DNS 관련 문제
-> 도메인 HEX로 표현할 수 있어야 한다.

ipconfig/displaydns
ipconfig/flushdns

udp.port==53
tcp.port==53
-> filter of DNS

z.y.x.w.in-addr.arpa
z.y.x.w.ip6.arpa

cnn.com의 ANS에게 직접 묻는 방법
1. ns로 cnn.com의 ANS을 찾는다.
2. server로 ANS의 DN을 기본으로 설정한다
3. a로 cnn.com을 찾는다.

nslookup에서 server 명령어를 사용하면, 질문을 받아줄 서버를 특정할 수 있다.
-> set querytype=ns로 서버 dns를 찾으면 된다.

타입	의미	번호
A	IPv4 주소	1
AAAA	IPv6 주소	28
NS	DNS 서버	2
MX	메일 서버	15
CNAME	본명 (canonical name)	5
PTR	IP 주소에 대응되는 도메인 이름	12
SOA	도메인 관리 정보	6
SRV	NS, MX 이외의 서버 정보	33
TXT	임의의 텍스트 정보	16

```
> set querytype=a
> korea.ac.kr
서버:    kns.kornet.net
Address:  168.126.63.1

권한 없는 응답:
이름:    korea.ac.kr
Address:  163.152.6.10
```

```
> set querytype=aaaa
> korea.ac.kr
서버:    kns.kornet.net
Address:  168.126.63.1

이름:    korea.ac.kr
```

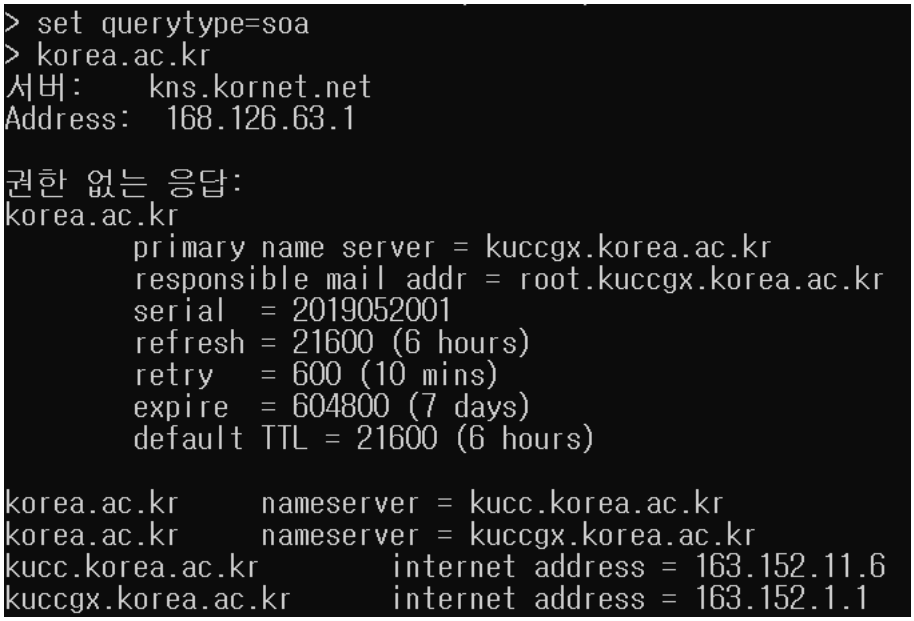
```
> set querytype=ns
> korea.ac.kr
서버:    kns.kornet.net
Address:  168.126.63.1

권한 없는 응답:
korea.ac.kr     nameserver = kucc.korea.ac.kr
korea.ac.kr     nameserver = kuccgx.korea.ac.kr

kucc.korea.ac.kr        internet address = 163.152.11.6
kuccgx.korea.ac.kr      internet address = 163.152.1.1
```

