chapter 3

LAN – 토폴로지와 구조

목표

- LAN의 토폴로지를 살펴본다. 버스, 스타, 링, 무선 형태가 있다.
- 물리적, 논리적 토폴로지의 차이를 알아본다.
- LAN 구조를 정의한다.
- 이더넷 LAN구조를 정의하고, 이더넷 표준에 대해 알아본다.
- 이더넷 연결방식에 대해 알아본다.

목표

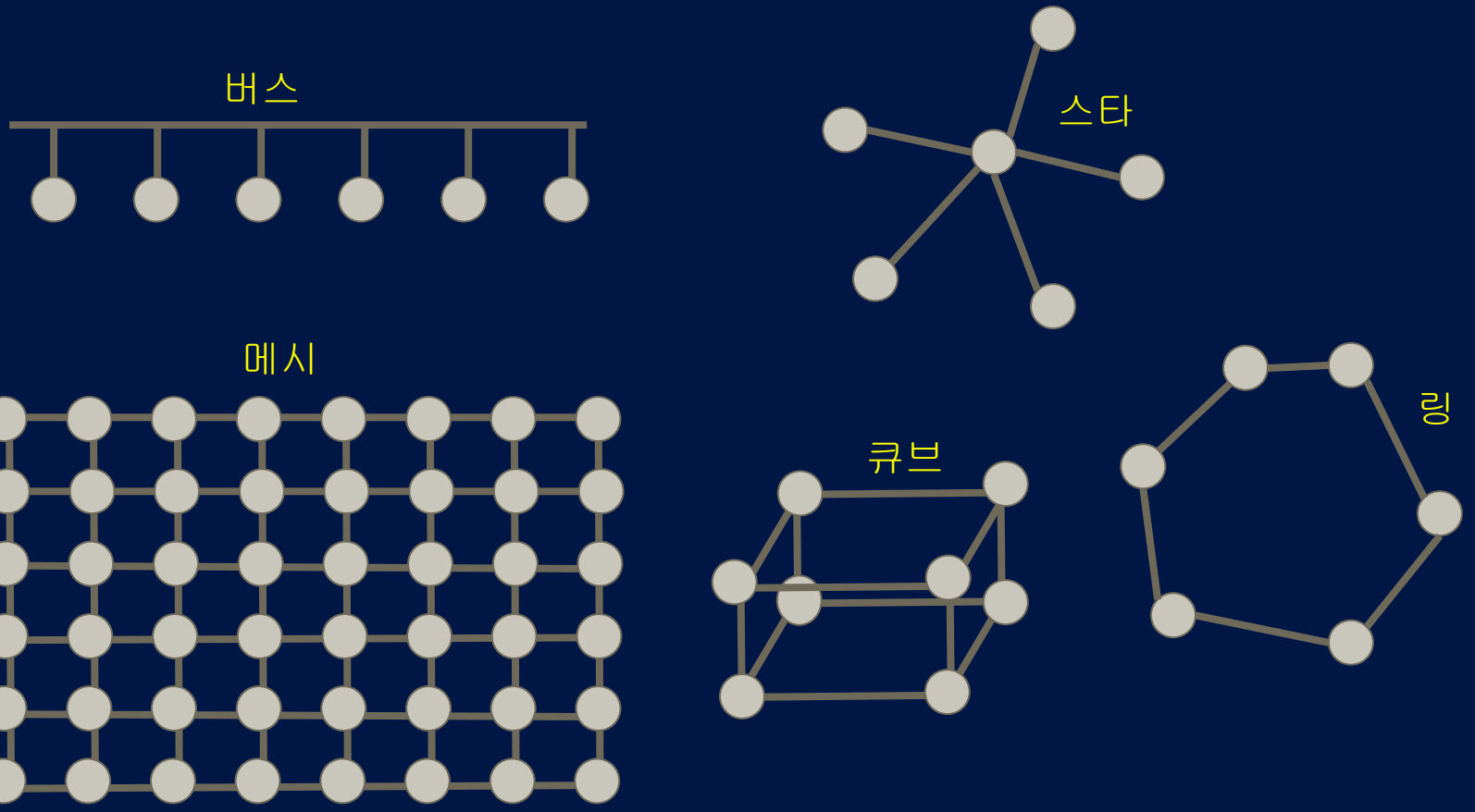
- 무선랜의 구조를 살펴본다..
- 무선랜과 무선 PAN기술과 역사를 살펴본다.
- 802.11 표준과 Bluetooth에 대해 알아본다.
- 무선랜 구조에 대한 기술적, 사업적 측면에 대해 알아본다.
- FDDI와 Infiniband에 대해 알아본다.

LAN 토폴로지

- LAN topology는 각 노드들의 연결된 모양을 이야기한다.
- 버스, 스타, 링, 무선 방식이 많이 사용된다.
- 논리적, 물리적으로 어떻게 노드들이 연결되었는 가에 따라 LAN의 성격이 결정된다.

LAN 토폴로지

- 여러가지 토폴로지



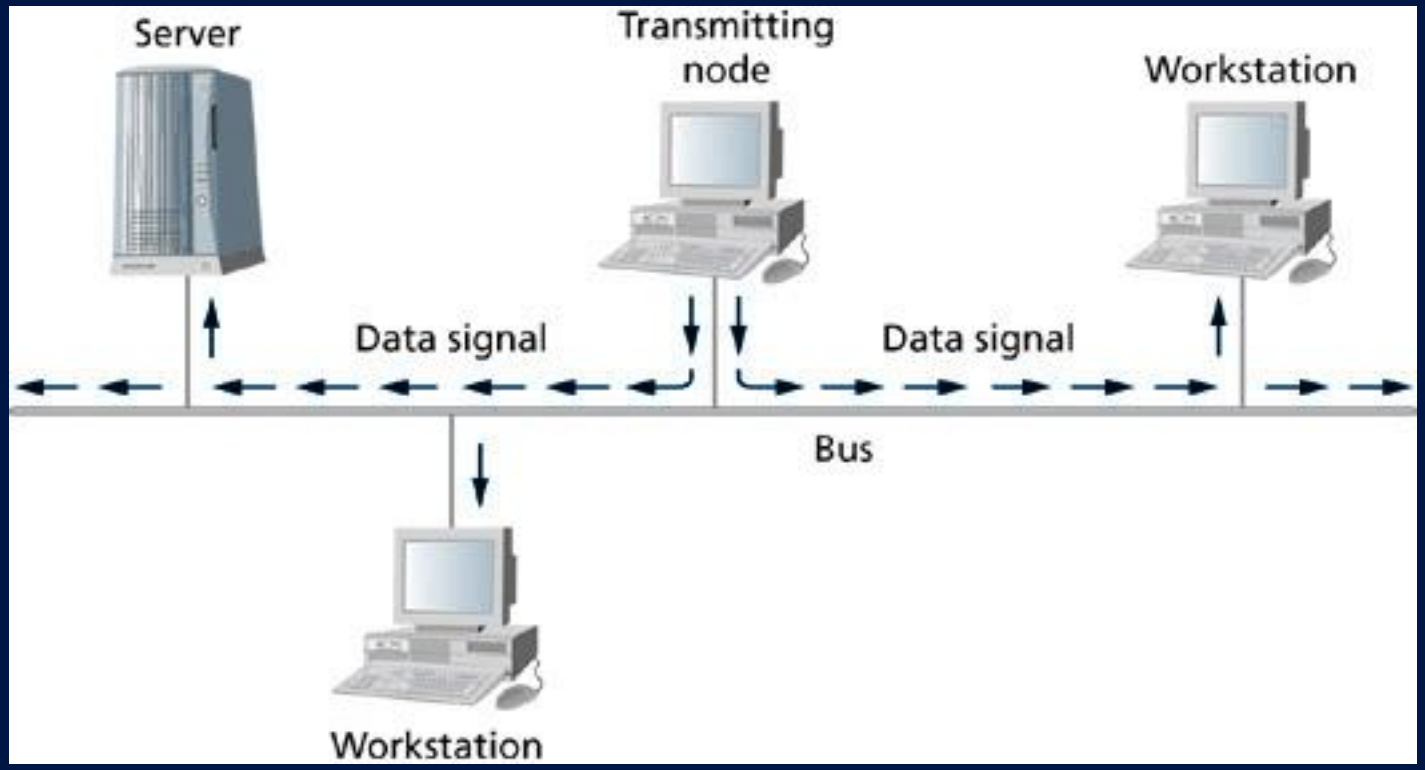
LAN 토폴로지

- 논리적 토폴로지 : 데이터가 어느 경로로 전달되는지를 의미한다.
- 물리적 토폴로지 : 전선 연결 같은 실제 노드들의 물리적 연결 방식을 의미한다.

