chapter 7

Wide Area Networks

CHAPTER OBJECTIVES

- 회선 교환(circuit switching)방식의 연결을 살펴본다.
- 전용 회선(dedicated-circuit)서비스에 대해 알아본다.
- 패킷 교환(packet-switched)네트워크와 그 종류
- 다른 고속 통신 서비스에 대해 알아본다.
- 다중화(multiplexing)의 개념과 용도를 알아본다.

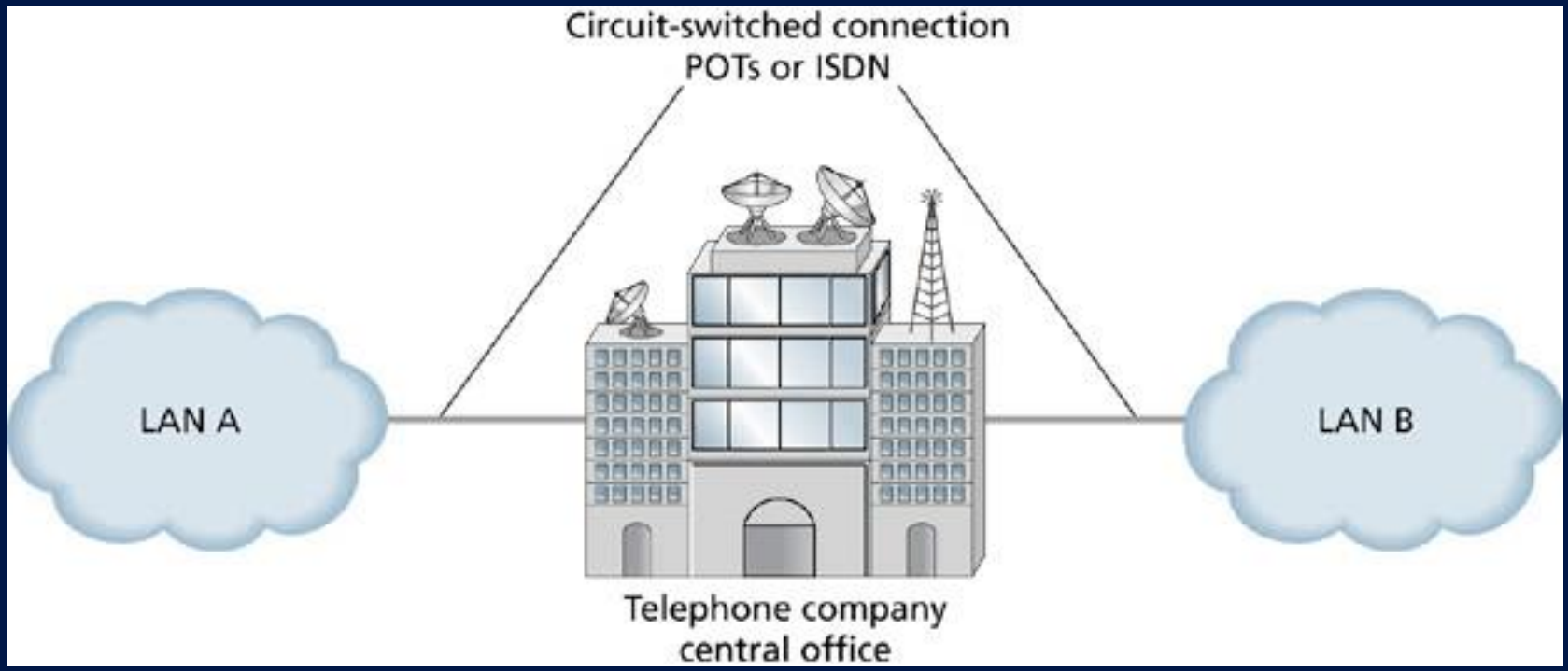
회선 교환

- 회선 교환(Circuit switching)은 통신점 A와 통신점 B를 직접 연결하여 연결되어 있는 동안 다른 연결이 끼어들지 않는 연결방식이다.
- 초기 전화망의 기본 연결방식이다.
- 회선 교환을 사용할 경우 전화망에서 모뎀으로 28.8Kbps에서 56Kbps의 속도가 나오며 전용기계로 1.544Mbps까지 나올 수 있다.
- 일반적으로 문자데이터를 전송하고, 계속 (365일 24시간)연결되어 있을 필요가 없을 때 사용.

회선 교환

- 기존의 PSTN으로 연결되는 것을 뜻한다
  - 원격 접속이 자유롭다. 전화가 되는 곳이면 어디든 연결할 수 있다. 전화번호를 눌러서 연결하고 전화를 끊을 때 까지 연결이 지속된다.
  - 연결되어 있는 동안 비용이 계속 올라간다. (전화요금과 동일하다.)

