네트워크 계층 3, 4

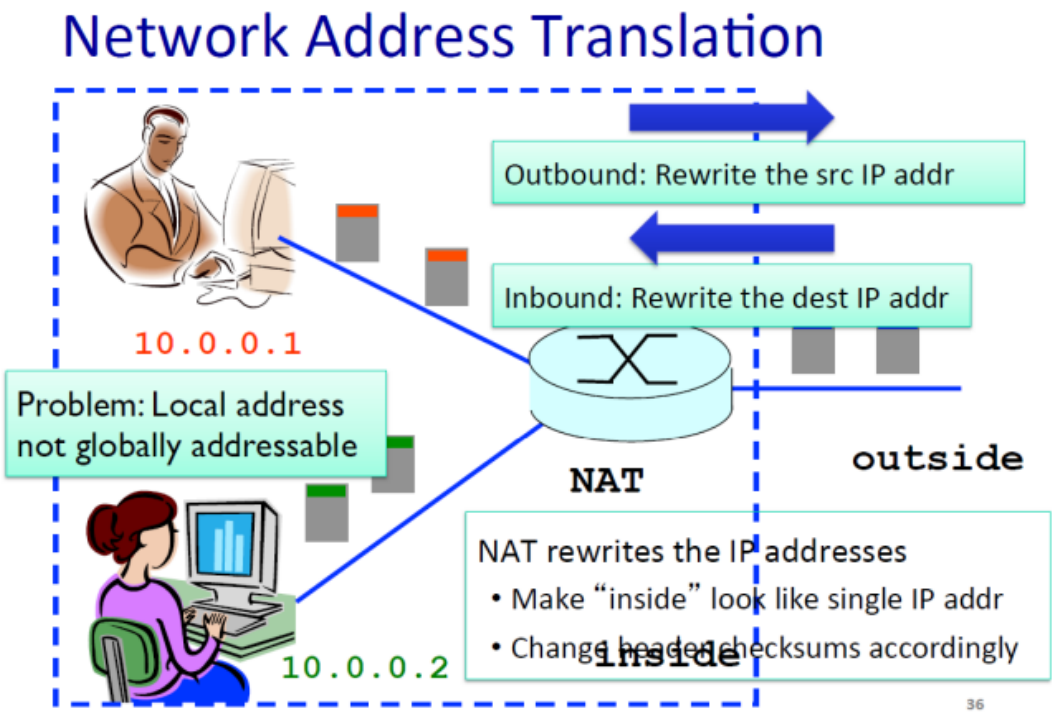
Network Address Translation (NAT)
    NAT : network address tranlation
Dynamic Host Configurations Protocol (DHCP)
ICMP : Internet Control Message Protocol
IPv6
    tunneling
Routing Algorithms
    Graph abstraction
        1. 네트워크 상황이 모두 파악이 가능한 경우 → “link state” algorithm
        2. 이웃 node간의 정보만 알 수 있는 경우 → “distance vector” algorithm

Network Address Translation (NAT)

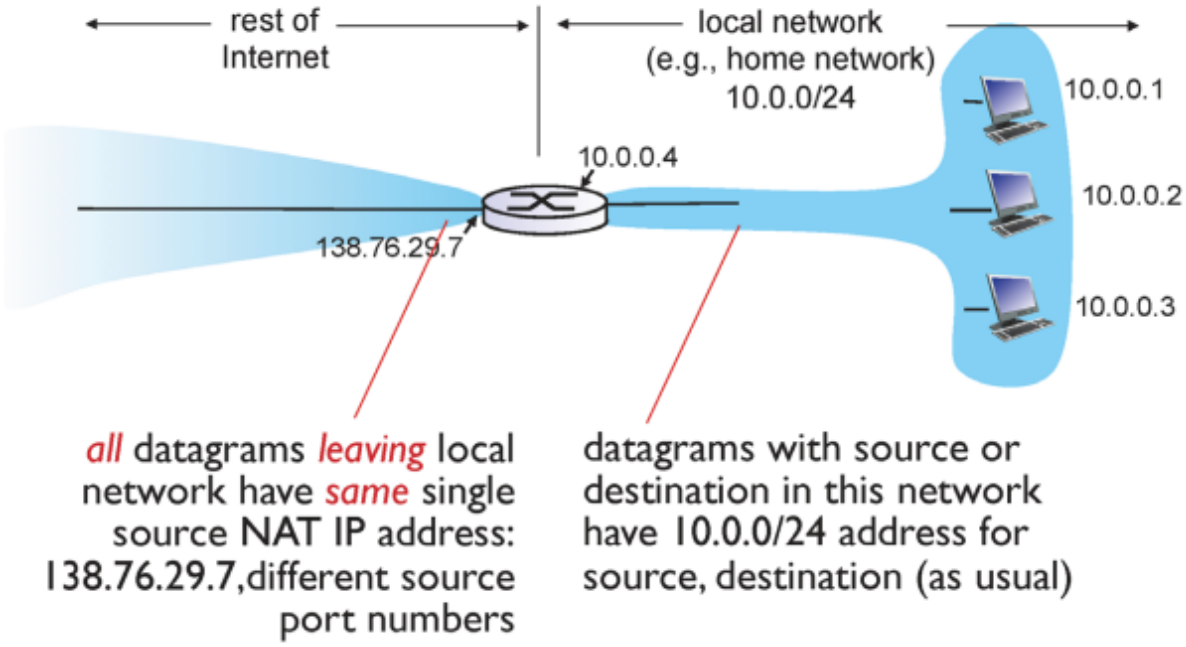
같은 네트워크 안에서 사용하는 주소와 밖에서 사용하는 주소를 다르게 하여 한정적인 주소 자원을 많이 사용할 수 있게 하는 방식.

