3️⃣ Network Layer

태그 8주차 CS

3.1 트랜스포트 계층 서비스 및 개요

multiplexing: 하나로 섞는 것

demultiplexing: 여러 개로 나뉘는 것

Transport services and protocols

- transport layer: process 간의 `logical communication` 제공
- `logical communication` : 애플리케이션의 관점에서 봤을 때 프로세스들이 호스트에 직접 연결된 것처럼 보이는 것을 의미
- `segment` : 트랜스포트 계층 패킷
  - sender: 애플리케이션에서 socket을 통해 transport layer로 내려오면 그 메시지를 segment로 나눠서 network layer로 내려보냄
  - receiver: 반대

Transport vs Network layer

- 트랜스포트 계층은 프로토콜 스택에서 네트워크 계층 바로 상위에 존재한다.
  - 트랜스포트 계층: 다른 호스트에서 동작하는 프로세스들 사이의 논리적 통신 제공
  - 네트워크 계층: 호스트들 사이의 논리적 통신 제공
- 예시

```
애플리케이션 메시지 = 봉투 안의 편지
프로세스 = 사촌 형제
호스트(종단 시스템) = 집
트랜스포트 계층 프로토콜 = '앤'과 '빌' -> 집 안에서 우편함(네트워크 경계)까지만 운반, 논리적 통신 제공
네트워크 계층 프로토콜 = 집배원 포함한 우편 서비스
```

Internet transport-layer protocols

- IP: 인터넷 네트워크 계층 프로토콜
  - best-effort delivery service: 호스트 간 논리적 통신 제공, 어떠한 보장도 하지 않음(순서, 무결성)
  - unreliable
- UDP: unreliable, unordered delivery, best-effort
  - 세그먼트 or 데이터그램
- TCP: reliable, in-order delivery
  - 세그먼트

3.2 다중화와 역다중화

- 종단 시스템 사이의 IP 전달 서비스(호스트-호스트)를 종단 시스템에서 동작하는 두 프로세스 간의 전달 서비스(프로세스-프로세스)로 확장하는 것
- 목적지 호스트에서 트랜스포트 계층은 아래 네트워크 계층으로부터 세그먼트를 수신
  - 트랜스포트 계층의 의무: 호스트에서 동작하는 해당 애플리케이션 프로세스에게 전달
- 트랜스포트 계층은 세그먼트를 소켓에게 전달
  - 소켓: 중간 전달자, 프로세스가 가지고 있음, 유일한 식별자(포트 번호)를 가짐
  - 세그먼트: 필드 집합으로 식별한 후 소켓으로 보내짐(트랜스포트 계층)

트랜스포트 계층이 네트워크 계층으로부터 데이터를 수신할 때 프로세스에게 수신한 데이터를 어떻게 전달하는지

- 소켓: 프로세스가 가지고 있는 것
  - 네트워크 ↔ 프로세스 데이터 전달(출입구 역할, 중간 매개자)
  - 하나의 유일한 식별자를 가지고 있음(UDP, TCP에 따라 포맷 달라짐)

역다중화

- 트랜스포트 계층 세그먼트의 데이터를 소켓으로 전달하는 작업
- 세그먼트 필드 검사 후 소켓으로 보냄
- 하나 → 여러 개

다중화

- 출발지 호스트에서 소켓으로부터 데이터를 모아서 생성한 세그먼트들을 네트워크 계층으로 전달하는 작업
- 여러 개 → 하나

트랜스포트 계층 다중화의 두 가지 요구사항

1. 소켓의 유일한 식별자
2. 세그먼트의 출발지 포트 번호 필드와 목적지 포트 번호 필드

역다중화 서비스 순서

1. 호스트의 각 소켓은 포트 번호를 할당받는다.
2. 세그먼트가 호스트에 도착하면, 트랜스포트 계층은 세그먼트 안의 목적지 포트 번호를 검사하고 상응하는 소켓으로 세그먼트를 보낸다.
3. 세그먼트의 데이터는 소켓을 통해 해당되는 프로세스로 전달된다

→ UDP의 기본 동작 방식

비연결형 다중화와 역다중화

- `목적지 IP 주소`와 `목적지 포트 번호`에 의해 UDP 소켓이 식별
- 출발지 포트 번호는 회신 주소의 부분으로 사용됨(받은 곳으로 답해야 할 때)

연결지향형 다중화와 역다중화

- TCP 소켓은 `출발지 IP 주소`, `출발지 포트 번호`, `목적지 IP 주소`, `목적지 포트 번호`로 식별됨

TCP 연결 설정

- `출발지 IP 주소`, `출발지 포트 번호`, `목적지 IP 주소`, `목적지 포트 번호`에 의해 식별

1. '환영 소켓'은 포트 번호 12000을 가진 TCP 클라이언트로부터 연결 설정 요청을 기다림
2. TCP 클라이언트는 소켓을 생성하고, 연결 설정 요청 세그먼트를 보냄
3. 서버 프로세스로 동작하는 컴퓨터 호스트 OS가 목적지 포트 12000을 포함하는 연결 요청 세그먼트를 받으면, 이 세그먼트를 포트 번호 12000으로 연결 수락을 기다리는 서버 프로세스로 보냄
4. 그 다음 도착하는 세그먼트의 4가지 튜플 값이 전부 일치하면, 이 소켓으로 역다중화

3.3 비연결형 트랜스포트: UDP

- RFC 768에 정의된 트랜스포트 계층 프로토콜이 할 수 있는 최소 기능으로 동작
  - 다중화 & 역다중화
  - 간단한 오류 검사

UDP 동작 순서

1. 애플리케이션 프로세스로부터 메시지를 가져와 다중화/역다중화 서비스에 대한 출발지 포트 번호와 목적지 포트 번호 필드를 첨부하고 다른 두 필드 추가 후 최종 세그먼트를 네트워크 계층으로 넘겨 줌
2. 네트워크 계층은 트랜스포트 계층 세그먼트를 IP 데이터그램으로 캡슐화하고, 세그먼트를 수신 호스트에게 전달한다.
3. 세그먼트가 수신 호스트에 도착한다면, 목적지 포트 번호를 사용해 데이터를 해당하는 애플리케이션 프로세스로 전달한다.
- UDP는 세그먼트를 송신을 위해 핸드셰이크를 사용하지 않음 → 비연결형

