# 네트워크 계층 1, 2



# Delay가 발생하는 이유

- packet이 처리할 수 있는 양보다 많이 들어왔을 때
- queue안에 공간이 가득 찼는데 packet이 들어온 경우
- packet을 받았을 때 processing delay
- packet이 처리를 기다리는 시간 queueing delay
- 처리되는 곳 까지 이동하는 시간 transmission delay
- link로 올라온 후에 다음 router까지 이동하는데 걸리는 시간 propagation delay

# HTTP

HTTP:hypertext transfer protocol

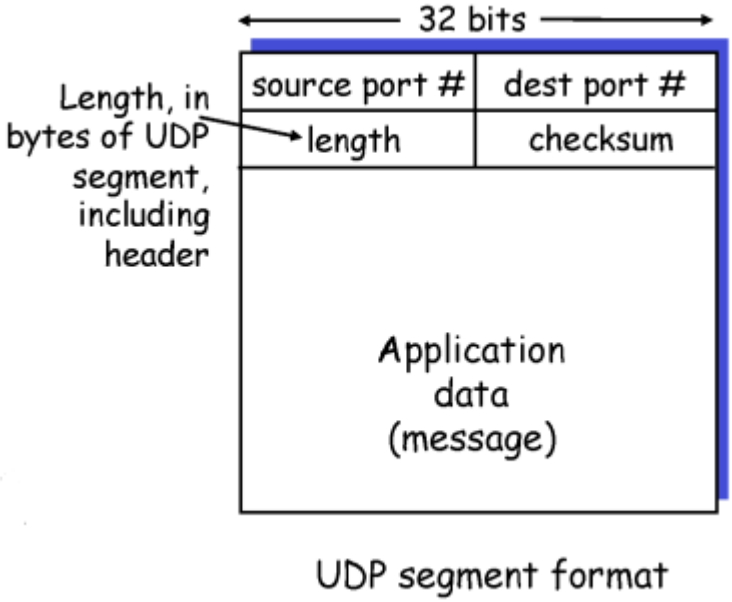

TCP 연결 재사용 유무에 따라 연결 방식 달라짐

- non-persistent HTTP
- persistent HTTP

DNS, Proxy 중요함

- What's end-to-end delay using persistent HTTP?
  - Control messages (e.g. TCP handshake, HTTP request) = **K** bit long
  - Base HTML object = **L** bits
  - **N** reference objects, each **L** bit long
  - Link bandwidth = **R** bps
  - Propagation delay = **d** seconds

# UDP : User Datagram Protocol




UDP segment format

- often used for straming multimedia apps
  - loss tolerant
  - rate sensitive

- other UDP uses
  - DNS
  - SNMP

- reliable transfer over UDP : add reliability ar application layer
  - application-specific error recovery!

checksum 있으므로 error detecting 해줌. data의 무결성 검사

# Principles of Reliable Data Transfer

What can happen over unreliable channel?

- packet error, packet loss

What mechanisms for packet error?

- Error detection, feedback, retransmission, sequence#

What mechanisms for packet loss?

- Timeout

We built simple reliable data transfer protocol

- Real-world protocol(e.g. TCP) is more complex, but with same principles!