PSTN을 통한 LAN대 LAN연결



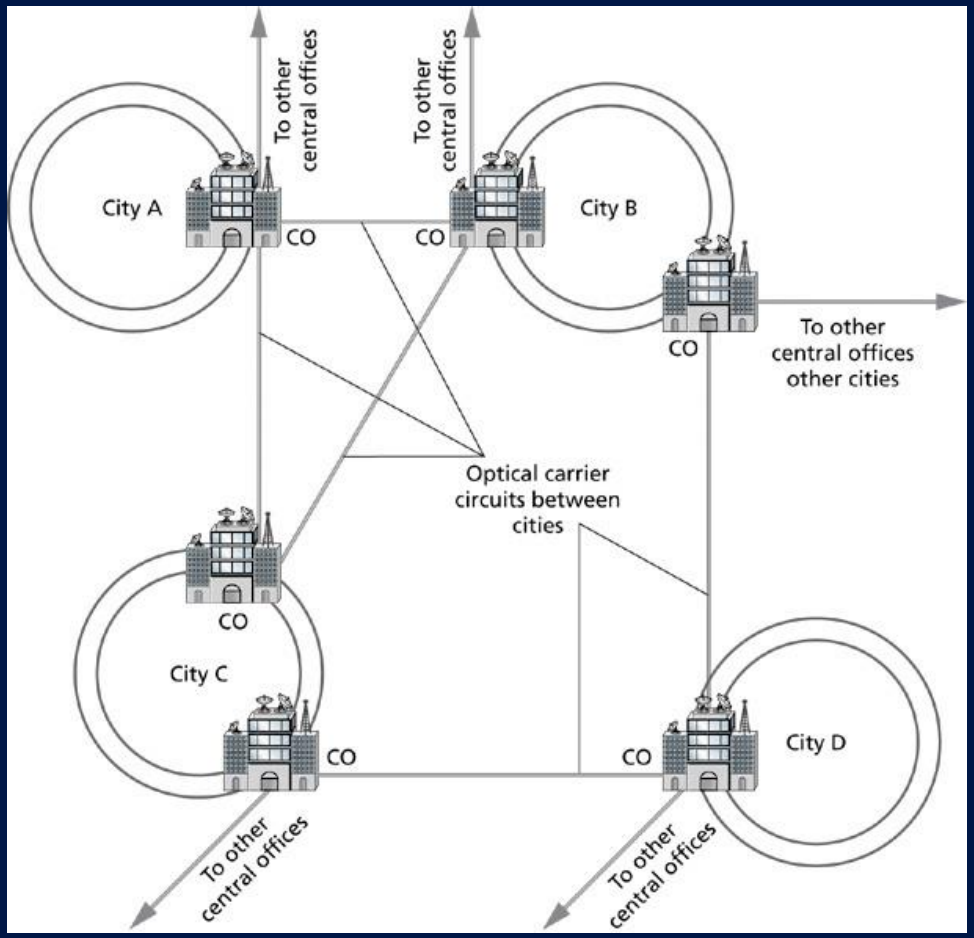
ISDN

- ISDN (Integrated Services Digital Network)
  - 디지털 회선 교환 연결방식이다.
  - 아날로그 전화선을 디지털로 바꾸기 위해 1960년대에 개발 되었다.
  - ISDN은 기존의 아날로그 전화회선과 매우 유사하다.
  - ISDN은 디지털로 표현될 수 있는 데이터, 그래픽, 동영상, 소리를 전달할 수 있고, 음성전달도 가능하다.
  - 미국에서는 AT&T의 독점시기에 보급되어 널리 퍼지지 않았다. 우리나라에서는 너무 비쌌고, 보급 시기가 늦어서 다른 기술에 밀려서 사장되었다.

SONET

- 동기식 광 통신망, SONET (Synchronous Optical Network)
  - 광케이블 고속 통신용 ANSI표준
  - 광케이블을 통해 여러 디지털 비트 연결을 유지하는 표준 프로토콜이다.
  - 여러가지 optical carrier level (OC-*x*)있고, 각각 속도가 정해져 있다.
    - OC-1은 50Mbps, OC-768은 39Gbps
  - 여러 개의 링으로 구성된 네트워크 구조를 갖는다. (원격 접속을 위한 선분 구조도 지원)

SONET Ring Infrastructure



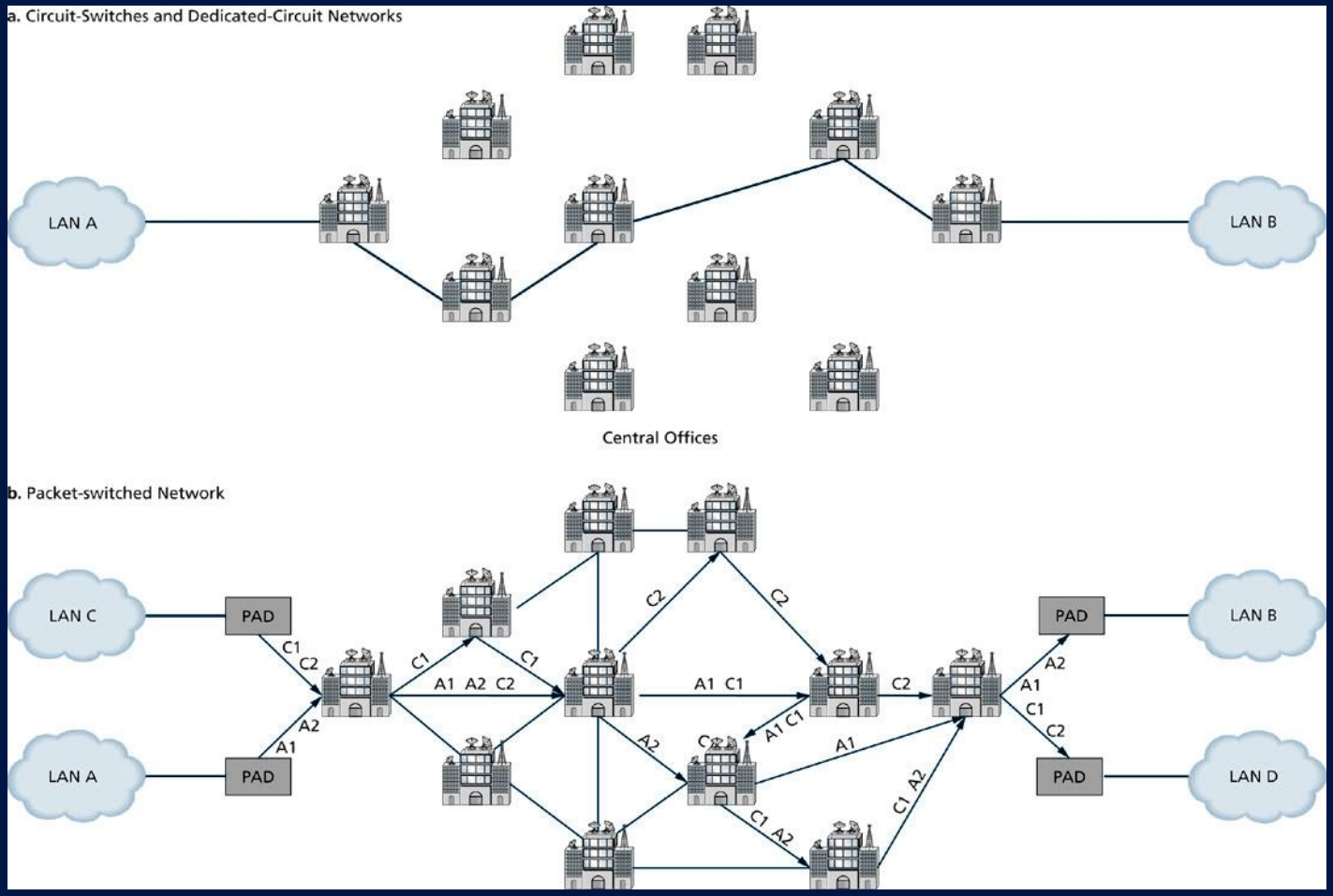
패킷 교환 망

- 패킷 교환 캐리어 서비스(Packet-Switched Carrier Services)
  - 보다 효율적인 데이터와 음성 전송을 위해 개발되었다. – 패킷 교환방식은 전송 회로가 노는 시간을 줄인다.
  - 패킷 교환망은 항상 켜져 있으며 전송 가능하다.
  - 패킷 교환망은 호출 준비(call setup)가 없다.
  - 연결된 점끼리 데이터를 주고 받지 않아도 회선의 낭비가 없다.

패킷 교환 망

- 패킷 교환 망
  - 연결을 그림으로 표시할 때 구름(Cloud)로 표현한다.
  - Public Data Network (PDN)라고도 표현한다.
  - 패킷 분해/조립기(assembler/disassembler (PAD))를 통해 PDN에 연결된다.

