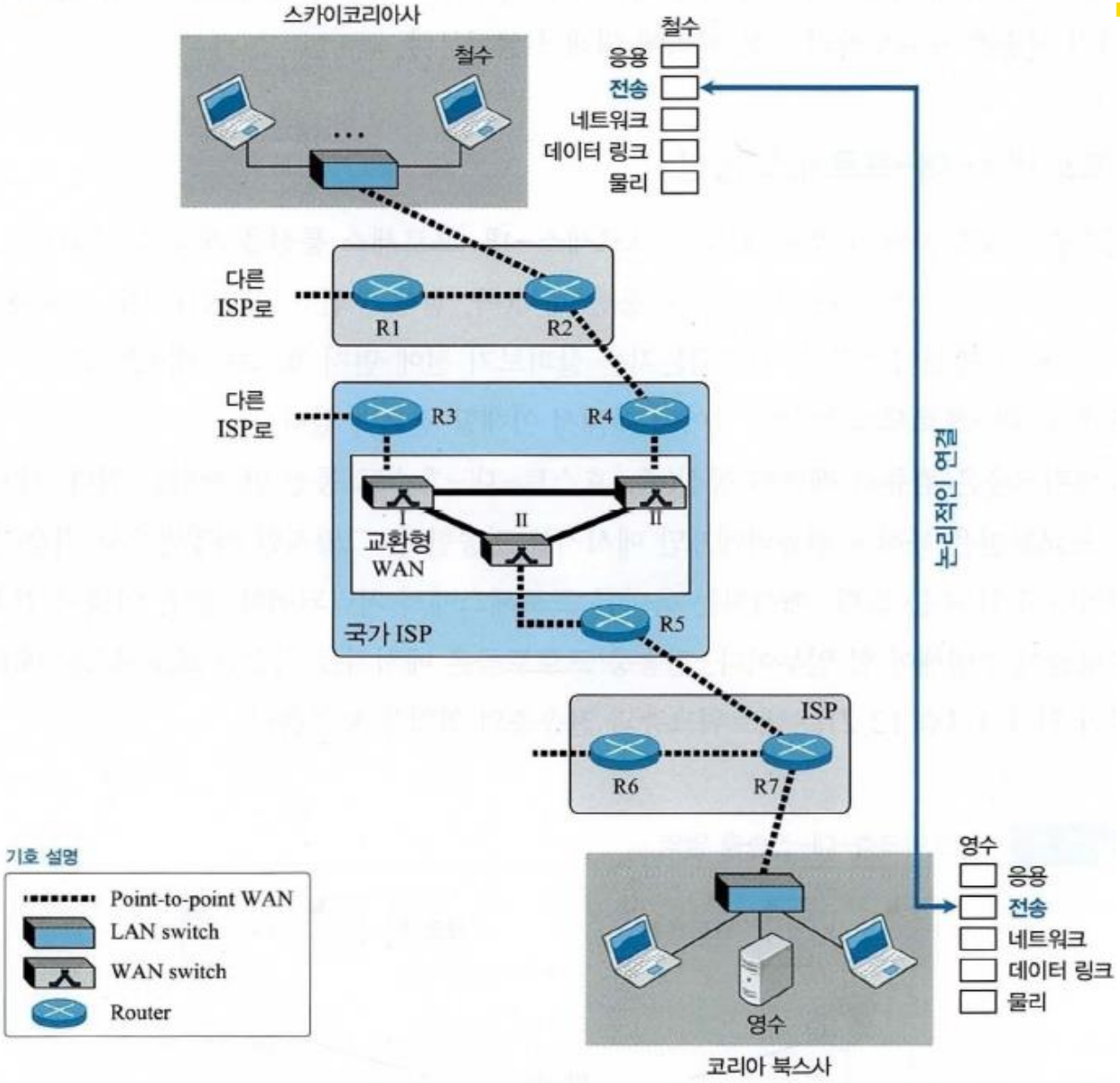
Chapter 10

전송층 개요

1. 개요

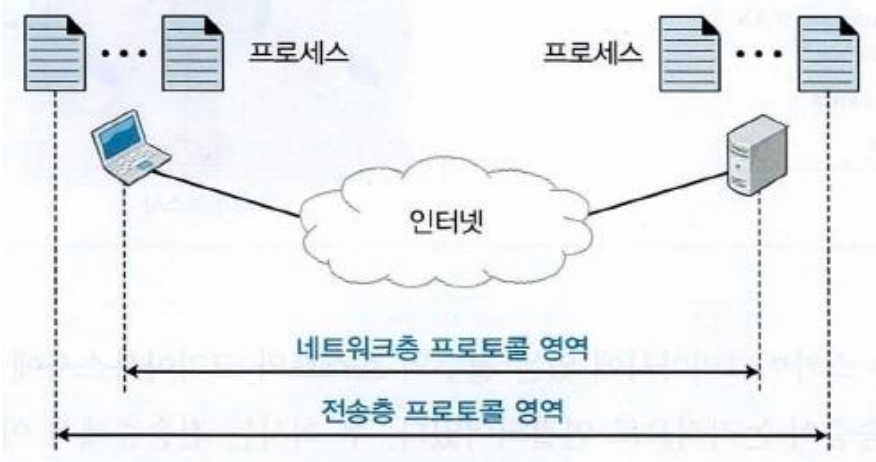
- 전송층은 네트워크층과 응용층 사이에 위치한다.
- 전송층은 두 응용층 사이에서 프로세스-대-프로세스 통신을 제공하는데, 하나는 로컬 호스트이고 다른 하나는 원격 호스트이다.
- 그림은 이러한 논리적인 연결의 개념을 보여준다.



2. 전송층 서비스

프로세스-대-프로세스 통신

- 전송층 프로토콜의 첫 번째 임무는 프로세스-대-프로세스 통신을 제공하는 것이다.
- 프로세스는 전송층 서비스를 이용하는 응용층 개체(즉 실행 중인 프로그램)이다.
- 네트워크층은 컴퓨터 레벨의 통신(즉, 호스트-대-호스트 통신)만 책임을 진다.
- 네트워크층 프로토콜은 목적지 컴퓨터에게만 메시지를 전송한다.
- 그렇지만 이것만으로 전송이 완료되었다고 할 수는 없다.
- 메시지는 올바른 프로세스에서 처리되어야 한다.
- 이것이 전송층 프로토콜이 수행해야 할 임무이다.
- 전송층 프로토콜은 메시지를 적절한 프로세스로 배달할 책임이 있다.
- 그림은 네트워크층과 전송층의 영역을 보여준다.



2. 전송층 서비스

주소 체계: 포트 번호

- 프로세스-대-프로세스 통신을 수행하기 위한 여러 가지 방법이 있을 수 있지만 가장 보편적인 방법은 클라이언트 서버 패러다임을 이용하는 것이다.
- 클라이언트(client) 라는 로컬 호스트에 있는 프로세스는 보통 서버(server)라는 원격 호스트에 있는 프로세스로부터 제공되는 서비스를 필요로 한다.
- 클라이언트와 서버 프로세스는 같은 이름을 가지고 있다.
- 그렇지만 오늘날의 운영체제는 다중 사용자와 다중 프로그래밍 환경을 제공한다.
- 원격 컴퓨터는 동시에 여러 개의 서버 프로그램을 실행할 수 있으며 마찬가지로 로컬 컴퓨터는 동시에 하나 이상의 클라이언트 프로그램을 실행할 수 있다.
- 따라서 통신을 위해서는 다음과 같은 사항이 정의되어야 한다.
  - 로컬 호스트(Local host)
  - 로컬 프로세스(Local process)
  - 원격 호스트(Remote host)
  - 원격 프로세스(Remote process)

2. 전송층 서비스

주소 체계: 포트 번호

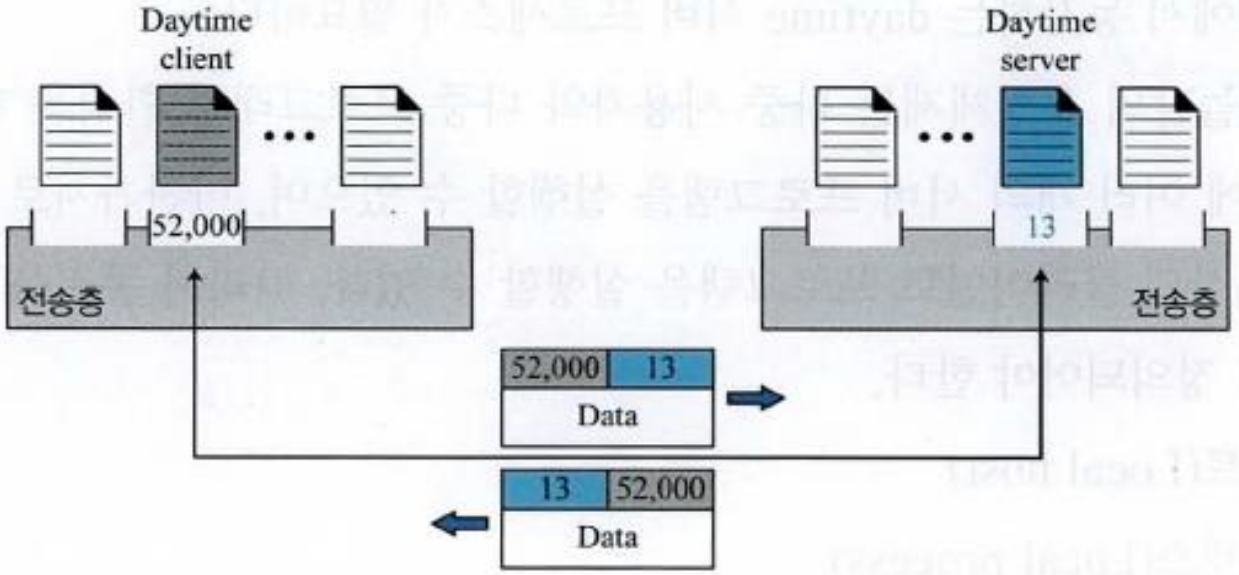
- 프로세스-대-프로세스 통신을 수행하기 위한 여러 가지 방법이 있을 수 있지만 가장 보편적인 방법은 클라이언트 서버 패러다임을 이용하는 것이다.
- 클라이언트(client) 라는 로컬 호스트에 있는 프로세스는 보통 서버(server)라는 원격 호스트에 있는 프로세스로부터 제공되는 서비스를 필요로 한다.
- 클라이언트와 서버 프로세스는 같은 이름을 가지고 있다.
- 그렇지만 오늘날의 운영체제는 다중 사용자와 다중 프로그래밍 환경을 제공한다.
- 원격 컴퓨터는 동시에 여러 개의 서버 프로그램을 실행할 수 있으며 마찬가지로 로컬 컴퓨터는 동시에 하나 이상의 클라이언트 프로그램을 실행할 수 있다.
- 따라서 통신을 위해서는 다음과 같은 사항이 정의되어야 한다.
  - 로컬 호스트(Local host)
  - 로컬 프로세스(Local process)
  - 원격 호스트(Remote host)
  - 원격 프로세스(Remote process)

2. 전송층 서비스

- 주소 체계: 포트 번호
  - 로컬 호스트와 원격 호스트는 IP 주소를 이용하여 정의된다.
  - 프로세스를 정의하기 위해서는 포트 번호(port number) 라는 두 번째의 식별자가 필요하다.
  - TCP/IP 프로토콜 모음에서 포트 번호는 0과 65,535 사이의 정수이다.
  - 클라이언트 프로그램은 임시 포트 번호(ephemeral port number) 라는 임의의 포트 번호로 자신을 지정한다.
  - 임시(ephemeral)라는 단어는 단명하다는 것을 의미하며 이것은 일반적으로 클라이언트의 수명이 짧기 때문이다.
  - 클라이언트-서버 프로그램이 원활히 동작하기 위해서 임시 포트 번호는 1,023보다 큰 값을 지정하도록 권장한다.
  - 서버 프로세스도 포트 번호로 자신을 정의하여야 한다.
  - TCP/IP는 서버를 위해서 범용 포트 변호를 사용하고, 이를 잘 알려진 포트 번호(well-known port number) 라고 한다.