안에서 사용하던 주소가 밖으로 나가면 발생하는 문제?

→ 다시 돌아 올 수가 없음… ⇒ Gateway Router가 이 문제를 해결해 줌



NAT : network address tranlation



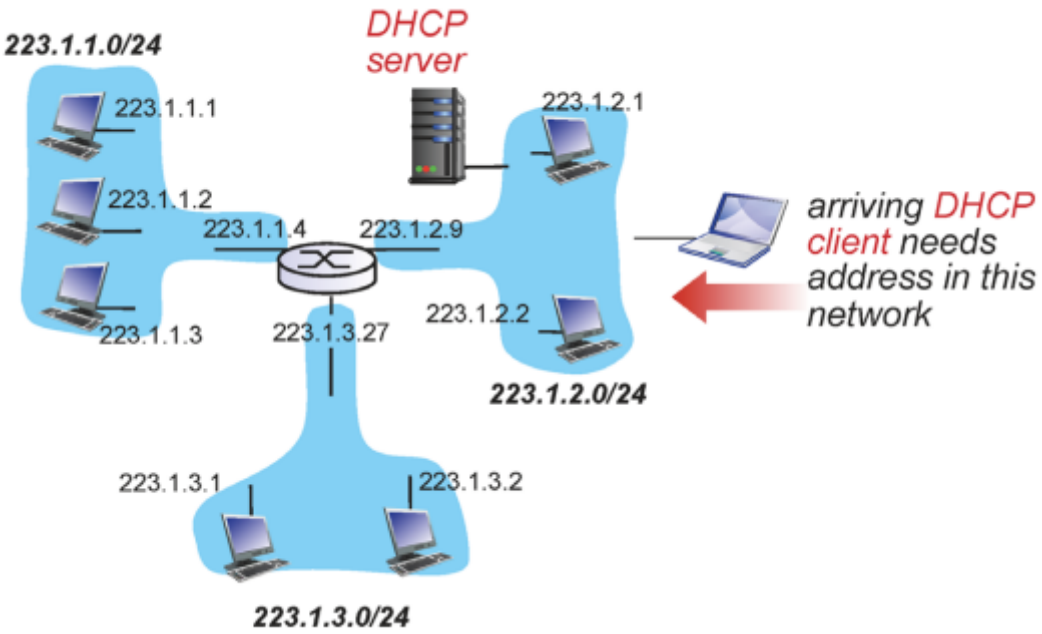
local network → rest of Internet으로 data가 나갈 때 IP주소가 바뀜

IPv4 : 주소 공간 부족, 보안 문제

IPv6 로 교체 x ? → IPv6가 충분히 신뢰가 가지않고, protocol은 한번 정하면 바꾸기 쉽지 않기 때문

Dynamic Host Configurations Protocol (DHCP)

