

# Advanced Lecture on Internet Infrastructure

## 7. NAT, DHCP

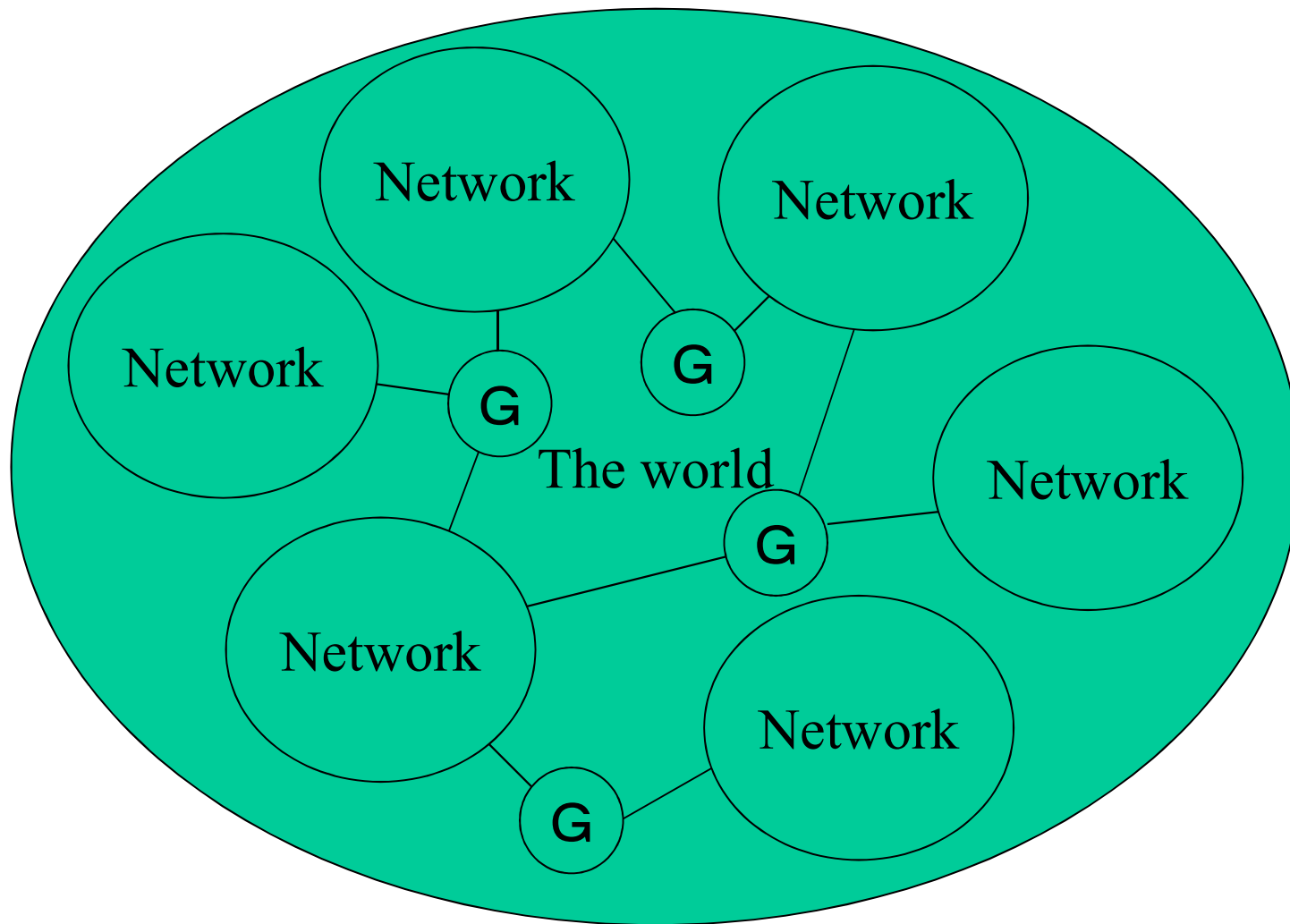
Masataka Ohta

[mohta@necom830.hpcl.titech.ac.jp](mailto:mohta@necom830.hpcl.titech.ac.jp)

<ftp://chacha.hpcl.titech.ac.jp/infra7e.ppt>

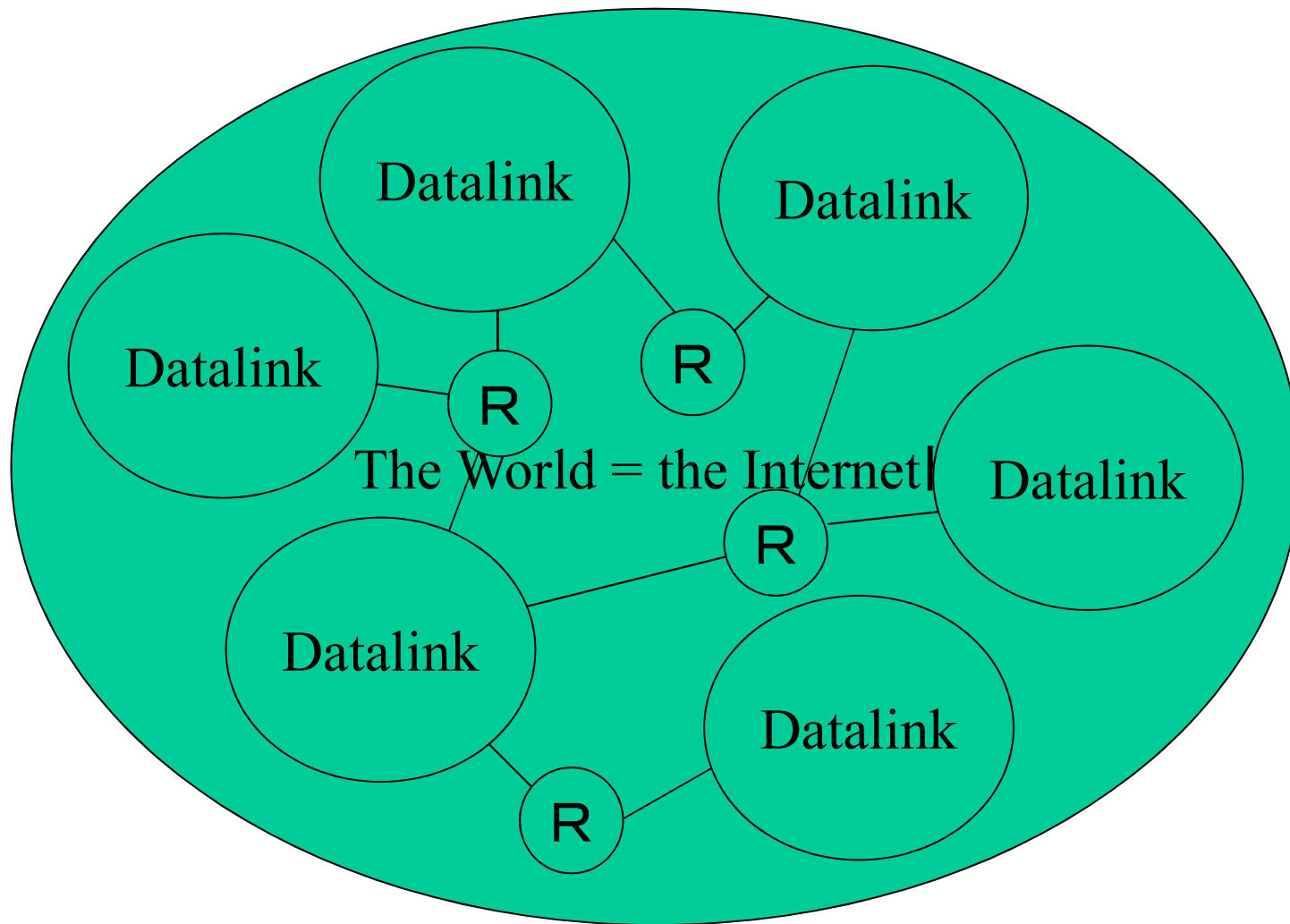
# Structure of the Internet

- CATENET Model
  - Many small (w.r.t. # of devices) datalinks interconnected by IP (Internet Protocol) routers



 : Interworking Unit

The World and Networks (before Internet)



 : Router

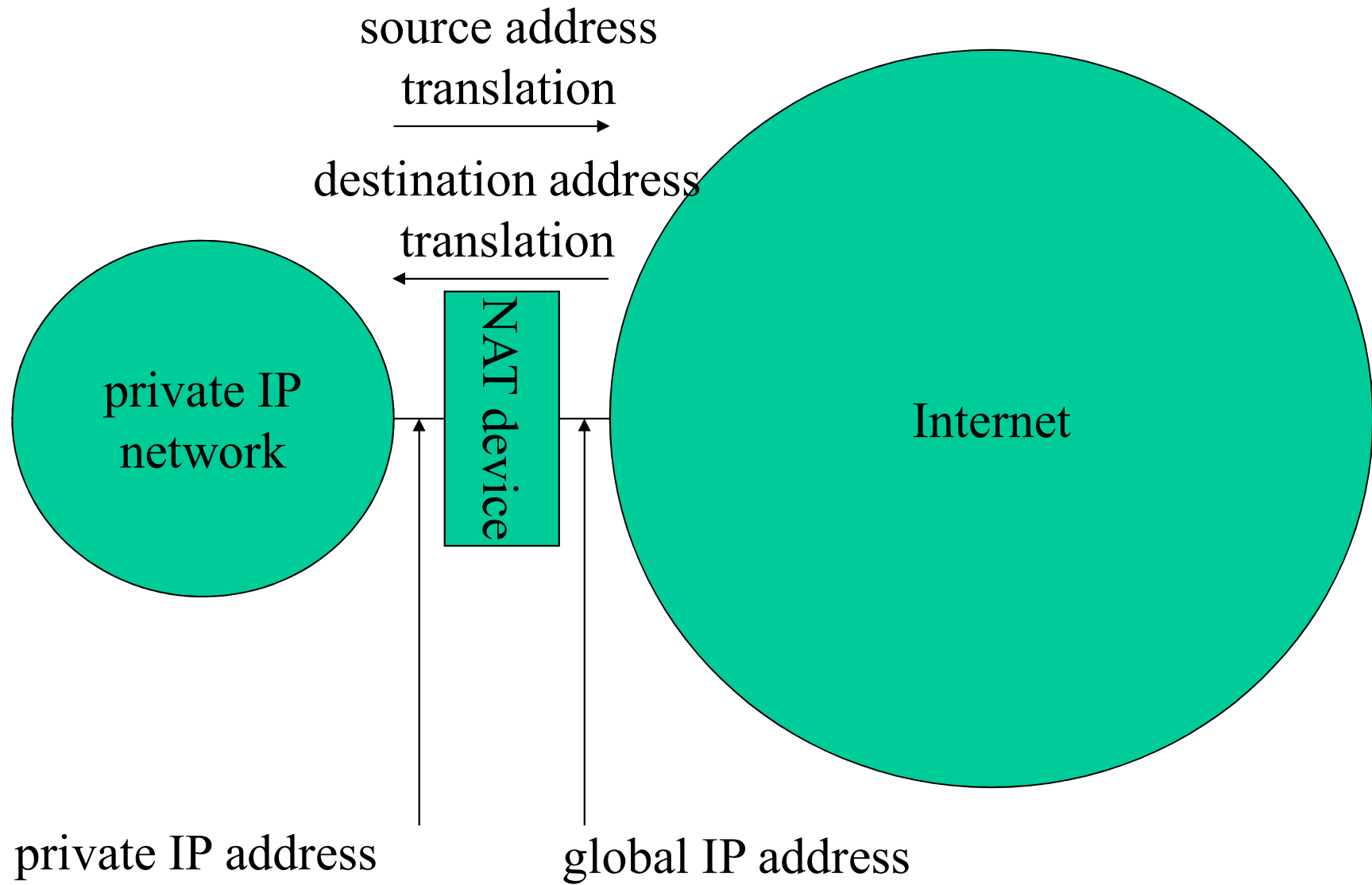
CATENET Model

# NAT (Network Address Translator, rfc1631)

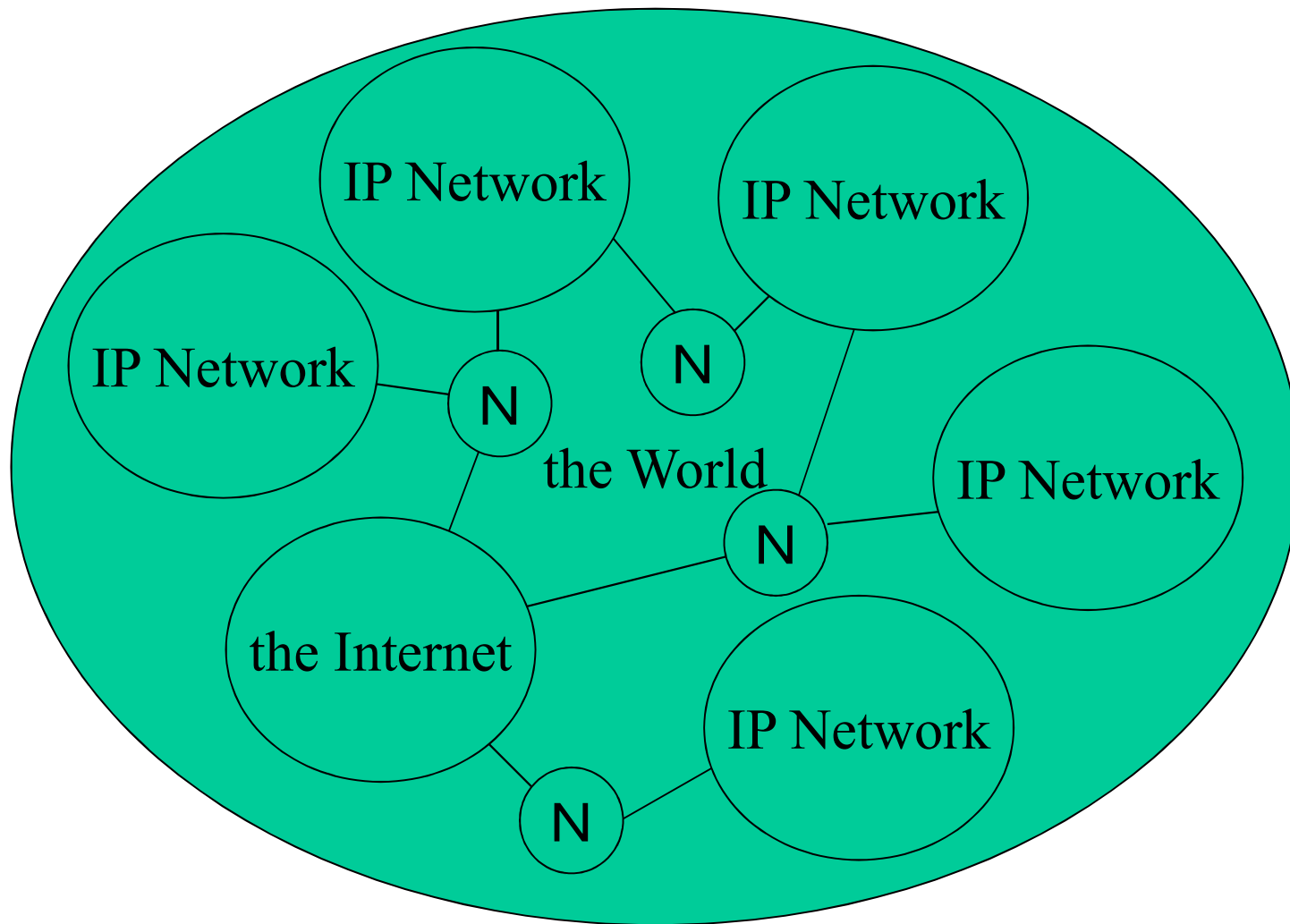
- placed at the border between the Internet and a private IP network
- maintain small # of IP addresses of the Internet
  - (dynamically) correspond private IP addresses and IP addresses of the Internet
  - idea as a measure for IPv4 address shortage
    - not many hosts need fixed IP address? (only servers)

# Dial-up and NAT

- when dial-up internet access was dominant
  - intermittent internet connection was dominant
    - such users can have clients but no servers
    - server is owned by special entity (ISP) and cost a lot
  - even companies with persistent connectivity
    - only needs small number of servers
  - ISP rules the Internet
- (should) not be the Internet in persistent connectivity era
  - cloud?



