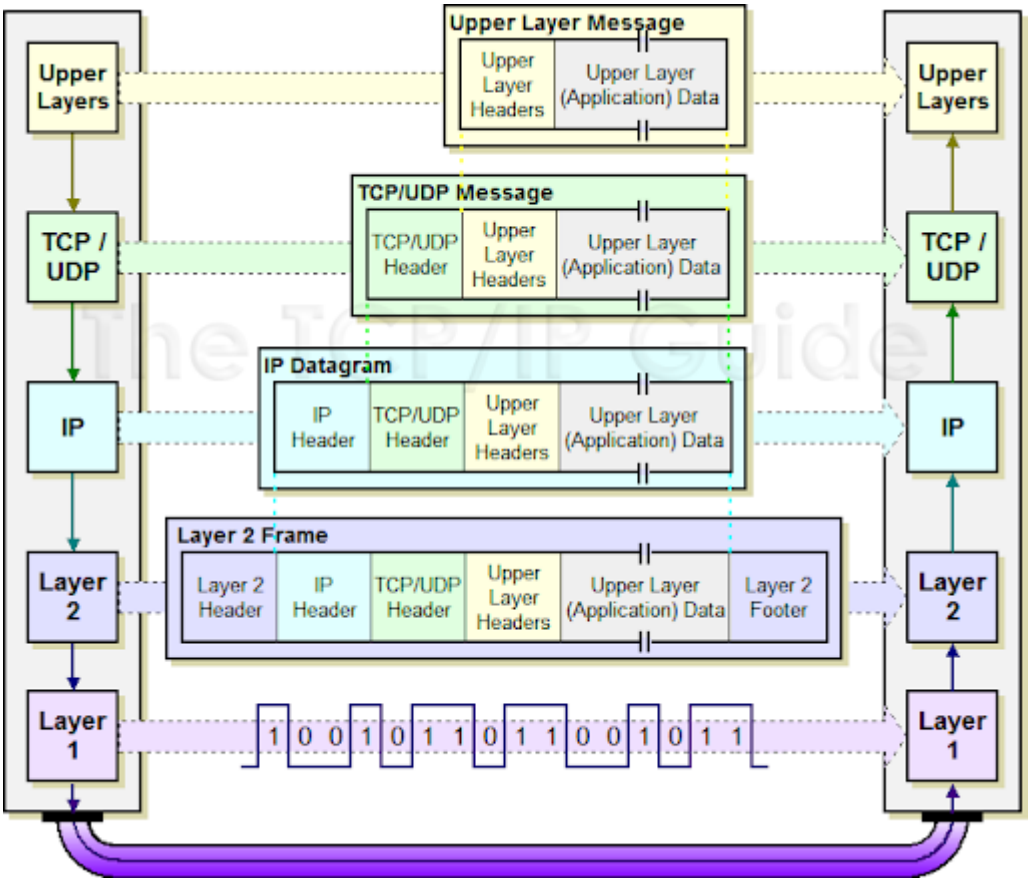
Ch2. Link Layer

Created	@Jun 16, 2020 2:23 AM
Tags	컴망



- 물 / 데 / 네 / 전 / 세 / 표 / 응
- Bit / Frame / Datagram(Packet) / Segment / Message...
- 하위 계층으로 내려오면서 Encapsulate. 상위 계층으로 올라가면서 Decapsulate.

Link Layer(Data-Link Layer) : Introduction

- **Node** : 호스트와 라우터

- **Link** : 통신 경로 상의 인접한 노드들을 연결하는 통신 채널 → 유선 링크, 무선 링크, LAN, ...
- **Frame** : **Data-Link Layer의 패킷. 3계층(Network Layer)의 Datagram을 캡슐화한 것.**

데이터그램을 한 노드에서 물리적으로 인접한 다른 노드로 Link를 통해 전달.