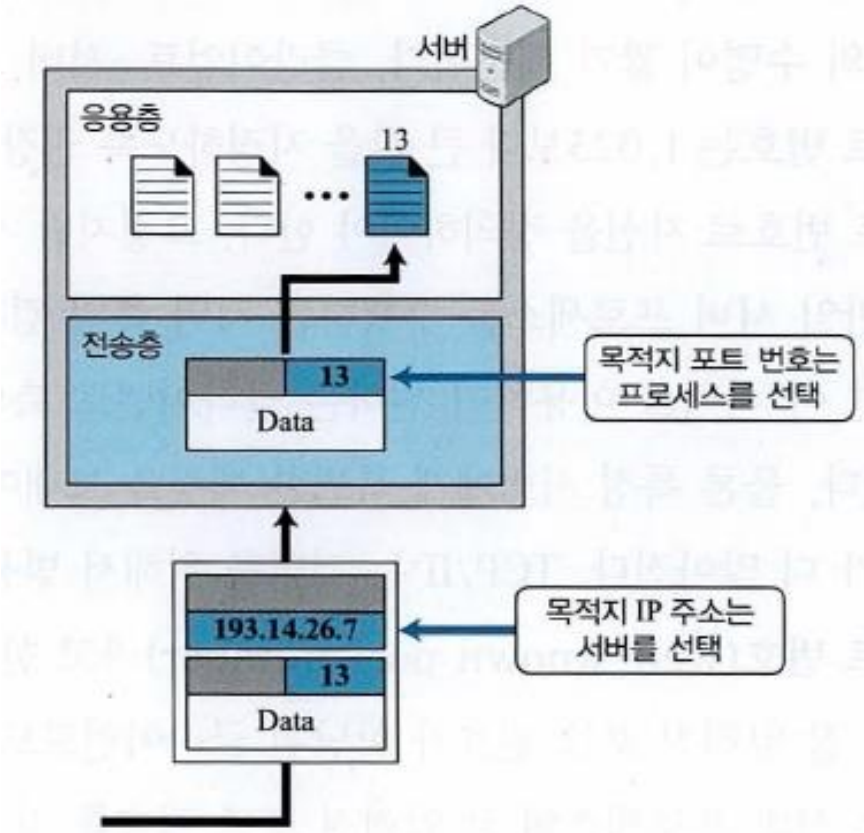
2. 전송층 서비스

- 주소 체계: 포트 번호
  - 그림은 이러한 개념을 보여준다.



2. 전송층 서비스

- 주소 체계: 포트 번호
  - 목적지 IP 주소는 전 세계의 호스트 중에서 특정 호스트를 정의하기 위하여 사용된다.
  - 특정 호스트가 선택된 후에는 포트 번호는 이 특정 호스트 내에 있는 여러 프로세스 중에서 하나의 프로세스를 정의한다.

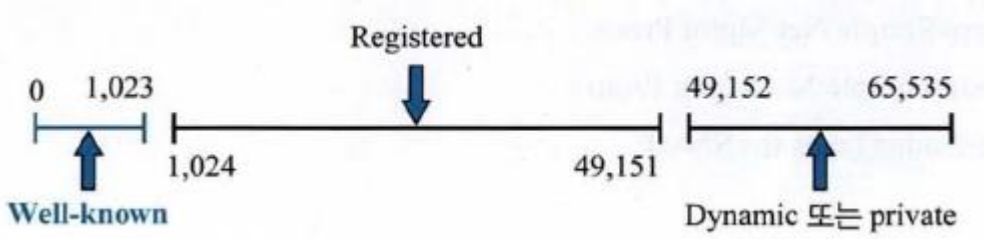


2. 전송층 서비스

주소 체계: 포트 번호

ICANN 범위

- ICANN은 그림에 나타나 있는 것과 같이 포트 번호를 잘 알려진, 등록된, 동적(또는 사설)의 세 범위로 나누었다.



- 잘 알려진 포트 번호(well-known port number).
  - 0과 1,023 범위의 포트 번호는 ICANN에 의해 배정되고 제어된다.
  - 이것은 잘 알려진 포트(well-known ports) 이다.

- 등록된 포트 번호(registered port number).
  - 1,024에서 49,151 사이의 포트 번호는 ICANN에 의해서 배정되거나 제어되지 않는다.
  - 그러나 중복을 피하기 위해 ICANN에 등록(registered) 될 수 있다.

2. 전송층 서비스

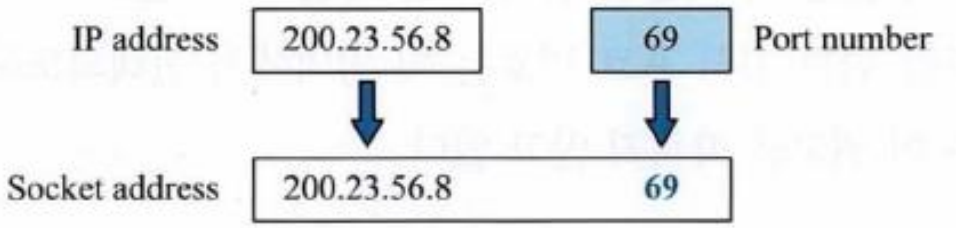
주소 체계: 포트 번호

ICANN 범위

- 동적 포트 번호(dynamic port number).
  - 49,152와 65,535 사이의 포트 번호는 제어되거나 등록되지 않는다.
  - 이들은 임시 또는 사설 포트 변호로 사용될 수 있다.
  - 처음에는 클라이언트를 위한 임시 포트 번호는 이 영역에서 선택되도록 권고되었지만, 대부분의 시스템은 이 권고를 따르지 않고 있다.

소켓 주소

- TCP/IP 모음의 전송층 프로토콜은 연결을 설정하기 위하여 각 종단마다 IP 주소와 포트 변호가 필요하다.
- IP 주소와 포트 변호의 조합을 소켓 주소(Socket address) 라고 한다.
- 클라이언트 소켓 주소는 클라이언트 프로세스를 유일하게 정의하고 서버 소켓 주소는 서버 프로세스를 유일하게 정의한다.



2. 전송층 서비스

캡슐화와 역캡슐화

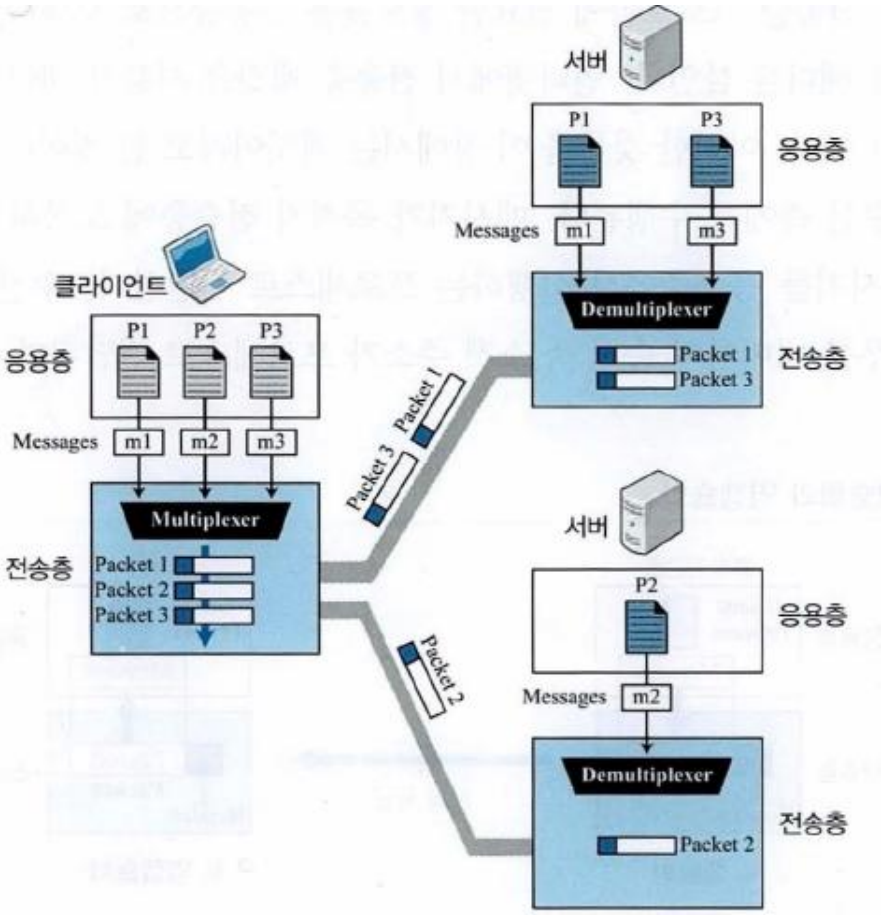
- 한 프로세스에서 다른 프로세스로 메시지를 전송하기 위하여 전송층 프로토콜은 메시지를 캡슐화(encapsulation) 하고 역캡슐화(decapsulation)한다.

- 캡슐회는 전송 측에서 수행된다.
- 전송할 메시지가 있는 프로세스는 메시지와 한 쌍의 소켓 주소, 그리고 전송층 프로토콜에 필요한 정보들을 전송층으로 보낸다.
- 역캡슐화는 수신 측에서 수행된다.
- 메시지가 목적지 전송층에 도착하면, 전송층은 헤더를 제거하고 메시지를 응용층에서 실행하는 프로세스로 전달한다.

2. 전송층 서비스

다중화와 역다중화

- 개체가 여러 발신지로부터 정보를 수신하는 경우를 다중화(multiplexing) 라고 하며, 개체가 여러 목적지로 정보를 전달하는 경우를 역다중화(demultiplexing) 라고 한다.
- 그림은 하나의 클라이언트와 두 개의 서버 간의 통신을 보여준다.

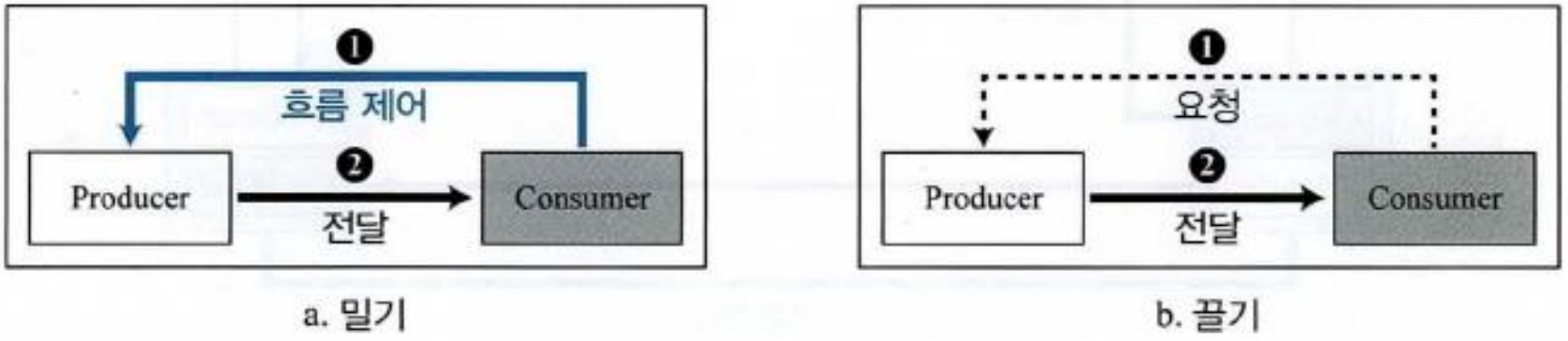


2. 전송층 서비스

흐름 제어

밀기와 끌기

- 정보를 생산자(producer)에서 소비자(consumer)로 전달하는 방법은 밀기 (push)와 끌기(pull)라는 두 가장 방법 중 하나로 이루어질 수 있다.
- 소비자의 요청 없이 정보가 생성될 때마다 전송 측에서 정보를 전달하는 경우를 밀기(pushing) 라고 한다.
- 반면 소비자가 요청한 경우에만 생산자가 정보를 전달하는 경우를 끌기(pulling) 라고 한다.
- 그림은 이러한 두 가지 형태의 전달을 보여준다.



- 생산자가 밀기 방식으로 정보를 전달하면 소비자는 정보의 과도한 수신에 직면할 수 있고 따라서 정보의 손실을 방지하기 위하여 반대 방향에 대한 흐름 제어가 필요하게 된다.

