네트워크 I

(13주차)

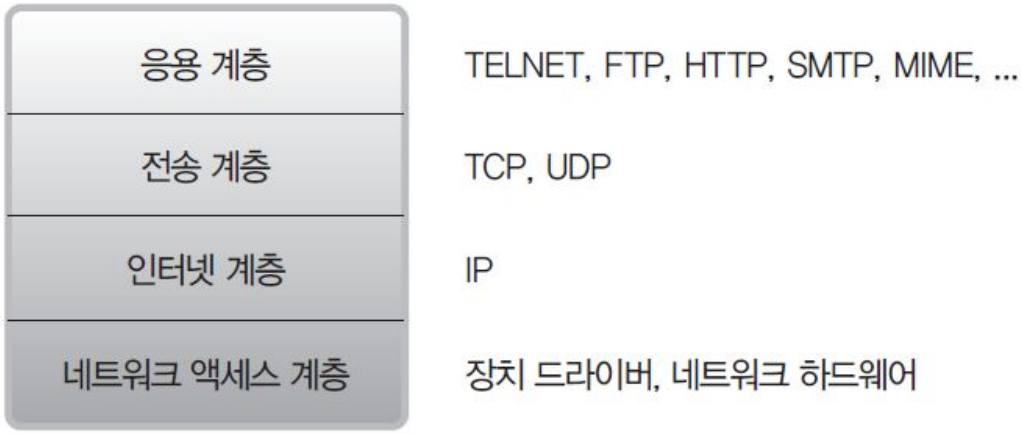
학습개요

- 학습 목표
  - TCP/IP 기초를 다진다.
  - MFC를 이용한 소켓 프로그래밍 기법을 익힌다.

- 학습 내용
  - TCP/IP 기초
  - 윈도우 소켓
  - MFC 소켓 프로그래밍 (콘솔)
  - 실습

TCP/IP 개요

- TCP/IP 구조



TCP/IP 개요

- 계층별 역할
  - 네트워크 액세스 계층
    - 물리적인 신호를 처리하여 실제 데이터를 보내고 받는 네트워크 하드웨어와 장치 드라이버를 포함
    - 인터넷 계층에서 내려 보낸 데이터를 처리하여 통신 상대(Peer)에게 보내고, 통신 상대로부터 받은 데이터를 처리하여 인터넷 계층으로 전달
    - 물리 주소(Physical Address)를 사용하여 통신을 수행
  - 인터넷 계층
    - 전송 계층이 내려 보낸 데이터를 네트워크 액세스 계층의 도움을 받아 목적지까지 전달
    - IP 주소(Internet Protocol Address)를 이용하여 주소 지정과 라우팅 수행

TCP/IP 개요

- 계층별 역할
  - 전송 계층
    - 최종적인 통신 목적지 지정
      ⇨ 포트 번호(Port Number) 이용
    - 데이터를 오류 없이 전송

