2026/10/14, 2026 네트워크 세미나

TCP/IP 프로토콜

-20장 TELNET 및 SSH, 21장 FTP와 TFTP-

송 민 희(minhee@pel.sejong.ac.kr)

세종대학교 프로토콜공학연구실

목 차

- 보충
- 원격 로그인
- 파일 전송

목 차

- 보충
- 원격 로그인
- 파일 전송

보충

- TELNET 실습
  - 실습 환경
    - Client: Windows Powershell
    - Server: Ubuntu
  - Windows TELNET 클라이언트 활성화

```
PS C:₩WINDOWS₩system32> dism /online /Enable-Feature /FeatureName:TelnetClient

배포 이미지 서비스 및 관리 도구
버전: 10.0.26100.8972

이미지 버전: 10.0.26300.9457

기능을 사용하도록 설정하는 중
[==========================100.0%==========================]
작업을 완료했습니다.
PS C:₩WINDOWS₩system32> _
```

보충

- TELNET 실습
  - 원격 로그인 (1/4)
    - TELNET 접속
      - TCP 23번 포트로 접속 요청

```
PS C:\WINDOWS\system32> telnet localhost 23
연결 대상 localhost...
```

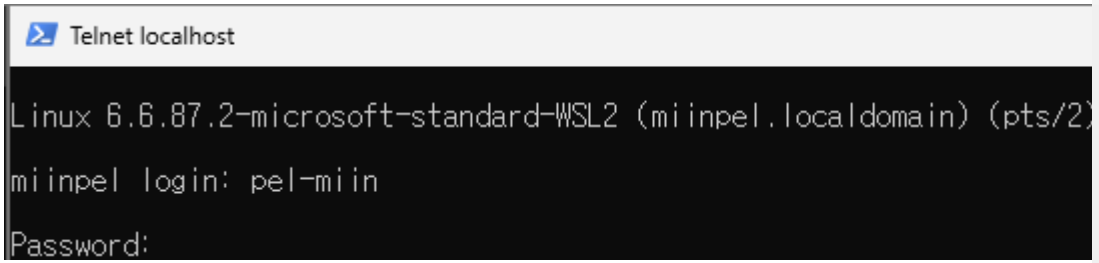
      - TCP 3-Way Handshake를 통한 클라이언트와 서버 간 연결 설정

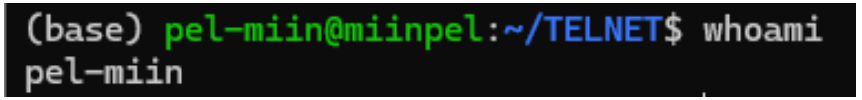
frame.number >= 47 && frame.number <= 49

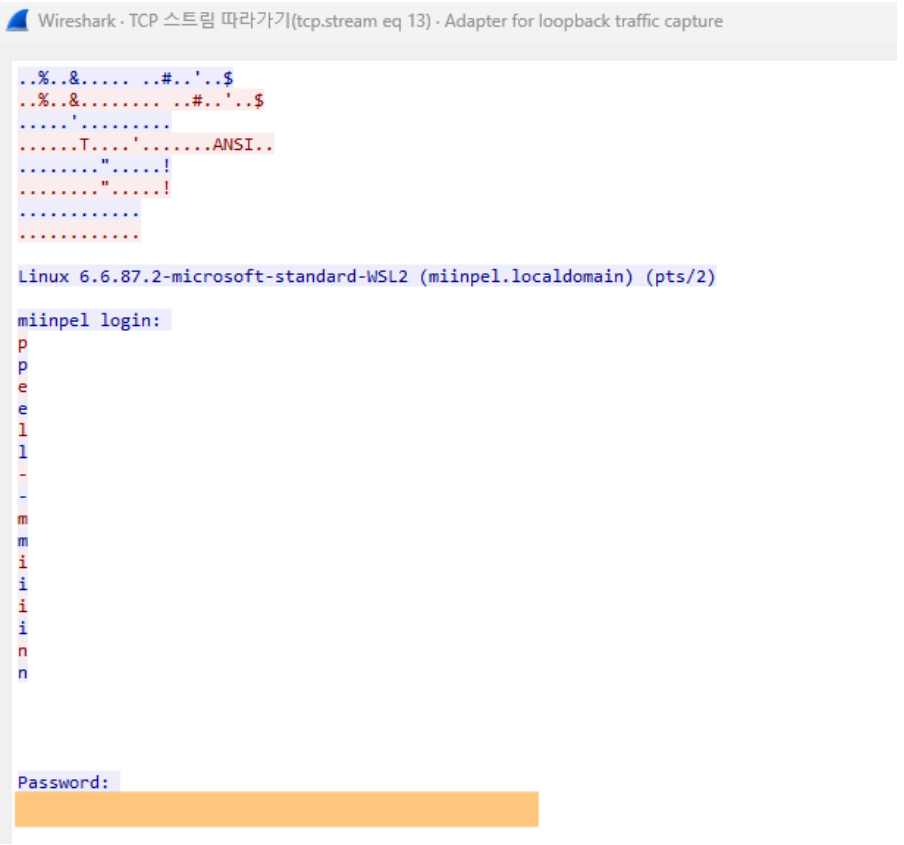
No.	Time	Source	Destination	Protocol	Lengt	Info
47	14.283091200	127.0.0.1	127.0.0.1	TCP	56	56904 → 23 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
48	14.283157100	127.0.0.1	127.0.0.1	TCP	56	23 → 56904 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
49	14.283174700	127.0.0.1	127.0.0.1	TCP	44	56904 → 23 [ACK] Seq=1 Ack=1 Win=65280 Len=0

보충

- TELNET 실습
  - 원격 로그인 (2/4)
    - TELNET 접속
      - WSL 내 사용자 이름과 비밀번호로 로그인

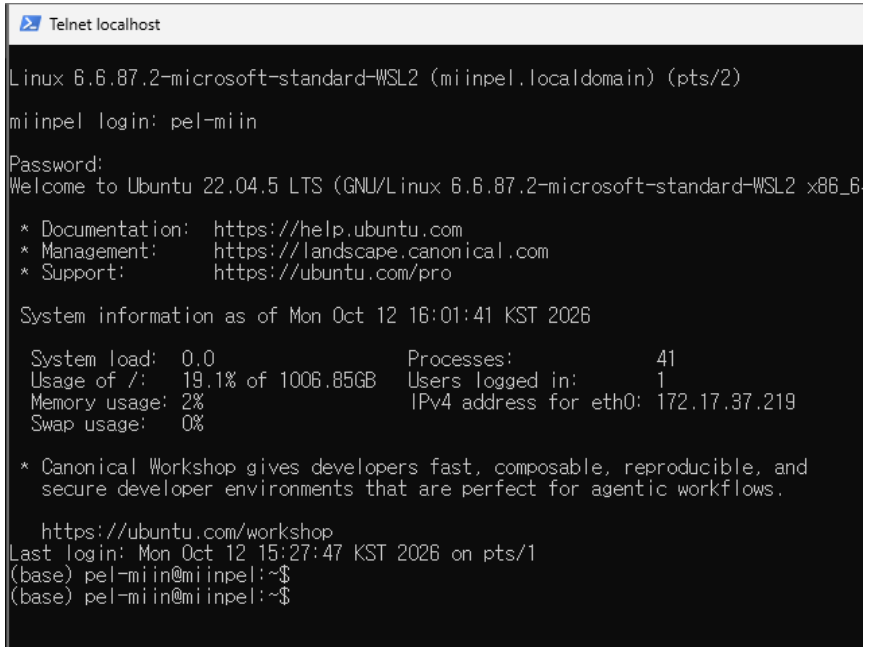






보충

- TELNET 실습
  - 원격 로그인 (3/4)
    - TELNET 접속
      - 로그인 성공




```
Password:


Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.6.87.2-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Mon Oct 12 16:01:41 KST 2026

  System load:  0.0                Processes:             41
  Usage of /:   19.1% of 1006.85GB   Users logged in:       1
  Memory usage: 2%                 IPv4 address for eth0: 172.17.37.219
  Swap usage:   0%

 * Canonical Workshop gives developers fast, composable, reproducible, and
   secure developer environments that are perfect for agentic workflows.

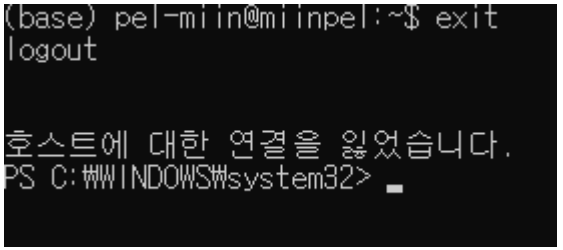
   https://ubuntu.com/workshop
Last login: Mon Oct 12 15:27:47 KST 2026 on pts/1
.[?2004h(base) pel-miin@miinpel:~$
.[?2004l
..[?2004h(base) pel-miin@miinpel:~$
```

보충

- TELNET 실습
  - 원격 로그인 (4/4)
    - 원격 명령어 수행



    - 종료



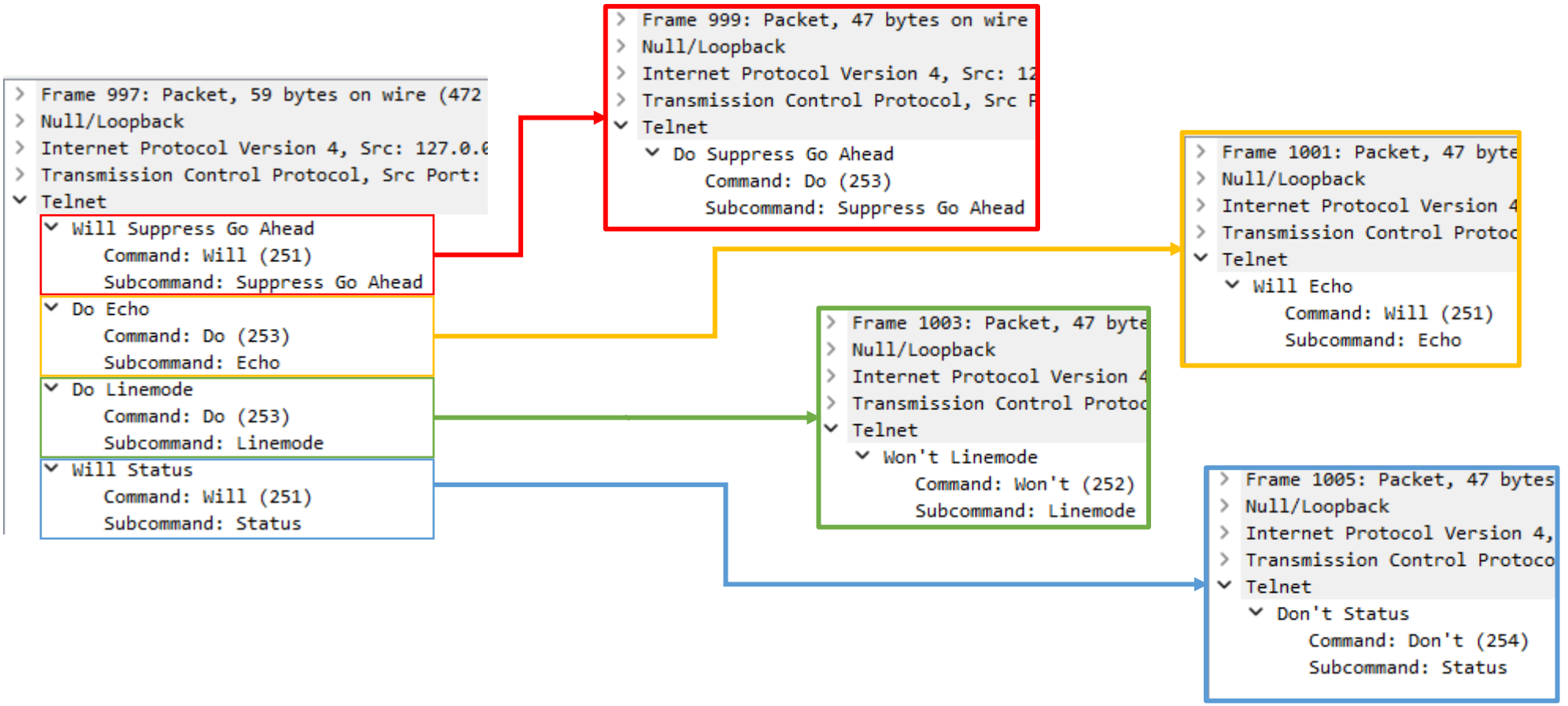
보충

- TELNET 실습
  - 옵션 협상 (1/4)
    - 연결 설정 후 4가지 조건에 대해 (자동으로) 협상 진행 (1/2)

```
> Frame 997: Packet, 59 bytes on wire (472 bits), 59 bytes captured (472 bits) on interface \Device\NPF_Loopback, id 0
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 23, Dst Port: 56763, Seq: 37, Ack: 50, Len: 15
v Telnet
  v Will Suppress Go Ahead
      Command: Will (251)
      Subcommand: Suppress Go Ahead
  v Do Echo
      Command: Do (253)
      Subcommand: Echo
  v Do Linemode
      Command: Do (253)
      Subcommand: Linemode
  v Will Status
      Command: Will (251)
      Subcommand: Status
  v Do Remote Flow Control
      Command: Do (253)
      Subcommand: Remote Flow Control
```

보충

- TELNET 실습
  - 옵션 협상 (2/4)
    - 연결 설정 후 4가지 조건에 대해 (자동으로) 협상 진행 (2/2)



보충

- TELNET 실습
  - 옵션 협상 (3/4)
    - 파이썬 코드를 통한 협상 재현 (1/2)

```
= RESTART: C:₩Users₩minhe₩Desktop₩PEL₩세미나₩TCP,IP 프로토콜 - 20, 21장₩실습₩tel
net_manual.py
TELNET 서버 연결 성공
초기 수신: ff fb 25 ff fb 26 ff fd 18 ff fd 20 ff fd 23 ff fd 27 ff fd 24
Enter를 누르면 DO ECHO를 전송합니다.
전송: IAC DO ECHO
HEX: FF FD 01
서버 응답: ff fb 01
Enter를 누르면 연결을 종료합니다.
```

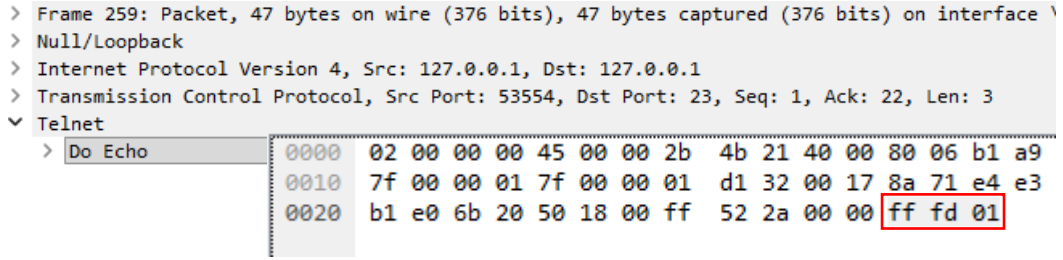
tcp.port == 23

No.	Time	Source	Destination	Protocol	Lengt	Info
26	4.203573800	127.0.0.1	127.0.0.1	TCP	56	53554 → 23 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK
27	4.203610100	127.0.0.1	127.0.0.1	TCP	56	23 → 53554 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495
28	4.203623300	127.0.0.1	127.0.0.1	TCP	44	53554 → 23 [ACK] Seq=1 Ack=1 Win=65280 Len=0
35	4.206973700	127.0.0.1	127.0.0.1	TELNET	65	Will Authentication Option, Will Encryption Option, Do Termi
36	4.206981500	127.0.0.1	127.0.0.1	TCP	44	53554 → 23 [ACK] Seq=1 Ack=22 Win=65280 Len=0
259	117.344174400	127.0.0.1	127.0.0.1	TELNET	47	Do Echo
260	117.344189200	127.0.0.1	127.0.0.1	TCP	44	23 → 53554 [ACK] Seq=22 Ack=4 Win=65280 Len=0
263	117.344600600	127.0.0.1	127.0.0.1	TELNET	47	Will Echo
264	117.344607800	127.0.0.1	127.0.0.1	TCP	44	53554 → 23 [ACK] Seq=4 Ack=25 Win=65280 Len=0

[표 2] TELNET 옵션

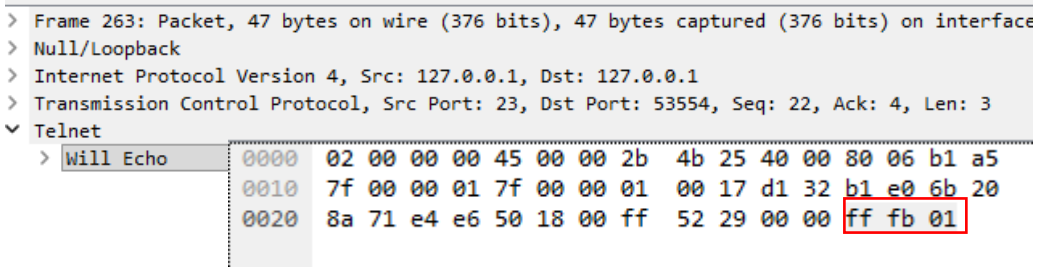
보충

- TELNET 실습
  - 옵션 협상 (4/4)
    - 파이썬 코드를 통한 협상 재현 (2/2)
      - 협상 요청



제어 문자	10 진수	16 진수	의미
SE	240	F0	서브옵션 종료
SB	250	FA	서브옵션 시작
WILL	251	FB	옵션 활성화 동의
WONT	252	FC	옵션 활성화 거절
DO	253	FD	옵션 사용 요청 승인
DONT	254	FE	옵션 사용 요청 거절
IAC	255	FF	다음 문자를 제어 문자로 해석

      - 협상 응답



코드값	옵션
1	Echo

보충

- TELNET 실습
  - 서브 옵션 협상

```
> Frame 995: Packet, 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF_Loopback, id 0
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 56763, Dst Port: 23, Seq: 40, Ack: 37, Len: 10
v Telnet
  v Suboption Terminal Type
      Command: Suboption (250)
    v Subcommand: Terminal Type
        Here's my Terminal Type
        Value: ANSI
  v Suboption End
      Command: Suboption End (240)
```

제어 문자	10 진수	16 진수	의미
SE	240	F0	서브옵션 종료
SB	250	FA	서브옵션 시작
WILL	251	FB	옵션 활성화 동의
WONT	252	FC	옵션 활성화 거절
DO	253	FD	옵션 사용 요청 승인
DONT	254	FE	옵션 사용 요청 거절
IAC	255	FF	다음 문자를 제어 문자로 해석

코드값	옵션
1	Echo

목 차

- 원격 로그인
- 파일 전송

원격 로그인

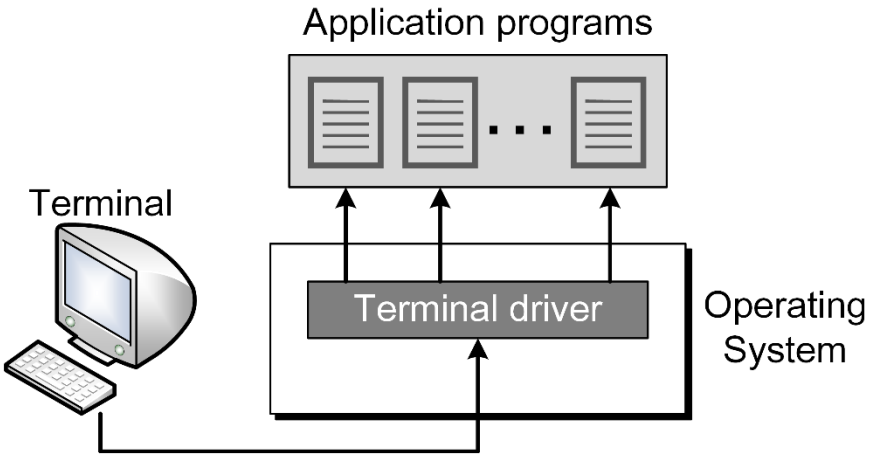
- 시분할 환경(Time Sharing System)
  - 정의
    - 하나의 컴퓨터 시스템 처리 시간을 짧은 시간 단위로 나누어 여러 사용자가 동시에 시스템을 이용할 수 있도록 할당하는 환경
  - 특징
    - 여러 사용자가 하나의 중앙 시스템과 CPU·메모리·입출력 장치 등의 자원을 공동 사용
    - 빠른 자원 전환을 통한 독립적 사용 환경의 체감 제공
    - 터미널을 통한 명령 입력 및 처리 결과 확인

원격 로그인

- 로그인(Login) (1/3)
  - 정의
    - 사용자가 시스템이나 서비스에 접근하기 위해 자신의 사용자 계정을 식별하고 인증하는 과정
  - 특징
    - 사용자별 작업 환경과 시스템 자원에 대한 접근 권한 관리 필요
    - 시분할 환경에서는 여러 사용자가 하나의 중앙 시스템을 공동으로 사용하므로 사용자 구분 필요
  - 분류
    - 로컬 로그인(Local Login)
    - 원격 로그인(Remote Login)

원격 로그인

- 로그인 (2/3)
  - 로컬 로그인
    - 정의
      - 사용자가 자신이 사용하는 시스템의 터미널을 통해 해당 시스템에 직접 로그인하는 방식

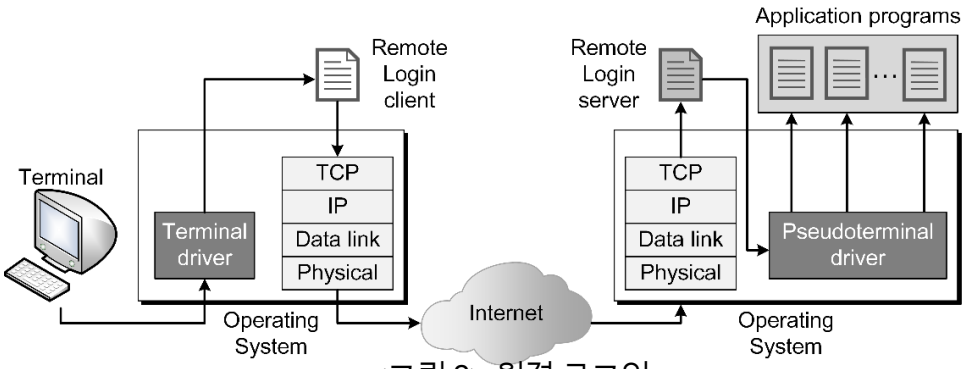


<그림 1> 로컬 로그인

    - 동작 과정
      1. 사용자가 로컬 터미널에서 로그인 정보 입력
      2. Terminal Driver가 입력을 수신하여 운영체제에 전달
      3. 운영체제가 로그인 프로그램 등을 통해 사용자 인증 수행
      4. 인증 성공 후 사용자가 입력한 명령을 해당 응용 프로그램이나 유틸리티에 전달
      5. 명령 처리 결과를 사용자 터미널에 출력

원격 로그인

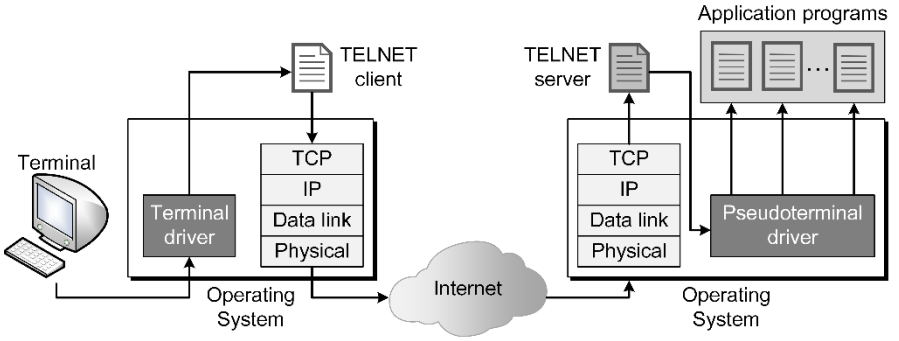
- 로그인 (3/3)
  - 원격 로그인
    - 정의
      - 사용자가 네트워크를 통해 원격 시스템에 접속하여 해당 시스템의 명령 실행 환경과 자원을 이용하는 로그인 방식
    - 동작 과정
      1. 사용자가 로컬 터미널에서 로그인 정보 및 명령 입력
      2. 입력 내용이 원격 로그인 클라이언트로 전달
      3. 네트워크를 통해 원격 시스템의 원격 로그인 서버로 전송
      4. 원격 시스템이 사용자 인증 및 입력을 처리
      5. 처리 결과를 네트워크를 통해 사용자에게 반환
    - 프로토콜
      - TELNET
      - SSH



<그림 2> 원격 로그인

원격 로그인

- TELNET (Teletype Network)
  - 정의
    - TCP/IP 네트워크를 통해 원격 시스템에 로그인하여 명령을 실행할 수 있도록 하는 응용 계층의 원격 로그인 프로토콜
  - 특징
    - 전송 계층에서 TCP 기반 통신
    - 클라이언트-서버 구조
      - 서버 기본 포트 23번 사용
    - 문자(Text) 기반의 원격 터미널 환경 제공
    - 서로 다른 시스템 간 원격 터미널 접속 지원
    - 사용자 인증정보 및 데이터의 평문 전송으로 보안 취약



<그림 3> TELNET을 통한 원격 로그인

원격 로그인

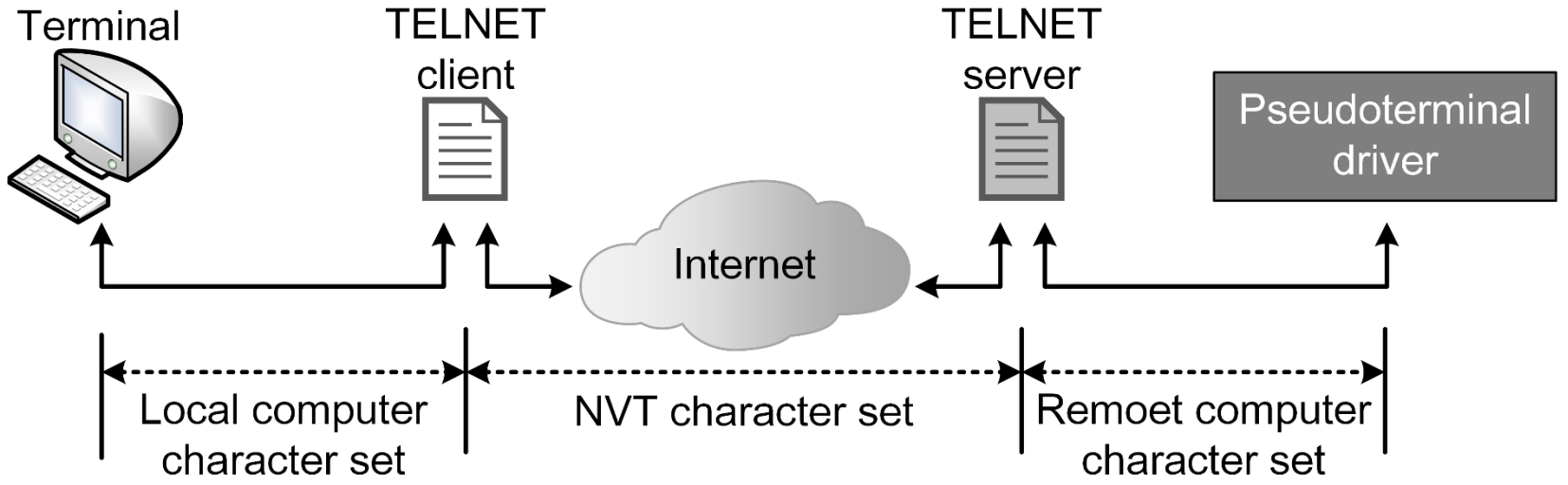
- TELNET
  - 동작 메커니즘
    - 네트워크 가상 터미널(NVT, Network Virtual Terminal)
    - 끼워넣기(Embedding)
    - 옵션(Option)
    - 서버 제어(Server Control)
    - 대역외 신호(Out-of-Band Signaling)

