

# Introduction To Network Layer

---

## Network Layer Services

---

- 네트워크 레이어 프로토콜들의 역할

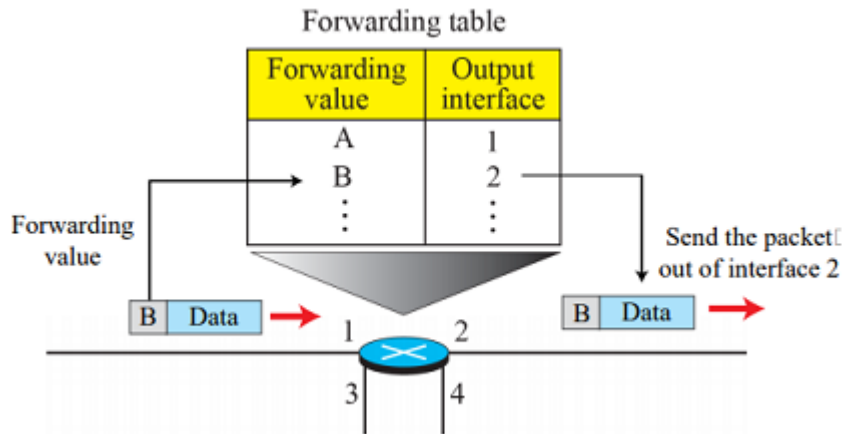
### Packetizing

- 네트워크 레이어의 제 1 역할
- Source Host : Encapsulation
  - 위쪽의 프로토콜로부터 Payload를 받음
  - src, dst 주소를 헤더에 추가하고, 다른 추가적인 정보를 넣음
  - datalink layer에 이를 전달
  - SAR(Segmentation and Reassembly)
    - Payload의 내용을 변경하는건 허용되지 않음(전달하기에 용량이 너무 큰 경우를 제외하면)
    - 용량이 너무 큰 경우 Fragmentation(Segmentation)을 거쳐야함
- Destination Host : Decapsulation
  - 데이터 링크 레이어로부터 패킷을 전달받음
  - 패킷을 Decapsulate하고, Payload를 위쪽 프로토콜로 넘긴다
  - SAR
    - 패킷이 조각난 상태면, 모든 조각이 도착할때까지 기다리고, 재조립해서 위쪽 프로토콜로 전달함
- Routers in the path
  - 패킷을 Decapsulate하는 것이 **허용되지 않음**
    - 패킷의 조각내야 하는 경우는 가능
    - 패킷을 조각내고 같은 헤더를 각각의 조각에 넣고, 추가적인 변경을 거친 후 보냄
  - src / dst를 바꾸는것도 허용 안됨
  - 패킷을 올바른 다음 네트워크로 보내기 위해 주소를 참고하기만 함
- 네트워크 레이어는 우체국의 역할을 한다고 보면 됨, 패키지를 Sender에서 Receiver로 **내용 변화 없이 무사히** 전송하는 것을 목적으로 함

### Routing and Forwarding

- Routing : Destination으로 가는 최적의 길을 찾아야 할 책임이 있음
- Forwarding
  - Unicast Routing : 라우터가 Attached Network로 부터 패킷을 받으면, 해당 패킷을 다른 Attached 네트워크로 보내야 함
  - Multicasting Routing : 라우터가 Attached Network로 부터 패킷을 받으면, 해당 패킷을 **여러** Attached 네트워크로 보내야 함
  - 이 결정을 위해서 라우터가 목적지 주소와 헤더의 라벨을 사용해서 routing table에서 다음 output interface 을 결정함