회선 교환 vs. 패킷 교환 망

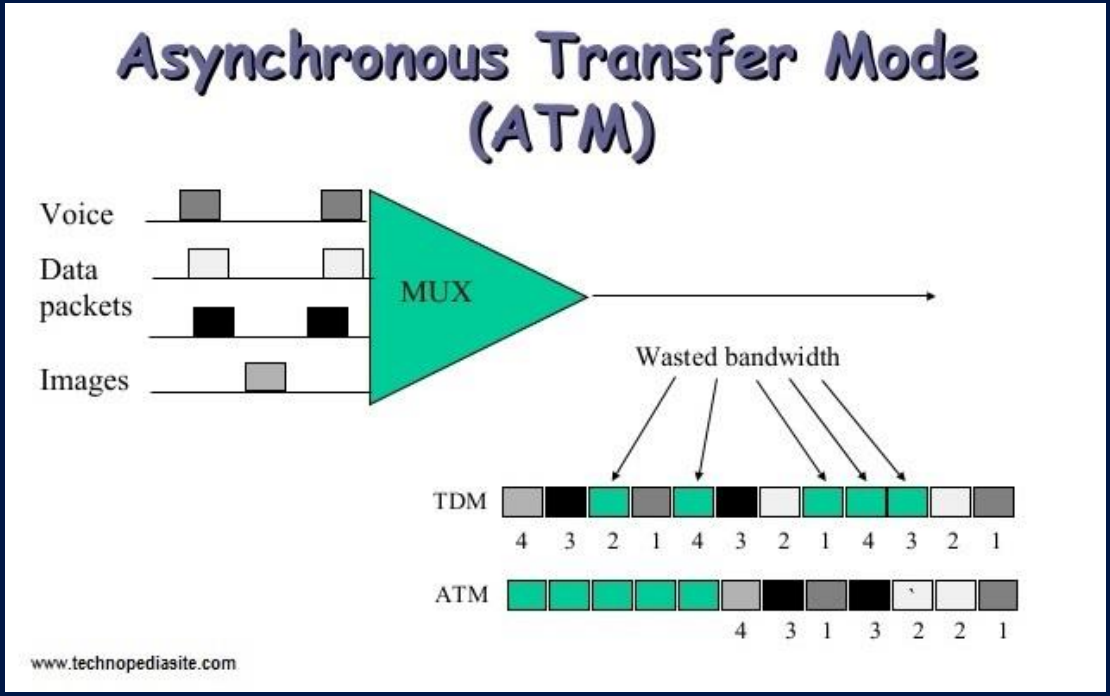


패킷 교환망 표준

- X.25
  - PSTN을 통한 컴퓨터끼리의 접속을 정의하며 최대 64Kbps까지 지원
- Frame Relay
  - 56 Kbps에서 45 Mbps 까지 지원
- Asynchronous Transfer Mode (ATM)
  - 현재 많이 사용되는 고속 패킷 교환 시스템

패킷 교환망 표준

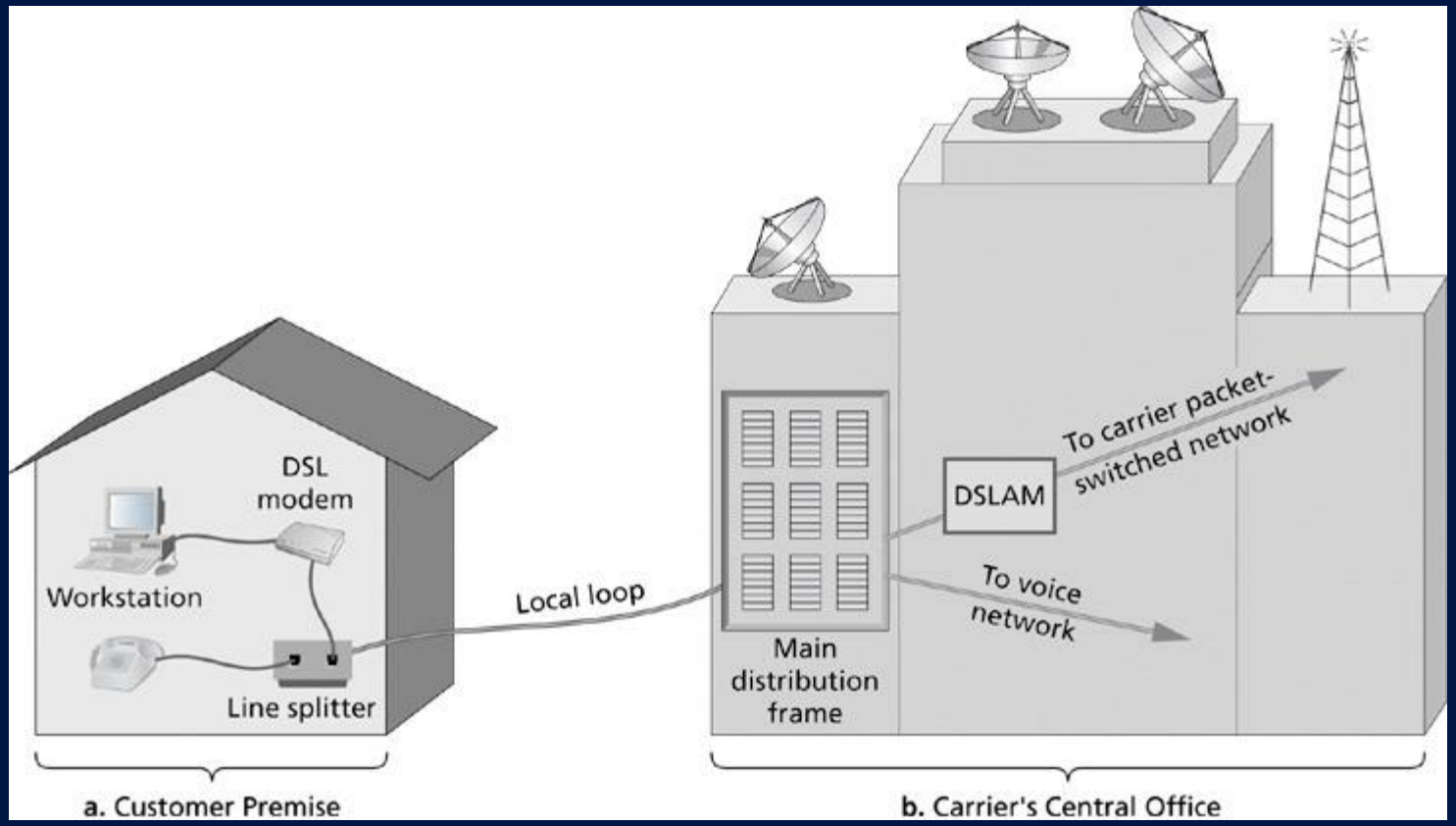
- ATM
  - 모든 데이터를 Cell(48바이트)로 바꾸어 전송
    - 그 중 5바이트가 추가된 헤더



디지털 가입자 회선 - DSL

- 디지털 가입자 회선 Digital Subscriber Line Technologies (DSL)
  - 전화선을 통해서 고속 디지털 데이터를 전송하는 시스템
  - DSL은 사용자와 전화국(CO) 두 곳에 별도의 기기를 설치해야 한다.
    - 이전 모뎀방식은 CO에 추가 장비가 필요 없다.
  - 사용자는 DSL모뎀과 분배기(line splitter)가 필요.
  - CO에는 음성과 데이터신호 감지 장치가 필요
  - CO에서는 DLS데이터 흐름을 다중화 기기(digital subscriber line access multiplexer, DSLAM)를 통해 ATM으로 전송
  - 우리나라에서는 ADSL이라는 이름으로 90년대 후반에 보급
    - ADSL(Asymmetric Digital Subscriber Line) – 다운속도가 업로드 속도보다 빠름.
    - 초고속 인터넷 보급의 주역 – 정액제, 2Mbps