원격 로그인

- TELNET
  - 네트워크 가상 터미널(NVT) (1/4)
    - 정의
      - 서로 다른 컴퓨터와 운영체제 간의 터미널 표현 차이를 해결하기 위해 TELNET에서 사용하는 표준화된 가상 터미널 형식
    - 필요성
      - 운영체제마다 문자 및 제어 문자의 표현·해석 방식에 차이
        - e.g., EOF (End Of File) → DOS의 Ctrl+Z, UNIX의 Ctrl+D
      - 원격 시스템마다 다른 터미널 방식을 직접 고려해야 하는 호환성 문제 발생
      - 서로 다른 시스템 사이에서 사용할 공통된 데이터 및 명령 표현 방식 필요

원격 로그인

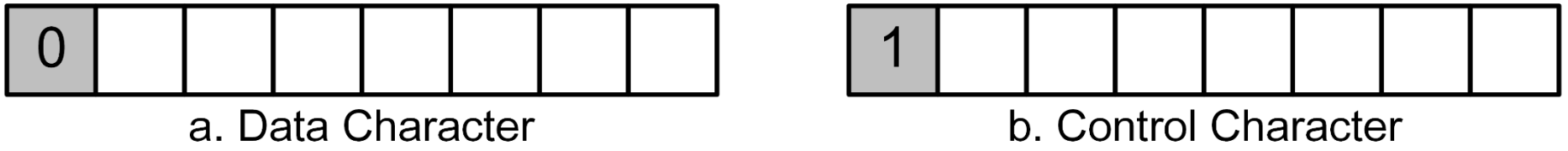
- TELNET
  - 네트워크 가상 터미널(NVT) (2/4)
    - 동작 과정
      1. TELNET Client가 로컬 시스템의 문자·명령을 NVT 형식으로 변환
      2. 변환된 NVT 데이터를 네트워크를 통해 TELNET Server로 전송
      3. TELNET Server가 NVT 형식을 원격 시스템이 이해할 수 있는 형식으로 변환



<그림 4> NVT의 동작 방식

원격 로그인

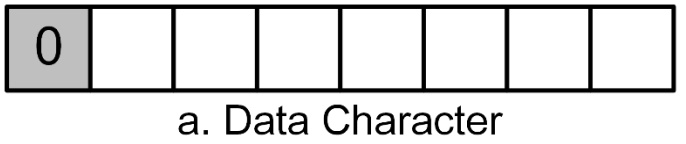
- TELNET
  - 네트워크 가상 터미널(NVT) (3/4)
    - 문자 집합
      - 데이터 문자(Data Character)
        - 실제 문자나 텍스트 등의 사용자 데이터 표현
        - 기본적으로 NVT-ASCII 사용
          - 8비트 중 최상위 비트(MSB, Most Significant Bit)는 0
          - 나머지 7비트는 US-ASCII와 동일
        - 8비트 전체를 데이터로 사용할 시 클라이언트와 서버 간 옵션 협상 필요



<그림 5> NVT에서 데이터 및 제어 문자의 형식

원격 로그인

- TELNET
  - 네트워크 가상 터미널(NVT) (4/4)
    - 문자 집합
      - 제어 문자(Control Character)
        - 클라이언트와 서버 사이에서 TELNET 동작을 제어하기 위한 명령 표현
        - TELNET 제어 명령은 특정 제어 코드로 정의
        - 데이터값 255([IAC])를 전송할 경우 [IAC] [IAC] 형태로 연속 전송하여 구분



<그림 5> NVT에서 데이터 및 제어 문자의 형식

[표 1] NVT 제어 문자

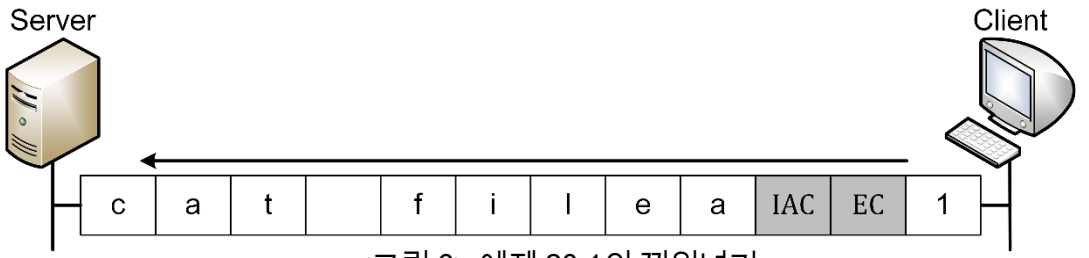
제어문자	10진수	2진수	의미	제어문자	10진수	2진수	의미
EOF	236	11101100	파일 끝(End of file)	EC	247	11110111	문자 삭제(Erase character)
EOR	239	11101111	레코드 끝(End of record)	EL	248	11111000	줄 삭제(Erase line)
SE	240	11110000	서브옵션 종료(Suboption end)	GA	249	11111001	전송 진행 허용(Go ahead)
NOP	241	11110001	동작 없음(No operation)	SB	250	11111010	서브옵션 시작(Suboption begin)
DM	242	11110010	데이터 표시(Data mark)	WILL	251	11111011	옵션 활성화 동의(Agreement to enable option)
BRK	243	11110011	중단(Break)	WONT	252	11111100	옵션 활성화 거절(Refusal to enable option)
IP	244	11110100	프로세스 중단 (Interrupt process)	DO	253	11111101	옵션 사용 요청 승인 (Approval to option request)
AO	245	11110101	출력 중단(Abort output)	DONT	254	11111110	옵션 사용 요청 거절(Denial of option request)
AYT	246	11110110	응답 여부 확인 (Are you there?)	IAC	255	11111111	다음 문자를 제어 문자로 해석 (Interpret the next character as control)

원격 로그인

- TELNET
  - 끼워넣기(Embedding) (1/2)
    - 정의
      - TELNET에서 데이터와 제어 명령을 하나의 TCP 연결로 전송하기 위해 제어 명령을 데이터 스트림 사이에 삽입하는 방식
    - 필요성
      - 단일 TCP 연결에서 데이터와 제어 명령의 구분 필요
        - TELNET은 클라이언트와 서버 사이에 하나의 TCP 연결 사용
        - 일반 데이터와 TELNET 제어 명령을 동일한 데이터 스트림으로 전송
        - 수신 측에서 두 종류의 정보를 구분할 수 있는 식별 방식 필요
    - 동작 방식
      - 일반 문자 데이터는 그대로 데이터 스트림에 삽입
      - TELNET 제어 명령 앞에는 [IAC]를 삽입
      - 수신 측은 [IAC] 이후의 문자를 일반 데이터가 아닌 TELNET 제어 명령으로 해석

원격 로그인

- TELNET
  - 끼워넣기 (2/2)
    - 예제 20.1



<그림 6> 예제 20.1의 끼워넣기

사용자가 TELNET을 이용해 원격 서버에 접속한 뒤, file1 파일의 내용을 확인하기 위해 다음 명령을 입력하려고 한다.

➢ cat file1

하지만 사용자가 실수로 1 대신 a를 입력하여 다음과 같이 입력했다.

➢ cat filea

이후 사용자는 Backspace를 눌러 a를 삭제하고 1을 다시 입력하였다.

TELNET의 끼워넣기 방식에 따라 서버로 전송되는 데이터 스트림을 나타내시오.

- cat filea [IAC][EC] 1
  - [IAC]는 뒤의 문자가 TELENT 제어 명령임을 표기
  - [EC]는 직전 문자를 삭제
  - 이후 새롭게 작성할 1을 추가

원격 로그인

- TELNET
  - 옵션(Option) (1/13)
    - 정의
      - TELNET에서 기본 기능 외에 사용할 수 있도록 정의된 추가 기능
    - 필요성
      - 터미널 종류와 처리 능력의 차이에 따른 기능 조정 필요
      - 기본 NVT 기능만으로 처리하기 어려운 추가 기능 지원
      - 필요한 기능만 선택적으로 사용 가능

원격 로그인

- TELNET
  - 옵션 (2/13)
    - 종류 (1/2)
      - Binary
        - 수신한 8비트 문자를 이진 데이터로 해석하여 전송
      - Echo
        - 서버가 클라이언트로부터 받은 문자를 다시 클라이언트에 반환
        - 클라이언트는 서버로부터 반환된 문자를 화면에 표시
      - Suppress Go-Ahead
        - 데이터 전송 순서를 알리는 [GA] 제어 문자의 사용 억제
        - 양방향 통신에서 불필요한 [GA] 전송을 생략
      - Status
        - 현재 TELNET 세션에서 사용 중이거나 사용 가능한 옵션의 상태 확인

[표 2] TELNET 옵션

코드값	옵션	의미
0	Binary	Interpret as 8-bit binary transmission
1	Echo	Echo the data received on one side to the other
3	Suppress go-ahead	Suppress go-ahead signals after data
5	Status	Request the status of TELNET
6	Timing mark	Define the timing marks
24	Terminal type	Set the terminal type
32	Terminal speed	Set the terminal speed
34	Line mode	Change to line mode

원격 로그인

- TELNET
  - 옵션 (3/13)
    - 종류 (2/2)
      - Timing Mark
        - 이전까지 전송된 데이터가 모두 처리되었음을 확인하기 위한 동기화 표시
      - Terminal Type
        - 클라이언트가 사용하는 터미널의 종류를 서버에 전달
        - 서버가 터미널 특성에 맞게 동작하도록 지원
      - Terminal Speed
        - 클라이언트 터미널의 송·수신 속도 정보를 서버에 전달
      - Line Mode (TELNET 모드에 대한 설명은 부록#2 참고)
        - 클라이언트가 입력한 내용을 한 줄 단위로 처리
        - 사용자가 한 줄 입력을 완료한 뒤 서버로 전달하여 불필요한 문자별 전송 감소
        - 로컬에서 일부 입력 편집을 처리할 수 있어 대화형 입력 효율 향상

[표 2] TELNET 옵션

코드값	옵션	의미
0	Binary	Interpret as 8-bit binary transmission
1	Echo	Echo the data received on one side to the other
3	Suppress go-ahead	Suppress go-ahead signals after data
5	Status	Request the status of TELNET
6	Timing mark	Define the timing marks
24	Terminal type	Set the terminal type
32	Terminal speed	Set the terminal speed
34	Line mode	Change to line mode

원격 로그인

- TELNET
  - 옵션 (4/13)
    - 옵션 협상(Option Negotiation) (1/6)
      - 정의
        - TELNET에서 클라이언트와 서버가 특정 옵션의 활성화 또는 해제 여부를 제어 명령을 통해 협의하는 과정
      - 특징
        - 서비스 시작 전 또는 통신 중 옵션 협상 가능
        - 클라이언트와 서버 양쪽 모두 협상 시작 가능
        - 대칭적 협상 구조(Symmetry)
          - 클라이언트와 서버 모두 옵션의 활성화를 제안하거나 요청 가능
          - 상대방의 옵션 지원 여부 및 사용 의사에 따라 수락 또는 거절

원격 로그인

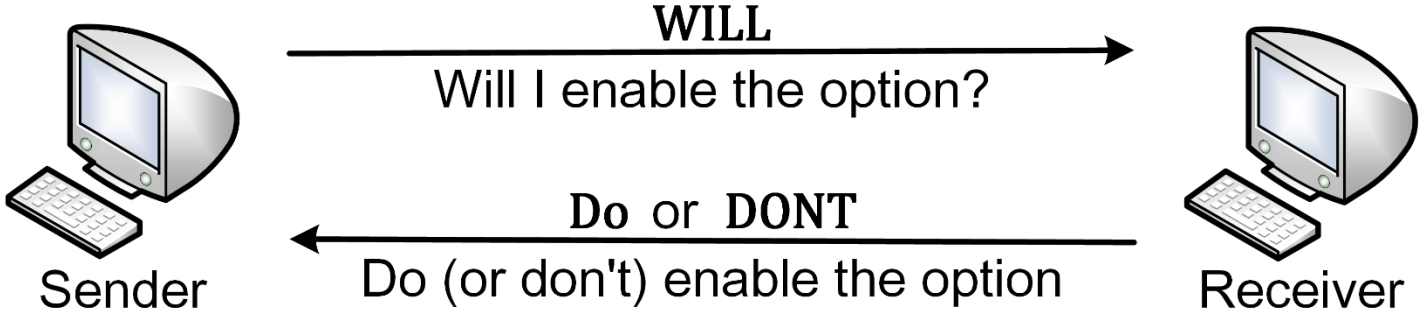
- TELNET
  - 옵션 (5/13)
    - 옵션 협상 (2/6)
      - 옵션 협상 제어 문자
        - [WILL]: 자신이 해당 옵션을 사용하겠다는 의사 표현
        - [WONT]: 자신이 해당 옵션을 사용하지 않겠다는 의사 표현
        - [DO]: 상대방에게 해당 옵션을 사용하도록 요청 또는 승인
        - [DONT]: 상대방에게 해당 옵션을 사용하지 않도록 요청 또는 거절

[표 3] TELNET 옵션 협상을 위한 NVT 문자 집합

문자	코드값	의미 1	의미 2	의미 3
WILL	251	활성화 제안	활성화 수락	-
WONT	252	활성화 거절	비활성화 제안	비활성화 수락
DO	253	활성화 승인	활성화 요청	-
DONT	254	활성화 거절	비활성화 승인	비활성화 요청

원격 로그인

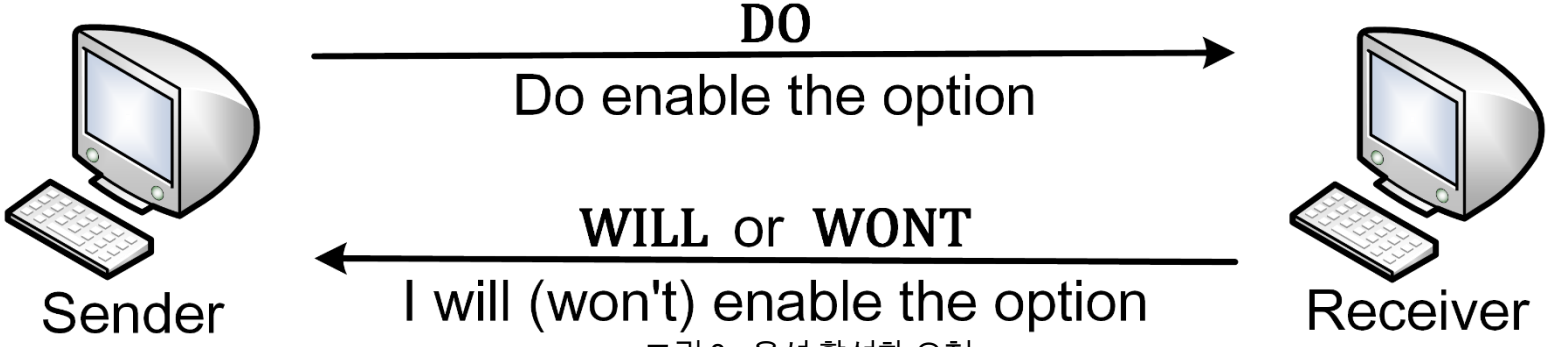
- TELNET
  - 옵션 (6/13)
    - 옵션 협상 (3/6)
      - 활성화 제안
        - 한쪽이 자신의 옵션 활성화를 제안
          - [WILL]: “내가 이 옵션을 사용해도 되는가?”
        - 상대방의 응답
          - [DO]: 활성화 제안 승인
          - [DONT]: 활성화 제안 거절



<그림 7> 옵션 활성화 제안

원격 로그인

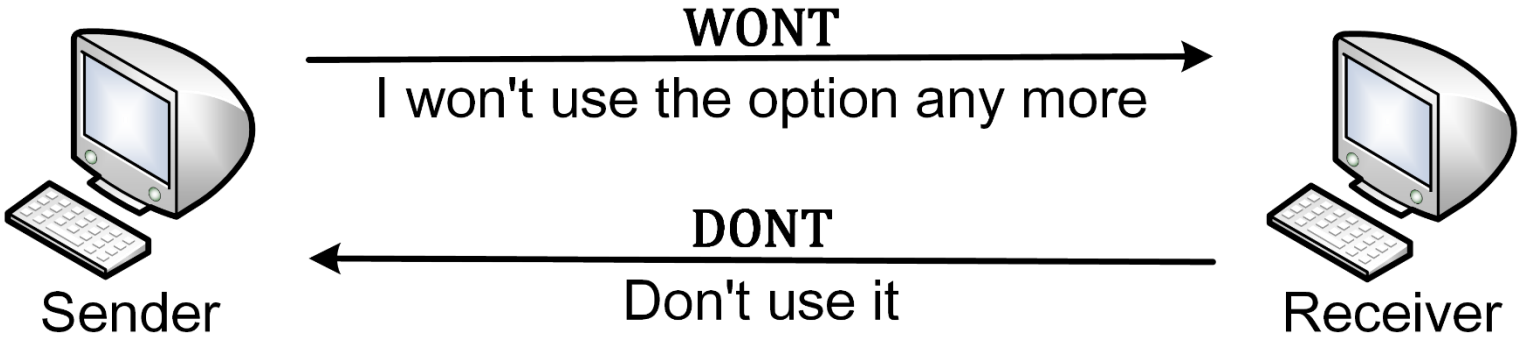
- TELNET
  - 옵션 (7/13)
    - 옵션 협상 (4/6)
      - 활성화 요청
        - 한쪽이 상대방에게 옵션 활성화를 요청
          - [DO]: “이 옵션을 사용해 달라”
        - 상대방의 응답
          - [WILL]: 활성화 요청 수락
          - [WONT]: 활성화 요청 거절



<그림 8> 옵션 활성화 요청

원격 로그인

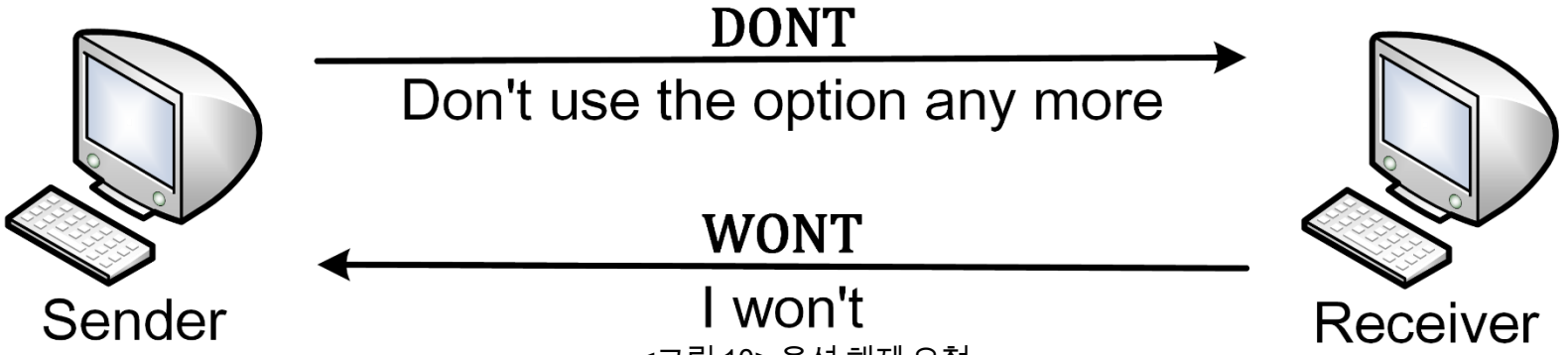
- TELNET
  - 옵션 (8/13)
    - 옵션 협상 (5/6)
      - 활성화 해제 제안
        - 한쪽이 자신이 사용하던 옵션을 더 이상 사용하지 않겠다고 통보
          - [WONT]: “이 옵션을 더 이상 사용하지 않겠다”
        - 상대방의 응답
          - [DONT]: 해제 제안 승인
          - 해제 제안 거절 불가능



<그림 9> 옵션 해제 제안

원격 로그인

- TELNET
  - 옵션 (9/13)
    - 옵션 협상 (6/6)
      - 활성화 해제 요청
        - 한쪽이 상대방에게 해당 옵션을 더 이상 사용하지 말 것을 요청
          - [DONT]: “이 옵션을 더 이상 사용하지 마라”
        - 상대방의 응답
          - [WONT]: 해제 요청 수락
          - 해제 요청 거절 불가



<그림 10> 옵션 해제 요청

원격 로그인

- TELNET
  - 옵션 (10/13)
    - 서브 옵션 협상(Suboption Negotiation)
      - 정의
        - TELNET에서 활성화된 옵션에 필요한 세부 정보나 설정값을 추가로 교환하는 협상 과정
      - 특징
        - 옵션 활성화 이후 필요한 추가 정보 전달
        - 터미널 종류, 속도 등 구체적인 설정값 교환
        - [SB]와 [SE]를 사용하여 서브옵션 데이터의 시작과 끝을 구분
        - 옵션 종류에 따라 문자열이나 숫자 등의 부가 정보 포함 가능

[표 4] TELNET 서브 옵션 협상을 위한 NVT 문자 집합

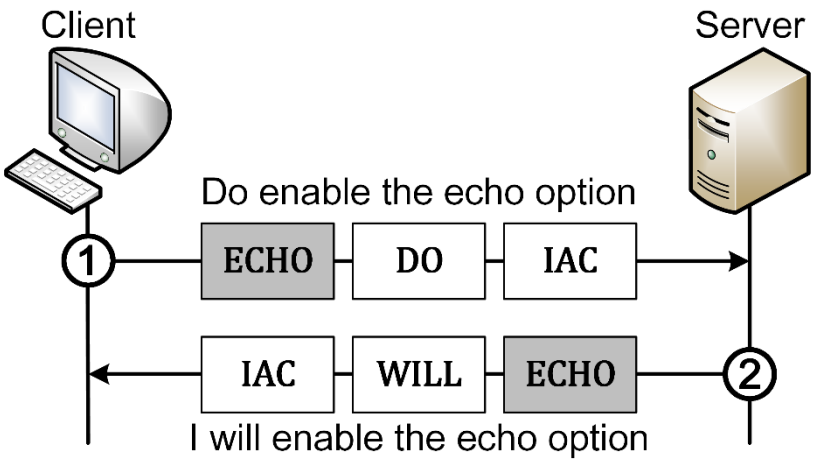
문자	10진수	2진수	의미
SE	240	11110000	서브옵션 종료(Suboption end)
SB	250	11111010	서브옵션 시작(Suboption begin)

원격 로그인

- TELNET
  - 옵션 (11/13)
    - 예제 20.2

TELNET 클라이언트는 사용자가 입력한 문자를 서버가 다시 클라이언트로 전송하도록 Echo 옵션을 활성화하려고 한다.클라이언트가 서버에 Echo 옵션의 활성화를 요청하고, 서버가 이를 수락하는 경우 주고받는 TELNET 제어 명령을 나타내시오.

1. 클라이언트 → 서버
   - [IAC] [DO] ECHO
     - [IAC]: TELNET 제어 명령임을 표시
     - [DO]: 상대방에게 해당 옵션의 사용을 요청
     - ECHO: 활성화할 옵션
2. 서버 → 클라이언트
   - [IAC] [WILL] ECHO
     - [WILL]: 해당 옵션을 사용하겠다는 응답
     - 서버가 Echo 옵션을 활성화



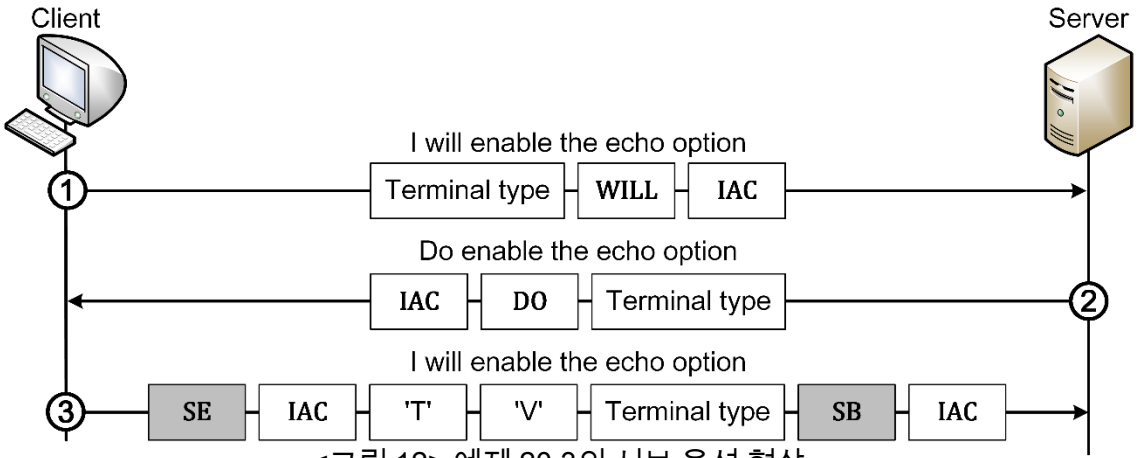
<그림 11> 예제 20.2의 Echo 옵션

원격 로그인

- TELNET
  - 옵션 (12/13)
    - 예제 20.3 (1/2)

TELNET 클라이언트가 서버와 Terminal Type 옵션을 활성화한 후, 자신의 터미널 종류가 VT임을 서버에 전달하려고 한다. 옵션 활성화부터 서브옵션 정보 전달까지 주고받는 TELNET 제어 명령을 나타내시오.

1. 클라이언트 → 서버 (옵션 활성화 제안)
   - [IAC] [WILL] TERMINAL – TYPE
     - [WILL]: 클라이언트가 Terminal Type 옵션을 사용하겠다는 제안
2. 서버 → 클라이언트 (활성화 승인)
   - [IAC] [DO] TERMINAL – TYPE
     - [DO]: 서버가 클라이언트의 Terminal Type 옵션 사용을 승인



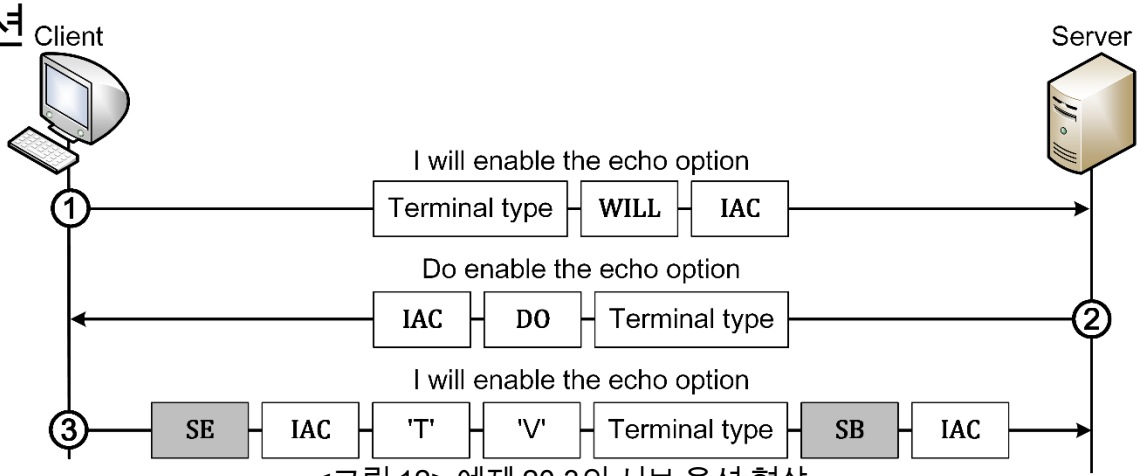
<그림 12> 예제 20.3의 서브 옵션 협상

원격 로그인

- TELNET
  - 옵션 (13/13)
    - 예제 20.3 (2/2)

TELNET 클라이언트가 서버와 Terminal Type 옵션을 활성화한 후, 자신의 터미널 종류가 VT임을 서버에 전달하려고 한다. 옵션 활성화부터 서브옵션 정보 전달까지 주고받는 TELNET 제어 명령을 나타내시오.