사용

- DNS: 전형적인 UDP 사용 애플리케이션 계층 프로토콜의 예
- 실시간 멀티미디어 애플리케이션

UDP 장점

- 연결 설정이 없다. (지연도 없다.)
- 연결 상태가 없어서 간단다.
- 작은 헤더 사이즈(8바이트, TCP: 20바이트)
- UDP는 프로세스가 데이터를 UDP에 전달하자마자 데이터를 세그먼트를 만들고, 즉시 네트워크 계층으로 전달하는 것이 가능 (↔ TCP는 아님, 혼잡 제어 메커니즘 존재)

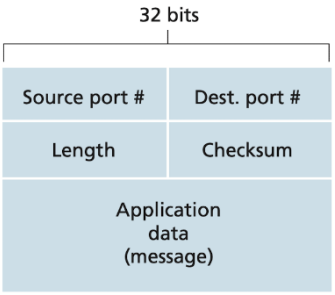
UDP 단점

- 혼잡 제어를 하지 않음
  - 네트워크가 꼭 필요한 작업을 할 수 없게 되는 폭주 상태에 빠지는 것을 막지 못 함

UDP를 사용한 신뢰적인 데이터 전송

- UDP를 사용할 때도 신뢰적인 데이터 전송이 가능하다.
- 애플리케이션 자체에서 신뢰성을 제공한다면 가능
- 예: QUIC(Quic UDP Internet Connection) 프로토콜

UDP 세그먼트 구조



- 헤더: 2바이트씩 구성된 4개의 필드
- 데이터 필드

UDP checksum

- goal: 출발지→ 목적지로 이동할 때 오류(변경사항 검사) 검출

방법

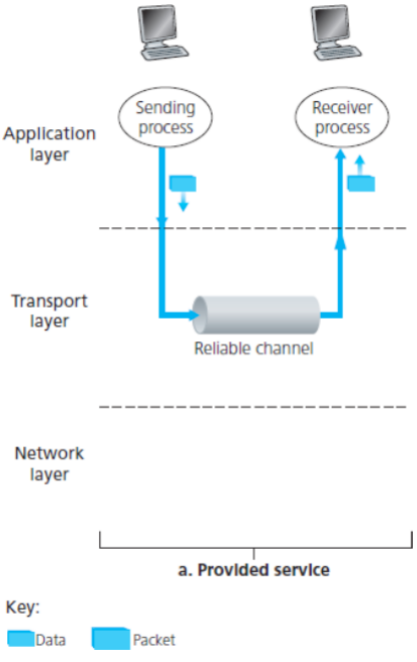
1. 송신자 측에서 세그먼트 안에 있는 모든 16비트 워드의 합산에 대해 1의 보수 수행
   오버플로는 윤회식 자리올림
2. 결과값이 UDP 세그먼트의 checksum 필드로
3. 수신자 측에서는 체크섬을 포함한 모든 16비트 워드들이 더해짐
4. 패킷에 어떤 오류도 없다면 수신자에서 합은 1111111111111111
   비트 중 0이 있으면 오류 발생한 것

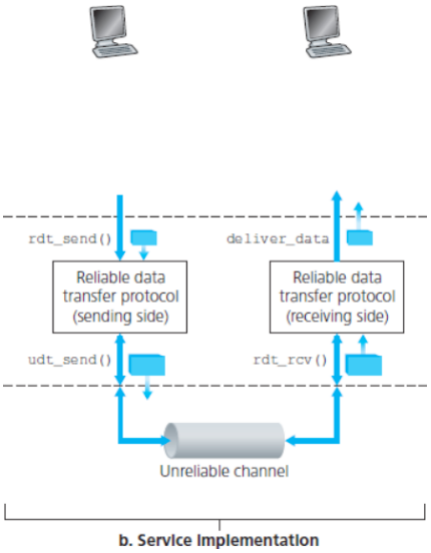
체크섬 제공 이유

- 출발지 ↔ 목적지에서 모든 링크가 오류 검사를 제공한다는 보장이 없기 때문
- UDP는 오류 회복은 하지 않음
  - 손상된 세그먼트 버리거나 경고와 함께 세그먼트를 애플리케이션으로 넘

3.4 신뢰적인 데이터 전송의 원리

- 신뢰적인 데이터 전송 연구의 프레임워크: 제공된 서비스와 서비스 구현





- 신뢰적인 채널에서는 데이터 손상 x, 순서 보장
- TCP가 인터넷 애플리케이션에게 제공하는 서비스 모델
- 신뢰적인 데이터 전송 프로토콜의 의무는 신뢰적인 채널의 서비스 추상화를 구현하는 것!
- 가정: 패킷은 순서대로 전달된다.

b. 데이터 전송 프로토콜의 인터페이스

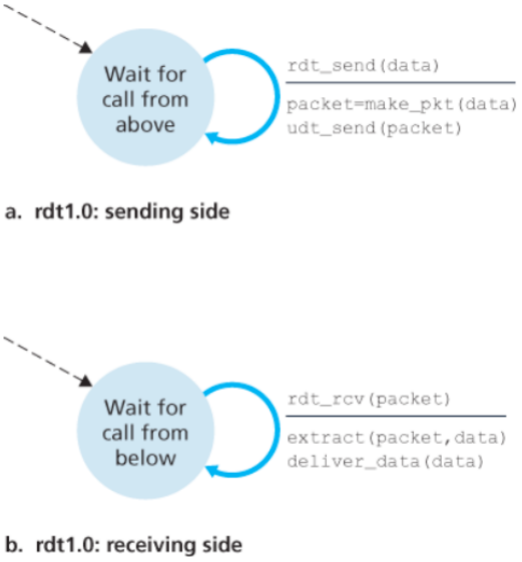
- rdt: 신뢰적인 데이터 전송 프로토콜(reliable data transfer)