- Forwarding Table / Routing Table : 다음 목적 네트워크를 결정하는 테이블



## Other services

- Error Control
  - 직접적으로 에러 컨트롤을 제공하지 않음(CRC 등)
  - 패킷이 조각 난 경우가 있을 수 있기 때문에, 이 단계에서의 오류 검사는 비효율적
  - 그러나 datagram에 Checksum Field가 추가됨(손상 탐지)
  - ICMP : 에러 컨트롤 제공
- Flow Control
  - Receiver를 과부하시키지 않는 선에서 Source가 보낼 수 있는 데이터 양을 규제
  - 위쪽 레이어가 Flow Control Service를 제공, 따라서 네트워크 레이어에서의 추가적인 Flow Control은 전체 시스템을 비효율적으로 만듦
- Congestion Control
  - Source가 보내는 datagram의 수가 네트워크/라우터의 한계용량 이상일 경우 발생
  - 라우터가 datagram 몇개를 잃어버릴 수도 있음
    - Sender가 똑같은 packet을 보내는 경우가 생길 가능성이 높음
  - Congestion이 누적되면 시스템이 털썩해버려서 아무 Datagram도 전달 안되는 정도까지 이어질 수도 있음
- Quality Of Service(QoS)
  - 통신 서비스의 질이 점점 더 중요해짐
  - 네트워크 레이어를 건드리지 않기 위해, 위쪽레이어에서 다룸
- Security
  - Security Provision 없이 설계됨

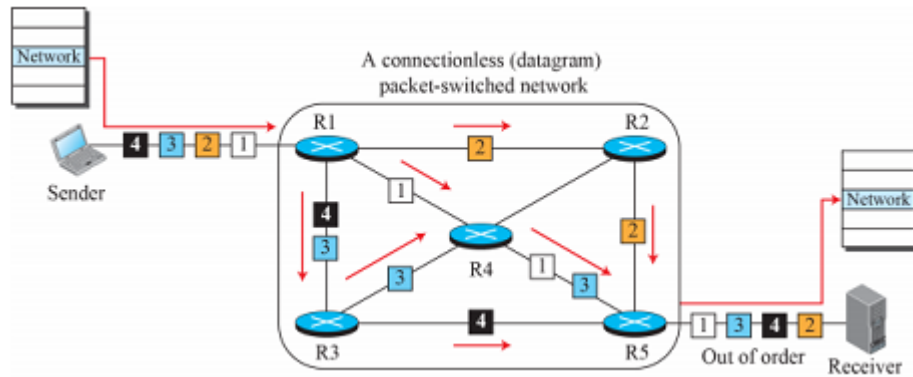
## Packet Switching

- 네트워크 레이어에서 Switching이 발생
- 라우터는 Input Port와 Output Port 사이의 연결을 만드는 스위치임
  - 전달하는 역할

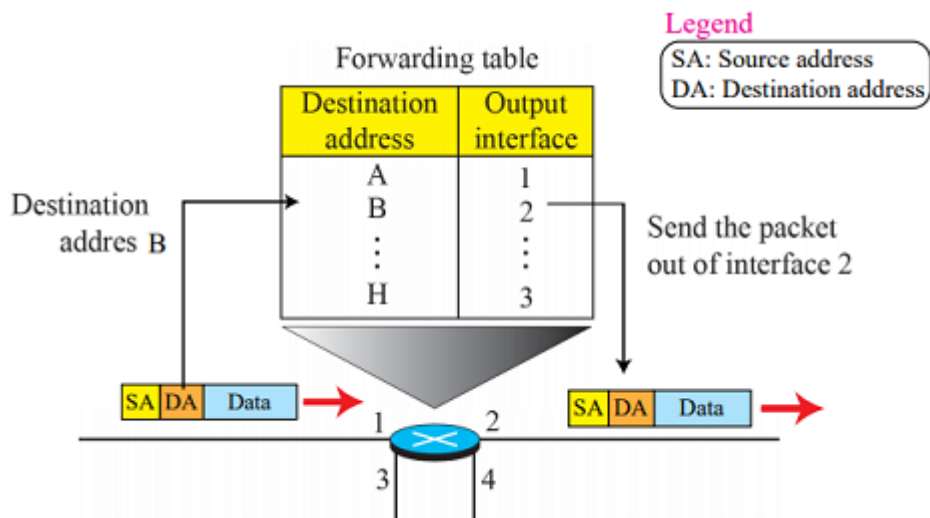
## Datagram Approach : Connectionless Service