# Pipelined protocols

## go-Back-N

동작이 단순함

packet하나가 유실되면  window안에 있는 모든 packet을 재전송함

timer 1개

## selective repeat

유실된 packet만 재전송 함

구현이 어려움

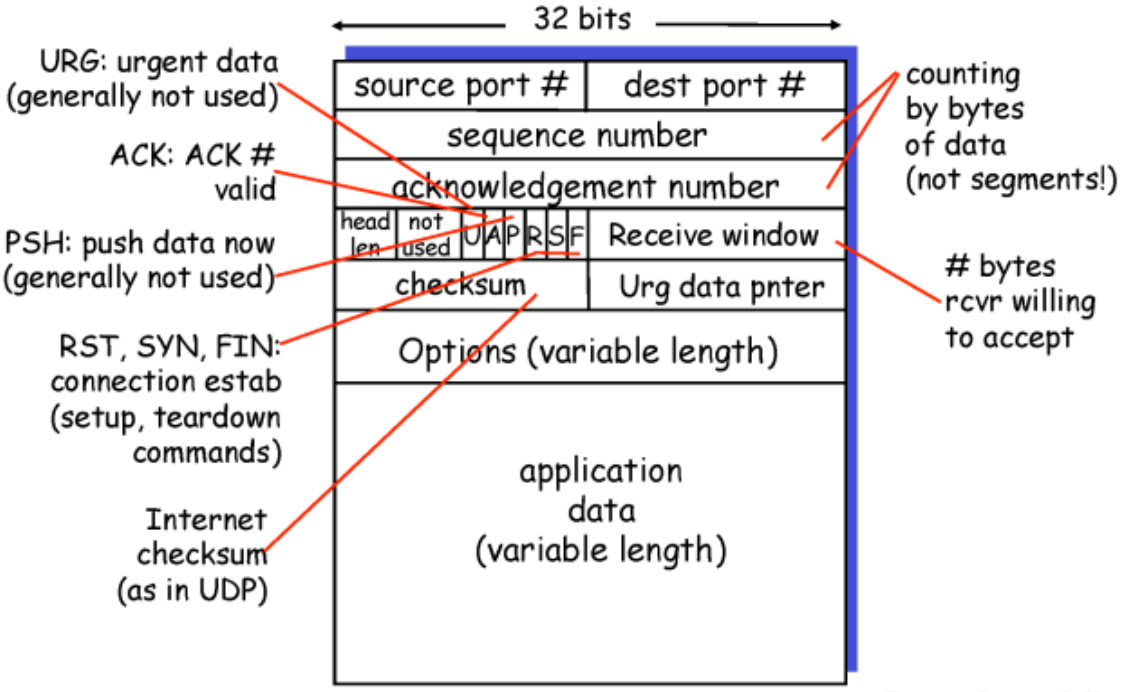
저장공간이 많아야함

# TCP

- point-to-point:
  - one sender, one receiver
- reliable, in-order *byte stream:*
  - no "message boundaries"
- pipelined:
  - TCP congestion and flow control set window size
- *send & receive buffers*


- full duplex data:
  - bi-directional data flow in same connection
  - MSS: maximum segment size
- connection-oriented:
  - handshaking (exchange of control msgs) init's sender, receiver state before data exchange
- flow controlled:
  - sender will not overwhelm receiver








TCP에서 SEQ# 와 ACK#가 가지는 의미 알기

## Fast Retransmit

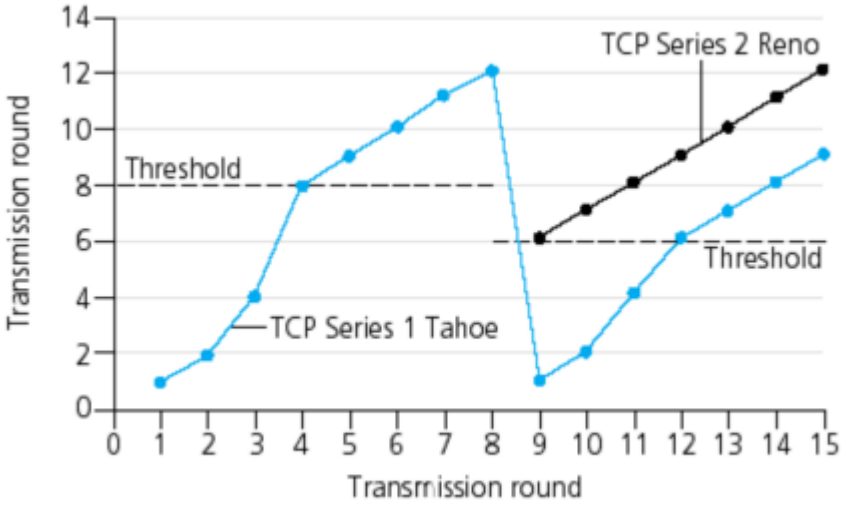
- Time-out period often relatively long:
  - long delay before resending lost packet
- Detect lost segments via duplicate ACKs.
  - Sender often sends many segments back-to-back
  - If segment is lost, there will likely be many duplicate ACKs.
- If sender receives 3 ACKs for the same data, it supposes that segment after ACKed data was lost:
  - fast retransmit: resend segment before timer expires

TCP에서는 같은 번호의 ACK4개를 받으면 time out으로 판단. 재전송 함.