DSL Configuration



VDSL

- VDSL (very-high-data-rate DSL)
  - 업로드 속도 16 Mbps.
  - 다운로드 속도 52 Mbps.
  - 가입자와 CO사이의 거리가 1Km이하여야 함.
    - 아파트 같은 경우 중계기 설치
  - VSDL2에서는 100Mbps 서비스

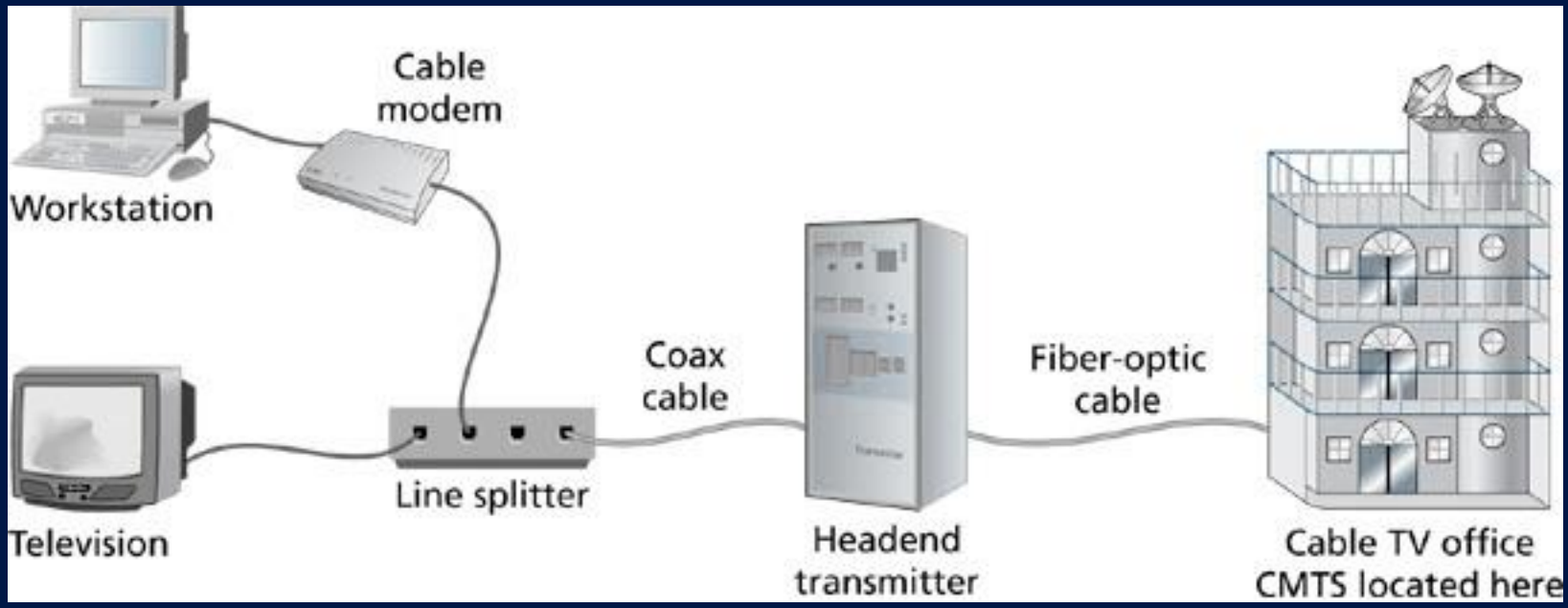
VDSL

http://www.hanuribiz.com/news/articleView.html?idxno=50931

케이블 회선

- 케이블 회선
  - 케이블 TV의 회선을 이용하는 방식
  - 업로드 3 Mbps, 다운로드 27-56 Mbps
  - 가입자는 전용 케이블모뎀을 사용해 TV신호와 분리해서 사용
  - 케이블 TV회사에서는 중간에 (hybrid fiber coax, HFC)기기를 두어 중앙 기지국의 (cable modem termination system, CMTS)과 광케이블로 연결
  - 현재는 케이블 TV사업자가 망하고, 통신사가 IPTV를 통해 케이블 TV를 서비스 하고 있다.

케이블 회선

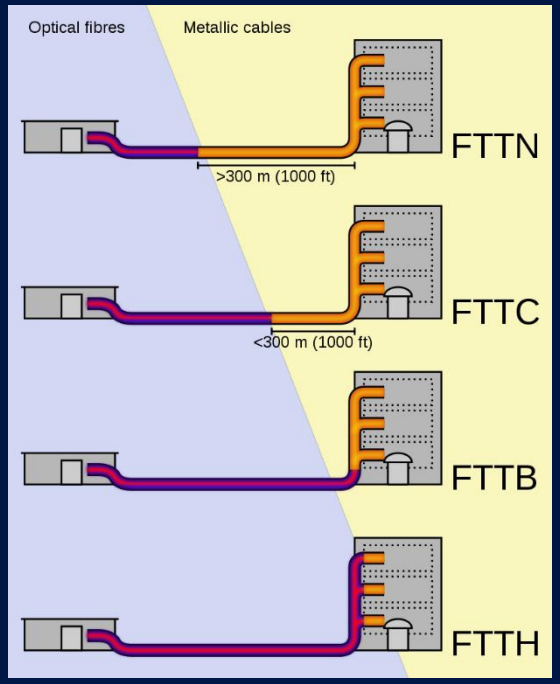

Cable
modem
Workstation
Coax
cable
Fiber-optic
cable
Line splitter
Television
Headend
transmitter
Cable TV office
CMTS located here

광케이블

- FTTH(Fiber to the Home)
  - 수동광가입자망(passive optical network, PON)을 통해 CO에서 집까지 연결
    - 수동이 아니면 별도 전원을 필요로 하는 기계가 필요
  - 광 분배기를 통해 여러명의 가입자를 연결.
  - 1Gbps이상의 속도 제공 가능
  - FTTP(Fiber to the Premises)의 최종진화형

