데이터통신 과제

<Packet Capture Using Wireshark>

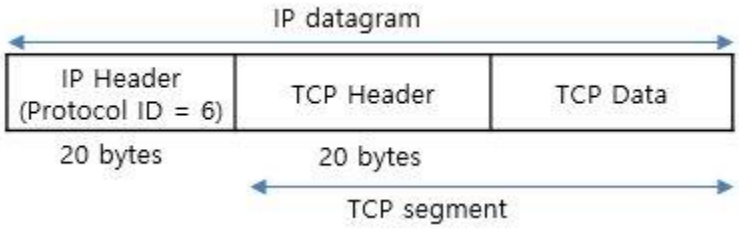
2017320122 김정규

TCP와 ARP

TCP란 TCP/IP 프로토콜 群 중 하나로서, OSI 계층모델의 관점에서 트랜스포트 계층(4계층)에 해당한다. 양 종단 호스트 내 프로세스 상호 간에 신뢰적인 연결지향성 서비스를 제공하며 IP의 비신뢰적인 최선형 서비스에 신뢰적인 연결지향성 서비스를 제공한다. TCP는 이러한 신뢰적인 전송을 보장함으로써, 어플리케이션 구현이 한층 쉬워지게 한다.

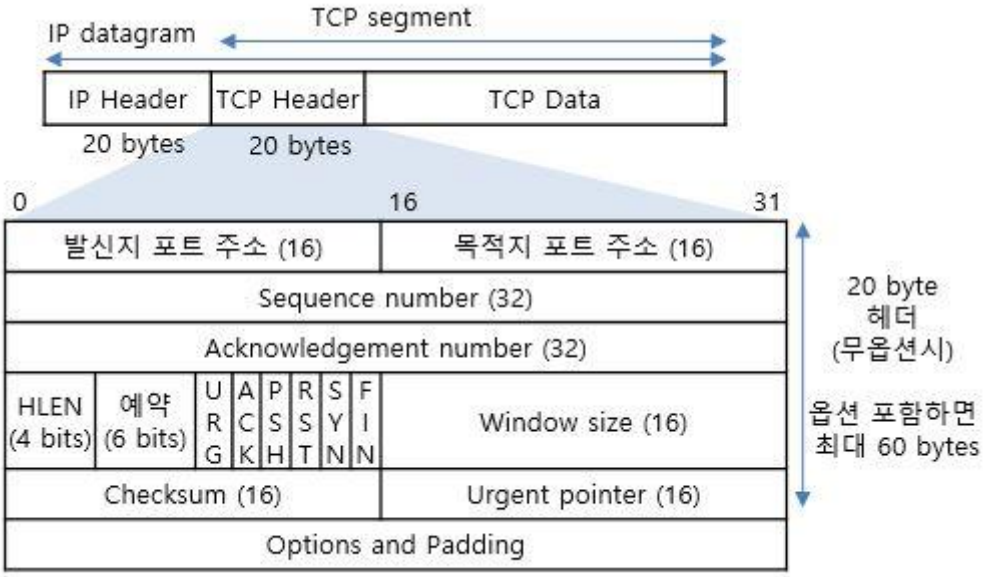
ARP는 논리적인 IP 주소를 (망계층), 물리적인 MAC 주소로 (데이터링크 계층), 바꾸어주는 역할을 하는 주소 해석 프로토콜로서 라우터에서 동작할 때 크게 ARP를 요청하고 응답하는 방식으로 진행된다. 단, ARP 패킷은 3계층(네트워크계층)을 통해 타 네트워크로 넘어 갈 수 없다.

와이어샤크를 통한 TCP/ARP 패킷 분석에 본격적으로 들어가기 전에 TCP와 ARP의 구조에 대해 먼저 알아보도록 하자.