2. 전송층 서비스

흐름 제어

전송층에서 흐름 제어

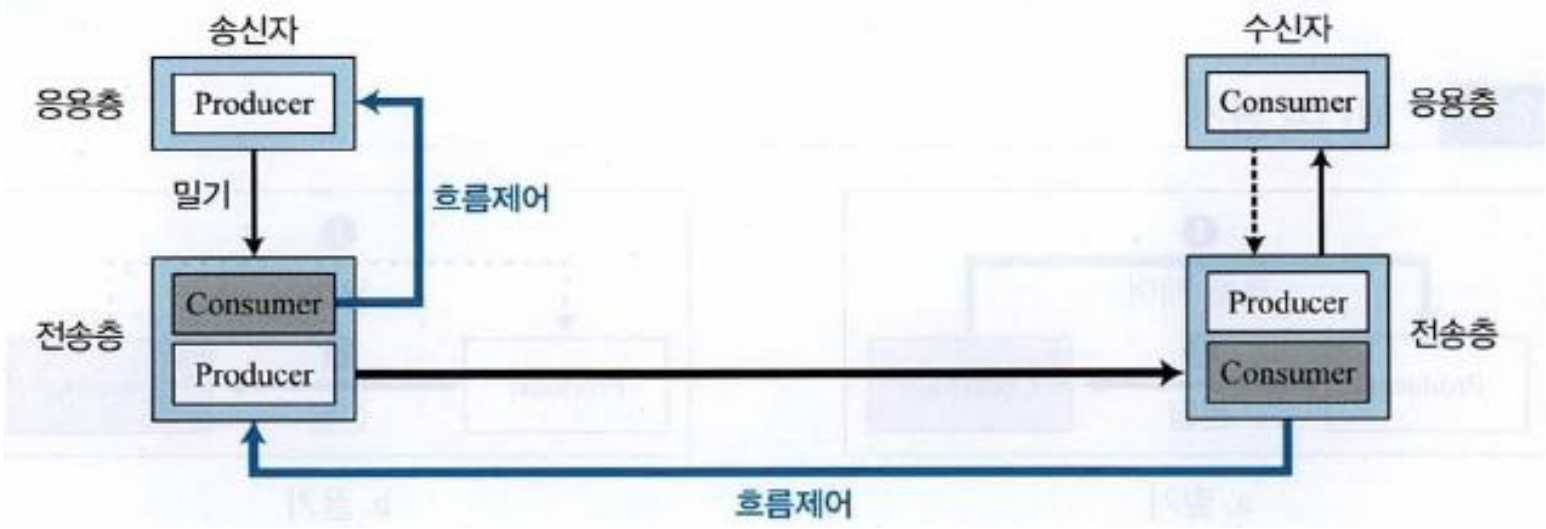
- 전송층 통신을 위하여 송신 프로세스 송신 전송층, 수신 전송층, 그리고 수신 프로세스라는 4개의 개체가 필요하다.
- 응용층의 송신 프로세스는 단순한 생산자이다.
- 송신 프로세스는 메시지를 생산하고 밀기 방식을 이용하여 전송층으로 전송한다.
- 송신 전송층은 소비자와 생산자라는 두 가지 임무를 수행한다.
- 송신 전송층은 생산자가 전송한 메시지를 소비한다.
- 수신 전송층은 송신측이 전송한 패킷을 소비한다.
- 수신 전송층은 또한 생산자 역할을 수행한다.
- 그렇지만 마지막 전달 방법은 끌기(pulling) 방식으로 이루어진다.
- 즉, 전송층은 응용층 프로세스가 메시지를 요청하기 전까지는 기다린다.

2. 전송층 서비스

흐름 제어

전송층에서 흐름 제어

- 그림은 송신 전송층에서 송신 응용층에 대해 그리고 수신 전송층에서 송신 전송층에 대해 흐름 제어가 필요하다는 것을 보여준다.



2. 전송층 서비스

흐름 제어

버퍼

- 비록 여러 가지 방법으로 흐름 제어가 구현될 수 있지만, 한 가지 방법은 일반적으로 송신 전송층에 하나, 그리고 수신 전송층에 또 다른 하나 등 두 개의 버퍼 (buffer)를 이용하는 것이다.
- 수신 전송층의 버퍼가 가득 차면, 수신 전송층은 패킷의 전송을 멈추도록 송신 전송층에게 알린다.
- 버퍼에 빈 공간이 생기면, 수신 전송층은 송신 전송층에게 다시 전송해도 된다고 메시지를 보낸다.

2. 전송층 서비스

오류 제어

- **인터넷에서는 송신 전송층에서 수신 전송층으로 패킷을 전달할 책임이 있는 네트워크층(즉, IP)이 신뢰성을 제공하지 않기 때문에, 응용층에서 신뢰성을 요구하는 경우에는 전송층에서 신뢰성을 제공할 수 있어야 한다.**
- **신뢰성은 전송층에 오류 제어(error control) 서비스를 추가함으로서 제공할 수 있다.**
- **전송층의 오류 제어에서 제공하는 기능은 다음과 같다.**
  - 훼손된 패킷의 감지 및 폐기
  - 손실되거나 제거된 패킷을 추적하고 재전송
  - 중복 수신 패킷을 확인하고 폐기
  - 손실된 패킷이 도착할 때까지 순서에 어긋나게 들어온 패킷을 버퍼에 저장

- **그림은 송신과 수신 전송층 간의 오류 제어를 보여준다.**

2. 전송층 서비스

오류 제어

순서 번호

- 오류 제어가 수행되기 위해서 송신 전송층은 어떤 패킷이 재전송되어야 하는지를 알아야 하며, 수신 전송층은 어떤 패킷이 중복되었는지 또는 어떤 패킷이 순서에 어긋나게 도착했는지를 알아야 한다.
- 이것은 패킷이 변호를 가지고 있으면 기능하다.
- 이는 패킷의 순서 번호(sequence number)를 저장할 수 있도록 전송층 패킷에 한 필드를 추가하는 것이다.
- 패킷에는 순서 번호가 매겨진다.
- 그렇지만 헤더 안에 각 패킷의 순서 변호를 포함하기 위해서는 최대값이 설정되어야 한다.
- 만일 패킷의 헤더에 순서 번호를 위해서 m비트가 설정된다면, 순서 번호는 0에서 2^m - 1 값의 범위를 가진다.

2. 전송층 서비스

오류 제어

확인 응답

- 오류 제어를 위하여 긍정과 부정 신호를 모두 사용할 수 있다.
- 수신 측에서는 오류 없이 잘 수신된 패킷들에 대해서 확인응답(ACK)을 전송할 수 있다.
- 수신 측은 훼손된 패킷을 단순히 버린다.
- 송신 측에서 타이머를 사용하면 패킷의 손실을 감지할 수 있다.
- 패킷을 전송한 후 송신 측은 타이머를 구동하고, 타이머가 만료되기 전까지 ACK가 도착하지 않으면 송신 측은 패킷을 재전송한다.
- 수신 측에서는 중복 수신된 패킷은 바로 폐기한다.
- 순서에 어긋나게 들어온 패킷은 폐기되거나(송신 측에서는 손실 패킷으로 간주함) 또는 손실된 패킷이 도착하기 전까지 버퍼에 저장된다.

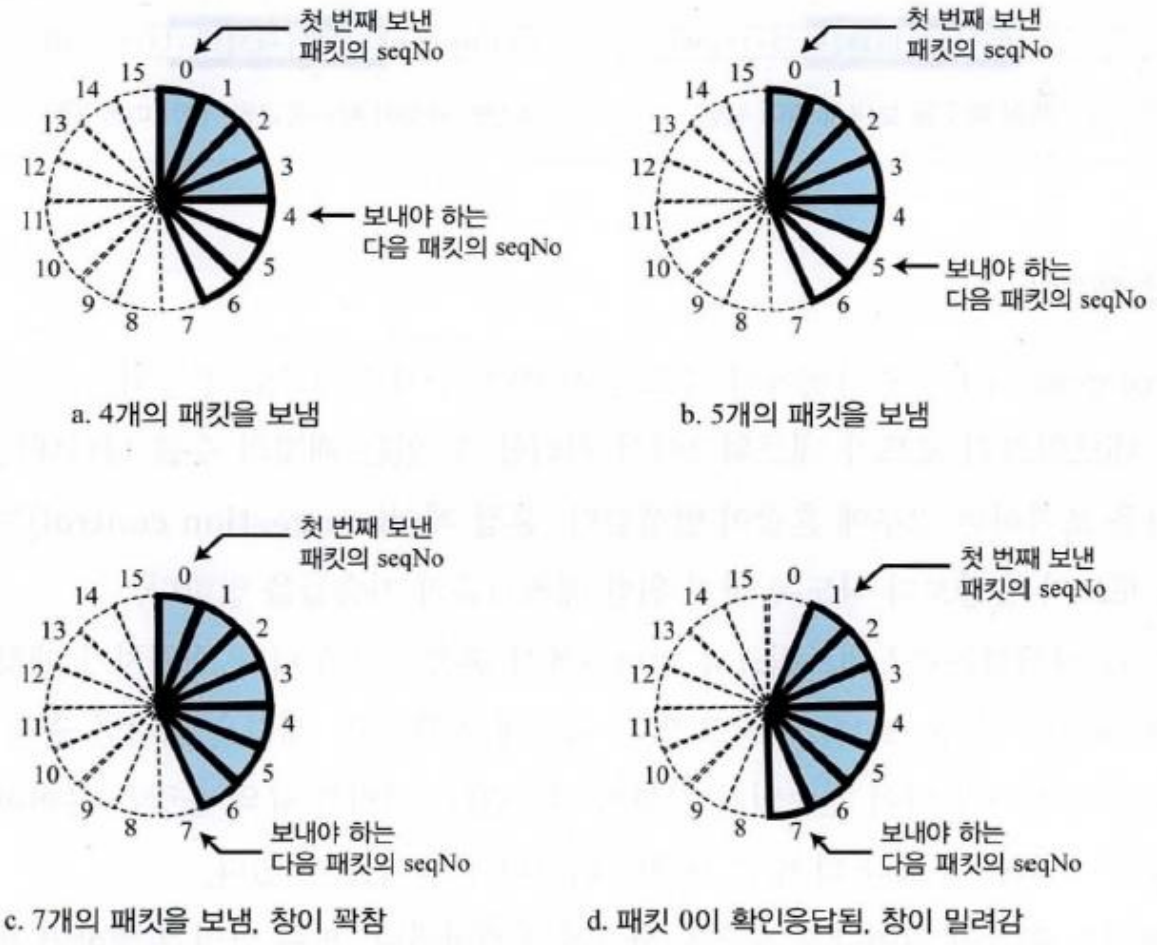
2. 전송층 서비스

- 흐름 제어와 오류 제어의 결합
  - 두 개의 버퍼에 번호가 매겨지면 오류 제어와 흐름 제어 기능이 결합할 수 있다.
  - 전송할 패킷이 있는 송신 측은 패킷의 순서 번호로서 버퍼 내의 비어있는 영역의 제일 앞 부분의 번호인 x를 사용한다.
  - 패킷이 전송되면 패킷의 복사본은 메모리 영역 x 에 저장되며 상대방으로부터 확인응답을 기다린다.
  - 전송된 패킷에 대한 확인응답이 도착하면, 패킷은 제거되고 메모리 영역은 비워진다.
  - 수신 측에 순서 번호 y를 갖는 패킷이 도착하면, 수신 측은 응용층에서 수신할 준비가 되기 전까지는 수신된 패킷을 메모리 영역 y에 저장한다.
  - 그리고 패킷 y의 도착을 알리기 위하여 확인응답이 전송된다.

흐름 제어와 오류 제어의 결합

미닫이 창

- 순서 번호가 modulo 2^m을 사용하기 때문에, 0부터 $2^m - 1$ 까지의 순서 번호를 원형으로 표현할 수 있다.



2. 전송층 서비스

흐름 제어와 오류 제어의 결합

미닫이 창

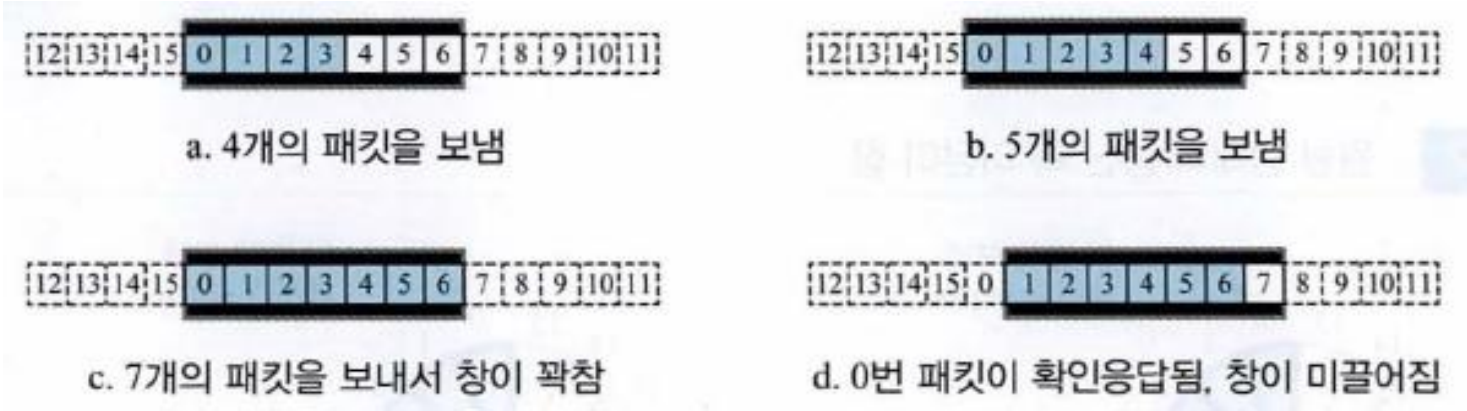
- 버퍼는 미닫이 창(sliding window) 이라고 하는 일련의 패킷으로 표현되며, 각 패킷은 언제나 원형의 한 부분을 차지한다.
- 송신 측에서 패킷이 전송되면 관련 패킷이 마크된다.
- 모든 패킷이 마크된다는 것은 버퍼가 다 차서 더 이상의 메 시지를 응용층으로부터 수신할 수 없다는 것을 의미한다.
- 확인응답이 수신되면 해당 패킷의 마크는 해제된다.
- 창의 시작부터 몇 개의 연속적인 패킷의 마크가 해제되면 창의 끝에서 좀 더 많은 빈 패킷을 허용하기 위하여 창을 관련 순서 번호의 범위를 자연스럽게 돌리게 된다.

