Layer 2 데이터 링크	Ethernet	
Layer 1 물리 계층	전기 신호 변환, 입출력	
----- 하위 계층(전송 담당, 네트워크 장비/전송 프로토콜 담당)		

- Application 계층
- 사용자 입장에서 서비스가 구현되는 계층
- 응용 프로그램

5. ICMP (Internet Control Message Protocol)

- IP 프로토콜을 이용하여 데이터 전송이 가능한지 확인하기 위해서 메시지를 생성하여 요청 및 응답을 실시하는 프로토콜이다.
- 부록 (결제부록)
 - 있어도 되고 없어도 된다.
- 서버를 연결한뒤 테스트를 할 때 쓰는 프로토콜
 - 데이터가 전송되는지 안되는지를 테스트할때 사용
- 요청 메시지와 응답 메시지를 만들어주고 전송해주는 프로토콜

1) 기본적인 ICMP 메시지 유형

유형	타입	내용
echo	8	ICMP 요청 메시지
echo-reply	0	ICMP 응답 메시지
Destination Unreachable	3	ICMP 목적지 도달 불가능 응답 메시지
TTL Exceeded	11	TTL 이 만료된 응답 메시지

8번은 응답, 0번은 무응답

Internet Control Message Protocol

Type: 0 (Echo (ping) reply)

Code: 0

Checksum: 0x505c [correct]

Identifier (BE): 512 (0x0200)

Identifier (LE): 2 (0x0002)

Sequence number (BE): 768 (0x0300)

Sequence number (LE): 3 (0x0003)

[\[Request frame: 1359\]](#)

[Response time: 0.423 ms]

Data (32 bytes)

2) 기본적인 'Destination Unreachable' 메시지 유형

유형	코드	내용
Network Unreachable	0	목적지에 대한 경로가 없는 경우
Host Unreachable	1	최종 목적지 호스트에 도달할 수 없는 경우
Protocol Unreachable	2	목적지에서 특정 프로토콜을 사용할 수 없는 경우
Port Unreachable	3	목적지 호스트에서 특정 포트가 닫혀 있는 경우

1. 목적지 경로가 없음
2. 목적지 네트워크까지 왔는데 시스템이 없는 경우
3. 목적지에서 특정 프로토콜 못사용하는 경우
4. 목적지 시스템까지 도착은 했는데 포트가 안 열려있는 경우 (UDP)
 - a. 많이 사용함

3) ICMP 헤더 내용

ICMP Echo	ICMP Echo-Reply	Destination Unreachable
Internet Control Message Protocol Type: 8 (Echo (ping) request) Code: 0 Checksum: 0x485c [correct] Identifier (BE): 512 (0x0200) Identifier (LE): 2 (0x0002) Sequence number (BE): 768 (0x0300) Sequence number (LE): 3 (0x0003) [Response frame: 1362]	Internet Control Message Protocol Type: 0 (Echo (ping) reply) Code: 0 Checksum: 0x505c [correct] Identifier (BE): 512 (0x0200) Identifier (LE): 2 (0x0002) Sequence number (BE): 768 (0x0300) Sequence number (LE): 3 (0x0003) [Request frame: 1359] [Response time: 0.423 ms]	Internet Control Message Protocol Type: 3 (Destination unreachable) Code: 3 (Port unreachable) Checksum: 0xa76d [correct]

4) Ping/tracert 명령어

Ping 명령어:

'ping' 명령어를 실행하면 ICMP 메시지를 생성하고 전송하여 IP 패킷이 전송이되는지 확인할 수 있다.