3. 클라이언트 → 서버 (터미널 종류 전달)
   - [IAC] [SB] TERMINAL – TYPE VT [IAC] [SE]
     - [SB]: 서브옵션 협상 시작
     - TERMINAL – TYPE: 대상 옵션
     - "VT": 실제 터미널 종류 정보
     - [SE]: 서브옵션 협상 종료



<그림 12> 예제 20.3의 서브 옵션 협상

원격 로그인

- TELNET
  - 서버 제어(Server Control) (1/4)
    - 정의
      - TELNET에서 사용자가 원격 서버에서 실행 중인 응용 프로그램의 동작을 제어할 수 있도록 특수 제어 명령을 전달하는 기능
    - 필요성
      - 원격 프로그램에 대한 중단·출력 취소·입력 수정 등의 제어 필요
      - 로컬에서 사용하는 입력을 원격 서버에서 동일한 의미로 처리 필요
        - e.g., Ctrl+C, Backspace
      - 사용자 입력을 TELNET 제어 문자로 변환하여 원격 서버에 전달

원격 로그인

- TELNET
  - 서버 제어 (2/4)
    - 제어 문자 (1/2)
      - [IP] (Interrupt Process)
        - 원격 서버에서 실행 중인 프로세스의 실행 중단
        - 로컬 환경의 Ctrl+C와 유사한 역할
      - [AO] (Abort Output)
        - 실행 중인 프로세스는 유지하면서 출력만 중단
        - 이후 처리 자체는 계속 수행

[표 5] 원격 서버에서 동작하는 프로그램을 제어하기 위해 사용되는 문자

문자	10진수	2진수	의미
IP	244	11110100	프로세스 중단(Interrupt process)
AO	245	11110101	출력 중단(Abort output)
AYT	246	11110110	응답 여부 확인(Are you there?)
EC	247	11110111	마지막 문자 삭제(Erase the last character)
EL	248	11111000	현재 줄 삭제(Erase line)

원격 로그인

- TELNET
  - 서버 제어 (3/4)
    - 제어 문자 (2/2)
      - [AYT] (Are You There?)
        - 원격 서버가 정상적으로 동작하고 있는지 확인
        - 서버는 수신 시 사용자에게 확인 가능한 응답 전송
      - [EC] (Erase Character)
        - 현재 입력에서 마지막 문자 삭제
        - Backspace와 유사한 기능
      - [EL] (Erase Line)
        - 현재 입력 중인 한 줄 전체 삭제

[표 5] 원격 서버에서 동작하는 프로그램을 제어하기 위해 사용되는 문자

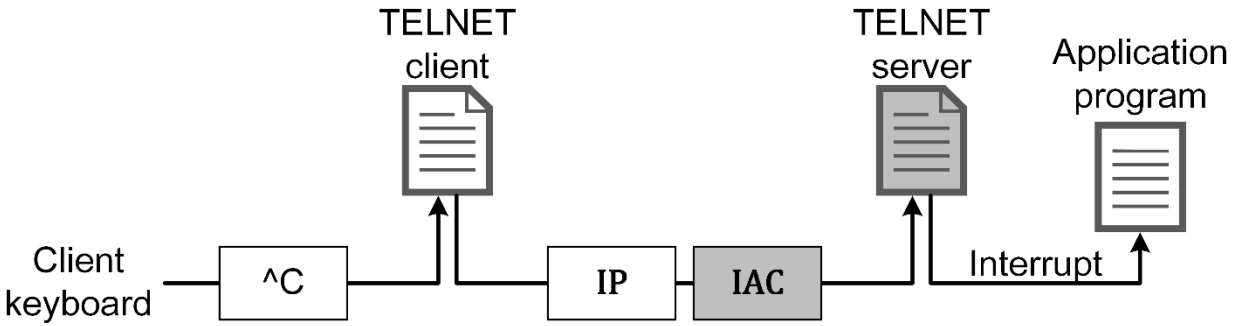
문자	10진수	2진수	의미
IP	244	11110100	프로세스 중단(Interrupt process)
AO	245	11110101	출력 중단(Abort output)
AYT	246	11110110	응답 여부 확인(Are you there?)
EC	247	11110111	마지막 문자 삭제(Erase the last character)
EL	248	11111000	현재 줄 삭제(Erase line)

원격 로그인

- TELNET
  - 서버 제어 (4/4)
    - 예제 20.4

사용자가 TELNET을 통해 원격 서버에서 프로그램을 실행하고 있다. 프로그램이 무한 루프에 빠져 실행을 중단하기 위해 사용자가 Ctrl+C를 입력하였다. TELNET에서 이 입력이 원격 서버의 프로그램을 중단시키는 과정을 나타내시오.

1. 사용자가 클라이언트에서 Ctrl+C 입력
2. TELNET Client가 이를 원격 프로세스 중단 명령인 [IP]로 변환
3. TELNET 제어 명령임을 나타내기 위해 [IAC]와 함께 전송
4. TELNET Server가 [IAC] [IP]를 수신
5. 원격 서버에서 실행 중인 응용 프로그램 Interrupt



<그림 13> 예제 20.4의 응용 프로그램을 인터럽트 하는 예시

원격 로그인

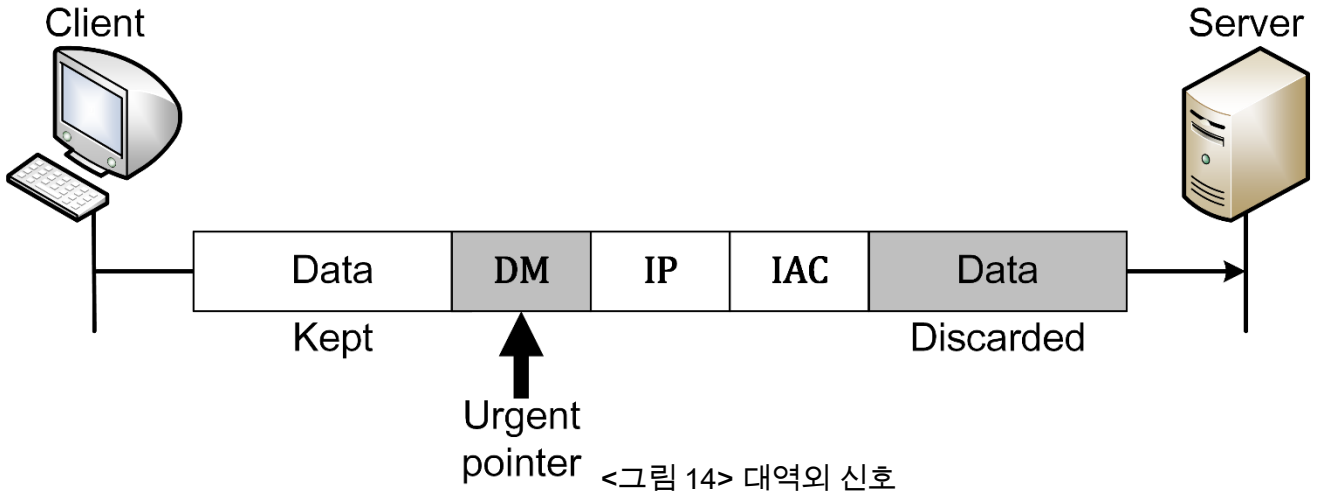
- TELNET
  - 대역외 신호(Out-of-Band Signaling) (1/2)
    - 정의
      - TELNET에서 긴급한 제어 명령을 일반 데이터 처리 순서와 관계없이 우선적으로 전달하기 위해 TCP의 긴급(Urgent) 메커니즘을 이용하는 방식
    - 필요성
      - 일반 데이터 처리가 불가능한 상황에서도 제어 명령 전달 필요
        - 원격 응용 프로그램의 무한 루프 등으로 입력 데이터 처리 불가
        - TCP 수신 버퍼가 가득 차 일반 데이터의 정상적인 처리 지연 가능
        - [IP]와 같은 긴급 제어 명령을 일반 데이터보다 우선적으로 처리할 필요

원격 로그인

- TELNET
  - 대역외 신호 (2/2)
    - 동작 과정
      1. TELNET 제어 명령을 데이터 스트림에 삽입
      2. [DM]을 긴급 데이터의 기준점으로 사용
      3. TCP의 Urgent Pointer를 이용해 DM 위치를 긴급하게 알림
      4. 수신 측은 긴급 상태에서 [DM] 이전의 처리되지 않은 데이터 폐기
      5. [DM]에 도달하면 긴급 상태를 종료하고 이후 데이터를 정상 처리

[표 6] 대역외 신호에서 사용되는 제어 문자

문자	10진수	2진수	의미
DM	242	11110010	데이터 표시(Data mark)



<그림 14> 대역외 신호

원격 로그인

- TELNET
  - 보안 문제
    - 사용자 ID, 비밀번호, 명령 및 응답 데이터가 평문으로 전송
    - 네트워크 패킷을 도청할 경우 인증정보와 통신 내용 노출
    - 별도의 강력한 암호화 및 서버 인증 기능 부재
    - 공개 네트워크 환경에서의 사용에 부적합

원격 로그인

- SSH (Secure Shell)
  - 정의
    - 안전하지 않은 네트워크에서 암호화된 통신 채널을 구성하여 원격 로그인, 명령 실행 등의 서비스를 안전하게 제공하는 응용 계층 프로토콜
  - 필요성
    - TELNET의 평문 전송에 따른 보안 취약성 보완
      - 사용자 ID 및 비밀번호 노출 가능
      - 명령 및 응답 데이터 도청 가능
    - 원격 시스템과의 통신에서 기밀성·무결성·인증 제공 필요

원격 로그인

- SSH
  - 버전
    - SSH-1
      - 초기 SSH 프로토콜
      - 여러 보안상 결함으로 인해 현재 사용 중단
        - CRC-32 기반 무결성 검사의 취약성
          - 암호학적 MAC이 아니라 CRC를 사용하여 패킷 변조 탐지 능력이 부족
        - 취약한 패킷 무결성 보호
          - 공격자가 암호문을 조작해도 이를 강하게 검증하기 어려운 구조
        - 구식 키 교환 및 암호 구조
          - 세션 키 설정 방식이 SSH-2보다 단순하고 낮은 보안성
    - SSH-2
      - SSH-1의 문제점을 개선한 현재 표준 방식
      - SSH-1과 호환 불가

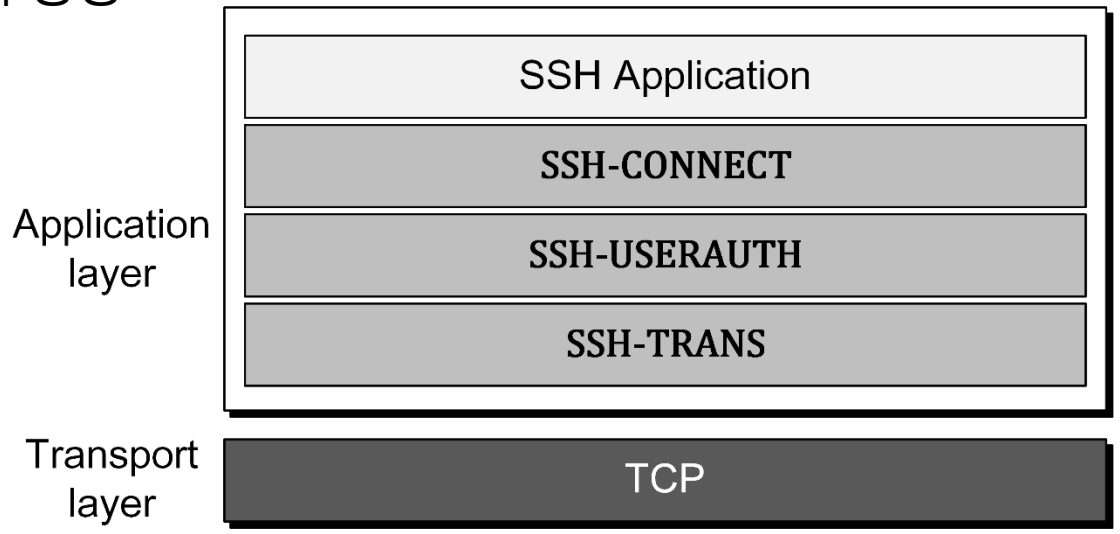
*순환 중복 검사(CRC, Cyclic Redundancy Check)
전송하거나 저장한 데이터에 오류가 발생했는지 확인하기 위해 데이터로부터 검사값을 계산하여 함께 전송하고, 수신 측에서 이를 다시 계산해 비교하는 오류 검출 방식

원격 로그인

- SSH
  - 특징
    - 일반적으로 TCP/IP 연결과 같은 신뢰 가능한 데이터 스트림 위에서 동작
    - 클라이언트-서버 구조
      - 서버 기본 포트 22번 사용
    - 문자(text) 기반의 원격 터미널 환경 제공
    - 서로 다른 시스템 간 안전한 원격 터미널 접속 지원
    - 암호화된 통신 채널 활용
    - 사용자 인증정보 및 데이터의 암호화 전송으로 보안성 제공
    - 3개의 주요 프로토콜 구성요소로 구성
      - SSH 전송 계층 프로토콜(SSH-TRANS)
      - SSH 사용자 인증 프로토콜(SSH-USERAUTH)
      - SSH 연결 프로토콜(SSH-CONNECT)

원격 로그인

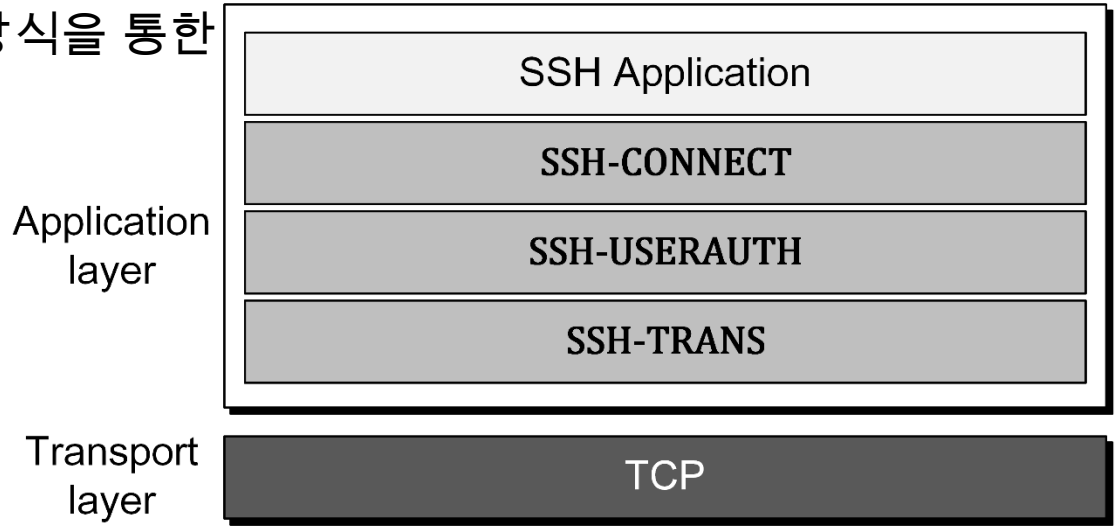
- SSH
  - 구성요소 (1/3)
    - SSH 전송 계층 프로토콜(SSH-TRANS)
      - 정의
        - TCP 연결 위에 안전한 SSH 통신 채널을 설정하는 프로토콜
      - 기능
        - 클라이언트와 서버 간 암호화 알고리즘 및 보안 매개변수 협상
        - 키 교환을 통한 암호화 키 생성
        - 서버 인증
        - 메시지의 기밀성 제공
        - 데이터의 무결성 보호
        - 선택적인 데이터 압축



<그림 15> SSH의 구성요소

원격 로그인

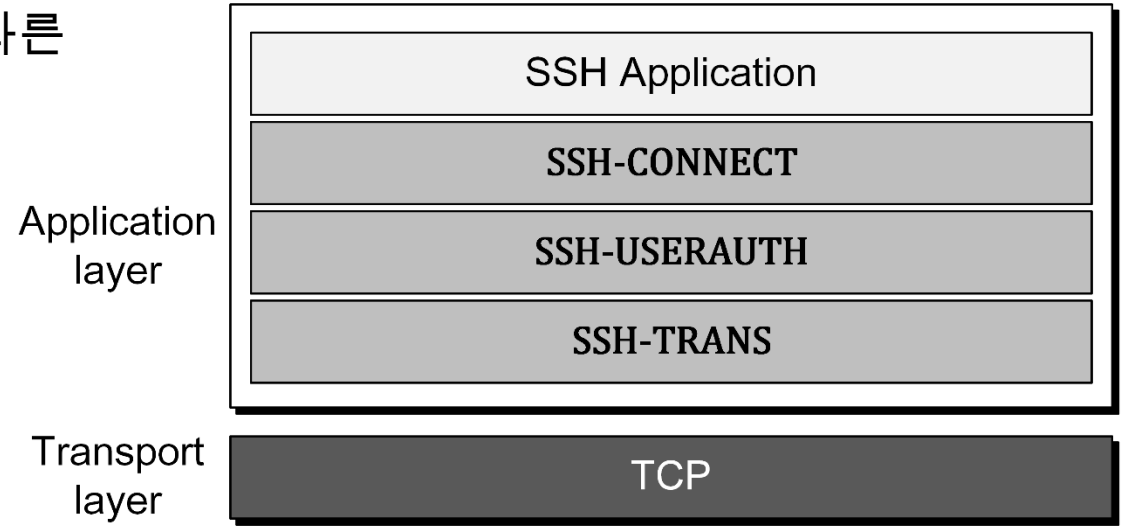
- SSH
  - 구성요소 (2/3)
    - SSH 인증 프로토콜(SSH-USERAUTH)
      - 정의
        - SSH-TRANS를 통해 안전한 채널이 생성된 후, 서버가 접속하려는 사용자를 인증하는 프로토콜
      - 기능
        - SSH-TRANS를 통해 생성된 암호화 채널 위에서 수행
        - 비밀번호, 공개키 등의 방식을 통한 사용자 인증
        - 인증 성공 후 SSH 서비스 사용 가능



<그림 15> SSH의 구성요소

원격 로그인

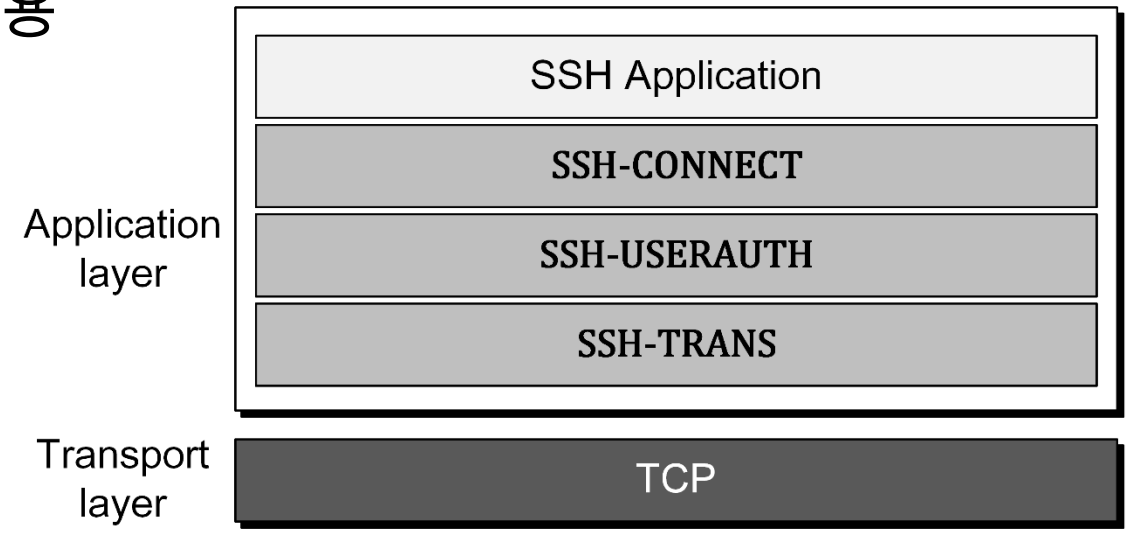
- SSH
  - 구성요소 (3/3)
    - SSH 연결 프로토콜(SSH-CONNECT)
      - 정의
        - 설정된 안전한 SSH 연결을 여러 개의 논리 채널로 나누어 다양한 서비스를 동시에 사용할 수 있도록 하는 프로토콜
      - 기능
        - 하나의 SSH 연결에서 여러 논리 채널 생성
        - 각 논리 채널에서 서로 다른 서비스 수행 가능
        - 원격 로그인, 명령 실행, 포트 전달 등에 활용



<그림 15> SSH의 구성요소

원격 로그인

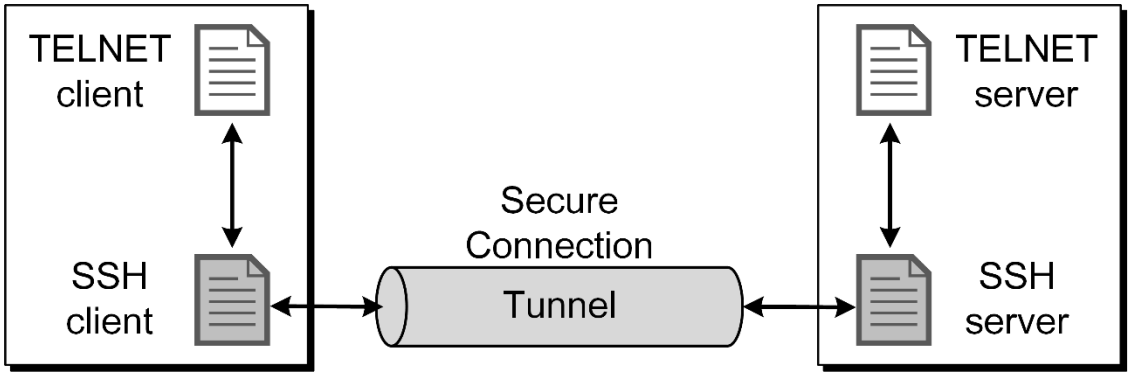
- SSH
  - SSH 응용(SSH Application)
    - 정의
      - SSH-CONNECT이 제공하는 논리 채널을 이용하여 실제 네트워크 서비스를 제공하는 응용 프로그램
    - 기능
      - 원격 로그인 및 Shell 사용
      - 원격 명령 실행
      - 파일 전송
      - TCP/IP 포트 전달



<그림 15> SSH의 구성요소

원격 로그인

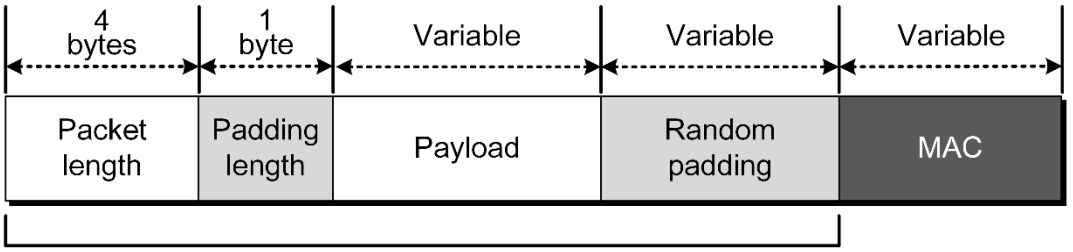
- SSH
  - SSH 터널링(SSH Tunneling)
    - 정의
      - 다른 응용 프로토콜의 트래픽을 SSH의 암호화된 채널을 통해 안전하게 전달하는 SSH의 포트 전달(Port Forwarding) 서비스
    - 특징
      - SSH Client와 SSH Server 사이의 암호화된 터널 생성
      - 보안 기능이 없는 응용 프로토콜의 통신 내용 보호
      - 다른 프로토콜의 요청·응답을 SSH 터널을 통해 전달
      - TELNET 등 평문 기반 프로토콜의 안전한 전송 지원



<그림 16> SSH 터널링(포트 전달)

원격 로그인

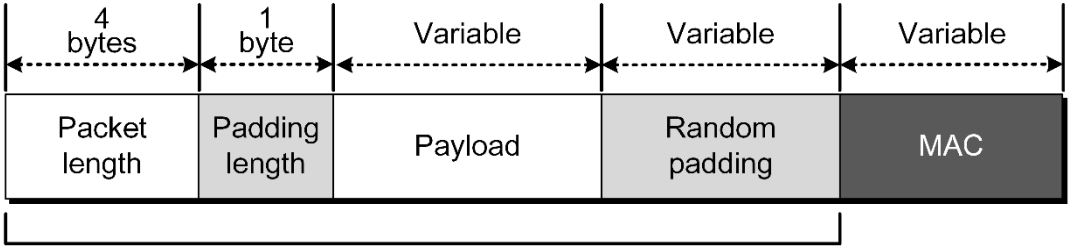
- SSH
  - 패킷 구조 (1/2)
    - 패킷 길이(Packet length)
      - 패딩 길이, 페이로드, 랜덤 패딩을 포함한 패킷 길이
        - 길이 필드 자체와 MAC 길이는 제외
      - 4바이트 고정 길이
    - 패딩 길이(Padding length)
      - 뒤에 추가되는 랜덤 패딩의 길이 정보
      - 1바이트 고정 길이
    - 페이로드(Payload)
      - 실제 전송되는 SSH 메시지 데이터
      - SSH 메시지의 종류를 나타내는 값도 페이로드 내부에 포함
      - 가변 길이



<그림 17> SSH 패킷 형식

원격 로그인

- SSH
  - 패킷 구조 (2/2)
    - 랜덤 패딩(Random padding)
      - 패킷 길이를 암호화 블록 크기에 맞추기 위한 임의의 패딩 데이터
        - 패킷 패턴 분석을 어렵게 하는 효과
      - 최소 4바이트 이상 사용
    - MAC (Message Authentication Code)
      - 패킷의 무결성 및 인증 검증
      - 협상된 MAC 알고리즘에 따라 길이 결정
      - SSH-1의 단순 CRC보다 강력한 암호학적 무결성 보호