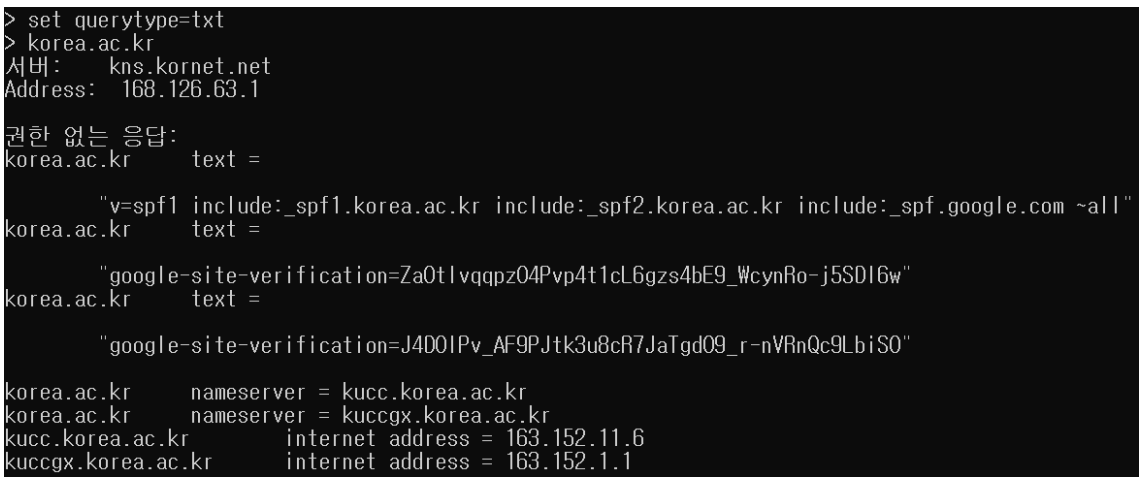
```
> set querytype=mx
> korea.ac.kr
서버:    kns.kornet.net
Address:  168.126.63.1

권한 없는 응답:
korea.ac.kr     MX preference = 5, mail exchanger = ALT1.ASPMX.L.GOOGLE.COM
korea.ac.kr     MX preference = 5, mail exchanger = ALT2.ASPMX.L.GOOGLE.COM
korea.ac.kr     MX preference = 10, mail exchanger = ALT4.ASPMX.L.GOOGLE.COM
korea.ac.kr     MX preference = 1, mail exchanger = ASPMX.L.GOOGLE.COM
korea.ac.kr     MX preference = 10, mail exchanger = ALT3.ASPMX.L.GOOGLE.COM

korea.ac.kr     nameserver = kucc.korea.ac.kr
korea.ac.kr     nameserver = kuccgx.korea.ac.kr
ASPMX.l.google.com      internet address = 74.125.203.27
ALT1.ASPMX.l.google.com internet address = 108.177.10.27
ALT2.ASPMX.l.google.com internet address = 209.85.200.26
ALT3.ASPMX.l.google.com internet address = 64.233.177.27
alt4.ASPMX.l.google.com internet address = 209.85.144.26
kucc.korea.ac.kr        internet address = 163.152.11.6
kuccgx.korea.ac.kr      internet address = 163.152.1.1
ASPMX.l.google.com      AAAA IPv6 address = 2404:6800:4008:c01::1b
ALT1.ASPMX.l.google.com AAAA IPv6 address = 2607:f8b0:4003:c14::1b
ALT2.ASPMX.l.google.com AAAA IPv6 address = 2607:f8b0:4001:c16::1b
alt4.ASPMX.l.google.com AAAA IPv6 address = 2607:f8b0:400d:c0e::1b
```

```
> set querytype=cname
> kucc.korea.ac.kr
서버:    kns.kornet.net
Address:  168.126.63.1

korea.ac.kr
        primary name server = kuccgx.korea.ac.kr
        responsible mail addr = root.kuccgx.korea.ac.kr
        serial  = 2019052001
        refresh = 21600 (6 hours)
        retry   = 600 (10 mins)
        expire  = 604800 (7 days)
        default TTL = 21600 (6 hours)
```

```
> set querytype=ptr
> 10.6.152.163.in-addr.arpa
서버:    kns.kornet.net
Address:  168.126.63.1

권한 없는 응답:
10.6.152.163.in-addr.arpa       name = www.korea.ac.kr
10.6.152.163.in-addr.arpa       name = sub.korea.ac.kr

152.163.in-addr.arpa    nameserver = kucc.korea.ac.kr.152.163.in-addr.arpa
152.163.in-addr.arpa    nameserver = kuccgx.korea.ac.kr.152.163.in-addr.arpa
kucc.korea.ac.kr.152.163.in-addr.arpa   internet address = 163.152.11.6
kuccgx.korea.ac.kr.152.163.in-addr.arpa internet address = 163.152.1.1
```

```
> set querytype=srv
> _ldap._tcp.openldap.org
서버:    kns.kornet.net
Address:  168.126.63.1

권한 없는 응답:
_ldap._tcp.openldap.org SRV service location:
          priority       = 0
          weight         = 0
          port           = 389
          svr hostname   = www.openldap.org

openldap.org    nameserver = ns4.he.net
openldap.org    nameserver = ns1.he.net
openldap.org    nameserver = ns3.he.net
openldap.org    nameserver = ns2.he.net
openldap.org    nameserver = ns5.he.net
www.openldap.org        internet address = 23.92.27.230
ns1.he.net      internet address = 216.218.130.2
ns2.he.net      internet address = 216.218.131.2
ns3.he.net      internet address = 216.218.132.2
ns4.he.net      internet address = 216.66.1.2
ns5.he.net      internet address = 216.66.80.18
www.openldap.org        AAAA IPv6 address = 2600:3c01::f03c:91ff:fedb:ad59
ns1.he.net      AAAA IPv6 address = 2001:470:100::2
ns2.he.net      AAAA IPv6 address = 2001:470:200::2
ns3.he.net      AAAA IPv6 address = 2001:470:300::2
ns5.he.net      AAAA IPv6 address = 2001:470:500::2
```