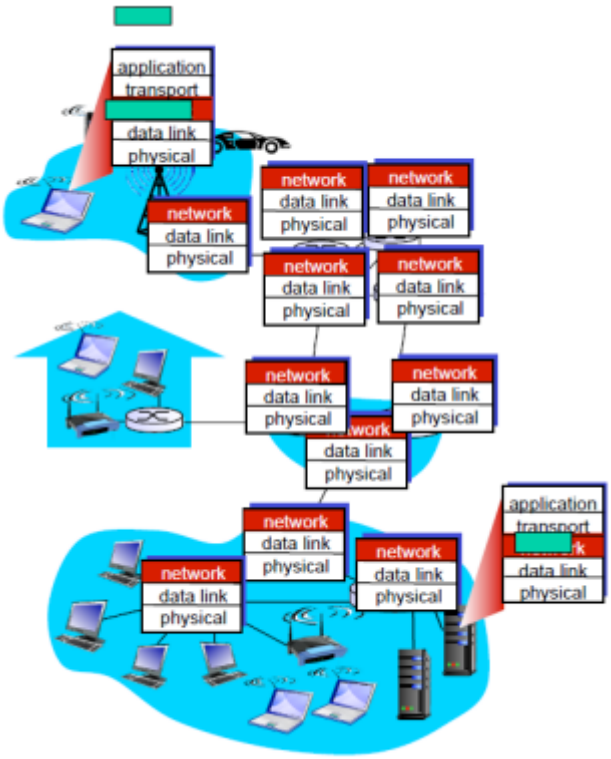
# TCP congestion control

- addictive increase : increase CongWin by 1MSS every RTT until loss detected
- multiplicative decrease : cut CongWin in half after loss