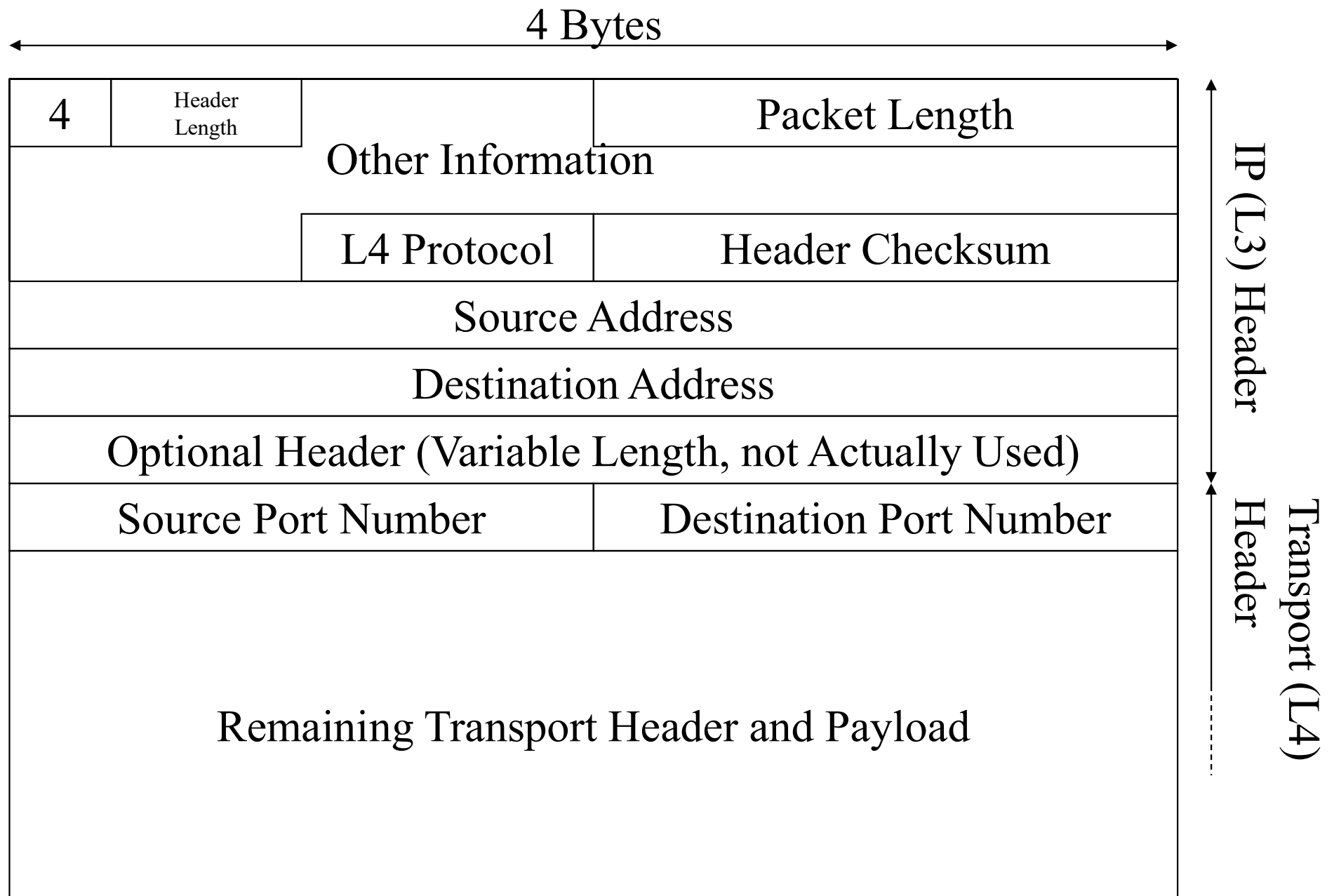
mechanism of NAT



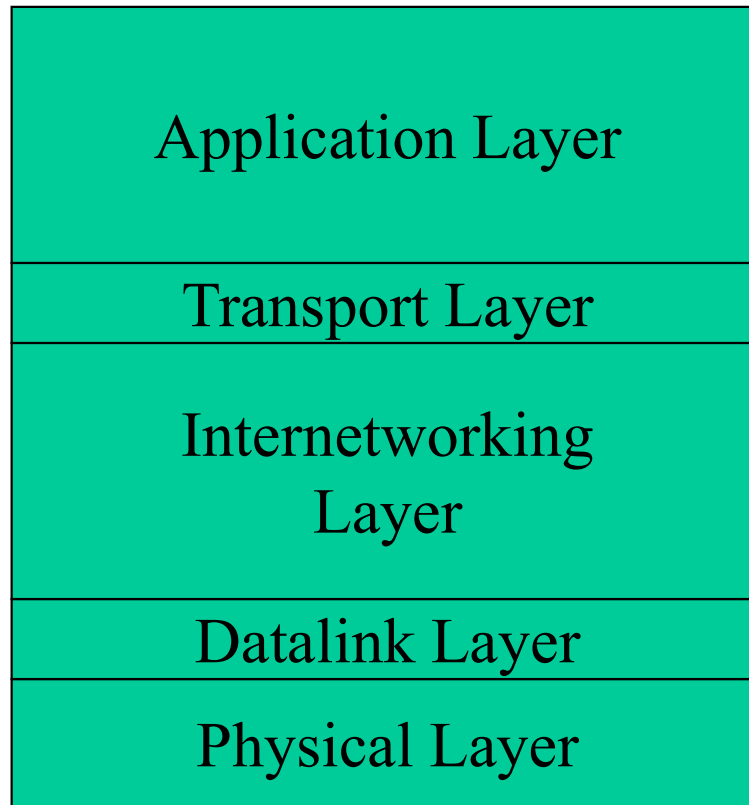
 : NAT Device

NAT and the Internet



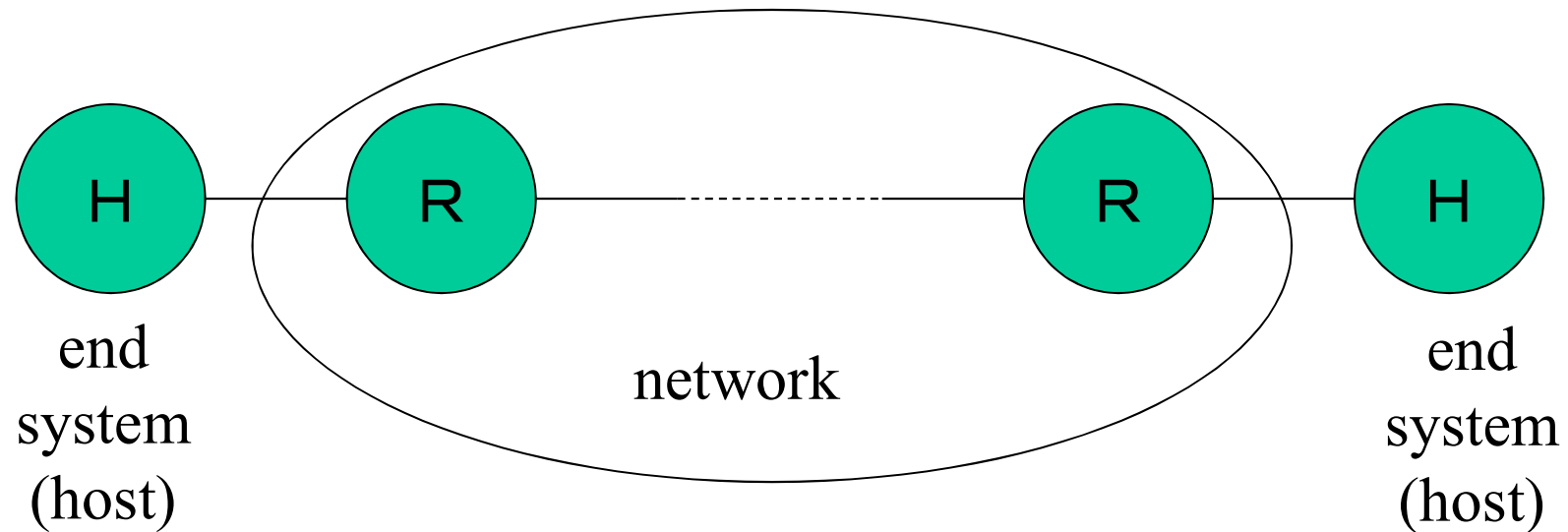
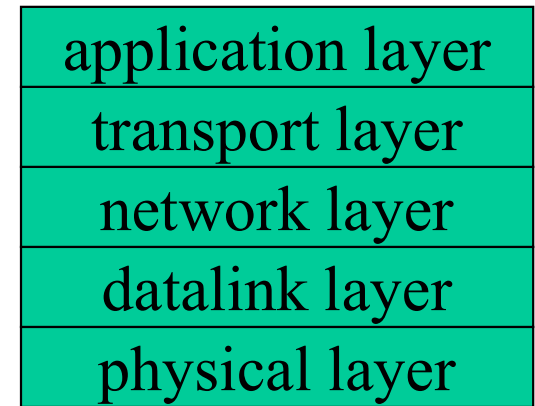
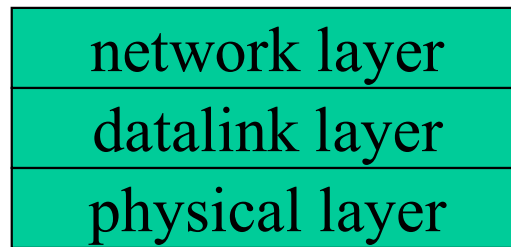
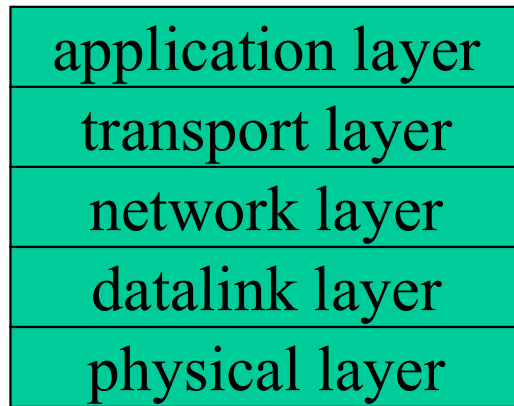


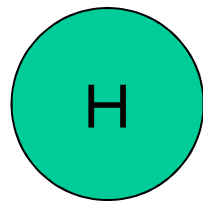
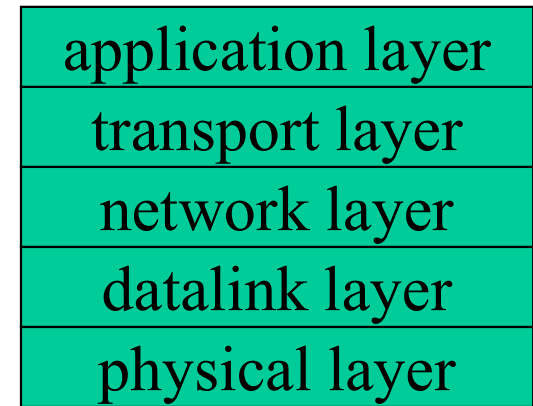
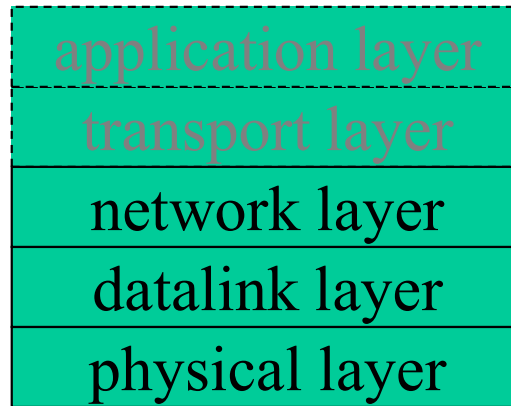
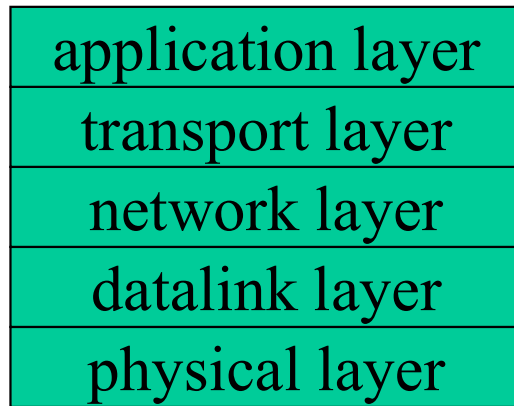
Format of IPv4 Packets



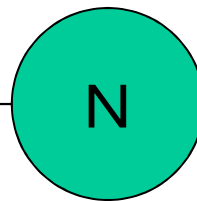
Here is the Essence of  
the Internet

Layering Structure of the Internet

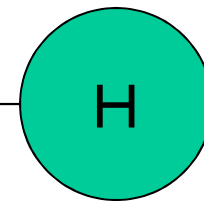




end  
system  
(host)



primitive  
NAT device



end  
system  
(host)

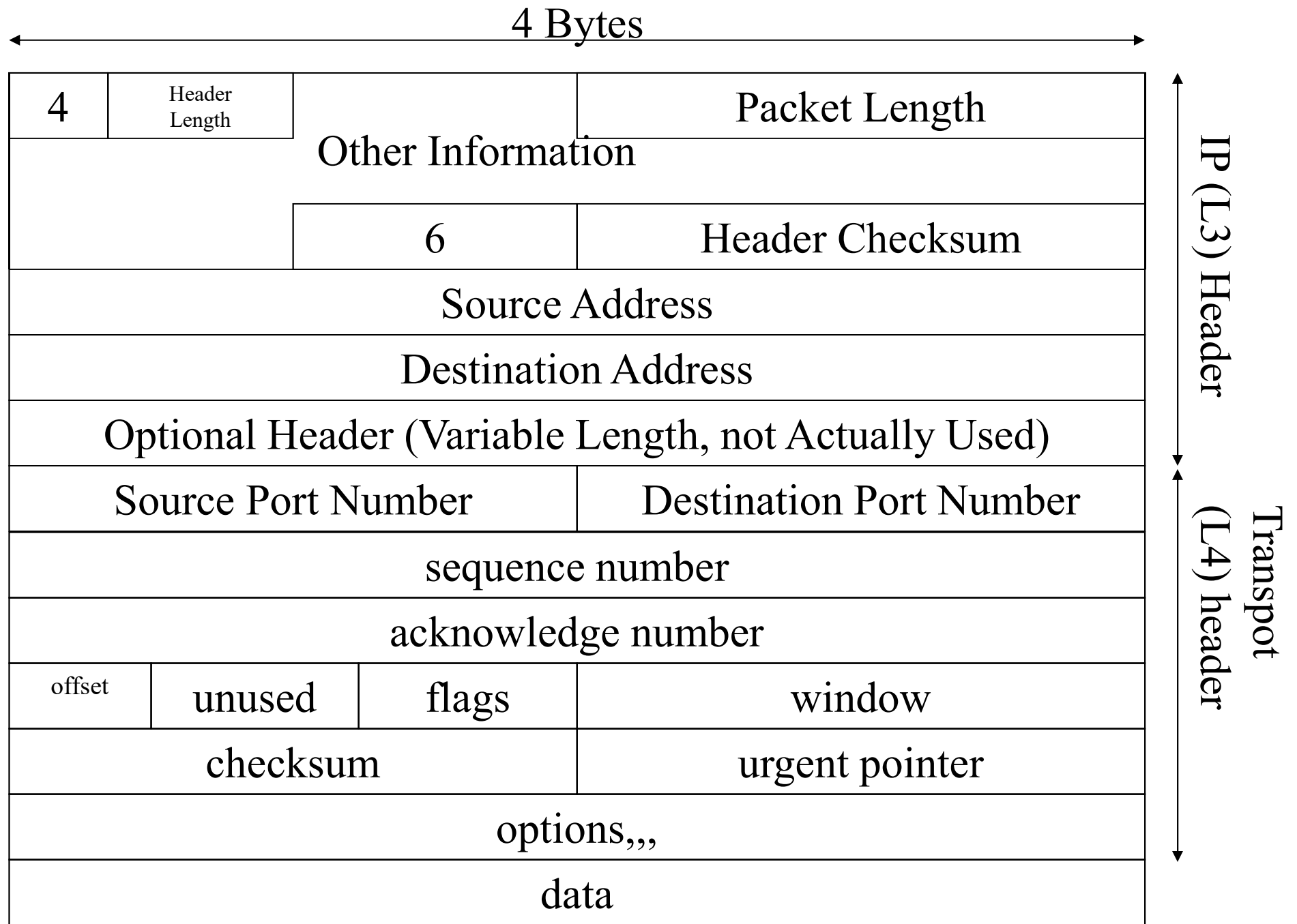
# Primitive NAT

- translate address
  - modify IP header only?
    - source address, destination address, header check sum
    - not so against the E2E principle?
    - addresses in payload cannot be translated
      - ftp, ICMP etc. (protocol dependent translation necessary)
  - transport checksum must also be modified
    - NAT device must know transport protocols
      - what if, new transport protocols are introduced?

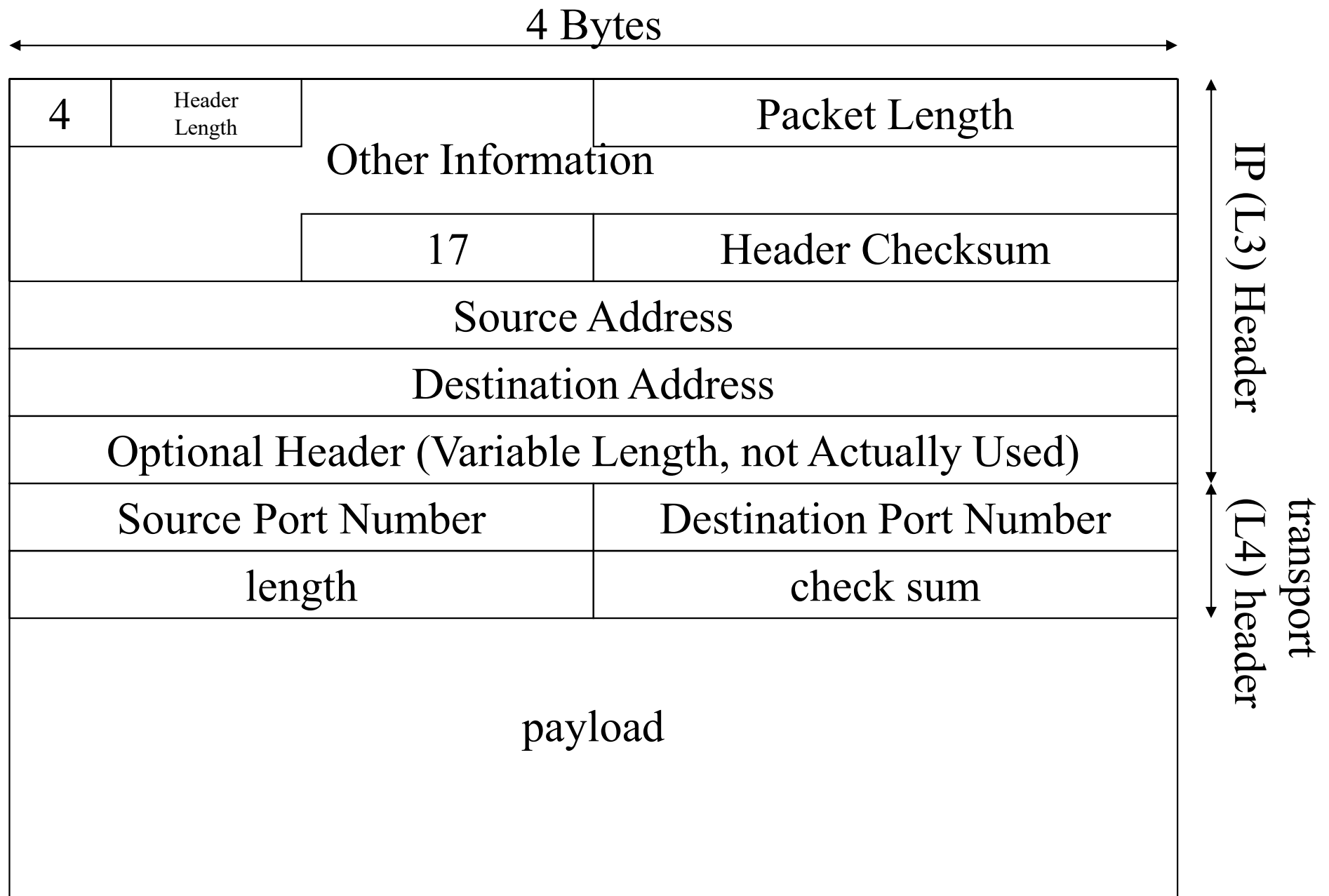
# End to End Argument in Original Paper by Saltzer et. al.

<http://groups.csail.mit.edu/ana/Publications/PubPDFs/End-to-End%20Arguments%20in%20System%20Design.pdf>

- The **function** in question **can completely and correctly be implemented only with the knowledge and help of the application standing at the end points of the communication system. Therefore, providing that questioned function as a feature of the communication system itself is not possible.** (Sometimes an incomplete version of the function provided by the communication system may be useful as a performance enhancement.)