<그림 17> SSH 패킷 형식

목 차

- 원격 로그인
- 파일 전송

파일 전송

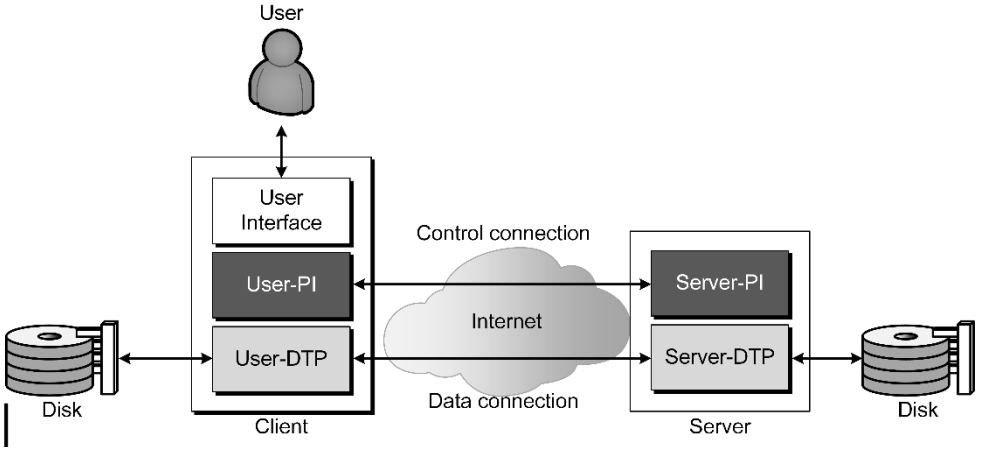
- 파일 전송 프로토콜(FTP, File Transfer Protocol)
  - 정의
    - TCP/IP 네트워크에서 서로 다른 호스트 간 파일 전송을 제공하는 응용 계층 프로토콜
  - 필요성
    - 서로 다른 시스템 간 파일 이름 지정 방식의 차이
    - 텍스트 및 데이터 표현 방식의 차이
    - 시스템별 디렉터리 구조의 차이
    - 이기종 시스템 간 일관된 파일 전송 방식 제공 필요

파일 전송

- 파일 전송 프로토콜(FTP)
  - 특징
    - 전송 계층에서 TCP 기반 통신
    - 클라이언트-서버 구조
    - 명령·응답을 위한 제어와 데이터 전송을 분리하기 위해 두 개의 TCP 연결 사용
      - 서버 기본 제어 포트: TCP 21번
      - 기본 Active 데이터 연결의 서버 포트: TCP 20번
      - [PORT],[PASV]를 통해 비기본 데이터 포트 사용 가능
    - 파일 업로드·다운로드 및 디렉터리 관련 기능 제공

파일 전송

- 파일 전송 프로토콜(FTP)
  - FTP 모델
    - 클라이언트
      - User Interface
        - 사용자의 명령 입력 및 결과 표시
      - User Protocol Interpreter (User-PI)
        - FTP 명령 및 응답 처리
      - User Data Transfer Process (User-DTP)
        - 실제 파일 및 데이터 전송
    - 서버
      - Server Protocol Interpreter (Server-PI)
        - 클라이언트의 명령 수신 및 응답
      - Server Data Transfer Process (Server-DTP)
        - 실제 파일 및 데이터 송수신



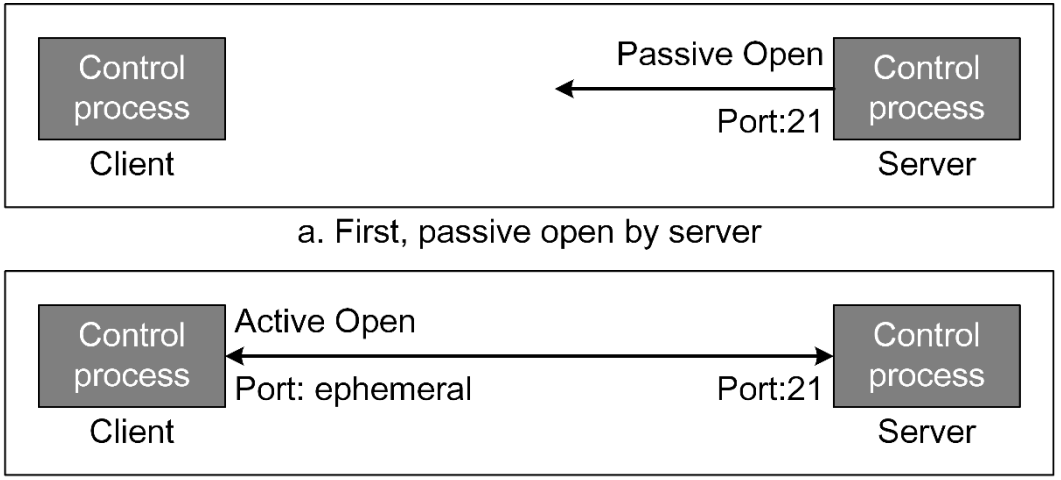
<그림 18> 파일 전송 프로토콜

파일 전송

- 파일 전송 프로토콜(FTP)
  - 연결 종류 - 제어 연결(Control Connection) (1/3)
    - 정의
      - FTP에서 클라이언트와 서버 간 명령과 응답을 주고받기 위해 사용하는 TCP 연결
    - 특징
      - FTP 세션 시작 시 설정 후 전체 세션 동안 연결 유지
      - 서버의 기본 포트 TCP 21번 사용
      - 클라이언트 측 임시(Ephemeral) 포트 사용
      - 사용자와 서버 간 상호 대화형 통신 수행
      - 명령 입력 후 빠른 응답을 위해 지연 최소화가 중요
      - 여러 파일 전송 시에도 하나의 제어 연결 지속 사용

파일 전송

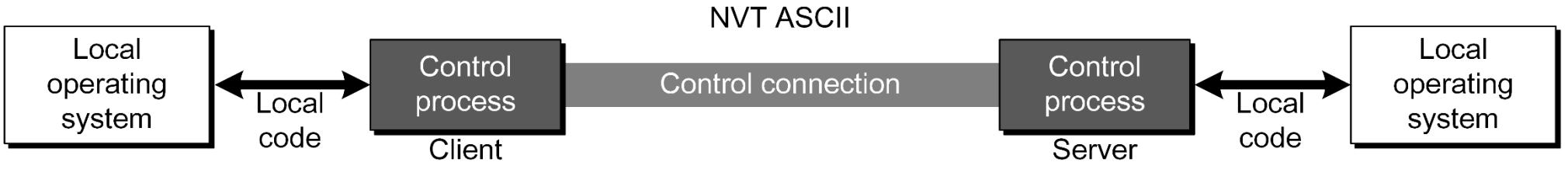
- 파일 전송 프로토콜(FTP)
  - 연결 종류 - 제어 연결 (2/3)
    - 연결 생성 과정
      1. 서버가 TCP 21번 포트에서 연결 요청 대기(Passive Open)
      2. 클라이언트가 임시 포트를 사용하여 서버의 21번 포트로 연결 요청(Active Open)
      3. TCP 연결 설정 후 FTP 명령 및 응답 교환



<그림 19> 제어 연결 생성

파일 전송

- 파일 전송 프로토콜(FTP)
  - 연결 종류 - 제어 연결 (3/3)
    - 통신
      - 제어 연결은 TELNET 프로토콜 규칙 사용
      - 명령(Command)과 응답(Reply) 기반 통신
      - 하나의 명령 또는 응답을 한 줄 단위로 전송
      - 각 줄의 끝을 [CR] [LF]로 구성된 [CRLF]로 표시
        - [CR] (Carriage Return): 커서를 현재 줄의 맨 앞으로 이동
        - [LF] (Line Feed): 커서를 다음 줄로 이동
      - 클라이언트와 서버의 로컬 문자 표현을 NVT-ASCII로 상호 변환



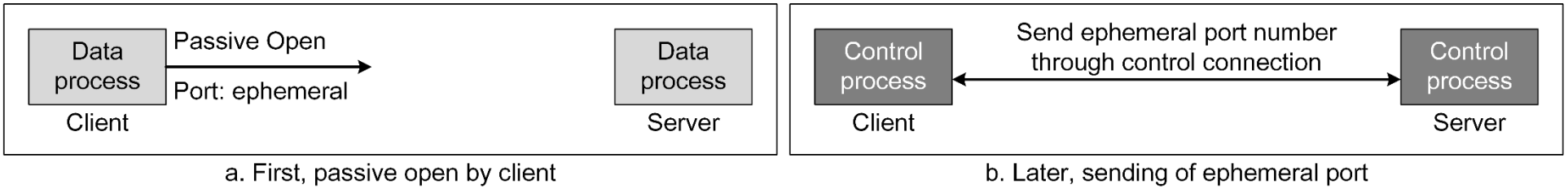
<그림 20> 제어 연결 통신

파일 전송

- 파일 전송 프로토콜(FTP)
  - 연결 종류- 데이터 연결(Data Connection) (1/8)
    - 정의
      - FTP에서 클라이언트와 서버 간 실제 파일 및 데이터를 전송하기 위해 사용하는 TCP 연결
    - 특징
      - 파일 전송이 필요할 때마다 별도로 생성
      - 제어 연결과 분리된 독립적인 TCP 연결 사용
      - Active 방식에서는 서버 측 기본 포트 TCP 20번 사용
      - 클라이언트 측 임시(Ephemeral) 포트 사용
      - 대용량 파일 전송을 위한 데이터 처리량 중심의 통신

파일 전송

- 파일 전송 프로토콜(FTP)
  - 연결 종류 - 데이터 연결 (2/8)
    - 연결 생성 과정
      1. 클라이언트가 임시 포트에서 대기(Passive Open)
      2. 그 포트 번호를 [PORT] 명령으로 서버에 전달
      3. 서버가 자신의 20번 포트에서 클라이언트의 임시 포트로 연결을 먼저 시작(Active Open)



a. First, passive open by client

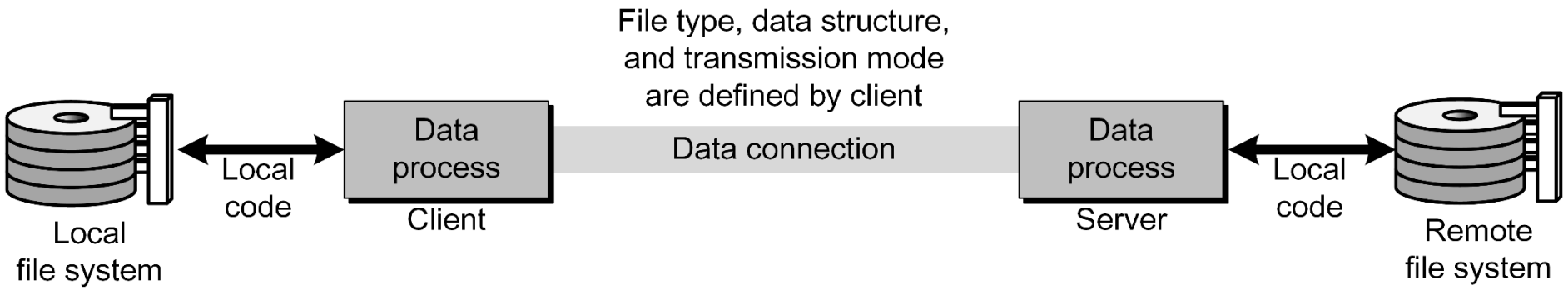
b. Later, sending of ephemeral port

c. Third, activ eopen by server

<그림 21> 데이터 연결 생성

파일 전송

- 파일 전송 프로토콜(FTP)
  - 연결 종류 - 데이터 연결 (3/8)
    - 통신
      - 실제 파일 및 데이터의 내용 전송
      - 이기종 시스템 간 파일 표현 차이 해결을 위한 전송 방식 사전 설정
      - 데이터 전송 전 제어 연결을 통해 파일 종류, 데이터 구조, 전송 모드 지정



<그림 22> 데이터 연결 통신

파일 전송

- 파일 전송 프로토콜(FTP)
  - 연결 종류 - 데이터 연결 (4/8)
    - 표현 형식(Representation Type) (1/2)
      - ASCII 파일
        - 텍스트 파일 전송을 위한 기본 형식
        - 송신 측의 로컬 문자 표현을 NVT-ASCII로 변환 후 전송
        - 수신 측에서 NVT-ASCII를 자신의 로컬 표현으로 변환
        - 파일 출력 형식 추가 지정 가능
      - EBCDIC 파일
        - EBCDIC 문자 부호화를 사용하는 시스템 간 파일 전송
        - 파일 내용을 EBCDIC 형식으로 전송
        - 파일 출력 형식 추가 지정 가능
      - Image 파일
        - 이진 파일 전송을 위한 기본 형식
        - 데이터의 해석이나 문자 변환 없이 연속된 비트열 그대로 전송
        - 실행 파일, 이미지 등 바이너리 데이터 전송에 사용

파일 전송

- 파일 전송 프로토콜(FTP)
  - 연결 종류 - 데이터 연결 (5/8)
    - 표현 형식 (2/2)
      - ASCII/EBCDIC 파일 출력 형식
        - Nonprint
          - 출력 제어 문자를 포함하지 않는 일반 텍스트 형식
          - 별도의 출력 처리 없이 저장·사용
        - TELNET
          - [CR], [LF], [NL], [VT] 등의 NVT-ASCII 제어 문자 포함
          - 전송 후 직접 출력 가능한 텍스트 형식

파일 전송

- 파일 전송 프로토콜(FTP)
  - 연결 종류 - 데이터 연결 (6/8)
    - 데이터 구조(Data Structure)
      - 파일 구조(File Structure)
        - 별도의 논리 구조 없이 연속된 바이트 흐름으로 구성
        - FTP의 기본 데이터 구조
      - 레코드 구조(Record Structure)
        - 파일을 여러 개의 레코드 단위로 구분
        - 주로 텍스트 기반 데이터에 사용
      - 페이지 구조(Page Structure)
        - 파일을 여러 페이지 단위로 구분
        - 각 페이지에 페이지 번호와 헤더 포함
        - 페이지별 임의 또는 순차 접근 지원

파일 전송

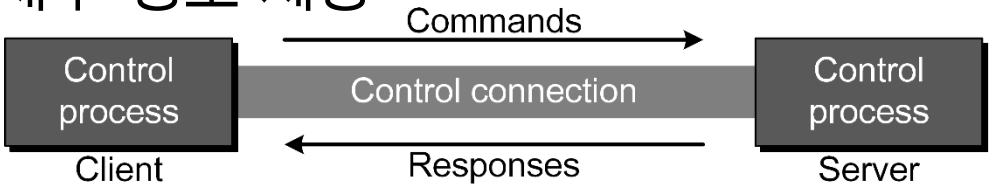
- 파일 전송 프로토콜(FTP)
  - 연결 종류 - 데이터 연결 (7/8)
    - 전송 모드(Transmission Mode) (1/2)
      - 스트림 모드(Stream Mode)
        - FTP의 기본 전송 모드
        - 데이터를 연속된 바이트 흐름으로 TCP에 전달
        - 파일 구조에서는 데이터 연결 종료를 통한 [EOF] 표시
        - 레코드 구조에서는 [EOR] 및 [EOF] 표시 사용
      - 블록 모드(Block Mode)
        - 데이터를 블록 단위로 분할하여 전송
        - 각 블록 앞에 3바이트 헤더 추가
          - 1바이트: 블록 설명자
          - 2바이트: 블록 크기

파일 전송

- 파일 전송 프로토콜(FTP)
  - 연결 종류 - 데이터 연결 (8/8)
    - 전송 모드(Transmission Mode) (2/2)
      - 압축 모드(Compressed Mode)
        - 데이터를 압축한 뒤 전송하는 방식
        - 주로 Run-Length Encoding 방식 사용
        - 반복되는 데이터 값을 값 + 반복 횟수로 표현
        - 텍스트의 연속 공백이나 이진 데이터의 연속 Null 등에 효과적

파일 전송

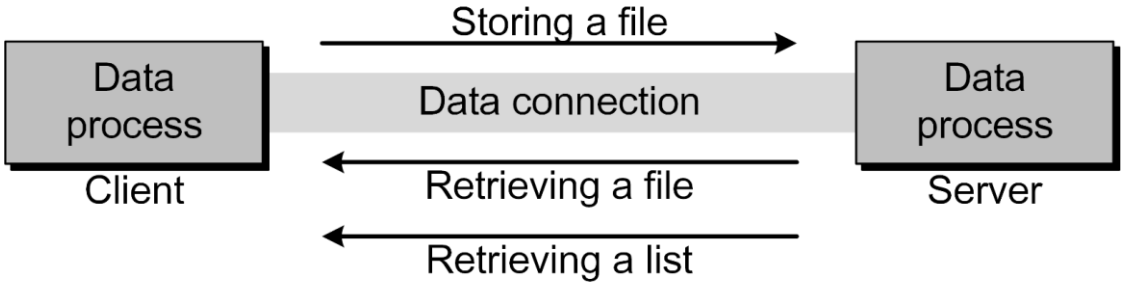
- 파일 전송 프로토콜(FTP)
  - 명령어 처리 (FTP 명령어는 부록#4, 부록#5 참고)
    - 제어 연결을 통해 명령과 응답을 교환
      - 명령(Command): 클라이언트 → 서버 방향으로 전송
      - 응답(Response): 서버 → 클라이언트 방향으로 전송
    - 모든 FTP 명령에 대해 서버가 하나 이상의 응답 반환
    - 명령어는 4자 이하의 영문자로 문자로 구성 (부록#4)
      - 대소문자 미 구분
      - 명령어에 따라 선택적으로 인수(Argument) 사용
    - 응답은 3자리 숫자 코드와 텍스트 메시지로 구성 (부록#5)
      - 숫자 코드를 통해 응답의 상태와 종류 표현
      - 텍스트를 통해 추가 설명 및 세부 정보 제공



<그림 23> FTP 명령어 처리

파일 전송

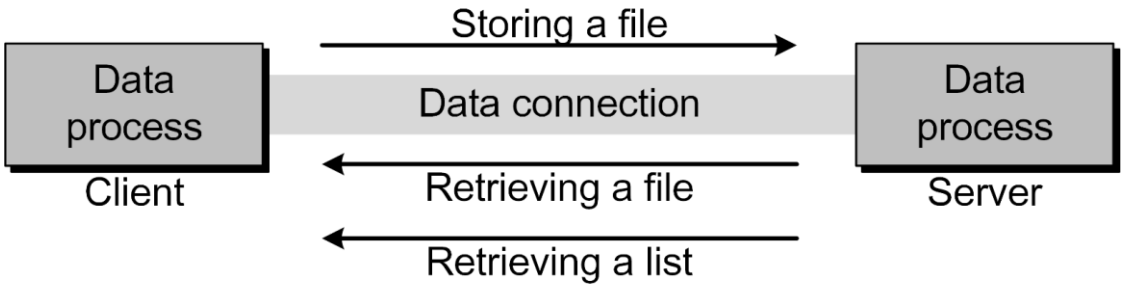
- 파일 전송 프로토콜(FTP)
  - 파일 전송 (1/12)
    - 개요
      - 제어 연결을 통해 전달되는 FTP 명령의 제어 하에 데이터 연결을 사용한 파일 전송 수행
    - 구분 (1/2)
      - 파일 읽기(Retrieving a File)
        - 서버의 파일을 클라이언트로 전송
        - [RETR] 명령에 의해 수행
        - 일반적인 다운로드 동작



<그림 24> 파일 전송

파일 전송

- 파일 전송 프로토콜(FTP)
  - 파일 전송 (2/12)
    - 구분 (2/2)
      - 파일 저장(Storing a File)
        - 클라이언트의 파일을 서버로 전송
        - [STOR] 명령에 의해 수행
        - 일반적인 업로드 동작
      - 목록 조회(Retrieving a List)
        - 서버의 디렉터리 또는 파일 이름 목록을 클라이언트로 전송
        - [LIST] 명령에 의해 수행
        - 목록 정보를 파일과 같이 취급하여 데이터 연결을 통해 전송

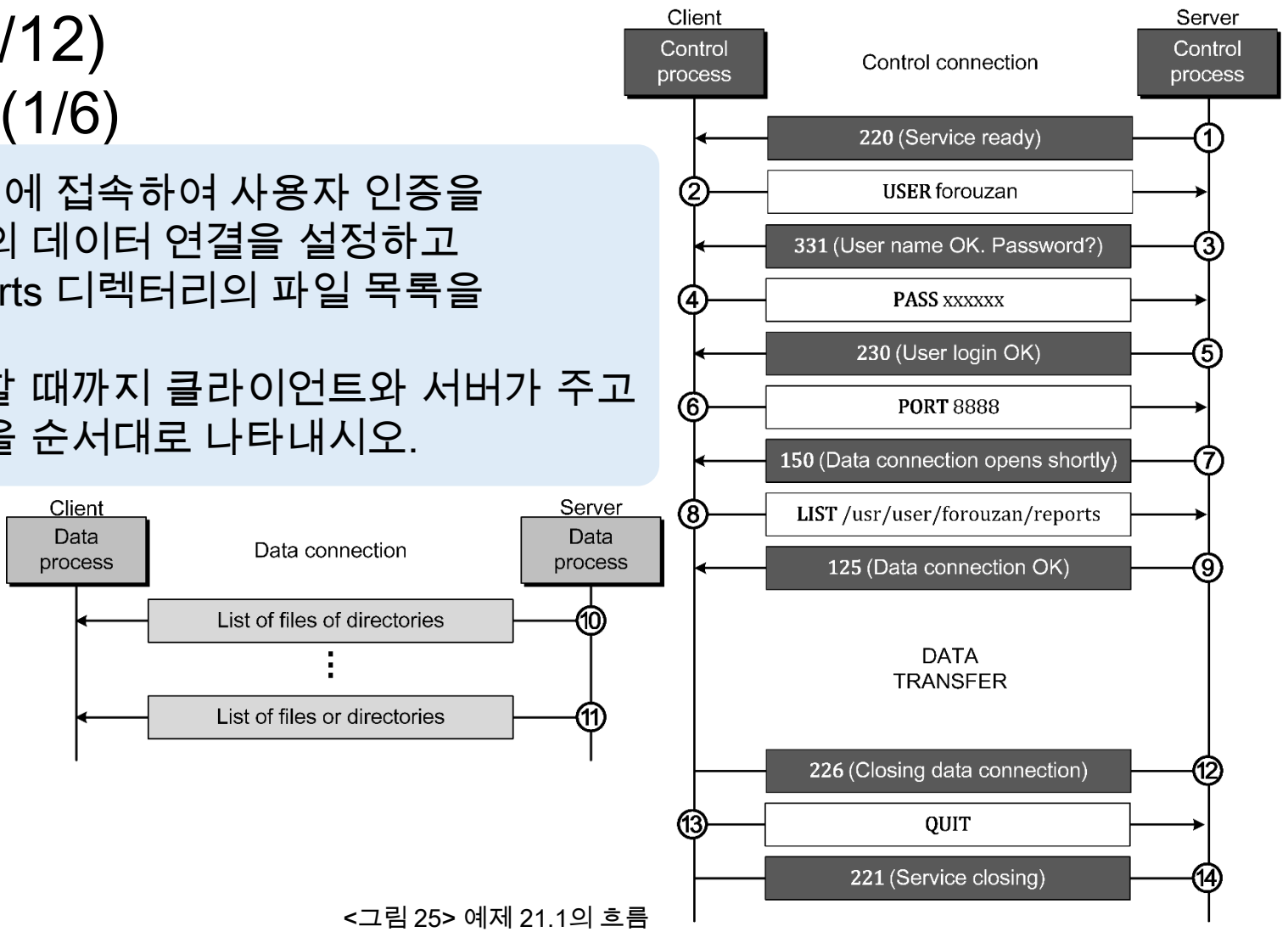


<그림 24> 파일 전송

파일 전송

- 파일 전송 프로토콜(FTP)
  - 파일 전송 (3/12)
    - 예제 21.1 (1/6)

FTP 클라이언트가 서버에 접속하여 사용자 인증을 수행한 뒤, Active 방식의 데이터 연결을 설정하고 /usr/user/forouzan/reports 디렉터리의 파일 목록을 조회하려고 한다.
이후 FTP 세션을 종료할 때까지 클라이언트와 서버가 주고받는 주요 명령과 응답을 순서대로 나타내시오.