- 초기 인터넷 디자인에서는 네트워크 레이어가 Connectionless Service로 디자인 됨

- 패킷 하나하나를 독립적으로 보며, 패킷 간의 관계가 없는것으로 취급
- 패킷을 운송하는데에만 역할이 있다고 봄 (SRC -> DST)
- 메시지 안의 패킷이 Destination 방향으로 같은 Path를 이용 할 수도 안할 수도 있음

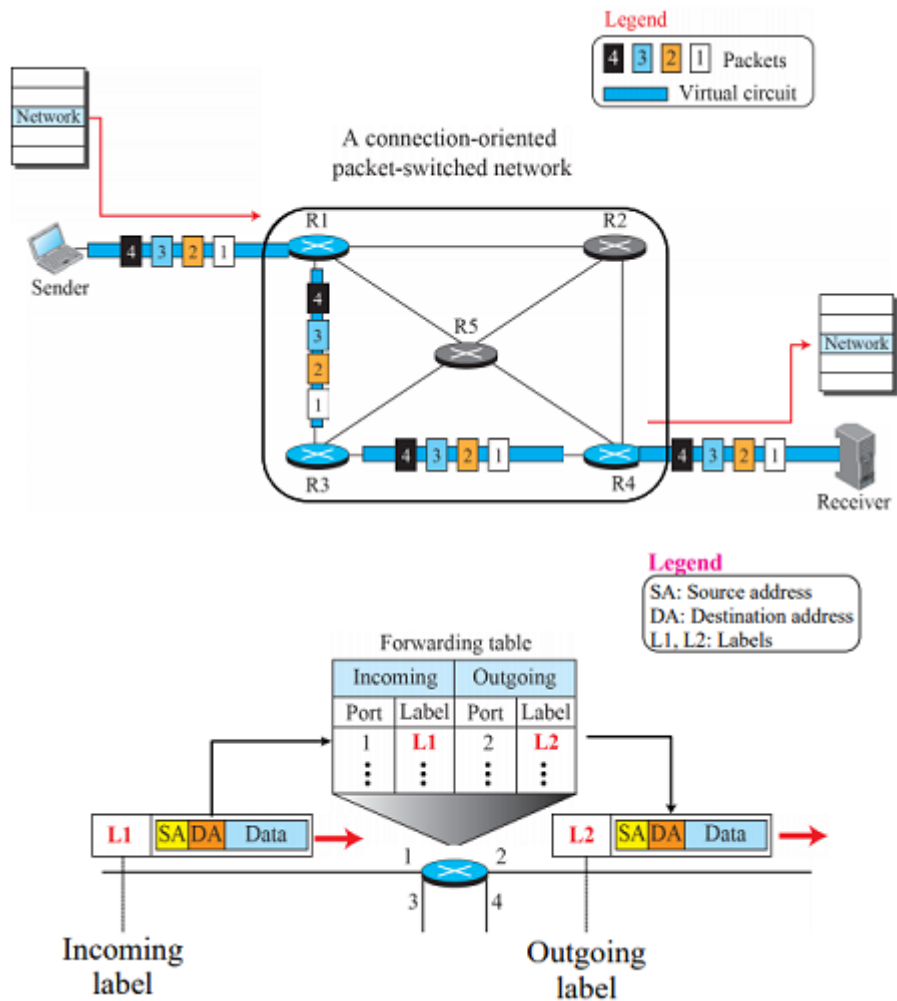


- 어디로 보낼지는 패킷의 목적지 주소가 결정



## Virtual-Circuit Approach : Connection-Oriented Service

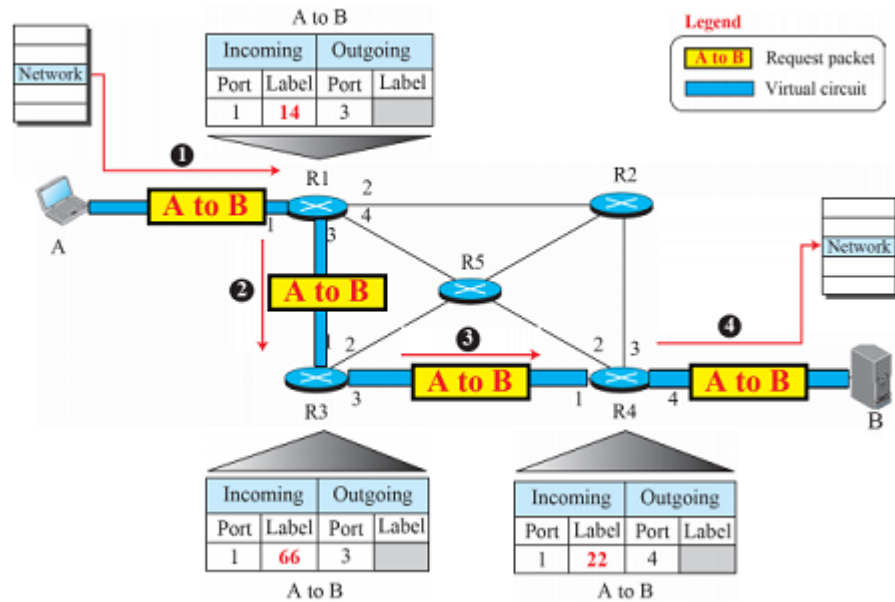
- 메시지에 속하는 패킷들이 모두 관계가 있다
- 데이터그램의 모든 메시지가 전송되기전에, Datagram들의 이동경로가 설정하기 위해 Virtual Connection이 이뤄져야 함