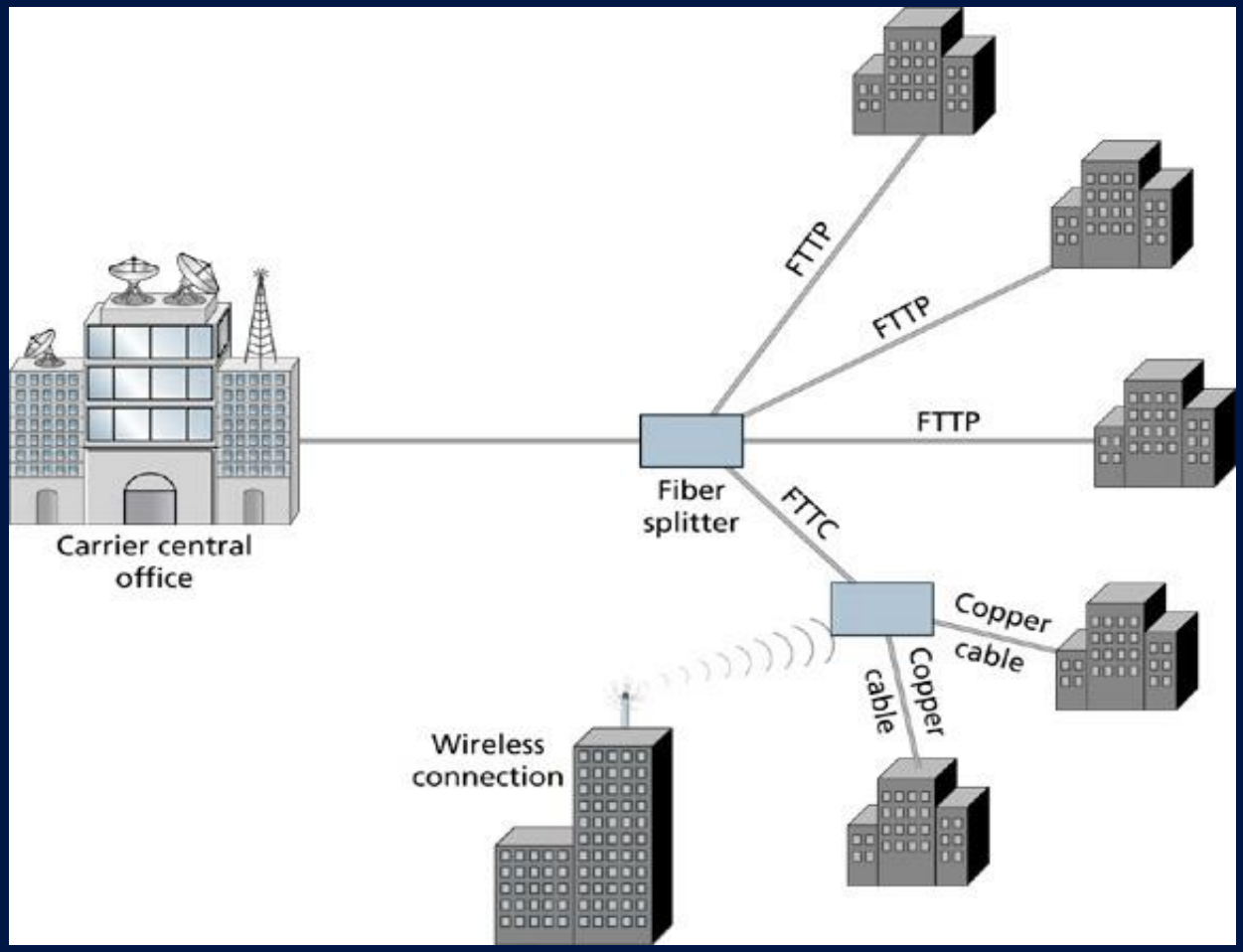
광케이블

- FTTH(Fiber to the Home)
  - FTTN(Node), FTTC(Curb, Cabinet, Closet), FTTB(Building), FTTD(Desktop PC)같은 것들이 있다.



https://en.wikipedia.org/wiki/Fiber_to_the_x

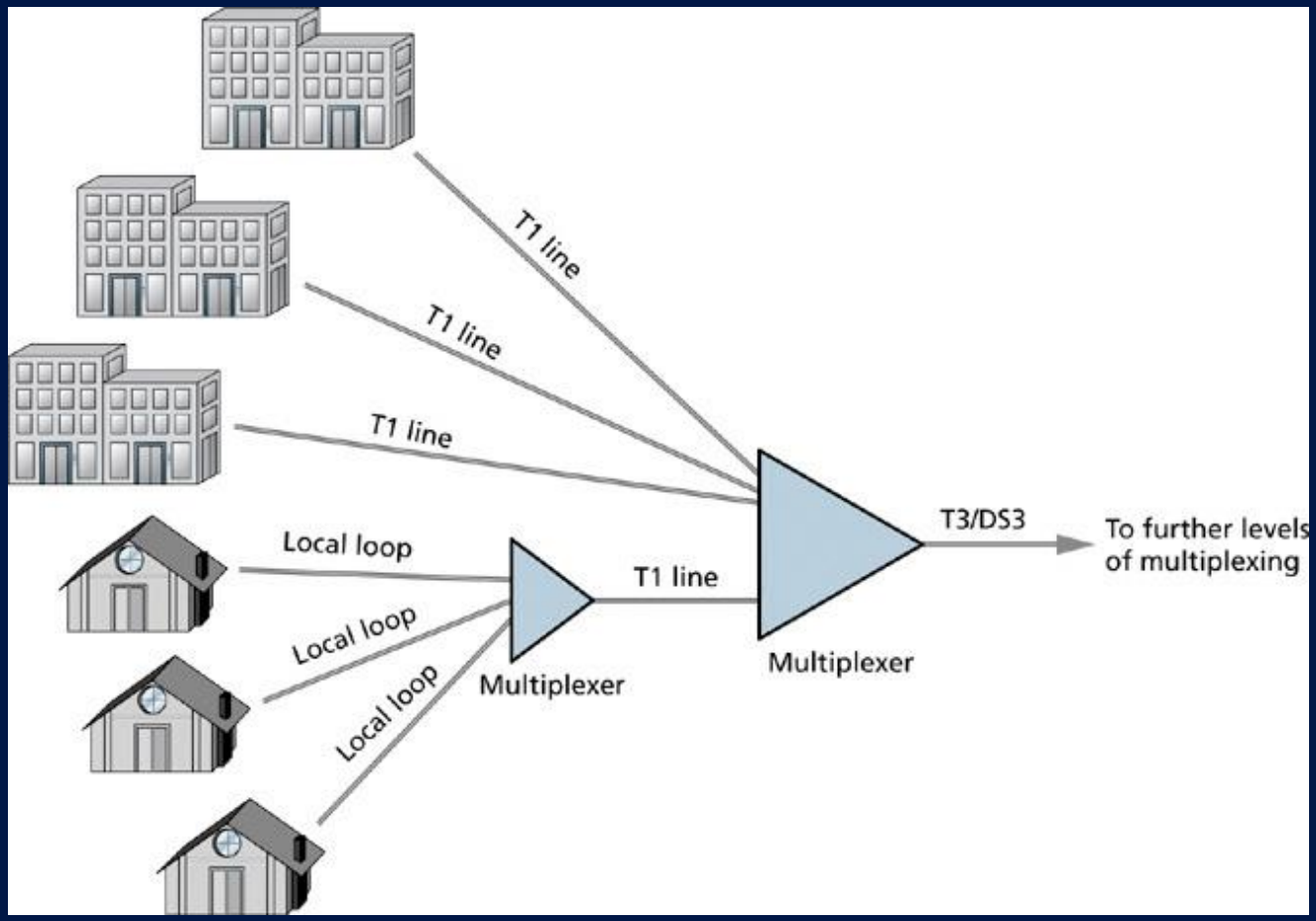
Passive Optical Network



회선 공유

- 다중화(Multiplexing)
  - 하나의 회선에 여러 개의 신호를 보내는 방법 – 하나의 회선을 여러 가입자가 공유
  - 여러 개의 신호가 하나의 신호로 합성되어서 전달
  - 다중화는 통신사업자의 설비를 효율적으로 사용할 수 있게 해준다.
    - 속도를 낮추지 않고 가입자의 사용료를 낮춰주는 효과가 있다.