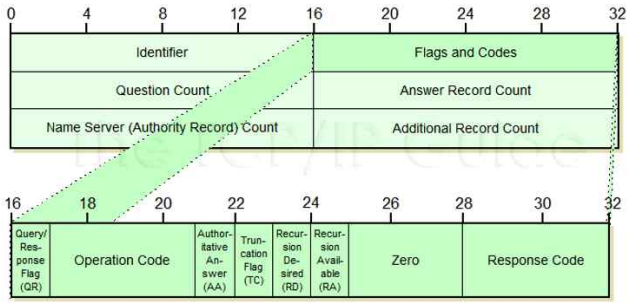
refresh: 매 6시간 마다 한번씩 secondary를 업데이트한다.
retry: refresh가 일어나지 않았을 때, 10분마다 primary를 찔러본다.
expire: 7일이 지나도 응답이 없으면 그 primary를 버림
TTL: 6시간이 지나면 그 응답은 무효한 것이다.
6시간 뒤 그 응답이 expire된다.

SRV: _서비스이름._트랜스포트 프로토콜 명._도메인 이름
-> 이 도메인에서 이 프로토콜을 사용해서 이 서비스를 하는 애가 누구야?

-> 답 자체를 준다. (recursive)
-> 답을 알고 있는 서버의 정체를 준다. (iterative)

상층 DNS: arpa, gTLD, ccTLD
anycast로 상층 root DNS를 간다. (hop이 적은 곳으로)

Identification	Flags
# of questions	# of answer RRs
# of authority RRs	# of additional RRs
Questions (variable #)	
Answers (variable # of RRs)	
Authority (variable # of RRs)	
Additional information (variable # of RRs)	



Query name (variable length)	
Query type	Query class

Domain name (variable length)	
type	class
TTL	
Resource data length	
Resource data	

QR: 0이면 질문(query), 1이면 답변(response)이다.
Opcode:
Authority Answer: 0이면 cache에서 온 것이다. 1이면 Authority name server에서 온 것이다.