관리자(192.168.1.100)	C(10.1.1.1)
ICMP Echo ->	<- ICMP Echo-Reply
SA 192.168.1.100	SA 10.1.1.1
DA 10.1.1.1	DA 192.168.1.100

ex) 4-2 데이터 전송 프로토콜pkt 파일을 실행하여 ping test를 실시한다.

```

Packet Tracer PC Command Line 1.0
PC>
PC>ping 10.1.1.1

Pinging 10.1.1.1 with 32 bytes of data:

Request timed out.
Reply from 10.1.1.1: bytes=32 time=2ms TTL=126
Reply from 10.1.1.1: bytes=32 time=2ms TTL=126
Reply from 10.1.1.1: bytes=32 time=2ms TTL=126

Ping statistics for 10.1.1.1:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 2ms, Maximum = 2ms, Average = 2ms
  
```

요청보내고 응답까지 2ms가 걸린다.

ping -t [ip]

- 계속 응답을 보낸다

Ctrl + C

- 정지시킨다.

tracert 명령어:

```
PC>tracert 10.1.1.1

Tracing route to 10.1.1.1 over a maximum of 30 hops:

  1  0 ms      0 ms      0 ms      192.168.1.254
  2  7 ms      0 ms      1 ms      13.13.12.2
  3  0 ms      4 ms      5 ms      10.1.1.1

Trace complete.

PC>
```

- 어디 라우터를 통해서 보내는지 알 수 있다.
- 출발지에서 목적지까지 경로를 추적할 수 있다.
- 어디까지 가는지 알 수 있다.
- 라우터의 IP주소를 알 수 있다.
- ICMP 메시지를 생성하고 전송한다.
-

어떻게 작동되느냐:

ICMP Echo

-----ICMP

SA 192.168.1.100

DA 10.1.1.1

-----IP(TTL = 1)

-처음에는 ttl을 0으로 함으로서 처음 달는
라우터로밖에 안가게 한다.

라우터에 달는다. 하지만 목적지에 안달는다

ttl을 2로 설정한다 +1

라우터에 달는다. 같은 네트워크지만 목적지가 아니다.

ttl을 3으로 설정한다. +1 >> 목적지에 달는다

다시 값을 관리자 PC로 보낸다.

```
C:\Users\Administrator>tracert 8.8.8.8

최대 30홉 이상의
dns.google [8.8.8.8](으)로 가는 경로 추적:

  1  <1 ms    <1 ms    <1 ms    192.168.1.1
  2  *        *        *        요청 시간이 만료되었습니다.
  3  2 ms     2 ms     2 ms     100.80.28.13
  4  2 ms     1 ms     1 ms     100.80.28.13
  5  *        3 ms     2 ms     1.213.145.13
  6  *        3 ms     2 ms     61.42.201.237
  7  10 ms    3 ms     4 ms     1.213.114.9
  8  37 ms    38 ms    37 ms    100.67.30.2
  9  38 ms    39 ms    38 ms    100.67.30.50
 10  38 ms    39 ms    38 ms    142.250.168.244
 11  39 ms    39 ms    39 ms    64.233.175.91
 12  61 ms    61 ms    62 ms    74.125.253.93
 13  38 ms    38 ms    38 ms    dns.google [8.8.8.8]

추적을 완료했습니다.
```

'tracert' 명령어를 실행하면 ICMP 메시지를 생성하고 전송하여 목적지까지 라우팅을 실시한 라우터의 IP 주소를 확인할 수 있다. 이를 통해서 출발지에서 목적지까지 경로를 추적할 수 있다.

관리자_PC>tracert 10.1.1.1

Tracing route to 10.1.1.1 over a maximum of 30 hops:

①	1	0 ms	0 ms	0 ms	192.168.1.254
②	2	2 ms	0 ms	2 ms	13.13.12.2
③	3	6 ms	1 ms	0 ms	10.1.1.1

④ Trace complete.

- ① TTL=1로 'Echo'를 전송하고 'TTL Exceeded' 응답을 수신한 내용이다.
- ② TTL=2로 'Echo'를 전송하고 'TTL Exceeded' 응답을 수신한 내용이다.
- ③ TTL=3으로 'Echo'를 전송하고 'Echo-Reply'를 수신한 내용이다.
- ④ 목적지로부터 'Echo-Reply'를 수신했기 때문에 경로 추적 테스트 완료되었다.