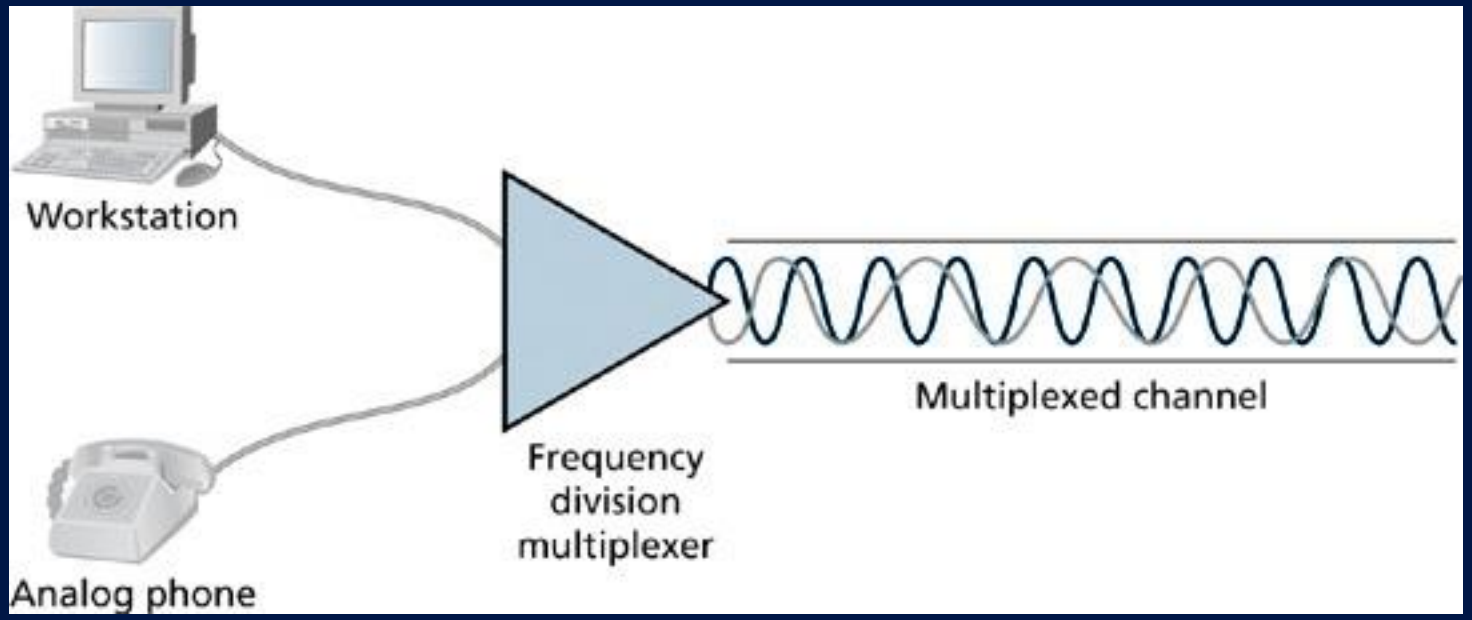
다중화 (Multiplexing)



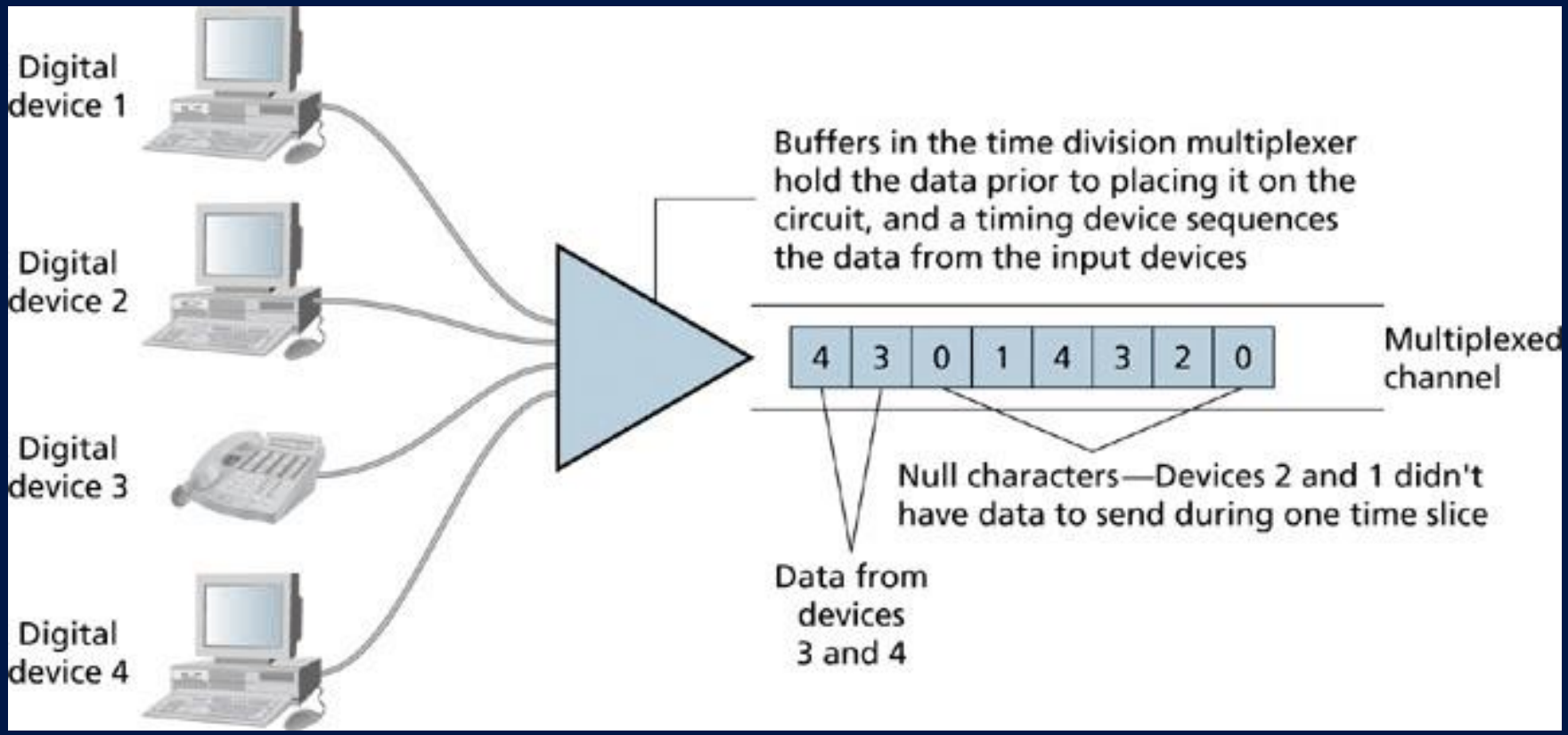
회선 공유

- 다중화 방식
  - Frequency division multiplexing (FDM)
  - Time-division multiplexing (TDM)
  - Statistical time-division multiplexing (STDM)
  - Wavelength division multiplexing (WDM)
  - Dense wavelength division multiplexing (DWDM)
  - Inverse multiplexing (IMUX)
  - CDM (Code division multiplexing)