- _send: 송신 측 호출되고 있음을 나타냄
- rdt_rcv(): 패킷이 수신 측에 도착했을 때
- deliver_data(): rdt 프로토콜이 상위 계층으로 데이터를 전달하려고 할 때
- 단방향 데이터 전송은 송신→수신만 고려

신뢰적인 데이터 전송 프로토콜의 구축

rdt1.0: 완벽하게 신뢰적인 채널상에서의 신뢰적인 데이터 전송

- 하위 채널이 완전히 신뢰적인 가장 간단한 경우



송신자와 수신자에 대한 분리된 FSM

송신자

1. rdt_send(data): 상위 계층으로부터 data 받음
2. make_pkt(data): data를 포함한 패킷 생성
3. udt_send(packet): 패킷을 채널로 송신

수신자

1. rdt_rcv(packet): 하위 채널로부터 패킷 수신
2. extract(packet, data): 패킷으로부터 데이터 추출
3. deliver_data(data): 상위 계층으로 데이터 전달

특징

- 데이터 단위와 패킷의 차이가 없음
- 오류가 생길 수 없는 상황이므로 수신 측이 송신 측에게 어떤 피드백도 제공할 필요가 없다.
- 수신자는 송신자가 데이터를 송신하자마자 수신할 수 있다고 가정해야 함

rdt2.0: 비트 오류가 있는 채널상에서의 신뢰적인 데이터 전송

- 패킷 안의 비트들이 하위 채널에서 손상
  - 일반적으로 패킷이 전송되는 네트워크의 물리적인 구성 요소에서 발생함

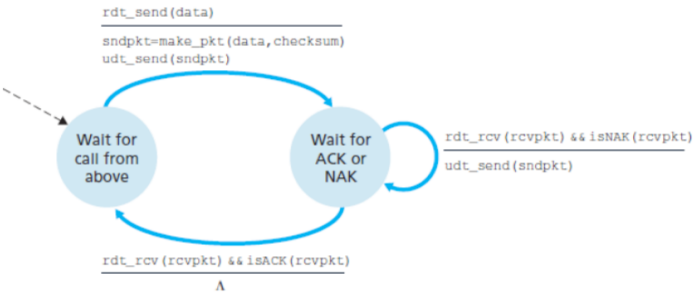
자동 재전송 요구(Automatic Repeat reQuest, ARQ) 프로토콜

- 긍정 확인응답(positive acknowledgment, “OK”)
- 부정 확인응답(negative acknowledgment, “그것을 반복해주세요”)

비트 오류를 처리하기 위한 세 가지 부가 프로토콜 기능

1. 오류 검출
2. 수신자 피드백
3. 재전송

송신자

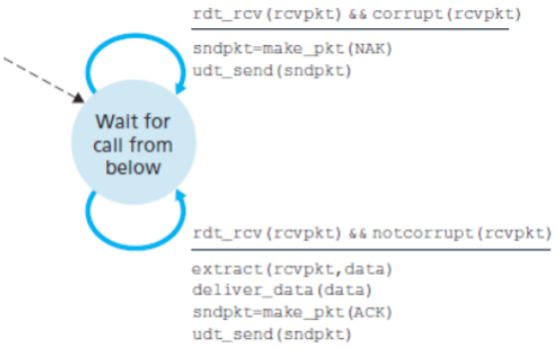


a. rdt2.0: sending side

상태

1. 상위 계층으로부터 데이터 전달되기를 기다리는 중
   a. 체크섬과 함께 전송될 데이터를 포함하는 패킷 생성
   b. udt_send(sndpkt) 통해 전송
2. 수신자로부터 ACK 또는 NAK 기다리는 중
   a. ACK이 수신되면 다시 데이터 기다리는 상태로 감
   b. NAK이 수신되면 마지막 패킷을 재전송하고, 다시 기다리는 상태가 됨

수신자



b. rdt2.0: receiving side

- 패킷이 도착했을 때, 패킷이 손상되었으면 NAK, 아니면 ACK로 응

특징

- 기다리는 상태에서 데이터를 받을 수 없음 → 그래서 전송 후 대기(stop-and-wait) 프로토콜
- 송신자는 수신자가 현재의 패킷을 정확하게 수신했음을 확신하기 전까지 새로운 데이터를 전달하지 않음

결함

- ACK, NAK도 손상될 수 있음을 고려 x
- 대안 1: 비트 오류로부터 회복할 수 있도록 충분한 체크섬 비트들 추가
- 대안 2: ACK, NAK 수신할 때 현재 데이터 패킷을 단순히 다시 송신
  - 중복 패킷