LAN 토폴로지 – 버스

- 모든 노드가 하나의 통신 매체를 공유하는 형태이다.
- 보통 동축케이블을 사용한다.
- 물리적, 논리적 토폴로지가 동일하다.

간단한 버스 토폴로지


Server
Transmitting node
Workstation
Data signal
Data signal
Bus
Workstation

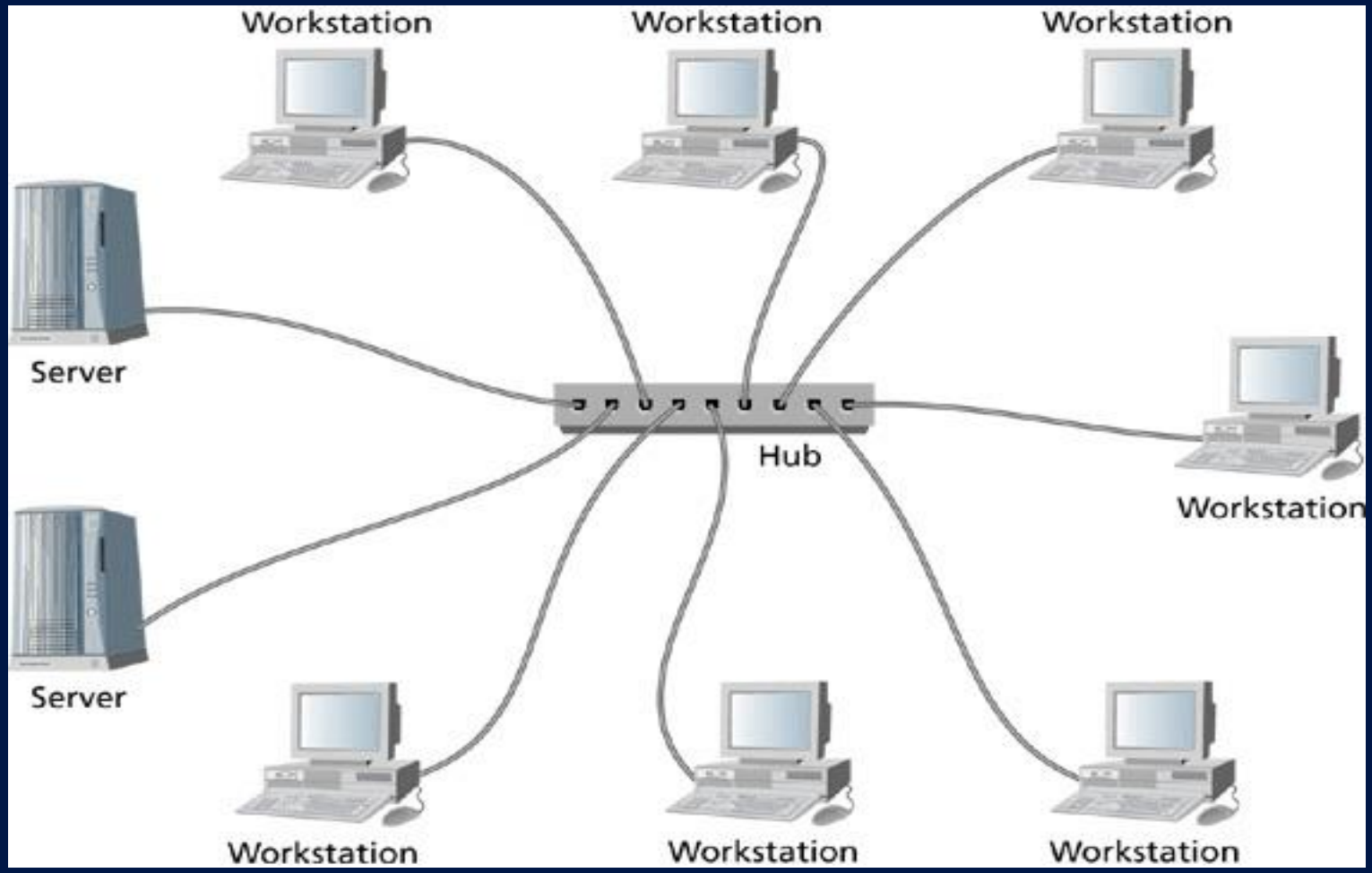
LAN 토폴로지 – 버스

- 장점
  - 쉽고 저렴하게 구현할 수 있다.
- 단점
  - 케이블이 손상되면 전체 네트워크가 불통이된다.
  - 케이블상의 손상된 위치를 찾는 것이 어렵다.
  - 잘못된 노드 하나가 전체 네트워크에 문제를 일으키기 쉽다.

LAN 토폴로지 – 스타

- 하나의 중심 노드가 있어서 다른 모든 노드가 중심 노드에 1:1로 연결된다.
- 예) 공유기에 여러 컴퓨터가 UTP케이블로 연결되어 있는 형태