- Connection Setup이 끝나면, 해당 데이터그램들은 같은 이동경로를 이용
- 패킷은 소스와 목적지의 주소는 물론 해당 패킷의 Virtual Path를 정의하는 Flow Label(Virtual Circuit Identifier)도 가지고 있어야 함



- CO Service를 만들기 위해서 3단계가 프로세스가 필요함

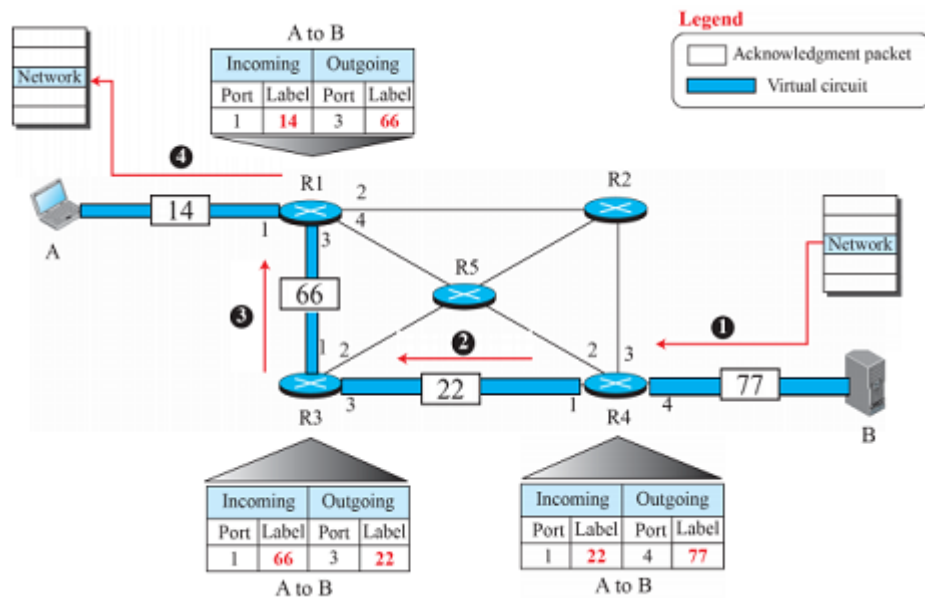
## Setup Phase

- 라우터가 Virtual Circuit을 위해 Entry를 만듦
- 두개의 예비 패킷이 Sender와 Receiver 사이에서 교환되어야 함
  - Request Packet
    - 소스 -> 목적지
    - 소스와 목적지 주소 포함
    - 라우터들은 outgoing part를 정해주고, 최종목적지에서 Label을 지정해줌