IPv4 TCP packet format

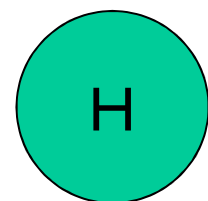
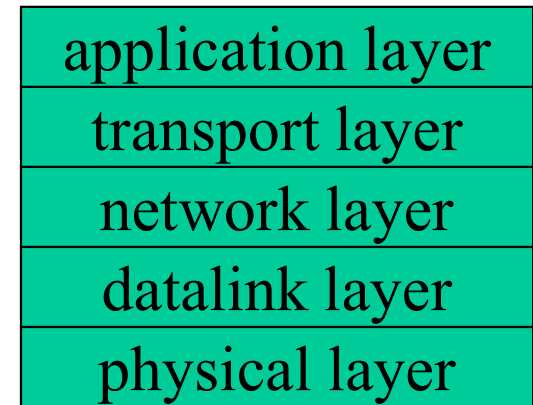
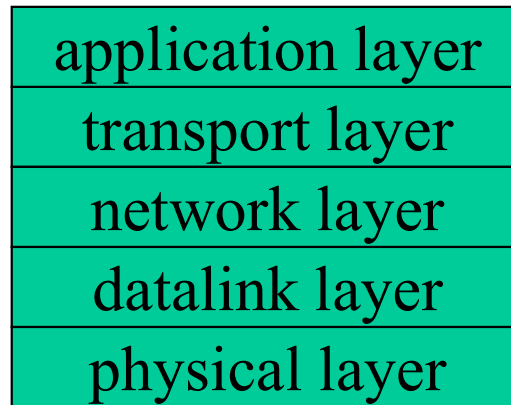
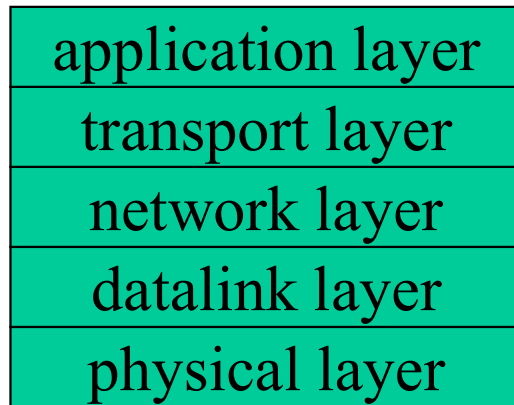


IPv4 UDP packet format

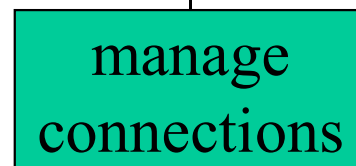
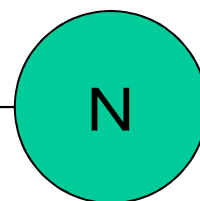


# IP Address Saving by NAT

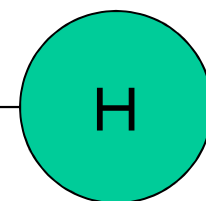
- for hosts in private IP network
  - assign global address only for active hosts
  - cannot know which host is active
    - guess by timeout?
      - inaccurate
    - TCP packet may be monitored to detect start and end of connections
    - not applicable to UDP
      - needs detailed knowledge on application layer



end  
system  
(host)



NAT  
device



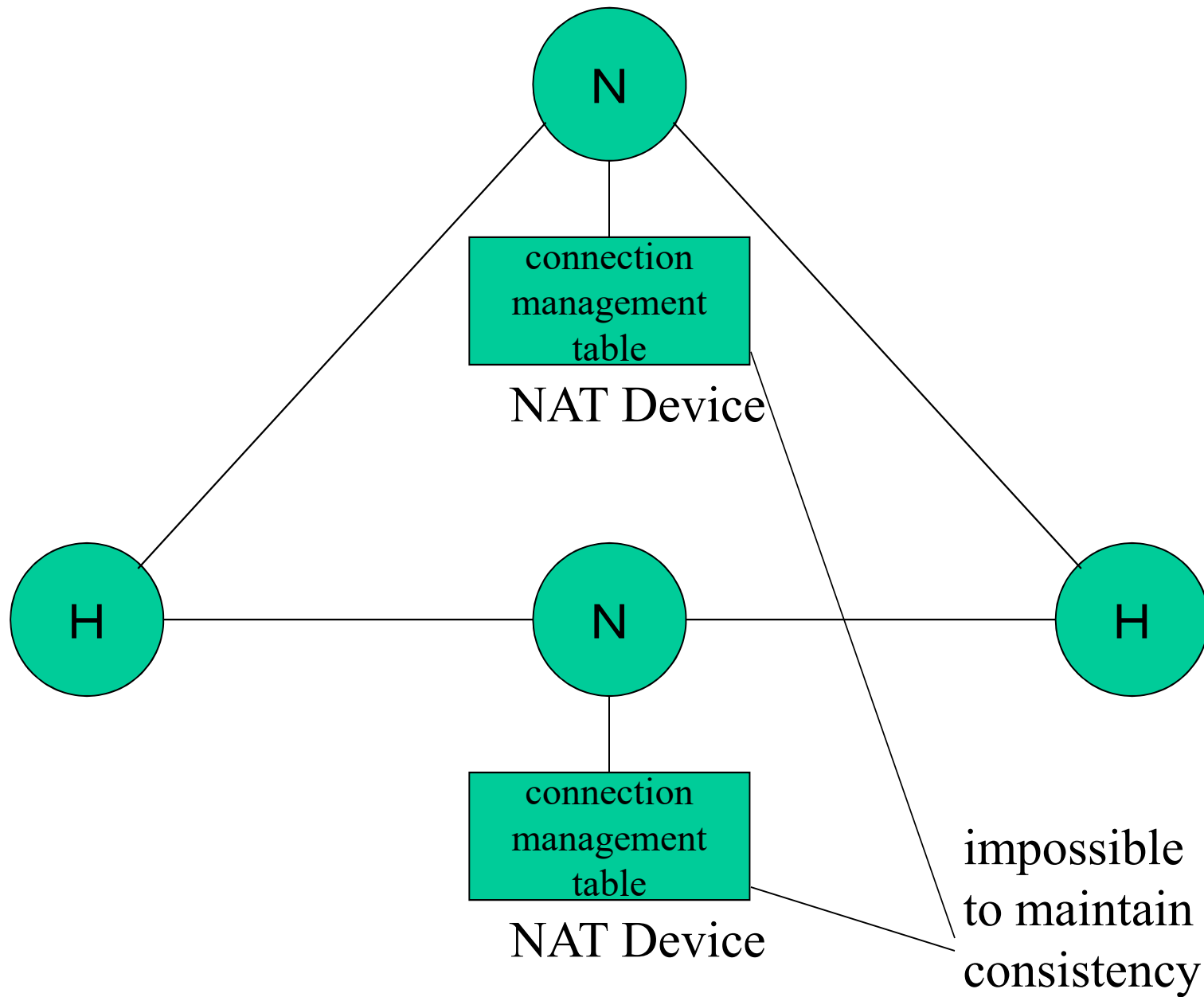
end  
system  
(host)

# Advanced NAT

- knows various application protocols to maintain connection
  - ICMP messages are properly translated
  - ASCII address in ftp may be translated
  - various UDP protocols supported
  - translate port numbers, too
    - a global address may be shared by multiple active connections of multiple hosts

# Problems of NAT

- modification to NAT devices necessary for every new applications
  - new applications is practically impossible to be deployed
- against E2E principle
  - slow (must deeply investigate payload)
  - no redundancy, load concentration
    - parallel NAT devices
      - cannot make connection management consistent



Multiple NAT Devices in Parallel?

# NAT Offers Security(?)

- NAT devices, by default, do not accept incoming connections
  - acting as a (incomplete and incorrect) firewall
  - may be configured to pass some protocol (SMTP, HTTP etc.) only to specific server
  - may be configured to block outgoing connections from most clients
    - clients are forced to use proxy server in private network
      - protocols not supported by proxy server unusable

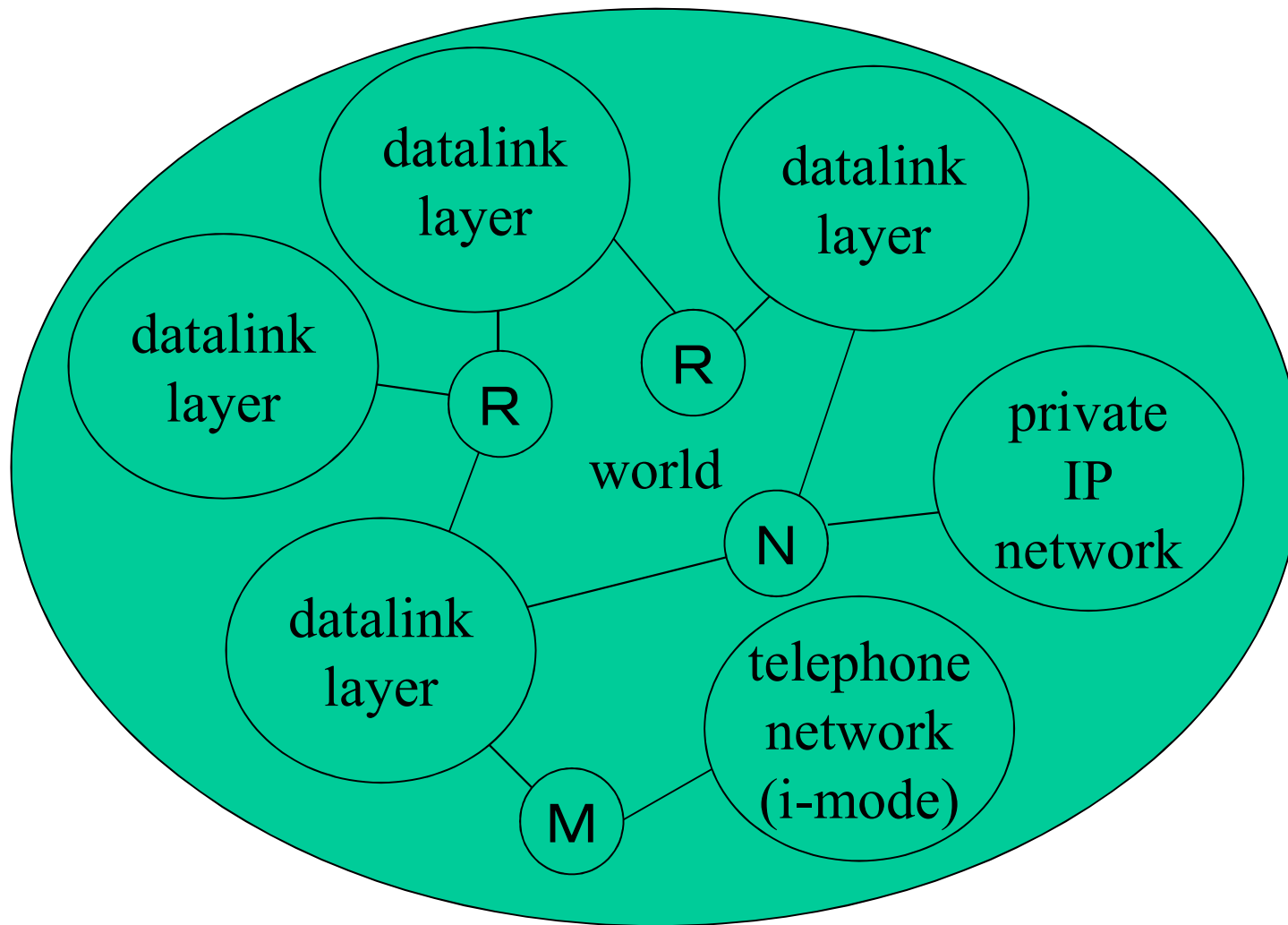
# True Security

- end to end security
  - the principle of the Internet or networking in general
- to make all the ends secure
  - there is no royal road
  - there is no magic

# Implication of Accepting NAT

- NAT device connects IP networks
  - some internet engineers think “even if not the Internet, IP should be good”
- NAT friendly protocols do not have IP addresses other than IP header
  - NAT friendly protocols works over non-IP networks
  - NAT makes IP not necessary



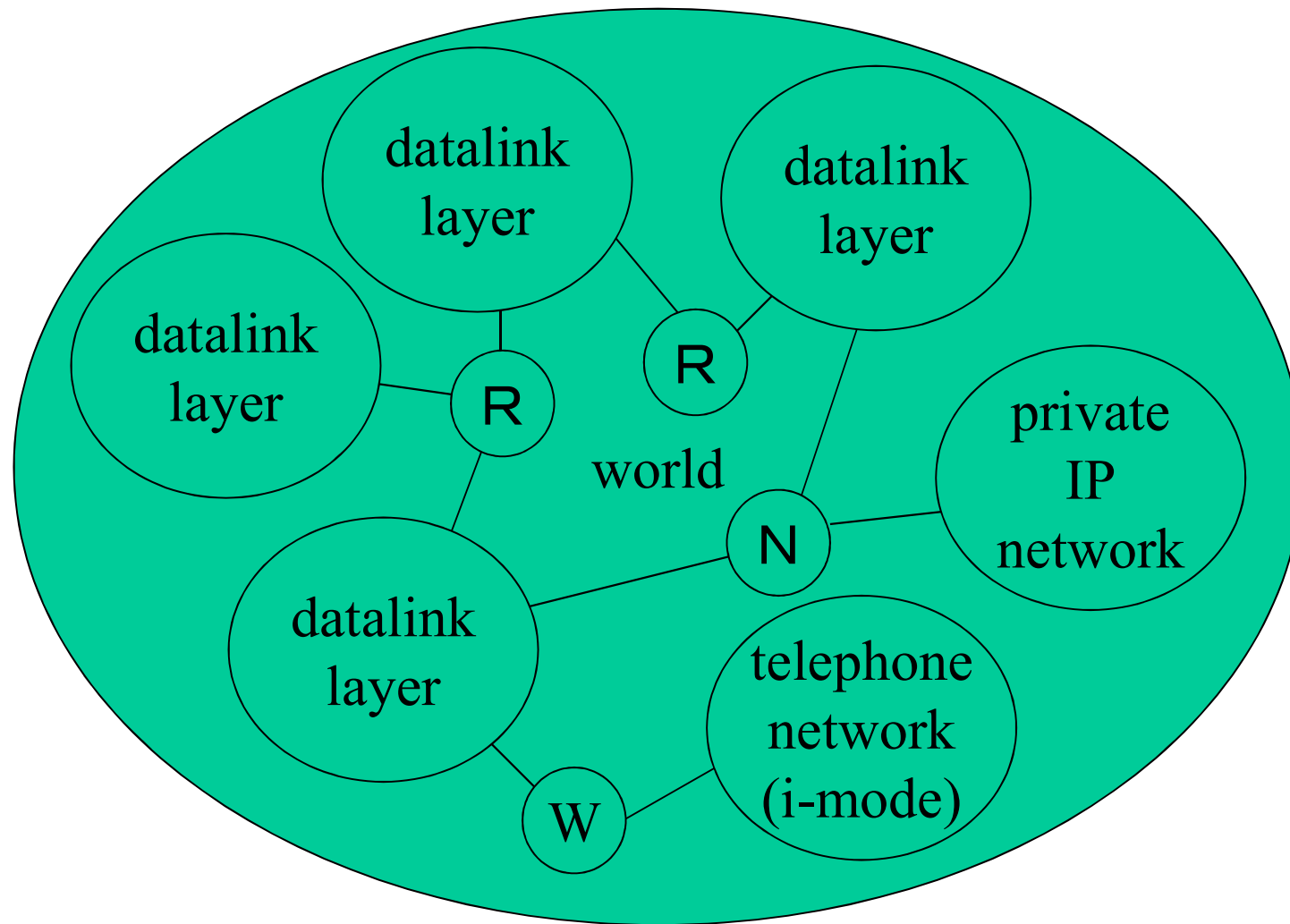


Ⓡ : Router    Ⓝ : NAT Device    Ⓜ : Mail Gateway

e-mail environment today

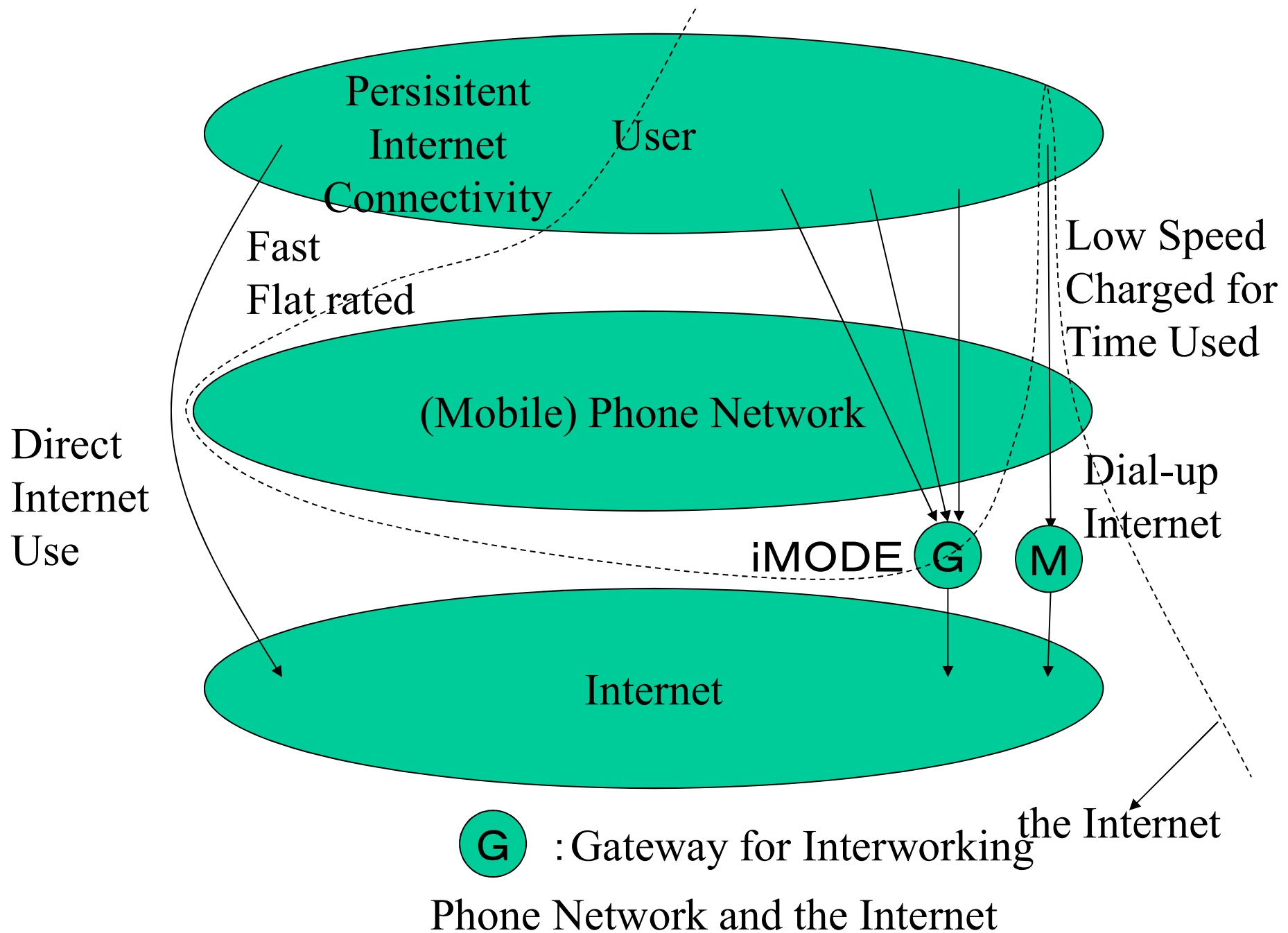
# The Internet and Structure of Networks