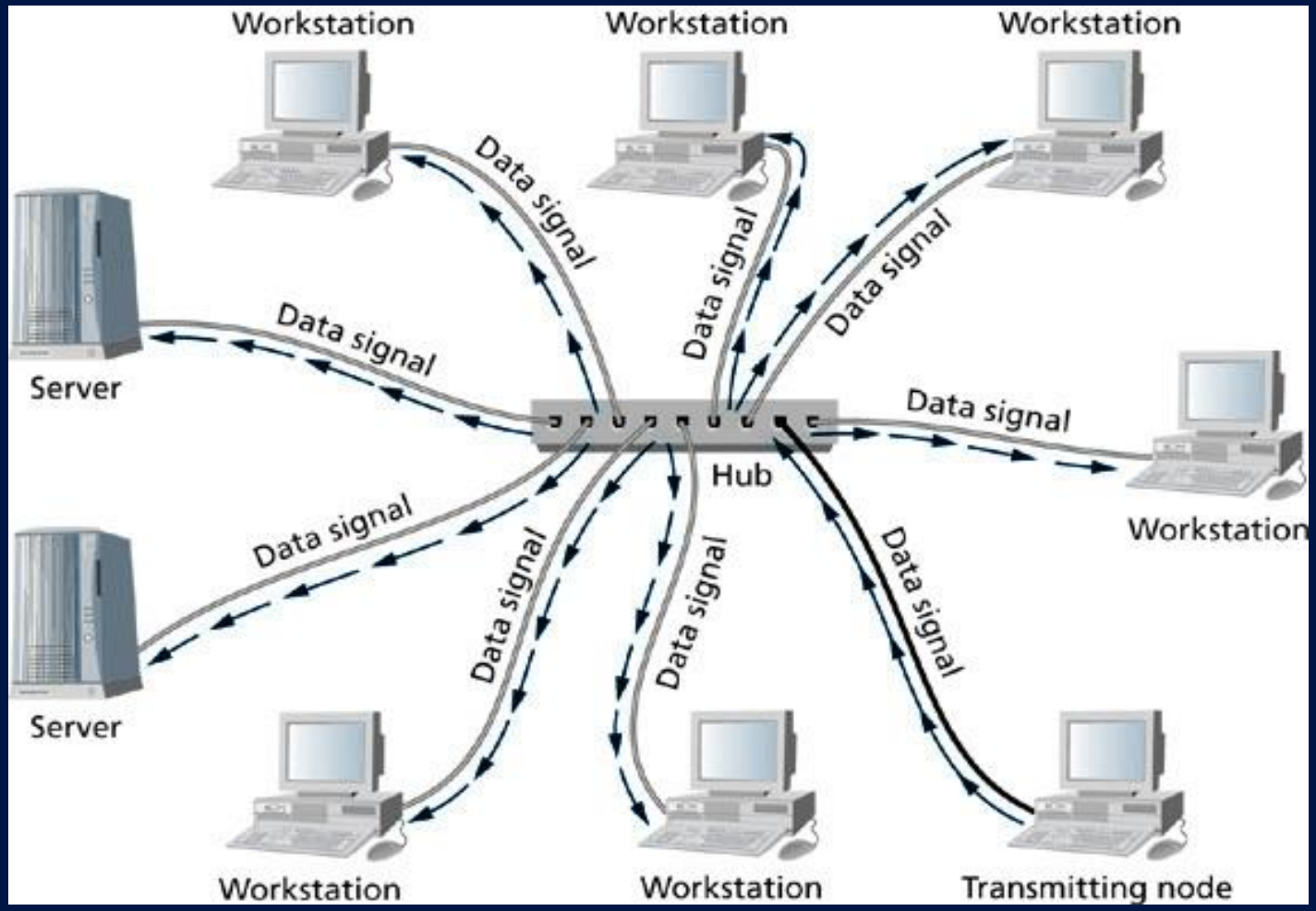
스타 토폴로지


Workstation
Workstation
Workstation
Server
Hub
Workstation
Server
Workstation
Workstation
Workstation

LAN 토폴로지 – 스타

- 물리적 스타/논리적 버스
  - 우리가 사용하는 공유기가 이런 형태임
  - UTP/STP로 허브와 다른 노드들을 1대1 연결
    - 허브 : 중심 노드를 뜻함.
  - 허브에 연결된 즉시 허브에 연결된 모든 노드들의 신호를 받음, 즉 논리적으로는 버스

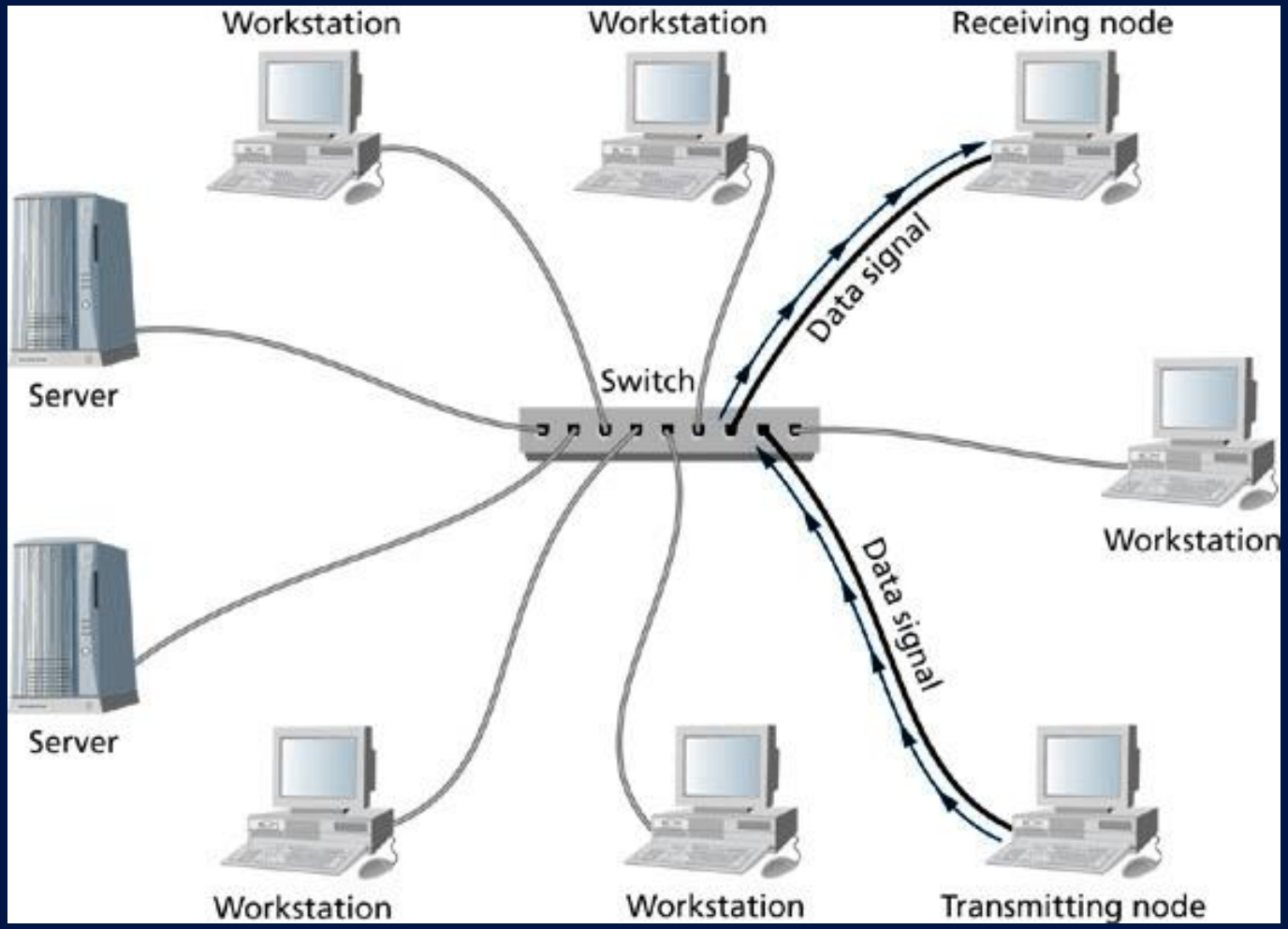
물리적 스타/논리적 버스


Workstation
Workstation
Workstation
Data signal
Data signal
Data signal
Data signal
Server
Data signal
Hub
Data signal
Workstation
Data signal
Server
Data signal
Data signal
Data signal
Workstation
Workstation
Transmitting node

LAN 토폴로지 – 스타

- 물리적 스타/논리적 스타
  - 여기서는 중심노드를 스위치라 부름
  - 스위치에 다른 모든 컴퓨터들이 UTP/STP로 스타 모양으로 연결
  - 데이터는 목표 노드로만 전달됨. 따라서 논리적으로도 스타연결 구조임.

물리적 스타/논리적 스타


Workstation
Workstation
Receiving node
Server
Switch
Data signal
Workstation
Server
Data signal
Workstation
Workstation
Transmitting node