<그림1 TCP 패킷 구조>

상기 그림은 TCP 패킷의 구조를 나타낸다. IP datagram 안에 TCP data가 캡슐화 된 형태이며 이를 자세히 알기 위해 TCP 헤더를 살펴볼 필요가 있다.



<그림2 TCP 헤더 구조>

상기 그림에 나타난 각 필드의 설명은 다음과 같다.

Sourse/Destination Port Number (각 16 비트)

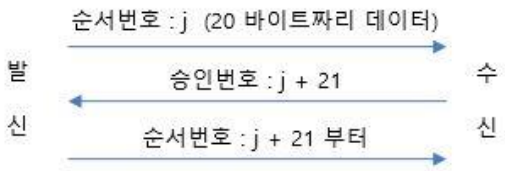
- IP 주소 + 포트 번호 = 소켓 주소
  - 양쪽 호스트 내 종단 프로세스 식별

Sequence Number (32 비트)

- 바이트 단위로 구분되어 순서화 되는 번호로 신뢰성 및 흐름제어 기능 제공

Acknowledgement Number(확인응답번호/승인번호) (32 비트)

- 수신하기를 기대하는 다음 바이트 번호 = (마지막 수신 성공 순서번호 + 1)



<그림3 확인 번호>

헤더 길이 필드 (Header length, HLEN, 4 비트)

- TCP 헤더 길이를 4 바이트(32 비트) 단위로 표시

6개의 Flag bits (URG, ACK, PSH, RST, SYN, FIN)

- TCP 세그먼트 전달과 관련되어 TCP 회선 및 데이터 관리 제어 기능을 하는 플래그
- 관계된 기능 : 흐름제어, 연결설정, 연결종료, 연결리셋, 데이터전송모드 등

윈도우 크기 (Window size, 16 비트)

- 흐름제어를 위해 사용하는 16 비트 필드 (65,535 bytes까지 가능)
- TCP 흐름제어를 위해 송신자에게 자신의 수신 버퍼 여유용량 크기를 지속적으로 통보
  .. 이 필드에 자신의 수신 버퍼 용량 값을 채워 보내게 됨 (지속적인 현행화)
    - 결국, 수신측에 의해 능동적으로 흐름제어를 수행하게 됨

Checksum (16 비트)

- 검사합

Urgent pointer (16 비트)

- TCP 세그먼트에 포함된 긴급 데이터의 마지막 바이트에 대한 일련번호
- 일련번호(sequence number)로부터 긴급 데이터까지의 바이트 오프셋(offset)
- 해당 세그먼트의 일련번호에 urgent point 값을 더해 긴급 데이터의 끝을 알 수 있음

옵션

- 최대 40 바이트까지 옵션 데이터 포함 가능
- TCP MSS 옵션을 협상하거나, 주어진 윈도우 크기 보다 더 크게 사용하거나 선택확인응답을 하거나, 타임스탬프 옵션 정의 등

