Stateless

Non-persistent HTTP

* Response time = 2RTT + file transmission time
* RTT -> 데이터가 클라이언트에서 서버에 갔다가 다시 클라이언트로 돌아오는데 걸리는 시간

FTP

POP3 protocol

DNS : domain name system,

* DNS service => hostname을 ip 주소로 변환시켜주는 서비스
* a distributed, hierarchical database
* root name servers -> 13개잇음
* Local DNS name server 반드시 hierarchy를 따를 필요 없음
* DNS records
  + RR format : (name, value, type, ttl)
  + Type = A
    - name is hostname
    - value는 ip 주소
  + type = NS
    - name is 도메인
    - value is 해당 도메인에서의 호스트네임
  + type = CNAME
    - name은 별명
    - value는 실제 이름
    - www.ibm.com => 별명, servereast.backup2.ibm.com => 진짜 이름
* DNS의 Transport Layer Protocol은 UDP, (TCP)

P2P

* DHT(Distributed Hast Table)

Socket programming

* UDP : 속도, 성능에 유리, no handshaking
* TCP : 데이터 전송에 있어서 정확, 3-way handshaking
  + TCP에서는 listen과 accept이 있음( = connection establishment)

Transport services and protocols

* End to end 통신
* Message – Application Layer
* Segment : transport Layer
* Datagram(packet) – Network layer
* Frame – data link layer
* Packet

Network layer

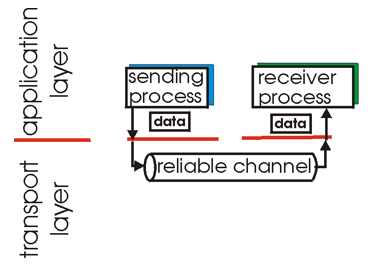
* Host to Host : door

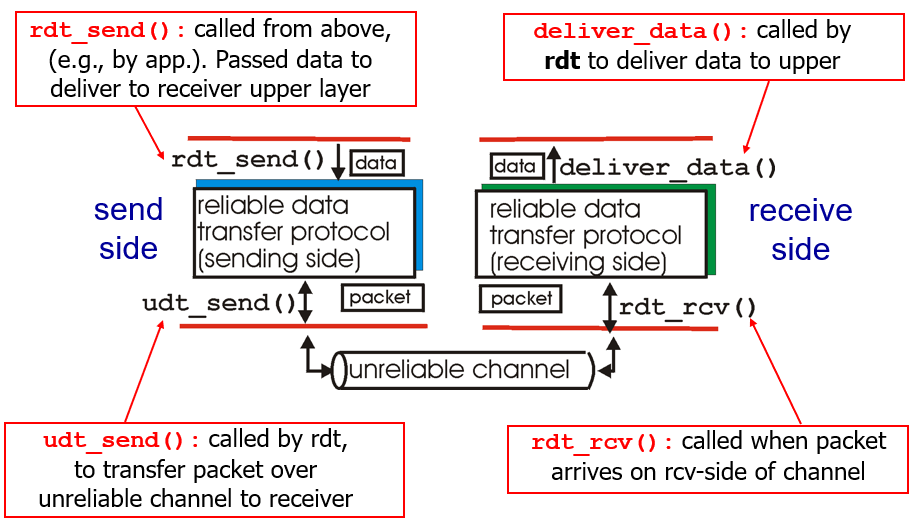
Multiplexing and demultiplexing

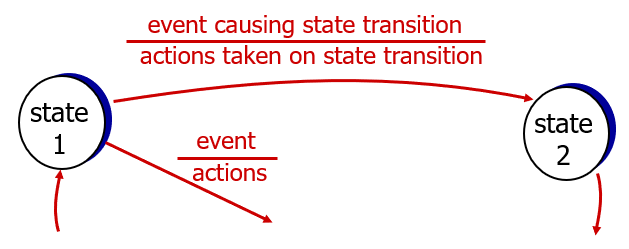
* TCP 구별
  + Source ip address
  + Source port number
  + Dest ip address
  + Dest port number

UDP(user datagram protocol)

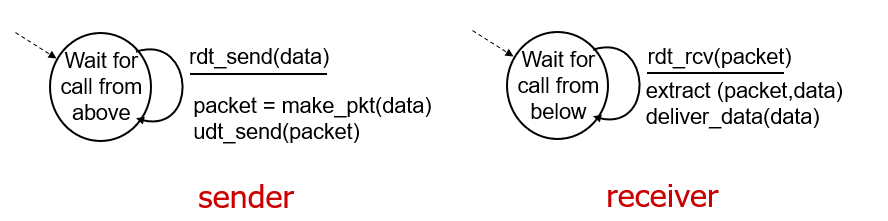
Principles of reliable data transfer(RDT)