Truncated: 잘렸다.
질문은 그럴 일이 적다.

한편, 512B라는 기준은 어디에서부터 온것인가?
-> 인터넷에서 쓰이는 interface는 적어도 최소 576B은 처리할 수 있어야 한다.
(MTU가 576B까지는 담을 수 있어야 한다.)

IP header가 최소 20B이고, UDP header가 8B정도 된다.
->약 28B정도 된다.
576-28=548
여유 공간으로 적당히 빼고나니, 약 512B정도가 남는다.
이 정도로 설정하니, 576B를 잘 넘기지 않았다.

응답은 512B을 훨씬 넘길 수도 있다.
-> 이러면 잘린다. 512B로 잘라서 잘린 부분만 보냄.
-> host는 이것을 받아서 그냥 버리고,
-> 동일한 질문을 TCP로 보냄.

Recursive Desired: 1이면, “그냥 제발 답만 줘!”라고 떼쓰는 거다.
Recursive Available: 1이면, “그래, 그냥 답만 줄게...”하는 것이다.
-> 상위 DNS 서버는 0으로 설정되어 있다. “안돼! 나 바쁘니까 귀찮게 하지마!”라고 한다.

ResponseCode=0 그냥 0으로 채워져 있으면 정상이다.

Authoritive DNS 서버를 왜 다른 subnet에 넣는가?
-> 같은 subnet이면 하나의 subnet에만 문제가 생겨도, DNS 서비스 자체가 위험해질 수 있기 때문,

surfix가 반복되면 FQDN이 아니다.
-> invaild한 값으로 여김

FQDN을 위해서 맨 마지막에 .을 자동으로 붙여준다.

Internet computer들은 적어도 576B 이상처리해야 한다.
512B는 IP 패킷 size - (최소 IP, UDP header 길이)와 유사하다.

잘리면:
DNS 서버가 TC flag=1로 설정
DNS Resolver: 응답을 받아서 버림
DNS Resolver: 같은 질문을 TCP로 한다

Pointer와 compressed label은 다르다.
Pointer는 c0 xx면 전부 포인터고,
Compressed label은 Domain name안에 pointer가 있어야 한다.

Resolver는 오직 FQDN이 아닐 때에만 뒤에 surfix를 붙인다.

-Multicast-
IGMPv2 group-specific queries: igmp.type=17
IP multicast datagram: ip.addr>=224.0.0.0

Korea university glop: 233.36.236.0/24
Korea university AS Number: 9452

In response to the IGMPv2 Leave message, the multicast router sends an IGMP query message at time T.
When will the router send the same query above?
-> T+1
그냥 주기적으로 패킷을 보내는데, 그때의 address는 224.0.0.1이다.
IGMP의 TTL은 1이다.

IP protocol number of ICMP: 1
IP protocol number of IGMP: 2

multicast router가 IGMP를 듣게 하는 방법:
set the Router Alert option
let them implicitly join all multicast groups

IGMPv3에서 host가 group을 join하려고 한다. source에 상관없이
그 모드는?-> CHANGE-to-EXCLUDE

setupbox는 host이다.

RPF Check (Reverse Path Forwarding) -> unicast routing table lookup

glop의 단점:
AS 번호를 가져야만 사용가능하다.
최대 256개 밖에 못 쓴다.

01:00:5e:(0~7)x:xx:x

IGMPv2의 Router Alert option은 IP 패킷이 지나가는 자리에 전부 라우터들이 그 패킷을 처리하도록 강제하기에 유용하다

Multicast router가 IGMP message를 듣게 하는 방법 두 가지
Router Alert 옵션을 IP 패킷에 달기
암암리에 all multicast groups들에 join하기

-TCP-
TCP의 특징
1. connection-oriented
2. reliable
3. inorder delivery
4. full-duplex

MSS = MTU - 40B
MSS = MTU - IP header (20B) - TCP header (20B)

-TCP header-
Source port, dest port
Sequence number
Ack number
Header length (In units of 4 bytes)
8 flags
Window size
Checksum
Urgent pointer
Options

flag SIN=1
-> data 없는 header만 온다.
ISN, Window size, MSS

Window size: flow control이 가능
socket buffer: window process가 항상 가동되는 것이 아니기에 임시 저장 공간이 필요
MSS: A, B 서로가 같은 path를 이용할 확률이 매우 높기 때문

RST:
1. 다 보낸 것을 서버가 알면, 그냥 reset해버림
2. port에 아무도 안 살면 그냥 reset해 버림

ACK=0은 처음에 SYN Segment뿐이다.