2. 전송층 서비스

흐름 제어와 오류 제어의 결합

미닫이 창

- 대부분의 프로토콜에서는 선형 표현 방식을 이용하여 미닫이 창을 보여준다.
- 이 아이디어는 좀 더 적은 공간을 이용하여 미닫이 창을 동일하게 표현할 수 있다.
- 그림은 이러한 표현 방식을 보여준다.



2. 전송층 서비스

혼합 제어

- 혼잡(congestion)은 인터넷에서 중요한 문제이다.
- 네트워크로 전송되는 패킷의 수를 나타내는 네트워크의 로드가 네트워크에서 처리할 수 있는 패킷의 수를 나타내는 네트워크의 용량을 초과하는 경우에 혼잡이 발생한다.
- 혼잡 제어(congestion control)는 혼잡을 제어하고 로드가 용량보다 작도록 하기 위한 메커니즘과 기술들을 말한다.
- 혼잡 제어는 혼잡이 일어나지 않도록 사전에 방지하거나 또는 만일 발생하면 혼잡을 제거하기 위한 기술과 메커니즘을 말한다.

2. 전송층 서비스

혼합 제어

개방 루프 혼잡 제어

- 개방 루프 혼잡 제어 (open-loop congestion control)에서는 혼잡을 사전에 방지하기 위한 정책들이 적용된다.
- 이러한 메커니즘에서 혼잡 제어는 발신지나 목적지에서 수행된다.
- 재전송 정책
  - 재전송은 때때로 피할 수 없다.
  - 만일 송신 측에서 전송한 패킷이 손실됐거나 훼손되었다고 판단되면, 이러한 패킷은 재전송되어야 한다.
  - 일반적으로 재전송은 네트워크에서 혼잡을 가중시킨다.
  - 그렇지만 적절한 재전송 정책은 혼잡을 방지할 수 있다.
  - 효율을 최적화하면서도 동시에 혼잡을 방지할 수 있는 재전송 정책과 재전송 타이머가 설계되어야 한다.

- 창 정책
  - 송신 측에서 창(window)의 형태 역시 혼잡에 영향을 미친다.
  - 다음 절에서는 혼잡제어의 관점에서 선택적 반복(selective-repeat) 창 방식이 N-프레임-후퇴 (go-back-N) 창보다 좋은 성능을 제공하는 것을 알 수 있다.

2. 전송층 서비스

혼합 제어

개방 루프 혼잡 제어

- 확인응답 정책
  - 수신 측 확인응답 정책도 역시 혼잡에 영향을 미칠 수 있다.
  - 만일 수신 측에서 수신한 각 패킷에 대해 확인응답을 천천히 하면 송신 측의 전송 속도는 느려질 것이고, 따라서 혼잡 방지에 도움이 될 것이다. 이러한 경우에 여러 가지 방법이 사용된다.
  - 수신 측에서는 자신이 전송할 패킷이 있거나 또는 특정한 타이머가 만료되는 경우에만 확인 응답을 전송한다.
  - 또한 수신 측에서는 한번에 N개의 패킷에 대한 확인응답을 전송할 수 있다.
  - 확인응답 역시 네트워크 부하의 한 부분이라는 것을 알아야 한다.
  - 더 적은 확인응답을 전송한다는 것은 네트워크의 부하를 줄이는 것을 의미한다.

2. 전송층 서비스

혼합 제어

폐 루프 혼잡 제어

- 폐 루프 혼잡 제어 (closed-loop congestion control)는 혼잡이 발생한 후에 혼잡을 완화시키기 위한 방식이다.
- 여러 가지 메커니즘이 서로 다른 프로토콜에서 사용되었다.
- 송신 창의 크기는 가변적이다.
- 인터넷의 혼잡은 송신 창 크기를 결정할 수 있는 하나의 요인이다.
- 송신 측 전송층은 패킷의 손실을 관찰하여 인터넷에서 혼잡을 감시하고 혼잡이 증가하는 경우에는 창의 크기를 작게 하거나 또는 혼잡이 완화되는 경우에는 창의 크기를 늘리는 전략을 시용한다.

2. 전송층 서비스

비연결형과 연결형 서비스

- **네트워크층 프로토콜과 마찬가지로 전송층 프로토콜도 비연결형 (connectionless)과 연결형 (connection-oriented) 서비스를 제공한다.**
- **전송층에서는 패킷의 물리적인 경로에 대해서는 관여하지 않는다(두 전송층 사이에 논리 연결이 있다고 가정).**
- **전송층의 비연결형 서비스는 패킷 간의 독립성을 의미하며 연결형은 종속성을 의미한다.**

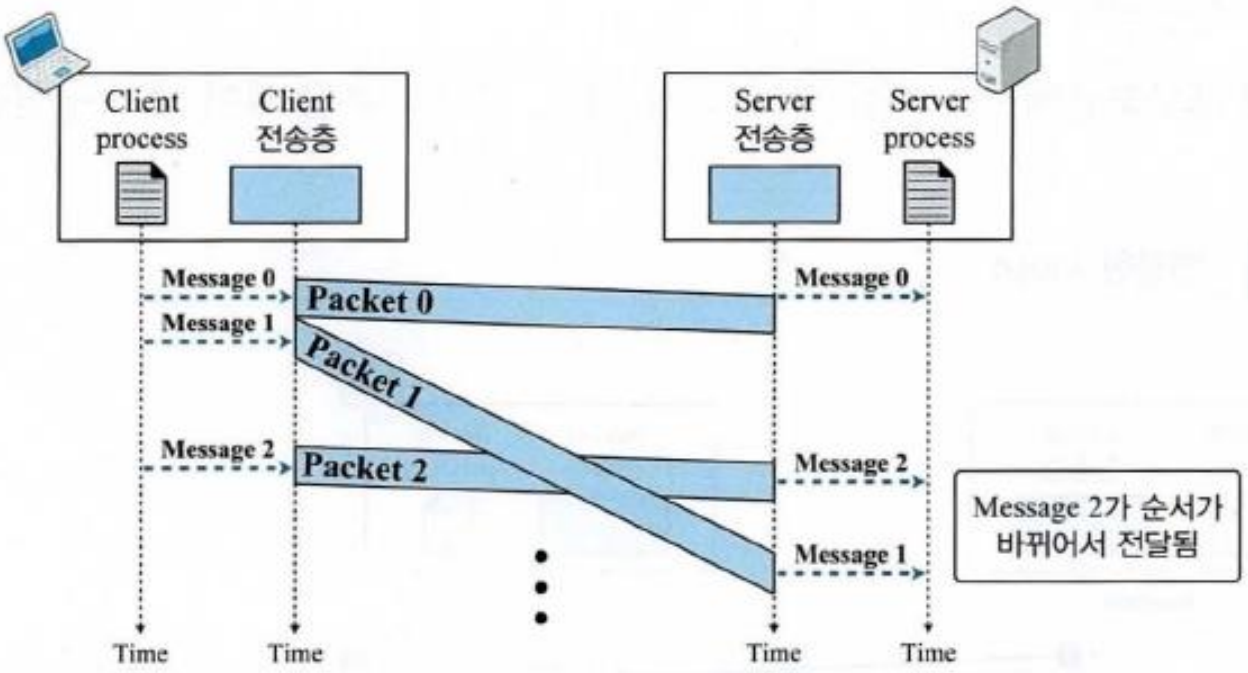
- **비연결형 서비스**
  - 비연결형 서비스에서 발신지 프로세스(응용 프로그램)는 메시지를 전송층에서 수신 가능한 크기의 여러 개의 데이터 단편으로 나눈 후 각 데이터 단편을 하나씩 전송층으로 전달한다.
  - 전송층은 데이터들의 관계를 고려하지 않고 각 데이터 단편을 독립적인 하나의 단위로 간주한다.
  - 응용층으로부터 하나의 조각이 들어오면 전송층은 단편을 패킷으로 캡슐화한 후에 전송한다.

2. 전송층 서비스

■ 비연결형과 연결형 서비스

▪ 비연결형 서비스

- 그림은 시간 축을 이용한 패킷의 전송 과정을 보여준다.



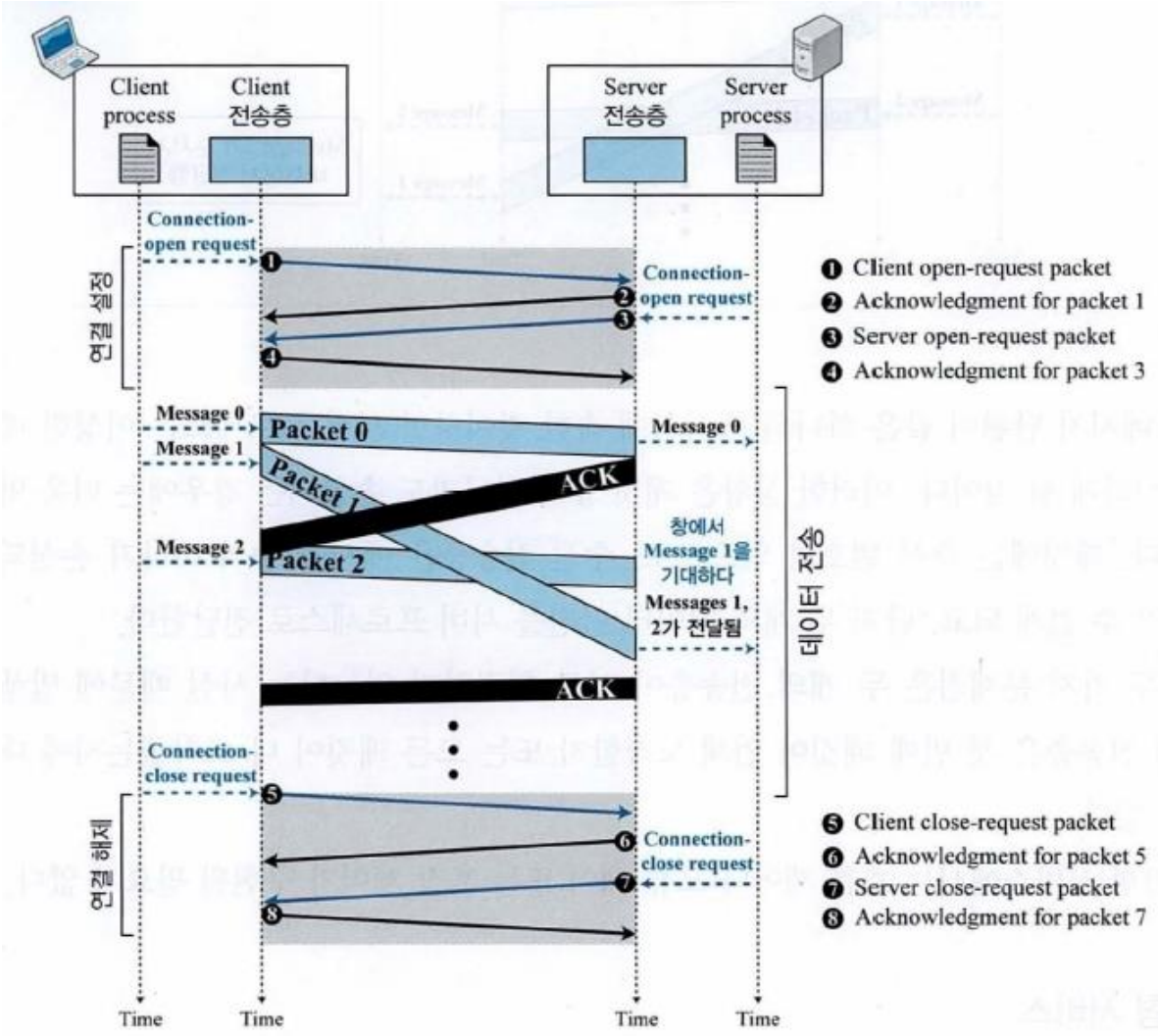
- 비연결형 서비스에서는 흐름 제어나 오류 제어 또는 혼잡 제어가 구현될 필요가 없다.

2. 전송층 서비스

비연결형과 연결형 서비스

연결형 서비스

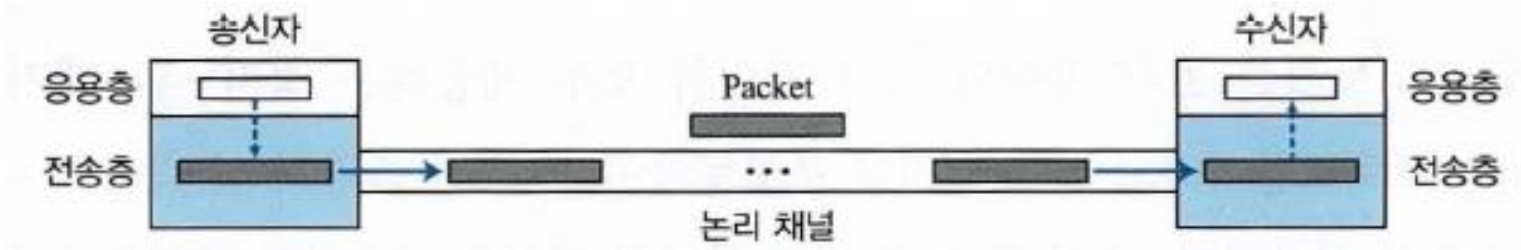
- 연결형 서비스에서 클라이언트와 서버는 먼저 연결을 설정한다.
- 데이터 교환은 연결이 설정된 이후에나 가능하다.
- 데이터의 교환이 완료된 후에는 연결은 해제된다.
- 그림은 전송층의 연결 설정 데이터 전송 그리고 연결 해제 단계를 보여준다.