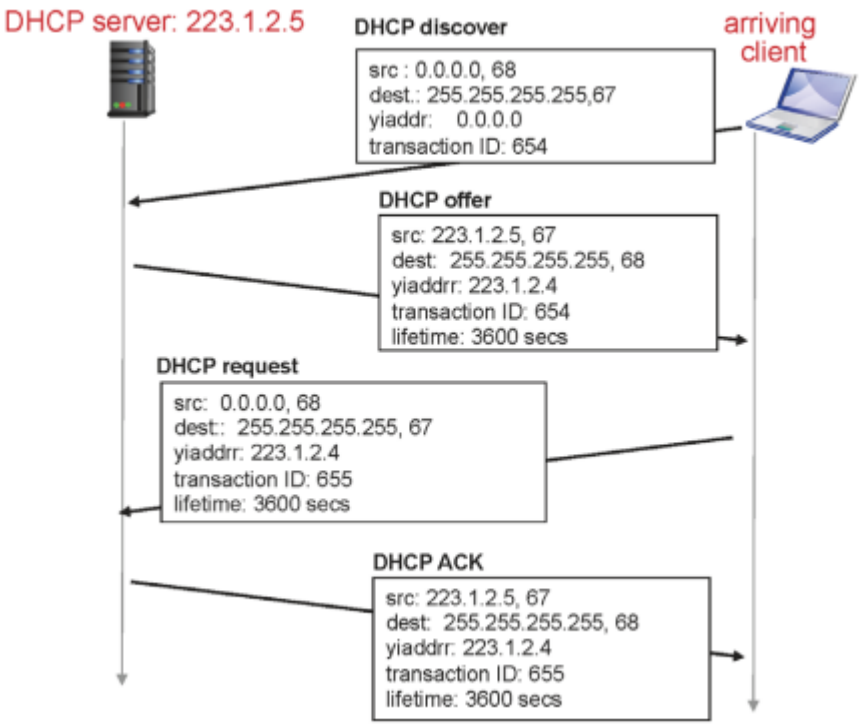
서버로부터 동적으로 IP를 부여받는 configuaration



→ Gateway Router에서 DNS, DHCP Server모두 갖고 있음 (Application Layer)

무선 공유기 : Gateway 역할을 함

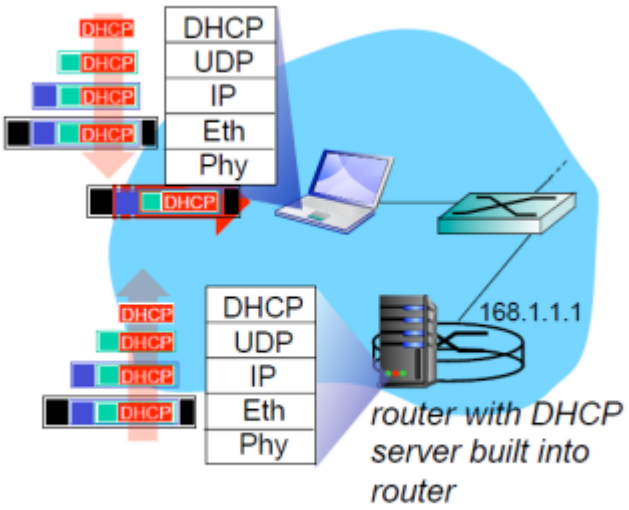
⇒ NAT 굉장히 지저분함.. (router에서 계속 변환 변환..)



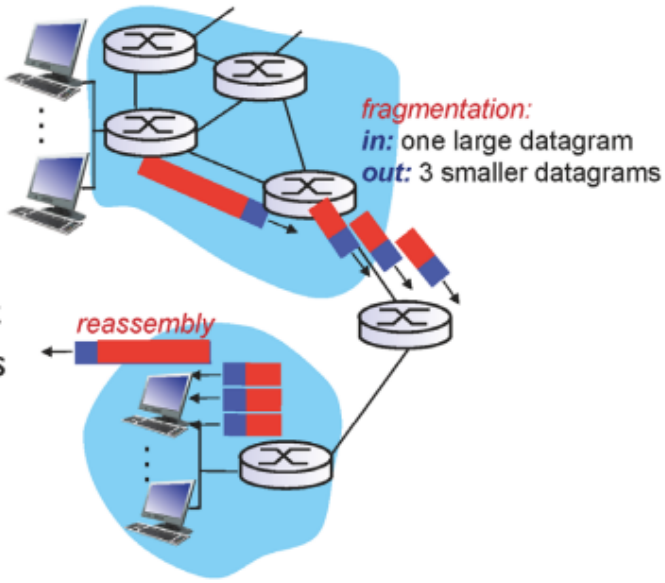
255.255.255.255.0 → 이 서버안에 존재하는 host는 모두 받아라