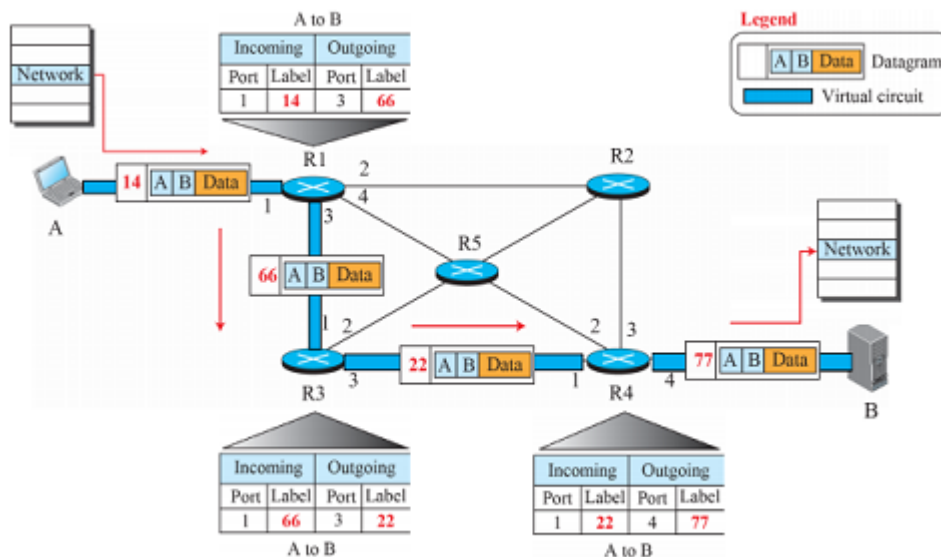
#### ACK Packet

- Switching Table에서 Entry를 마무리 지음
- 라우터 별로 Incoming/Outgoing Label이 달라짐



## Data-Transfer Phase

- 모든 라우터가 특정 Virtual Circuit을 위한 Forwarding Table을 만든 후에, 한 메시지에 속하는 네트워크 레이어 패킷들을 보낼 수 있게 됨



## Teardown Phase

- 모든 패킷을 다 보내고 난 뒤, 소스가 Teardown Packet을 보냄
- 목적지는 이를 받고 Confirmation Packet을 보냄
- 이를 확인한 라우터들은 해당하는 Entry들을 테이블에서 삭제함

## Network-Layer Performance

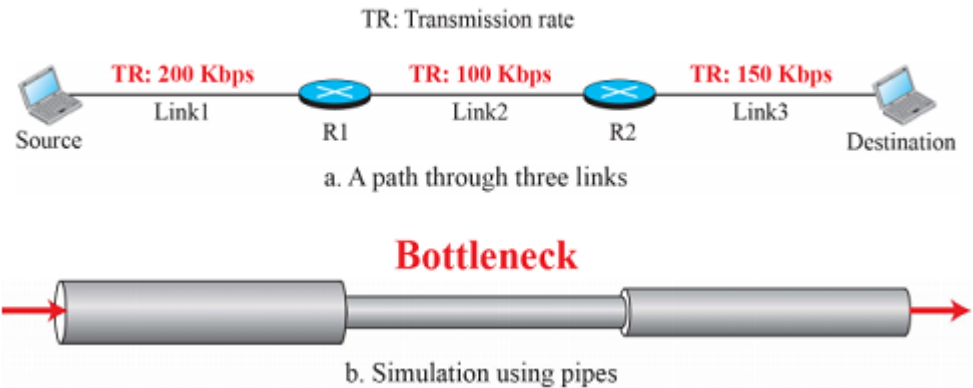
- 네트워크 프로콜을 사용하는 위쪽의 프로토콜에서는 이상적인 서비스를 원하지만 네트워크 레이어는 완벽하지 못함
- 성능을 Delay, Throughput, Packet Loss로 측정 가능
- Congestion Control은 성능을 향상시킬 수 있는 요소