3. 전송층 프로토콜

단순 프로토콜

- 첫 번째 프로토콜은 흐름 제어나 오류 제어가 없는 단순(simple) 비연결형 프로토콜이다.
- 이 프로토콜에서 수신 측은 수신한 패킷을 즉시 처리할 수 있다고 가정한다.
- 그림은 이 프로토콜의 간략한 동작을 보여준다.



- 자신의 응용층으로부터 전달된 메시지를 수신한 전송층은 메시지를 패킷으로 만든 후, 이 패킷을 전송한다.
- 수신 측 전송층은 자신의 네트워크층을 통하여 들어온 패킷을 수신한 후, 이 패킷으로부터 메시지를 추출하고, 이 메시지를 자신의 응용층으로 전달한다.

3. 전송층 프로토콜

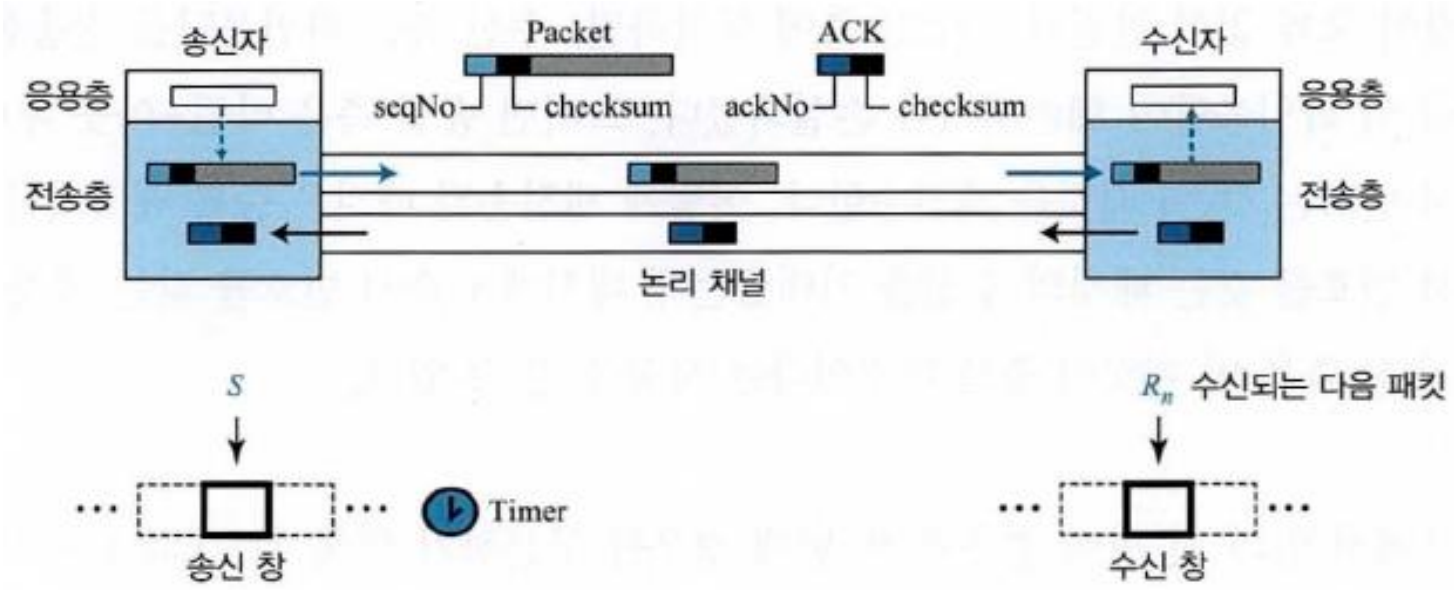
정지-후-대기 프로토콜

- 두 번째 프로토콜은 흐름 제어와 오류 제어를 모두 제공하는 정지-후-대기 (Stop-and-Wait) 프로토콜이라고 하는 연결형 프로토콜이다.
- 송신 측과 수신 측은 모두 크기가 1인 미닫이 창을 사용한다.
- 송신 측은 한 번에 하나의 패킷을 전송하고 확인응답이 오기 전까지는 다음 패킷을 전송하지 않는다.
- 패킷이 훼손되었는지를 검사하기 위하여 각 데이터 패킷에 검사합(checksum)을 추가한다.
- 패킷을 수신한 수신 측은 패킷을 검사하여 패킷의 검사합이 틀리면 패킷이 훼손되었다고 간주하고 조용히 버린다.

3. 전송층 프로토콜

정지-후-대기 프로토콜

- 그림은 정지-후-대기 프로토콜의 개괄적인 동작을 보여준다.



- 정지-후-대기 프로토콜은 흐름 제어와 오류 제어를 제공하는 연결형 프로토콜이다.

정지-후-대기 프로토콜

순서 번호

- 패킷을 중복 수신하지 않도록 프로토콜은 순서 번호(sequence number)와 확인응답 번호(acknowledgment number)를 사용한다.
- 패킷의 헤더에는 패킷의 순서 번호를 나타내는 필드가 추가된다.
- 여기서 고려해야 할 사항은 순서 번호의 범위이다.
- 패킷의 크기를 최소화하기 위해서는 모호하지 않고 완벽한 통신을 제공할 수 있는 최소 범위를 찾아야 한다.
- 송신 측에서 순서 번호 x 의 패킷을 전송했다고 가정해보자.
- 그러면 다음과 같은 세 가지 경우가 발생할 수 있다.
  - 패킷이 오류 없이 안전하게 수신 측에 도착하면, 수신 측은 확인응답을 전송한다. 확인응답이 송신 측에 도착하면 송신 측은 x + 1 의 순서 번호를 갖는 다음 패킷을 전송할 것이다.
  - 패킷이 훼손되어 수신 측에 도착하지 않으면 송신 측은 타임-아웃 후에 (x의 순서 번호를 갖는) 패킷을 재전송한다. 이 패킷을 수신한 수신 측은 확인응답으로 회신한다.
  - 패킷이 오류 없이 안전하게 수신 측에 도착하면 수신 측은 확인응답을 전송한다. 그런데 이 확인응답이 훼손되거나 손실되었다. 그러면 송신 측은 타임—아웃 후에 (x의 순서 번호는 갖는) 패킷을 재전송한다. 이렇게 재전송된 패킷은 중복 패킷이다. x + 1 순서 번호를 갖는 패킷의 수신을 기대했는데 대신에 x 순서 번호를 갖는 패킷을 수신한 수신 측은 이 패킷이 중복 패킷이라는 사실을 알 수 있다.

정지-후-대기 프로토콜

순서 번호

- 패킷을 중복 수신하지 않도록 프로토콜은 순서 번호(sequence number)와 확인응답 번호(acknowledgment number)를 사용한다.
- 패킷의 헤더에는 패킷의 순서 번호를 나타내는 필드가 추가된다.
- 여기서 고려해야 할 사항은 순서 번호의 범위이다.
- 패킷의 크기를 최소화하기 위해서는 모호하지 않고 완벽한 통신을 제공할 수 있는 최소 범위를 찾아야 한다.
- 송신 측에서 순서 번호 x 의 패킷을 전송했다고 가정해보자.
- 그러면 다음과 같은 세 가지 경우가 발생할 수 있다.
  - 패킷이 오류 없이 안전하게 수신 측에 도착하면, 수신 측은 확인응답을 전송한다. 확인응답이 송신 측에 도착하면 송신 측은 x + 1 의 순서 번호를 갖는 다음 패킷을 전송할 것이다.
  - 패킷이 훼손되어 수신 측에 도착하지 않으면 송신 측은 타임-아웃 후에 (x의 순서 번호를 갖는) 패킷을 재전송한다. 이 패킷을 수신한 수신 측은 확인응답으로 회신한다.
  - 패킷이 오류 없이 안전하게 수신 측에 도착하면 수신 측은 확인응답을 전송한다. 그런데 이 확인응답이 훼손되거나 손실되었다. 그러면 송신 측은 타임—아웃 후에 (x의 순서 번호는 갖는) 패킷을 재전송한다. 이렇게 재전송된 패킷은 중복 패킷이다. x + 1 순서 번호를 갖는 패킷의 수신을 기대했는데 대신에 x 순서 번호를 갖는 패킷을 수신한 수신 측은 이 패킷이 중복 패킷이라는 사실을 알 수 있다.

정지-후-대기 프로토콜

순서 번호

- 수신 측에서 앞의 첫 번째 경우와 세 번째 경우를 구분하기 위해서는 x와 x + 1 의 순서 번호가 필요하지만, 패킷이 x+2 순서 번호를 가질 필요는 없다는 것을 알 수 있다.
- 첫 번째 경우에서 x와 x + 1 의 순서 번호를 갖는 패킷들이 모두 확인응답되면, 그 다음 패킷은 다시 x 의 순서 변호를 갖더라도 양쪽에서 명확하게 구분할 수 있을 것이다.
- 두 번째와 세 번째의 경우에서 새 패킷의 순서 번호는 x+2가 아닌 x + 1 이다.
- 따라서 단지 x와 x + 1만이 필요하다면 x= 0으로 하고 또한 x + 1 = 1 로 할 수 있을 것이다.
- 이것은 0, 1, 0, 1, 0, 1의 순서를 갖는다는 것을 의미하며 다시 말하면 모듈러 (modulo) 2 연산을 따른다.

정지-후-대기 프로토콜

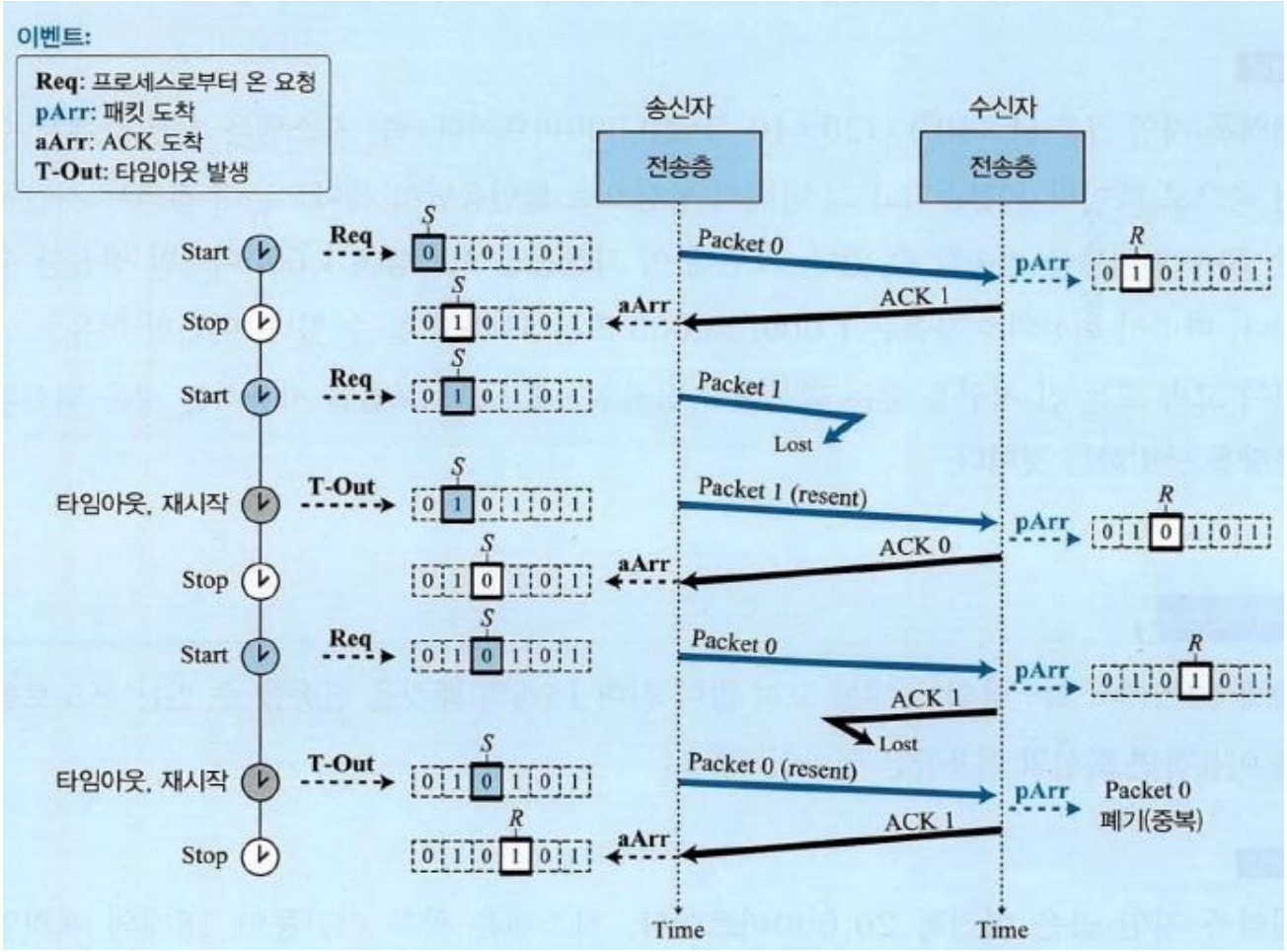
확인응답 번호

- 순서 번호가 데이터 패킷과 확인응답에 포함되어야 하기 때문에 여기서는 다음과 같이 정의하고자 한다.
- 확인응답 변호는 항상 수신 측에서 받기를 기대하는 다음 패킷의 순서 번호를 나타낸다
- 예를 들어 패킷 0 이 안전하게 들어오면, 수신 측은 확인응답 번호로 1을 갖는 ACK(즉, 패킷 1이 다음에 수신되기를 기대한다는 것을 의미함)를 전송한다.
- 만일 패킷 1이 안전하게 도착하면, 수신 측은 확인응답 번호 0을 갖는 ACK(즉, 패킷 0을 기대한다는 것을 의미)를 전송한다.
- 송신 측은 S(송신자)라는 송신 창의 한 틈새를 가리키는 제어 변수를 가지고 있다.
- 수신 측도 R(수신자)이라는 수신 창의 한 틈새를 가리키는 제어 변수를 가지고 있다.