- Example of Internet
  - Dial-up Internet
- Example of non Internet
  - i-mode
    - IP, but, relaying at transport layer
  - Legacy NAT
    - IP, but, addresses etc. are modified, which is not visible to terminals
      - Interworking at the transport layer and above



● R : Router    ● N : NAT Device    ● W : Web Gateway

web environment today



# How to Introduce New Applications to the Internet

- design a protocol
- implement it on some hosts (end systems)

# Introduce New Applications with Application Gateways

- design a protocol
- implement it on some hosts (end systems)
- implement it on application gateways
  - often involving network operators
  - application-wise modifications of gateways necessary

# Evolution of Applications with Application Gateways

- major application of the Internet was e-mail
  - using internet means using e-mail
- major application of the Internet is web
  - using internet means using web
- application gateways adopts to SMTP, HTTP and DNS
  - new application must be developed over HTTP?
    - insecure applications may be developed over HTTP

# Collapse of the Internet

- (unautomatic) renumbering is too painful
  - dynamic address allocation by DHCP/PPP
    - no persistent addresses of servers
  - stable private IP addresses by NAT
    - only small # of global addresses need renumbering
- NAT is widely deployed
  - no global connectivity
  - no transparency



# IP over HTTP

# IP over DNS

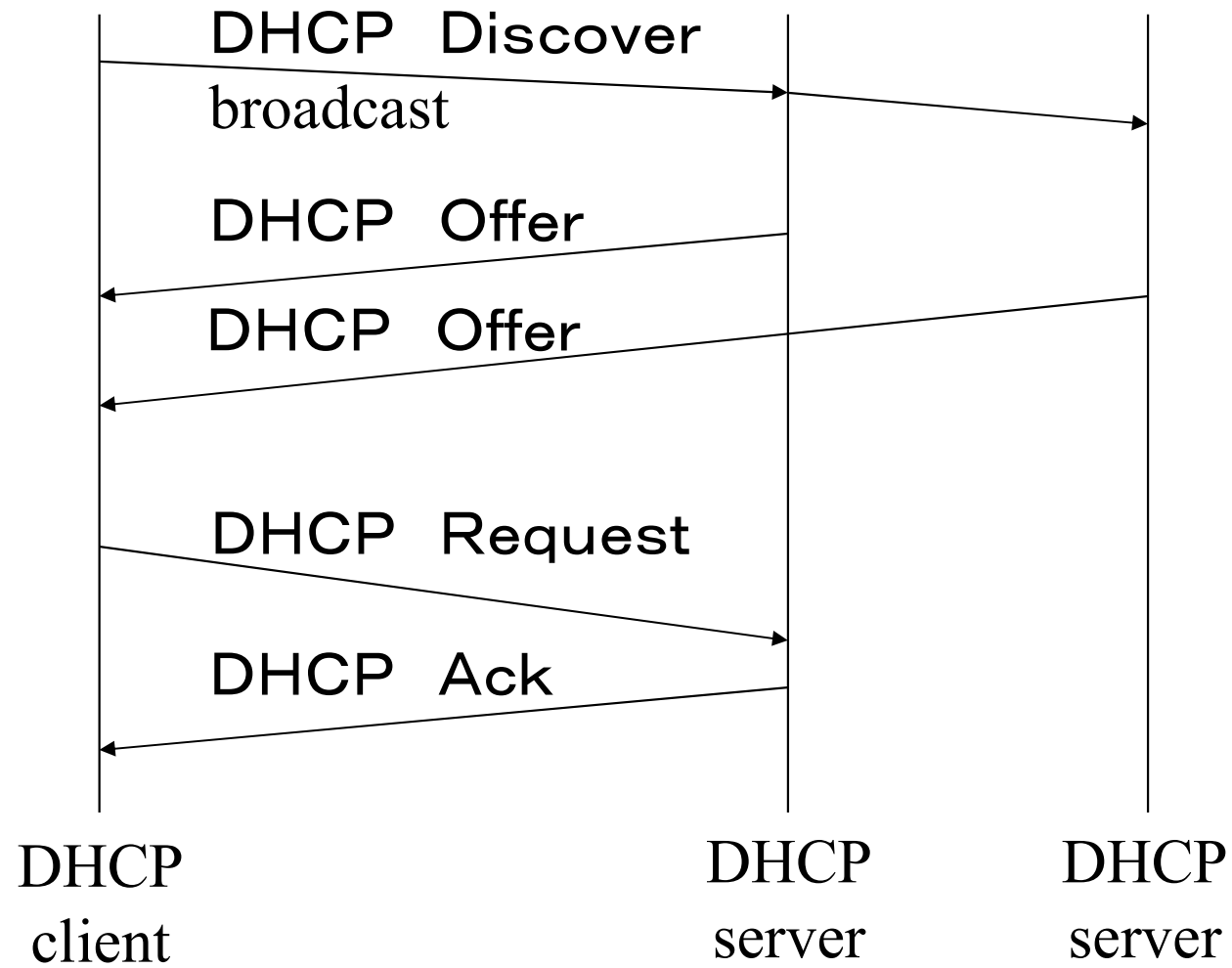
- works within private IP network
- cooperate with a relay in the Internet
  - global IP packets over HTTP/DNS through NAT
- Ethernet over HTTP also available

# DHCP (Dynamic Host Configuration Protocol, rfc2131)

- DHCP server provide configuration information such as IPaddress to hosts (DHCP clients)
- reduce management effort by link broadcast with UDP packets
- management on servers necessary
- useful only on links considered to be secure
  - cryptographic security needs keys configured

# DHCP

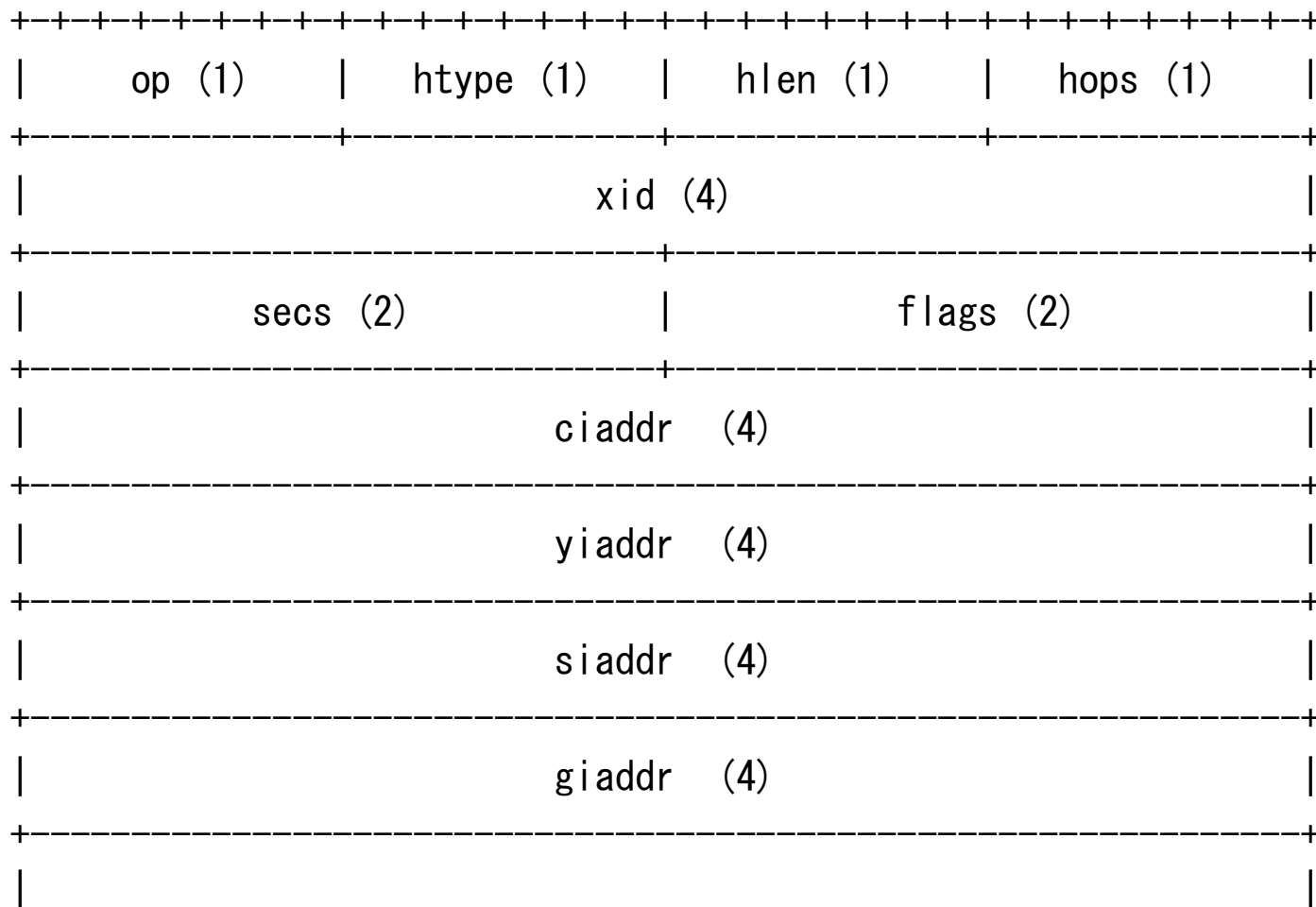
- originally BOOTP
  - to start up diskless clients (thin clients) (with no locally configured state)
- these days, primarily used for dynamic address allocation



# Packet Format of DHCP (1)

- if address is unknown
  - destination address is broadcast address
    - only used with DHCP Discover
  - source address is 0.0.0.0
    - after configuration, use configured address
- over UDP
  - server port number: 67
  - client port number: 68

# Packet Format of DHCP (2)



# Fields of DHCP (1)

- op
  - 1: request, 2: reply
- htype
  - hardware type (1=10M Ethernet)
- hlen
  - MAC (hardware) address length (6 for Ethernet)
- hops
  - # of relayed hops (initially 0)

# Fields of DHCP (2)

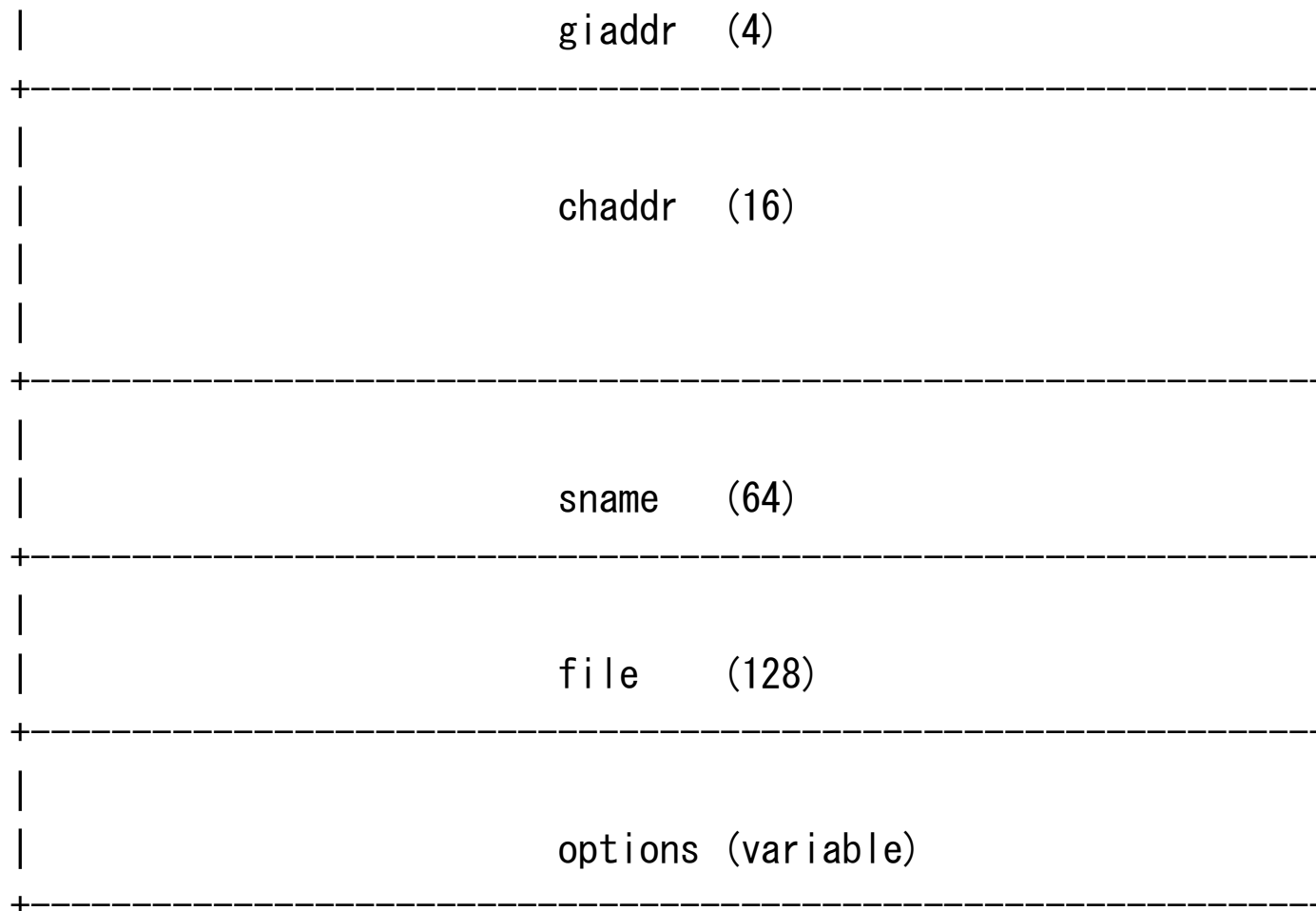
- xid
  - transaction ID
- secs
  - seconds after the first request
- flags
  - flags
- ciaddr
  - IP address of client



# Fields of DHCP (3)

- yiaddr
  - address allocated to client (Your address)
- siaddr
  - IP address of server
- giaddr
  - IP address of relay

# Packet Format of DHCP (3)



# DHCPのフィールド(4)

- chaddress
  - client MAC (hardware) address
- sname
  - hostname of server (ASCII string)
- file
  - boot file name (ASCII string)
- options
  - option sequence in TLV (type (tag, code), length, value) format

# DHCP and DNS (1)

- hosts in the Internet has DNS entries of
  - hostname→IP address (forward lookup)
  - IP address→hostname (reverse lookup)
- how can IP address from DHCP registered?
  - by DDNS (Dynamic DNS) (rfc2136)
  - security by shared secret key
    - reverse lookup by DHCP server
    - forward lookup by hosts
      - key must be configured
        - » hostname must be configured, anyway

## DHCP and DNS (2)

- how should we configure address of DNS server?
  - by ND (of IPv6)
    - ND have too much functionality
    - ND is (formally) optional
  - by DHCP (DHCPv6)
    - should be proper, if DHCP is available
  - by well known anycast address
    - may be used anywhere
    - may be overridden by DHCP etc.

# Wrap-up

- **legacy** NAT has been
  - against E2E principle
  - obstacle to new applicaitons
  - such problems of NAT can be removed
    - slides follow
- DHCP
  - useful
  - some configuration is always necessary (at least on server side)

# End to End NAT

Masataka Ohta

# IP Addresses are Running Out!

- IETF should be able to address the issue
  - so far, not so much
    - address saving by NAT
      - destroy E2E Internet
      - many protocols do not work over NAT
    - IP address space extension by IPv6
      - political compromise without considering real world operations
      - too much useless/harmful functions such as optional header, PMTUD, SLAAC, link local address, etc.



# End to End Argument in Original Paper by Saltzer et. al.

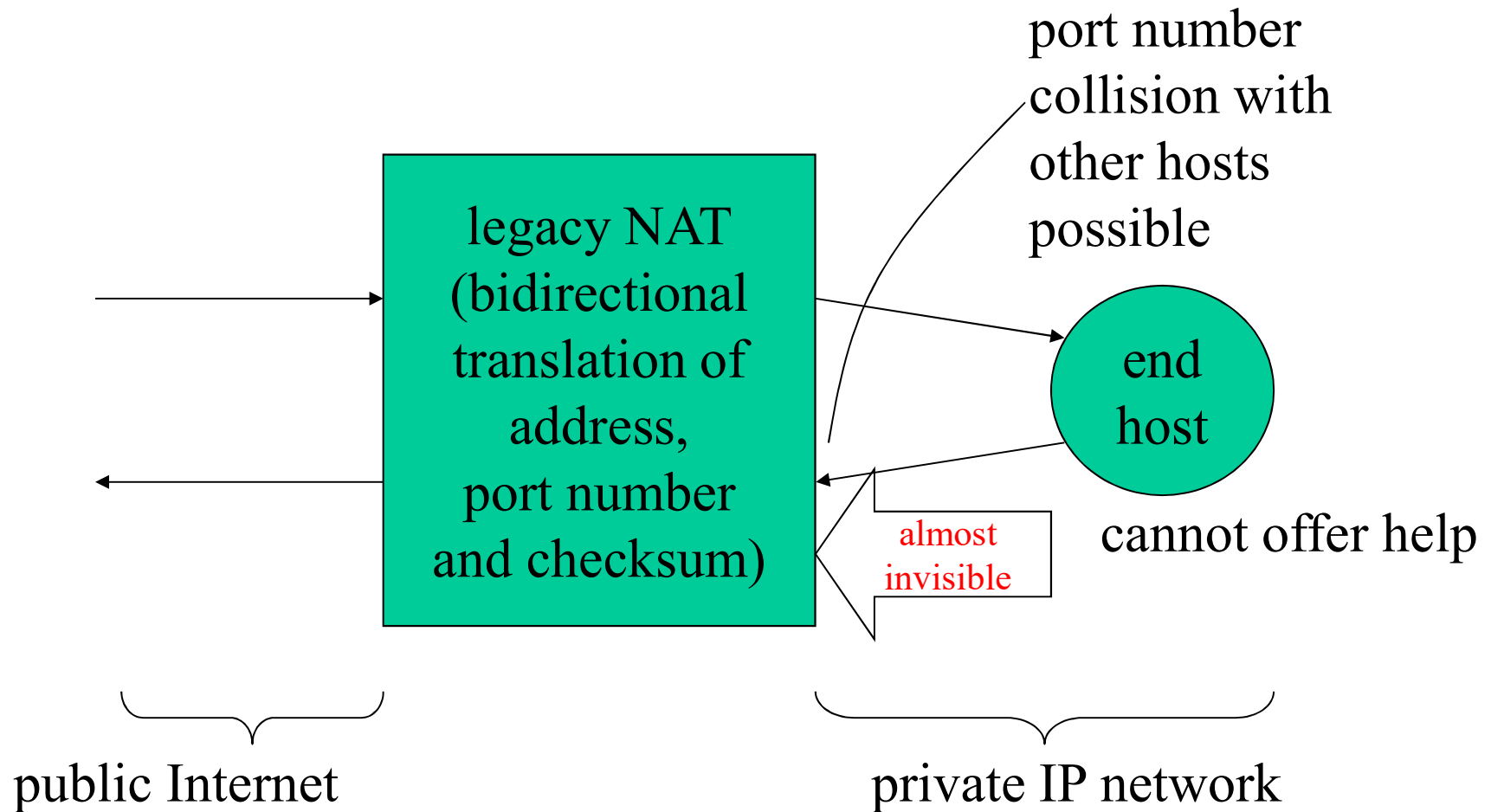
<http://groups.csail.mit.edu/ana/Publications/PubPDFs/End-to-End%20Arguments%20in%20System%20Design.pdf>

- The **function** in question **can completely and correctly be implemented only with the knowledge and help of the application standing at the end points of the communication system. Therefore, providing that questioned function as a feature of the communication system itself is not possible.** (Sometimes an incomplete version of the function provided by the communication system may be useful as a performance enhancement.)