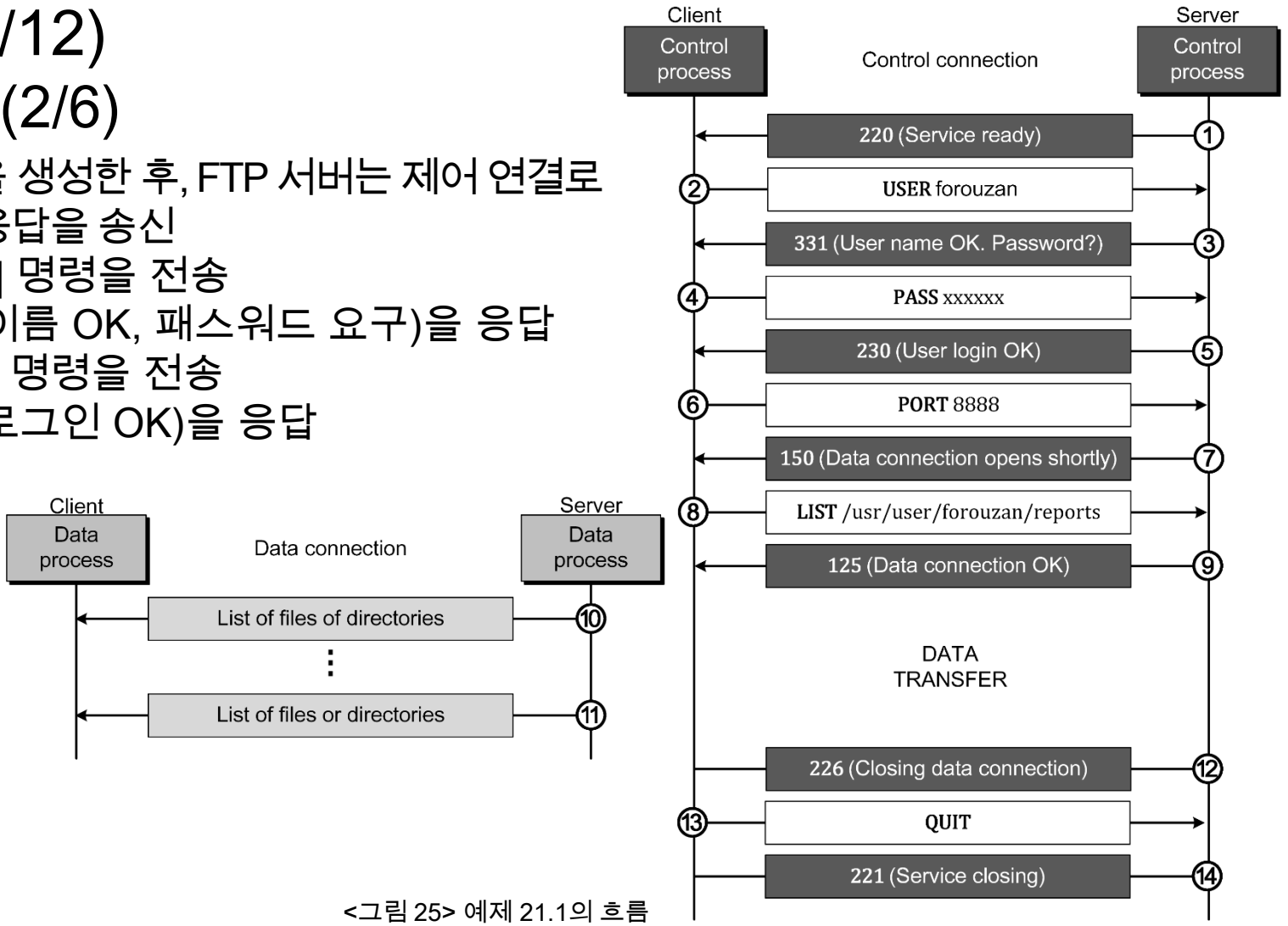


<그림 25> 예제 21.1의 흐름

파일 전송

- 파일 전송 프로토콜(FTP)
  - 파일 전송 (4/12)
    - 예제 21.1 (2/6)

1. 포트 21에 대한 연결을 생성한 후, FTP 서버는 제어 연결로 220 (서비스 준비됨) 응답을 송신
2. 클라이언트는 [USER] 명령을 전송
3. 서버는 331(사용자 이름 OK, 패스워드 요구)을 응답
4. 클라이언트는 [PASS] 명령을 전송
5. 서버는 230(사용자 로그인 OK)을 응답

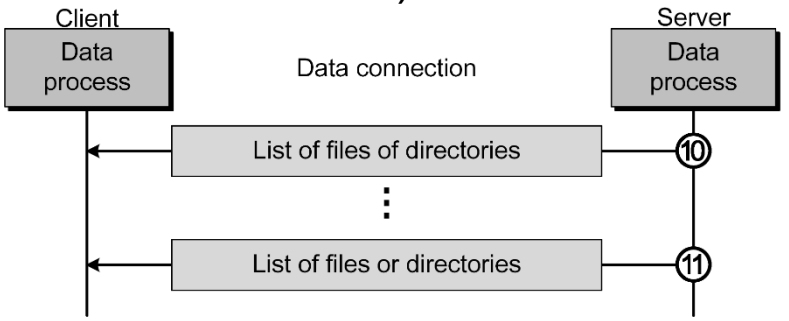


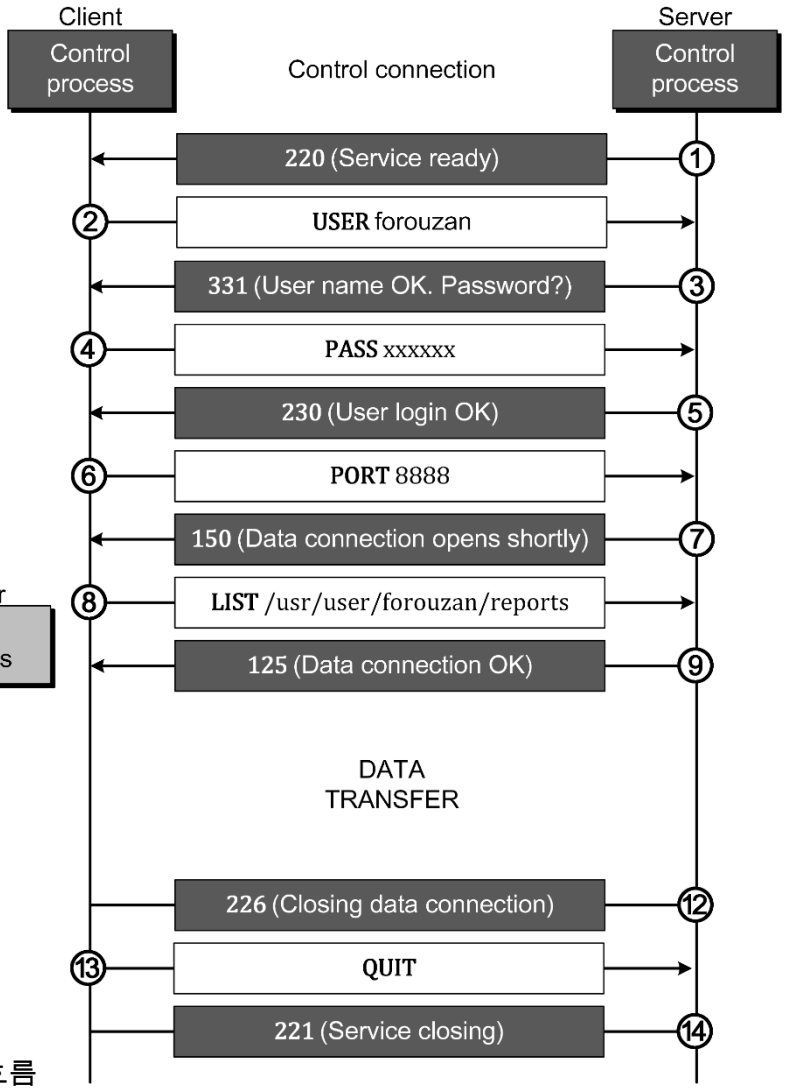
<그림 25> 예제 21.1의 흐름

파일 전송

- 파일 전송 프로토콜(FTP)
  - 파일 전송 (5/12)
    - 예제 21.1 (3/6)

6. 클라이언트는 데이터 연결에 대한 임시 포트로 수동적 설정을 하고 서버에 이 포트 번호를 보내기 위해 [PORT] 명령을 (제어 연결로) 전달
7. 서버는 클라이언트로부터 받은 임시 포트와 포트20(서버 측) 사이에 데이터 연결을 위한 능동적 설정을 시도하기 위해 준비하고 서버는 응답 150(데이터 연결이 곧 열릴 것이다)을 송신



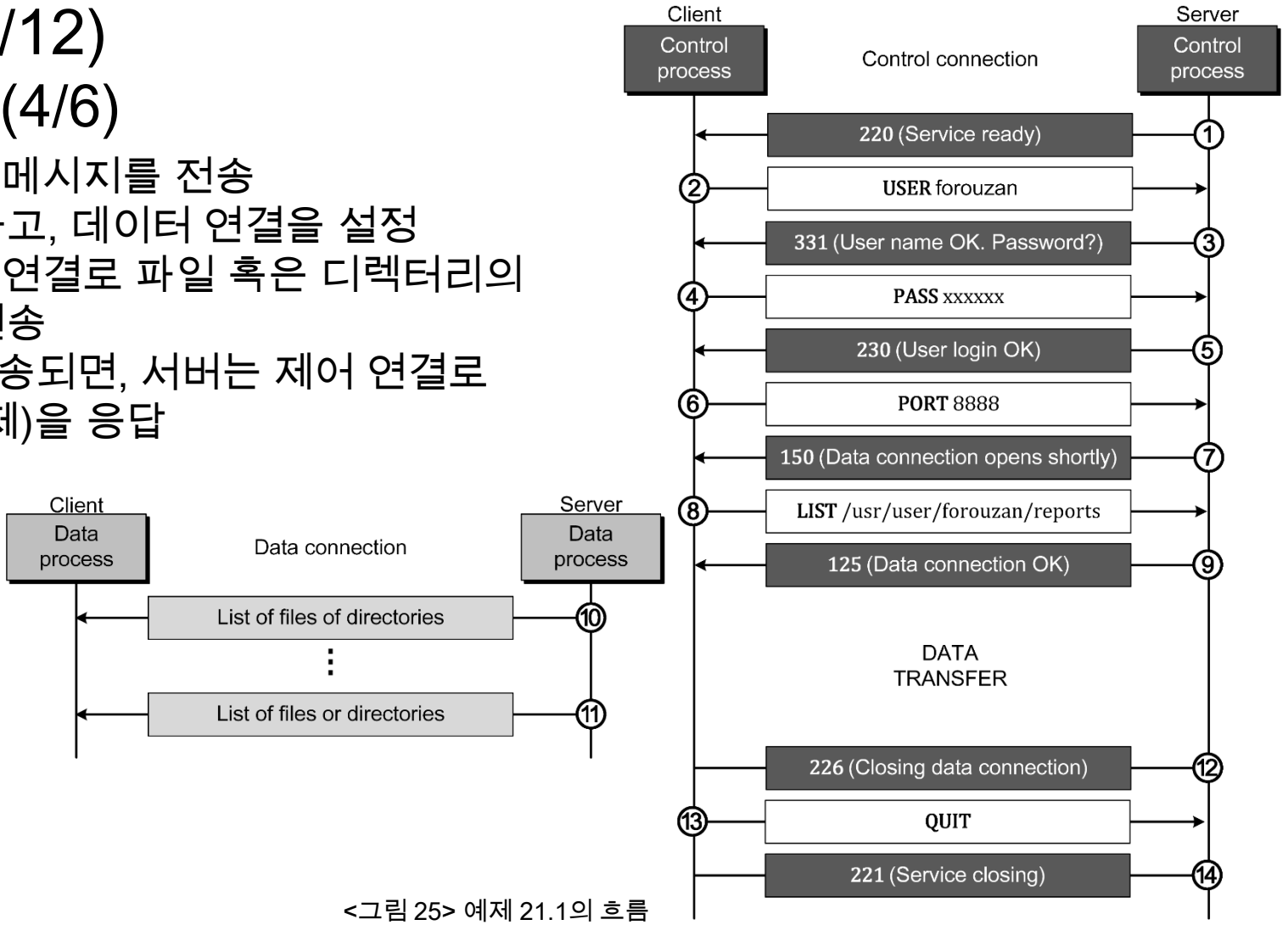


<그림 25> 예제 21.1의 흐름

파일 전송

- 파일 전송 프로토콜(FTP)
  - 파일 전송 (6/12)
    - 예제 21.1 (4/6)

8. 클라이언트는 [LIST] 메시지를 전송
9. 서버는 125로 응답하고, 데이터 연결을 설정
10. ~ 11. 서버는 데이터 연결로 파일 혹은 디렉터리의 목록을 (파일로서) 전송
12. 모든목록(파일)이 전송되면, 서버는 제어 연결로 226(데이터 연결 해제)을 응답



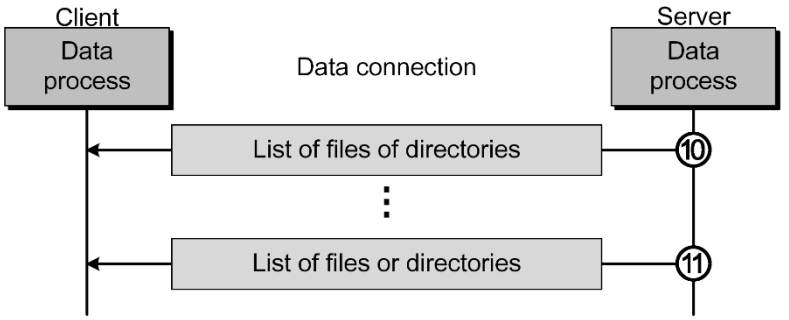
<그림 25> 예제 21.1의 흐름

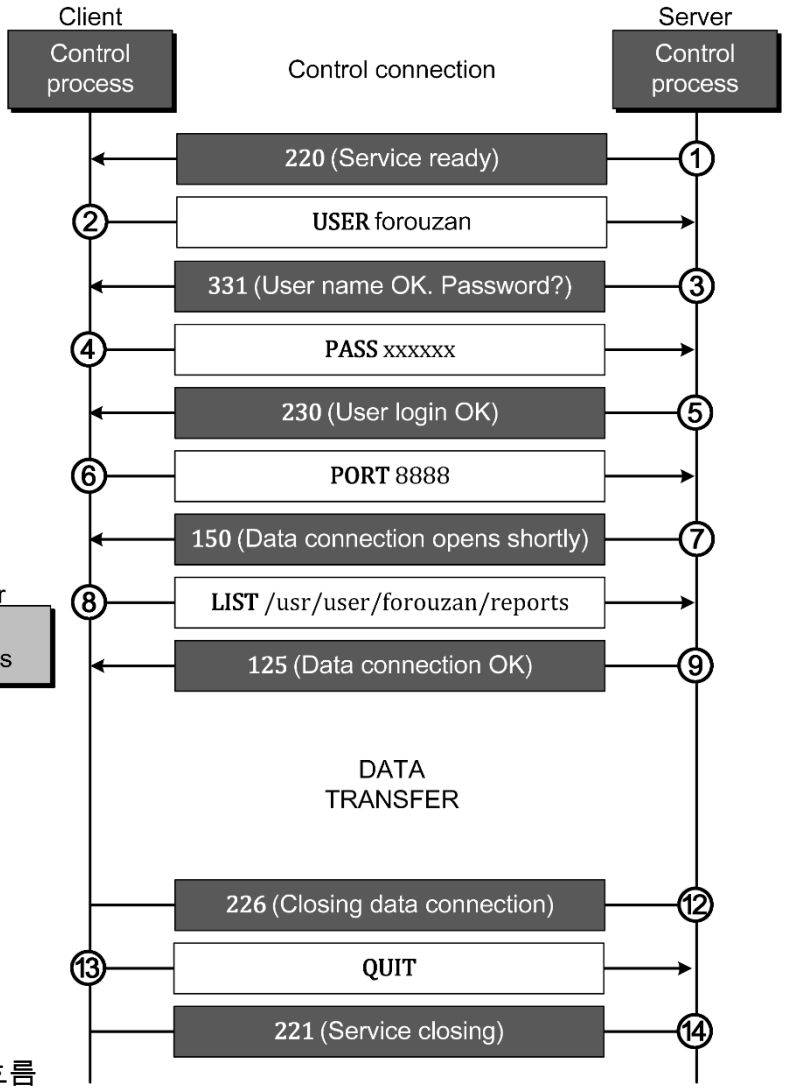
파일 전송

- 파일 전송 프로토콜(FTP)
  - 파일 전송 (7/12)
    - 예제 21.1 (5/6)

13. 클라이언트는 2개의 선택권 획득
    a. 클라이언트는 제어 연결의 해제를 요구하기 위해 [QUIT] 명령 사용 가능
    b. 또 다른 작업을 시작하기 위한 새로운 명령을 보낼 수 있음 (새로운 데이터 연결 설정)

    (해당 예제에서는 a로 클라이언트가 [QUIT] 명령전달)



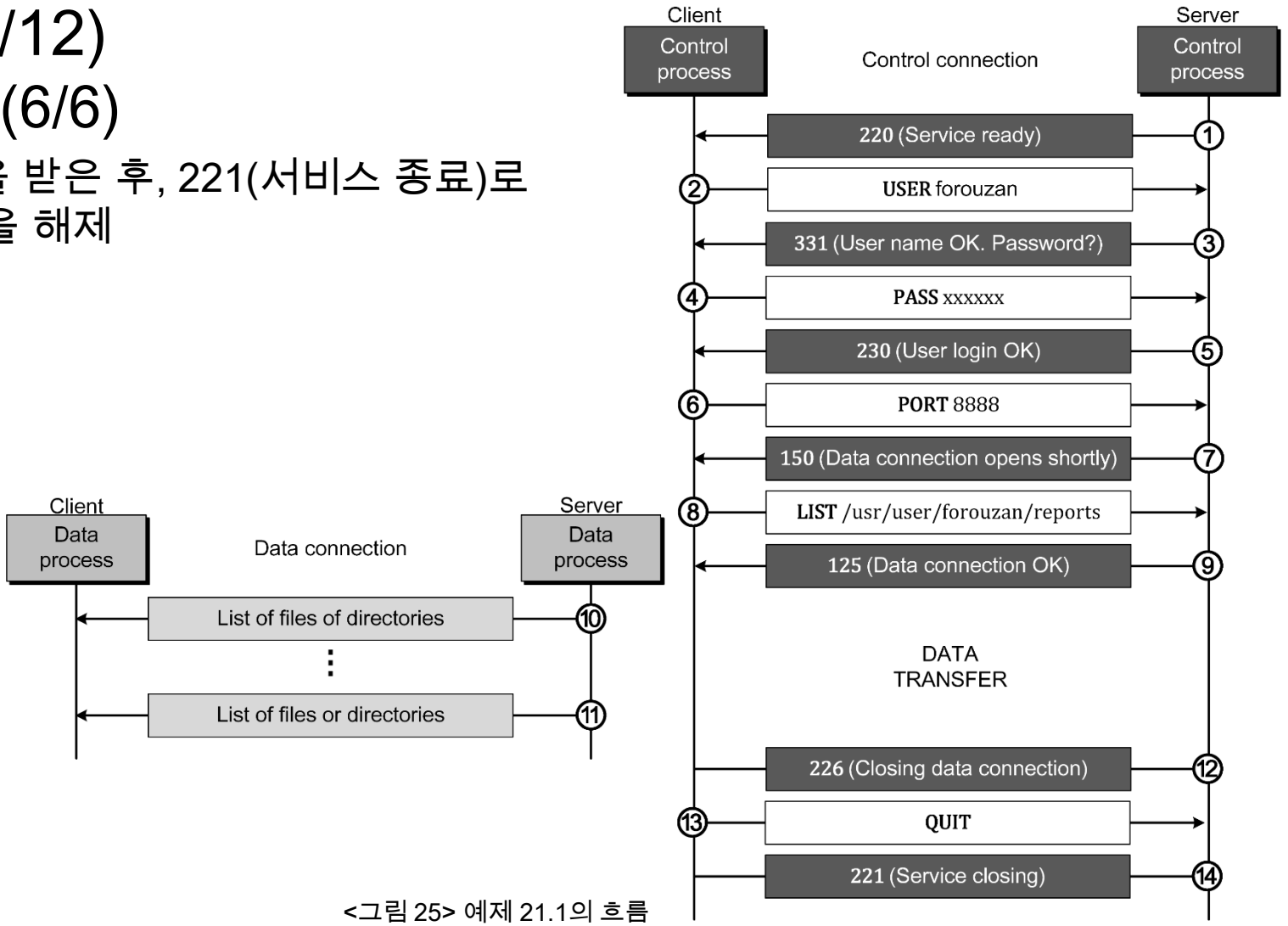


<그림 25> 예제 21.1의 흐름

파일 전송

- 파일 전송 프로토콜(FTP)
  - 파일 전송 (8/12)
    - 예제 21.1 (6/6)

14. 서버는 [QUIT] 명령을 받은 후, 221(서비스 종료)로 응답하고 제어 연결을 해제



<그림 25> 예제 21.1의 흐름

파일 전송

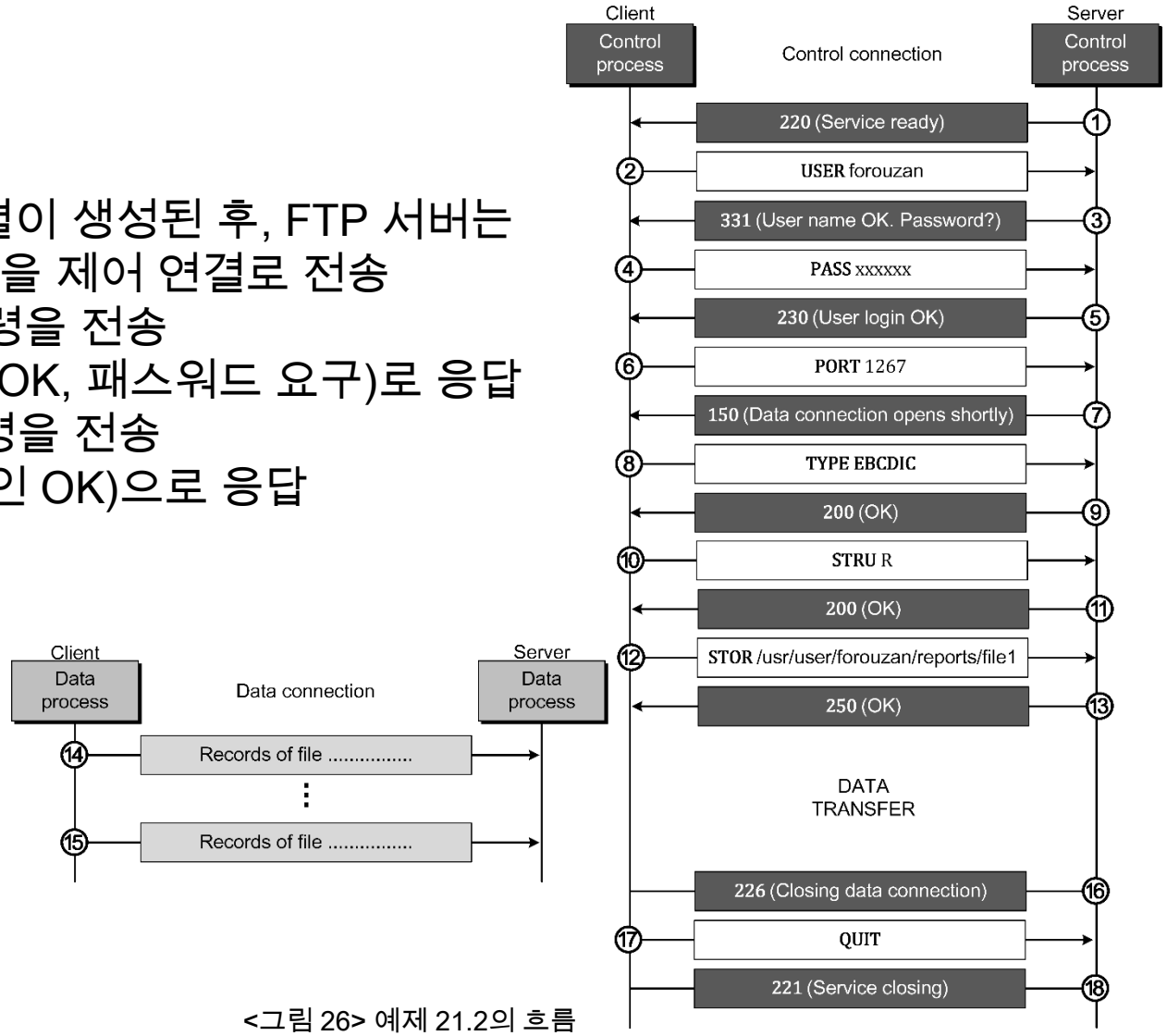
- 파일 전송 프로토콜(FTP)
  - 파일 전송 (9/12)
    - 예제 21.2 (1/4)

FTP 클라이언트가 서버에 로그인한 후 Active 방식의 데이터 연결을 설정하고, 다음 조건으로 /usr/user/forouzan/reports/file1 파일을 서버에 저장하려고 한다.
클라이언트 데이터 포트: 1267
파일 종류: EBCDIC
데이터 구조: Record
저장할 파일: /usr/user/forouzan/reports/file1
파일 저장 후 FTP 세션을 종료할 때까지 클라이언트와 서버가 주고받는 주요 명령과 응답을 순서대로 나타내시오.

파일 전송

- 파일 전송 프로토콜(FTP)
  - 파일 전송 (10/12)
    - 예제 21.2 (2/4)

1. 포트 21에 대한 제어 연결이 생성된 후, FTP 서버는 220(서비스 준비됨) 응답을 제어 연결로 전송
2. 클라이언트는 [USER] 명령을 전송
3. 서버는 331(사용자 이름 OK, 패스워드 요구)로 응답
4. 클라이언트는 [PASS] 명령을 전송
5. 서버는 230(사용자 로그인 OK)으로 응답

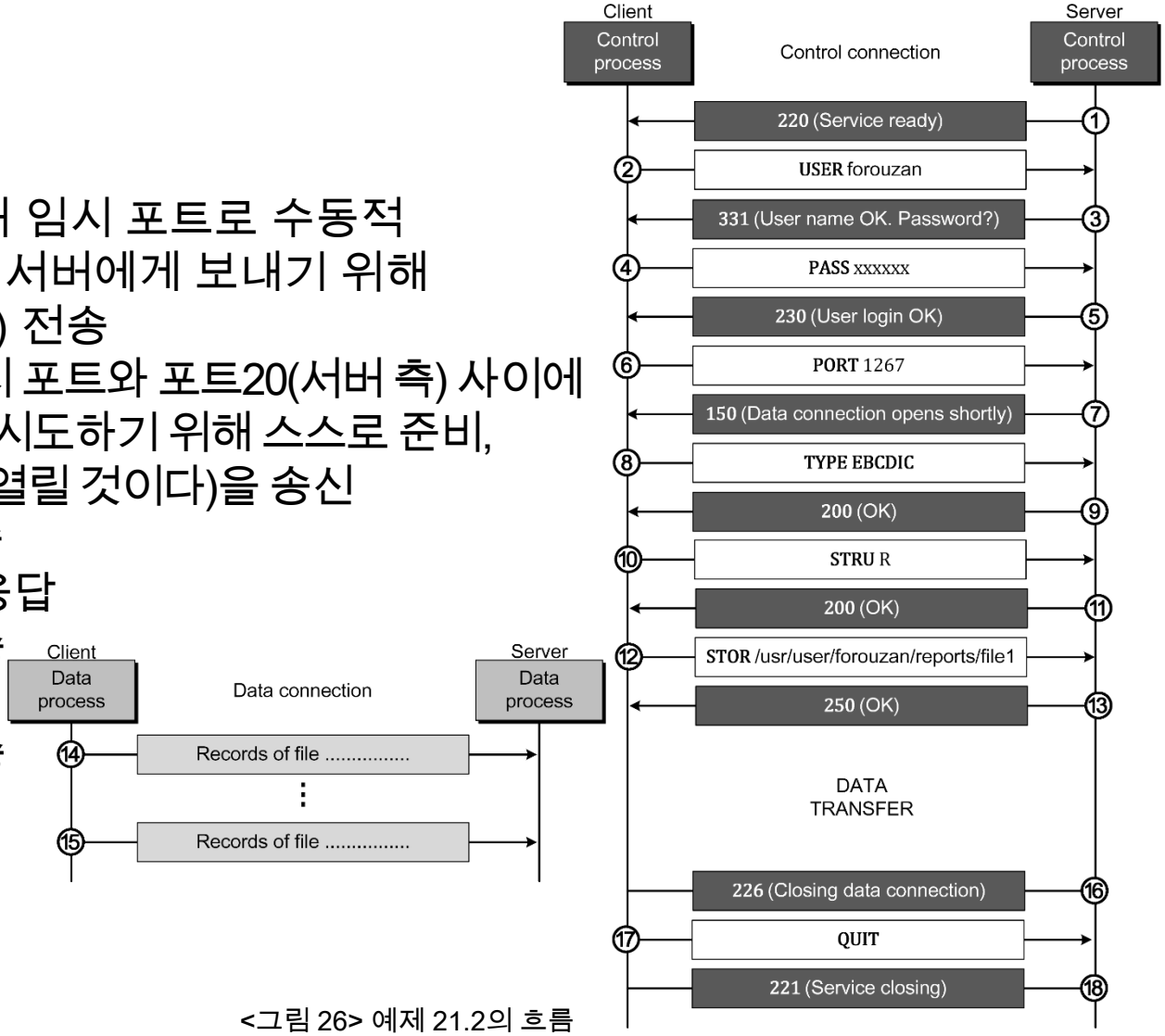


<그림 26> 예제 21.2의 흐름

파일 전송

- 파일 전송 프로토콜(FTP)
  - 파일 전송 (11/12)
    - 예제 21.2 (3/4)

6. 클라이언트는 데이터 연결에 대해 임시 포트로 수동적 설정을 시도하고, 이 포트 번호를 서버에게 보내기 위해 [PORT] 명령을 (제어 연결 상으로) 전송
7. 서버는 클라이언트로부터 받은 임시 포트와 포트20(서버 측) 사이에 데이터 연결을 위한 능동적 설정을 시도하기 위해 스스로 준비, 서버는 응답 150(데이터 연결이 곧 열릴 것이다)을 송신
8. 클라이언트는 [TYPE] 명령을 전송
9. 서버는 응답 200(명령 OK)으로 응답
10. 클라이언트는 [STRU] 명령을 전송
11. 서버는 200(명령 OK)으로 응답
12. 클라이언트는 [STOR] 명령을 전송