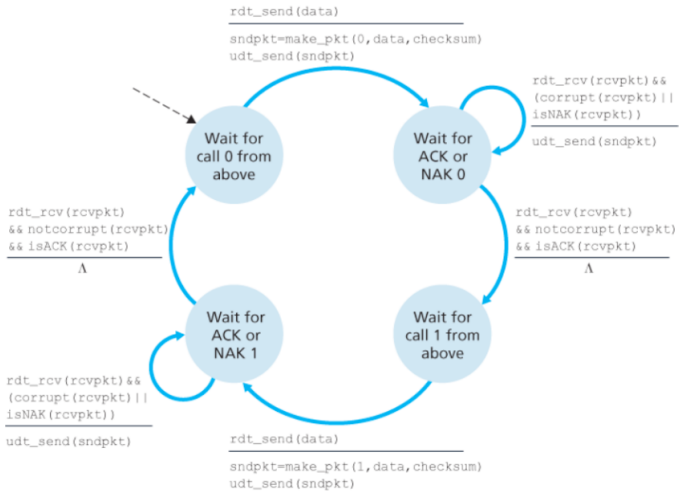
해결책

데이터 패킷에 순서 번호(송신자가 번호 붙임)를 삽입하는 새로운 필드 추가

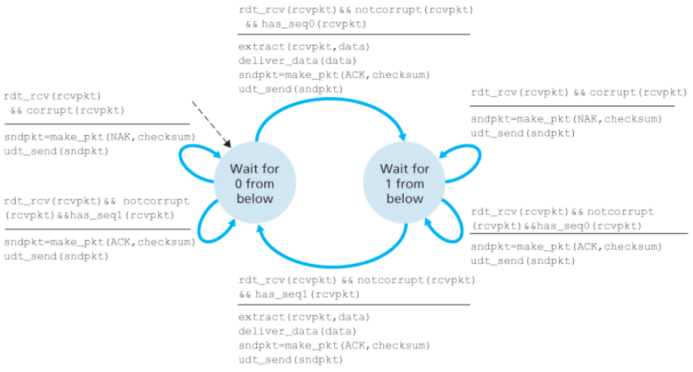
rdt2.1: rdt2.0의 수정 버전

- 2.0에 비해 상태가 두 배
  1. 프로토콜 상태가 현재 송신자에 의해 전송되고 있는지를 반
  2. 수신자가 기다리고 있는 패킷이 순서 번호 0 또는 1을 가져야 하는지를 반영
- 긍정 확인응답과 부정 확인응답을 포함
  - 순서가 바뀐 패킷이 수신되면, 수신자는 이미 전에 수신한 패킷에 대한 긍정 확인응답 전송
  - 손상된 패킷이 수신되면, 수신자는 부정 확인응답을 전송
- NAK 대신 가장 최근에 정확하게 수신된 패킷에 대해 ACK 송신
  - 같은 패킷에 대해 2개의 ACK를 받은 송신자는 수신자가 두 번 ACK한 패킷의 다음 패킷을 수신하지 못한 것을 알게 된다.

송신자



수신자

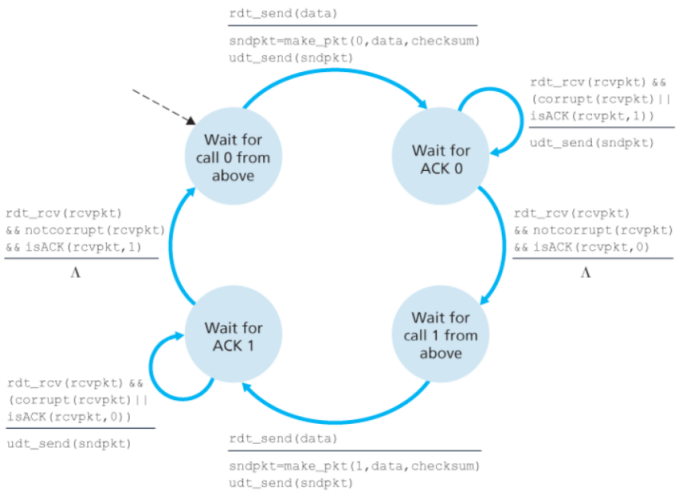


rdt2.2: 비트 오류를 갖는 채널을 위한 NAK 없는 신뢰적인 데이터 전송 프로토콜

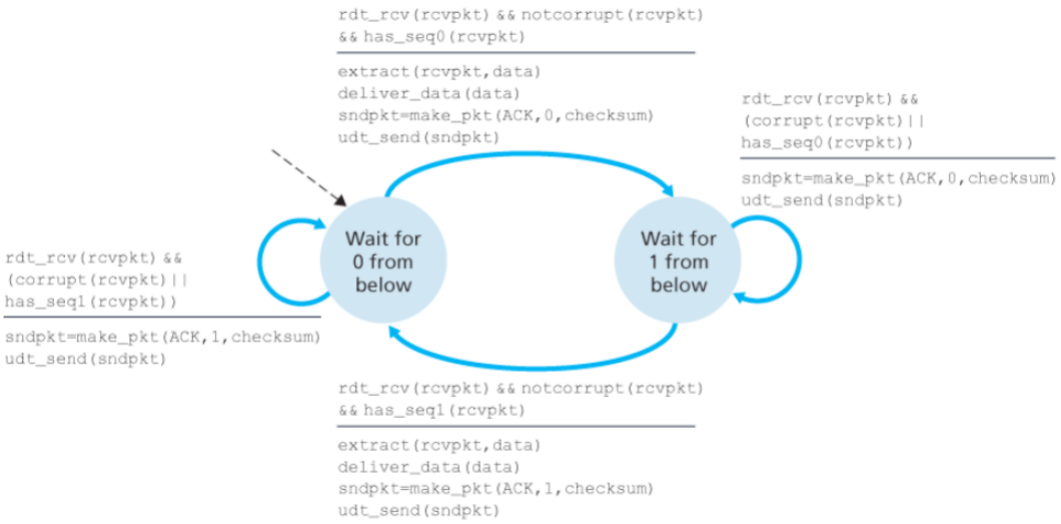
rdt2.1과 다르게

1. 수신자가 반드시 ACK 메시지에 의해 확인 응답되는 패킷의 순서 번호를 포함
    a. make_pkt(ACK, 0) or make_pkt(ACK, 1)
2. 송신자는 수신된 ACK 메시지에 의해 확인 응답된 패킷의 순서 번호를 반드시 검사
    a. isACK(0 or 1)