### TCP Tahoe VS TCP Reno

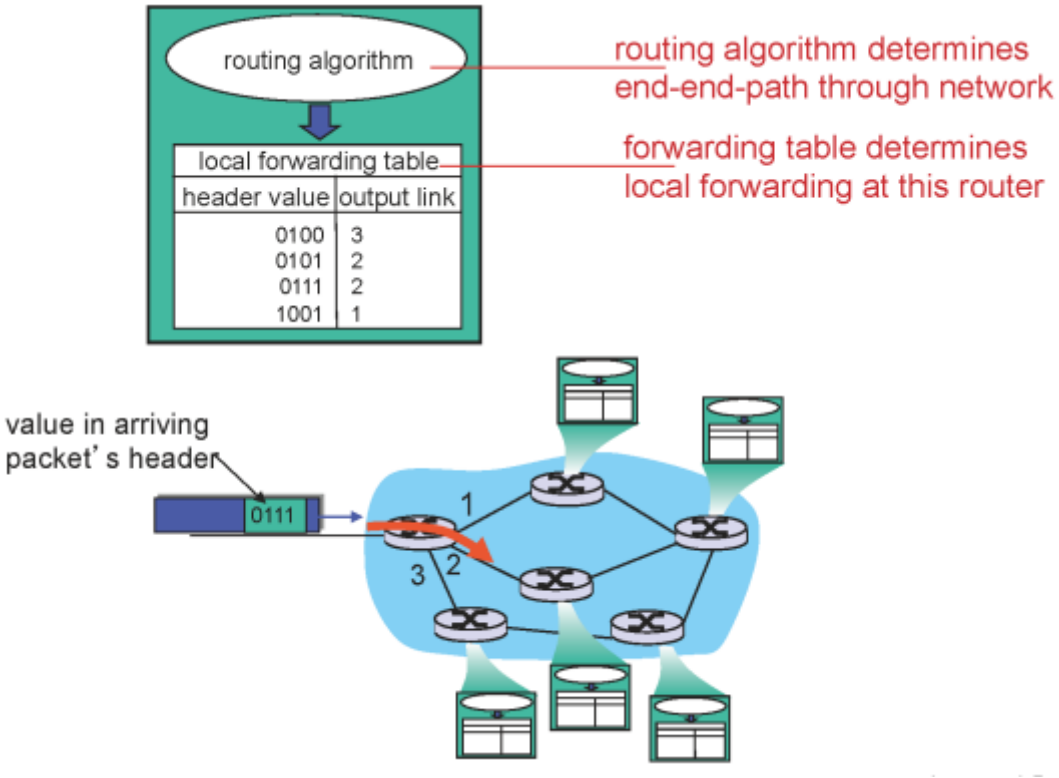



# Network Layer

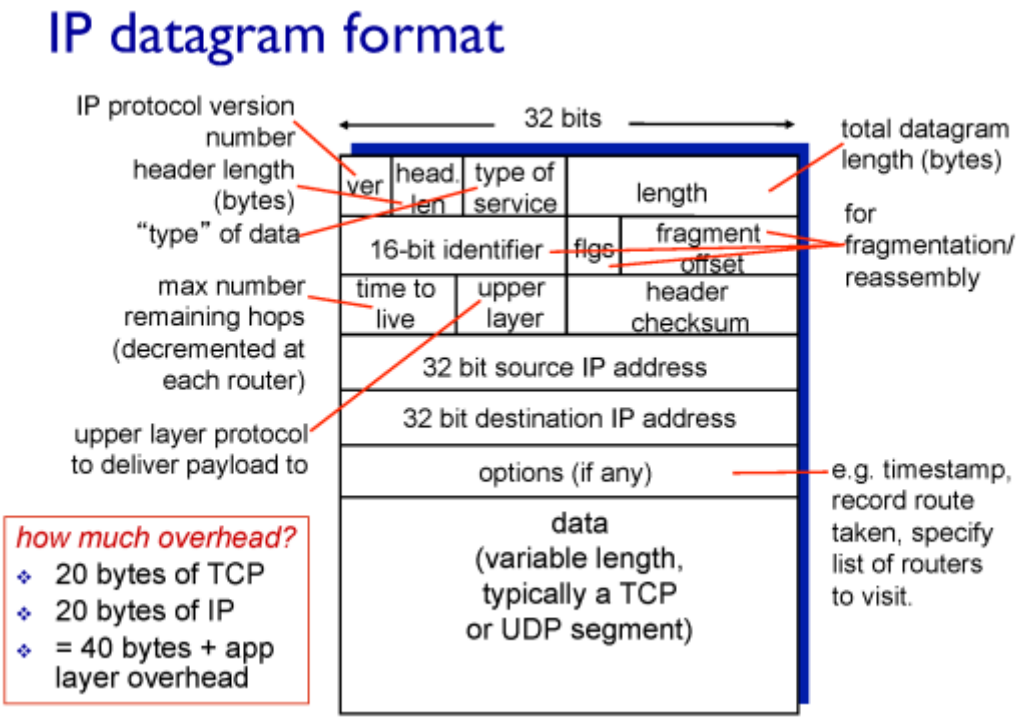


# router의 작업

- forwarding : move packets from router’s input to apptopriate router output
- routing : determine route taken by packets from source to dest



