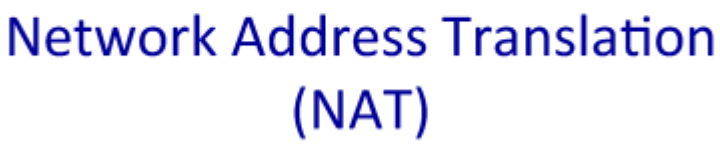
네트워크계층3

작성일시	@2022년 10월 24일 오전 11:38
강의날짜	
편집일시	@2022년 10월 24일 오후 12:49
분야	
공부유형	
복습	☐
태그	



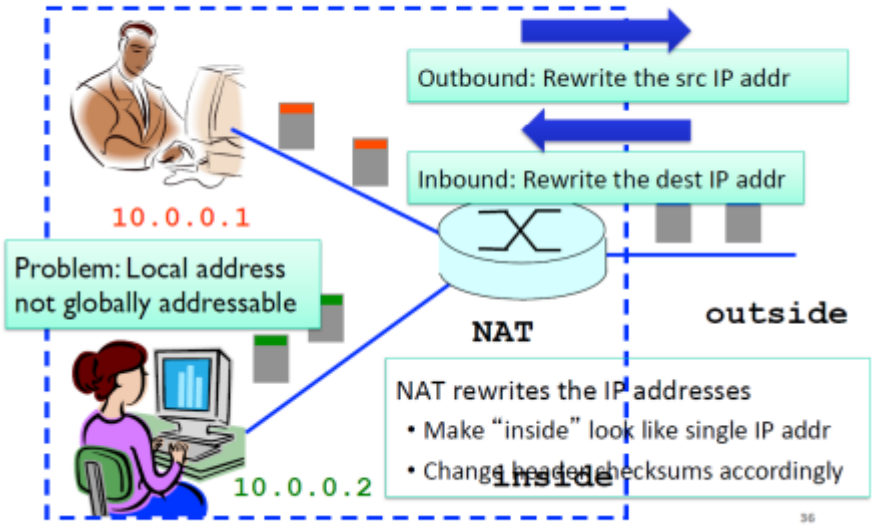
34

History of NATs

- IP address space depletion
  - Clear in early 90s that 2^{32} addresses not enough
  - Work began on a successor to IPv4
- In the meantime…
  - Share addresses among numerous devices
  - … without requiring changes to existing hosts
- Meant as a short-term remedy
  - Now: NAT is widely deployed, much more than IPv6

35

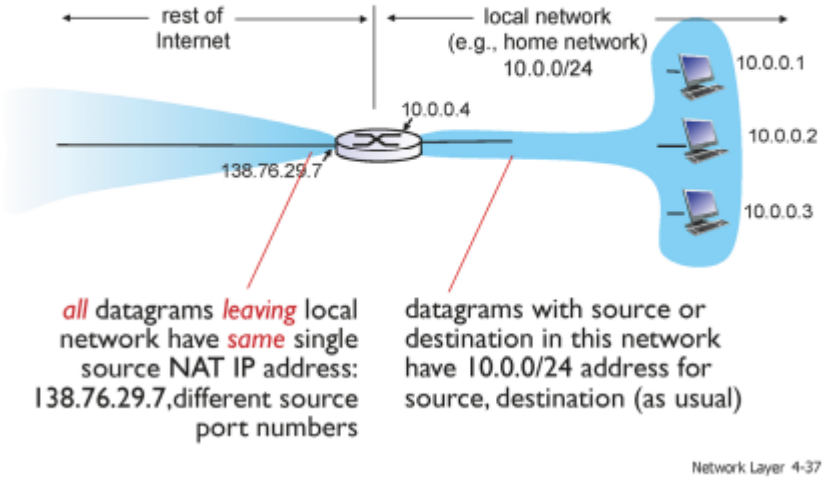
Network Address Translation



36

- 내부적으로 유일한 IP 외부로 나가면 유일하지 않다
- 이러한 ip패킷이 외부로 나가면 돌아올 수가 없음
- 라우터가 ip주소를 바꿔줌 소스 ip → 라우터의 ip
- NAT가 변환작업을 해줌

NAT: network address translation



Network Layer 4-37

- ip 주소는 인터페이스 주소를 지칭 여러 ip 여러 인터페이스
- 라우터가 하는 일 ip주소를 고치고 데이터도 고치고, 레이어링 바이올렛?
- → ip4로는 안되겠다

Principled Objections Against NAT

- Routers are not supposed to look at port #s
  - Network layer should care only about *IP* header
  - … and not be looking at the *port numbers* at all
- NAT violates the end-to-end argument
  - Network nodes should not modify the packets
- IPv6 is a cleaner solution
  - Better to migrate than to limp along with a hack

That's what happens when network puts power in hands of end users!

46

- IP 버전을 바꾸는 것은 일임
- 따라서 좀 더 유연하게 동적으로 변화 가능한 ip프로토콜 만들자
- 라우터는 각자 소유자가 다름 → 생태계적인 문제로 유연한 ip프로토콜 가능할까?