Link Layer Services

1. **Framing, Link access**
    - 데이터그램에 Header, Trailer를 추가해서 → 프레임으로 캡슐화.
    - Source, Dest 구분을 위해 **프레임 헤더에 MAC 주소 포함**.
2. **Reliable delivery**
    - 비트 오류가 낮은 링크 (광섬유)에서는 안 쓴다.
    - 무선 링크는 에러율이 높다.
3. **Flow control**
    - 버퍼 오버플로우를 막기 위해, **송신 측에서 수신 측의 받는 속도에 맞게 보내는 양을 제어**
4. **Error detection, Error correction**
    - 에러 : 신호 약화, 잡음에 의해 발생
    - 수신 노드에서 Error detection 해서 송신 노드에게 재전송 신호 보내거나 프레임 걍 버림
    - 재전송 요청하지 않고 Correction해도 된다.
    - **Parity**
        - **Single bit parity** : **Single bit error의 Detection**만 가능. (하나 틀린 거 적발)
        - **Two-dimensional bit parity** : **Single bit error의 Detection, Correction** 가능. (하나 틀린 거 적발, 수정 가능)
    - **Forward Error Correction** : 에러 위치를 앞에서부터 찾아서 직접 수정.
    - **CRC (Cyclic Redundancy Check)** : 보내고자 하는 데이터 D가 있고, 송수신 측에서 G를 알고 있다면 → 송신 측에서는 나머지를 붙여서 보내고, 받는 측에서

는 나머지가 안 생기면 잘 받은 것.

Multiple access links, protocols

Link에는 두 가지가 있다.

- Point-to-Point
- **Broadcast (Shared wire / Shared medium)**
  - Medium에 access하는 protocol이 있어야 한다
  - **MAC : Medium Access Control** protocol

MAC protocols

세 종류로 나눌 수 있다.

- **Channel partitioning** : channel을 3개로 나눈다 (시간, 주파수 관리 필요)
- **Taking turns** : 토큰을 갖고 있어야지만 보낼 수 있다. 내가 다 보내고 토큰을 다른 사람에게 넘긴다.
- **Random access** : 걍 보낼거 있으면 보낸다. 채널 할당필요 X.
  - 막 보내니까 충돌 생길 수 있다. → collision detection 필요. 충돌 감지했다면, collision recover도 필요.

Channel partitioning, Taking turns 두 방법은 오버헤드가 크다.

Taking turns protocols

- **Polling** : 마스터 하나, 슬레이브 여러 개. → 폴링 오버헤드, 레이턴시, 마스터 고장나면 끝.
- **Token passing** : token ring (돌아가면서 토큰 전달), token bus, ...

Random access protocols

두 개 이상의 노드가 데이터를 보내면 → **"collision"**

Random access MAC 프로토콜의 예시:

- **Pure (Unslotted) ALOHA**
- **Slotted ALOHA**
- **CSMA**

Pure ALOHA

데이터 준비되면 바로 보낸다. 동기화 없다.

어느 시점에서 데이터가 보내질지 모른다.

끝에서 데이터가 충돌나면 둘 다 의미없는 정보가 된다.

Slotted ALOHA

보내고자 하는 데이터가 있어도 바로 보내지 않고, 시간을 나눈다 C E C S E C E S S …

충돌이 일어나는 경우는 같은 시간에 데이터 보내는 경우밖에 없다.

CSMA (Carrier Sense Multiple Access)

데이터 보내기 전에 확인하고 아무도 안 보내면 내가 보내면 되잖아? → Listen Before Transmit

- **Carrier Sense** : **그 주파수 (carrier)를 미리 확인해서 (sense) 누가 보내고 있는지 확인**하는 것
  - If channel sensed **idle** → 전체 프래임 보낸다
  - If channel sensed **busy** → 나중에 보낸다

CSMA collisions : Multiple Access이기 때문에 충돌 발생하면, 전체 패킷 전송시간이 낭비된다.