## Delay

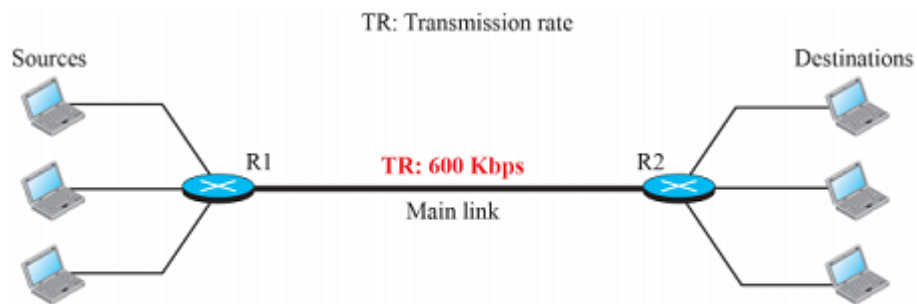
- 네트워크로부터 즉각적인 응답을 원하지만, 소스에서 목적지로 이동하며 딜레이 발생
- 4개의 type으로 구분
  - Transmission Delay = packet length / transmission rate
  - Propagation Delay = distance / propagation speed
  - Processing Delay = 라우터/목적지 호스트에서 패킷을 가공하는데 소모되는 시간
  - Queuing Delay = 패킷이 라우터의 Input/Output 큐에서 기다리는 시간
- Total Delay : 라우터 수를 알면 계산 할 수 있음
  - $(n+1)(\text{Delay}_{tr} + \text{Delay}_{pg} + \text{Delay}_{pr}) + (n)(\text{Delay}_{qu})$
  - n개의 라우터가 있으면 n+1개의 링크가 있음

## Throughput

- 1초안에 특정 지점을 지나가는 비트의 수
- 패킷은 Path를 지나며 각자 다른 링크, 다른 Transmission Rate를 겪음
- 전체 Path의 Throughput 측정
  - $\text{Throughput} = \min \{TR_1, TR_2, \dots, TR_n\}$
  - 평균 Throughput은 Bottleneck으로 결정됨



- 메인 링크의 Transmission Rate는 200, 3개로 나뉘지기 때문(18.12)



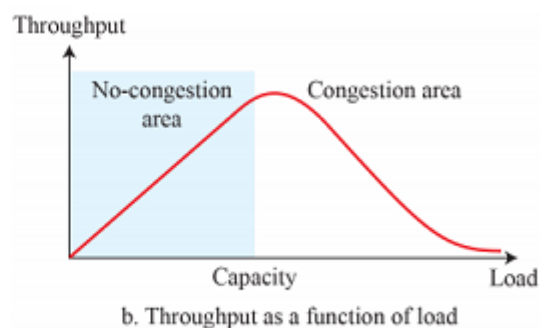
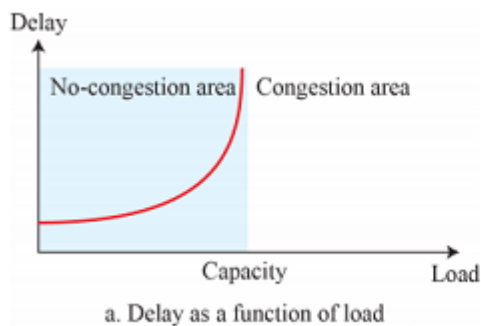
**Figure 18.12:** Effect of throughput in shared links

## Packet Loss

- 통신 성능에 심각한 영향을 미치는 문제점
  - 통신 도중 잃어버리는 패킷의 수
- 라우터가 하나의 패킷을 가공하는 도중 다른 패킷을 받으면, 버퍼에 임시로 저장해둬야 함
  - 버퍼의 크기가 제한적임
  - 버퍼가 가득차면 다음 패킷은 DROP
  - 재송신을 해야하고, 오버플로우 발생에 따른 더 많은 Packet Loss로 이어짐