SYN Segment:
sequence number = ISN
ACK number = 0

SYN_SENT -> 3-way-handshake를 하고 있는 중이다.
ESTABLISHED -> 정상적 연결
TIME_WAIT -> 첫 번째 FIN, 마지막 ACK를 보냄

signal packet -> len=0이다.
WS=256 -> ACK에 표시가 안되어 있어도 사용가능?
data는 [FIN, ACK]에서 seq#, ack#에서 1씩 뺀 값이다.

retransmit이 오는 이유:
ACK가 오지 않아서 보내는 것이다. (ACK 죽었든, Segment가 죽었든 어차피 ACK가 안온다.)

fast retransmit을 보내는 이유:
Segment가 죽어서 보내는 것이다.

button "X"가 눌릴 때, 일단 FIN이 처음으로 나가고, 그다음에 RST이 나간다.

SLE = SACK의 일종이다.

Nagle`s algorithm:
ACK#의 간격 차이가 다르면 Nagle`s algorithm이 작동한 것이라 볼 수 있다.

half-close: TCP 연결에서 A->B 또는 B->A 둘 중 하나가 끊어진 경우
half-open: 두 TCP 중 하나가 그냥 집에 가버림. -> 이미 하나가 죽었는데 그걸 모른다.

SYN Segment가 가다가 죽거나, SYN Segment에 대한 ACK Segment가 죽었다.
-> Timeout of connection establishment

tcp.options==sack

TIME_WAIT 상태가 필요한 이유?
1. 패킷 전송이 지연되고, 그 사이에 이미 다른 연결이 성립되고 난 뒤에 지연된 패킷이 도착한다면, 지연 패킷과 새로운 패킷 간의 구분이 어려워지기 때문이다. 이는 TCP의 reliability를 해친다.
2. FIN을 하는 과정에서 마지막 ACK가 유실되지 않을 것이라는 보장이 없다. 만일 마지막 ACK가 유실되면 상대는 FIN을 재전송하게 된다. 그렇기에 TIME_WAIT 사애에서 2MSL 만큼 기다리면서 올지도 모르는 FIN 패킷을 기다려야하는 것이다.

netstat -a -t

부지런한 응용
+ delayed ACK algorithm
+ window update rule
-> 2 패킷에 대한 하나의 ACK 패킷

TCP
TCP Retransmit : ACK가 예상에 비해 너무 안옴
Fast Retransmit : 똑같은 ACK가 3개
Delayed ACK algorithm : ACK를 최대한 늦게 보낸다.
Window update rule : 수신 측의 window가 2MSS 이상이 되면 강제로 ACK가 나간다.
Nagle`s algorithm : 송신 측에서 MSS의 단위로 보냄 (small segment를 안 만드려고)
Persist timer : window size 0인 수신자
Silly Window Syndrome : window size가 2MSS 이상이 안되면 0으로 표시
Keepalive timer : 살았는지 죽었는지 확인

TCP는 MSS에 대한 정보를 option에 달아 붙인다.
-> MSS도 엄밀히 말하면 option이다.

CLOSE_WAIT는 상대편이 먼저 FIN을 하는 경우에 해당된다.