Dynamic Host Configuration Protocol (DHCP)

47

- IP 192 168 1 47
- mask 255 255 255 0
- router 192 168 1 1
- DNS 192 168 1 1
- 기본적인 정보이며, 모두 필요한 정보

IP addresses: how to get one?

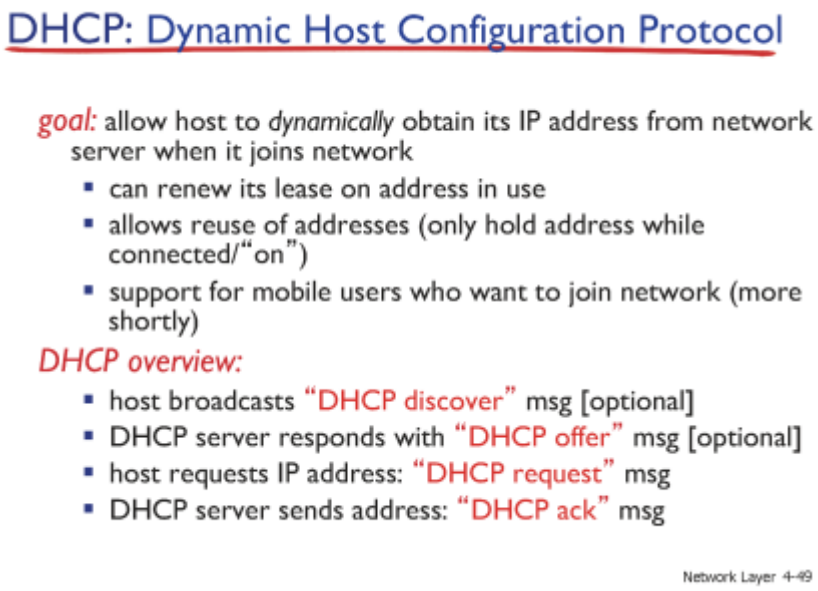
Q: How does a *host* get IP address?

- hard-coded by system admin in a file
  - Windows: control-panel->network->configuration->tcp/ip->properties
  - UNIX: /etc/rc.config
- DHCP: Dynamic Host Configuration Protocol: dynamically get address from as server
  - "plug-and-play"

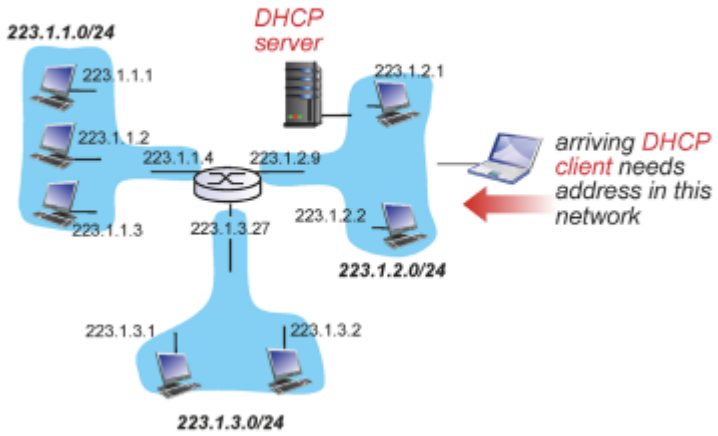
Network Layer 4-48

- DHCP 어디로 가든간에 동적으로 해준다

- 동적으로 해주는 이유: 새로운 곳에 갔을 때 그 지역의 ip 라우터 dns 를 알기 어렵기 때문
- 하지만 항상 그럴 필요가 없다 자기 자신만의 고정된 ip 라우터를 쓰는 경우도 있음 ⇒ static 라우터
- 고정 ip 정책 : 금전적으로 큰 손해
- DHCP : ip 주소 렌탈해주고 회수
  - 고정 ip에 비해 address pool을 유연하게 사용할 수 있음
  - 빌려주고 돌려받고 하는 과정이 필요함