Frequency Division Multiplexing



Time-Division Multiplexing



Code-Division Multiplexing

- 기존 데이터에 마킹을 함
  - Data 1 : (0, 1, 0, 1) -> (-1, 1, -1, 1) // 0을 -1로 노말라이즈
  - Marking 1 : (1, -1)
  - Signal 1 : (-1, 1, 1, -1, -1, 1, 1, -1) // Data 1 xor - Marking 1
  - Data 2 : (0, 0, 1, 1) -> (-1, -1, 1, 1)
  - Marking 2 : (1, 1)
  - Signal 2 : (-1, -1, -1, -1, 1, 1, 1, 1)
  - 전송 시그날 = (-2, 0, 0, -2, 0, 2, 2, 0) // Signal1 + Signal 2
- 전송시그날 * Marking 1
  - (-2, 0, 0, -2, 0, 2, 2, 0) * (1, -1)
  - -2, 2, -2, 2 => -1, 1, -1, 1 => Data 1

중간 고사 문제 풀이

숙제 4 해설

숙제 5

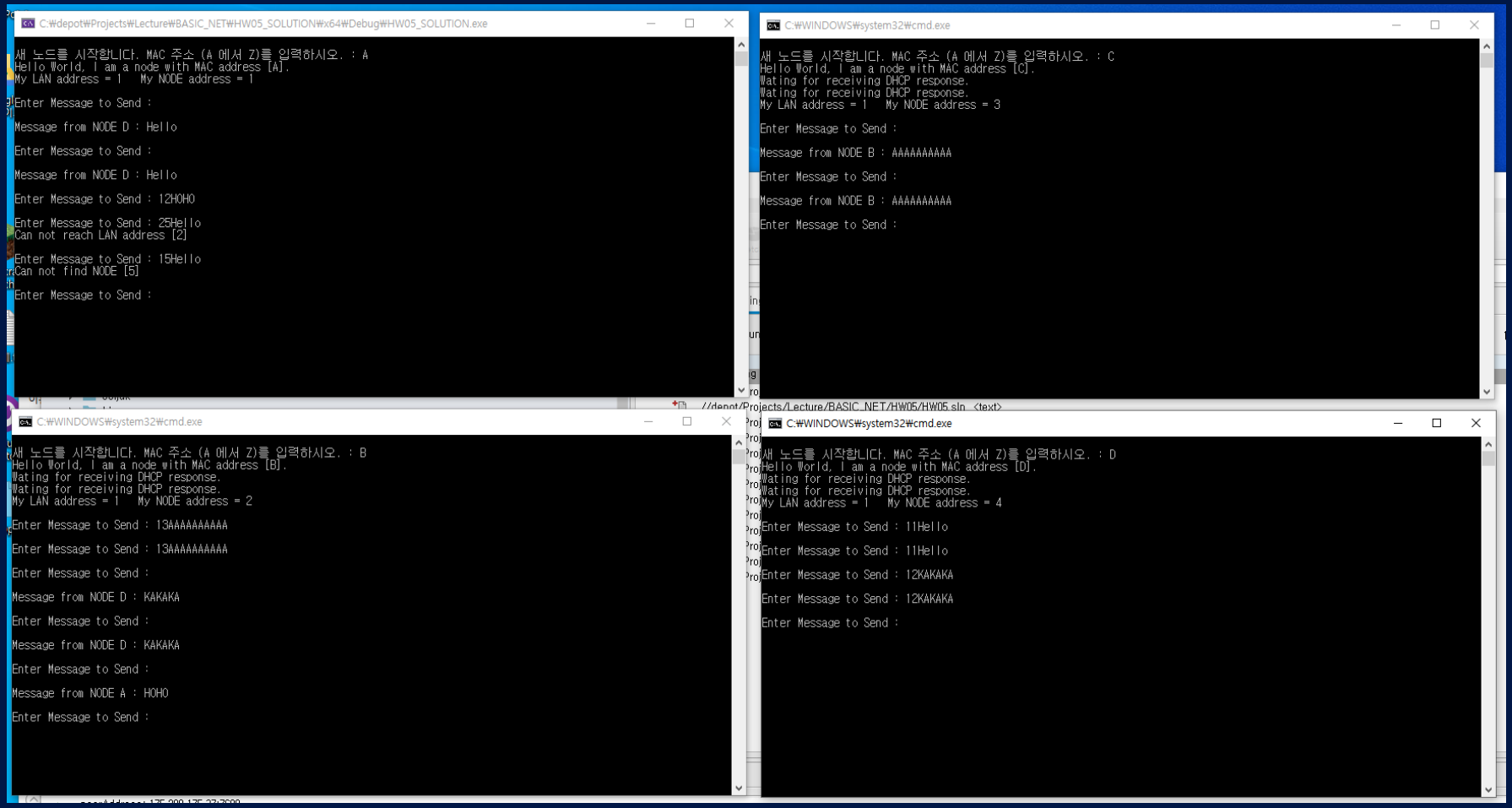
- DHCP와 Network layer 구현
- N 개의 노드가 존재
  - LAN 주소가 1인 LAN에 연결되어 있음
  - NODE 마다 MAC 주소가 있음
    - MAC 주소는 A 부터 Z 까지
    - 사용자가 선택
- NODE 주소
  - NODE 주소는 1부터 100까지의 숫자
  - 각 NODE는 NODE주소를 DHCP 프로토콜로 할당 받음
  - 노드의 네트워크 주소는 LAN주소 + NODE 주소임
    - 예) 13 => LAN주소 1 + NODE주소 3

숙제 5

- 목표
  - LAN주소와 NODE주소로 NODE를 선택해서 메시지 보내기
    - LAN주소가 불일치 할 경우 에러 메세지
      - “LAN [x]가 연결되어 있지 않습니다.”
    - NODE주소에 해당하는 NODE가 없을 경우 에러메세지
      - “NODE [x]가 존재하지 않습니다.”
  - 노드들이 정보를 주고 받아 가면서 중복 없이 노드 주소를 세팅한다.
    - 같은 MAC 주소를 갖는 NODE가 재 연결 했을 경우 같은 NODE주소를 갖도록 한다.
    - ‘A’ 노드가 주소를 관리하도록 한다.
      - ‘A’노드가 재 접속하는 경우의 주소관리는 정의하지 않는다. (다음 과제에서 구현)