5) TTL이 만료된 경우

C:\Users\Administrator>ping -i 3 168.126.63.1

Ping 168.126.63.1 32 바이트 데이터 사용:

61.78.42.165의 응답: 전송하는 동안 TTL이 만료되었습니다.
61.78.42.165의 응답: 전송하는 동안 TTL이 만료되었습니다.
61.78.42.165의 응답: 전송하는 동안 TTL이 만료되었습니다.
61.78.42.165의 응답: 전송하는 동안 TTL이 만료되었습니다.

168.126.63.1에 대한 Ping 통계:

패킷: 보냄 = 4, 받음 = 4, 손실 = 0 (0% 손실),

Internet Control Message Protocol
Type: 11 (Time-to-live exceeded)
Code: 0 (Time to live exceeded in transit)
Checksum: 0x9fa3 [correct]

6) ICMP 공격을 방지하기 위해서 방화벽에서 차단한 경우

C:\Users\Administrator>ping www.naver.com

Ping www.naver.com.nheos.com [223.130.200.107] 32 바이트 데이터 사용:

요청 시간이 만료되었습니다.
요청 시간이 만료되었습니다.
요청 시간이 만료되었습니다.
요청 시간이 만료되었습니다.

223.130.200.107에 대한 Ping 통계:

패킷: 보냄 = 4, 받음 = 0, 손실 = 4 (100% 손실),

6. ARP(Address Resolution Protocol)

목적지 IP 주소에 대한 MAC 주소를 설정하는 프로토콜이다. 만약, 목적지 IP 주소에 대한 MAC 주소가 ARP 테이블에 없는 경우, 다음과 같이 ARP 요청 및 응답 실시한다.

1) ARP 동작 과정

'4-2.데이터 전송 프로토콜.pkt' 파일 환경에서 ARP 동작 과정을 알아보도록 한다.

```
C_PC>arp -d           // arp 테이블 정보 삭제
```

```
C_PC>arp -a           // arp 테이블 확인
```

```
No ARP Entries Found
```

① C 에서 D 로 'ICMP Echo'을 전송할때, D 에 대한 MAC 주소 정보가 ARP 테이블에 없는 경우

```
C(10.1.1.1)      ->      D(10.1.1.2)
00e0.f739.d44a      MAC...?

ICMP Echo
----- ICMP
SA 10.1.1.1
DA 10.1.1.2
----- IP
SA 00E0.F739.D44A
DA X           // ARP 테이블에 D(10.1.1.2)에 대한 MAC 주소 정보가 없기 때문에 목적지 MAC 주소 설정 못함
----- ETH X           // 그렇기 때문에 Ethernet 프로토콜 인캡슐레이션 실패된다.
```

② C 는 D(10.1.1.2)에 대한 MAC 주소를 학습하기 위해서 ARP 요청 메시지를 브로드캐스트로 전송한다.

```
C(10.1.1.1)      D(10.1.1.2)
00e0.f739.d44a      000A.4196.458A

"10.1.1.2, MAC...?"
----- ARP 요청 메시지
SA 00E0.F739.D44A
DA FFFF.FFFF.FFFF           // 브로드캐스트
----- ETH
```

③ D 는 자신의 IP 주소와 MAC 주소 정보를 ARP 응답 메시지를 생성하여 유니케스트로 C 에게 전송한다.

C(10.1.1.1)	D(10.1.1.2)
00E0.F739.D44A	000A.4196.458A
"10.1.1.2, MAC 000A.4196.458A"	
----- ARP 응답 메시지	
SA 000A.4196.458A	
DA 00E0.F739.D44A // 유니케스트	
----- ETH	

④ ARP 응답 메시지를 수신한 C 는 ARP 테이블에 D 의 IP 주소와 MAC 주소 정보를 등록한다.

C_PC>arp -a

Internet Address	Physical Address	Type
10.1.1.2	000A.4196.458A	dynamic

⑤ C 는 ARP 테이블에 D 에 대한 IP 주소와 MAC 주소 정보가 있기 때문에 'ICMP Echo'를 전송할 수 있다.