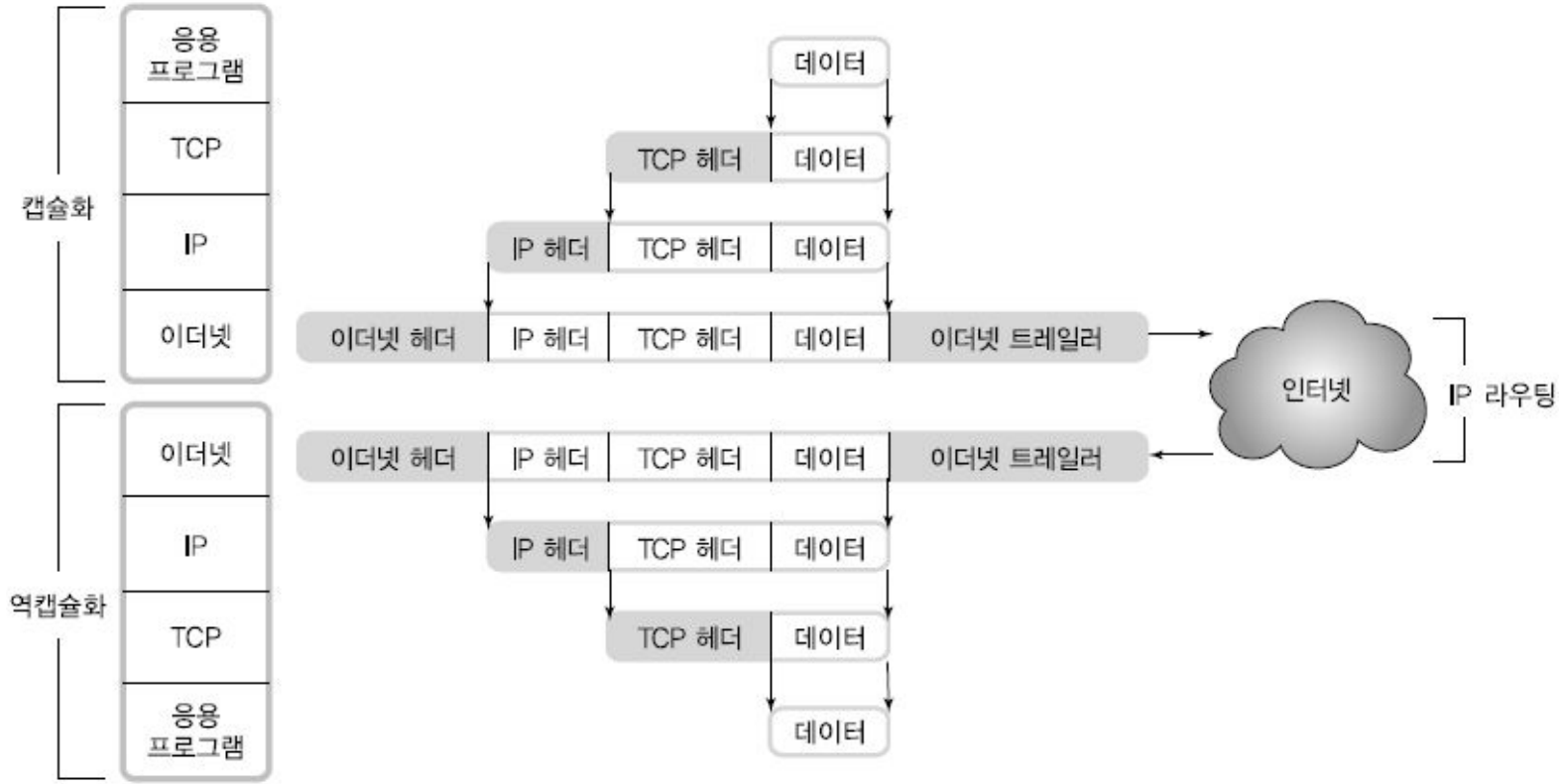
TCP	UDP
연결형 프로토콜	비연결형 프로토콜
1 대 1 통신(Unicast)	1 대 1 통신(Unicast), 1 대 다 통신(Broadcast), 다 대 다 통신(Multicast)
신뢰성 있는 데이터 전송	신뢰성 없는 데이터 전송
데이터 경계 구분 안 함	데이터 경계 구분함

TCP/IP 개요

- 계층별 역할
  - 응용 프로그램 계층
    - 다양한 응용 프로그램을 위한 프로토콜을 제공. 전송 계층을 기반으로 만들어진 다수의 프로토콜 포함
      - (예) FTP, TELNET, HTTP, ...