정지-후-대기 프로토콜

확인응답 번호

- 그림은 정지-후-대기 프로토콜의 예를 보여준다.



정지-후-대기 프로토콜

효율

- 정지-후-대기 프로토콜은 채널이 두껍고 긴 경우에는 상당히 비효율적이다.
- 여기에서 두껍다는 것은 채널이 큰 대역폭(즉, 고속)을 가지고 있는 것을 의미하고, 길다는 것은 왕복 지연이 길다는 것을 의미한다.
- 이 두 항목을 곱한 것을 대역폭-지연 곱(Bandwidth-Delay product)이라고 한다.
- 채널을 파이프라고 생각해보자.
- 대역폭 지연 곱은 비트로 표현된 파이프의 용량이라 할 수 있다.
- 파이프의 용량은 같다.
- 파이프의 용량을 모두 이용하지 않는 것은 비효율적이라 할 수 있다.
- 대역폭-지연 곱은 수신 측의 확인응답을 기다리면서 송신 측에서 파이프를 통하여 전송할 수 있는 비트의 수이다.

정지-후-대기 프로토콜

효율

- 예제. 전송-후-대기 시스템에서 회선의 대역폭이 1 Mbps이고 1비트가 왕복하는 데 20msec가 걸린다고 가정해보자. 대역폭 지연 곱은 얼마인가? 시스템의 데이터 패킷이 1,000 비트의 길이이면, 회선의 이용율은 얼마인가?

정지-후-대기 프로토콜

효율

- 예제. 앞의 예제에서 확인응답에 대한 고려 없이 최대 15개의 패킷을 전송할 수 있는 프로토콜을 이용하면 회선의 이용률은 얼마인가?

3. 전송층 프로토콜

정지-후-대기 프로토콜

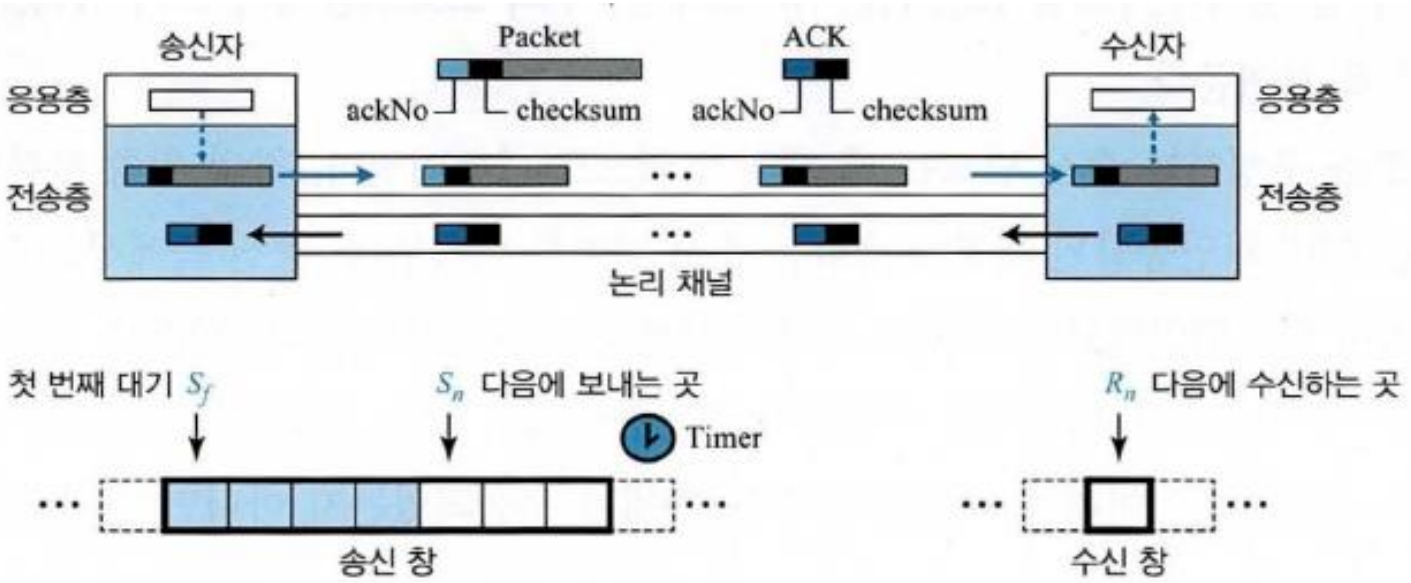
파이프라인

- 네트워크와 다른 분야에서 이전 업무가 완료되기 전에 새로운 업무가 시작되는 경우가 있는데 이것을 파이프라인 (pipeline) 이라고 한다.
- 정지-후-대기 프로토콜에서는 송신 측에서 패킷이 목적지에 도착하고 확인응답되기 전에는 다음 패킷을 전송할 수 없기 때문에 파이프라인의 기능을 제공하지 못한다.
- 그렇지만 다음의 두 프로토콜은 송신 측에서 앞서 전송한 패킷에 대한 피드백을 수신하기 전에 추가적인 패킷을 전송할 수 있기 때문에 파이프라인이 적용된다고 할 수 있다.
- 파이프라인은 대역폭 지연 곱의 관점에서 전송 중인 비트의 수가 많은 경우에는 전송 효율을 향상시킨다.

3. 전송층 프로토콜

N -프레임-후퇴 프로토콜

- 전송 효율을 향상시키기 위해서는(즉 파이프를 채우기 위해서는) 송신 측은 확인응답을 기다리는 동안에 여러 개의 패킷을 전송할 수 있어야 한다.
- 첫 번째 프로토콜은 N-프레임-후퇴(Go-Back-N 또는 GBN) 라고 한다.
- N-프레임-후퇴에서 중요한 점은 확인응답을 수신하기 전에 여러 개의 패킷을 전송할 수 있다는 것이다.
- 그렇지만 수신 측은 단지 하나의 패킷을 버퍼에 저장할 수 있다.
- 그림은 이 프로토콜의 간략한 동작을 보여준다.



3. 전송층 프로토콜

N -프레임-후퇴 프로토콜

순서 번호

- 순서 번호는 modulo 2^m이며, m은 순서 번호 필드의 비트수이다.

확인응답 번호

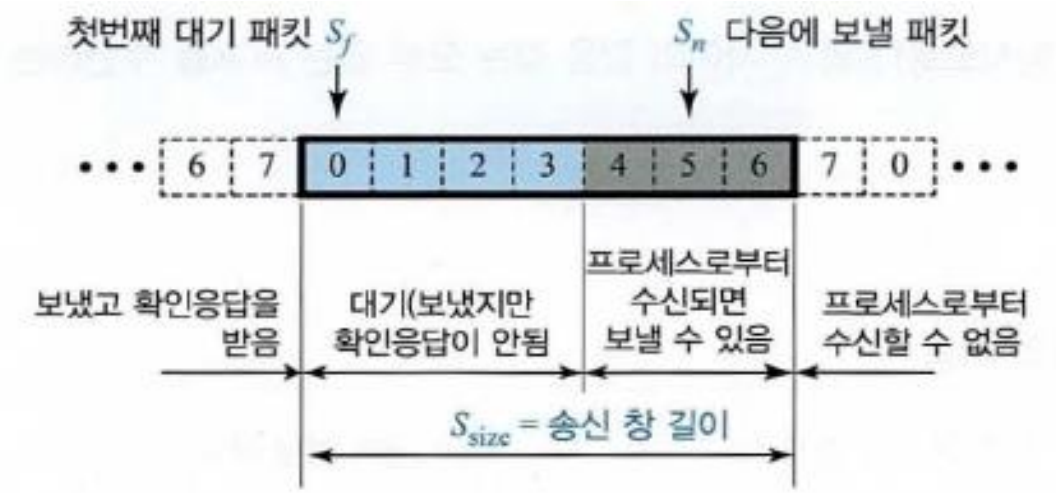
- 이 프로토콜에서 확인응답 변호는 누적 값이며 수신하기를 기대하는 다음 패킷의 순서 번호를 나타낸다.
- 예를 들어 확인응답 번호(ackNo)가 7이라는 것은 처음부터 6번까지의 순서 번호를 갖는 패킷 모두가 안전하게 도착했고, 수신 측은 순서 번호 7을 갖는 패킷의 수신을 기다리고 있다는 것을 의미한다.

3. 전송층 프로토콜

N -프레임-후퇴 프로토콜

송신 창

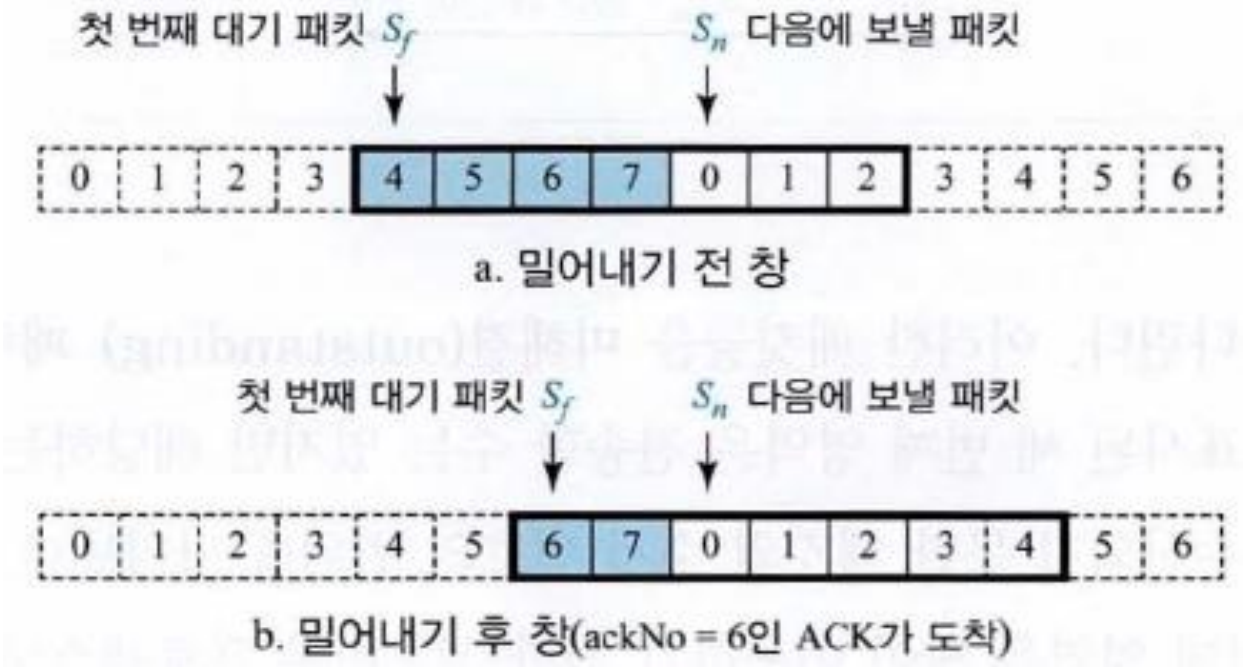
- 송신 창은 전송 중이거나 전송될 데이터 패킷의 순서 번호를 포함하는 가상의 상자를 나타낸다.
- 각 창의 위치에서 어떤 순서 변호들은 이미 전송된 패킷을 나타내고, 또 다른 순서 번호는 전송되고자 하는 패킷을 나타낸다.
- 창의 최대 크기는 2^m -1 이다.
- 그림은 N-프레임-후퇴 프로토콜에서 크기 7(즉, m=3)의 미닫이 창을 보여준다.