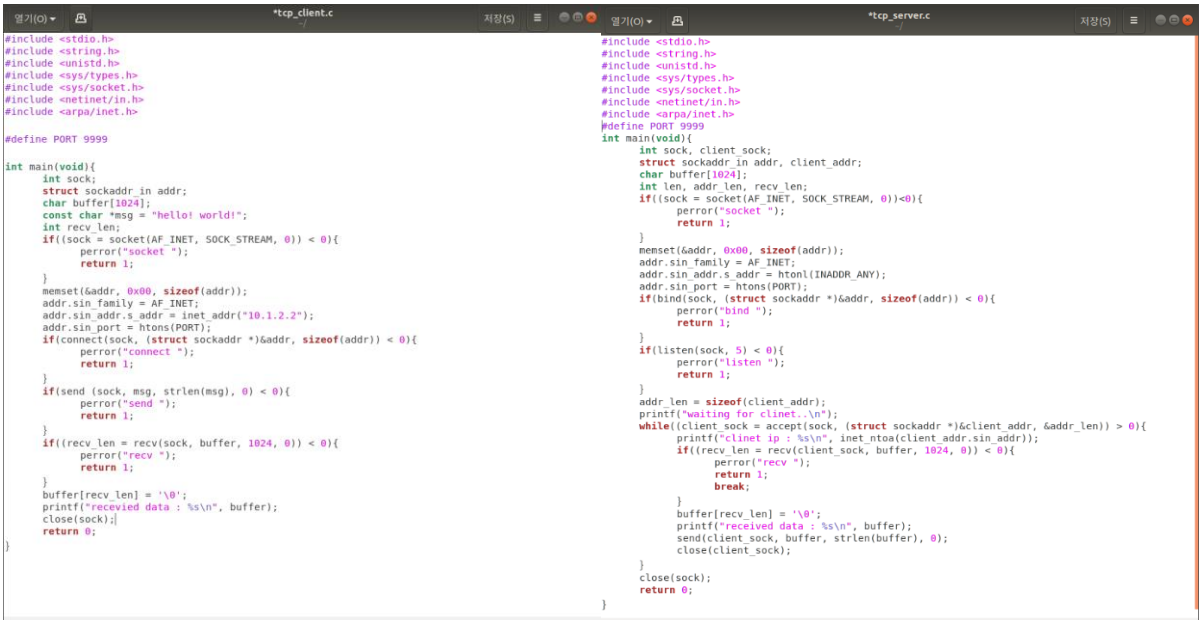
ARP 패킷은 주로 요청과 응답을 통해 망계층의 IP 주소 및 데이터링크 계층의 MAC 주소 정보를 조회하고 알려주는 패킷으로 그 구조는 다음과 같다.

0		31
Hardware type		Protocol Type
Hardware Length	Protocol Length	Operation
Sender Hardware Address		
Sender Protocol Address		
Target Hardware Address		