TCP/IP 개요

- 데이터 전송 과정



IP 주소, 도메인 이름, 포트 번호

- IP 주소
  - 32비트 숫자
  - 8비트 단위로 구분하여 각각 10진수로 표시
    - (예) 147.46.114.70
  - 전 세계적으로 유일하며, IP 라우팅에 사용됨
- 도메인 이름(Domain Name)
  - 사람이 기억하고 사용하기 쉬운 이름
  - IP 주소로 변환해야 통신 가능

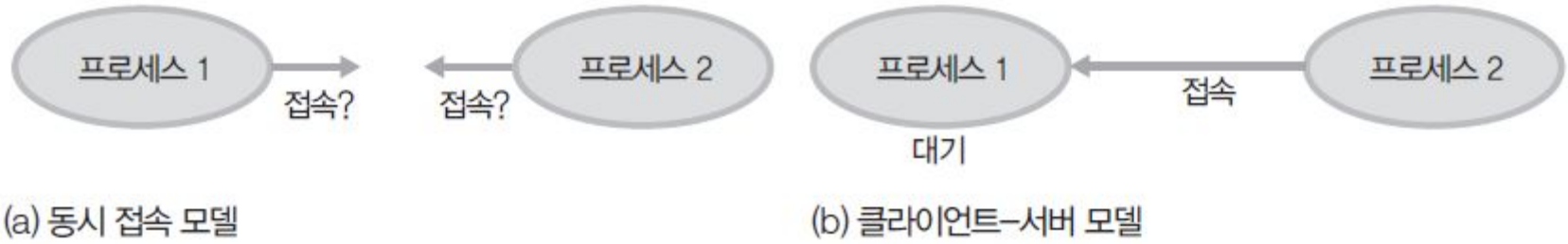
IP 주소, 도메인 이름, 포트 번호

- 포트 번호
  - 목적지 프로세스를 지정
    ※ IP 주소와 도메인 이름은 목적지 호스트를 지정
  - 부호없는 16비트 정수를 사용(0~65535)

포트 번호	분류
0~1023	시스템 포트 (System Ports)
1024~49151	사용자 포트(User Ports)
49152~65535	동적/사설 포트(Dynamic and/or Private Ports)

클라이언트-서버 모델

- 두 프로세스 간 통신

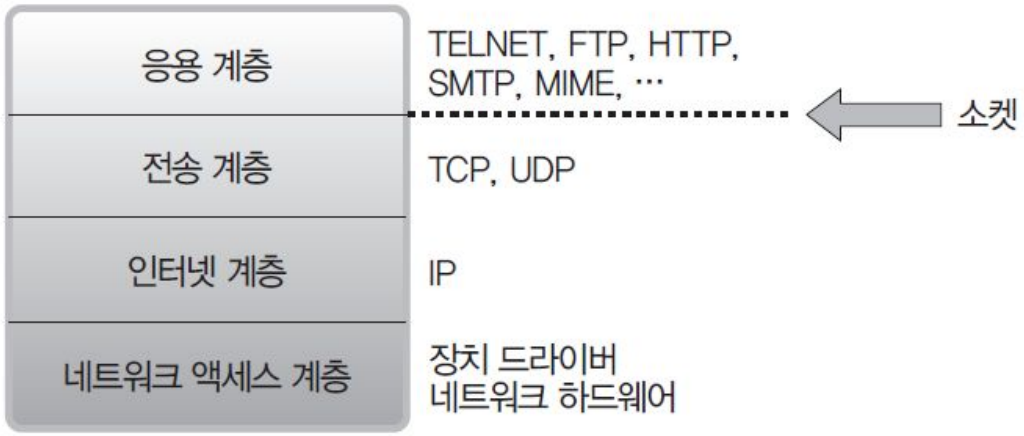


(a) 동시 접속 모델

(b) 클라이언트–서버 모델

윈도우 소켓

- 소켓
  - 다양한 프로토콜을 일관된 인터페이스로 다룰 수 있게 만든 함수 집합
    - 윈도우는 API로, 유닉스/리눅스는 시스템 콜로 제공됨