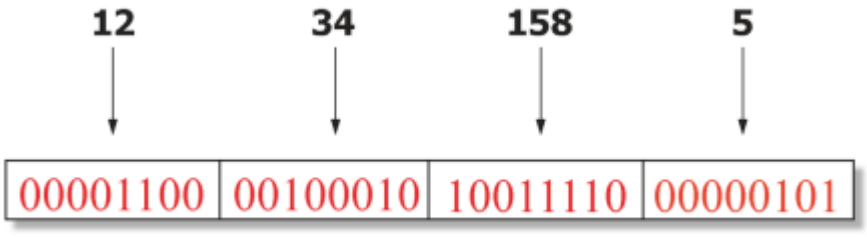

router가 알고리즘을 사용해서 forwarding table을 만듦



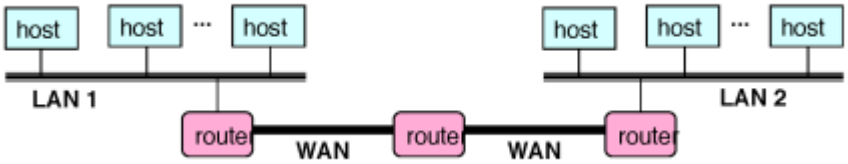

header만 40byte

# IP Address (IPv4)

- Aunique 32-bit number
- Identifies an interface (on a host, on a router, …)
- Represented in dotted-quad notation