Target Protocol Address		

<그림4 ARP 패킷 구조>

Wireshark 패킷 분석

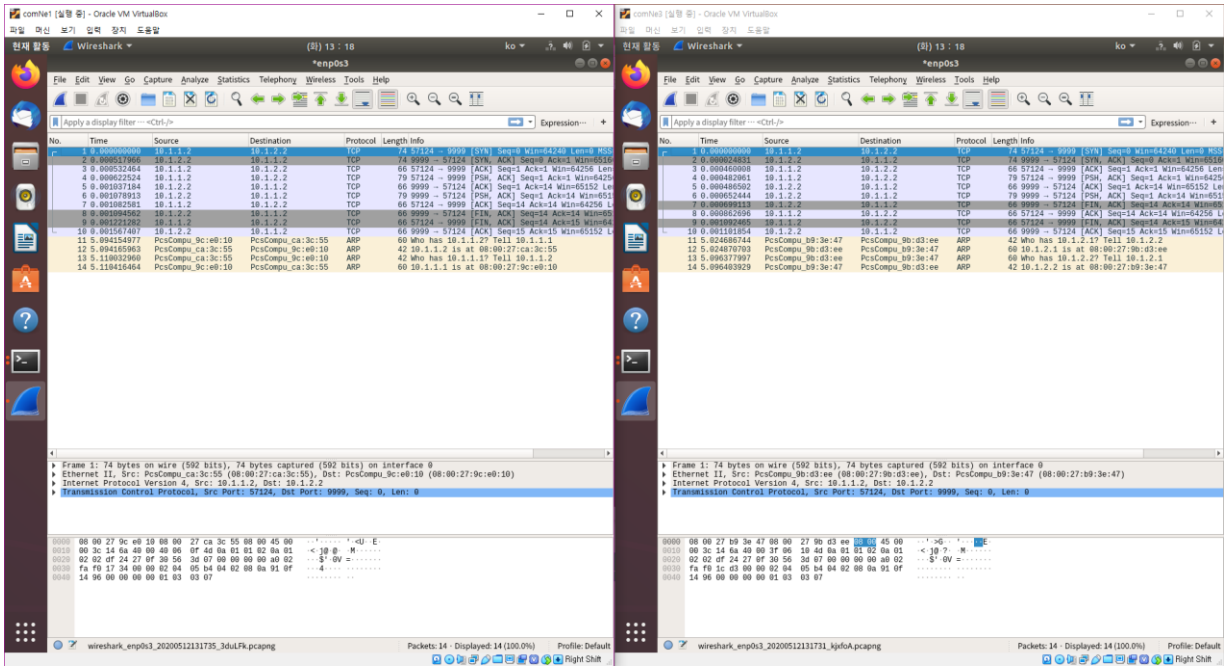
우분투에 다음 코드를 컴파일하여 tcp 서버와 클라이언트를 실행하였고 실행 중 일어난 패킷 교환을 와이어샤크로 캡처하였다. 1번 머신이 클라이언트, 3번 머신이 서버로 동작했다. 밑의 코드 중 서버에 해당하는 코드에서 클라이언트로부터 받는 소켓을 토대로 로컬 IP, 로컬 포트, 원격 IP, 원격 포트, IP 프로토콜을 identify하여 처리하고 통신한다.