Expected = SEQ# && 다른 segment들이 전부 ACK됨
-> 200 ms까지 ACK를 Delay시키다가, 다음 segment가 없으면 그냥 ACK를 보냄
Expected = SEQ# && 다른 segment의 ACK가 delay되고 있음
-> 즉시 그 두개의 segment에 대한 cumulative ACK를 보낸다.
Expected < SEQ#, GAP이 생겼다.
-> 즉시 Expected SEQ#에 대한 duplicated ACK를 보낸다.
부분적으로, 혹은 완전히 GAP이 매워졌다.
-> 즉시 lower end of gap으로 시작하는 segment를 달라는 ACK를 보낸다.
Expected > SEQ#
-> 즉시 다음에 받을 segment를 달라는 ACK를 보낸다.
-> 받은 segment는 버린다.

persist timer가 돌고 있다는 것은 outstanding packet이 없다는 의미이다.
-> 재전송할 것이 없다
->전부 ACK 받았다.
이때 persist timer는 그냥 RTO를 사용한다. (재전송 할 것이 없으므로!)
-> 두 개의 타이머가 따로 존재해야 할 이유가 없다.

Persist timer의 prove:
data가 1B이다. 수신측이 보낸 ACK number와 일치, prove에 대한 ACK가 정상적으로 delay된다.

Keepalive timer의 prove:
data가 없다. 수신측이 보낸 ACK number보다 작다. prove에 대한 ACK가 정상적으로 delay되지 않고, 즉시로 ACK 패킷이 나간다.

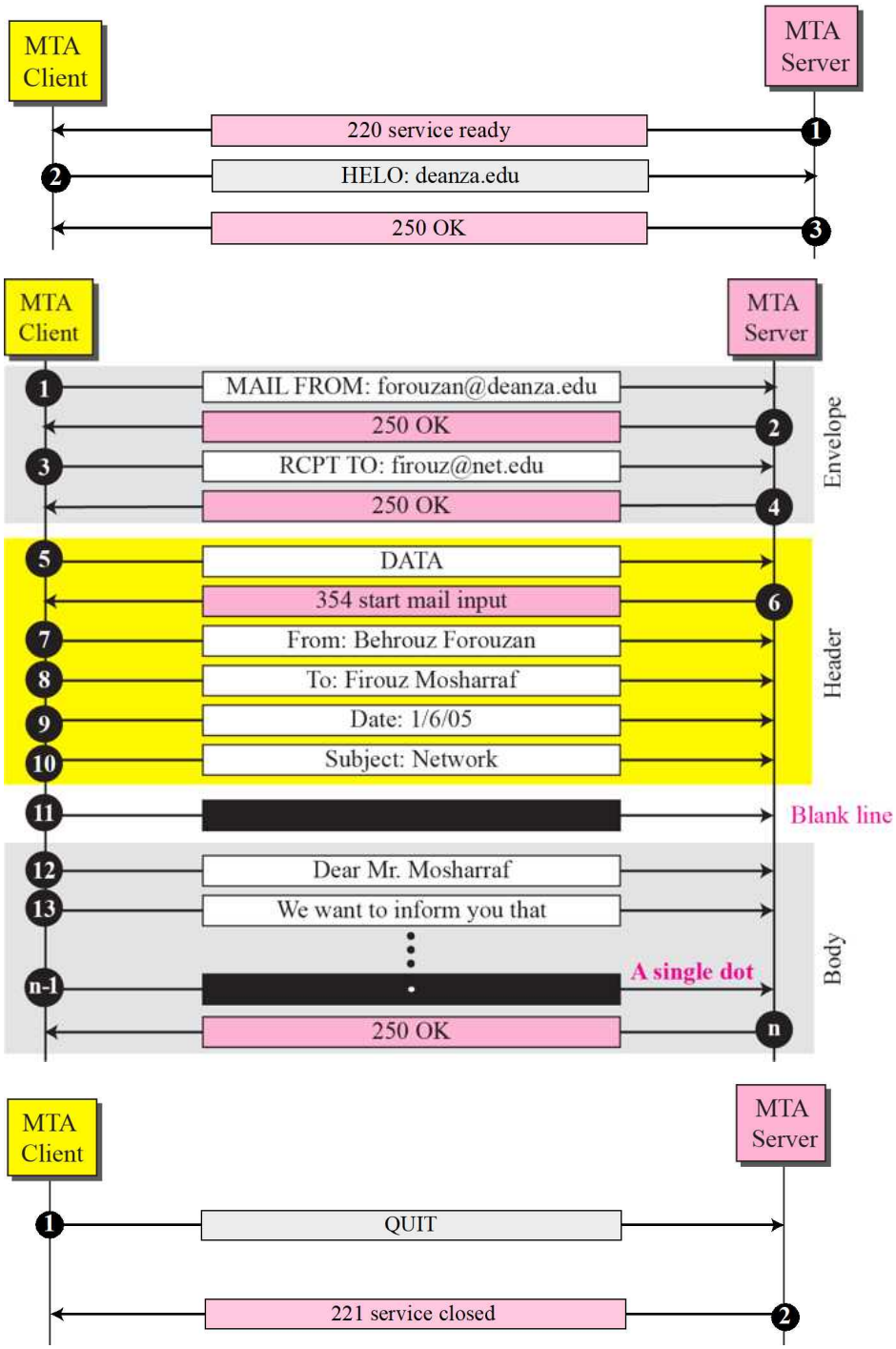
Retransmit: ACK가 와야 할 것이 아직 오지 않았다.
Persist: 받을 것은 전부 받은 상태이다.