N -프레임-후퇴 프로토콜

송신 창

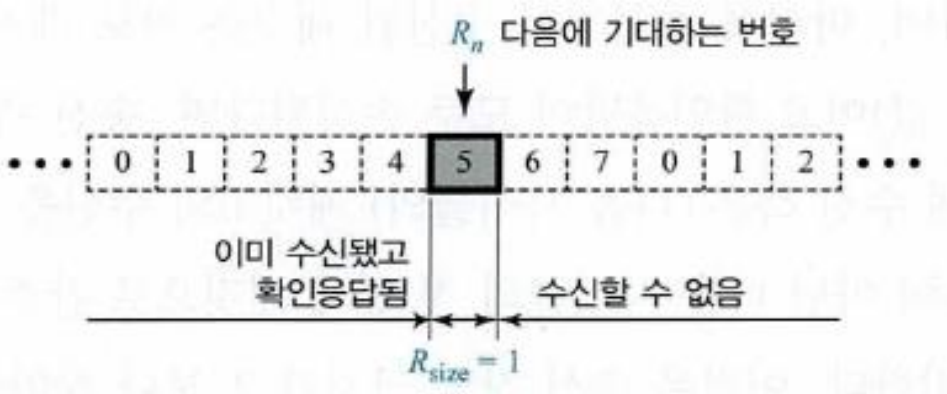
- 그림은 상대방으로부터 확인응답을 수신하는 경우에 어떻게 송신 창이 오른쪽으로 하나 이상의 틈새를 이동하는지를 보여준다.
- 이 그림에서 ackNo=6 인 확인응답이 도착하였다.
- 이것은 수신 측에서 순서 번호 6을 가지는 패킷을 기다리고 있다는 것을 의미한다.



N -프레임-후퇴 프로토콜

수신 창

- 수신 창은 올바른 데이터 패킷을 수신하고 또한 올바른 확인응답이 전송될 수 있도록 해준다.
- N-프레임-후퇴에서 수신 창의 크기는 항상 1 이다.
- 수신 측은 항상 특정한 패킷을 기다린다.
- 순서에 어긋나게 도착한 패킷은 폐기되며 재전송될 것이다.
- 그림은 수신 창을 보여준다.



N -프레임-후퇴 프로토콜

타이머

- 전송된 각각의 패킷에 대한 타이머(timer)가 있지만, 여기에서는 단지 하나의 타이머만 사용한다고 가정한다.
- 이 이유는 첫 번째 미해결 패킷을 위한 타이머는 항상 먼저 만료되기 때문이다.
- 이 타이머가 만료되면 모든 미해결 패킷은 재전송된다.

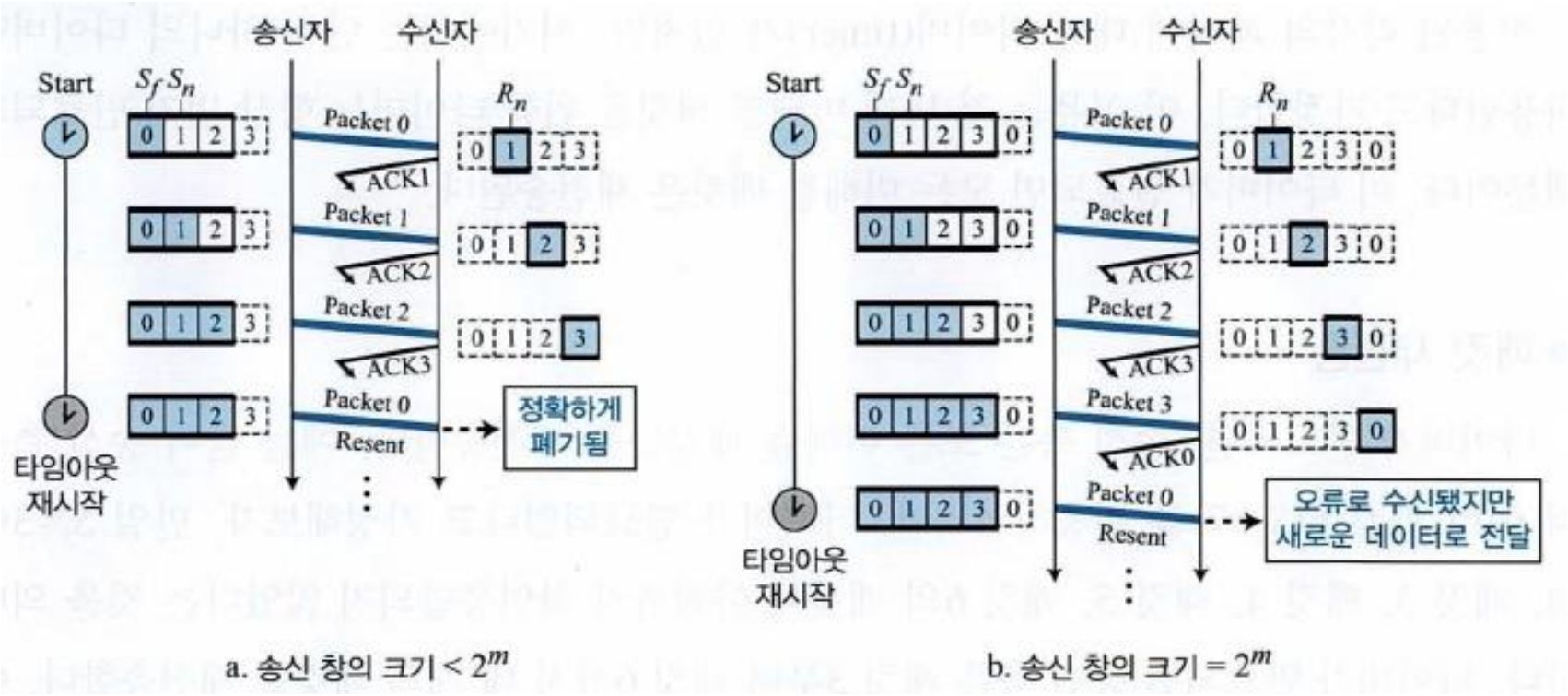
패킷 재전송

- 타이머가 만료되면, 송신 측은 모든 미해결 패킷들을 재전송한다.
- 예를 들어 송신 측에서 이미 패킷 6(Sn =7)을 전송하였지만, 타이머가 만료되었다고 가정해보자.
- 만일 S_f=3 이면, 패킷 3, 패킷 4, 패킷 5, 패킷 6의 패킷이 아직까지 확인응답되지 않았다는 것을 의미한다.
- 타이머가 만료되면 송신 측은 패킷 3부터 패킷 6까지 네 개의 패킷을 재전송한다.
- 이것이 프로토콜을 N-프레임-후퇴라고 하는지에 대한 이유이다.
- 타임-아웃이 발생하면 기기는 N 위치만큼 후퇴하며 모든 패킷을 재전송한다.

N -프레임-후퇴 프로토콜

송신 창 크기

- 그림은 창의 크기가 3 인 경우와 4 인 경우를 비교하여 보여준다.



a. 송신 창의 크기 < 2^m

b. 송신 창의 크기 = 2^m

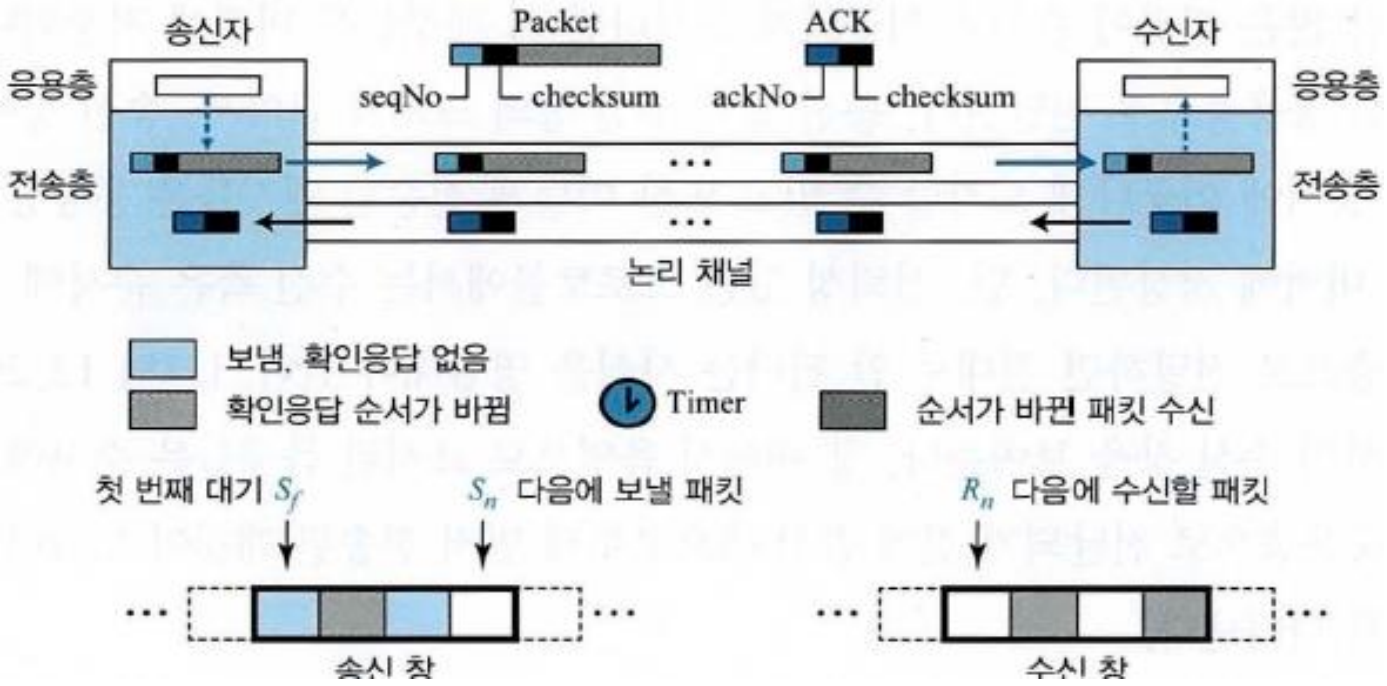
N -프레임-후퇴 프로토콜

N-프레임-후퇴와 정지-후-대기 비교

- N-프레임-후퇴 프로토콜과 정지-후-대기 프로토콜과는 유사한 면이 있다는 것을 발견할 수 있을 것이다.
- 정지-후-대기 프로토콜은 실제로 순서 번호가 0과 1인 두 번호만 가지며 송신 창의 크기가 1 인 N-프레임-후퇴 프로토콜이다.
- 다시 말하면, m=1 이고 2^m-1=1 이다.N 프레임—후퇴에서는 modulo 2^m의 연산방식을 사용한다.
- 정지 대기에서 연산 방식은 modulo 2 이며, 이것은 m=1 인 경우의 2^m의 값과 같다.

4. 선택적 반복 프로토콜

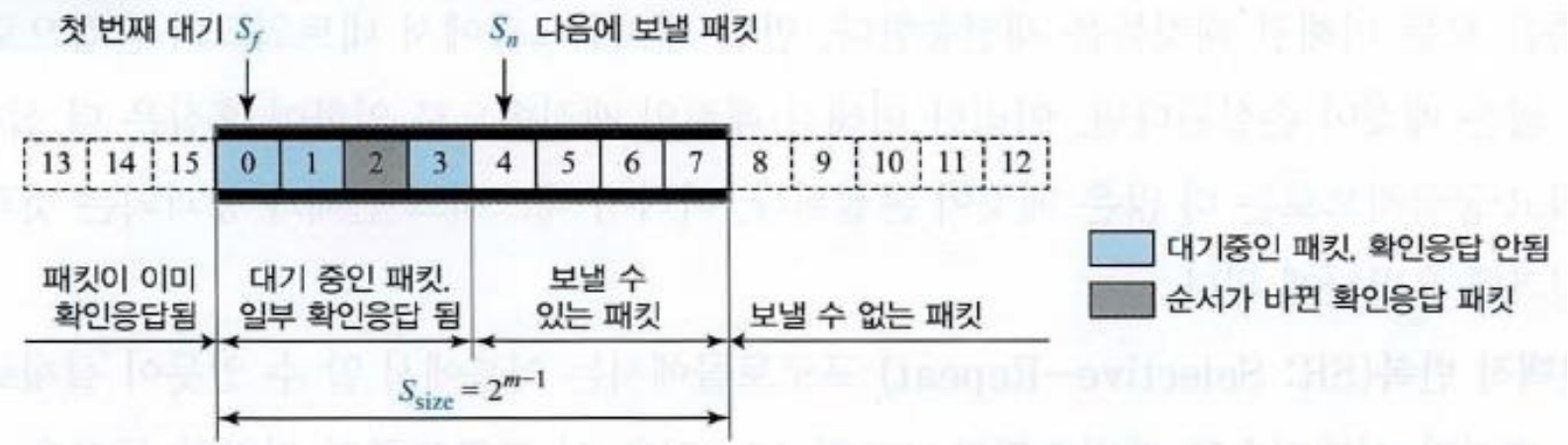
- 선택적 반복(SR: Selective-Repeat) 프로토콜에서는 이름에서 알 수 있듯이 실제로 손실된 패킷만 선택적으로 재전송된다.
- 그림은 이 프로토콜의 간략한 동작을 보여준다.