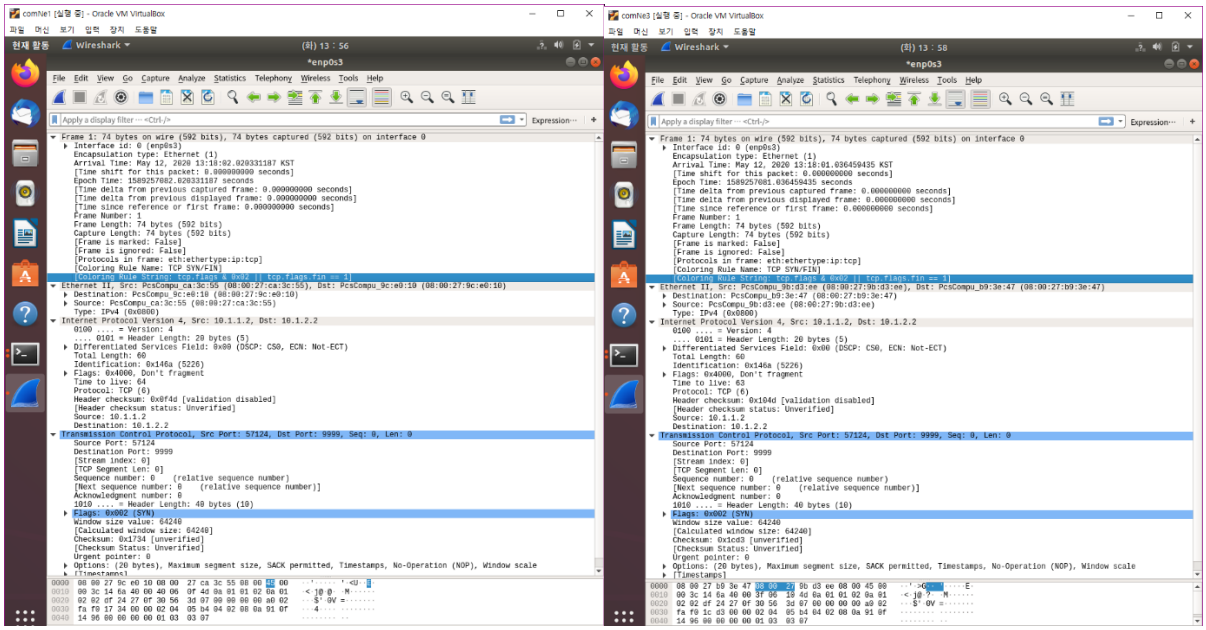
```
*tcp_client.c
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define PORT 9999

int main(void){
	int sock;
	struct sockaddr_in addr;
	char buffer[1024];
	const char *msg = "hello! world!";
	int recv_len;
	if((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0){
		perror("socket ");
		return 1;
	}
	memset(&addr, 0x00, sizeof(addr));
	addr.sin_family = AF_INET;
	addr.sin_addr.s_addr = inet_addr("10.1.2.2");
	addr.sin_port = htons(PORT);
	if(connect(sock, (struct sockaddr *)&addr, sizeof(addr)) < 0){
		perror("connect ");
		return 1;
	}
	if(send (sock, msg, strlen(msg), 0) < 0){
		perror("send ");
		return 1;
	}
	if((recv_len = recv(sock, buffer, 1024, 0)) < 0){
		perror("recv ");
		return 1;
	}
	buffer[recv_len] = '\0';
	printf("recevied data : %s\n", buffer);
	close(sock);
	return 0;
}
```

```
*tcp_server.c
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#define PORT 9999
int main(void){
	int sock, client_sock;
	struct sockaddr_in addr, client_addr;
	char buffer[1024];
	int len, addr_len, recv_len;
	if((sock = socket(AF_INET, SOCK_STREAM, 0))<0){
		perror("socket ");
		return 1;
	}
	memset(&addr, 0x00, sizeof(addr));
	addr.sin_family = AF_INET;
	addr.sin_addr.s_addr = htonl(INADDR_ANY);
	addr.sin_port = htons(PORT);
	if(bind(sock, (struct sockaddr *)&addr, sizeof(addr)) < 0){
		perror("bind ");
		return 1;
	}
	if(listen(sock, 5) < 0){
		perror("listen ");
		return 1;
	}
	addr_len = sizeof(client_addr);
	printf("waiting for clinet..\n");
	while((client_sock = accept(sock, (struct sockaddr *)&client_addr, &addr_len)) > 0){
		printf("clinet ip : %s\n", inet_ntoa(client_addr.sin_addr));
		if((recv_len = recv(client_sock, buffer, 1024, 0)) < 0){
			perror("recv ");
			return 1;
			break;
		}
		buffer[recv_len] = '\0';
		printf("received data : %s\n", buffer);
		send(client_sock, buffer, strlen(buffer), 0);
		close(client_sock);
	}
	close(sock);
	return 0;
}
```