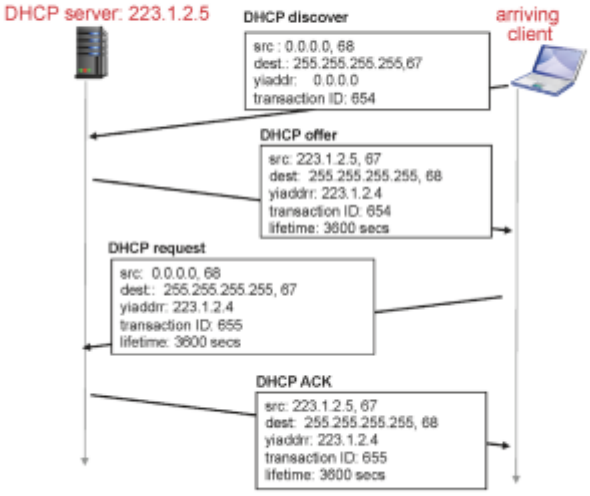


DHCP client-server scenario



Network Layer 4-50

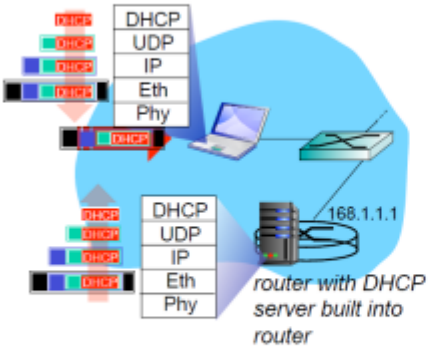
DHCP client-server scenario



Network Layer 4-51

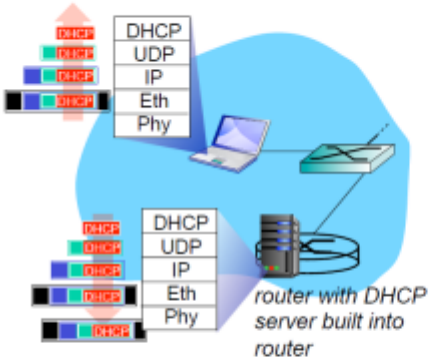
- broadcast : 서브넷에 있는 모든 멤버들은 다 이 메세지를 받아라
- destination 주소 255 255 255기 때문에 다들 반응하게 됨

DHCP: example



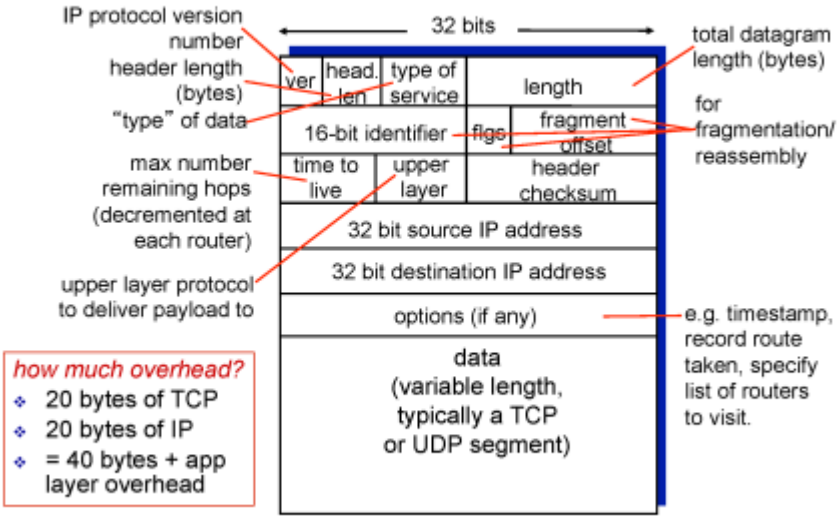
- connecting laptop needs its IP address, addr of first-hop router, addr of DNS server: use DHCP
- DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in 802.1 Ethernet
- Ethernet frame broadcast (dest: FFFFFFFFFFFF) on LAN, received at router running DHCP server
- Ethernet demuxed to IP demuxed, UDP demuxed to DHCP

DHCP: example



- DCP server formulates DHCP ACK containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- encapsulation of DHCP server, frame forwarded to client, demuxing up to DHCP at client
- client now knows its IP address, name and IP address of DSN server, IP address of its first-hop router

IP datagram format

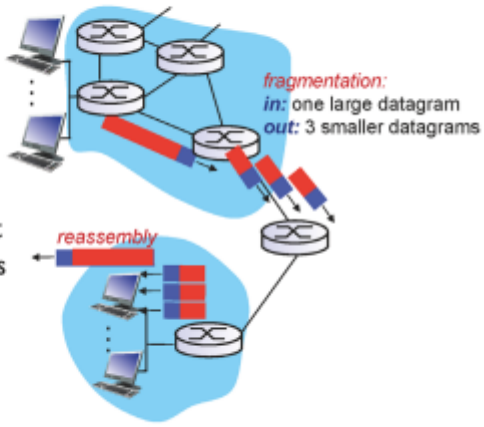


how much overhead?

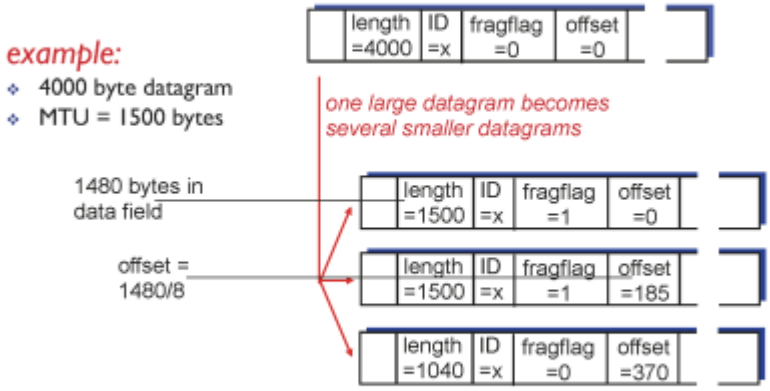
- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + app layer overhead

IP fragmentation, reassembly

- 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



IP fragmentation, reassembly



Network Layer 4-60