# Background

- according to the end to end argument
  - NAT can completely and correctly be implemented **only with** the knowledge and help of the application standing at the end points of the communication system
    - legacy NAT without relying on the knowledge and help of end hosts is incomplete and incorrect without E2E transparency
- a natural question is “how is converse?”
  - With the knowledge and help of the application standing at the end points of the communication system
    - Can NAT be implemented completely and correctly?

# Legacy NAT



# End to End NAT

- actively inform end existence of NAT-
- inform each end system in private IP network
  - public address shared by the end system
  - range of port # assigned to the end system
  - how to communicate with NAT GW (address, port)

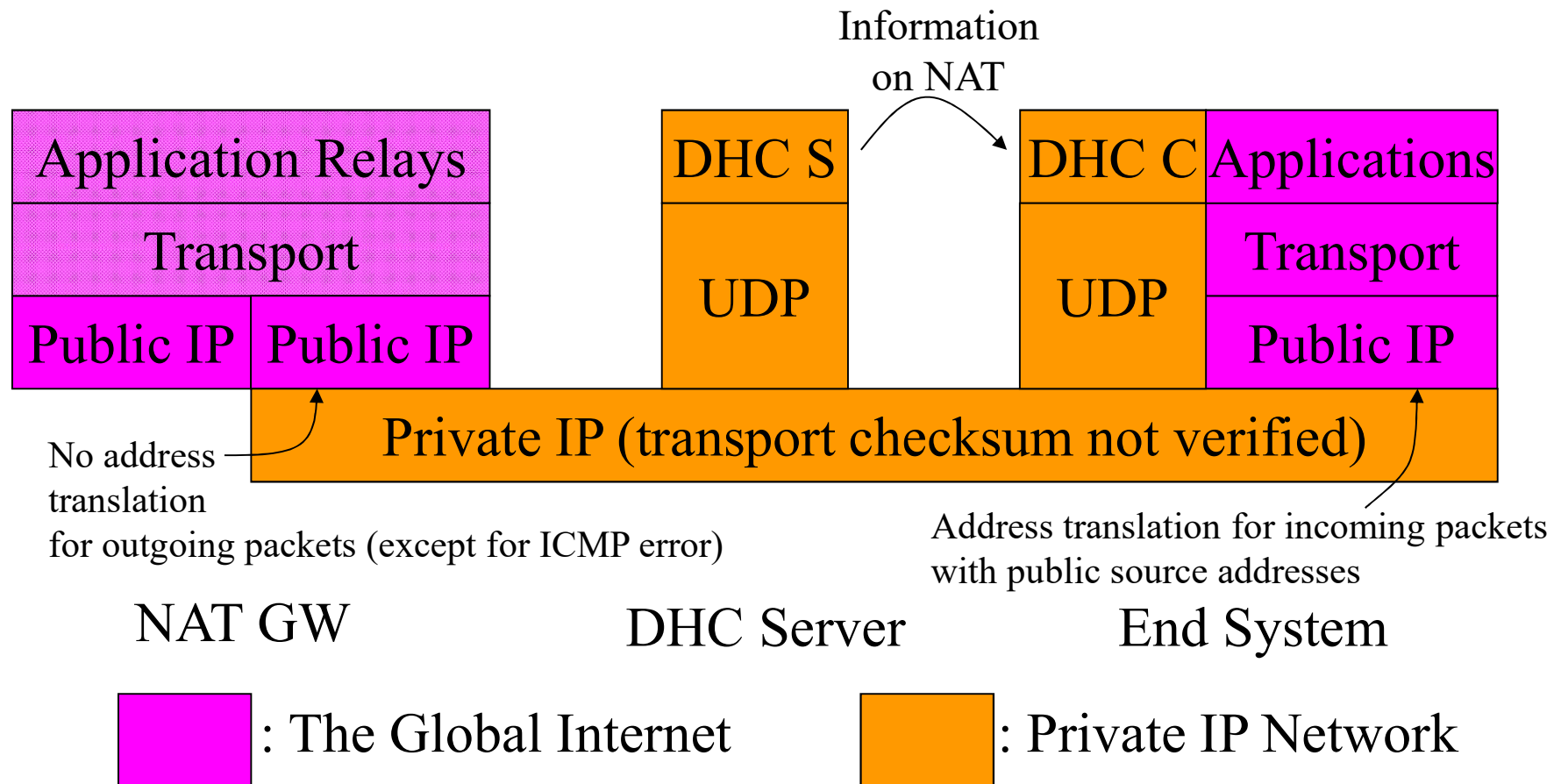
through DHCP or PPP

- each end system
  - help NAT GW with its knowledge to make NAT operation complete and correct

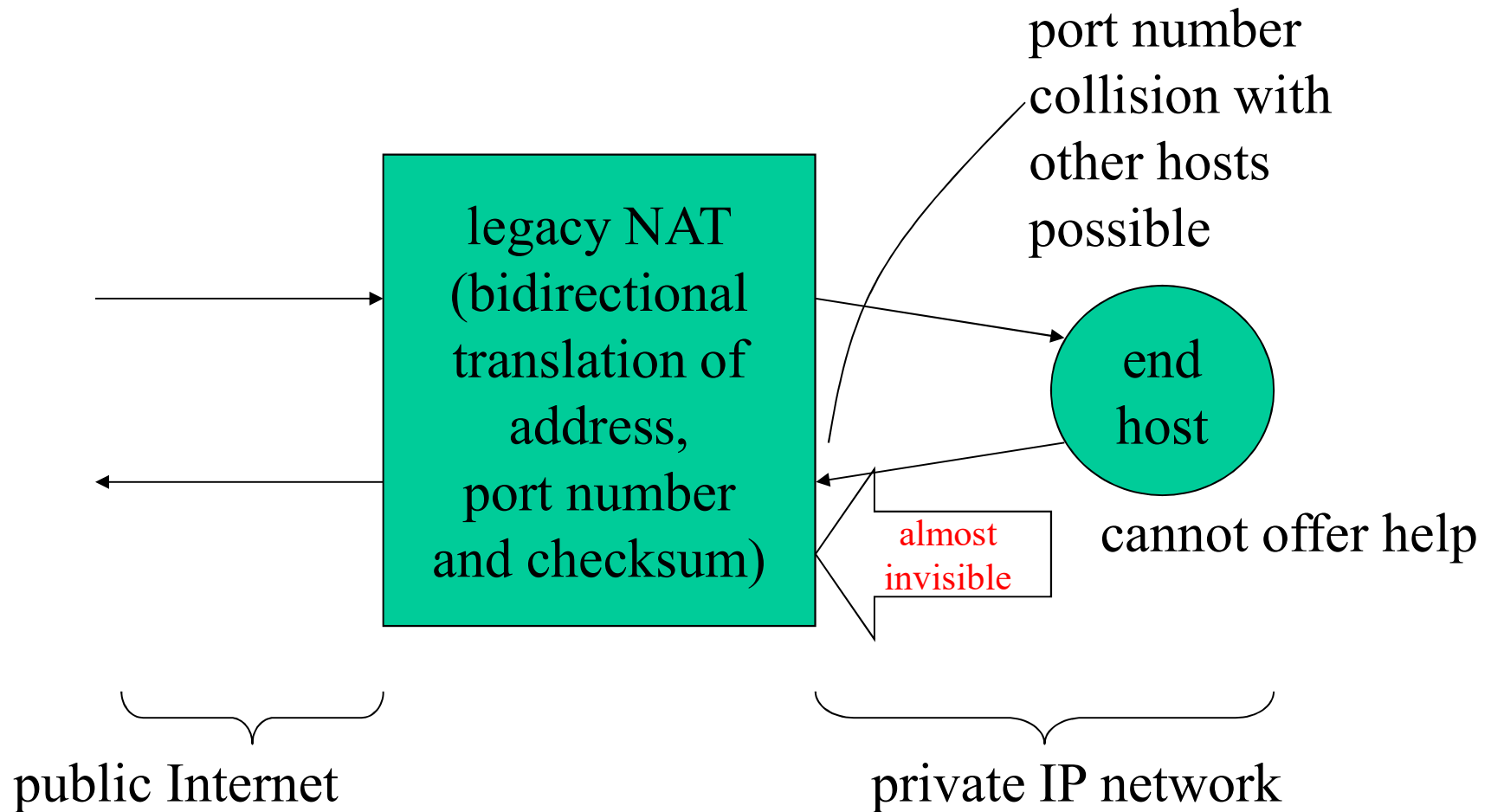
# Operations of End to End NAT

- NAT gateway
  - translate destination address of incoming packets by destination port number
    - no translation of port # or transport check sum
- end system behind NAT
  - translate of destination address **back to public address**
    - transport checksum is, now, correct
  - restrict source port # of outgoing packets to those assigned to the end system

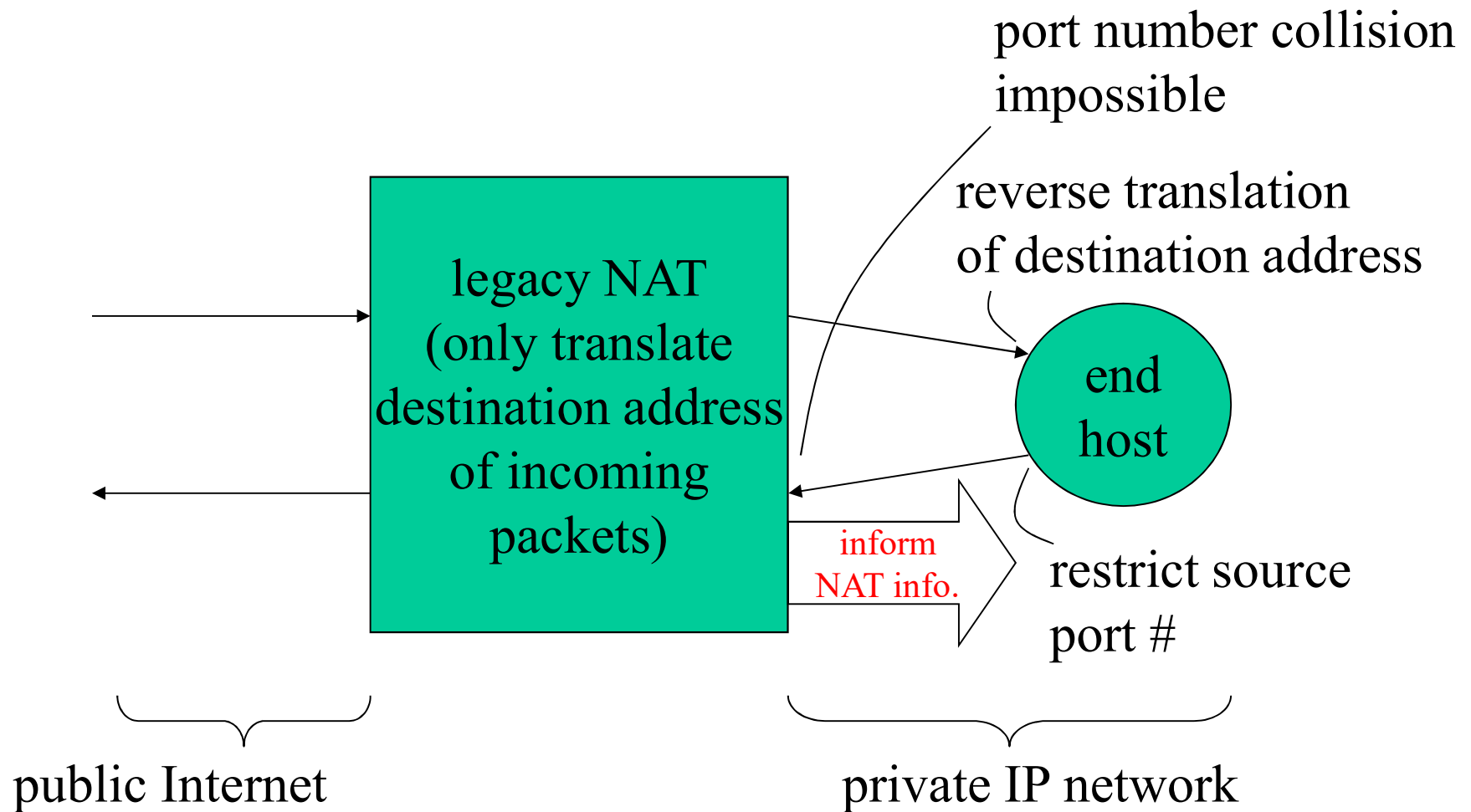
# Layering Structure of End to End NAT



# Legacy NAT

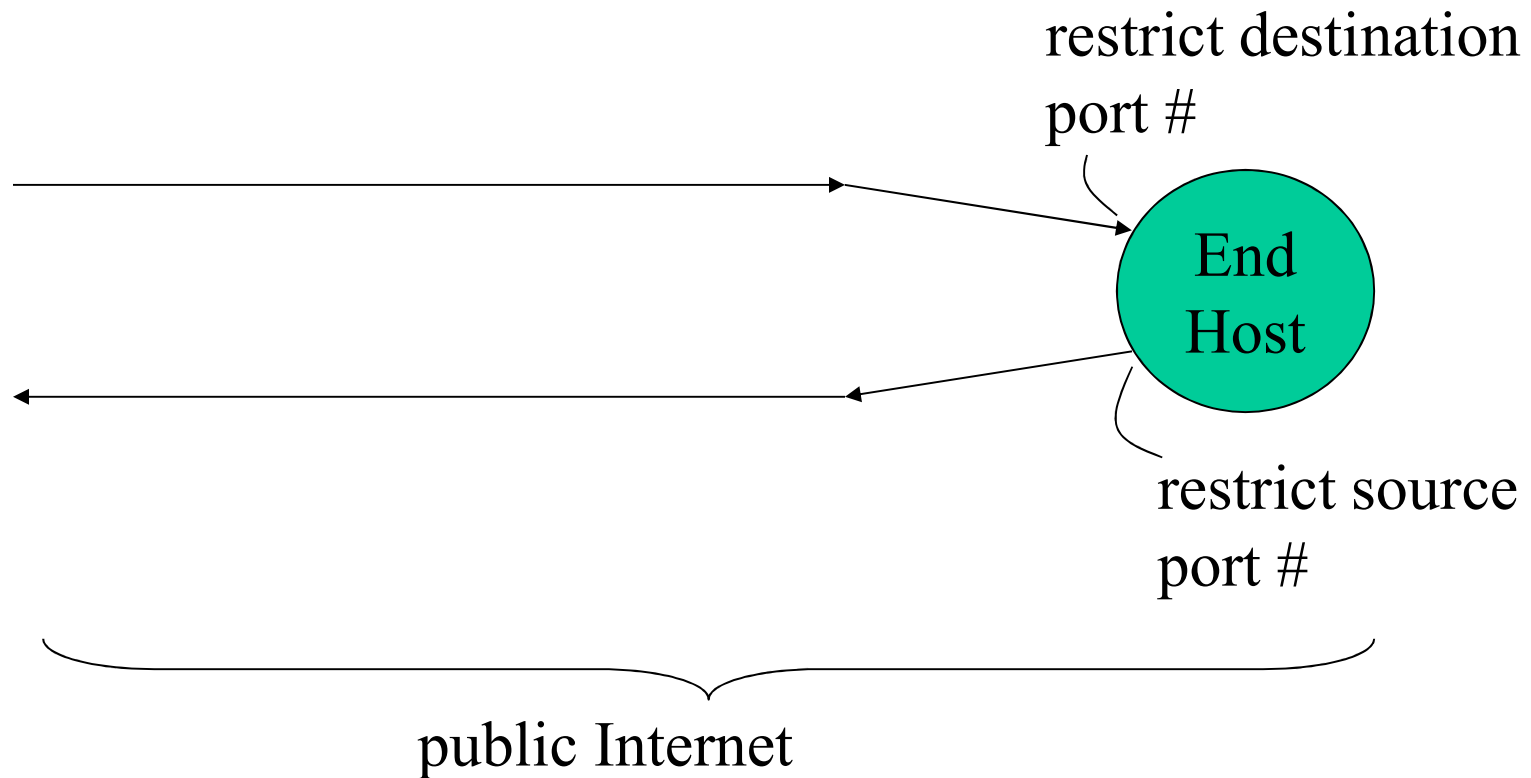


# End to End NAT





# Equivalent to Direct Internet Connection



# Properties of End to End NAT

- full E2E transparency
  - as long as protocols have port # or equivalent
  - PORT command of ftp works, naturally
- may be nested with hierarchy
- compatible with
  - legacy NAT, ICMP, DNS reverse look up (incl. port #), multicast, mobility (with extension for port wise mobility), IPsec,,,

# Static NAT and Dynamic NAT

- Static NAT
  - assign fixed port # range to each terminal
    - with enough (hundreds?) # or port, should be sufficient
- Dynamic NAT
  - each terminal request NAT GW port #
    - port assignment state in NAT GWs may be updated actively involving terminals
      - no guess by timeout necessary, multiple GWs may be synchronized

# Difference between Static E2ENAT and Port Forwarding

- port forwarding of legacy NAT
  - some port is statically assigned to some host
    - **transport layer** relaying by legacy NAT
    - can operate as servers?
      - not actually transparent for some applications
- Static E2ENAT
  - fully E2E transparent

# End to End NAT and Port Numbers

- non default port number may be specified by URLs or DNS SRC RRs
- E2ENAT works almost in IP layer
  - except for source port # **outside of IP header**
    - with pure transport protocols, 16 bit next to IP header is source port # and 16 bit destination port # follows
  - **format of ICMP for packet errors**
    - **(64B ICMP header)+(IP header and 64 bits after the header of packets casing the error)**
      - 64 bits should contain something like port#

# SRV RR (rfc2782)

- similar to MX for applications in general
  - Name TTL Class MX  
Priority Target
  - \_Service.\_Proto.Name TTL Class SRV  
Priority **Weight Port** Target
    - Weight and Port number may be specified
  - \_http.\_tcp.www.example.com SRV  
0 1 9 server.example.com.