## Grouping Realated Hosts

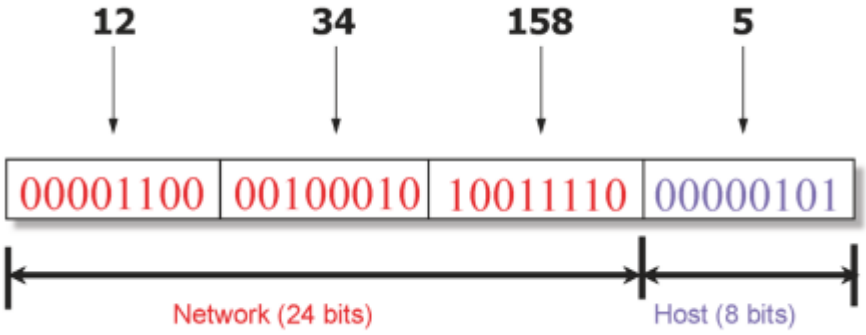



LAN = Local Area Network

WAN = Wide Area Network

주소를 아무렇게나 지정하면 너무 복잡해짐

## Hierarchical Addressing : IP Prefixes



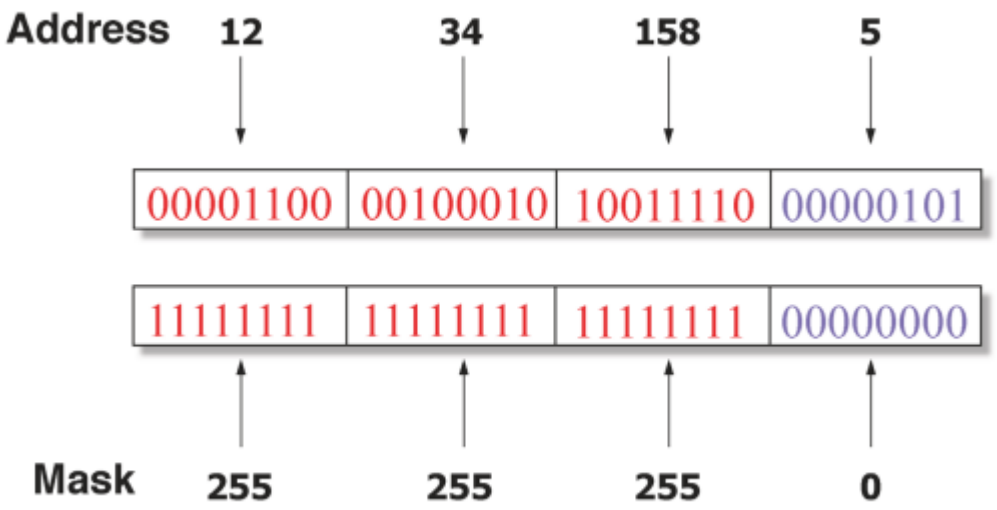

같은 Network에 속한 IP의 주소는 같은 Network ID를 사용함

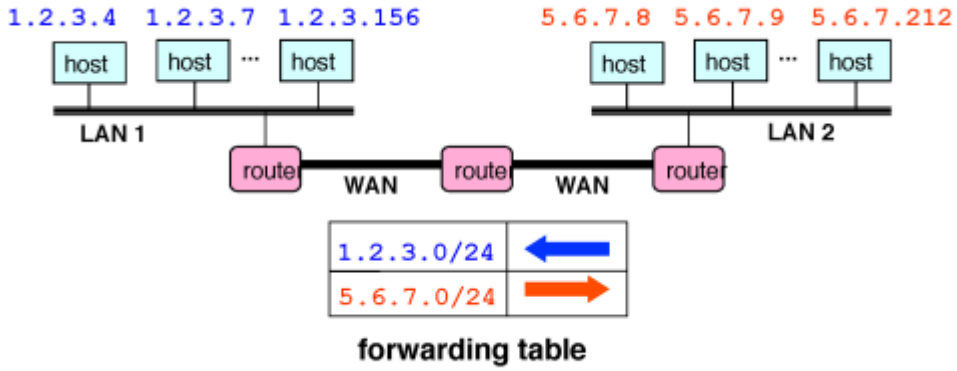
Network ID (prefix , subnet) 24bits

Host ID 8 bits

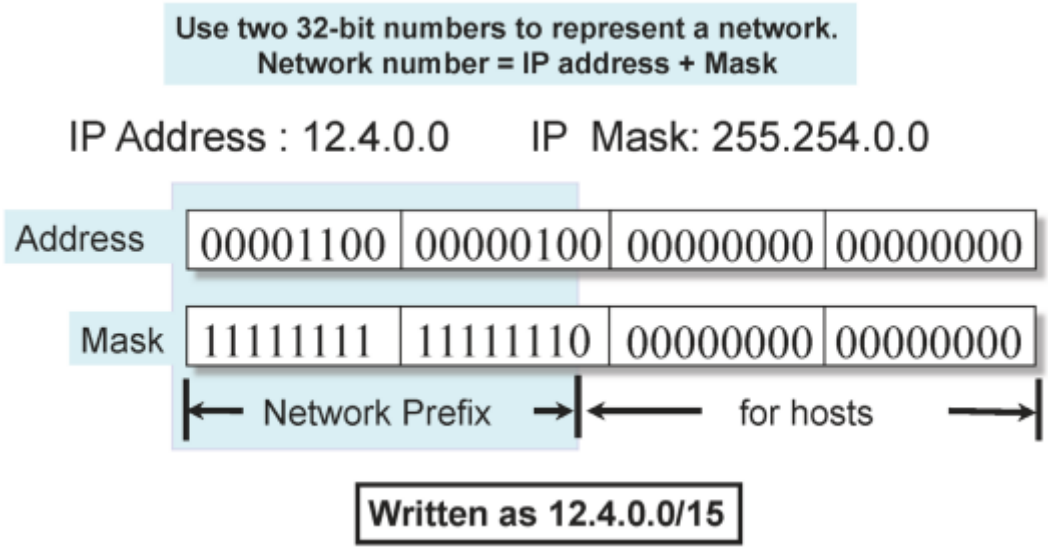
인 경우 밑과 같은 24 bits Submet Mask를 가짐

## IP Address and 24-bit Subnet Mask



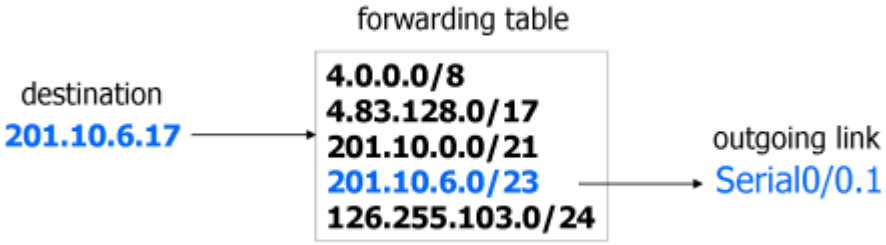

## Classless Inter-Domain Routing (CIDR)



정해져 있는? Prefix를 사용할 경우 너무 비효율적임.