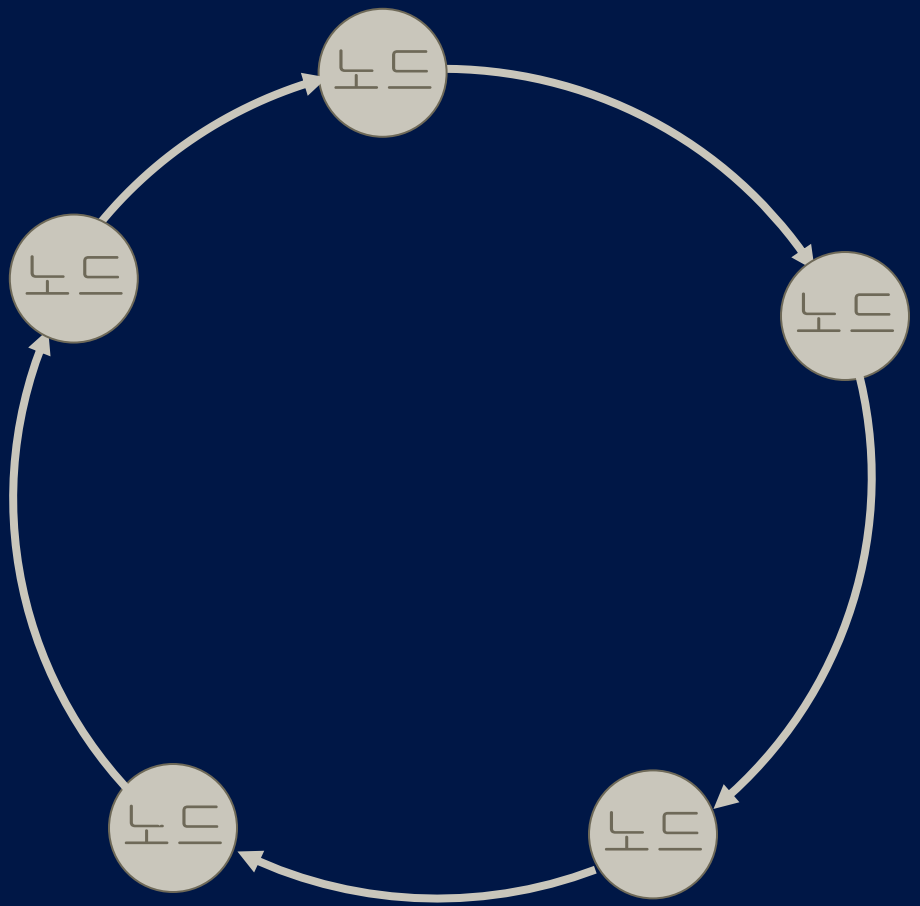
LAN 토폴로지 – 스타

- 장점
  - 하나의 케이블이나 노드가 오동작을 해도 나머지는 잘 동작함.
- 단점
  - 비싼 중심 노드가 필요하다
    - 연결 가능 노드 개수가 많을 수록 비쌈.
    - 현재는 가격이 많이 떨어짐
  - 중심 노드가 고장나면 전체가 동작불능이됨
    - 취약점(SPOF:Single Point Of Failure) 존재

LAN 토폴로지 – 링

- 링 토폴로지topology는 모든 디바이스가 하나의 원을 그리며 연결되고, 데이터가 한방향으로만 전달되는 형태이다.

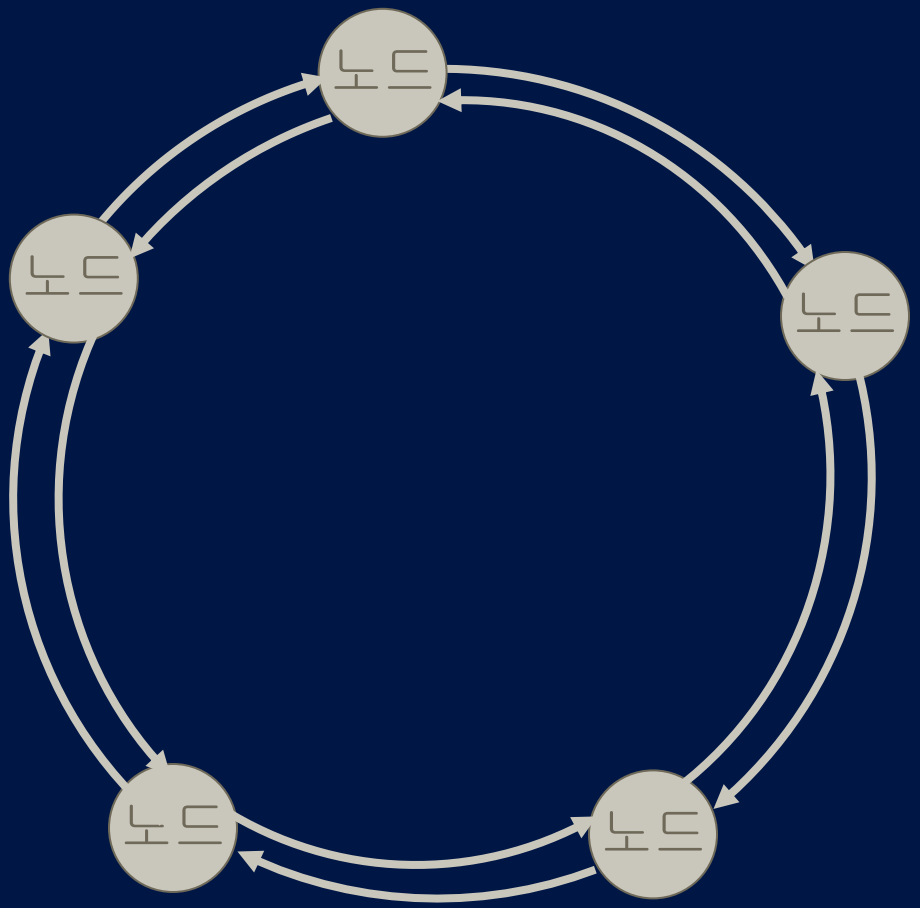
링 토폴로지



LAN 토폴로지 – 링

- 장점
  - 비싼 중심 노드가 필요 없다.
  - 속도가 빠르다 (1:1 연결, 단방향 통신)
- 단점
  - 하나의 케이블이나 노드가 고장 나도 전체가 작동 불능
  - 원하는 노드까지의 거리가 멀면 전달 지연이 일어날 수 있다.
  - 다른 노드의 데이터 때문에 나의 LAN사용량이 제한될 수 있다.
- 단점을 줄이기 위해 전송 방향이 서로 반대인 듀얼링을 사용하기도 한다.

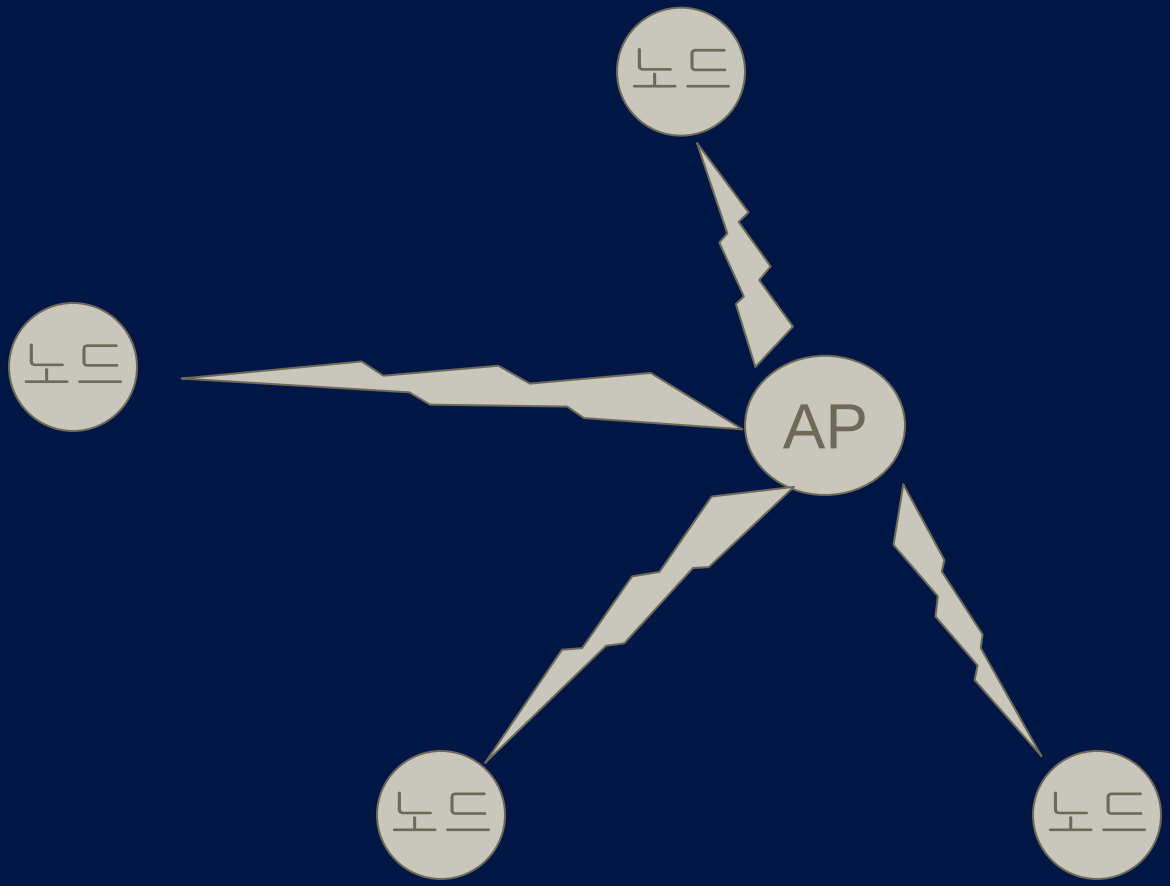
듀얼 링 토폴로지



LAN 토폴로지 – 무선

- 무선 토폴로지 Wireless topologies는 전파를 통신 매체로 이용한다.
- 널리 쓰이는 형태는 AP(Access Point)라는 중심노드를 사용한 스타 형태이다.
- WAN(Wireless Area Network)라는 표준 단어가 있다.
  - WAN(Wide Are Network)와 헷갈리기 쉽다. 그래서 안쓴다.