# End to End NAT and ICMP

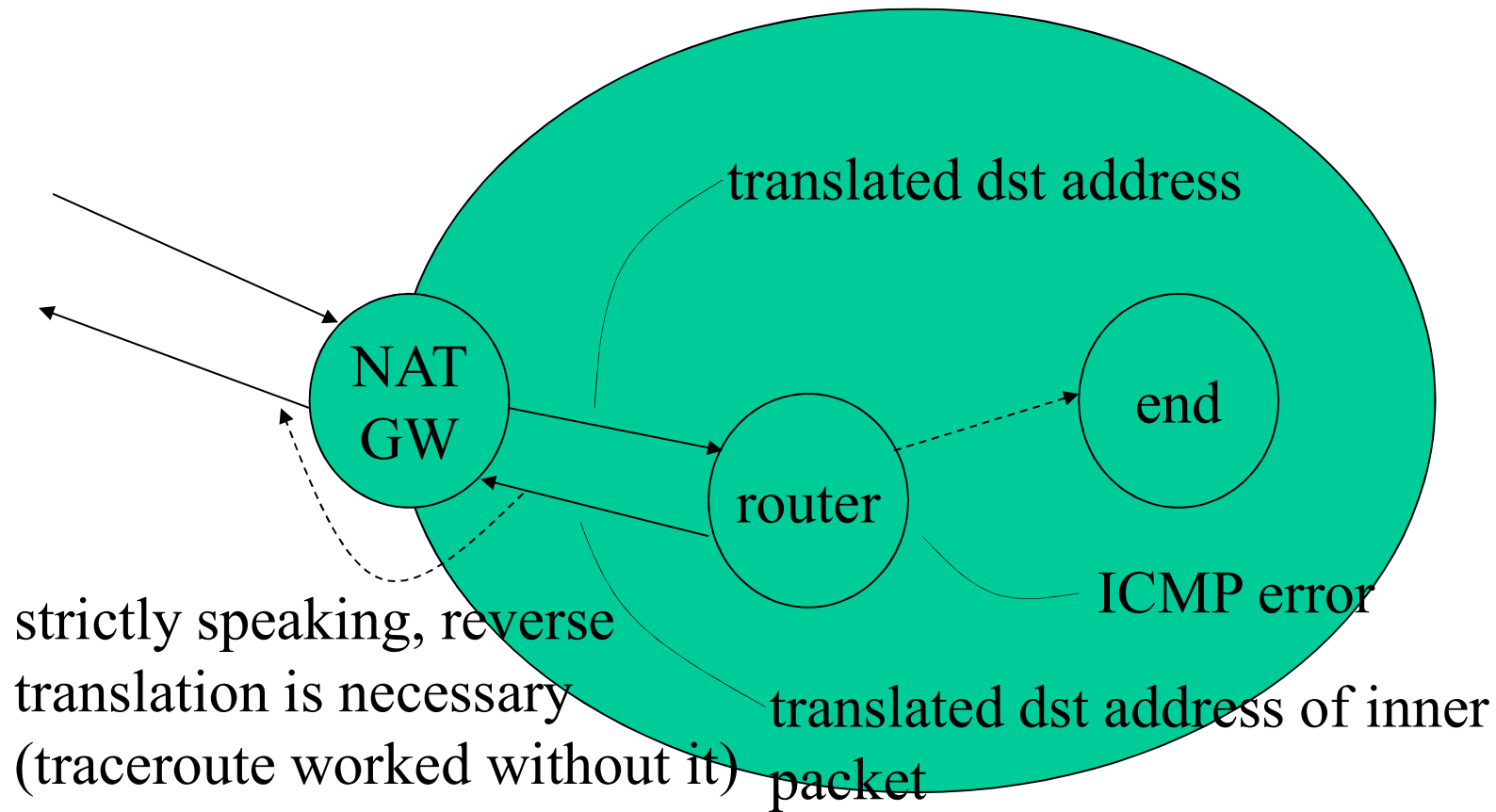
- basically interoperate as is
  - strictly speaking, reverse address translation of inner IP header necessary
    - for ICMP error generated in private network to public Internet
- port # of ICMP packets?
  - ICMP error
    - inner src as dst, dst as src
  - ICMP Echo etc.
    - regard ID as src and seqno as dst

# End to End NAT and ICMP Error

- ICMP error is translated based on source port # of inner IP packet
- ICMP Host Unreachable
  - affect other hosts sharing public address
  - as it is a soft error, not a serious problem
    - TCP connection won't reset
    - shouldn't be **intelligently** translated to port unreachable, because it is a hard error



# Address Translation within ICMP



# End to End NAT and ping/traceroute

- want to make ping/traceroute work
  - use port #s assigned to a host as ID (source port#)
  - extend to be able to specify seqno (dst port #)
    - ping ... host[:dstport[,incr[,count]]]
    - traceroute ... [-p port[.incr]] ... host ...
    - assume each host is assigned port, port+incr, port+incr\*2,,,

# End to End NAT and ICMP Echo

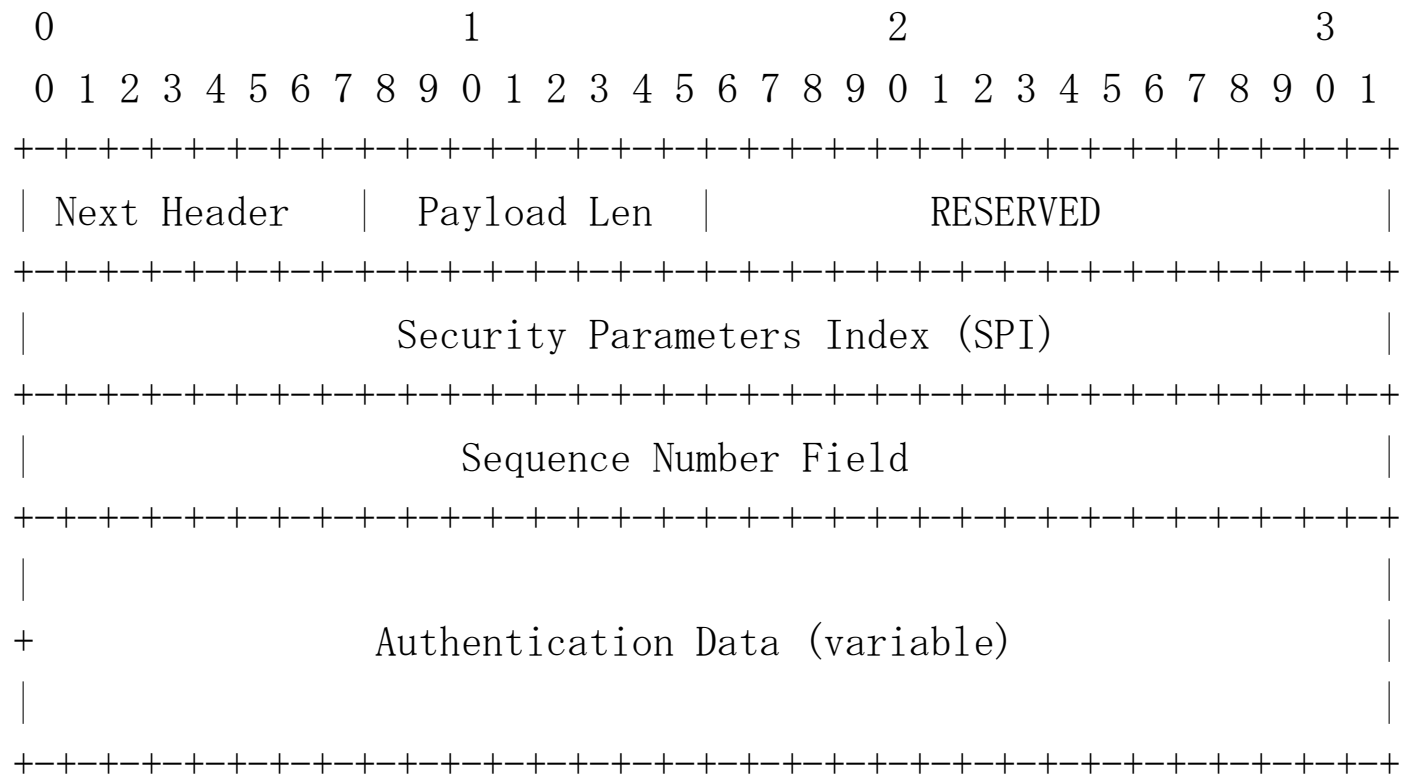
- ICMP echo request regards:
  - ID as src port# and seq# as dst port #
    - restrict ID and address translation by seq#
- ICMP echo reply, conversely, regards:
  - ID as dst port# and seq# as src port #
    - ID and seq# is copied from those of request

# End to End NAT and IPsec

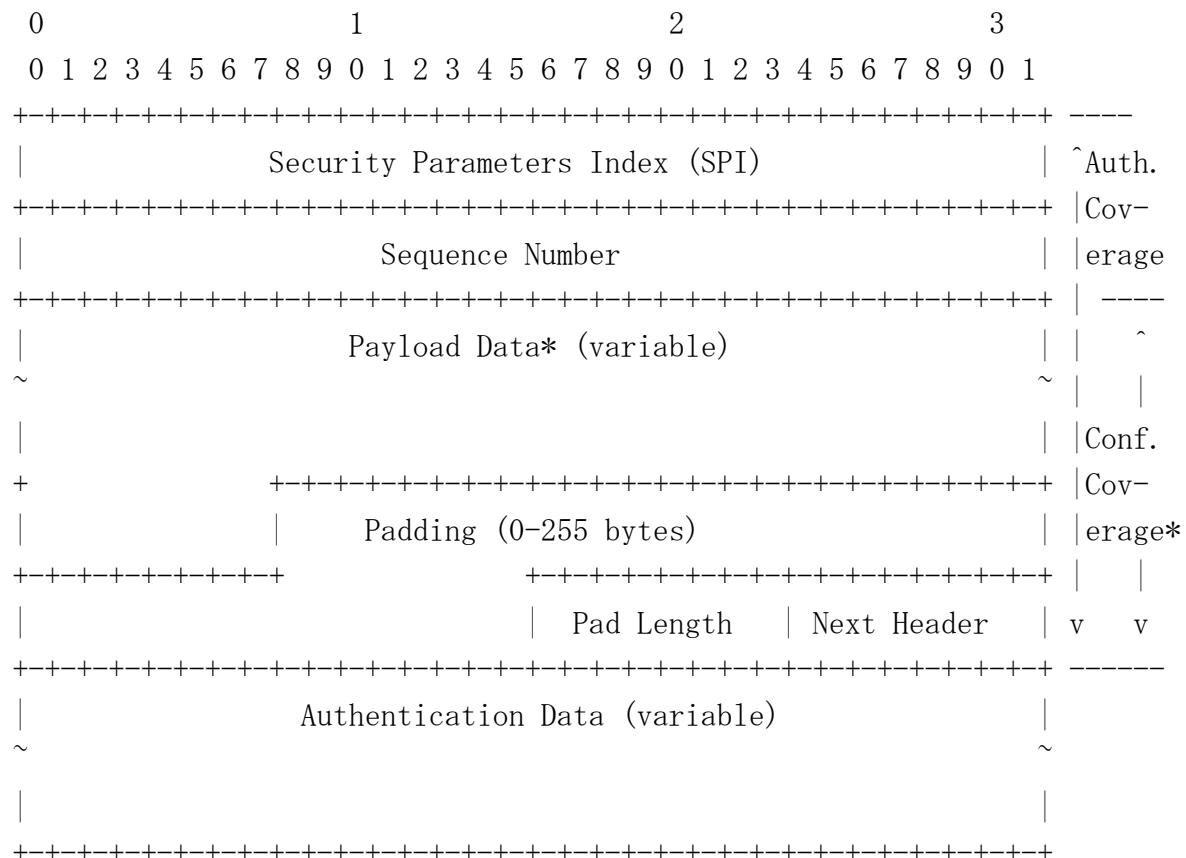
- both AH and ESP has 32 bit SPI within 8B after IP header
- if first 16 bit of SPI is specified by src and last 16 bit of SPI is specified by dst
  - may be used as src and dst port numbers

# AH (Authentication Header)

## (rfc2402)



# ESP (Encapsulating Security Payload) (rfc2406)



# End to End NAT and Application Relays

- with DNS, SMTP, HTTP, request may be received by default port and multiplexed by information (domainname etc.) in request
  - port# may not be specified by http URL
  - port# of DNS and SMTP is implied by NS and MX and cannot be changed
    - multiplexing by application relays necessary, if servers must be placed in private IP network

# End to End NAT and Source Address Used by End Host

- end host host knows its public and private IP addresses
  - packet destined to private network use private address as source address
  - other packets use public address as source address
  - packets destined to its own public address should be output unless destination port # is not ones assigned to it



# End to End NAT and Legacy Terminals

- legacy terminals in private IP network behind E2ENAT GW
  - may use DHCP
    - though cannot recognize NAT information
- packets from legacy terminals
  - may be treated by E2ENAT GW as legacy NAT
    - distinguished by source address is private or public

# Nesting End to End NAT

- E2ENAT GWs may be nested
- subscriber statically assigned many (hundreds of) port #s from ISP
  - may use some statically by servers
  - others may be assigned to inner E2ENAT GW
    - to be dynamically shared by hosts in inner private IP network

# End to End NAT and DNS Revers Lookup

- shared address may be looked up as usual  
www.example.com A 208.77.188.166  
166.188.77.208.in-addr.arpa PTR www.example.com
- port# wise reverse lookup as follows  
p1.example.org CNAME www.example.com  
1.0. 166.188.77.208.in-addr.arpa PTR p1.example.org  
p2.example.org CNAME www.example.com  
2.0. 166.188.77.208.in-addr.arpa PTR p2.example.org
  - though PTR referring CNAME is superficial violation of rfc1034, it will not cause problems

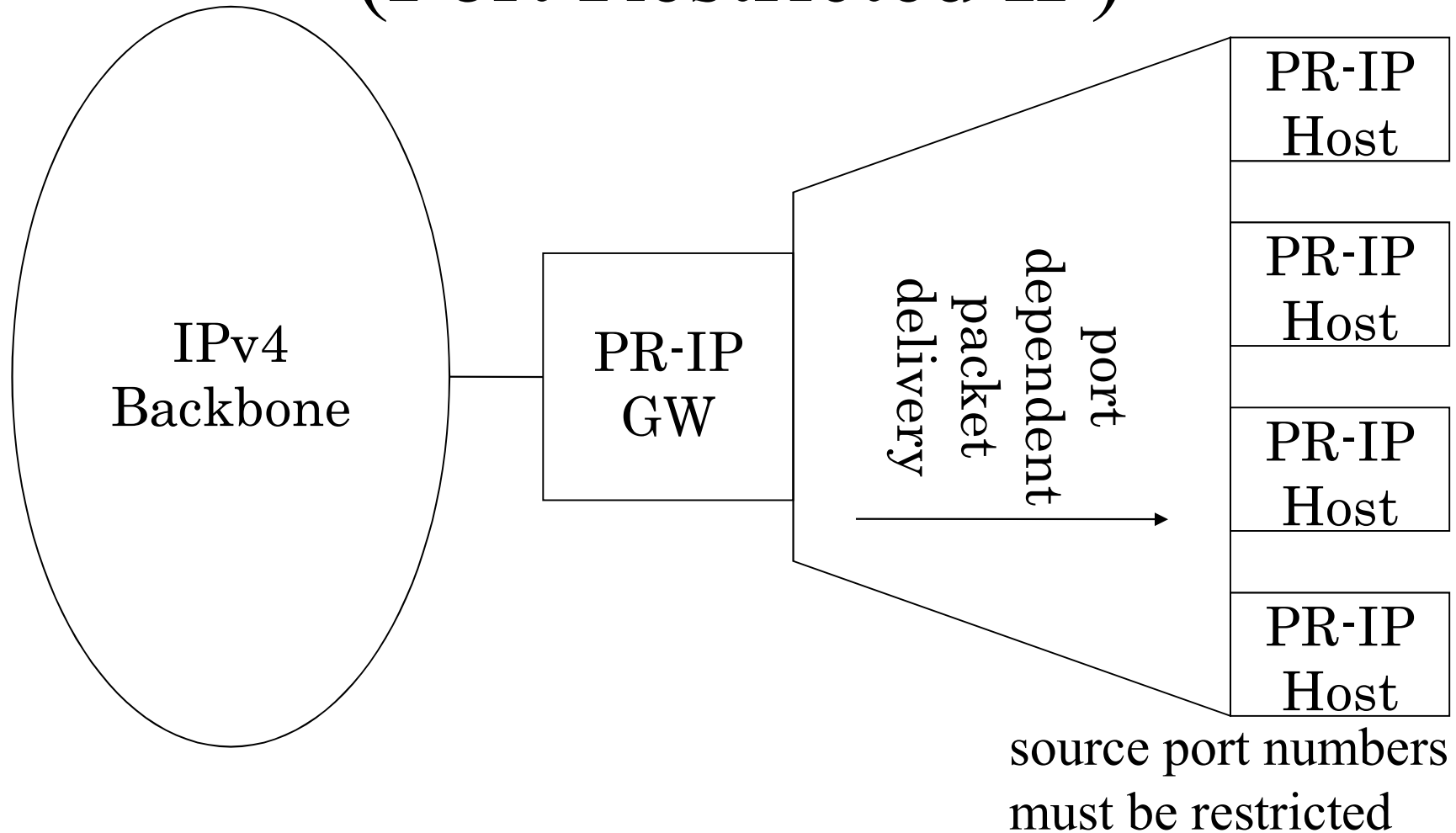
# End to End NAT and Multicast

- multicast address should be common and globally unique in public internet and private IP network
- as some multicast routing protocols use source address for routing, multicast packet should be sourced by NATGW
  - hosts in private IP network should use unicast IP over IP to NATGW
    - PIM (protocol independent multicast) already has mechanism to do so