[SYN]: seq = 0, ack = 0
[SYN, ACK] seg = 0, ack = 1
[ACK] seg = 1, ack = 0

[FIN, ACK] seg = x, ack = y
[ACK] seg = y, ack = x+1
[FIN, ACK] seg = y, ack = x+1
[ACK] seg = x+1, ack = y+1

TCP의 MSS option은 4B이다.

-SMTP-



telnet spam.korea.ac.kr 25
-> TCP 연결을 보여준다.

client가 FIN을 보낸 다음에 server가 221: bye를 보내면,
-> client가 RST을 보내버린다.

telnet (Mail server DNS) 25

MAIL FROM:
:가 붙는다!

MTA는 호스크가 정확히 어디에 있는지 모르기 때문에,
(host의 IP주소가 계속 바뀌기 때문에)
MTA를 사용하지를 못한다.
->다른 프로토콜을 사용한다.

MIME: Type에 없으면 Application/Octet-stream에 넣어버린다.

DNS: 53
SMTP: 25
POP3는 port번호가 110이다.

메일 쓸 때 QUIT까지 해줘라
SMTP에서의 VRFY 명령어는 그 이메일이 그 메일서버의 메일 박스에 있는지 없는지 여부를 알 수 있다. 그러나 이것을 허용하지 않는 이유는 헤커가 정보를 빼내갈 위험이 있기 때문이다.

Email 원본을 보자!

Pop3
telnet (domain name) 101
user (username)
pass (password)
retr 1
quit

-SNMP-
IGMP에서 체널을 바꾸는데, 자꾸 SNMP 패킷이 나간다.
-> KT

SNMP
MIB
SMI

netstat -p udp

1.3.6.1.2.1.7.5 : udp socket
1.3.6.1.2.1.7.5.(1 or 2) : udp socket

2	2
3	3
5	2
7	2
8	3
9	3

1.3. ()
1.3.2.8-> index column이 8를 사용하는 row의 2번째 column의 값이 뭐니?->3

UDP: 1.3.6.1.2.1.7

SNMP의 OID를 사용할 때에는 the OID of a row in the table을 사용한다.
즉, the OID of the table과 the OID of the table entry 중 the OID of the table entry 을 선택해야 한다.
왜 row value를 안쓰고 column index를 쓰냐?
-> table entry가 유동적으로 더해지거나 빼질 수 있기 때문이다.
-> 그렇기에 어디에 원하는 entry가 있는지 알 수가 없다.

SNMP에서 [2, 2]는 행2, 열2 를 의미한다.
BER Encoding에서는 10진수로
First number * 40 + second number 한 것을 16진수로 바꾼다
그래서 1.3은 2b로 표시된다.