6/14 링크 계층

유선 링크 계층

- application 단의 메시지가 스택을 타고 내려오면서 세그먼트, 패킷에 담기고, 링크 계층에 와서는 프레임에 담겨서 전송됨.
- 패킷이 목적지까지 도달하기 위해서는 우선 패킷을 게이트웨이 라우터에 보내야 한다. 근데 이 말은 개념화된 말이고, 실질적으로는 링크 레이어의 작동방식을 생략한 말임
- 실제로 게이트웨이까지 가는 길에는 링크 단에서 많은 사람들이 공유하는 채널이 존재해서, physical layer로 가서 전자기파의 형태로 바뀌면 채널 전역으로 퍼진다. 따라서 게이트웨이에 물린 모든 사람들에게 전달되고, 사실상 broadcast되는 것. 목소리가 주변에 다 들리는 것과 같은 느낌으로…
- 그래서 두명 이상이 동시에 얘기하면 섞임(collision) → 라우터가 이를 정상적으로 해독할 수 없어서 채널이 낭비됨
- 링크 레이어는 이 단에서 충돌이 발생했을 때 이를 해결한다.

Introduction

- 연결의 문제가 아니라, 딱 한 단계에서 충돌을 어떻게 해결해서 다음으로 넘길 수 있을까
- 일단 유선 상황을 가정
- transport layer / network layer는 OS단에서 코드로 구현되어 있지만 Link layer는 랜카드(network interface card) 안에 구현되어 있음
- 위에서 나온 패킷은 프레임에 담긴다

Multiple access links, protocols

- 호스트와 호스트간의 링크가 전용선이었다면 신경쓸 필요가 없음
- 위에서 언급한 전자기파의 특성 때문에 medium access ctrl(MAC)이 필요함
- 그래서 모든 네트워크 전자기기에는 mac이 있는거고, 무선에서는 wi-fi라는 일종의 mac protocol이 서로 간섭 안되게 잘 조절하고 있는것

MAC Protocol 0. IDEAL

broadcast channel of rate R bps가 있을 때

1. 하나의 노드가 데이터를 전송하고 싶으면 bandwidth R을 모두 사용할 수 있어야함

2. n개의 노드가 보내고 싶으면 R/n씩 사용할 수 있어야함
3. 분산처리(decentralized)되었으면 좋겠다
    a. 처리를 조절하는 중앙서버 같은게 없었으면 좋겠다
4. 단순했으면 좋겠다

MAC protocols: taxonomy

정말 많은 MAC protocols가 제안되었는데, 그걸 분류하면 아래 세 카테고리가 있음.

- channel partitioning, 채널 쪼개서 보냄
    - TDMA(time division multiple access), 시간을 쪼개 여러 사람들이 접근할 수 있도록 함. 즉 내 time slot에만 내가 보낼 수 있음. 자원이 아주 많이 낭비됨(슬롯 할당 문제)
    - FDMA(frequency -), 주파수가 배정되어 있음. TDMA랑 같은 장단점을 가짐
- random access, 보내고 싶을때 보냄
    - 당연히 충돌 발생함, 충돌 탐지 및 해결이 중요한 이슈
    - ALOHA (1970s)
        - 하와이 대학에서 군도상의 통신을 위해 개발
    - CSMA, CSMA/CD, CSMA/CA
        - Carrier Sense Multiple Access, 운반자를 탐지함
        - listen before transmit, 할 말 생각나도 누군가 얘기하고 있다면 기다려야
        - 근데 조용한 상황에서 두명이 동시에 치고들어오면 충돌남
- “taking turns”

MAC protocols: CSMA

현실적인 상황에서의 **Collisions**

- Node 4개가 같은 채널을 공유할 때, N1이 시그널을 보내면 채널 내에 시그널이 퍼질 것
- 만약 시그널이 퍼지고 있는 중에 (carrier sensing이 감지되기 전) N4가 시그널을 보내면 (propagation delay) 프레임이 충돌함
- 그러면 두 개의 프레임은 다 날아감, 두 개의 시그널이 끝날 때까지 채널은 낭비되는 것
- propagation delay를 0으로 만들면 충돌이 없을 건데, 이거 기술적으로 불가능함 (빛의 속도기 때문)

- 그래서 충돌은 날 수밖에 없다가 결론임
- Quiz. CSMA에서 어떤 두 사람이

CSMA**/CD (Collision Detection) — 유선 상황 가정 (used in 802.11)**

- 충돌을 감지한 즉시 멈춘다
- 이게 없을 땐 오디오 끝까지 겹치는데, 이게 있으면 충돌 난 순간 둘다 멈춤 → 결과적으로 낭비되는 시간을 줄일 수 있음
- 근데 그러면 그 다음엔 어떡함?
- `NIC enters binary(exponential) backoff`
  - n번의 충돌이 있었다면, 0부터 2^(n-1) 중에 하나를 골라서 그 시간을 자기 대기 시간으로 삼는다
  - 이거는 n개가 충돌난게 아니고 충돌의 횟수가 n번째라는 뜻임, 연속적으로 충돌횟수가 늘어나면 이 랜덤 시간의 범위가 넓어진다
    - 왜이렇게 만듬? 충돌이 감지되면 각 노드가 알 수 있는건 '나 말고 누군가 있다'임
    - 그래서 계속 충돌나면 사람이 많다고 생각하고 숫자 범위를 늘리는 것임
    - 같은 숫자를 골라서 충돌이 또 나도 뭐 어쩔수 없는거임
- 결국 backoff란 딜레이를 뜻하고, 사람들이 많으면 딜레이가 커질 수밖에 없음. 네트워크 전체의 문제라기보다는 exponential backoff의 문제일 확률이 높음