- **CSMA/CD** (collision detection)
  1. NIC (network interface card)가 상위 계층인 NL로부터 datagram을 받는다. (아빠 → 엄마)
  2. **NIC가 Channel idle을 감지**하면, **프레임 전송**을 시작. (Channel busy라면 idle 할 때까지 대기 후 전송)
  3. NIC가 collision detection 없이 모든 프레임을 다 보냈다면, 성공.
  4. NIC가 **전송 중 다른 transmission을 감지**했다면, **abort하고 jam signal 보냄** (다른 친구들도 충돌임을 감지할 수 있도록 이상한 무작위 정보 막 보냄)

5. abort 후, Backoff만큼 기다리고 다시 보낸다.
    - Backoff : "지금 당장은 못 보내니까 나중에 다시 하겠다"
    - **Binary Backoff** : **m번째 충돌 이후, {0, 1, 2, ..., 2^{m-1}} 중 하나 골라서, 그거 곱하기 512 비트 시간만큼 기다렸다가, 다시 2번으로 돌아간다.**

ARP protocol : same LAN

A가 B에게 datagram 보내려 하는데, B의 맥주소를 모른다. (ARP table에 B의 맥주소가 없다. 현재 A가 아는 건 B의 IP주소 뿐, 물리적 주소는 모르는 상태.)

→ **A는 Shared medium에 있는 모든 노드에게 ARP query packet을 Broadcast한다.**

> 💡 ARP table : IP주소에 해당하는 맥주소 적어놓은 테이블

1. A가 B의 IP주소를 포함하는 **ARP query packet**을 방송한다.
    - **ARP query packet**은 **Dest MAC주소에 111111...11** 넣는 걸로 식별한다.
2. 노드들은 패킷 받아서 Dest MAC이 자신 주소랑 같으면 자기거고, 아니면 자기거 아니니까 버린다. **근데 목적지 주소가 111...1이면 다 받으라는 소리니까 일단 읽는다.**
3. 뜯어보니까 "B의 IP주소에 해당하는 MAC주소 뭔지 아냐?"라는 내용. 읽고 무시할 애들은 무시하고, **B가 받아서 자기 맥주소 뿌리겠지.**

Ethernet

가장 많이 쓰는 랜 기술.

Terminator (저항) 달아놔야 함.

- **Bus 모양** : **중간에 끊어지면** 거기서 데이터들이 가다가 돌아오니까 엉망이 된다. **전체가 통신이 불가능해짐.** 한 쪽이 통신 중이면 다른 통신 불가.
- **Star 모양** : **한쪽 라인이 끊어져도 걔만 통신이 안 될 뿐**이고, 다른 애들은 다 가능. 한 쪽이 통신 중이여도 다른 통신 가능.

Conclusion : Star 모양이 더 좋다.

특징 두 가지:

- **Unreliable** : 데이터 받는 애가 잘 받았는지 못 받았는지 CRC 보고 판단하는데, **에러가 있으면 걍 버린다. 보내는 애가 걔가 잘 받았는지 확인할 방법이 없다.**
- **Connectionless** : Handshaking을 하지 않는다. 연결을 맺고 시작하는 게 아니라 걍 보낸다.

Ethernet frame 구조

- **preamble (7바이트)** : **10101010 패턴이 6번** 반복됨. **마지막만 10101011**. **보내는 애와 받는 애의 싱크를 맞추기 위함.**
- **dest address (6바이트)** : 받는 애 MAC 주소. 아까 말했듯이 111...1이면 모든 애들이 다 받음.
- **source address (6바이트)** : 보내는 애 MAC 주소.
- **type** : 상위 레이어 프로토콜.
- **data (payload)** : 전달할 데이터
- **CRC**

Ethernet switch

- **Transparent** : 호스트들은 자기들이 스위치에 연결되어 있는지 모른다.
- **Plug and play, Self-learning** : 스위치에 대해 따로 설정할 필요가 없다.

Virtual LANs

LAN Segment : Single broadcast domain. **Broadcast 하나는 그 LAN Segment 안에서만 전파되어야 한다.** 다른 LAN Segment로 넘어가면 안 된다.

물리적으로 떨어져 있는 LAN Segment끼리 Broadcat를 전달하기 위해 → **VLAN.**

- **Port based VLAN** : 포트 기반으로 관리해줌. 스위치에서 몇 번 포트까지는 어디 랜, 몇 번까지는 어디 랜, ...
    - **Trunk port** : 스위치 여러개를 같이 연결할 수도 있다. **스위치끼리 연결하려고 쓰는 포트.**