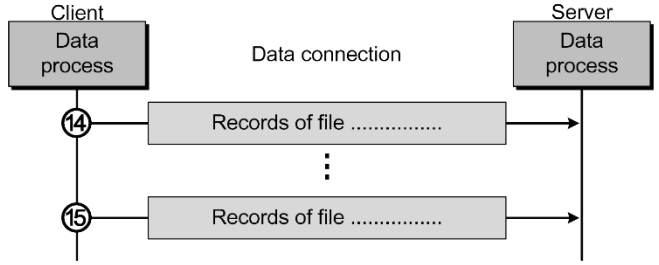


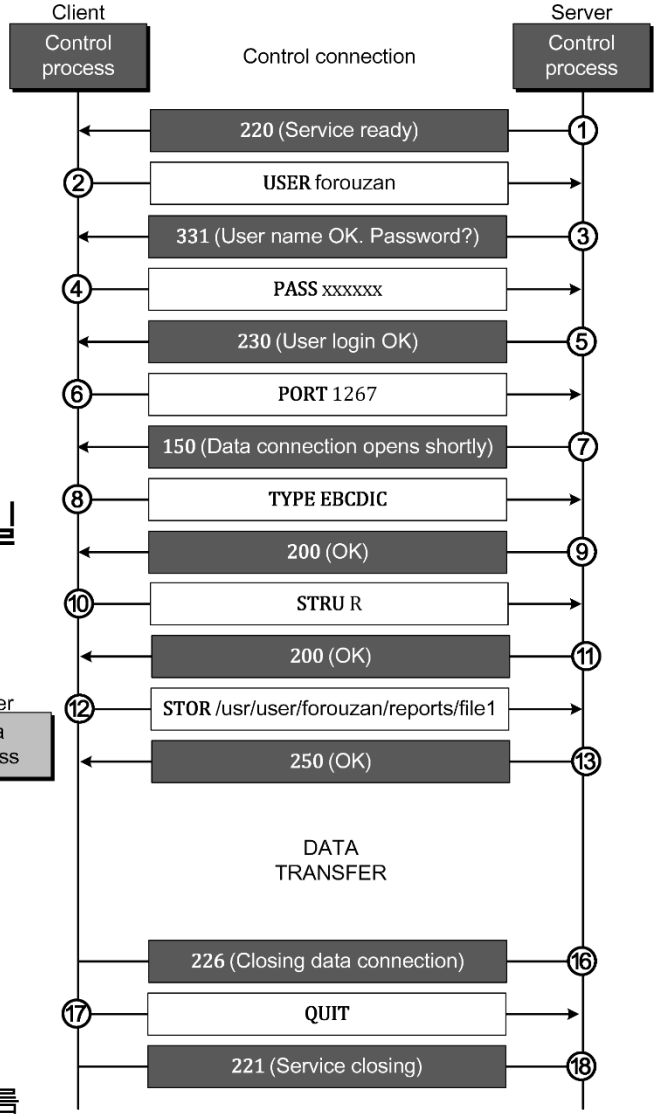
<그림 26> 예제 21.2의 흐름

파일 전송

- 파일 전송 프로토콜(FTP)
  - 파일 전송 (12/12)
    - 예제 21.2 (4/4)

13. 서버는 데이터 연결을 열고 메시지 250을 전송
14. 클라이언트는 데이터 연결로 파일을 전송
15. 전체 파일이 전송된 후 데이터 연결을 해제
    데이터 연결의 해제는 EOF를 의미
16. 서버는 응답 226을 제어 연결로 전송
17. 클라이언트는 [QUIT] 명령을 전송하거나 또는 또 다른 파일을 전송하기 위해 새로운 데이터 연결을 설정하는 새로운 명령을 사용 (이 예제에서는 [QUIT] 명령 전송)
18. 서버는 221(서비스 종료)로 응답하고 제어 연결 종료





<그림 26> 예제 21.2의 흐름

파일 전송

- 파일 전송 프로토콜(FTP)
  - 익명 FTP
    - 정의
      - 별도의 개인 계정 없이 공개된 파일에 접근할 수 있도록 제공되는 FTP 접속 방식
    - 특징
      - 사용자 이름으로 anonymous 사용
      - 패스워드로 guest 또는 이메일 주소 등의 임의 문자열 사용 가능
      - 일반 FTP 계정에 비해 제한된 접근 권한 부여
      - 주로 공개 파일의 다운로드 목적
      - 서버 설정에 따라 디렉터리 탐색, 업로드 등의 기능 제한
      - 공개 자료 배포를 위한 간편한 파일 접근 방식
    - 일반 FTP와 비교
      - 일반 FTP: 등록된 사용자 계정과 비밀번호를 통한 인증
      - 익명 FTP: 별도의 개인 계정 없이 공개용 계정으로 제한적 접근

파일 전송

- 파일 전송 프로토콜(FTP)
  - 보안 문제
    - 인증정보 보호 부재
      - 사용자 이름과 비밀번호의 평문 전송
      - 네트워크 도청에 의한 인증정보 노출 가능
    - 전송 데이터 보호 부재
      - 제어 명령 및 파일 데이터의 평문 전송
      - 전송 내용의 기밀성 보장 불가
    - 기본 보안 기능 부족
      - 자체적인 암호화 기능 부재
      - 안전한 통신을 위한 별도 보안 계층 필요

파일 전송

- SFTP (SSH File Transfer Protocol)
  - 정의
    - SSH의 안전한 통신 채널을 이용하여 파일을 전송하고 관리하기 위한 파일 전송 프로토콜
  - 특징
    - SSH 기반의 독립적인 파일 전송 프로토콜
    - SSH의 암호화·인증·무결성 보호 기능 활용
    - 하나의 SSH 연결을 통한 파일 송수신 및 파일 관리
    - SSH 서버 기본 포트인 TCP 22번 사용
    - FTP와 유사한 사용자 인터페이스 제공 가능

파일 전송

- TFTP (Trivial File Transfer Protocol)
  - 정의
    - FTP의 복잡한 기능을 제거하고, 네트워크를 통해 간단하게 파일을 읽거나 쓰기 위해 설계된 응용 계층의 파일 전송 프로토콜
  - 필요성
    - 단순한 파일 전송 기능만 필요한 환경
      - FTP의 파일 관리·인증 등 복잡한 기능이 불필요한 경우
      - 부팅 파일이나 장비 설정 파일 등 소규모 파일의 간단한 전송 필요
    - 제한된 시스템 자원에서의 사용
      - 디스크가 없거나 메모리가 제한된 장비에서도 동작 가능한 단순한 프로토콜 필요

파일 전송

- TFTP
  - 특징
    - 전송 계층에서 UDP 기반 통신
    - 클라이언트-서버 구조
    - 서버의 최초 요청 수신에 UDP 69번 포트 사용
    - FTP보다 단순한 프로토콜 구조
    - 별도의 로그인·사용자 인증 기능 부재
    - 자체적인 암호화 및 보안 기능 부재
    - 작은 크기 구현 가능
      - 부팅 및 네트워크 장비 환경에 적합

파일 전송

- TFTP
  - 기능
    - 파일 읽기(Read)
      - 서버 → 클라이언트 방향의 파일 전송
    - 파일 쓰기(Write)
      - 클라이언트 → 서버 방향의 파일 전송
  - 활용
    - 디스크 없는 워크스테이션의 부팅 파일 전송
    - 라우터·스위치 등의 설정 파일 및 펌웨어 전송
    - 단순하고 제한된 네트워크 환경에서의 파일 복사

파일 전송

- TFTP
  - 메시지 종류 (1/5)
    - RRQ (Read Request)
      - 용도
        - 클라이언트가 서버의 파일을 읽기 위한 파일 읽기 요청
        - 서버 → 클라이언트 방향의 파일 전송 시작 요청
      - 구성
        - OpCode
          - 메시지 종류를 정의하는 2바이트의 동작 코드
          - RRQ의 경우 값 1 사용
        - File Name
          - ASCII 문자열로 요청할 파일 이름을 나타내는 가변 길이 문자열 필드
          - 문자열의 끝을 0으로 채워진 1바이트 필드로 표시
        - Mode
          - 파일의 전송 형식을 지정하는 가변 길이 문자열 필드
            - 텍스트 파일 전송 시 netascii, 이진 파일 전송 시 octet 사용
          - 문자열의 끝을 0으로 채워진 1바이트 필드로 표시



<그림 27> RRQ 구조

*netascii (Network ASCII)
서로 다른 운영체제에서 텍스트 파일을 올바르게 읽을 수 있도록 문자 및 줄바꿈 표현을 공통 형식으로 변환하여 전송하는 방식

*octet
파일의 바이트 값을 변환하지 않고 그대로 전송하는 방식

파일 전송

- TFTP
  - 메시지 종류 (2/5)
    - WRQ (Write Request)
      - 용도
        - 클라이언트가 서버에 파일을 쓰기 위한 파일 쓰기 요청
        - 클라이언트 → 서버 방향의 파일 전송 시작 요청
      - 구성
        - OpCode
          - 메시지 종류를 정의하는 2바이트의 동작 코드
          - WRQ의 경우 값 2 사용
        - File Name
          - 서버에 저장할 파일 이름을 ASCII 문자열로 나타내는 가변 길이 문자열 필드
          - 문자열의 끝을 0으로 채워진 1바이트 필드로 표시
        - Mode
          - 파일의 전송 형식을 지정하는 가변 길이 문자열 필드
            - 텍스트 파일 전송 시 netascii, 이진 파일 전송 시 octet 사용
          - 문자열의 끝을 0으로 채워진 1바이트 필드로 표시



<그림 28> WRQ 구조

파일 전송

- TFTP
  - 메시지 종류 (3/5)
    - DATA

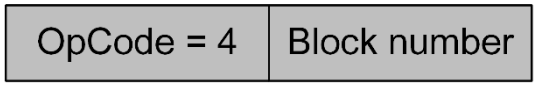


<그림 29> DATA 구조

      - 용도
        - 실제 파일 내용을 블록 단위로 전송하기 위한 메시지
      - 구성
        - OpCode
          - 메시지 종류를 정의하는 2바이트의 동작 코드
          - DATA의 경우 값 3 사용
        - Block Number
          - 전송되는 데이터 블록의 순서를 나타내는 2바이트 필드
          - 첫 번째 블록부터 1, 2, 3, … 순차적으로 증가
          - ACK 메시지와의 대응을 통한 수신 여부 및 순서 확인
        - Data
          - 실제 파일 데이터를 포함하는 0~512바이트의 가변 길이 필드
          - 마지막 블록을 제외한 일반 블록은 512바이트
          - 512바이트 미만의 블록을 마지막 블록으로 인식
          - 파일 크기가 512바이트의 배수인 경우 0바이트 DATA 블록 추가 전송

파일 전송

- TFTP
  - 메시지 종류 (4/5)
    - ACK (Acknowledgment)
      - 용도
        - DATA 블록의 정상 수신 여부를 확인하기 위한 응답 메시지
      - 구성
        - OpCode
          - 메시지 종류를 정의하는 2바이트의 동작 코드
          - ACK의 경우 값 4 사용
        - Block Number
          - 정상적으로 수신한 DATA 블록 번호를 나타내는 2바이트 필드
          - 수신한 DATA의 Block Number와 동일한 값 사용
          - WRQ 수락 시에는 파일 수신 준비를 알리기 위해 Block Number 0 사용



<그림 30> ACK 구조

파일 전송

- TFTP
  - 메시지 종류 (5/5)
    - ERROR
      - 용도
        - 연결 설정 또는 파일 전송 과정에서 발생한 오류 통보
        - RRQ·WRQ 요청의 거절 또는 데이터 전송 중 문제 발생 시 사용
      - 구성
        - OpCode
          - 메시지 종류를 정의하는 2바이트의 동작 코드
          - ERROR의 경우 값 5 사용
        - Error Number
          - 발생한 오류 종류를 나타내는 2바이트 필드
        - Error Data
          - 오류 내용을 설명하는 가변 길이 문자열 필드
          - 문자열의 끝을 0으로 채워진 1바이트 필드로 표시



<그림 31> ERROR 구조

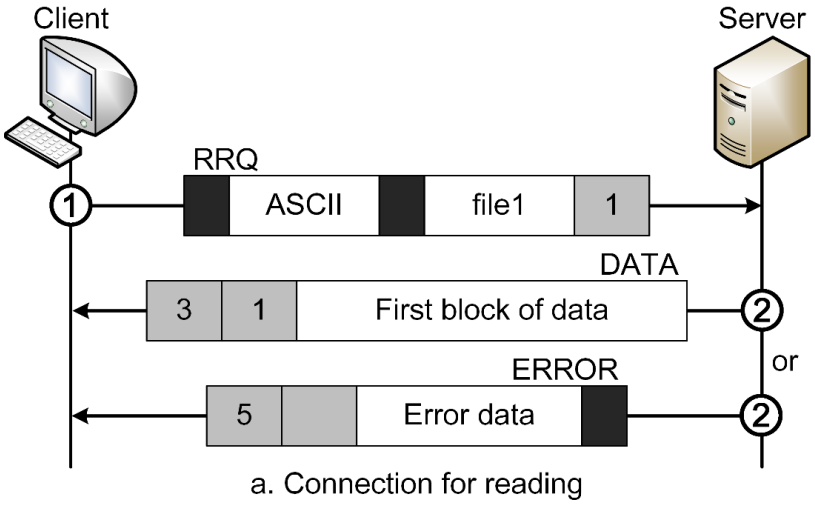
번호	의미
0	정의되지 않음
1	파일을 찾을 수 없음
2	접근 권한 위반
3	디스크 공간 부족 또는 할당량 초과
4	잘못된 동작
5	알 수 없는 포트번호
6	파일이 이미 존재함
7	해당 사용자가 없음

파일 전송

- TFTP
  - 연결 (1/3)
    - 개요
      - UDP의 연결 설정·종료 기능 부재
      - 하나의 파일을 구성하는 여러 DATA 블록을 연속적으로 전송하기 위한 TFTP 자체의 전송 절차 사용
      - RRQ, WRQ, ACK, ERROR 메시지를 통한 전송 시작 및 상태 설정
      - 512바이트 미만의 마지막 DATA 메시지를 통한 전송 종료

파일 전송

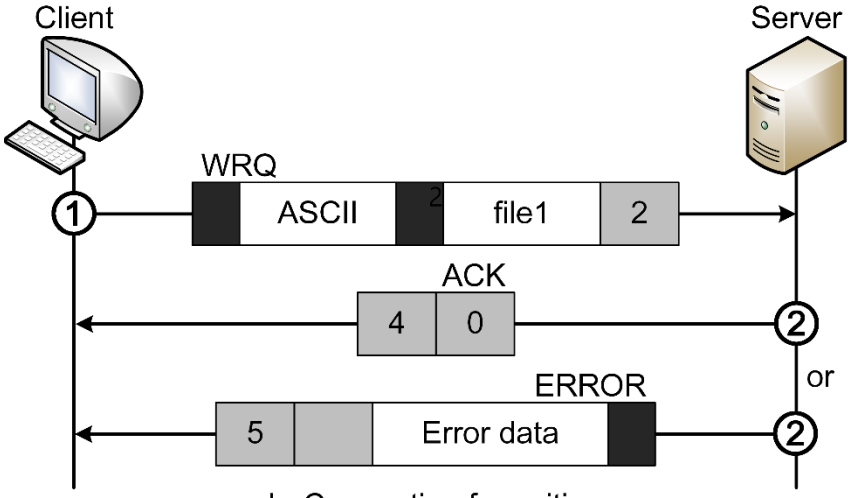
- TFTP
  - 연결 (2/3)
    - 연결 설정 (1/2)
      - 파일 읽기
        - 클라이언트의 RRQ 메시지 전송
          - 파일 이름과 전송 모드 지정
        - 서버가 요청을 수락한 경우 첫 번째 DATA 블록 전송
          - Block Number = 1
        - 요청 처리 불가 시 ERROR 메시지 전송



<그림 32> 파일 읽기를 위한 연결설정

파일 전송

- TFTP
  - 연결 (3/3)
    - 연결 설정 (2/2)
      - 파일 쓰기
        - 클라이언트의 WRQ 메시지 전송
          - 파일 이름과 전송 모드 지정
        - 서버가 요청을 수락한 경우 ACK Block 0 전송
          - 클라이언트의 데이터 수신 준비 완료 표시
        - 요청 처리 불가 시 ERROR 메시지 전송



b. Connection for writing
<그림 33> 파일 쓰기를 위한 연결설정

  - 연결 종료
    - 별도의 연결 종료 메시지 부재
    - 마지막 DATA 블록을 512바이트 미만의 크기로 전송하여 파일의 끝 표시
    - 마지막 DATA 블록의 수신 및 ACK 이후 파일 전송 종료

파일 전송

- TFTP
  - 데이터 전송 (1/5)
    - 개요
      - 파일을 데이터 블록 단위로 분할하여 전송
      - 마지막 블록을 제외한 각 DATA 블록의 크기 512바이트
        - 마지막 DATA 블록의 크기 0~511바이트
      - netascii 또는 octet 형식의 데이터 전송
      - UDP 사용으로 TFTP 자체 흐름 제어 및 오류 제어 필요
    - 흐름 제어 (1/2)
      - 하나의 DATA 블록 전송 후 해당 ACK 수신까지 대기
      - 각 DATA 블록에 Block Number 부여
      - ACK 수신 후 다음 번호의 DATA 블록 전송
      - 일정 시간 내 ACK 미수신 시 처리를 위한 타임아웃 사용
      - 한 번에 하나의 블록만 전송하는 Stop-and-Wait 방식

파일 전송

- TFTP
  - 데이터 전송 (2/5)
    - 흐름 제어 (2/2)
      - 파일 읽어오기
        - 클라이언트의 RRQ 메시지 전송
        - 서버의 DATA Block 1 전송
        - 클라이언트의 ACK Block 1 전송
        - 이후 DATA → ACK 과정의 반복
      - 파일 저장하기
        - 클라이언트의 WRQ 메시지 전송
        - 서버의 ACK Block 0 전송
          - 파일 수신 준비 완료 표시
        - 클라이언트의 DATA Block 1 전송
        - 서버의 ACK Block 1 전송
        - 이후 DATA → ACK 과정의 반복

파일 전송

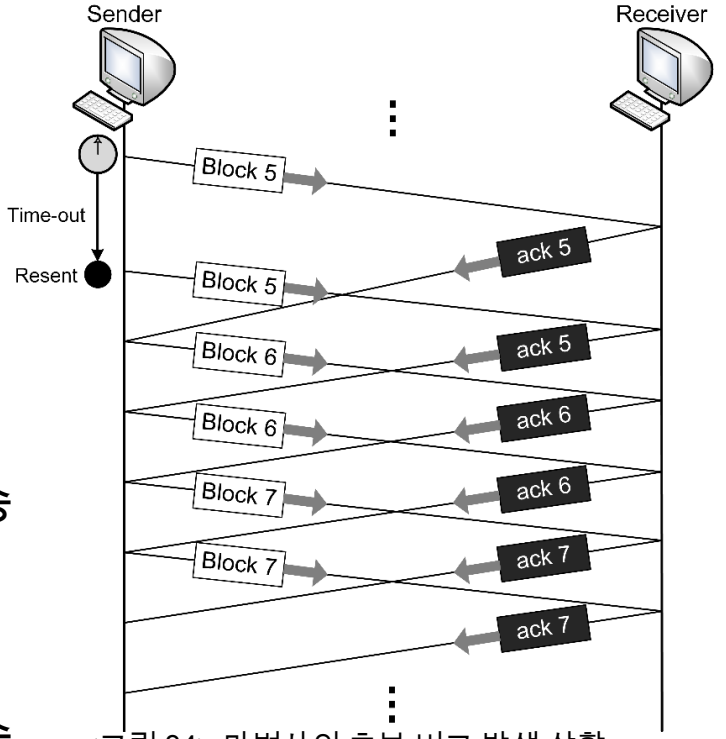
- TFTP
  - 데이터 전송 (3/5)
    - 오류 제어 (1/3)
      - 개요
        - 신뢰성 없는 UDP 사용에 따른 TFTP 자체 오류 제어 수행
        - DATA와 ACK에 대한 타임아웃 및 재전송 방식 사용
        - 메시지 손상·유실·ACK 유실·중복 수신 상황 처리
      - 메시지 처리 방법 (1/2)
        - 손상된 메시지
          - UDP 체크섬을 이용한 데이터 손상 검출
          - 손상된 DATA 블록의 수신 측 폐기
          - ACK 미수신에 따른 송신 측 타임아웃 후 DATA 재전송

파일 전송

- TFTP
  - 데이터 전송 (4/5)
    - 오류 제어 (2/3)
      - 메시지 처리 방법 (2/2)
        - 유실된 메시지
          - DATA 블록이 수신 측에 도달하지 않아 ACK 미발생
          - 송신 측의 타임아웃 후 해당 DATA 블록 재전송
        - 유실된 ACK
          - DATA는 정상 수신되었으나 ACK가 송신 측에 도달하지 않은 상태
          - 타임아웃 발생 시 동일 DATA 블록의 재전송 가능
          - 수신 측의 Block Number 확인을 통한 중복 DATA 식별
        - 중복된 메시지
          - 동일 Block Number의 DATA가 다시 도착한 경우 중복 블록으로 판단
          - 중복 DATA의 폐기 및 해당 블록에 대한 ACK 재전송

파일 전송

- TFTP
  - 데이터 전송 (5/5)
    - 오류 제어 (3/3)
      - 마법사의 초보 버그(Sorcerer's Apprentice Bug)
        - 정의
          - ACK의 지연으로 인해 송신 측의 재전송과 수신 측의 ACK 재전송이 중첩되면서, 이후 DATA와 ACK가 반복적으로 중복 전송되는 현상
        - 발생 과정
          1. DATA Block 5 전송
          2. 수신 측의 ACK 5가 지연
          3. 송신 측 타임아웃으로 Block 5 재전송
          4. 수신 측이 중복 Block 5에 대해 ACK 5 재전송
          5. 송신 측이 두 개의 ACK 5를 수신하면서 이후 Block 6을 중복 전송
          6. Block 6에 대한 ACK도 중복되어 이후 블록에서 중복 현상 반복



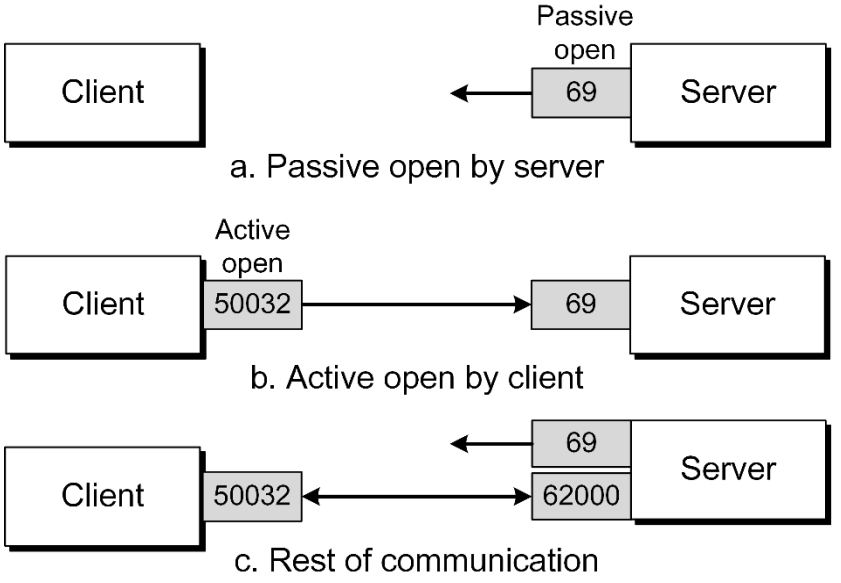
<그림 34> 마법사의 초보 버그 발생 상황

파일 전송

- TFTP
  - UDP 포트 (1/2)
    - 개요
      - TFTP 서버의 UDP 69번 포트를 초기 요청 수신용으로 사용
      - 초기 요청 이후 별도의 임시 포트를 사용하여 파일 전송 수행
    - 필요성
      - 하나의 클라이언트와 파일 전송하는 동안 69번 포트를 계속 사용할 경우 다른 클라이언트의 요청 처리 제한
      - 별도의 임시 포트를 통한 다중 클라이언트의 동시 서비스 지원

파일 전송

- TFTP
  - UDP 포트 (2/2)
    - 동작 과정
      1. 서버
         - UDP 69번 포트에서 클라이언트의 초기 요청 대기
      2. 클라이언트
         - 임시(Ephemeral) 포트를 사용하여 서버의 69번 포트로 RRQ 또는 WRQ 전송
      3. 서버
         - 요청 수신 후 새로운 임시 포트 선택
         - 해당 임시 포트에서 클라이언트의 임시 포트로 응답 전송
      4. 이후 통신
         - 클라이언트 임시 포트 ↔ 서버 임시 포트 간 DATA, ACK, ERROR 메시지 교환
         - 서버의 69번 포트는 다시 다른 클라이언트의 초기 요청 수신에 사용



<그림 35> TFTP가 사용하는 UDP포트 번호

파일 전송

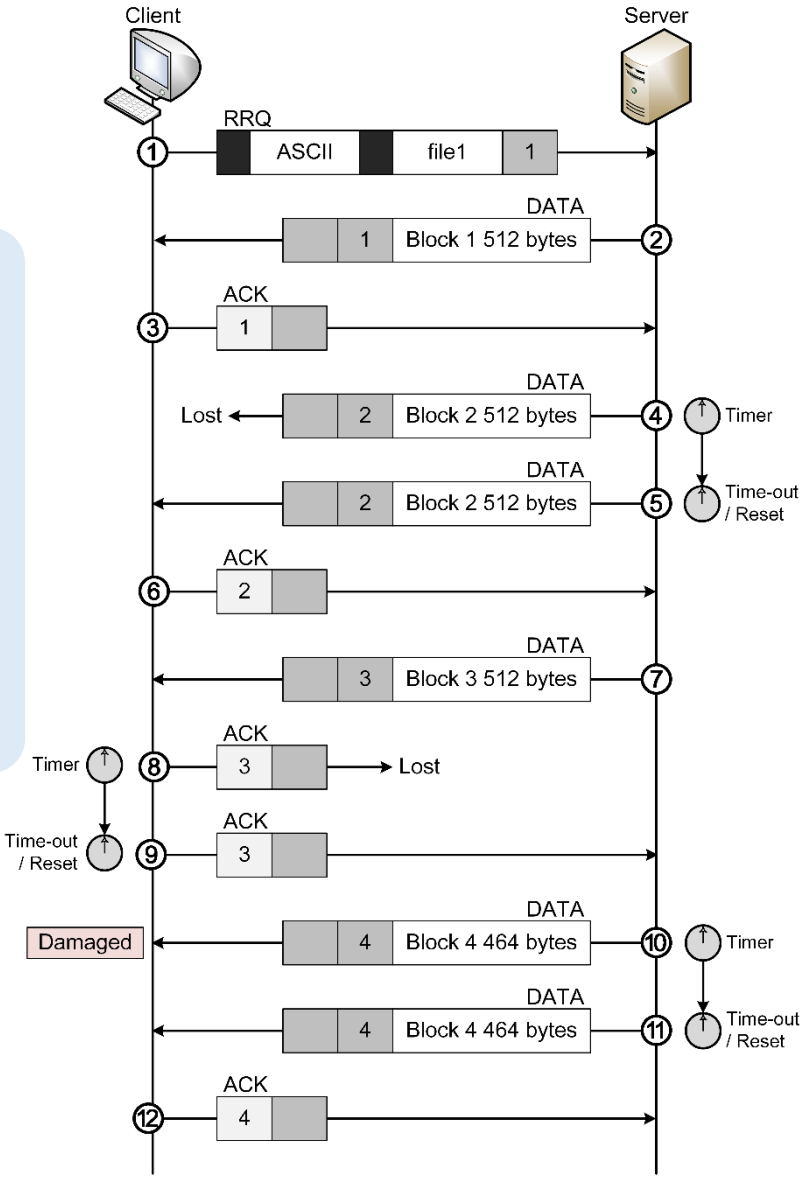
- TFTP
  - 예제 21.3 (1/3)

TFTP 클라이언트가 서버에 저장된 2,000바이트의 file1을 netascii 방식으로 읽으려고 한다. DATA 메시지는 최대 512바이트의 데이터를 전송하며, 다음과 같은 오류가 발생했다.