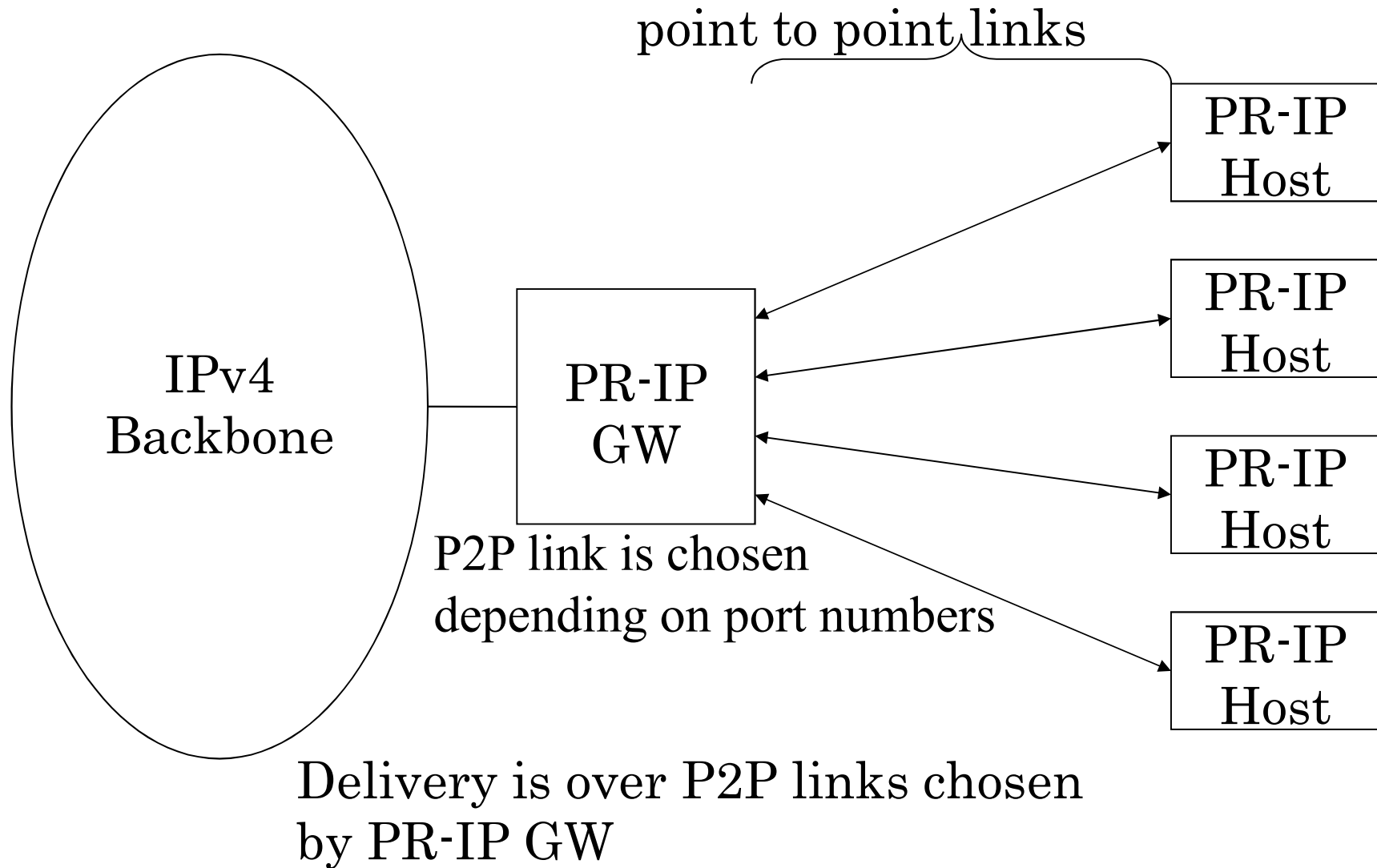
# End to End NAT and Mobility

- if home address/port is behind NAT GW
  - MH knows (static) NAT information of home
  - HA may relay communication between MH and home NAT GW
- if MH is behind foreign NAT GW
  - port number range of foreign and home addresses, in general, different
  - tunnel from HA to MH should be IP over UDP over IP
    - only one port # necessary at foreign network

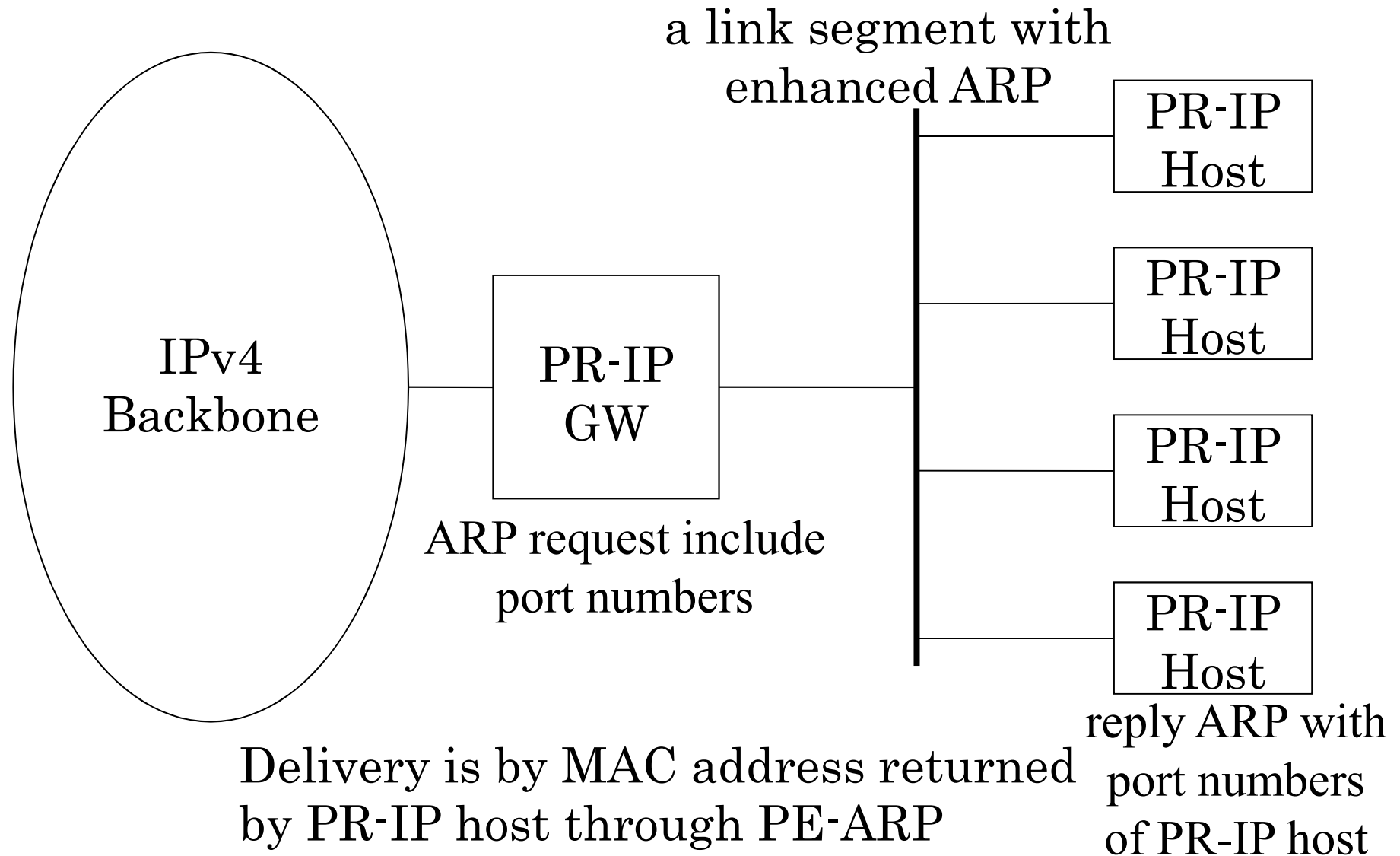
# E2ENAT and PR-IP (Port Restricted IP)



# PR-IP with A+P (Address+Port)

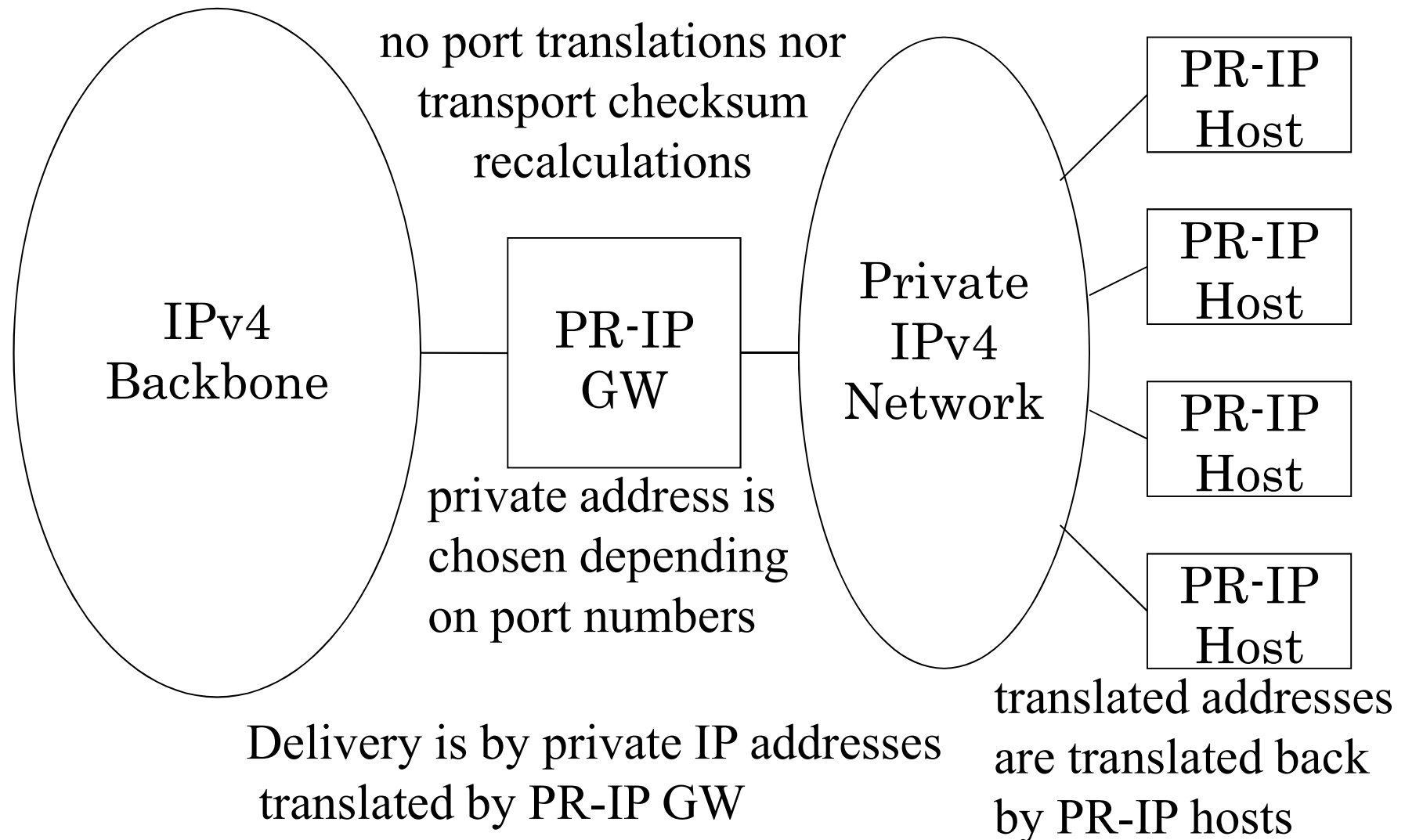


# PR-IP with Port Enhanced ARP





# PR-IP with End to End NAT

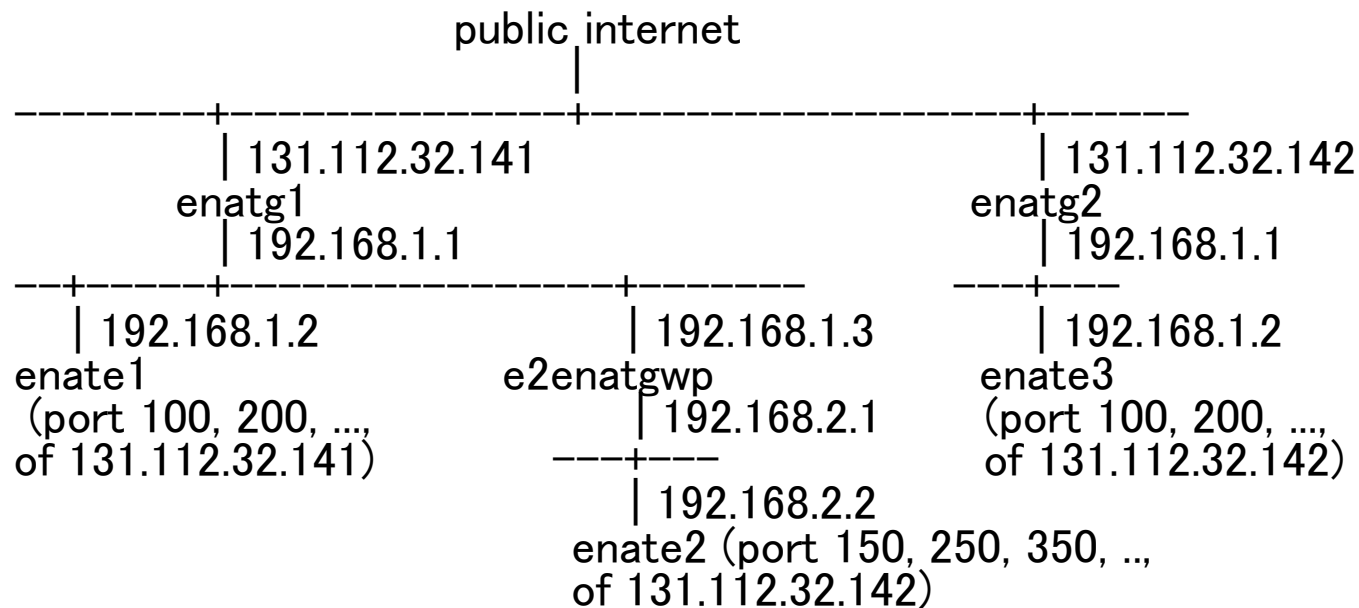


# Implementation

- based on NetBSD5.1 with static NAT only
- essential modifications
  - for forward and reverse address translation
    - several lines to ip\_input.c of hosts
    - several tens of lines to ip\_input.c of GW
  - to restrict source address and port#
    - several hundreds of lines to in\_pcb.c of hosts and GW
    - several lines to ip\_output.c of hosts and GW
      - disable transport checksum computation by NIC card

# Demonstration Environment for End to End NAT

- may use ssh (guest, guest)
  - port numbers 100, 150, 100 to e1, e2, e3



# End to End NAT and Fragmentation

- with IP fragmentation
  - ID should be unique within lifetime of packet
    - ID individually assigned by hosts may collide
- may be remedied by transport checksum
  - 16bit ID is not enough even for unshared address, anyway
    - turn around occur @ 13Mbps with 1500B packets if lifetime is 60s

# End to End NAT and Address Allocation

- E2ENAT can save addresses a lot
- should be assumed for address allocation
  - makes IPv6 unnecessary
  - makes remembering address human
    - remembering IPv6 address is divine
- class E address should also be used
  - in a long run

# End to End NAT and Class E Address

- Class E address is currently abandoned
  - not so many addresses for elaboration
- if address saving by E2ENAT is assumed
  - may use class E for unicast, after transition period
    - as hosts must be modified for E2ENAT
      - modifications for class E may be performed at the same time
    - routers must also be modified
      - a lot easier than adopting to IPv6

# End to End NAT and Prefix in Global Routing Table

- entries finer than /24 in global routing table?
  - /24 means 16M entries (usually too much)
- IPv6 failed to suppress # of entries
  - a lot more than 16M possible
- in a long run
  - end to end multihoming MUST be deployed

# End to End NAT and End Users

- by introducing end to end NAT, end users
  - may operate servers and clients as is
  - no IPv6 necessary
    - address (and port) is easy to remember
  - port # specification by URLs for http not necessary
  - must modify hosts, if legacy NAT is not enough
    - no worse than keep using legacy NAT



# TCP and UDP with Port Length Enhancement (TUPLE)

-- A Scribbled Slate Approach for Internet Addressing and Routing --

Masataka Ohta

Tokyo Institute of Technology

[mohta@necom830.hpcl.titech.ac.jp](mailto:mohta@necom830.hpcl.titech.ac.jp)

# TUPLE

## (TCP and UDP with Port Length Enhancement )

-- A Scribbled Slate Approach for Internet Addressing and Routing --

- TCP and UDP with 6B port numbers
  - and extended transport option field
    - length of “Data Offset” field of TCP is extended
    - unused UDP “Length” field is used as “Data Offset”
    - the option field may contain alternative source addresses for better aggregation
- Named after TUBA (was an IPng candidate)
  - “TCP and UDP with Bigger Addresses (TUBA), A Simple Proposal for Internet Addressing and Routing” (RFC1347)
  - Port numbers of other protocols may also be enhanced

# Header Format of the Current TCP

SRC port		DST port	
Sequence Number			
Acknowledgment Number			
Data Offset	Flags & Reserved bits		Window
Checksum		Urgent Pointer	
Options & Padding (at most 40B long)			

# Header Format of TUPLE TCP

SRC port H		DST port H	
SRC port M		DST port M	
SRC port L		DST port L	
Sequence Number			
Acknowledgment Number			
Data Offset	Flags & Reserved	Window	
Checksum		Urgent Pointer	
Options & Padding (at most 972B long)			

# Header Format of the Current UDP

SRC port	DST port
Length	Checksum

Not necessary



# Header Format of TUPLE UDP

SRC port H		DST port H
SRC port M		DST port M
SRC port L		DST port L
Data Offset	Reserved	Checksum
Options & Padding (at most 1004B long)		

# Packet Format of the Current ICMP ECHO Request & Reply

IP Header		
Type	Code	Checksum
Identifier		Sequence Number
Data (variable length)		

# Packet Format of TUPLE ICMP ECHO Request

IP Header		
Type (8)	Code	Checksum
SRC port H		DST port H
SRC port M		DST port M
SRC port L		DST port L
Remaining Data (variable length)		



# Packet Format of TUPLE ICMP ECHO Reply

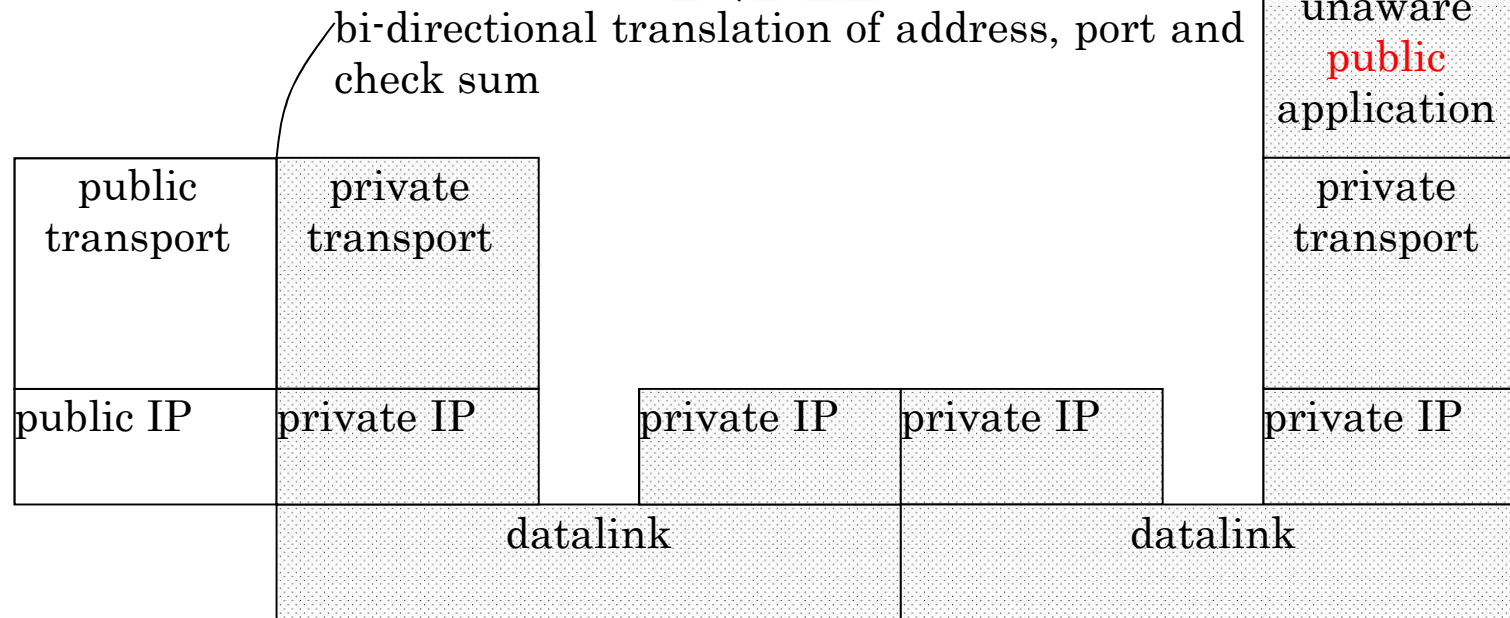
IP Header		
Type (0)	Code	Checksum
DST port H		SRC port H
DST port M		SRC port M
DST port L		SRC port L
Remaining Data (variable length)		