## Congestion Control

- 성능을 향상시키기 위한 메커니즘
- 인터넷 모델에서는 명쾌하게 다뤄지지 않음
- Throughput, Delay와 관련됨



- 네트워크의 용량보다 Load가 훨씬 작으면, 딜레이가 최소화

- Propagation Delay, Processing Delay으로 구성(무시해도 될 정도)
- Load가 네트워크의 용량을 채우게 되면, 딜레이가 확 늘어남(Queuing Delay 추가)
  - Load가 네트워크의 용량을 넘어서면 딜레이가 무한대가 됨
- 네트워크의 용량보다 Load가 작으면, Throughput이 Load에 비례해서 상승
- 네트워크의 용량보다 Load가 커지면, Throughput이 급락
  - 라우터가 패킷을 버려야 함
  - 소스는 계속해서 재송신하기 때문에, 패킷 수를 줄여주지 않음
- Congestion Control : Congestion을 예방하거나, 생겼을때 없애는 것

## Open-loop Congestion Control (Prevention)

- Congestion이 발생하기 전에 방지
- 소스 또는 목적지가 이를 Handle

### Retransmission Policies

- Sender가 보낸 데이터가 없어지거나 손상되었다고 생각되면 재송신
- retx policy / timer 설계로 효율성 최적화 해야함

### Window Policies

- Sender쪽에서의 window type
- Selective Repeat Window가 Go-Back-N window보다 나음

### Acknowledgement Policies

- ACK또한 네트워크에서 Load의 한 부분임
- 적은 ACK -> 적은 Load
- Piggybacking

### Discarding Policies

- 잘 버리기
- 덜 예민한 패킷을 버려야 함

### Admission Policies

- QoS Mechanism
- Virtual-Circuit Network의 Congestion을 예방
  - Flow에 포함된 Switch들이 Requirement를 먼저 체크함
  - 잠재적인 Congestion이 있거나 이미 Congestion이 있는 경우 Virtual-Circuit Connection을 Deny

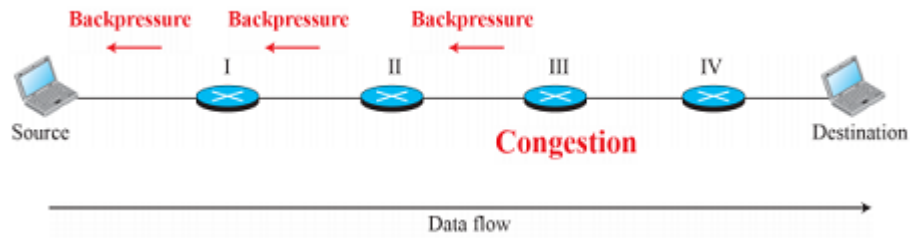
## Close-loop Congestion Control (Removal)

- Congestion이 발생한 이후 이를 완화하는것이 목표

### Backpressure

- 더이상의 데이터를 받는 것을 중지함
- Node-to-Node로 시작해서 source까지 전송됨
- **Virtual Circuit Network에만** 적용가능(Upstream Node가 누군지 아는애들)
- Pressure가 Source쪽으로 옮겨지는 것(시간이 지나면서 Congestion 완화)





### Choke Packet

- Congestion이 발생했다고 알리는 패킷
- 라우터에서 직접 소스로 보냄
- 이전 단계의 노드들은 정보를 받지 못함

### Implicit Signaling

- Congested Node, 다른 노드들, 소스간에 통신은 없음
- 소스가 Congestion 발생 여부를 Guess
- ACK가 한동안 없으면 Congest 됐다고 생각 -> 속도 늦춤
  - TCP Congestion Control

### Explicit Signaling

- Congested Node가 명료하게 소스 또는 목적지에 시그널을 보낼 수 있음
- Choke-Packet Method와 다른 개념, 다른 패킷이 사용됨
- Congested 시그널이 데이터와 함께 보내짐(Piggybacking)
- Forward/Backward 양쪽에서 발생 가능함