<그림5 클라이언트/서버 코드>



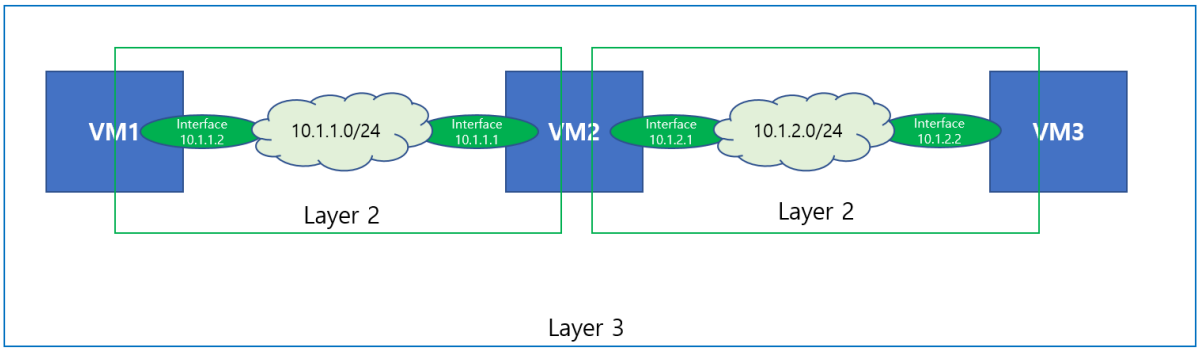
<그림5 패킷 캡처1>



<그림6 TCP 패킷>

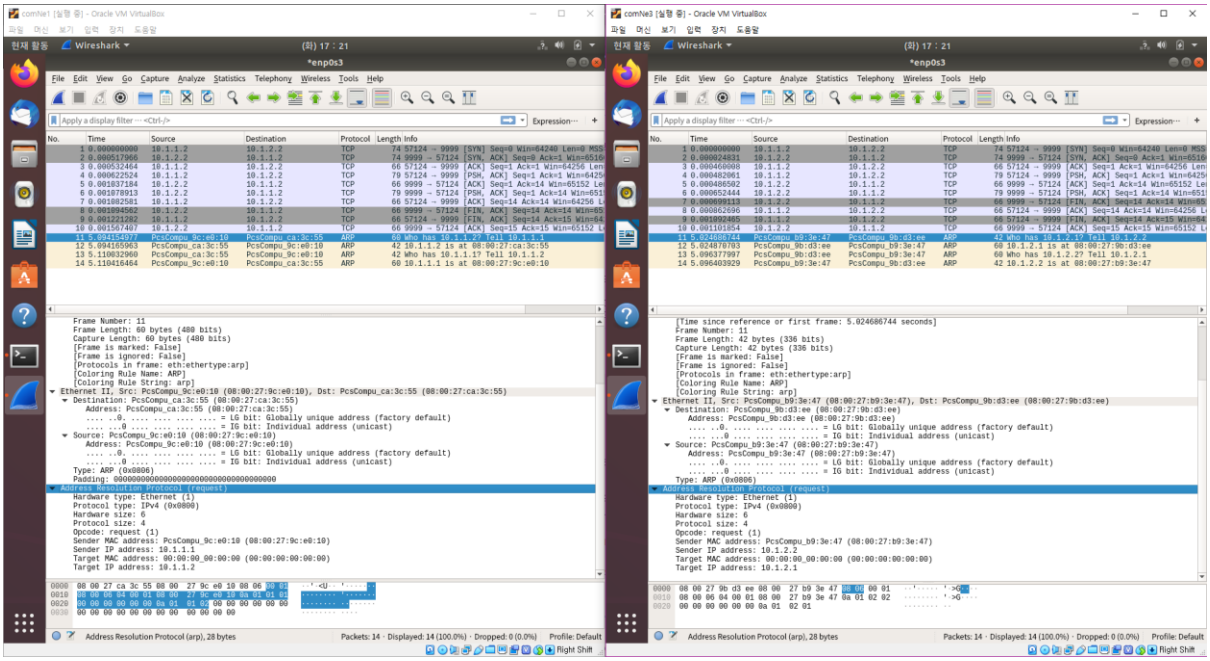
위의 그림을 살펴보면 패킷은 TCP 프로토콜을 통해 상호작용 하였으며, TCP는 3-Way-Handshaking을 통해 이루어진다는 것을 알 수 있다. 이는 말그대로 3단계로 진행된다. 1단계로는 송신자 SYN을 보낸다. 2단계로는 수신자는 송신자가 보낸 SYN에 대한 응답으로 SYN에서 1 증가한 ACK과 또다른 SYN을 보낸다. 3단계로는 송신자는 수신자로부터 받은 ACK를 SYN으로, 수신자로부터 받은 SYN에서 1 증가한 ACK을 다시 보낸다. 이 과정이 이루어진 뒤에 송신자는 수신자에게 데이터를 전송한다.

캡처한 내용에 따르면 소스와 데스티네이션의 주소가 머신1과 머신3의 주소가 번갈아 나오므로, 머신1은 머신2를 거쳐 머신3과 통신하고 있으며 9999의 포트 번호를 지녔다. 포트 번호는 인터넷이나 기타 다른 네트워크 메시지가 서버에 도착하였을 때, 전달되어야 할 특정 프로세스를 인식하기 위한 것이다. 다음 토폴로지를 참고하면 이해가 쉬울 것이다.