송신자



수신자

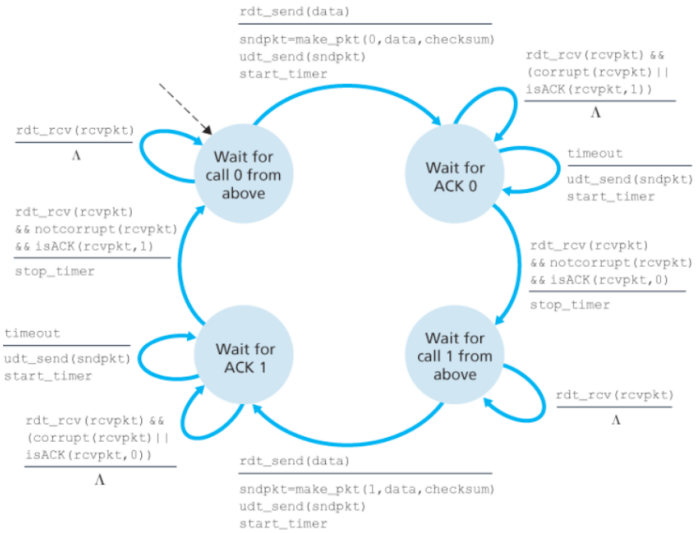


rdt3.0: 비트 오류와 손실 있는 채널상에서의 신뢰적인 데이터 전송

- 하위 채널이 패킷을 손실하는 경우
  1. 어떻게 패킷 손실을 검출할 것인가?
  2. 패킷 손실이 발생했을 때 어떤 행동을 할 것인가?
- 송신자에게 손실된 패킷의 검출과 회복에 대한 책임을 부여
- 송신자가 패킷을 잃어버렸다고 확실한 정도로 충분한 시간을 기다림
  - 충분한 시간: 송신자와 수신자 사이의 왕복 시간 지연(중간 라우터 버퍼링 포함) + 수신 측에서 패킷을 처리하는 데 필요한 시간
  - 시간 안에 수신되지 않으면 패킷은 재전송된다.
- 주어진 시간이 지난 후에 송신자를 인터럽트할 수 있는 카운트다운 타이머가 필요

1. 매 패킷이 송신된 시간에 타이머 시작
2. 타이머 인터럽트에 반응
3. 타이머 멈춤

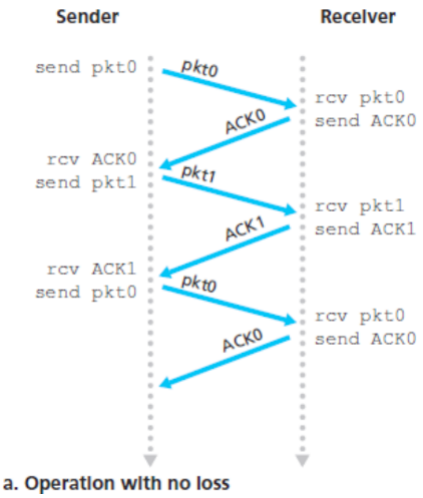
송신자



동작과 처리 과정

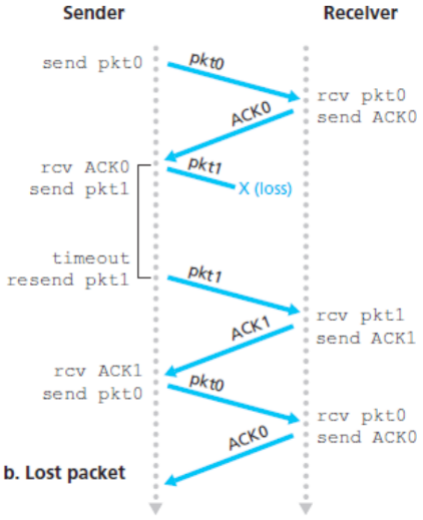
- 패킷에 대한 수신 시간은 전송 지연 + 전파 지연 때문에 패킷 전송 시간보다 크다.
- 패킷의 순서 번호과 0, 1 번걸아 일어나기 때문에 rdt3.0은 얼터네이팅 비트 프로토콜(alternating-bit protocol)이라고 불림

(a) 무손실 동작



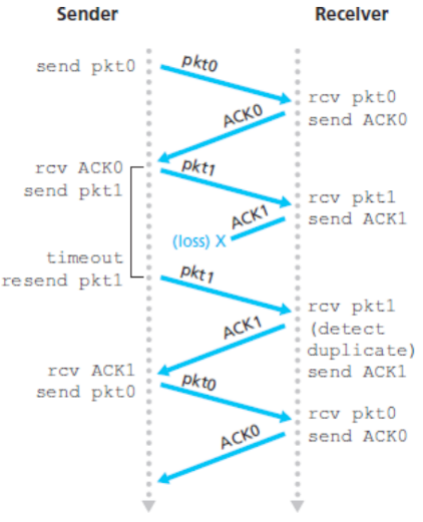
a. Operation with no loss

(b) 패킷 손실



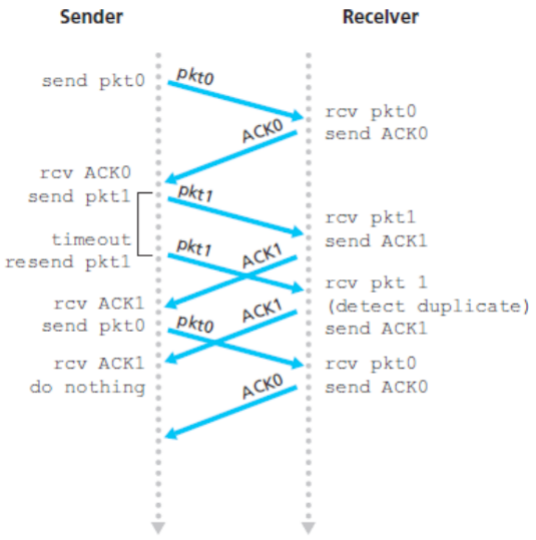
b. Lost packet

(c) ACK 손실



c. Lost ACK

(d) 조급한 타임아웃



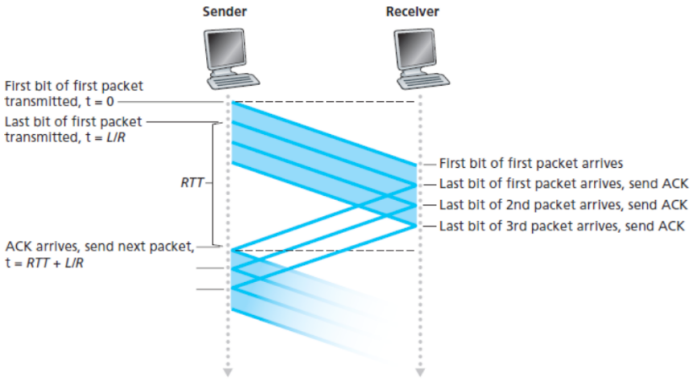
d. Premature timeout

파이프라이닝된 신뢰적인 데이터 전송 프로토콜

- rdt3.0의 핵심적인 성능 문제: 전송 후 대기 프로토콜이라는 점, 송신자 이용률이 형편 없다.