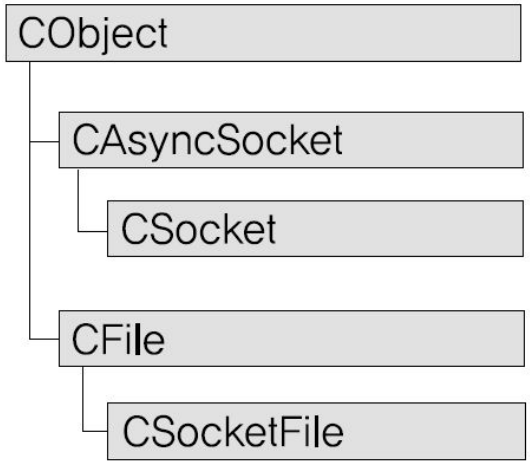


윈도우 소켓

- 윈도우 소켓
  - 윈도우 소켓 1.x 버전
    - TCP/IP 프로토콜과 버클리 소켓 호환 함수를 주로 지원
      - MFC의 소켓 클래스는 윈도우 소켓 1.1 버전만 지원
  - 윈도우 소켓 2.x 버전
    - 다양한 종류의 프로토콜을 지원하도록 구조를 변경하고 새로운 입출력 함수를 추가하여, 고성능 서버 프로그램 제작을 위한 기능을 제공

MFC 소켓 클래스

- MFC 클래스 계층도



MFC 소켓 프로그래밍 - 콘솔

- TCP 서버 (1/4)

```
01 void ErrQuit(int err)
02 {
03   LPVOID lpMsgBuf;
04   FormatMessage(
05     FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
06     NULL, err,
07     MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
08     (LPTSTR)&lpMsgBuf, 0, NULL);
09   MessageBox(NULL, (LPCTSTR)lpMsgBuf, _T("오류 발생"), MB_ICONERROR);
10   LocalFree(lpMsgBuf);
11   exit(1);
12 }
13
14 int _tmain(int argc, TCHAR* argv[], TCHAR* envp[])
15 {
16   int nRetCode = 0;
17
18   HMODULE hModule = ::GetModuleHandle(NULL);
```

MFC 소켓 프로그래밍 - 콘솔

- TCP 서버 (2/4)

```
19
20    if(hModule != NULL)
21    {
22      if(!AfxWinInit(hModule, NULL, ::GetCommandLine(), 0))
23      {
24        _tprintf(_T("심각한 오류: MFC를 초기화하지 못했습니다.₩n"));
25        nRetCode = 1;
26      }
27      else
28      {
29        _tsetlocale(LC_ALL, _T(""));
30        AfxSocketInit();
31
32        CSocket sock;
33        if(!sock.Create(8000))
34          ErrQuit(sock.GetLastError());
35
36        if(!sock.Listen())
37          ErrQuit(sock.GetLastError());
```

MFC 소켓 프로그래밍 - 콘솔

- TCP 서버 (3/4)

```
38
39        TCHAR buf[256+1];
40        int nbytes;
41
42        while(1){
43          CSocket newsock;
44          if(!sock.Accept(newsock))
45            ErrQuit(sock.GetLastError());
46
47            CString PeerAddress;
48            UINT PeerPort;
49            newsock.GetPeerName(PeerAddress, PeerPort);
50            _tprintf(_T("### IP 주소: %s, 포트 번호: %d ###Wn"),
51              (LPCTSTR)PeerAddress, PeerPort);
52
53            while(1){
54              nbytes = newsock.Receive(buf, 256);
55              if(nbytes = = 0 | | nbytes = = SOCKET_ERROR){
56                break;
```

MFC 소켓 프로그래밍 - 콘솔

- TCP 서버 (4/4)

```
57              }
58              else{
59                buf[nbytes] = _T('₩0');
60                _tprintf(_T("%s"), buf);
61              }
62            }
63            newsock.Close();
64            _tprintf(_T("### 접속 종료 ###₩n₩n"));
65          }
66        }
67      }
68      else
69      {
70        _tprintf(_T("심각한 오류: GetModuleHandle 실패₩n"));
71        nRetCode = 1;
72      }
73
74      return nRetCode;
75  }
```

MFC 소켓 프로그래밍 - 콘솔

- TCP 클라이언트 (1/3)

```
01 void ErrQuit(int err)
02 {
... // TCPServer1.cpp 파일의 ErrQuit() 함수와 동일하다.
12 }
13
14 int_tmain(int argc, TCHAR* argv[], TCHAR* envp[])
15 {
16   int nRetCode = 0;
17
18   HMODULE hModule = ::GetModuleHandle(NULL);
19
20   if(hModule != NULL)
21   {
22     if(!AfxWinInit(hModule, NULL, ::GetCommandLine(), 0))
23     {
24       _tprintf(_T("심각한 오류: MFC를 초기화하지 못했습니다.₩n"));
25       nRetCode = 1;
26     }
```