# Almost End to End NAT

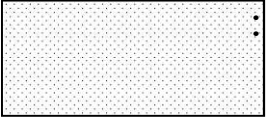
Masataka Ohta

# Layering Structure of Legacy

## NAT



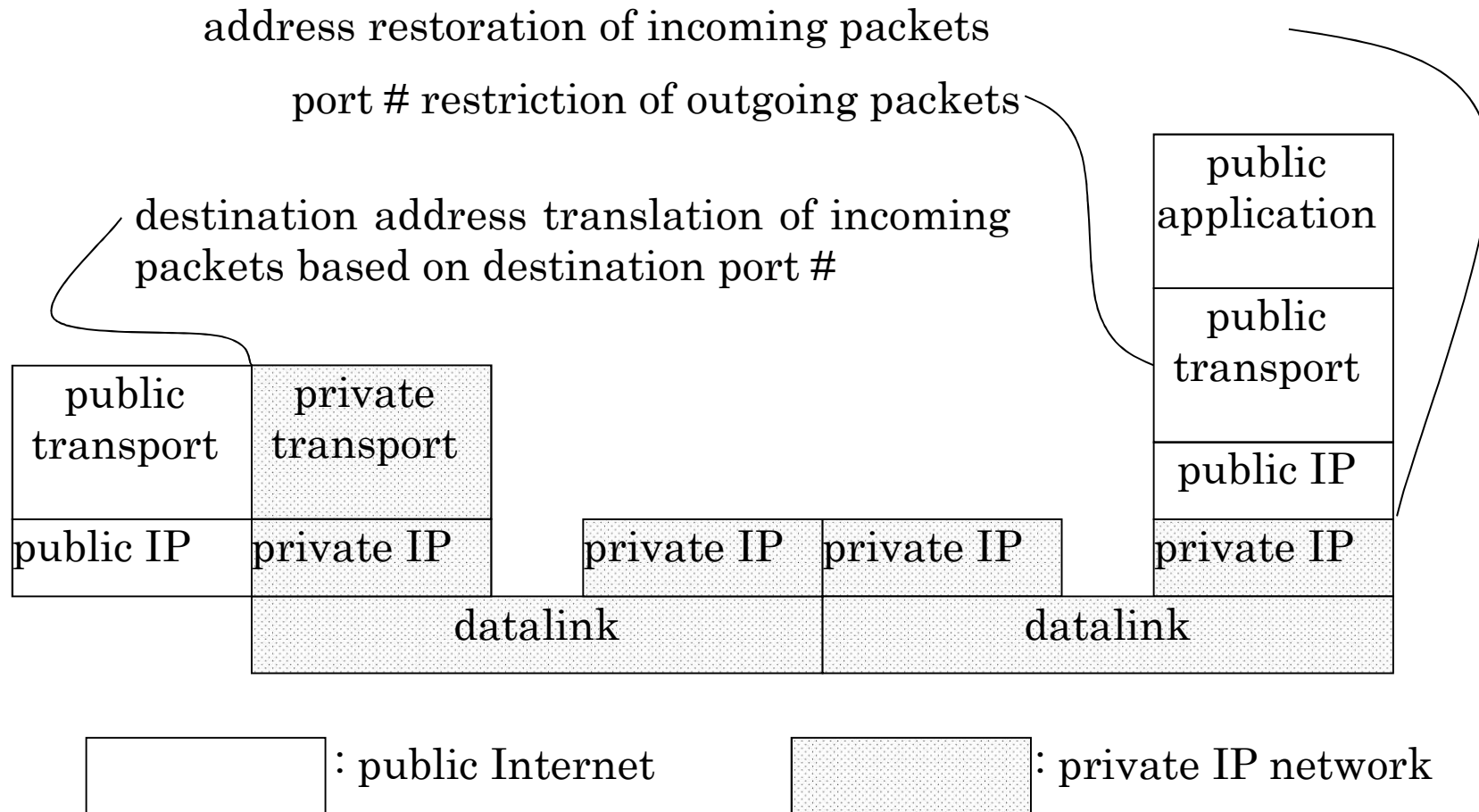
 : public Internet

 : private IP network

# End to End NAT

- NAT function offered by end systems
  - do almost nothing on NAT GW
    - translate destination IP address of incoming packets
      - keep port # and transport checksum as is
  - end systems restore destination address
    - transport checksum is correct
  - use global source address and restrict source port # to those assigned to end system
    - no port # collision at NAT GW
      - no port # translation necessary

# Layering Structure of End to End NAT



# Properties of End to End NAT

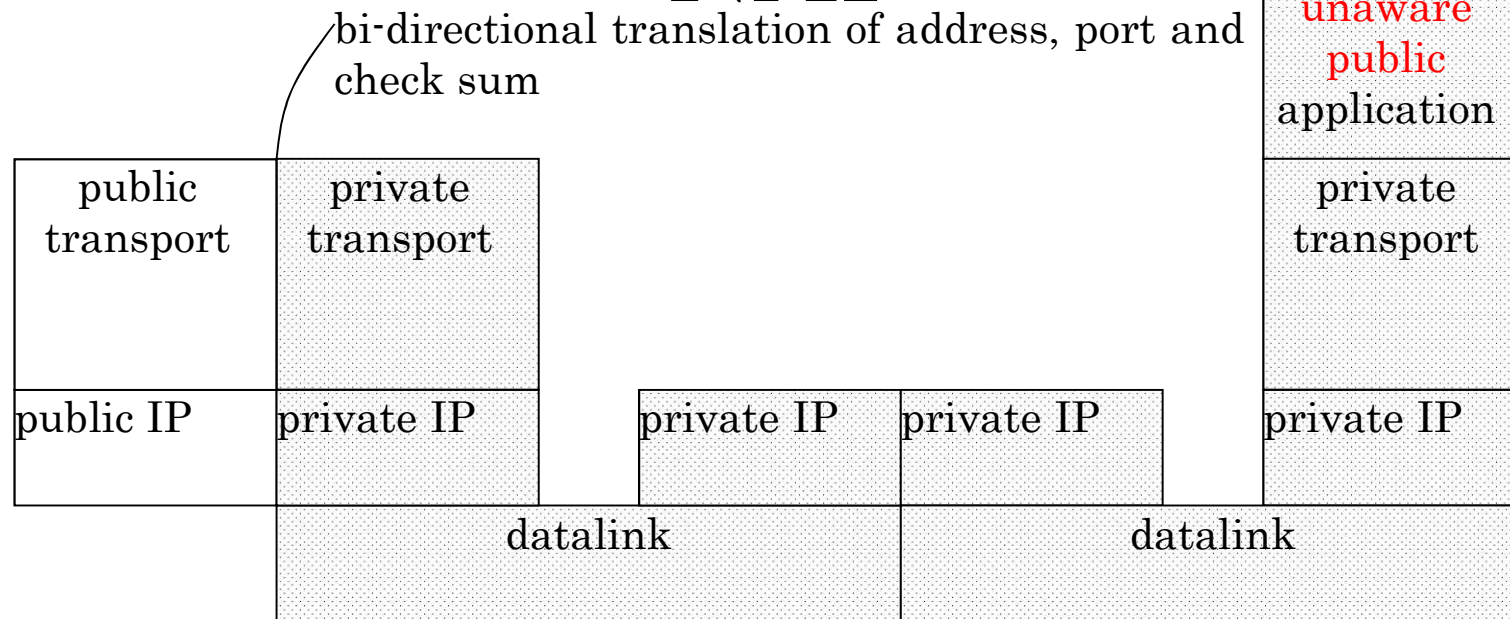
- All the existing transport protocols work
  - as long as port # or something like that exists
    - may regard ID and seq # of ICMP and SPI of IPsec as port \$#
- deployment is not easy
  - NAT GW must be modified
    - though backward compatible
  - needs some protocol (DHCP? PPP? PCP?)  
modifications to offer NAT information to end systems


# UPnP (Universal Plug and Play)

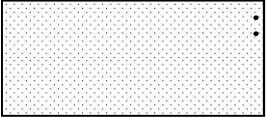
- protocol to autoconfigure hosts
  - include port mapping information of NAT
    - specifications on WANIPConnection service
  - assume applications on hosts modified to be UPnP capable
- implemented on most, if not all, NAT GWs
- only transport layers of TCP and UDP (and ICMP) are supported

# Layering Structure of Legacy

## NAT



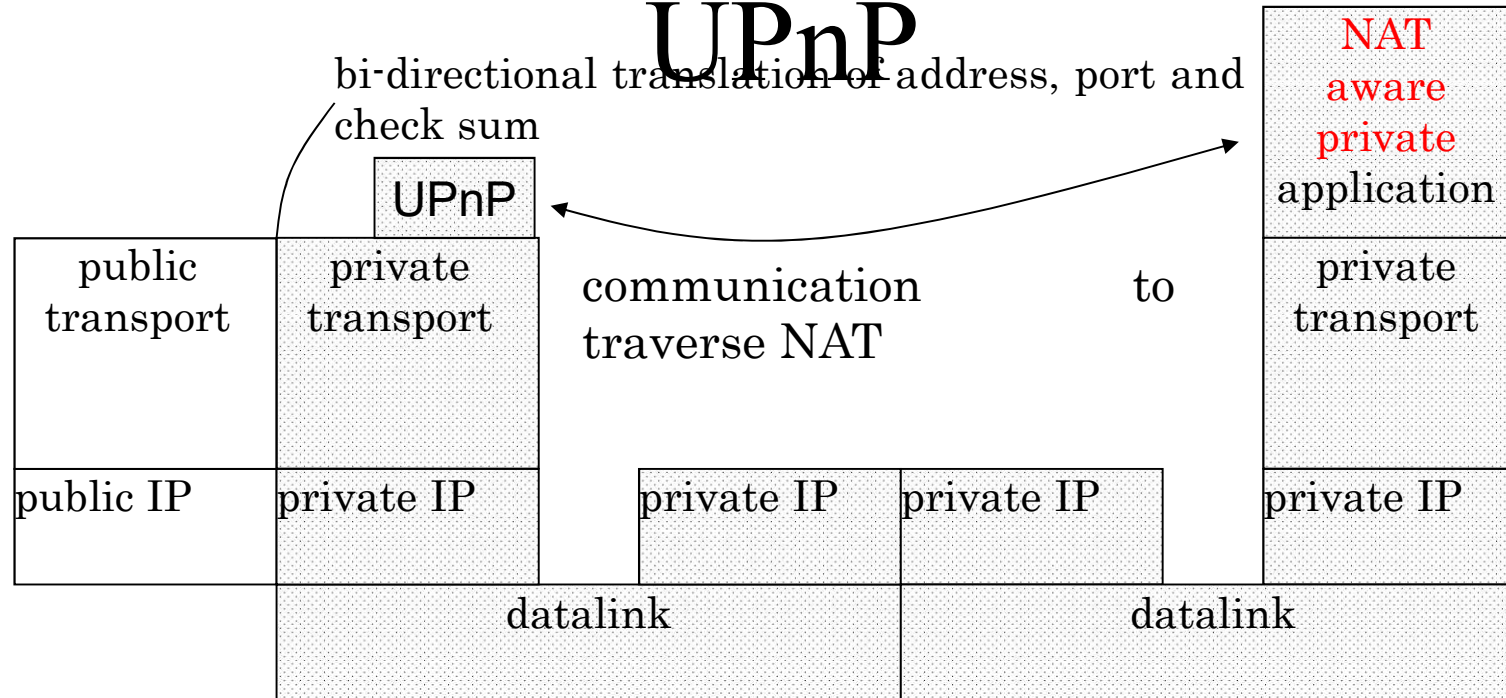
 : public Internet


 : private IP network

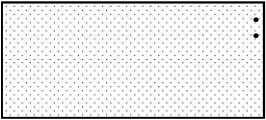


# Layering Structure Assumed by

## UPnP



 : public Internet

 : private IP network

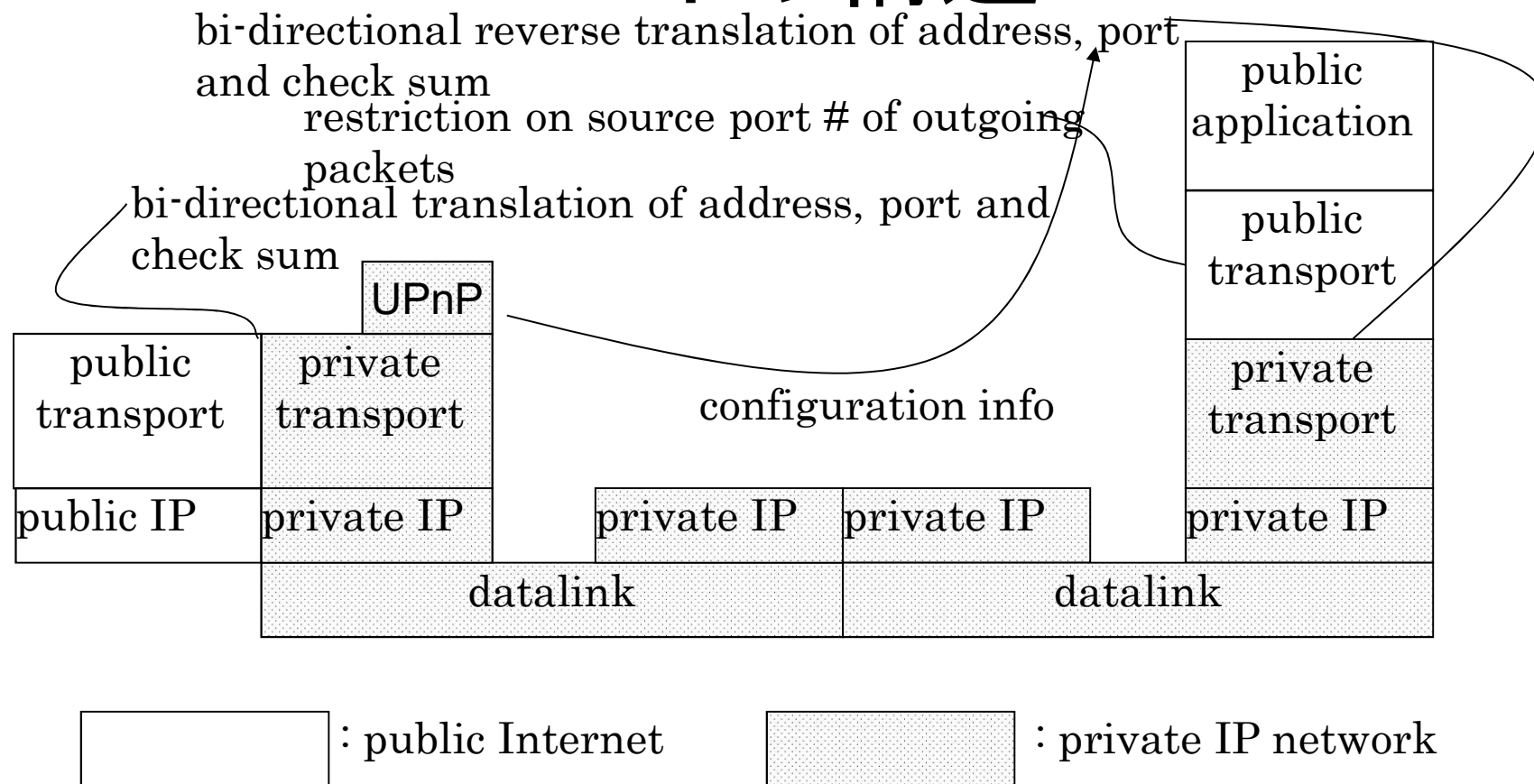
# UPnP and End to End Argument

- UPnP is literally
  - with the knowledge and help of the application standing at the end points of the communication
- but application-wise adoption to NAT needs modifications to all applications
- “application” in E2E Argument dose not mean application layer today
  - protocol stack at that time was not well layered
  - adoption may be done at lower layer

# Almost End to End NAT

- UPnP GW combined with End to End NAT
- reverse translation of port mapping by UPnP GW by transport layer of hosts
  - no modification necessary on applications
    - though port # range is restricted
  - “almost” because only TCP, UDP and ICMP are supported

# Almost End to End NATの レイヤ構造



# Properties of Almost End to End NAT

- can use UPnP capable existing NAT GW as is
- NAT configuration information is announced by UPnP
  - global address: `GetExternalIPAddress()`
  - port translation: `GetListOfPortMappings()`
    - `GetGenericPortMappingEntry()` for UPnP1.0
- host modification base on E2ENAT for NetBSD5.1 is trivially easy
  - unless port #s are not translated on NAT GWs
    - no reason to do so

# Port Number Shortage by NAT?

- some applications on clients needs large # of ports?
  - google map, for example
  - is not a problem of proper implementation
    - implementations (incl. google map) must take care of temporary shortage of port #s (was not detectable with legacy NAT, EAGAIN with (almost) E2ENAT)
    - source port of client may be shared by many connect(), of REUSEPORT is set with setsockopt()
      - connect(), after binding socket to shared port

# Conclusion

- Almost End to End NAT enables
  - UPnP unaware applications over TCP/UDP on hosts behind UPnP capable NAT GW works as is keeping full E2E transparency
  - price increase of IPv4 addresses motivate deployment
- address space of IPv4 will last forever?
  - especially with class E
- support for DNS SRV RR by browsers necessary

# Wrap-up

- E2ENAT save IPv4 address a lot, keeping almost all properties (**including E2E transparency**) of the Internet today
- address allocation assuming E2ENAT makes IPv4 address space (incl. class E) last forever
- who needs poor IPv6?