Wireless Topology



LAN 토폴로지 – 무선

- 많이 사용되는 무선 토폴로지는 물리적 토폴로지는 스타 형태이고 논리적 토폴로지는 버스이다.
  - 무선 디바이스끼리 서로 직접 통신을 하지 않으므로 물리적 버스 토폴로지는 아니다.

LAN 토폴로지 – 무선

- 장점
  - 설치가 간편 : 따로 HW 설치할 것이 없다.
  - 전선 연결이 필요 없고, 구멍을 뚫지 않아도 된다.
  - 디바이스의 위치에 구애 받지 않는다.
- 단점
  - AP와 디바이스사이의 최대 거리가 짧다. 10m 정도
    - 장애물에 민감하다.
  - 도청의 위험이 크다. (도청 감지가 힘들다.)
  - 유선에 비해 느리다.
  - 전파 간섭에 취약하다.

LAN 구현

- LAN 구현은 네트워크 매체의 종류와 연결 방법 매체를 통해 오가는 데이터의 프레임 구조를 정의하는 것이다.

LAN 구현 – 이더넷

- 1970년대에 시작.
- Alohanet이라는 네트워크를 기반으로 함.
- Bob Metcalfe가 개발.
- 첫번째 표준은 DIX라 불렸음.
- 첫번째 IEEE 이더넷 표준은 IEEE 802.3
- 현재 LAN으로 널리 사용 중
- 신뢰성 있고, 구현이 쉽고, 저렴하다.

LAN 구현 – 이더넷

- 처음은 thick 케이블로 시작해서 (10Base5) thin 케이블도 개발 (10Base2).
- UTP를 사용한 10Mbps의 10Base-T가 개발되어 물리적으로 버스 형식에서 스타형식으로 변경
- 100Base-T 는 100 Mbps로 Ethernet 동작
- 다른 이더넷 표준은 IEEE 802.3에 정의되어 있음.

IEEE 802.3 이더넷 표준

표 4-2 주요 이더넷 규격

규격 이름	통신 속도	케이블	케이블 최대 길이	표준화 연도
10BASE5	10Mbps	동축케이블	500m	1982년
10BASE2	10Mbps	동축케이블	185m	1988년
10BASE-T	10Mbps	UTP케이블(Cat3이상)	100m	1990년
100BASE-TX	100Mbps	UTP케이블(Cat5이상)	100m	1995년
1000BASE-T	1000Mbps	UTP케이블(Cat5이상)	100m	1999년
10GBASE-T	10Gbps	UTP케이블(Cat6a이상)	100m	2006년

LAN 구현 – 이더넷 접속 방식

- 이더넷은 통신 방법으로 CSMA/CD(반송파검출 다중접속 / 충돌검사)를 사용한다.
- CS(Carrier Sense) 현재 공유 중인 전송 매체가 사용중인가를 감지하는 기능
- MA(Multiple Access) 전송 매체를 여러 노드가 공유하며, 하나의 특별한 노드가 관리하지 않고 모두 동등한 자격으로 사용하는 방식

LAN 구현 – 이더넷 접속 방식

- CD(Collision Detection) 전송 중 충돌이 감지되면 즉시 전송을 중지하고 랜덤한 시간동안 기다렸다가 다시 전송을 시도하는 방식.
- CSMA/CD
  - 전송 매체가 놀고있는 것을 확인한 후 전송 시작.
    - 그래도 충돌하면 포기하고 대기 후 재전송
  - 다른 노드가 사용 중이면 끝날 때 까지 기다림.

LAN 구현 – 이더넷 접속 방식

- CSMA/CD의 장점
  - 알고리즘이 간단하고, 다른 방법에 비해 효율적이다.
- CSMA/CD의 단점
  - 매체에 연결되는 노드가 많아지면 충돌 확률이 높아진다.
    - 뻔한 이야기, 그렇다고 해서 대안은 없다. -_-

여러가지 유선 접속 방식

- 다중 접속 시 충돌을 해결하기 위한 방법
- Carrier Sense가 아닌 방식
  - ALOHA : 일단 보내고 상대방이 OK하면 종료, 아니면 다시 보냄
  - Slotted ALOHA : 시간대를 나누어서 정해진 시간에만 전송 시작 (예: 1분 0초 마다, 실패하면 다음 1분에, 1분 27초에는 시작하지 않음)
- CSMA/CA(Collision Avoidance) : 통신선이 사용 중이면 랜덤한 시간 동안 기다렸다가 다시 검사

이더넷 특성

- 거리 제한이 존재
  - 예) 100 Mbps 이더넷은 최대 간격 100미터 네트워크 총 길이 205미터.
- 이더넷은 산업 표준
  - 계속 속도가 업그레이드 되고 있다.
  - TCP/IP모델의 Link 계층에 해당
  - 48비트 MAC 주소로 노드들을 구분

```
C:₩Users₩nhjung>getmac /v

연결 이름       네트워크 어댑터  물리적 주소         전송 이름
=============== ============== =================== ============================================================
이더넷          Marvell AQtion  70-85-C2-33-06-00   미디어 연결 끊김
이더넷 3        Intel(R) Ethern 70-85-C2-33-05-FC   ₩Device₩Tcpip_{5E6E9D18-D858-4B3A-A077-A1A61E88E37C}
vEthernet (Defa Hyper-V Virtual 00-15-5D-7B-B7-26   ₩Device₩Tcpip_{4978690E-4D2C-4F85-BD2A-EFE9C83BBCE0}
Bluetooth 네트  Bluetooth Devic 해당 없음           하드웨어가 없음
Wi-Fi 3         Intel(R) Dual B F4-06-69-D4-8D-66   미디어 연결 끊김
이더넷 5        Intel(R) I211 G 70-85-C2-33-05-FE   미디어 연결 끊김
```

랜 구조 – 무선

- 무선 구조로는 WIFI, Bluetooth가 많이 사용된다.
  - WIFI라는 단어는 표준도 아니고 약자도 아닌 근본 없는 단어.
- IEEE 표준은 WIFI(IEEE 802.11), 블루투스(IEEE 802.15)이다.