C(10.1.1.1)	->	D(10.1.1.2)
ICMP Echo		
----- ICMP		
SA 10.1.1.1		
DA 10.1.1.2		
----- IP		
SA 00E0.F739.D44A		
DA 000A.4196.458A // ARP 테이블에 등록된 '10.1.1.2' IP 주소에 대한 MAC 주소를 설정한다.		
----- ETH		

⑥ C 에서 D(10.1.1.2), E(10.1.1.3)로 Ping 테스트를 실시한 이후, ARP 테이블 정보를 확인한다.

C_PC>ping 10.1.1.2

Pinging 10.1.1.2 with 32 bytes of data:

Reply from 10.1.1.2: bytes=32 time=0ms TTL=128

Reply from 10.1.1.2: bytes=32 time=0ms TTL=128

Reply from 10.1.1.2: bytes=32 time=0ms TTL=128

Reply from 10.1.1.2: bytes=32 time=1ms TTL=128

Ping statistics for 10.1.1.2:

Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),

Approximate round trip times in milli-seconds:

Minimum = 0ms, Maximum = 1ms, Average = 0ms

C_PC>ping 10.1.1.3

Pinging 10.1.1.3 with 32 bytes of data:

Reply from 10.1.1.3: bytes=32 time=1ms TTL=128

Reply from 10.1.1.3: bytes=32 time=1ms TTL=128

Reply from 10.1.1.3: bytes=32 time=1ms TTL=128

Reply from 10.1.1.3: bytes=32 time=0ms TTL=128

Ping statistics for 10.1.1.3:

Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),

Approximate round trip times in milli-seconds:

Minimum = 0ms, Maximum = 1ms, Average = 0ms

C_PC>arp -a

Internet Address	Physical Address	Type
10.1.1.2	000a.4196.458a	dynamic
10.1.1.3	000c.8562.d6c2	dynamic

2) 다른 네트워크로 ARP 요청 불가

라우터는 브로드캐스트를 다른 네트워크로 전송하지 않는다. 그렇기 때문에 다른 네트워크 환경에 있는 시스템에 대한 ARP 요청은 불가능하다. 그래서 로컬 PC 는 Gateway IP 주소에 대한 MAC 주소를 학습하기 위한 ARP 요청을 실시하여 Gateway MAC 주소를 학습한다.

C 에서 서버(192.168.1.100)로 Ping 테스트를 실시한 이후, ARP 테이블 정보를 확인한다.

C_PC>ipconfig

FastEthernet0 Connection:(default port)

Link-local IPv6 Address.....: ::
IP Address.....: 10.1.1.1
Subnet Mask.....: 255.255.255.0
Default Gateway.....: 10.1.1.254

C_PC>ping 192.168.1.100

Pinging 192.168.1.100 with 32 bytes of data:

Reply from 192.168.1.100: bytes=32 time=7ms TTL=126
Reply from 192.168.1.100: bytes=32 time=3ms TTL=126
Reply from 192.168.1.100: bytes=32 time=1ms TTL=126
Reply from 192.168.1.100: bytes=32 time=1ms TTL=126

Ping statistics for 192.168.1.100:

Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),

Approximate round trip times in milli-seconds:

Minimum = 1ms, Maximum = 7ms, Average = 3ms

PC>arp -a

Internet Address	Physical Address	Type
10.1.1.2	000a.4196.458a	dynamic
10.1.1.3	000c.8562.d6c2	dynamic
10.1.1.254	000a.41ad.3301	dynamic

C(10.1.1.1) -> 관리자(192.168.1.100)

ICMP Echo

----- ICMP

SA 10.1.1.1

DA 192.168.1.100

----- IP

SA 00E0.F739.D44A

DA 000A.41AD.3301 // ARP 테이블에 등록된 게이트웨이 '10.1.1.254' IP 주소에 대한 MAC 주소를 설정한다.

----- ETH

ICMP

- IP 프로토콜을 도와주는 역할

ARP

- ETH 도와주는 역할 (MAC 세팅)