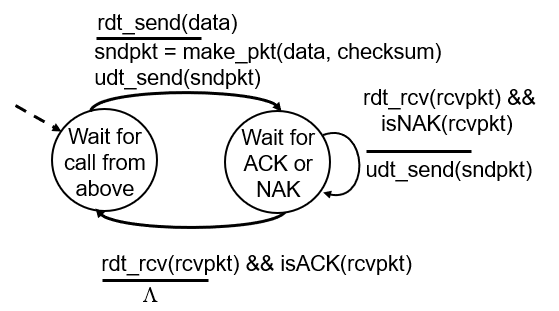




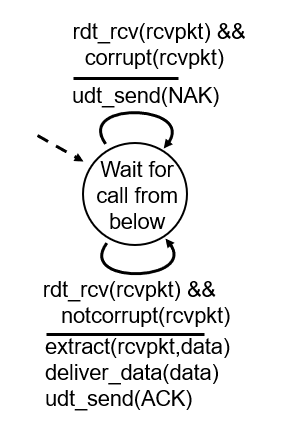
* RDT 1.0 : 채널 자체에 에러가 없다고 가정



* RDT 2.0 : 비트 에러가 일어날 수 있다고 생각하고 이때 에러 처리에 대해 생각함, ACK와 NAK라는 신호를 보냄(feedback)