유연한 Prefix 사용

## Longest Prefix Match Forwarding



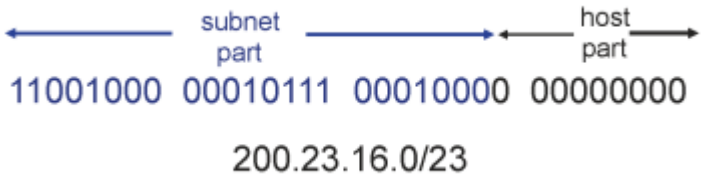

3번째, 4번째 matching가능

가장 구체적으로 matching 되는 것 → prefix의 크기가 가장 큰 것을 찾아서 matching

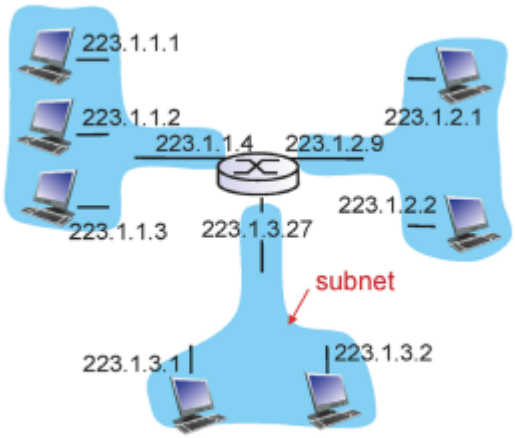
⇒ “Longest Prefix Mach Forwarding”