DHCP offer메세지를 받고 이걸로 하겠다고 확정하는 응답이 꼭 있어야함

다른 애들한테 너네가 선택한걸로 안한다~ 라는 것도 간접적으로 알려주기 위해서 request 다시 모두에게 요청함

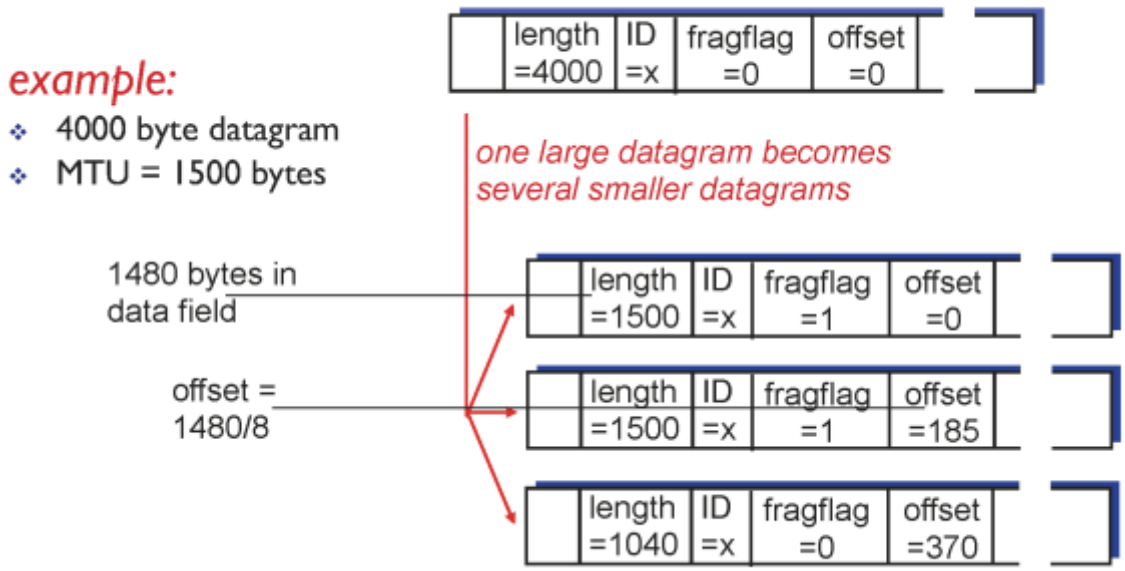


- network links have MTU (max.transfer size) - largest possible link-level frame
  - different link types, different MTUs
- large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams
  - "reassembled" only at final destination
  - IP header bits used to identify, order related fragments



Network Layer 4-59

data가 MTU보다 크면 쪼개서 받음



id : sender가 정함

flragflag : 뒤에 data가 더 있는지

offset 기존의 데이터가 잘려서 중간부터 시작하는 그 지점

ICMP : Internet Control Message Protocol

네트워크에서 일어나는 특정한 event를 source에게 알려주기 위한 message (control message)를 운반하기 위한 protocol