- 두 번째 DATA 블록 유실
- 세 번째 블록에 대한 ACK 유실
- 마지막 DATA 블록 손상

이때 클라이언트와 서버 사이의 메시지 전송 과정을 나타내시오.

1. 클라이언트가 서버에 RRQ 전송
2. 서버가 첫 번째 DATA 전송
3. 클라이언트가 ACK 전송
4. 서버가 다음 DATA를 전송하지만 유실
5. 타임아웃 발생 후 서버가 DATA 재전송
6. 클라이언트가 ACK 전송
7. 서버가 다음 DATA 전송



<그림 35> 예제 21.3의 흐름

파일 전송

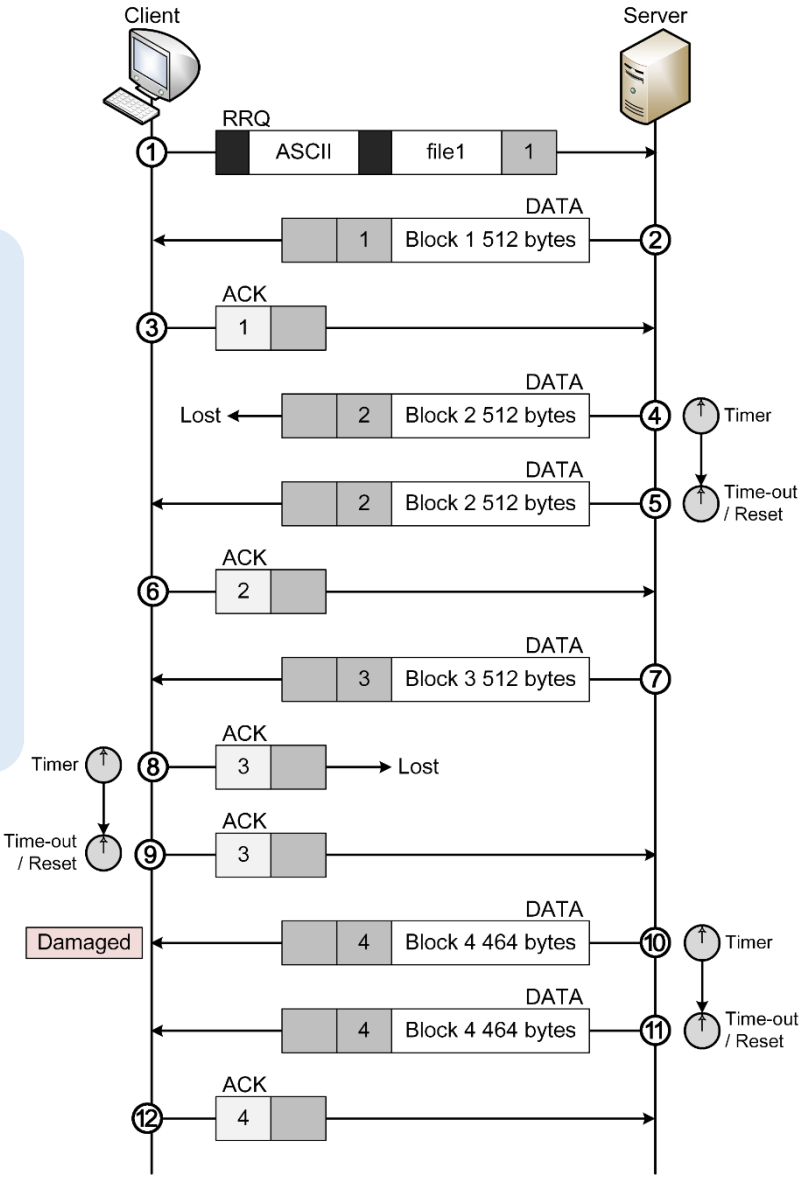
- TFTP
  - 예제 21.3 (2/3)

TFTP 클라이언트가 서버에 저장된 2,000바이트의 file1을 netascii 방식으로 읽으려고 한다. DATA 메시지는 최대 512바이트의 데이터를 전송하며, 다음과 같은 오류가 발생했다.

- 두 번째 DATA 블록 유실
- 세 번째 블록에 대한 ACK 유실
- 마지막 DATA 블록 손상

이때 클라이언트와 서버 사이의 메시지 전송 과정을 나타내시오.

8. 클라이언트가 ACK를 전송하지만 유실
9. 타임아웃 발생 후 클라이언트가 ACK 재전송
10. 서버가 마지막 DATA를 전송하지만 손상
11. 타임아웃 발생 후 서버가 DATA 재전송
12. 클라이언트가 ACK 전송 후 파일 전송 종료



<그림 35> 예제 21.3의 흐름

파일 전송

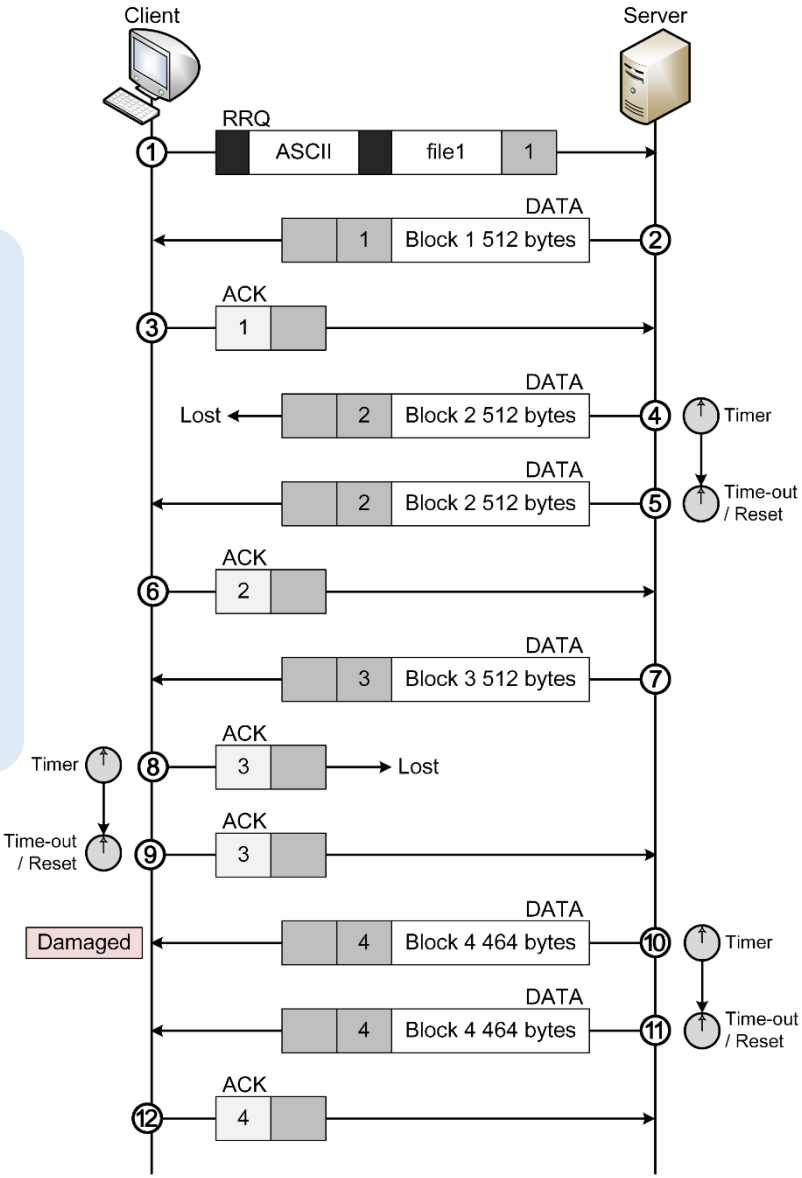
- TFTP
  - 예제 21.3 (3/3)

TFTP 클라이언트가 서버에 저장된 2,000바이트의 file1을 netascii 방식으로 읽으려고 한다. DATA 메시지는 최대 512바이트의 데이터를 전송하며, 다음과 같은 오류가 발생했다.

- 두 번째 DATA 블록 유실
- 세 번째 블록에 대한 ACK 유실
- 마지막 DATA 블록 손상

이때 클라이언트와 서버 사이의 메시지 전송 과정을 나타내시오.

- 종료 파일 크기 확인
  - Block 1: 512 bytes
  - Block 2: 512 bytes
  - Block 3: 512 bytes
  - Block 4: 464 bytes (마지막블록<512)
  - $512 \times 3 + 464 = 2{,}000$ bytes



<그림 35> 예제 21.3의 흐름

파일 전송

- TFTP
  - 옵션
    - 정의
      - 기본 TFTP 기능을 확장하기 위해 RRQ·WRQ 메시지에 추가하여 사용하는 선택적 기능
    - 역할
      - 데이터 블록 크기 등의 전송 매개변수 협상
        - 기본 512바이트 이외의 블록 크기 사용 지원
      - 클라이언트와 서버 간 필요한 전송 조건의 동적 설정
    - 옵션 협상
      - 클라이언트가 RRQ 또는 WRQ에 원하는 옵션 포함
      - 서버의 OACK(Option Acknowledgment) 메시지를 통한 옵션 수락 및 설정값 응답
      - 지원하지 않는 옵션에 대한 거절 또는 미적용

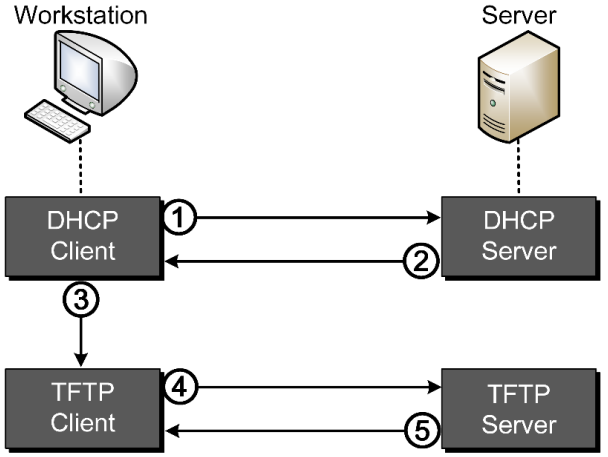
파일 전송

- TFTP
  - 보안 문제
    - 보안 기능 부재
      - 사용자 인증 및 비밀번호 기능 부재
      - 자체적인 데이터 암호화 기능 부재
      - 접근 권한 보호 기능의 제한
    - 보안 대책
      - 중요하지 않은 파일로 TFTP 제공 범위 제한
      - 서버 접근이 가능한 호스트 제한
      - 방화벽·라우터 등의 네트워크 장비를 통한 접근 제어

파일 전송

- TFTP
  - 활용
    - 라우터·스위치 등의 초기 설정 및 부팅
    - 장비의 구성 파일 다운로드
    - 적은 메모리를 가진 장비에서의 간단한 파일 전송
    - DHCP와 연계한 네트워크 부팅 환경 구성
      1. 장치 부팅 후 DHCP Client 실행
      2. DHCP Server로부터 네트워크 설정 및 구성 파일 이름 수신
      3. 전달받은 파일 이름을 TFTP Client에 전달
      4. TFTP Client가 TFTP Server에 파일 요청
      5. TFTP Server로부터 구성 파일 다운로드

*동적 호스트 구성 프로토콜(DHCP, Dynamic Host Configuration Protocol) 네트워크에 접속한 장치에 IP 주소, 서브넷 마스크, 게이트웨이, DNS 등의 네트워크 설정 정보를 자동으로 할당하는 프로토콜



<그림 36> DHCP와 TFTP의 공동 사용

Thanks!

송 민 희(minhee@pel.sejong.ac.kr)

부록#1

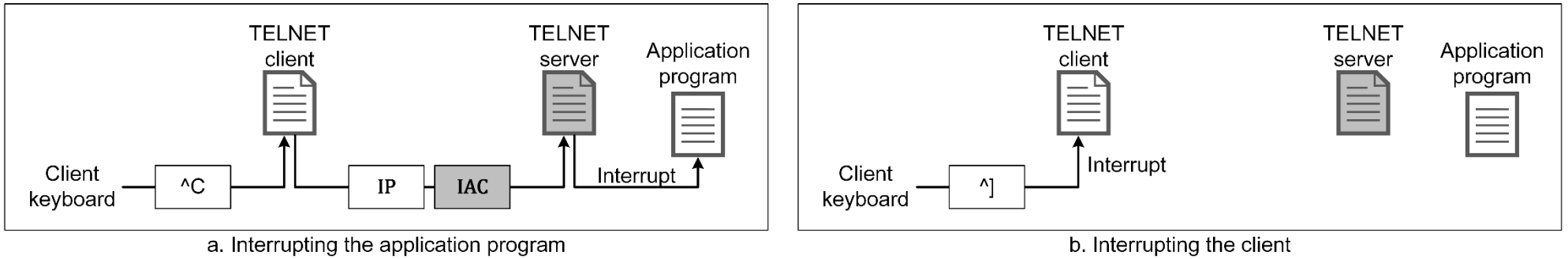
- TELNET
  - 사용자 인터페이스(User Interface)
    - 정의
      - 사용자가 TELNET 기능을 쉽게 사용할 수 있도록 제공되는 명령 체계
    - 특징
      - 사용자 명령을 실제 TELNET 프로토콜 동작으로 변환
        - 사용자가 직접 TELNET 제어 문자를 입력할 필요 없이 편의 명령 사용 가능
      - 사용자 명령을 내부적으로 TELNET 프로토콜 명령으로 변환
      - 연결 설정, 상태 확인, 동작 모드 변경, 종료 등의 기능 제공

부록#1

- TELNET
  - 사용자 인터페이스
    - 인터페이스 명령
      - open: 원격 컴퓨터에 연결
      - close: 현재 연결 종료
      - display: 현재 동작 매개변수 표시
      - mode: 문자 모드 또는 라인 모드로 변경
      - set: 동작 매개변수 설정
      - status: 현재 연결 및 상태 정보 표시
      - send: 특수 TELNET 제어 문자 전송
      - quit: TELNET 프로그램 종료

부록#1

- TELNET
  - 사용자 인터페이스
    - Escape 문자
      - 일반 입력은 원격 서버로 전달되지만, Escape 문자는 로컬 TELNET Client에서 처리
      - TELNET 명령 프롬프트로 전환하거나 클라이언트 동작을 제어하는 데 사용
      - 일반적으로 Ctrl+] 사용
        - 원격 서버에서는 일반적으로 Ctrl+C 사용



<그림 37> 2개의 서로 다른 인터럽트

부록 #2

- TELNET
  - 운영 모드(Operating Mode)
    - 정의
      - TELNET에서 사용자의 입력을 어떤 단위와 방식으로 처리하여 원격 서버에 전달할지를 결정하는 동작 방식
    - 종류
      - 기본 모드(Basic Mode)
      - 문자 모드(Character Mode)
      - 라인 모드(Line Mode)

부록 #2

- TELNET
  - 기본 모드 (1/3)
    - 정의
      - TELNET에서 옵션 협상을 통해 다른 운영 모드가 선택되지 않았을 때 사용되는 기본적인 입력·전송 방식
    - 특징
      - 클라이언트 측 로컬 에코 수행
        - 사용자가 입력한 문자를 클라이언트가 자신의 화면에 직접 표시
      - 라인 단위 데이터 전송
        - 입력한 문자를 바로 서버로 보내지 않고 한 줄이 완성된 후 전송
      - [GA]를 이용한 전송 순서 제어
        - 한 줄 전송 후 서버의 [GA]를 수신해야 다음 입력 진행
      - 반이중(Half-Duplex) 방식의 통신
        - 클라이언트와 서버가 번갈아 데이터를 전송
      - TCP 자체가 전이중 통신을 지원하므로 현대 환경에서는 비효율적

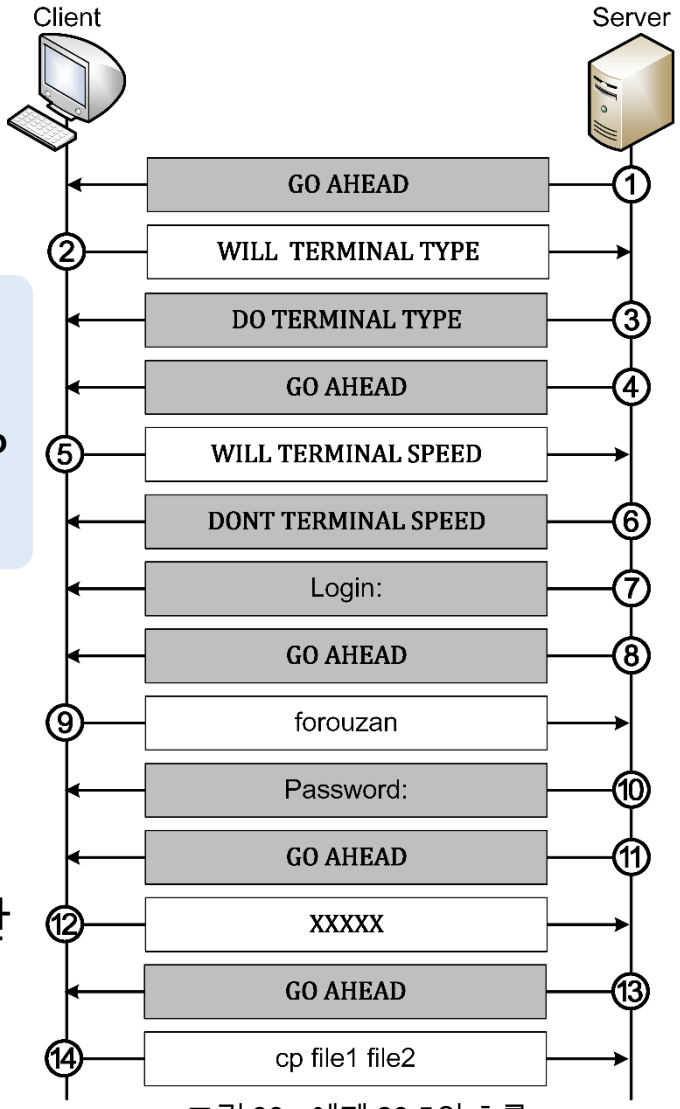
부록 #2

- TELNET
  - 기본 모드 (2/3)
    - 동작 과정
      1. 사용자가 클라이언트에서 한 줄 입력
      2. 클라이언트가 입력 문자를 로컬 화면에 Echo
      3. 한 줄 입력이 완료되면 서버로 전송
      4. 서버가 데이터를 처리한 뒤 [GA] 전송
      5. 클라이언트가 [GA]를 수신한 후 다음 입력 진행

부록 #2

- TELNET
  - 기본 모드 (3/3)
    - 예제 20.5

다음 그림은 TELNET 기본 모드에서 클라이언트와 서버가 통신하는 과정을 나타낸다.그림에서 GO AHEAD가 여러 차례 반복해서 전송되는 이유를 설명하고, 이러한 동작 방식이 TCP 통신에서 가지는 단점을 서술하시오.



<그림 38> 예제 20.5의 흐름

- 기본 모드에서는 한쪽이 데이터를 전송한 뒤 상대방에게 전송 권한을 넘기기 위해 [GA]를 사용
  - 따라서 클라이언트와 서버가 데이터를 번갈아 전송하는 반이중 방식으로 동작
- 그러나 TCP는 원래 전이중 통신을 지원하므로 [GA]를 이용한 순차적 전송 제어가 불필요
- 반복적인 [GA] 송수신과 대기로 인해 통신 효율 저하

부록 #2

- TELNET
  - 문자 모드 (1/3)
    - 정의
      - TELNET에서 사용자가 입력한 문자를 한 문자 단위로 즉시 서버에 전송하고, 서버가 이를 다시 Echo하여 표시하는 입력·전송 방식
    - 특징
      - 사용자 입력을 문자 단위로 즉시 서버에 전송
      - 서버에 의한 원격 Echo 수행
        - 서버가 수신한 문자를 다시 클라이언트로 전송하여 화면에 표시
      - Suppress Go-Ahead 옵션 활성화
        - [GA]를 기다리지 않고 양방향으로 데이터 전송
      - 문자 모드 사용을 위해 일반적으로 Suppress Go-Ahead와 Echo 옵션 협상
      - 문자 하나의 입력·Echo 과정에서 여러 TCP 세그먼트 발생 가능

부록 #2

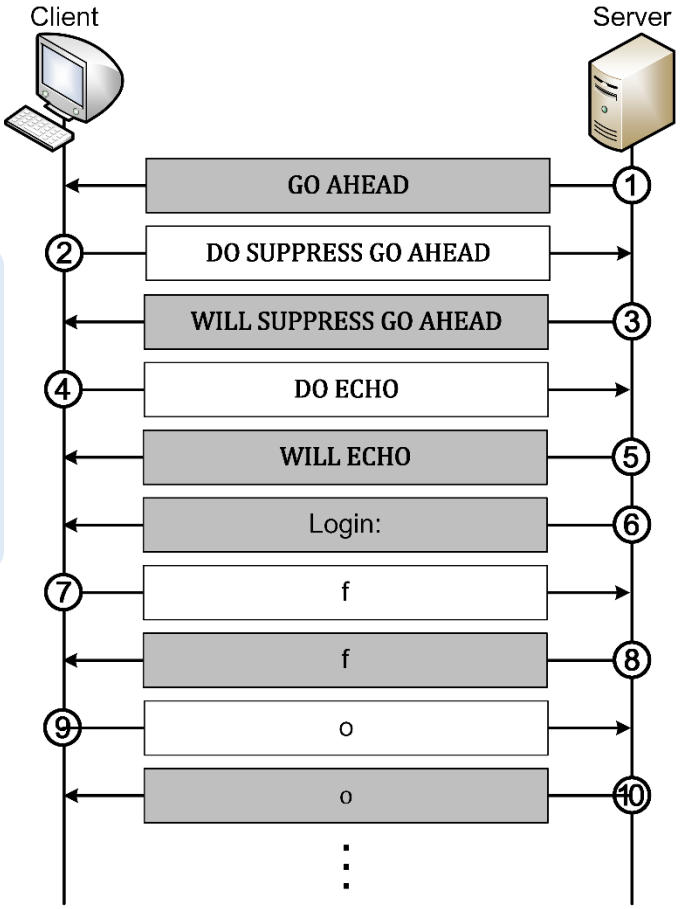
- TELNET
  - 문자 모드 (2/3)
    - 동작 과정
      1. 사용자가 문자 입력
      2. 클라이언트가 입력 문자를 서버로 전송
      3. 서버가 수신한 문자를 클라이언트로 Echo
      4. 클라이언트 화면에 문자 표시

부록 #2

- TELNET
  - 문자 모드 (3/3)
    - 예제 20.6

TELNET이 문자 모드로 동작할 때 사용자가 forouzan을 입력한다고 가정한다. 문자 하나를 입력할 때마다 서버로 문자 전송, 서버의 Echo 전송, Echo에 대한 확인 응답이 발생한다면 전체 입력 과정에서 발생하는 TCP 세그먼트 수를 구하고, 문자 모드의 문제점을 설명하시오.