## IP addressing : CIDR

CIDR : Classless Inter Domain Routing




## Subnets

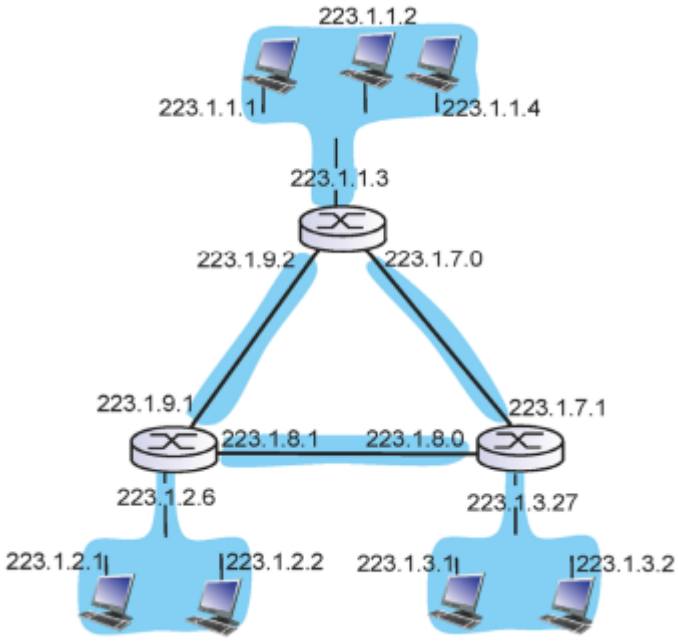


network consisting of 3 subnets

subnet : router를 거치지 않고 접근이 가능한 집합

⇒ IP주소의 prefix가 같음

router는 여러개의 subnet에 속함



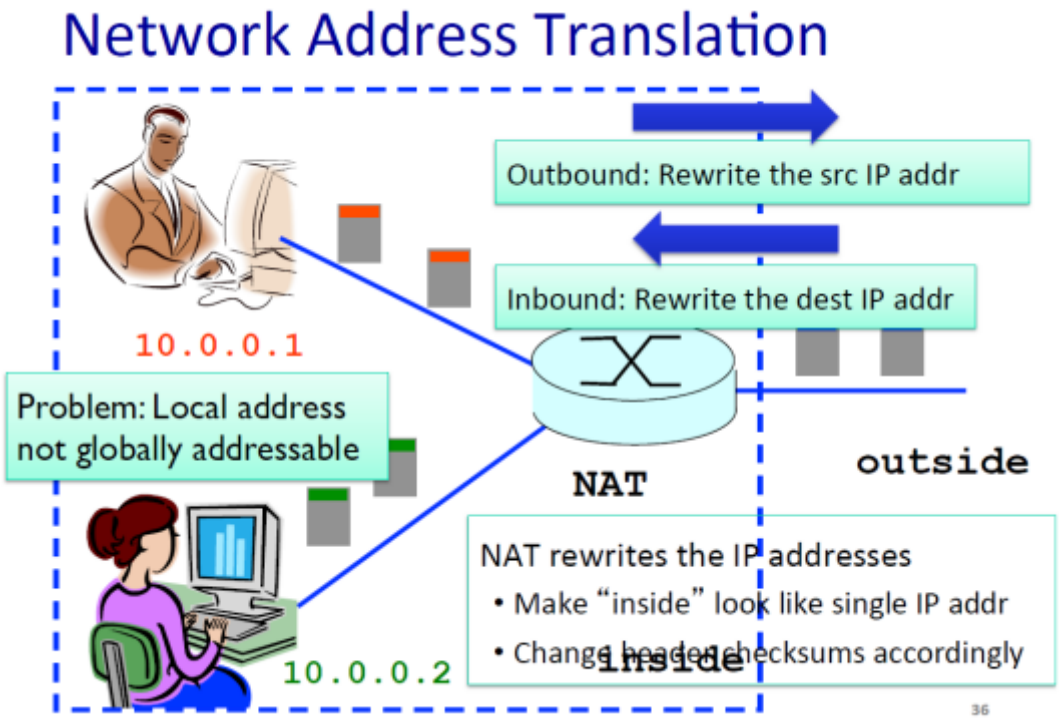

⇒ Subnet 6개

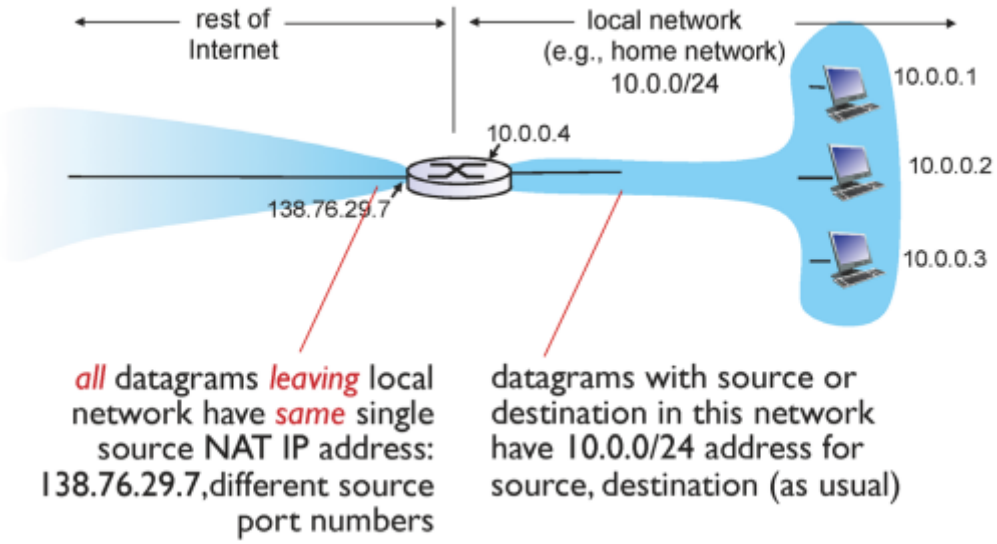
오래전 - IPv4 : 32bit

조금전 - IPv6 : 주소공간을 128bit로 늘림

최근 - IPv4 : 32bit의 주소공간을 나눠서 재활용해서 사용. 주소공간이 부족한 것을 근본적으로 해결하지는 x

# Network Address Translation (NAT)






NAT를 사용하면 내부 network에서는 서버를 사용할 수 없음