→ 오늘날 고속 네트워크에서 만족할 수 없는 성능

- 해결책: 송신자에게 확인 응답을 기다리기 전에 송신을 전송하도록 허용, 파이프라이닝

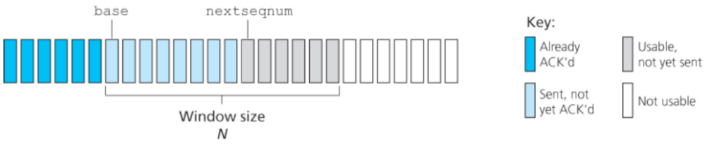


b. Pipelined operation

- 순서 번호의 범위가 커져야 함
  - 전송 중인 패킷은 유일한 순서 번호를 가져야 함
- 프로토콜의 송신 측과 수신 측은 패킷 하나 이상을 버퍼링해야 함
- 필요한 순서 번호의 범위와 버퍼링 조건은 데이터 전송 프로토콜이 손실 패킷, 손상 패킷, 지연 패킷에 대해 응답하는 방식에 달려 있음

GBN

- Go-Back-N, N부터 반복
- 송신자는 확인 응답을 기다리지 않고 여러 패킷을 보낼 수 있다.
- 파이프라인에서 확인 응답 안 된 패킷의 최대 허용 수가 N보다 크면 안 됨

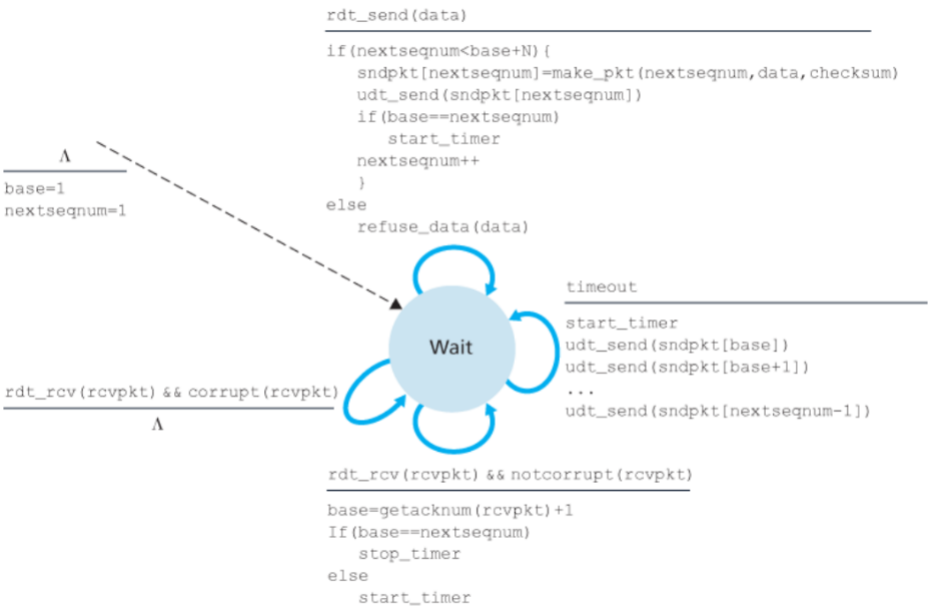


- 슬라이딩 윈도우 프로토콜
- 패킷의 순서 번호: 패킷 헤더 안의 고정된 길의 필드에 포함
  - 패킷 순서 번호 필드의 비트 수가 k라면 [0, 2^k - 1]
  - 모듈로 연산

ACK 기반의 NAK 없는 확장된 FSM

- base와 nextseqnum 추가

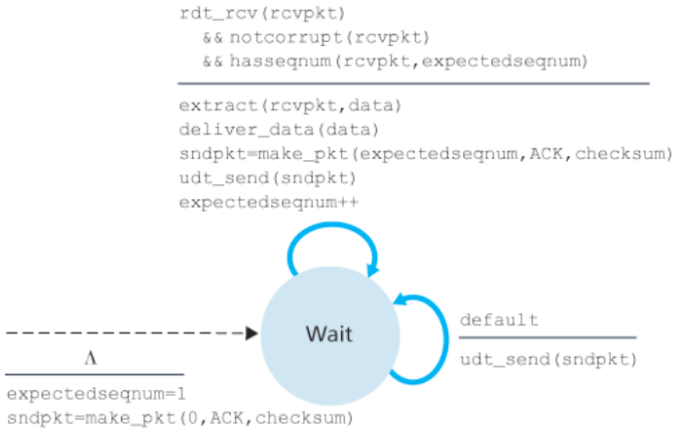
송신자



- 송신자가 반응해야 할 세 가지 타입의 이벤트

1. 상위로부터의 호출
2. ACK의 수신
3. 타임아웃 이벤트

수신자



- 패킷이 오류 없이 순서대로 수신되면 ACK 보내고 상위 계층에 패킷의 데이터를 전달
- 그 외의 경우에는 수신자가 그 패킷을 버리고 가장 최근에 제대로 수신된 순서의 패킷에 대한 ACK를 재전송