- 입력 문자 수: 8개
- 문자당 TCP 세그먼트: 3개
- 총 TCP 세그먼트 수: 8 × 3 = 24개
- 한 문자 입력마다 여러 TCP 세그먼트가 발생하여 네트워크 트래픽과 처리 부담 증가



<그림 39> 예제 20.6의 흐름

부록 #2

- TELNET
  - 라인 모드 (1/2)
    - 정의
      - TELNET에서 사용자의 입력을 클라이언트가 로컬에서 편집한 뒤, 완성된
        한 줄 단위로 서버에 전송하는 입력·전송 방식
    - 특징
      - 문자 삭제, 줄 삭제 등 라인 편집을 클라이언트에서 로컬 처리
      - 입력된 전체 라인을 한 번에 서버로 전송
      - 서버의 [GA] 없이 데이터 전송 가능
      - 완전 이중(Full-Duplex) 방식으로 동작

부록 #2

- TELNET
  - 라인 모드 (2/2)
    - 동작 과정
      1. 사용자가 클라이언트에서 한 줄 입력
      2. Backspace, 문자 삭제, 줄 삭제 등의 편집을 로컬에서 처리
      3. 입력이 완료되면 전체 라인을 서버로 전송
      4. 서버는 수신한 라인을 처리
      5. [GA]를 기다리지 않고 다음 입력 진행

부록 #3

- TELNET 실습
  - 실습 환경
    - Client: Windows Powershell
    - Server: Ubuntu
  - 환경 구성
    - wsl 및 우분투 버전 확인
      - WSL 2 및 Ubuntu 22.04.5 LTS 사용 중

```
PS C:\Users\minhe> wsl -l -v
  NAME            STATE           VERSION
* Ubuntu-22.04    Running         2
```

```
(base) pel-miin@miinpel:~$ cat /etc/os-release
PRETTY_NAME="Ubuntu 22.04.5 LTS"
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04.5 LTS (Jammy Jellyfish)"
VERSION_CODENAME=jammy
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=jammy
```

부록 #3

- TELNET 실습
  - 환경 구성
    - telnet 설치 및 서비스 설정 확인
      - 상태 확인: TELNET 서비스 비활성화

```
(base) pel-miin@miinpel:~/TELNET$ dpkg -l openbsd-inetd inetutils-telnetd
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name             Version          Architecture Description
+++-================-================-============-==============================
ii  inetutils-telnetd 2:2.2-2ubuntu0.2 amd64        telnet server
ii  openbsd-inetd     0.20160825-5     amd64        OpenBSD Internet Superserver
(base) pel-miin@miinpel:~/TELNET$ grep -n telnet /etc/inetd.conf
23:#<off># telnet       stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/telnetd
```

      - TELNET 서비스 활성화

```
(base) pel-miin@miinpel:~/TELNET$ sudo update-inetd --enable telnet
(base) pel-miin@miinpel:~/TELNET$ grep -n telnet /etc/inetd.conf
23:telnet       stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/telnetd
```

      - 서비스 재시작 및 TCP 23번 포트 확인

```
(base) pel-miin@miinpel:~/TELNET$ sudo service openbsd-inetd restart
(base) pel-miin@miinpel:~/TELNET$ sudo ss -ltnp | grep ':23 '
LISTEN 0      128          0.0.0.0:23        0.0.0.0:*    users:(("inetd",pid=1344,fd=7))
```

부록 #3

- TELNET 실습
  - 환경 구성
    - TELNET 서버 수신 주소 설정 및 서비스 재시작

```
(base) pel-miin@miinpel:~/TELNET$ sudo nano /etc/inetd.conf
```



```
#:STANDARD: These are standard services.
telnet  stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/telnetd
```

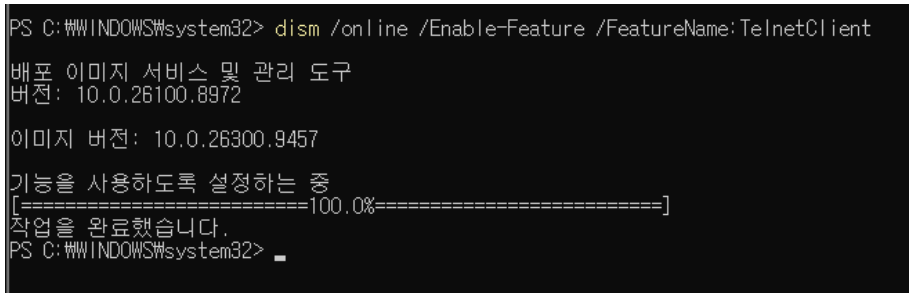
```
#:STANDARD: These are standard services.
127.0.0.1:telnet        stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/telnetd
```

```
(base) pel-miin@miinpel:~/TELNET$ sudo service openbsd-inetd restart
sudo ss -ltnp | grep ':23 '
LISTEN 0      128          127.0.0.1:23        0.0.0.0:*    users:(("inetd",pid=1419,fd=7))
```

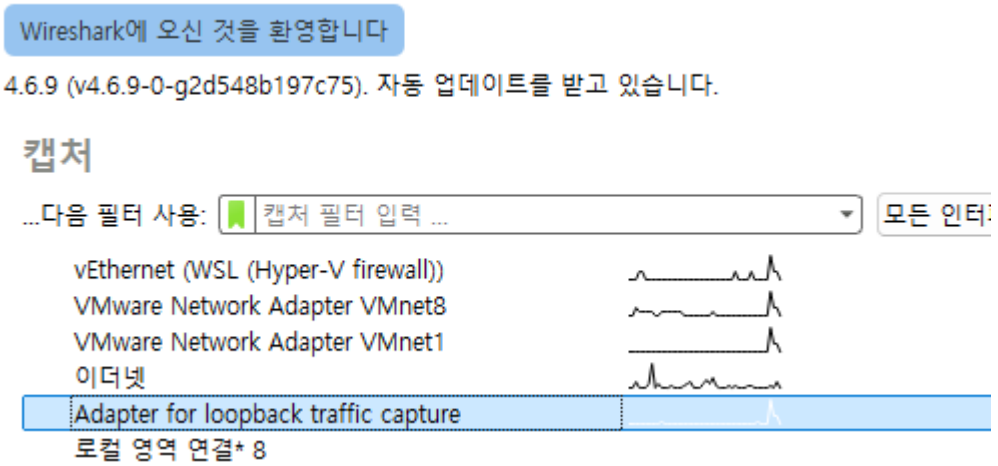
      - ‘Listen’으로 TCP 23번 포트 활성화 확인

부록 #3

- TELNET 실습
  - 환경 구성
    - Windows TELNET 클라이언트 활성화



    - 와이어 샤크 설치



부록 #3

- TELNET 실습
  - 원격 로그인
    - TELNET 접속
      - TCP 23번 포트로 접속 요청

```
PS C:\WINDOWS\system32> telnet localhost 23
연결 대상 localhost...
```

      - TCP 3-Way Handshake를 통한 클라이언트와 서버 간 연결 설정

frame.number >= 47 && frame.number <= 49

No.	Time	Source	Destination	Protocol	Lengt	Info
47	14.283091200	127.0.0.1	127.0.0.1	TCP	56	56904 → 23 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
48	14.283157100	127.0.0.1	127.0.0.1	TCP	56	23 → 56904 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
49	14.283174700	127.0.0.1	127.0.0.1	TCP	44	56904 → 23 [ACK] Seq=1 Ack=1 Win=65280 Len=0

부록 #3

- TELNET 실습
  - 원격 로그인
    - TELNET 접속
      - WSL 내 사용자 이름과 비밀번호로 로그인

Telnet localhost

```
Linux 6.6.87.2-microsoft-standard-WSL2 (miinpel.localdomain) (pts/2)

miinpel login: pel-miin

Password:
```

```
(base) pel-miin@miinpel:~/TELNET$ whoami
pel-miin
```

Wireshark · TCP 스트림 따라가기(tcp.stream eq 13) · Adapter for loopback traffic capture

```
..%..&..... ..#..'..$
..%..&........ ..#..'..$
.....'.........
......T....'.......ANSI..
........".....!
........".....!
............
............

Linux 6.6.87.2-microsoft-standard-WSL2 (miinpel.localdomain) (pts/2)

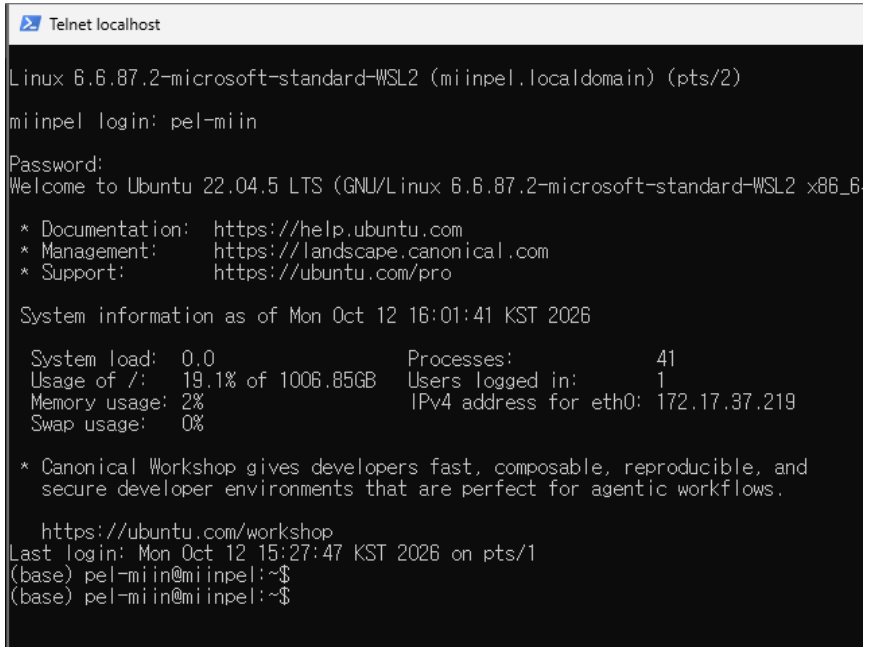
miinpel login:
p
p
e
e
l
l
-
-
m
m
i
i
i
i
n
n



Password:
```

부록 #3

- TELNET 실습
  - 원격 로그인
    - TELNET 접속
      - 로그인 성공




```
Password:


Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.6.87.2-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Mon Oct 12 16:01:41 KST 2026

  System load:  0.0                Processes:             41
  Usage of /:   19.1% of 1006.85GB   Users logged in:       1
  Memory usage: 2%                 IPv4 address for eth0: 172.17.37.219
  Swap usage:   0%

 * Canonical Workshop gives developers fast, composable, reproducible, and
   secure developer environments that are perfect for agentic workflows.

   https://ubuntu.com/workshop
Last login: Mon Oct 12 15:27:47 KST 2026 on pts/1
.[?2004h(base) pel-miin@miinpel:~$
.[?2004l
..[?2004h(base) pel-miin@miinpel:~$
```

부록 #3

- TELNET 실습
  - 원격 로그인
    - 원격 명령어 수행

```
ast login: Mon Oct 12 15:27:47 KST 2026 on pts/1
base) pel-miin@miinpel:~$
base) pel-miin@miinpel:~$ whoami
el-miin
base) pel-miin@miinpel:~$
base) pel-miin@miinpel:~$ hostname
iinpel
base) pel-miin@miinpel:~$
base) pel-miin@miinpel:~$ pwd
home/pel-miin
base) pel-miin@miinpel:~$
base) pel-miin@miinpel:~$
```

    - 종료

```
(base) pel-miin@miinpel:~$ exit
logout


호스트에 대한 연결을 잃었습니다.
PS C:₩WINDOWS₩system32> _
```

1529	1073.8778411…	127.0.0.1	127.0.0.1	TCP	44 23 → 56763 [FIN, ACK] Seq=1374 Ack=125 Win=65280 Len=0
1530	1073.8778595…	127.0.0.1	127.0.0.1	TCP	44 56763 → 23 [ACK] Seq=125 Ack=1375 Win=64000 Len=0
1531	1073.8779645…	127.0.0.1	127.0.0.1	TCP	44 56763 → 23 [FIN, ACK] Seq=125 Ack=1375 Win=64000 Len=0
1532	1073.8779981…	127.0.0.1	127.0.0.1	TCP	44 23 → 56763 [ACK] Seq=1375 Ack=126 Win=65280 Len=0

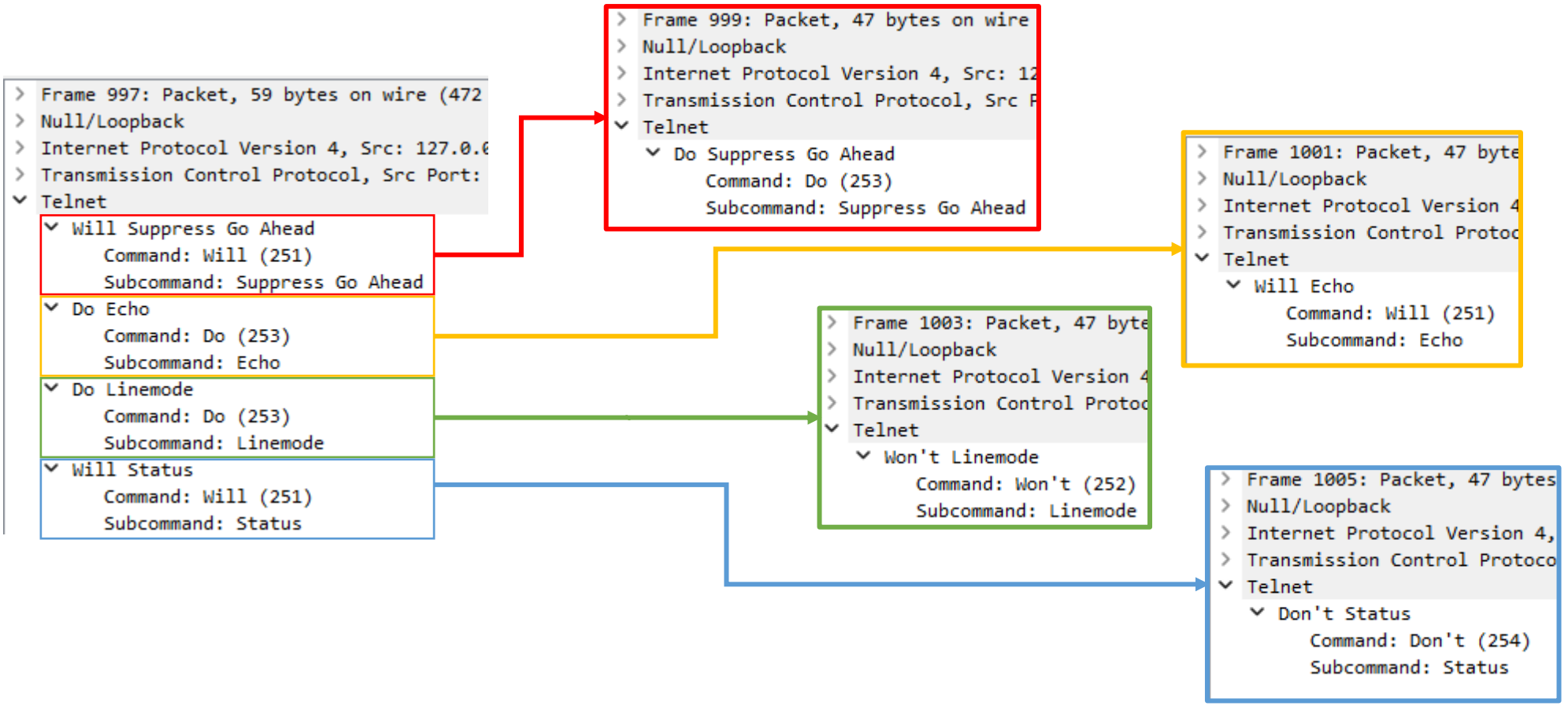
부록 #3

- TELNET 실습
  - 옵션 협상
    - 연결 설정 후 4가지 조건에 대해 (자동으로) 협상 진행

```
> Frame 997: Packet, 59 bytes on wire (472 bits), 59 bytes captured (472 bits) on interface \Device\NPF_Loopback, id 0
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 23, Dst Port: 56763, Seq: 37, Ack: 50, Len: 15
v Telnet
  v Will Suppress Go Ahead
      Command: Will (251)
      Subcommand: Suppress Go Ahead
  v Do Echo
      Command: Do (253)
      Subcommand: Echo
  v Do Linemode
      Command: Do (253)
      Subcommand: Linemode
  v Will Status
      Command: Will (251)
      Subcommand: Status
  v Do Remote Flow Control
      Command: Do (253)
      Subcommand: Remote Flow Control
```

부록 #3

- TELNET 실습
  - 옵션 협상
    - 연결 설정 후 4가지 조건에 대해 (자동으로) 협상 진행



부록 #3

- TELNET 실습
  - 옵션 협상
    - 파이썬 코드를 통한 협상 재현

```
= RESTART: C:₩Users₩minhe₩Desktop₩PEL₩세미나₩TCP,IP 프로토콜 - 20, 21장₩실습₩tel
net_manual.py
TELNET 서버 연결 성공
초기 수신: ff fb 25 ff fb 26 ff fd 18 ff fd 20 ff fd 23 ff fd 27 ff fd 24
Enter를 누르면 DO ECHO를 전송합니다.
전송: IAC DO ECHO
HEX: FF FD 01
서버 응답: ff fb 01
Enter를 누르면 연결을 종료합니다.
```

tcp.port == 23

No.	Time	Source	Destination	Protocol	Lengt	Info
26	4.203573800	127.0.0.1	127.0.0.1	TCP	56	53554 → 23 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK
27	4.203610100	127.0.0.1	127.0.0.1	TCP	56	23 → 53554 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495
28	4.203623300	127.0.0.1	127.0.0.1	TCP	44	53554 → 23 [ACK] Seq=1 Ack=1 Win=65280 Len=0
35	4.206973700	127.0.0.1	127.0.0.1	TELNET	65	Will Authentication Option, Will Encryption Option, Do Termi
36	4.206981500	127.0.0.1	127.0.0.1	TCP	44	53554 → 23 [ACK] Seq=1 Ack=22 Win=65280 Len=0
259	117.344174400	127.0.0.1	127.0.0.1	TELNET	47	Do Echo
260	117.344189200	127.0.0.1	127.0.0.1	TCP	44	23 → 53554 [ACK] Seq=22 Ack=4 Win=65280 Len=0
263	117.344600600	127.0.0.1	127.0.0.1	TELNET	47	Will Echo
264	117.344607800	127.0.0.1	127.0.0.1	TCP	44	53554 → 23 [ACK] Seq=4 Ack=25 Win=65280 Len=0

부록 #3

- TELNET 실습
  - 옵션 협상
    - 파이썬 코드를 통한 협상 재현
      - 협상 요청

```
> Frame 259: Packet, 47 bytes on wire (376 bits), 47 bytes captured (376 bits) on interface \
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 53554, Dst Port: 23, Seq: 1, Ack: 22, Len: 3
v Telnet
  > Do Echo
```

```
0000  02 00 00 00 45 00 00 2b  4b 21 40 00 80 06 b1 a9
0010  7f 00 00 01 7f 00 00 01  d1 32 00 17 8a 71 e4 e3
0020  b1 e0 6b 20 50 18 00 ff  52 2a 00 00 ff fd 01
```

      - 협상 응답

```
> Frame 263: Packet, 47 bytes on wire (376 bits), 47 bytes captured (376 bits) on interface \Device
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 23, Dst Port: 53554, Seq: 22, Ack: 4, Len: 3
v Telnet
  > Will Echo
```

```
0000  02 00 00 00 45 00 00 2b  4b 25 40 00 80 06 b1 a5
0010  7f 00 00 01 7f 00 00 01  00 17 d1 32 b1 e0 6b 20
0020  8a 71 e4 e6 50 18 00 ff  52 29 00 00 ff fb 01
```

부록 #3

- TELNET 실습
  - 옵션 협상
    - 파이썬 코드를 통한 협상 재현

```
1.  import socket
2.  import time
3.  HOST = "127.0.0.1"
4.  PORT = 23
5.  # TELNET 제어 코드
6.  IAC = 255
7.  WILL = 251
8.  WONT = 252
9.  DO = 253
10. DONT = 254
11. # TELNET 옵션
12. ECHO = 1
13. sock = socket.create_connection((HOST, PORT),
    timeout=5)
14. sock.settimeout(0.5)
15. print("TELNET 서버 연결 성공")
16. # 서버가 보내는 초기 협상 데이터 수신
17. time.sleep(0.5)
```

```
18. try:
19.     initial = sock.recv(4096)
20.     print("초기 수신:", initial.hex(" "))
21. except socket.timeout:
22.     print("초기 수신 데이터 없음")
23. # 직접 옵션 협상
24. input("Enter를 누르면 DO ECHO를 전송합니다.")
25. request = bytes([IAC, DO, ECHO])
26. sock.sendall(request)
27. print("전송: IAC DO ECHO")
28. print("HEX: FF FD 01")
29. # 응답 수신
30. time.sleep(0.5)
31. try:
32.     response = sock.recv(4096)
33.     print("서버 응답:", response.hex(" "))
34. except socket.timeout:
35.     print("응답 없음")
36. input("Enter를 누르면 연결을 종료합니다.")
37. sock.close()
```

부록 #3

- TELNET 실습
  - 서브 옵션 협상

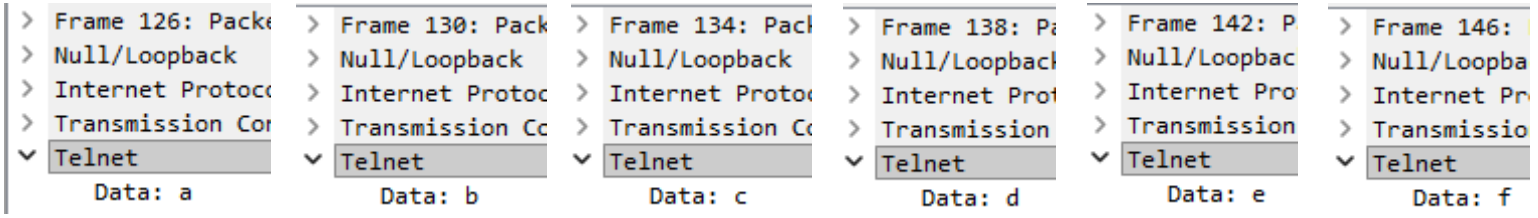
```
> Frame 995: Packet, 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF_Loopback, id 0
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 56763, Dst Port: 23, Seq: 40, Ack: 37, Len: 10
v Telnet
  v Suboption Terminal Type
      Command: Suboption (250)
    v Subcommand: Terminal Type
        Here's my Terminal Type
        Value: ANSI
  v Suboption End
      Command: Suboption End (240)
```

부록 #3

- TELNET 실습
  - 운영모드
    - 문자모드

```
(base) pel-miin@miinpel:~$
(base) pel-miin@miinpel:~$ abcdef
```

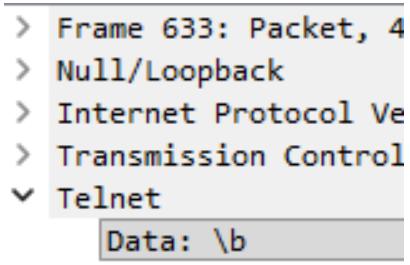
```
126 28.916410900  127.0.0.1  127.0.0.1  TELNET  45 1 byte data
130 29.284843000  127.0.0.1  127.0.0.1  TELNET  45 1 byte data
134 29.571768000  127.0.0.1  127.0.0.1  TELNET  45 1 byte data
138 29.841661800  127.0.0.1  127.0.0.1  TELNET  45 1 byte data
142 30.064584300  127.0.0.1  127.0.0.1  TELNET  45 1 byte data
146 30.261298500  127.0.0.1  127.0.0.1  TELNET  45 1 byte data
```



부록 #3

- TELNET 실습
  - 운영모드
    - 문자모드
      - backspace 입력

```
(base) pel-miin@miinpel:~$
(base) pel-miin@miinpel:~$ abcde
```




```
0000  02 00 00 00 45 00 00 29  49 42 40 00 80 06 b3 8a
0010  7f 00 00 01 7f 00 00 01  c5 c1 00 17 47 60 49 fc
0020  a3 7d 4b 95 50 18 00 fb  62 86 00 00 08
```

부록#4

- 파일 전송 프로토콜(FTP) 명령어 (1/5)
  - 액세스 명령어
    - 원격 FTP 시스템의 접속·인증 및 세션 제어

[표 7] 액세스 명령어

명령어	인수	설명
USER	사용자 ID	사용자 정보
PASS	사용자 비밀번호	비밀번호
ACCT	과금할 계정	계정 정보
REIN	-	재초기화
QUIT	-	시스템 로그아웃
ABOR	-	이전 명령 중단

부록#4

- 파일 전송 프로토콜(FTP) 명령어 (2/5)
  - 파일 관리 명령어
    - 원격 시스템의 디렉터리 및 파일 관리

[표 8] 파일 관리 명령어

명령어	인수	설명
CWD	경로명	다른 디렉터리로 이동
CDUP	-	상위 디렉터리로 이동
DELE	경로명	파일 삭제
LIST	경로명 (선택적)	하위 디렉터리 또는 파일 목록 조회
NLST	경로명 (선택적)	속성 정보를 제외한 하위 디렉터리 또는 파일 목록 조회
MKD	디렉터리 이름	새 디렉터리 생성
PWD	-	현재 디렉터리 이름 표시
RMD	디렉터리 이름	디렉터리 삭제
RNFR	기존 파일 이름	이름을 변경할 파일 지정
RNTO	새 파일 이름	파일 이름 변경
SMNT	경로명	파일 시스템 마운트

부록#4

- 파일 전송 프로토콜(FTP) 명령어 (3/5)
  - 데이터 형식화 명령어
    - 파일 전송 전에 사용할 데이터 표현 및 전송 방식 지정

[표 9] 데이터 형식화 명령어

명령어	인수	설명
TYPE	A[N/T/C] (ASCII), E[N/T/C] (EBCDIC), I (Image), L <byte size> (Local byte)	파일 종류 정의
STRU	F (File), R (Record), or P (Page)	데이터 구조 정의
MODE	S (Stream), B (Block), or C (Compressed)	전송 모드 정의

  - 포트 정의 명령어
    - 데이터 연결에 사용할 포트 결정

[표 10] 포트 정의 명령어

명령어	인수	설명
PORT	IPv4 주소 4 부분 + 포트 번호 2 부분	클라이언트가 포트 선택
PASV	-	서버가 포트 선택

부록#4

- 파일 전송 프로토콜(FTP) 명령어 (4/5)
  - 파일 전송 명령어
    - 실제 파일 송수신 및 전송 제어

[표 11] 파일 전송 명령어

명령어	인수	설명
RETR	경로명 (단일)	서버의 파일을 클라이언트로 전송
STOR	경로명 (단일)	클라이언트의 파일을 서버에 저장
APPE	경로명 (단일)	STOR과 유사하나, 파일이 존재하면 기존 파일 뒤에 데이터 추가
STOU	-	STOR과 동일하나, 디렉터리 내에서 고유한 파일 이름으로 저장
ALLO	10진수 정수	서버에 파일 저장 공간 할당
REST	재시작 지점	지정된 데이터 위치에 파일 마커 설정
STAT	경로명 (선택적)	파일 상태 정보 반환

부록#4

- 파일 전송 프로토콜(FTP) 명령어 (5/5)
  - 기타 명령어
    - FTP 서버의 상태 및 부가 정보 확인

[표 12] 기타 명령어

명령어	인수	설명
HELP	문자열 (선택적)	서버에 대한 정보 요청
NOOP	-	서버의 동작 여부 확인
SITE	문자열	서버별 특정 명령 지정
SYST	-	서버가 사용하는 운영체제 정보 요청

부록#5

- 파일 전송 프로토콜(FTP) 명령 응답 (1/3)
  - 구성(xyz)
    - 첫 번째 숫자(x): 처리 결과
    - 두 번째 숫자(y): 응답 종류
    - 세 번째 숫자(z): 세부 상태

[표 13] FTP 명령 응답의 첫 번째 숫자

값(x)	분류	의미
1	긍정적 예비 응답	명령 처리 시작, 추가 응답 예정
2	긍정적 완료 응답	명령 처리 정상 완료
3	긍정적 중간 응답	명령 수락 후 추가 정보 필요
4	일시 부정적 완료 응답	일시적 오류로 실패, 추후 재시도 가능
5	영구 부정적 완료 응답	명령 처리 실패, 동일 요청의 단순 재시도로 해결 불가

[표 14] FTP 명령 응답의 두 번째 숫자

값(y)	의미
0	문법 관련
1	정보 관련
2	연결 관련
3	인증 및 계정 관련
4	미지정
5	파일 시스템 관련

부록#5

- 파일 전송 프로토콜(FTP) 명령 응답 (2/3)

[표 15] FTP 명령 응답

코드값	설명
긍정적 예비 응답(Positive Preliminary Reply)	
120	서비스가 곧 준비됨
125	데이터 연결이 열림; 곧 데이터 전송 시작
150	파일 상태 정상; 데이터 연결이 곧 열림

코드값	설명
긍정적 완료 응답(Positive Completion Reply)	
200	명령 정상 처리
211	시스템 상태 또는 도움말 응답
212	디렉터리 상태
213	파일 상태
214	도움말 메시지
215	시스템 종류(운영체제) 정보
220	서비스 준비 완료
221	서비스 종료
225	데이터 연결 열림
226	데이터 연결 종료
227	수동 모드 진입; 서버 IP 주소와 포트 번호 전달
230	사용자 로그인 성공
250	요청한 파일 작업 정상 처리

부록#5

- 파일 전송 프로토콜(FTP) 명령 응답 (3/3)

[표 15] FTP 명령 응답

코드값	설명
긍정적 중간 응답(Positive Intermediate Reply)	
331	사용자 이름 확인; 비밀번호 필요
332	로그인에 계정 정보 필요
350	파일 작업 보류; 추가 정보 필요

코드값	설명
일시적 부정 완료 응답(Transient Negative Completion Reply)	
425	데이터 연결을 열 수 없음
426	연결 종료; 전송 중단
450	파일 사용 불가로 작업 수행 불가
451	로컬 오류로 작업 중단
452	저장 공간 부족으로 작업 중단

코드값	설명
영구적 부정 완료 응답(Permanent Negative Completion Reply)	
500	구문오류 또는 인식할 수 없는 명령
501	매개변수 또는 인자 구문 오류
502	구현되지 않은 명령
503	잘못된 명령 순서
504	구현되지 않은 명령 매개변수
530	로그인되지 않은 사용자
532	파일 저장을 위한 계정 정보 필요
550	요청 작업 실패; 파일 사용 불가
552	저장 공간 할당량 초과로 요청 작업 중단
553	허용되지 않은 파일 이름으로 요청 작업 수행 불가