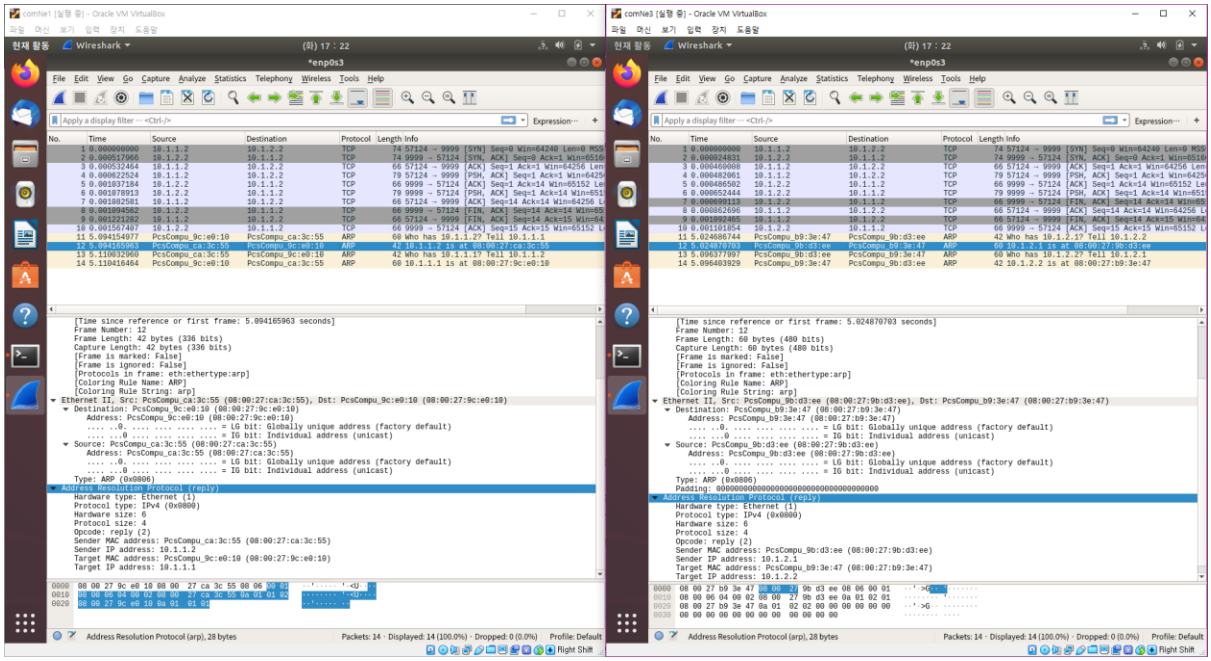


<그림7 네트워크 토폴로지>

이제는 좀 더 자세히 알아 보기 위해 ARP 패킷을 확인해보자.



<그림8 ARP Request>



<그림9 ARP Reply>

위 캡처를 그림 4 를 토대로 분석해보자.

Hardware type

이 필드는 사용 중인 MAC 주소나 데이터 링크 유형 / 물리 주소 길이를 정의하는데 사용되며, 여분의 Hardware Length 필드를 생성한다. Hardware type이 1이면, 이더넷(Ethernet)에 할당된 것을 의미한다. 캡처본은 1이니 이더넷이다.

Hardware Length(size)

이 필드는 해당 패킷에 사용되는 MAC 주소의 길이를 정의하며, 단위는 Byte이다. Hardware Length의 값은 Hardware type에 의해 결정되기 때문에 여분의 필드이다. 캡처본의 사이즈값은 6이다.

Protocol Type

이 필드는 사용 중인 프로토콜 주소 유형을 정의 하며, 이더넷(Ethernet) II 프레임 구조에 사용되는 표준 프로토콜 ID 값을 사용한다. 그리고, Hardware type과 같이 여분의 Protocol Length 필드를 생성한다. 캡처본의 타입은 IPv4이다.

자세한 Protocol Type은 http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xml 참조.