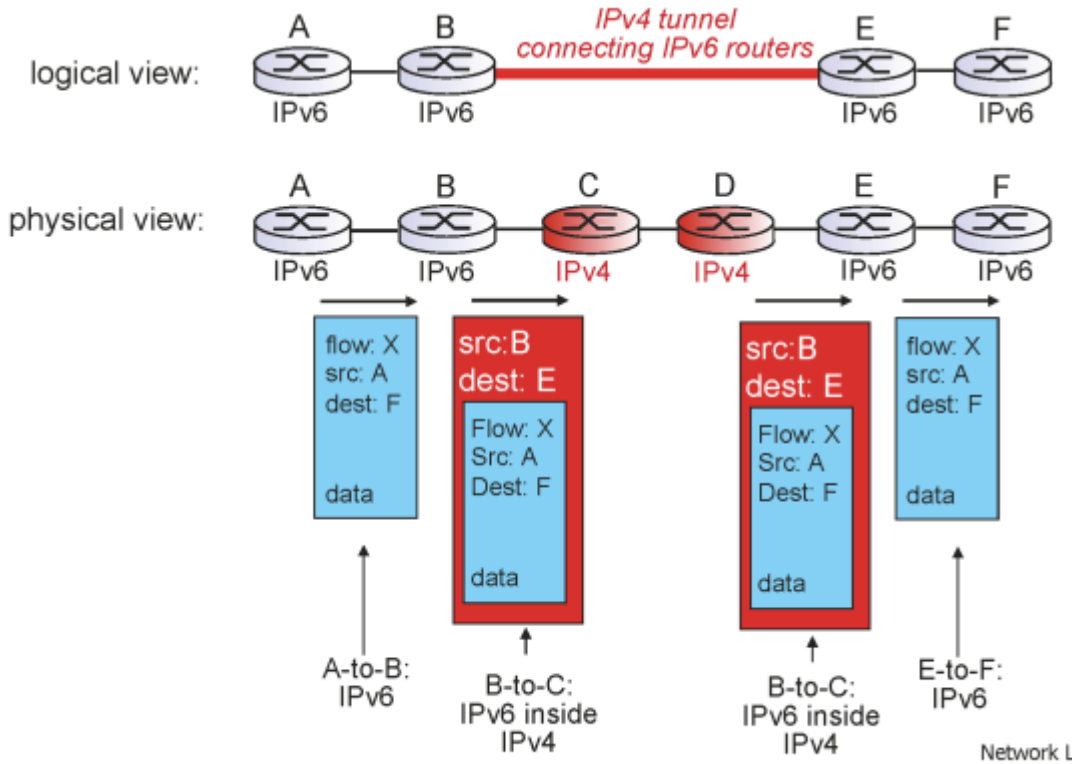
IPv6

address space : 128 bit

헤더가 IPv4보다 간단함

과도기 존재하며, 자연스럽게 넘어갈 것.

tunneling



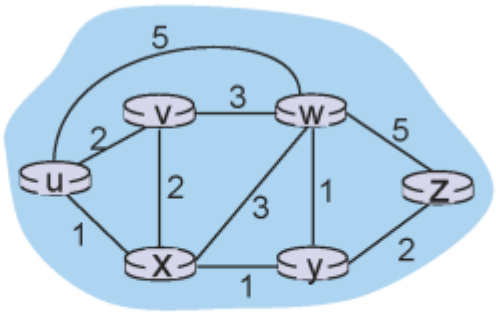
두 버전 사이에 호환을 가능하게 할 tunneling 이 필요함

Routing Algorithms

fowarding 작업 : forwading table을 보고 찾아가는 것에 불과함.

forwarding tabledms routing algorithms을 통해 작성

Graph abstraction



graph: G = (N,E)

N = set of routers = { u, v, w, x, y, z }

E = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

⇒ “최단경로구하기”

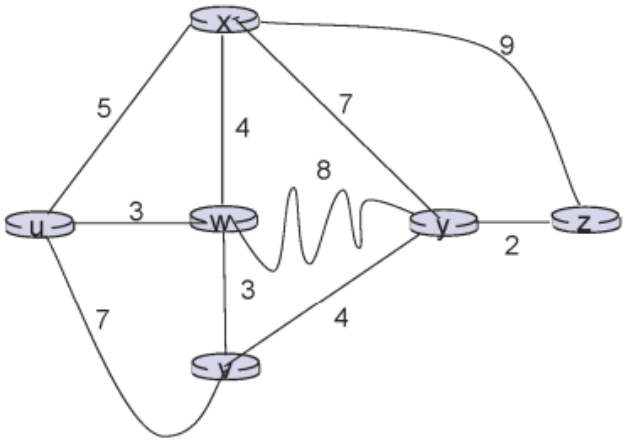
key question: what is the least-cost path between u and z ?
routing algorithm: algorithm that finds that least cost path

1. 네트워크 상황이 모두 파악이 가능한 경우 → “link state” algorithm

2. 이웃 node간의 정보만 알 수 있는 경우 → “distance vector” algorithm

v

Example




```
1  Initialization:
2    N' = {u}
3    for all nodes v
4      if v adjacent to u
5          then D(v) = c(u,v)
6      else D(v) = ∞
7
8  Loop
9     find w not in N' such that D(w) is a minimum
10    add w to N'
11    update D(v) for all v adjacent to w and not in N' :
12       D(v) = min( D(v), D(w) + c(w,v) )
13    /* new cost to v is either old cost to v or known
14     shortest path cost to w plus cost from w to v */
15 until all nodes in N'
```


Dijkstra's algorithm: example

Step	N'	D(v) p(v)	D(w) p(w)	D(x) p(x)	D(y) p(y)	D(z) p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,v
5	uwxvyz					

notes:

- construct shortest path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)