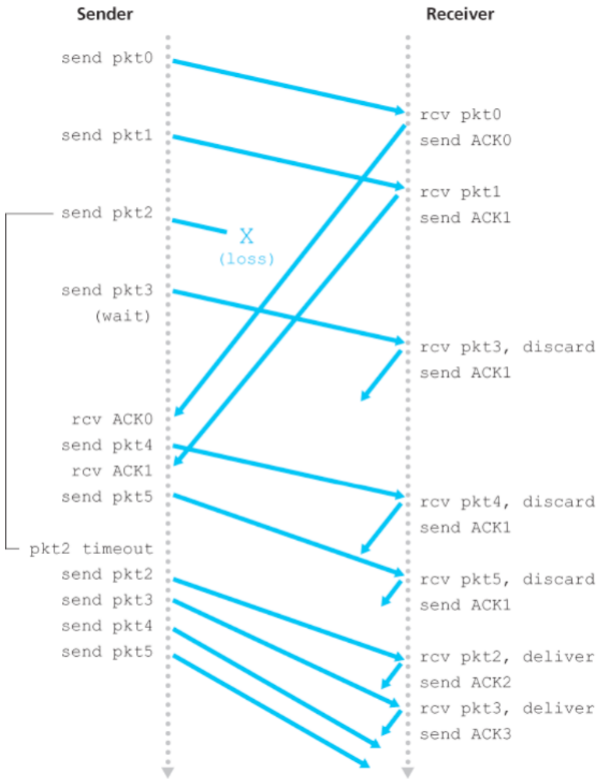
GBN 프로토콜에서 수신자는 순서가 잘못된 패킷들을 버린다.

패킷 n이 수신되어야 할 때 패킷 n+1이 먼저 도착했다고 가정

수신자는 패킷 n+1을 저장하고 나중에 패킷 n이 수신되면 차례대로 전달

그러나 패킷 n이 손실된다면 송신자에게 GBN 재전송 규칙에 따라 패킷 n, n+1이 모두 재전송

- 윈도 크기 4 패킷인 경우의 GBN 프로토콜 동작



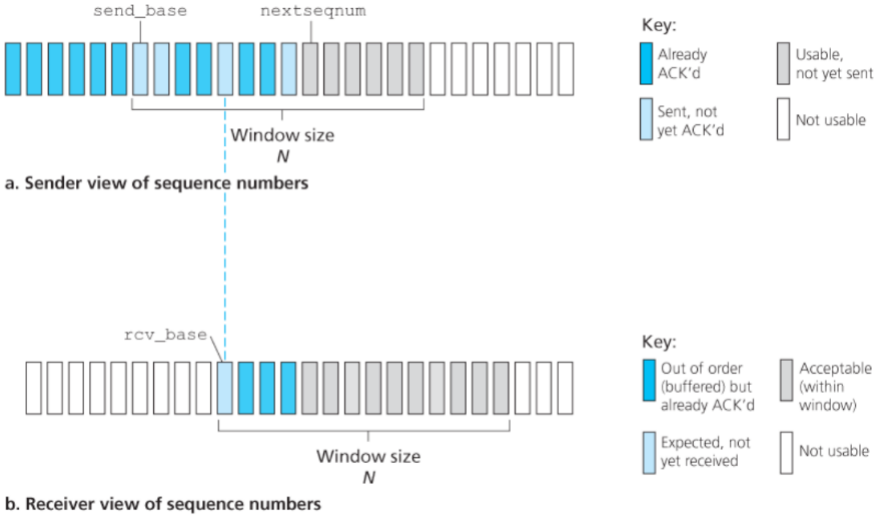
성능 문제

GBN 자체 성능 문제:

- 윈도 크기와 대역폭 지연 곱의 결과가 클 때, 많은 패킷이 파이프라인에 존재
- 하나의 오류 때문에 많은 패킷을 재전송해야되는데 불필요하게 많은 패킷을 전송해야 함

SR

- Selective Repeat, 선택적 반복
- 수신자에서 오류가 발생한 패킷을 수신했다고 의심되는 패킷만 재전송
  - 불필요한 재전송 피함
  - 패킷에 대하 개별적인 확인응답 요구
- 윈도 크기 N: 아직 확인응답이 안 된 패킷 수를 제한하는 데 사용



1. 순서가 바뀐 패킷은 빠진 패킷이 수신될 때까지 버퍼에 저장하고
2. 빠진 패킷이 수신된 시점에서 일련의 패킷을 순서대로 상위 계층에 전달

SR 송신자 이벤트와 행동

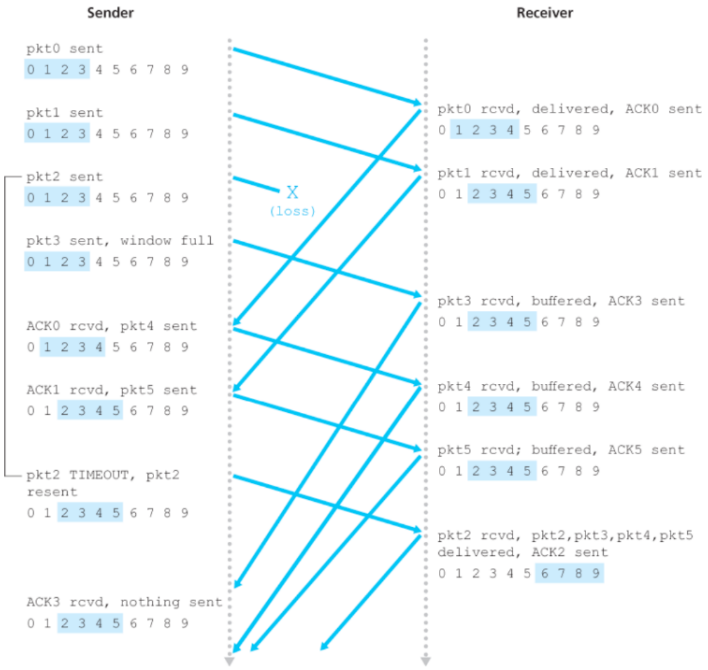
1. 상위로부터 데이터 수신

2. 타임아웃
3. ACK 수신

SR 수신자 이벤트와 행동

1. [rcv_base, rcv_base+N-1] 내의 순서 번호를 가진 패킷이 손상 없이 수신된다.
2. [rcv_base-N, rcv_base-1] 내의 순서 번호를 가진 패킷이 수신된다.
3. 그 외 패킷 무시