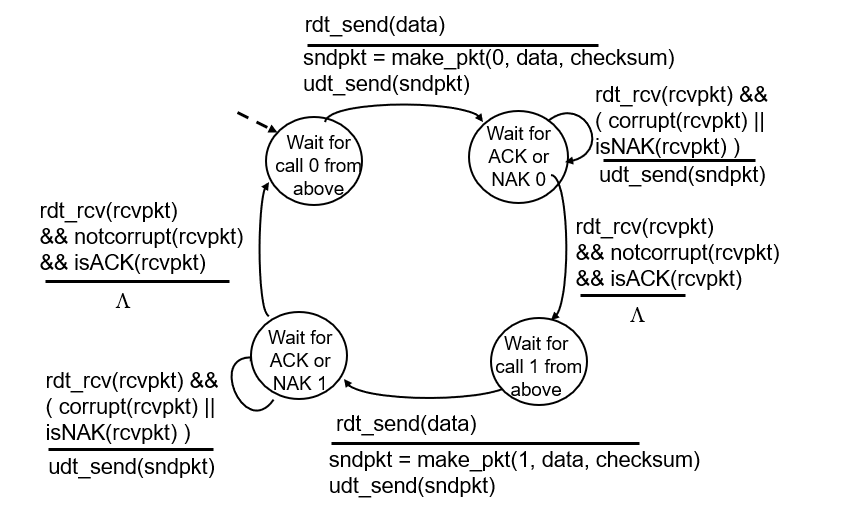
Sender



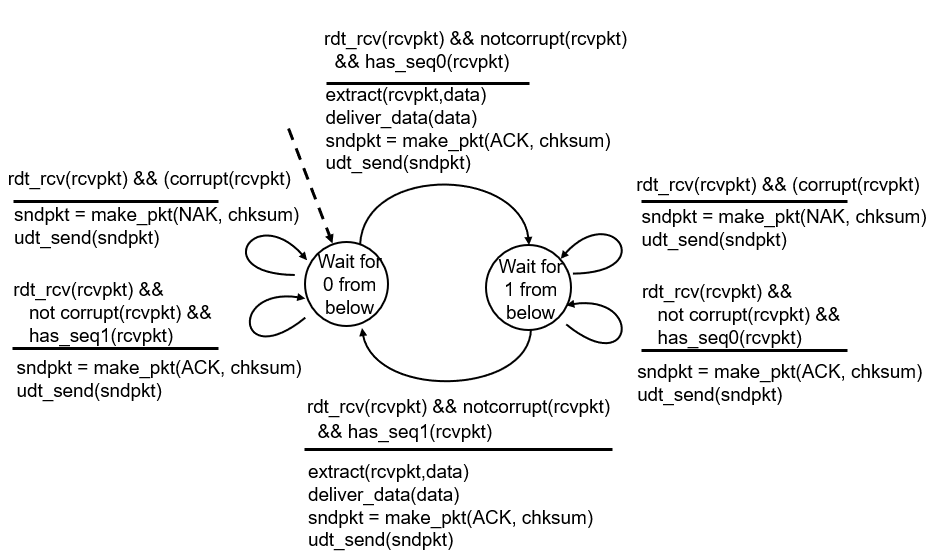
Receiver

* Sender -> receiver는 비트 에러 디텍션을 하지만 그 반대는 하지 않음

RDT 2.1 –



Sender

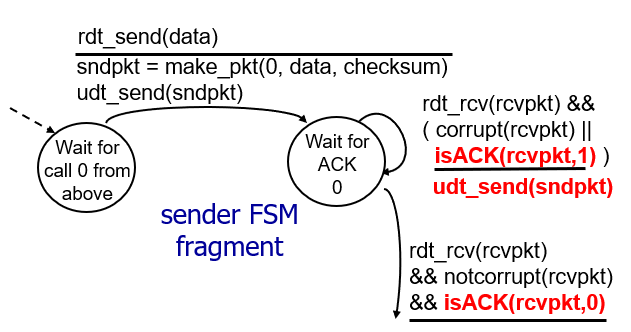


Receiver

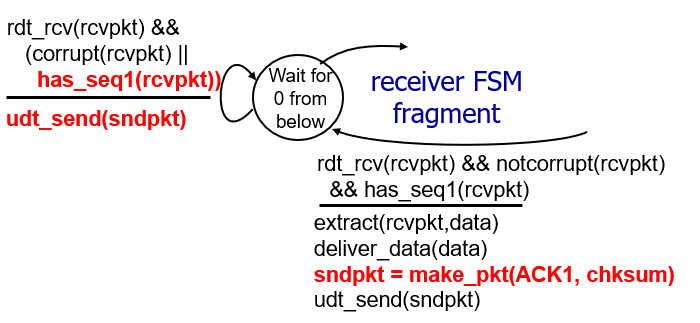
* Sequence #는 2개면 충분, stop & wait 기법을 사용한다고 가정하기 때문에
* Expected packet # -> 내가 원하는 패킷을 보내야만 함
* Receiver가 보낸 패킷이 사라질 경우 sender가 알 방법이 없음

RDT 2.2

* NAK 대신에 이전에 사용했던 번호의 ACK를 보냄

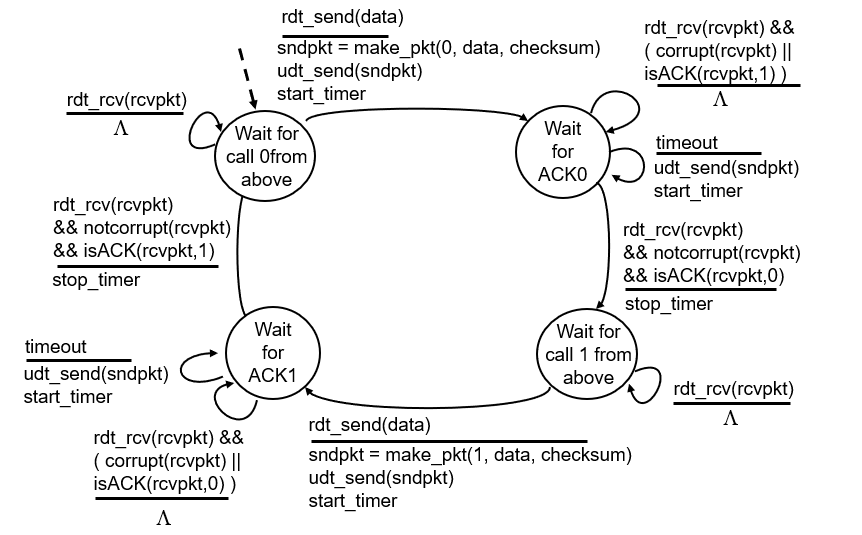


Sender



Receiver

RDT 3.0 : 비트 에러(by checksum)와 비트 손실(by timeout)을 모두 handle하는 채널



Pipelined protocols

Sender를 위한 버퍼 공간이 필요하다

Go-back-N => cumulative ack(순서대로 제일 마지막에 제대로 받은 PCK 번호를 ACK), Receiver Buffer가 없음, 제일 오래된 패킷만 타이머를 유지하고 타임아웃이 되면 전부 다 다시 보냄

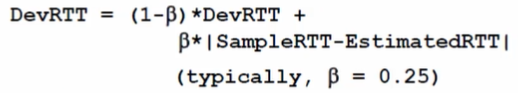
Selective Repeat => individual ack(각각의 PCK 번호에 ACK), Receiver Buffer Size : N, 보내는 패킷마다 타이머를 유지

Selective Repeat : dilemma – Window size(n) vs seq # size(m) – (m >= 2\*n)

TCP

Timeout의 기준 – 예측되는 RTT





TimeoutInterval = EstimatedRTT + 4\*DevRTT

TCP 특징

* Cumulative acks
* Single retransmission timer
* Retransmission 하는 경우
  + Timeout events
  + Duplicate acks

TCP fast retransmit

3개의 중복된 ACK를 받으면 timeout이 되기 전에 해당하는 seq number의 데이터를 다시 보냄

Connection Management // Connection-oriented => 3-way handshaking

Congesition control

* Congestion window를 만들자 ( AIMD - 1개씩 늘림)
* Slow Start (1로 시작해서 2배씩 증가. Loss가 일어나면 1에서 다시 시작)
* Fast Recovery (Loss가 일어나면 절반으로 줄이고 AIMD)