MFC 소켓 프로그래밍 - 콘솔

- TCP 클라이언트 (2/3)

```
27      else
28      {
29        _tsetlocale(LC_ALL, _T(""));
30        AfxSocketInit();
31
32        CSocket sock;
33        if(!sock.Create())
34          ErrQuit(sock.GetLastError());
35
36        if(!sock.Connect(_T("127.0.0.1"), 8000))
37          ErrQuit(sock.GetLastError());
38
39        TCHAR buf[256];
40        int nbytes;
41
42        for(int i=0; i<5; i++){
43           wsprintf(buf, _T("%d번째 테스트 메시지₩r₩n"), i);
44           nbytes = sock.Send(buf, 256);
```

MFC 소켓 프로그래밍 - 콘솔

- TCP 클라이언트 (3/3)

```
45          if(nbytes = = SOCKET_ERROR)
46            ErrQuit(sock.GetLastError());
47          else{
48            _tprintf(_T("<%d> %d바이트 전송₩n"), i, nbytes);
49            Sleep(1000);
50          }
51        }
52        sock.Close();
53      }
54    }
55    else
56    {
57      _tprintf(_T("심각한 오류: GetModuleHandle 실패₩n"));
58      nRetCode = 1;
59    }
60
61    return nRetCode;
62 }
```

주요 함수

- 서버 코드

```
BOOL CSocket::Create(
    UINT nSocketPort = 0,
    int nSocketType = SOCK_STREAM,
    LPCTSTR lpszSocketAddress = NULL
);
```

```
BOOL CAsyncSocket::Listen(int nConnectionBacklog = 5);
```

주요 함수

- 서버 코드

```
BOOL CAsyncSocket::Accept(
    CAsyncSocket& rConnectedSocket,
    SOCKADDR* lpSockAddr = NULL,
    int* lpSockAddrLen = NULL
);
```

```
BOOL CAsyncSocket::GetPeerName(
    CString& rPeerAddress,
    UINT& rPeerPort
);
```

주요 함수

- 서버 코드

```
int CAsyncSocket::Receive(
    void* lpBuf,
    int nBufLen,
    int nFlags = 0
);
```

```
void CAsyncSocket::Close();
```

주요 함수

- 클라이언트 코드

```
BOOL CAsyncSocket::Connect(
    LPCTSTR lpszHostAddress,
    UINT nHostPort
);
```

```
int CAsyncSocket::Send(
    const void* lpBuf,
    int nBufLen,
    int nFlags = 0
);
```

실습

학습정리

- 인터넷 계층은 IP 주소(Internet Protocol Address)를 이용하여 주소 지정과 라우팅 수행을 담당합니다.
- 전송 계층은 최종적인 통신 목적지 지정을 위해 포트 번호(Port Number)를 이용하며, 데이터를 오류 없이 전송하는 역할을 담당합니다.
- TCP는 연결 지향형 프로토콜로 1 대 1 통신(Unicast)을 하며 신뢰성 있는 데이터 전송을 하는 특징을 가집니다.
- 소켓은 다양한 프로토콜을 일관된 인터페이스로 다룰 수 있게 만든 함수 집합으로 윈도우에서는 윈도우 소켓 API를 제공합니다.
- 윈도우 소켓 1.x 버전은 TCP/IP 프로토콜과 버클리 소켓 호환 함수를 주로 지원하며, MFC의 소켓 클래스는 윈도우 소켓 1.1 버전만 지원합니다.
- MFC 소켓 프로그래밍은 CAsyncSocket 클래스를 이용해 비동기 방식의 소켓 프로그래밍을, CAsyncSocket 클래스를 상속받은 CSocket 클래스를 이용해 동기 방식의 소켓 프로그래밍을 합니다.