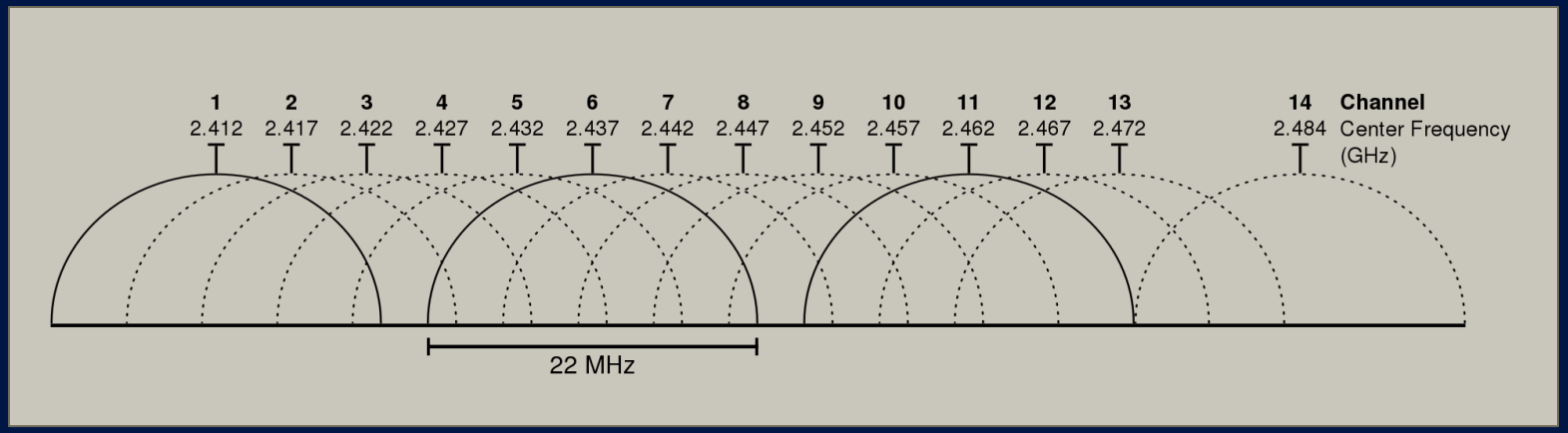
무선 IEEE 802.11 데이터 통신 표준

IEEE 802.11 network PHY standards								
Frequency range, or type	PHY	Protocol	Release date[13]	Frequency	Bandwidth	Stream data rate[14]	Approximate range[citation needed]	
				(GHz)	(MHz)	(Mbit/s)	Indoor	Outdoor
1–6 GHz	DSSS/FHSS[15]	802.11-1997	Jun 1997	2.4	22	1, 2	20 m (66 ft)	100 m (330 ft)
	HR-DSSS[15]	802.11b	Sep 1999	2.4	22	1, 2, 5.5, 11	35 m (115 ft)	140 m (460 ft)
	OFDM	802.11a	Sep 1999	5	5/10/20	6, 9, 12, 18, 24, 36, 48, 54 (for 20 MHz bandwidth, divide by 2 and 4 for 10 and 5 MHz)	35 m (115 ft)	120 m (390 ft)
		802.11j	Nov 2004	4.9/5.0			?	?
		802.11p	Jul 2010	5.9			?	1,000 m (3,300 ft)[17]
		802.11y	Nov 2008	3.7[A]			?	5,000 m (16,000 ft)
	ERP-OFDM	802.11g	Jun 2003	2.4			38 m (125 ft)	140 m (460 ft)
	HT-OFDM[18]	802.11n	Oct 2009	2.4/5	20	Up to 288.8[B]	70 m (230 ft)	250 m (820 ft)
					40	Up to 600[B]		
	VHT-OFDM[18]	802.11ac	Dec 2013	5	20	Up to 346.8[B]	35 m (115 ft)[20]	?
					40	Up to 800[B]		
					80	Up to 1733.2[B]		
					160	Up to 3466.8[B]		
	HE-OFDMA	802.11ax	Est. Feb 2021	2.4/5/6	20	Up to 1147[F]	30 m (98 ft)	120 m (390 ft) [G]
					40	Up to 2294[F]		
					80	Up to 4804[F]		
					80+80	Up to 9608[F]		
mmWave	DMG[21]	802.11ad	Dec 2012	60	2,160	Up to 6,757[22] (6.7 Gbit/s)	3.3 m (11 ft)[23]	?
		802.11aj	Apr 2018	45/60[C]	540/1,080[24]	Up to 15,000[25] (15 Gbit/s)	?	?
	EDMG[27]	802.11ay	Est. March 2021	60	8000	Up to 20,000 (20 Gbit/s)	10 m (33 ft)	100 m (328 ft)
Sub-1 GHz IoT	TVHT[29]	802.11af	Feb 2014	0.054–0.79	6–8	Up to 568.9[30]	?	?
	S1G[29]	802.11ah	Dec 2016	0.7/0.8/0.9	1–16	Up to 8.67 (@2 MHz)[31]	?	?
2.4 GHz, 5 GHz	WUR	802.11ba[E]	Est. March 2021	2.4/5	4.06	0.0625, 0.25 (62.5 kbit/s, 250 kbit/s)	?	?
Light (Li-Fi)	IR	802.11-1997	Jun 1997	?	?	1, 2	?	?
	?	802.11bb	Est. Jul 2022	60000-790000	?	?	?	?

https://en.wikipedia.org/wiki/IEEE_802.11

랜 구조 - 무선

- IEEE 802.11 는 CSMA/CA방식을 사용해서 충돌을 해결한다.
- 주파수를 여러 채널로 나누어서 복수의 연결 통로를 사용한다.
  - 하나의 AP를 여러 개의 무선 기기가 동시 사용
  - AP끼리 충돌방지를 위한 분할



랜 구조 – 무선 블루투스

- 블루투스(Bluetooth)

- 목표 : **저전력**, 근거리, 간단한 프로토콜
  - 컴퓨터가 아닌 소형 디바이스 용 (속도, 보안 희생)
- 이름의 유래 : 덴마크의 왕 하랄 1세 블로탄

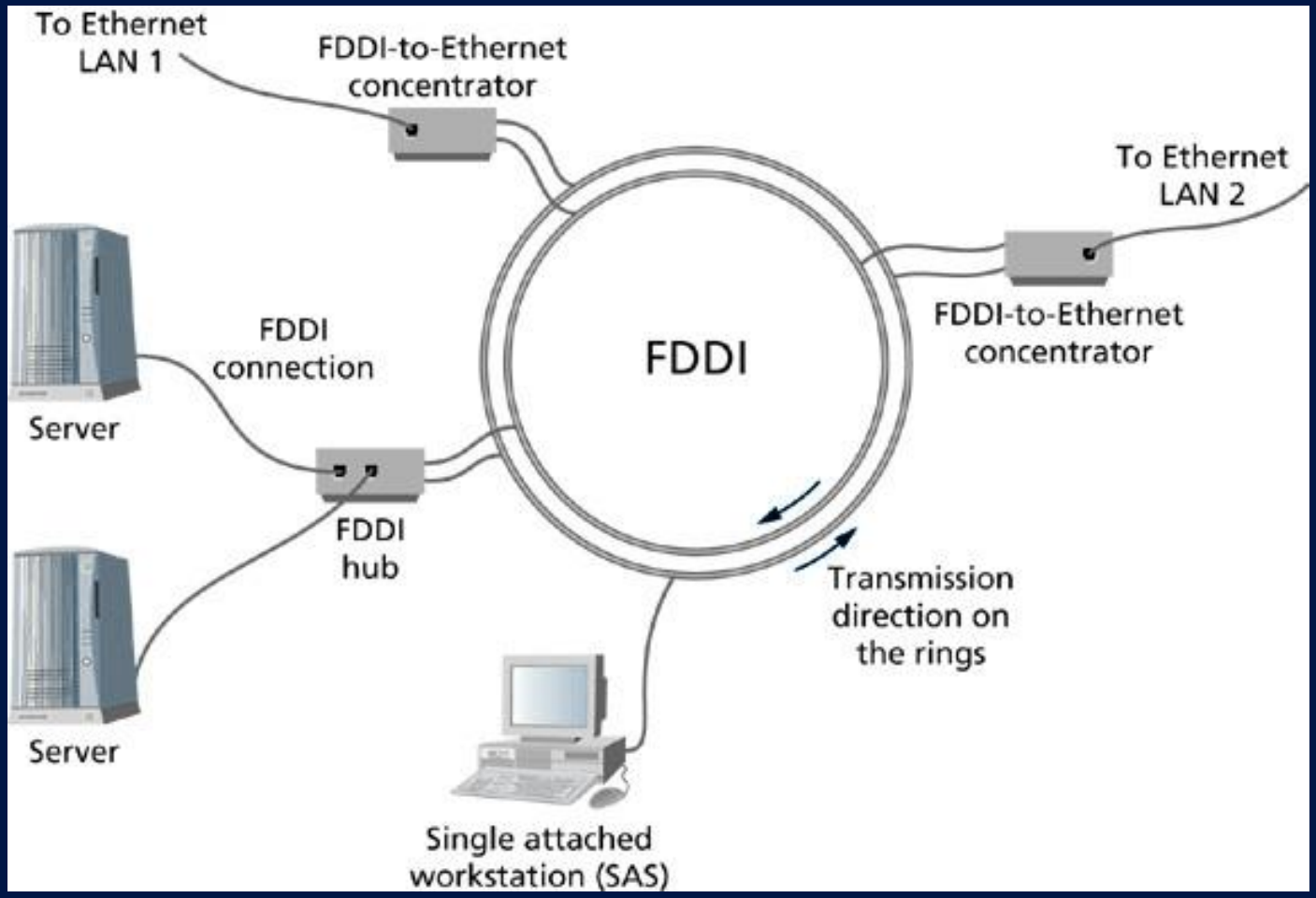
랜 구조 – 무선 블루투스

- 1994년 에릭슨에서 최초 개발, 1999년 공식 발표
- 사용 주파수 : 2.4GHz
  - WIFI와 충돌
- 1대1 통신 기반 : 페어링(Paring)이라 부름
  - AP가 없어도 동작
- 버전 1.0부터 5.2(2019년)까지 나와 있음
  - 속도가 721kbps에서 50Mbps까지 발전
    - 모든 장치가 50Mbps가 아님, 특수한 경우
  - 향상점 : 저전력 기능, 연결 편의성, 보안