숙제 5

- 실행 예제


C:₩depot₩Projects₩Lecture₩BASIC_NET₩HW05_SOLUTION₩x64₩Debug₩HW05_SOLUTION.exe
새 노드를 시작합니다. MAC 주소 (A 에서 Z)를 입력하시오. : A
Hello World, I am a node with MAC address [A].
My LAN address = 1   My NODE address = 1
Enter Message to Send :
Message from NODE D : Hello
Enter Message to Send :
Message from NODE D : Hello
Enter Message to Send : 12HOHO
Enter Message to Send : 25Hello
Can not reach LAN address [2]
Enter Message to Send : 15Hello
Can not find NODE [5]
Enter Message to Send :
C:₩WINDOWS₩system32₩cmd.exe
새 노드를 시작합니다. MAC 주소 (A 에서 Z)를 입력하시오. : C
Hello World, I am a node with MAC address [C].
Wating for receiving DHCP response.
Wating for receiving DHCP response.
My LAN address = 1   My NODE address = 3
Enter Message to Send :
Message from NODE B : AAAAAAAAAA
Enter Message to Send :
Message from NODE B : AAAAAAAAAA
Enter Message to Send :
C:₩WINDOWS₩system32₩cmd.exe
새 노드를 시작합니다. MAC 주소 (A 에서 Z)를 입력하시오. : B
Hello World, I am a node with MAC address [B].
Wating for receiving DHCP response.
Wating for receiving DHCP response.
My LAN address = 1   My NODE address = 2
Enter Message to Send : 13AAAAAAAAAA
Enter Message to Send : 13AAAAAAAAAA
Enter Message to Send :
Message from NODE D : KAKAKA
Enter Message to Send :
Message from NODE D : KAKAKA
Enter Message to Send :
Message from NODE A : HOHO
Enter Message to Send :
C:₩WINDOWS₩system32₩cmd.exe
새 노드를 시작합니다. MAC 주소 (A 에서 Z)를 입력하시오. : D
Hello World, I am a node with MAC address [D].
Wating for receiving DHCP response.
Wating for receiving DHCP response.
My LAN address = 1   My NODE address = 4
Enter Message to Send : 11Hello
Enter Message to Send : 11Hello
Enter Message to Send : 12KAKAKA
Enter Message to Send : 12KAKAKA
Enter Message to Send :

숙제 5

- 제한 사항
  - 첨부한 샘플 프로그램의 node.cpp를 수정하여 구현하라.
    - do_node()함수와 interrupt_from_link()함수를 수정하여 구현하라.
  - do_node()함수로 사용자의 입력을 수행하며
  - interrupt_from_link()함수로 LINK layer에서 전달된 frame데이터를 처리한다.
  - 이전 과제와 마찬가지로 표준라이브러리를 사용하거나 새로운 함수나 자료구조를 만드는 것은 허용되나. g_nic이외의 다른 방법으로 통신을 구현하는 것은 허용되지 않는다.

숙제 5

- 샘플 프로그램
  - NIC객체인 g_nic를 통해 frame 송수신
  - NIC 클래스의 메소드
    - SendFrame : Frame데이터를 전송매체에 전송
    - RecvFrame : 전송매체에서 Frame데이터를 하나 읽어 온다.
      - 읽어올 Frame이 없는 경우에는 frame_size에 0을 넣어서 리턴한다.
    - GetMACaddr : NIC에 할당된 MAC주소를 읽어 온다.
      - ‘A’ 부터 ‘Z’중 하나

숙제 5

- 힌트 #1
  - Frame 데이터를 구조화 한다.
  - 메시지의 Type을 정한다.

```
struct ARP_REQUEST {
   char from_mac; // 0
   char to_mac; // 0
   char packet_type; // 75 = ARP request
   int  node_address;
};

struct ARP_RESPONSE {
   char from_mac;
   char to_mac;
   char packet_type; // 76 = ARP response
   int  node_address;
   int  mac_address;
;

struct DHCP_REQUEST {
   char from_mac;
   char to_mac; // should be 0
   char packet_type; // 77 = DHCP request
};
```

```
struct DHCP_RESPONSE {
   char from_mac;
   char to_mac;
   char packet_type; // 78 = DHCP response
   int  lan_address;
   int  node_address;
};

struct SEND_MESS {
   char from_mac;
   char to_mac;
   char packet_type; // 79 = MESS packet
   int  size;
   char data[MAX_DATA_SIZE];
};
```

숙제 5

- 힌트 #2
  - 주소 관리자를 통해 주소 테이블을 관리한다.

```
#include <mutex>

class ADDR_MANAGER {
private:
   mutex table_lock;
   map <char, char> mac_to_node_table;
   map <char, char> node_to_mac_table;
public:
   int get_mac_addr(int node_addr)
   {
      int mac_addr = -1;
      table_lock.lock();
      if (node_to_mac_table.count(node_addr) != 0)
          mac_addr = node_to_mac_table[node_addr];
      table_lock.unlock();
      return mac_addr;
   }
   void set_addr(int mac_address, int node_address)
   {
      table_lock.lock();
      node_to_mac_table[node_address] = mac_address;
      mac_to_node_table[mac_address] = node_address;
      table_lock.unlock();
   }
```

```
int assign_node_addr(int mac_address)
   {
      int node_addr = -1;
      table_lock.lock();
      if (0 == mac_to_node_table.count(mac_address))
         for (int i = 2; i < 50; ++i)
            if (0 == node_to_mac_table.count(i)) {
               node_to_mac_table[i] = mac_address;
               mac_to_node_table[mac_address] = i;
               node_addr = i;
               break;
            }
      table_lock.unlock();
      return node_addr;
   }
};

ADDR_MANAGER g_addr_manager;
```

숙제 5

- 힌트 #3
  - do_node()에서는 사용자 입력 메시지를 전송하는 것만 실행
  - interrupt_from_link()에서 frame데이터 처리

```
void interrupt_from_link(NIC g_nic, int recv_size, char* frame)
{
   const char mac_addr = g_nic.GetMACaddr();
   switch (frame[2]) {
      case 75 : // ARP request {
          ARP_REQUEST* p = reinterpret_cast<ARP_REQUEST*>(frame);
          …
             g_nic.SendFrame(sizeof(rp), &rp);
          }
      return;
      }
      case 76: // ARP response {
         ARP_RESPONSE* p = reinterpret_cast<ARP_RESPONSE*>(frame);
         } return;
      case 77 : // DHCP request{
         if ('A' != mac_addr) return;
         DHCP_REQUEST* p = reinterpret_cast<DHCP_REQUEST*>(frame);
         int new_node_addr = g_addr_manager.assign_node_addr(p->from_mac);
         …
         rp.packet_type = 78;
         g_nic.SendFrame(sizeof(rp), &rp);
        } return;
```

```
      case 78: { // DHCP response
         DHCP_RESPONSE* p = reinterpret_cast<DHCP_RESPONSE*>(frame);
         if (mac_addr != p->to_mac) return;
         … }    return;
      case 79: {
         … } break;
   }
   cout << "WnEnter Message to Send : ";
}
```