SR 동작



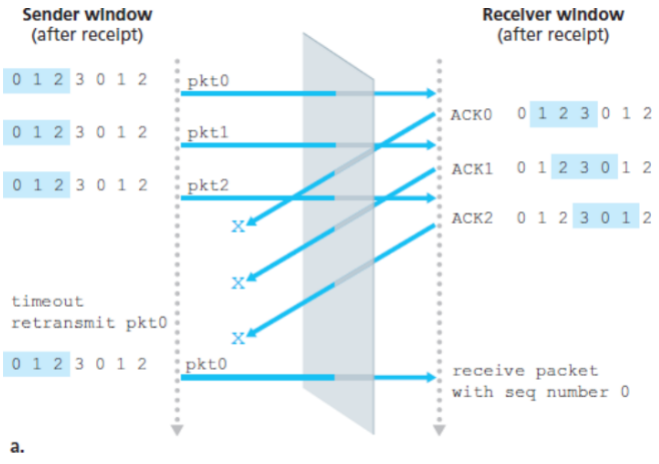
송신자와 수신자의 윈도는 항상 같지 않다

송신자가 보낸 패킷에 대한 응답이 없으면 송신자의 윈도는 절대 앞으로 안 감

가정

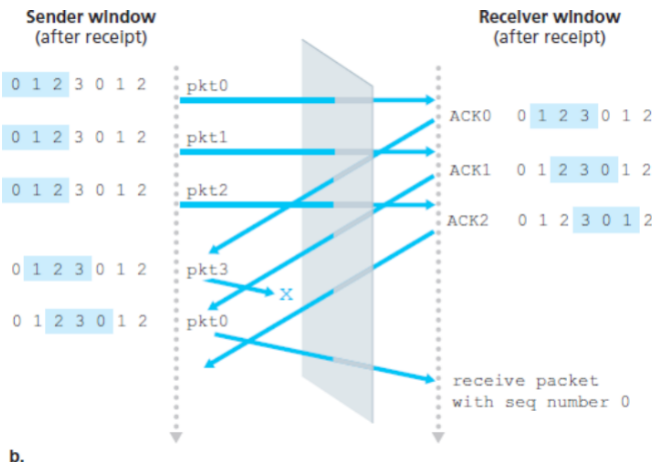
패킷 순서 번호 0, 1, 2, 3
윈도 크기 3
0 ~ 2는 올바르게 수신, 확인 완료
수신자 윈도: 순서 번호가 각각 3, 0, 1인 패킷

첫 번째 시나리오



1. 처음 3개 패킷에 대한 ACK 손실, 송신자는 이 패킷 재전송
2. 수신자는 순서 번호가 0인 패킷을 수신

두 번째 시나리오



b.

1. 처음 3개 ACK까지 올바르게 전달됨
2. 송신자는 자신의 윈도를 앞으로 이동시켜, 각각의 순서 번호가 3, 0, 1인 4, 5, 6번재 패킷을 보냄
3. 순서 번호 3을 가진 패킷이 손실되고, 순서 번호 0을 가진 패킷(**새로운** 데이터를 포함한 패킷)은 도착

수신자 관점

수신자는 송신자의 행동을 볼 수 없다.

다섯 번째 패킷의 원래 전송과 첫 번째 패킷의 재전송을 **구별할 방법은 없다.**

최소한의 윈도 크기

SR 프로토콜에 대한 순서 번호 공간 크기의 절반보다 작거나 같아야 한다.

패킷 순서 바뀜 현상

패킷들은 송신자와 수신자 사이의 채널 안에서 순서가 바뀔 수 없다.

→ 일반적으로 송신자와 수신자가 단일한 물리적 선으로 연결되어 있을 때 적합한 가정이다.

하지만 둘을 연결하는 '채널'이 네트워크일 때는 패킷 순서 바뀜이 일어날 수 있다.

신뢰적인 데이터 전송 메커니즘과 그 사용에 대한 요약

메커니즘	사용 설명
체크섬	전송된 패킷 안의 비트 오류를 발견하는 데 사용된다.
타이머	채널 안에서 패킷이 손실되었기 때문에 발생되는 패킷(또는 이것의 ACK)의 타임아웃/재전송에 사용된다. 타임아웃은 패킷이 지연되었지만 손실되지는 않았을 때(조기 타임아웃), 또는 패킷이 수신자에 의해 수신되었으나 수신자에서 송신자로의 ACK가 손실되었을 때 발생하기 때문에, 수신자에 의해 수신되는 패킷은 중복으로 복사(수신)된 패킷일 수 있다.
순서 번호	송신자에서 수신자로 가는 데이터 패킷의 순서 번호를 붙이기 위해 사용된다. 수신자 패킷의 순서 번호의 간격은 수신자가 손실된 패킷을 검사하게 한다. 중복된 순서 번호를 갖는 패킷은 수신자가 패킷의 중복 복사를 검사하게 한다.
확인응답	수신자가 한 패킷 또는 패킷 집합이 정확히 수신되었다는 응답을 송신자에게 하기 위해 사용된다. 확인응답은 일반적으로 패킷 또는 이미 확인응답된 패킷들의 순서 번호를 전달한다. 확인응답은 프로토콜에 따라, 개별적이거나 누적된 것일 수 있다.
부정 확인응답	수신자가 패킷이 정확히 수신되지 않았다는 응답을 송신자에게 하기 위해 사용된다. 부정 확인응답은 일반적으로 정확히 수신되지 않은 패킷의 순서 번호를 전달한다.
윈도, 파이프라이닝	송신자는 주어진 범위에 있는 순서 번호를 가진 패킷만 전송하도록 제한될 수 있다. 확인응답은 아직 없지만 허가된 패킷이 전송될 수 있으므로, 송신자의 이용률은 전송 후 대기 모드의 동작보다 증가할 수 있다. 우리는 곧 윈도 크기가 수신자의 메시지를 수신하고 버퍼링하는 능력, 네트워크의 혼잡 정도, 또는 두 가지 모두에 근거하여 설정된다는 것을 살펴볼 것이다.

표 3.1 신뢰적인 데이터 전송 메커니즘과 그 사용에 대한 요약