랜 구조 – FDDI

- FDDI – Fiber Distributed Data Interface
- 1980년대 부터 사용된 오래된 기술.
- 다양한 통신 프로토콜을 지원
- 대학 캠퍼스 통신 수단으로 많이 사용
  - 이더넷으로 연결하기에는 너무 먼 거리
  - 건물마다 다른 LAN을 설치하는 것은 비용증가

듀얼링 구조의 FDDI


To Ethernet
LAN 1
FDDI-to-Ethernet
concentrator
To Ethernet
LAN 2
FDDI-to-Ethernet
concentrator
FDDI
FDDI
connection
Server
FDDI
hub
Server
Transmission
direction on
the rings
Single attached
workstation (SAS)

랜 구조 - FDDI

- 듀얼링 구조
- 200Mbps의 속도, 최대 100km거리 연결
- 지금도 광통신을 사용하지만 FDDI가 아니라 ITU에서 정의한 Optical Transport Network(OTN)을 사용해서 100Gbps 이더넷을 지원

랜 구조 – INFINIBAND

- 이더넷을 뛰어넘는 고속 근거리 연결방식
- 1999년 InfiniBand Trade Association(IBTA)라는 협회에서 시작.
  - IBTA회원 : 컴팩, 델, HP, IBM, 인텔, MS, Sun

랜 구조 – INFINIBAND

- 지연시간 (Latency)와 전송 속도(Gbps)

	SDR	DDR	QDR	FDR10	FDR	EDR	HDR	NDR	XDR
Adapter latency (µs)[8]	5	2.5	1.3	0.7	0.7	0.5	less?	t.b.d.	t.b.d.
Encoding (bits)	8b/10b			64b/66b				t.b.d.	t.b.d.
Signaling rate (Gbit/s)	2.5	5	10	10.3125	14.0625[6]	25.7812 5	50	100	250
for 1 link	2	4	8	10	13.64	25	50	100	250
for 4 links	8	16	32	40	54.54	100	200	400	1000
for 8 links	16	32	64	80	109.08	200	400	800	2000
for 12 links	24	48	96	120	163.64	300	600	1200	3000
Year[9]	2001, 2003	2005	2007	2011	2011	2014[10]	2018[10]	2021[10]	after 20 23?

랜 구조 – Infiniband

- 가격

Key Features

- Performance
  - 36 56Gb/s ports in a 1U switch
  - Up to 4Tb/s aggregate switching capacity
  - 430ns latency between InfiniBand and Ethernet
- Optimized design
  - 1+1 redundant & hot-swappable power
  - N+1 redundant & hot-swappable fans
  - 80 gold+ and energy start certified power supplies
  - Dual-core x86 CPU

Availability: Limited

NVIDIA MSX6710G-FS2R2 SwitchX-2 InfiniBand to Ethernet Gateway 36 QSFP+ Ports 2 Power Supplies AC x86 Dual Core Standard Depth P2C Airflow Rail Kit RoHS6

MPN: MSX6710G-FS2R2

$40,549.00

Availability:	Limited
Standard Lead Time:	20 Weeks
Technology:	InfiniBand
Max Speed:	FDR/40GbE
Connector Type:	QSFP+
Ports:	36
ECCN:	5A991
Product Brief:	View MSX6710G-FS2R2

https://store.mellanox.com/products/nvidia-msx6710g-fs2r2-switchx-2-infiniband-to-ethernet-gateway-36-qsfp-ports-2-power-supplies-ac-x86-dual-core-standard-depth-p2c-airflow-rail-kit-rohs6.html

숙제 3 (1/8)

- TCP/IP 모델의 링크 계층 흉내 내기
- 임의의 개수의 노드 (A, B, C, D, E…H) 사이의 통신 구현
  - 모든 노드는 하나의 통신 매체 (g_conn)으로 연결되어 있다.
  - g_conn이 가질 수 있는 값은 true/false 뿐이다.
- 임의의 방향의 통신을 구현하라.
  - 예) A -> B, D -> C
  - 매체가 하나이므로 충돌이 발생할 수 있다. 이를 적절한 알고리즘으로 극복하라.

숙제 3 (2/8)

- 구현은 샘플 프로그램의 node.cpp의 do_node(char node_type)와 do_node_NIC(char node_type)함수를 수정해서 구현하시오.
  - 다른 .cpp .h 파일들은 수정하지 마시오
  - 숙제 검사는 제출된 node.cpp만을 사용할 예정
  - 임의의 표준 라이브러리 사용가능
  - g_conn이외의 다른 방법으로 노드 사이의 통신을 구현하지 마시오.

숙제 3 (3/8)

- 샘플 프로그램 설명
  - do_node()와 do_node_NIC()는 동시에 실행되므로 do_node()에서 cin입력을 기다리고 있어도 do_node_NIC()는 멈추지 않고 실행된다.
    - 멀티쓰레드로 구현되어 있다.
  - 구현 가이드 : 역할 추천
    - do_node()함수는 사용자의 입력을 받아서 do_node_NIC()에 전송하는 역할
      - do_node_NIC() 전달하는 문자열은 전역 변수를 통해서 전달하는 것이 바람직.
    - do_node_NIC()는 g_conn을 감시하면서 do_node()가 저장한 데이터를 전달하는 동시에 다른 노드에서 보낸 데이터를 수신하는 역할

숙제 3 (4/8)

- 힌트 : do_node()와 do_node_NIC() 데이터 전달
  - 전역 변수를 통해 데이터를 주고 받을 수 있으며, 상태 변수를 통해 전달 과정을 공유해야 한다.
    - send_state (0: 아무런 데이터가 없다, 1:데이터를 넣었으니 NIC는 받아 가라, 2: NIC가 다 받아서 처리했으니 다음 작업을 해라)

```
volatile int send_state;
char send_mess[200];

void do_node()
{
   cout << "Enter: ";
   while (true) {
      cin.getline(send_mess, 199);
      send_state = 1;
      while (2 != send_state);
      send_state = 0;
   }
}
```

```
void do_node_NIC()
{
   while (true) {
      while (1 != send_state);
      cout << send_mess;
      send_state = 2;
      cout << "WnEnter: ";
   }
}
```

숙제 3 (5/8)

- 힌트 : Timing
  - CS, MA, CA, CD등을 구현하기 위해서는 정밀한 타이밍이 필요하므로 chrono::high_resolution_clock()의 사용을 추천한다..
  - 구현 예 : num개의 비트를 CLOCK마다 g_conn에 실어주는 함수. 비트는 value에서 가져 온다. 다른 노드와 충돌해서 신호가 변경되었는가를 검사해서 리턴 한다. 전송이 끝나면 tp도 전송이 끝난 시간으로 업데이트 된다.

```
const chrono::microseconds CLOCK{ 1000 };
bool set_signal(chrono::high_resolution_clock::time_point& tp, int num, unsigned int value)
{
   bool collision = false;
   for (int i = 0; i < num; ++i) {
      bool bit = (value & (1 << (num - i - 1))) != 0;
      g_conn.set(bit);
      while (chrono::high_resolution_clock::now() < tp + CLOCK)
         if (g_conn.get() != bit) collision = true;
      tp += CLOCK;
   }
   return collision;
}
```