Channel Partitioning vs Random Access

- 채널 파티셔닝은 사람이 많을수록 유리
- 랜덤 액세스는 사람이 적을수록 유리

MAC Protocols: "Taking Turns"

- Master Node가 있어서 초대하는 방식. 문제는 얘가 멈추면 다같이 망하는 것 때문에 잘 쓰이지 않음
- Token을 가지고 있는 host만 데이터를 전송할 수 있게 하는 방식. 전송할 게 없으면 토큰을 다른데로 넘김. 근데 이건 토큰 유실시 망함

WLAN (Wired Lan)상황

MAC Protocols: Ethernet

- CSMA/CD(No slot, carrier sense, collision detection, random access), Stateless, Unreliable
- “Dominant” WLAN technology
- cheap ($20 for NIC)
- physical topology
  - bus (-mid 90s)
  - switch (prevails today)
- Ethernet Frame Structure
  - IP Packet이 Link Layer의 전송단위인 Frame의 data부분에 담김
  - header의 필드는 4개뿐임
    - preamble
    - src address (MAC)
    - dest address (MAC)
    - type : 데이터가 어떤 (상위 레이어의) 프로토콜인지. 보통 IP일것
  - 특이하게 CRC라는 tail이 있음(err check)
- Ethernet uses CSMA/CD
  - Ethernet에서 발생한 Collision을 찾지 못하는 경우가 있음?
    - 만약 Node 1에서 어떤 정보를 보냈는데, delay 때문에 충돌이 발생했다고 하자.
    - 근데 만약 Node 1이 메시지를 다 보낸 상황이었다면, 실제로는 충돌이 났지만 Node 1은 detect를 하지 못한 상황이 됨
    - 그러면 Node 1은 다음 메시지를 보냄
    - Collision이 났다고 판단하기 위해서는 Node 1이 다 보내기 전까지까지 발생한 충돌이 Node 1에게 다시 왔어야함.
      - 즉 Node 1이 좀 더 데이터를 많이(오래) 보냈다면 detect 되었을 것임
      - 결국 propagation delay 때문에 생긴건데, 이걸 고칠순 없으니까
      - 할 말이 없어도 최소한 Minimum Frame Size(64byte) 만큼은 말하라고 강제함
        - 그래서 의미없는 padding을 집어넣는 일이 발생함

- 그러면 결국 gateway로 나가는 건 packet을 감싼 frame임, 그럼 여기서 gateway router 어떻게 알고 보냄?
  - DHCP를 통해서 GWR의 IP는 알수 있음, IP의 Packet Header는 내 IP(DHCP), Google IP(DNS)를 갖고 있을 것
  - 근데 얘는 Frame의 data로 들어갔고, Frame의 header에는 MAC이 들어가고, dist에는 `GWR의 MAC` 이 들어갈 것임
    - 왜? broadcast이기 때문에 나머지는 자기게 아닌줄 알고 버릴수 있어야함
  - `GWR의 MAC` 어떻게알죠
    - 각 Host의 내부에 ARP(Address Resolution Protocol) table라는 게 있어서, IP 주소와 MAC address를 맵핑해놨음… 그래서 GWR의 IP에 해당하는 MAC를 찾으면 됨
    - 근데 처음엔 없었을 것 아님? 어떻게 채움? → 그게 ARP임.
      - ARP request라는 frame을 broadcast하고, dest는 FFFFFF임 (전체)
      - 그래서 dest에 게이트웨이 주소가 있으면 게이트웨이에서 보고 응답해드립니다
      - 이 테이블은 캐시 테이블이어서 TTL이라는 유효기간을 가짐. 그 이후에는 다시 ARP req를 보낸다.
- Ethernet 환경에서 그럼 어떻게 데이터가 감?
  - `HOST NIC — GWR — R1 — R2 — … — RN — Google`
  - 각 라우터는 Frame을 벗겨서 IP Packet을 확인한다.
  - 각 라우터는 forwarding table을 참조해 다음 라우터로 forwarding해야 함
  - 그러기 위해서는 이걸 또 frame으로 감싸야함 (src = GWR MAC ≠ NRC dist MAC, 왜냐하면 라우터는 interface가 여러개임)
  - GWR은 ARP로 R1 MAC를 찾고, R1, R2, RN은 위와 같은 과정을 반복해 Google로 이걸 전송

MAC Addresses

- 48bit, 앞의 24bit는 제조사, 뒤의 24bit는 인터페이스의 고유번호로 이루어져 있음
  - 랜카드 주소기 때문에 컴퓨터 네트워크의 고유번호가 됨

- 그래서 MAC를 변경한다는 의미는 src address를 바꿀 수 있는 것임 (MAC 스푸핑)