4. 선택적 반복 프로토콜

창

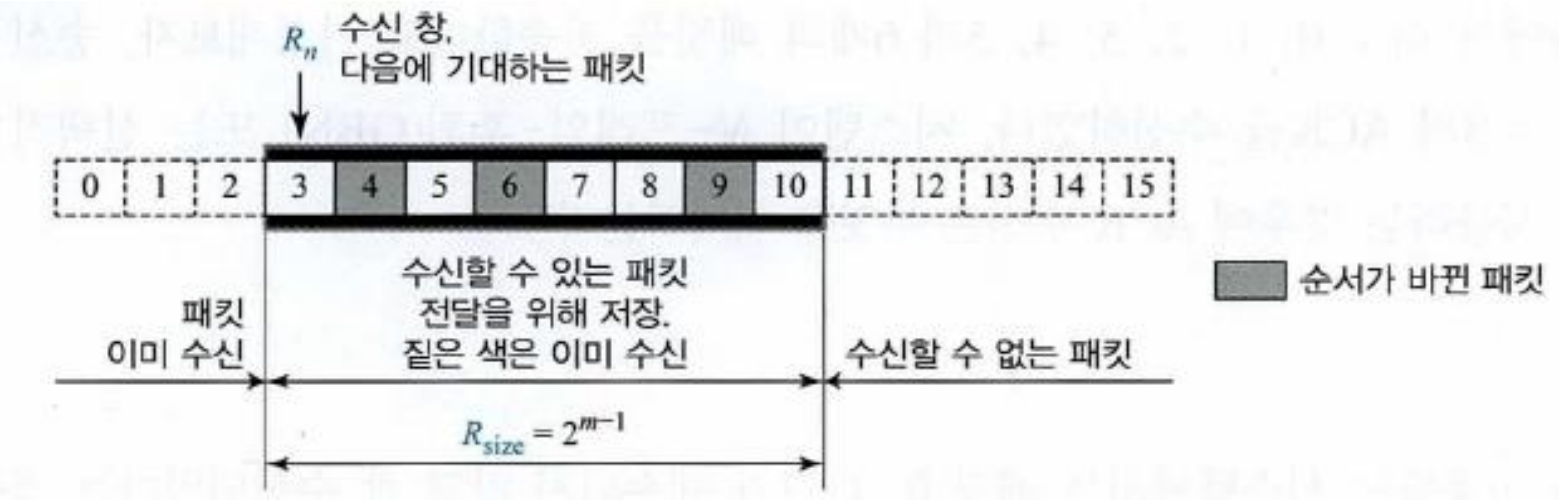
- 선택적 반복 프로토콜 역시 두 개의 창(송신 창과 수신 창)을 사용한다.
- 그렇지만 이 프로토콜에서 사용하는 창과 N-프레임-후퇴 프로토콜에서 사용하는 창과는 여러 가지 면에서 차이가 있다.
- 첫째, 송신 창의 크기는 2^{m-1} 로 상당히 작다.
- 둘째, 수신 창은 송신 창과 같은 크기를 갖는다.
- 그림은 선택적 반복 프로토콜의 송신 창의 크기를 보여준다.



4. 선택적 반복 프로토콜

창

- 선택적 반복에서 수신 창은 N-프레임-후퇴에서 사용하는 것과는 완전히 다르다.
- 수신 창의 크기는 송신 창의 크기(즉, 최대 2^{m-1})와 같다.
- 그림은 선택적 반복에서의 수신 창을 보여준다.



4. 선택적 반복 프로토콜

타이머

- 이론적으로 선택적 반복에서는 각각의 미해결 패킷마다 하나씩의 타이머를 사용한다.
- 타이머가 만료되면 해당 패킷이 재전송된다.
- 다시 말하면 N-프레임 -후퇴 프로토콜에서는 미해결 패킷들을 하나의 그룹으로 처리하지만, 선택적 반복 프로토콜에서는 미해결 패킷들을 독립적으로 처리한다.
- 그렇지만 선택적 반복을 구현하는 대부분의 전송층 프로토콜은 단지 하나의 단일 타이머만 사용한다.

확인 응답

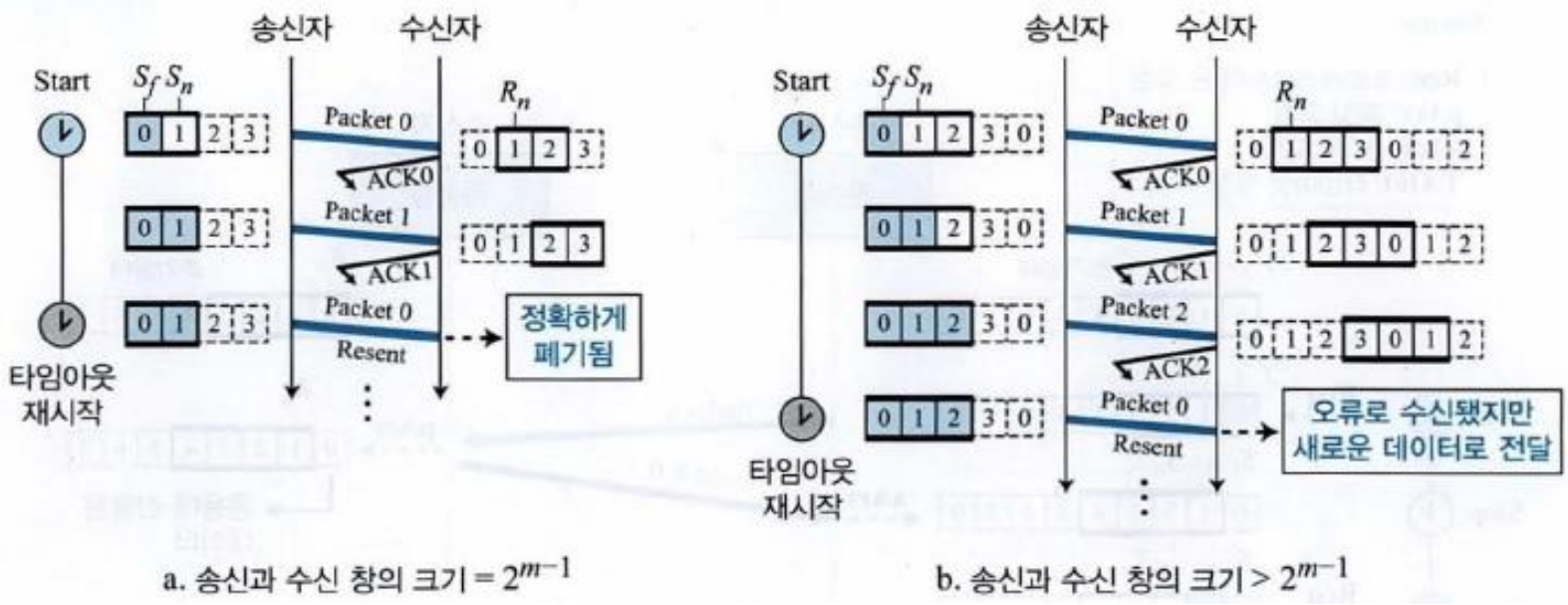
- 두 프로토콜 사이에는 또 다른 차이점이 있다.
- N-프레임-후퇴에서 ackNo는 누적이다.
- 즉, ackNo는 수신하기를 기대하는 다음 패킷의 순서 번호를 나타내며, 이 번호 이전의 모든 패킷은 모두 잘 도착했다는 것을 의미한다.
- 선택적 반복에서 확인응답의 의미는 다르다.
- 선택적 반복에서 ackNo는 오류 없이 수신된 하나의 단일 패킷의 순서 번호를 나타내며 다른 패킷들에 대한 어떠한 피드백도 제공하지 않는다.

4. 선택적 반복 프로토콜

- 확인 응답
  - 예제. 송신 측에서 패킷 0, 1, 2, 3, 4, 5 의 6개의 패킷을 전송한다고 가정해보자. 송신 측이 ackNo = 3의 ACK를 수신하였다. 시스템이 N 프레임 후퇴 (GBN) 또는 선택적 반복 (SR)을 사용하는 경우에 ACK 수신은 무엇을 의미하는가?

4. 선택적 반복 프로토콜

- 창의 크기
  - 그림은 창 크기 2와 창 크기 3의 비교를 보여준다.



a. 송신과 수신 창의 크기 = 2^{m-1}

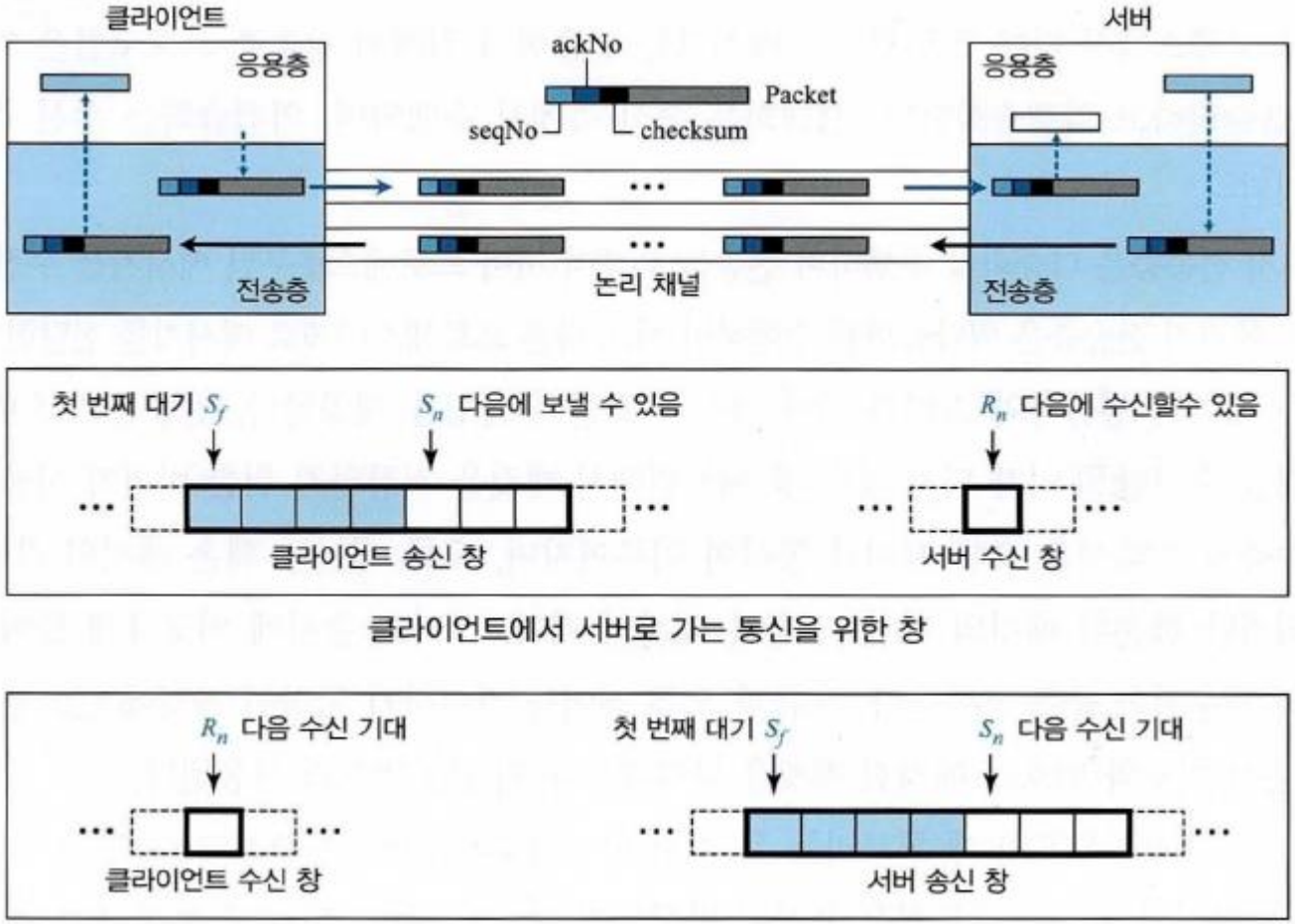
b. 송신과 수신 창의 크기 > 2^{m-1}

5. 양방향 프로토콜: 피기백킹

- 앞에서 설명한 네 개의 프로토콜은 모두 단방향이다.
- 즉, 데이터 패킷은 한 방향으로 전송되며 확인응답은 반대의 방향으로 전송된다.
- 실제로는 데이터 패킷은 클라이언트에서 서버로, 그리고 서버에서 클라이언트로 양방향 모두 전송되는 것이 일반적이다.
- 이것은 확인응답도 역시 양방향으로 전송될 필요가 있다는 것을 의미한다.
- 피기백킹(piggybacking) 이라는 기술은 양방향 통신의 효율을 향상시키기 위하여 사용된다.
- A에서 B로 데이터를 전달하는 패킷은 또한 B로부터 수신한 패킷에 대한 확인응답과 관련된 피드백 정보도 같이 전달할 수 있다.

5. 양방향 프로토콜: 피기백킹

- 그림은 피기백킹을 이용하여 양방향 통신이 가능한 GBN의 개략적인 동작을 보여준다.



Thank You