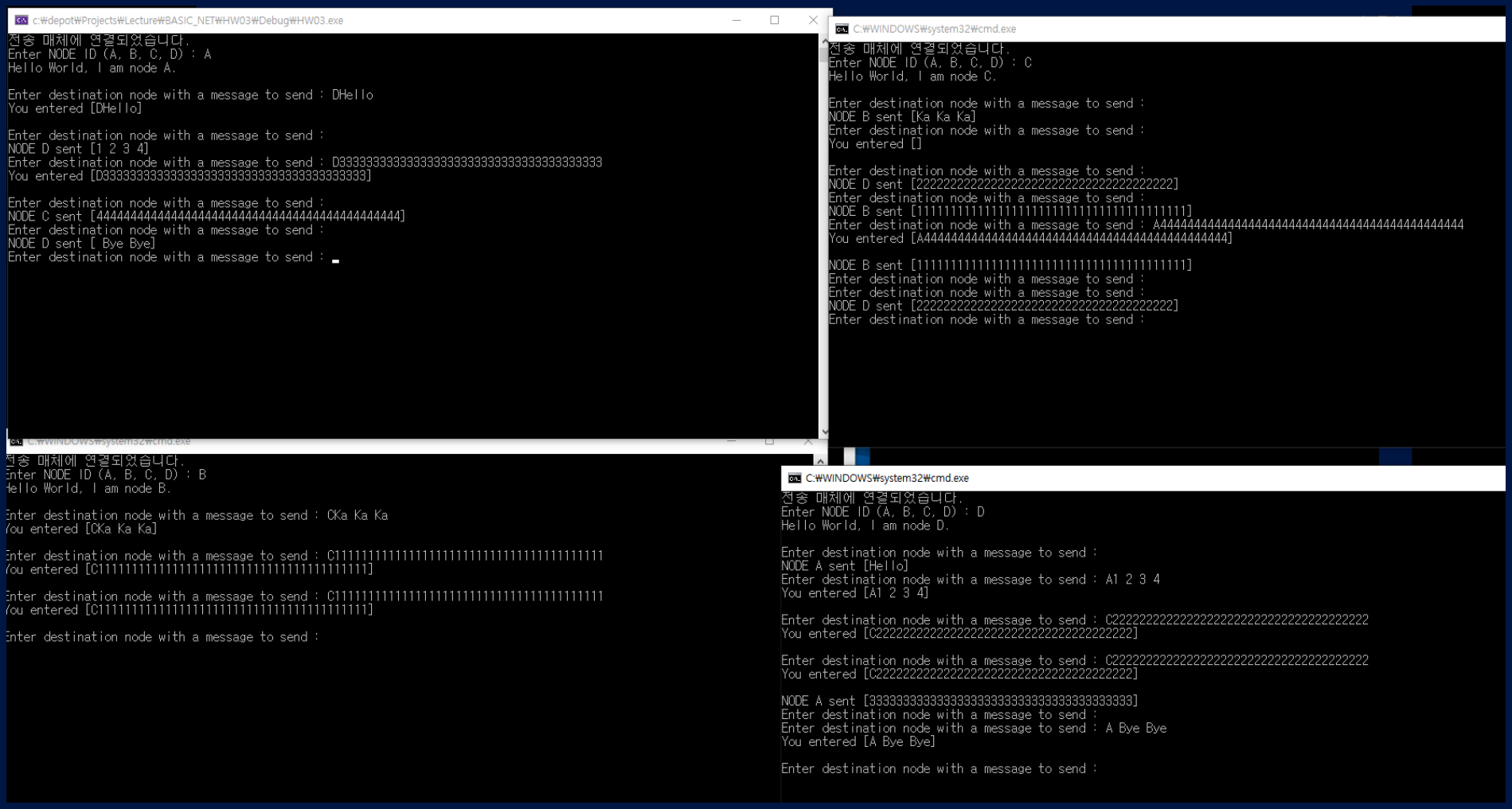
```
auto curr = high_resolution_clock::now();

bool is_collision = set_signal(curr, 4, source);
set_signal(curr, 4, dest);
set_signal(curr, 8, char_data);
```

숙제 3 (6/8)

- 제출 내용
  - node.cpp 파일
  - 구현한 방법에 대한 설명
  - 실행 스크린 샷
- 제출 방법
  - eclass에 제출

숙제 3 (7/8)

- 구현 예


c:₩depot₩Projects₩Lecture₩BASIC_NET₩HW03₩Debug₩HW03.exe
전송 매체에 연결되었습니다.
Enter NODE ID (A, B, C, D) : A
Hello World, I am node A.
Enter destination node with a message to send : DHello
You entered [DHello]
Enter destination node with a message to send :
NODE D sent [1 2 3 4]
Enter destination node with a message to send : D3333333333333333333333333333333333333333
You entered [D3333333333333333333333333333333333333]
Enter destination node with a message to send :
NODE C sent [4444444444444444444444444444444444444444444]
Enter destination node with a message to send :
NODE D sent [Bye Bye]
Enter destination node with a message to send :
C:₩WINDOWS₩system32₩cmd.exe
전송 매체에 연결되었습니다.
Enter NODE ID (A, B, C, D) : C
Hello World, I am node C.
Enter destination node with a message to send :
NODE B sent [Ka Ka Ka]
Enter destination node with a message to send :
You entered []
Enter destination node with a message to send :
NODE D sent [22222222222222222222222222222222222222]
Enter destination node with a message to send :
NODE B sent [1111111111111111111111111111111111111111]
Enter destination node with a message to send : A4444444444444444444444444444444444444444444444
You entered [A4444444444444444444444444444444444444444444444]
NODE B sent [1111111111111111111111111111111111111111]
Enter destination node with a message to send :
Enter destination node with a message to send :
NODE D sent [22222222222222222222222222222222222222]
Enter destination node with a message to send :
C:₩WINDOWS₩system32₩cmd.exe
전송 매체에 연결되었습니다.
Enter NODE ID (A, B, C, D) : B
Hello World, I am node B.
Enter destination node with a message to send : CKa Ka Ka
You entered [CKa Ka Ka]
Enter destination node with a message to send : C1111111111111111111111111111111111111111
You entered [C1111111111111111111111111111111111111111]
Enter destination node with a message to send : C1111111111111111111111111111111111111111
You entered [C1111111111111111111111111111111111111111]
Enter destination node with a message to send :
C:₩WINDOWS₩system32₩cmd.exe
전송 매체에 연결되었습니다.
Enter NODE ID (A, B, C, D) : D
Hello World, I am node D.
Enter destination node with a message to send :
NODE A sent [Hello]
Enter destination node with a message to send : A1 2 3 4
You entered [A1 2 3 4]
Enter destination node with a message to send : C22222222222222222222222222222222222222
You entered [C22222222222222222222222222222222222222]
Enter destination node with a message to send : C22222222222222222222222222222222222222
You entered [C22222222222222222222222222222222222222]
NODE A sent [3333333333333333333333333333333333333333]
Enter destination node with a message to send :
Enter destination node with a message to send : A Bye Bye
You entered [A Bye Bye]
Enter destination node with a message to send :

숙제 3 (8/8)

- 주의 사항
  - 복수의 노드에서 동시에 데이터를 전송할 때 충돌 처리가 구현되어 있어야 한다.
    - 전송속도가 너무 빠르면 충돌상황을 테스트하기 어려우므로 전송속도를 늦춰서 테스트 할 예정
      - 속도를 CLOCK const 변수로 설정할 수 있도록 하시오
  - Node의 개수가 CPU의 Core개수 보다 많을 경우 빠른 전송속도에서 오류가 많이 남.
    - Context Switch로 인한 timing 오류
  - Quad Core CPU에서 테스트할 예정이므로 4개의 노드까지만 테스트할 예정