Protocol Length(size)

이 필드는 패킷에 사용되는 프로토콜 주소의 길이를 정의하며, 단위는 Byte이다. Protocol Length의 값은 Protocol Type에 의해 결정되기 때문에 여분의 필드이다. 캡처본의 사이즈 값은 4이다.

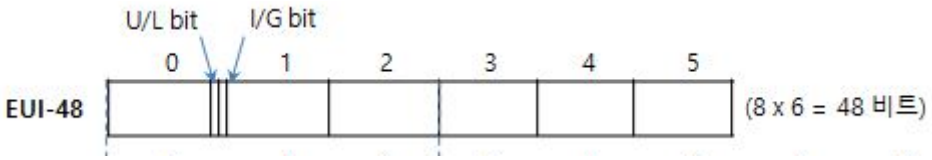
Operation(Opcode)

이 필드는 ARP Request / Reply 또는 RARP Request / Reply 패킷인지를 등을 정의한다. 밑의 표를 참고하자. 더 자세한 Operation Code는 http://www.iana.org/assignments/arp-parameters/arp-parameters.xml의 Operation Code 파트를 참조. 캡처본은 아래의 표대로 request일 때 1, reply일 때 2이다.

Number	Operation Code
1	ARP Request
2	ARP Reply
3	RARP Request
4	RARP Reply

Sender Hardware (MAC) Address

이 필드는 ARP 요청과 응답을 전송하는 장치의 MAC 주소를 정의 한다. 캡처본의 주소는 EUI-48을 따르며, 이의 자세한 구조는 아래의 그림10과 같다..



<그림10 Structure of EUI-48>

Sender Protocol (IP) Address

이 필드는 ARP 요청과 응답을 전송하는 장치의 IP 주소를 정의한다.

Target Hardware (MAC) Address

이 필드는 요구되는 대상의 MAC 주소를 정의한다. ARP Request 패킷의 경우 일반적으로 0으로 채워져 표시되며, ARP Reply 패킷의 경우 ARP Request 패킷을 전송한 장치의 MAC 주소를 표시 한다.

Target Protocol (IP) Address

이 필드는 요구되는 대상의 IP 주소를 정의한다. ARP Request 패킷의 경우 질의할 IP주소가 표시되며, ARP Reply 패킷의 경우 ARP Request 패킷을 전송한 IP 주소를 표시 한다.

IP와 MAC address의 정보를 보면 머신1의 캡처본에서는 머신1과 2가 통신한다는 것을 알 수 있다. 송수신자의 request와 reply의 주소가 번갈아 나오기 때문이다. 머신1의 첫 request의 타겟 맥 주소는 그냥 0인 것으로 나오는데 그 시점에서 타겟의 아이피만 알 뿐, 맥 주소는 아직 모르기 때문이다.

그리고 중간에 맥 주소가 바뀌는 것처럼 보이는데 이것은 MAC address table과 연관이 있다. 앞서 캡처한 내용은 클라이언트가 도착하고자 하는 목적지(서버)를 게이트웨이를 통해 가는 것이다. 머신1이 던진 패킷은 출발지가 10.1.1.1이고 목적지는 10.1.2.2이다. 그 말은 즉 IP로 가는 경로에 대해서 이미 학습을 하고 있다는 것이다. 학습한 내용을 라우팅테이블이란 곳에 기억하고 있다고 볼 수 있는데, 목적지 IP를 라우팅테이블을 참조해서 10.1.2.2에 연결된 인터페이스로 패킷을 보내게 된다. 참고로 라우팅테이블은 제 3계층에 존재하는 개념이다. 여기에선 또다시 제 2계층으로 넘어와 MAC address를 참조하여 패킷을 전달하는데 이러한 과정 속에서 MAC address가 바뀌게 된다. 제 3계층 장비를 지나치게 되면 IP주소는 변경되지 않지만 SRC MAC address는 마지막에 떠난 인터페이스로 변경된다