15-441
15-641

Computer Networking

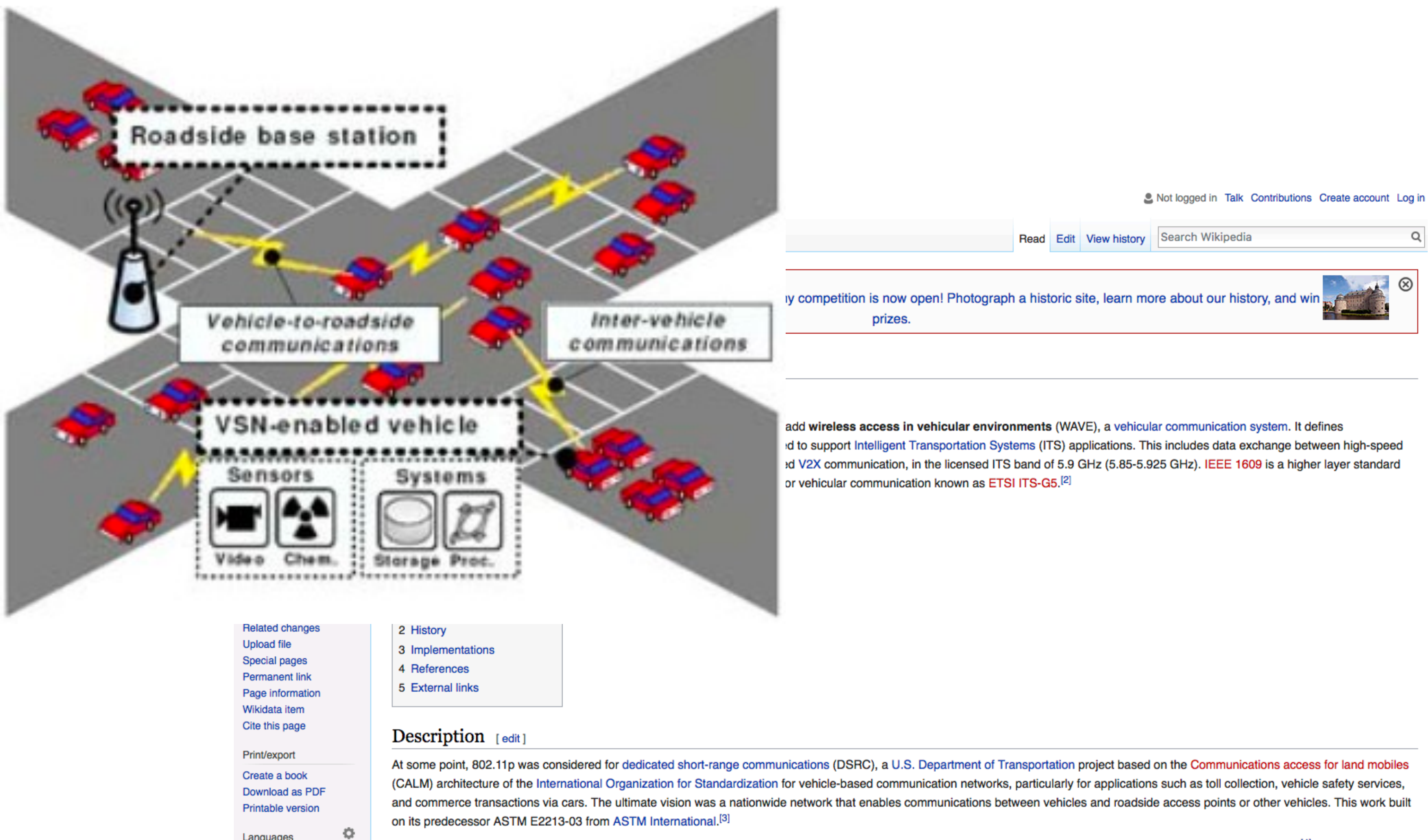
Lecture 6 – The Internet Protocol

Peter Steenkiste
Justine Sherry

Fall 2017

www.cs.cmu.edu/~prs/15-441-F16

What nerdy professors think about on the weekends



What nerdy professors do on the weekends

- https://twitter.com/justinesherry/status/905980721398984705

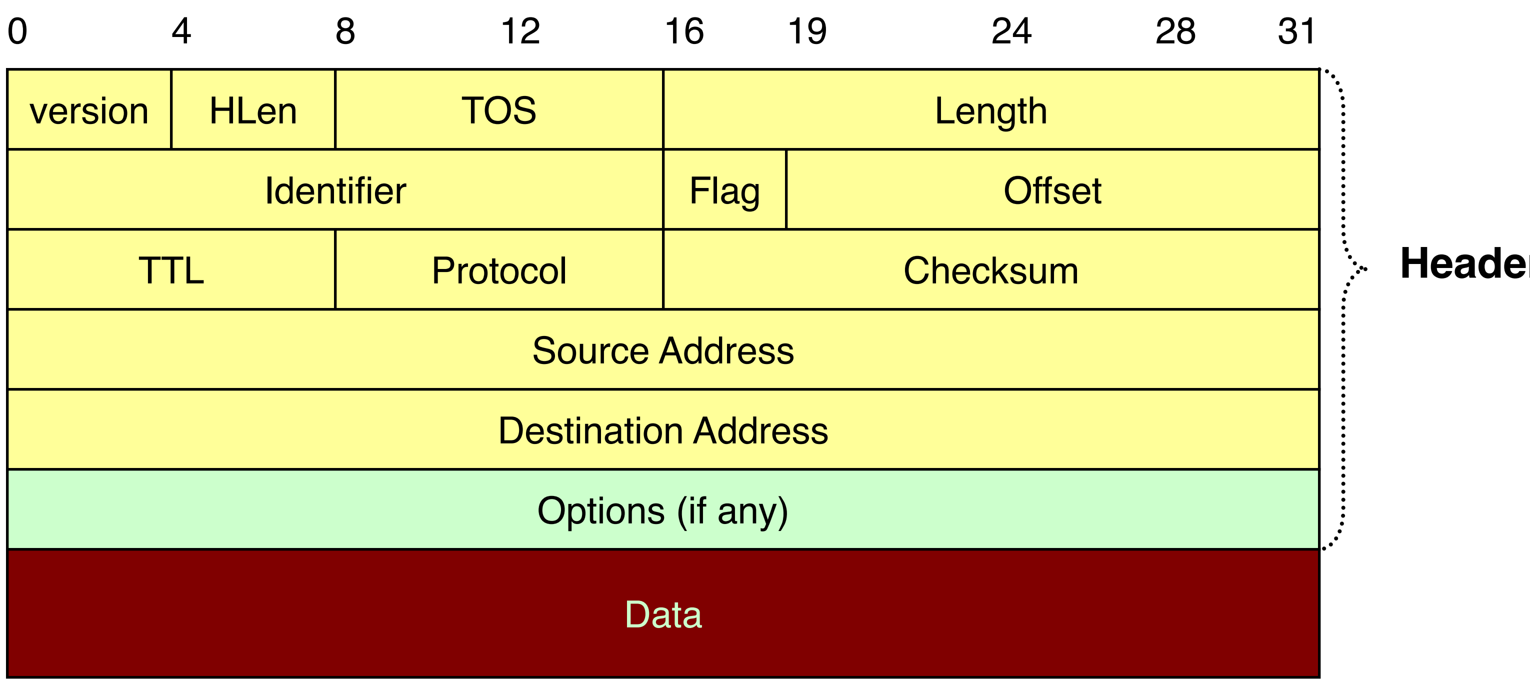
Outline

- The IP protocol
  - IPv4
  - IPv6
- IP in practice
  - Network address translation
  - Address resolution protocol
  - Tunnels

IP Service Model

- Low-level communication model provided by Internet
- Datagram
  - Each packet self-contained
    - All information needed to get to destination
    - No advance setup or connection maintenance
  - Analogous to letter or telegram

IPv4 Packet Format



IP Delivery Model

- *Best effort service*
  - Network will do its best to get packet to destination
- Does NOT guarantee:
  - Any maximum latency or even ultimate success
  - Informing the sender if packet does not make it
  - Delivery of packets in same order as they were sent
  - Just one copy of packet will arrive
- Implications
  - Scales very well (really, it does)
  - Higher level protocols must make up for shortcomings
    - Reliably delivering ordered sequence of bytes → TCP
  - Some services not feasible (or hard)
    - Latency or bandwidth guarantees

Designing the IP header

- Think of the IP header as an interface
  - between the source and destination end-systems
  - between the source and network (routers)
- Designing an interface
  - what task(s) are we trying to accomplish?
  - what information is needed to do it?
- Header reflects information needed for basic tasks

What are these tasks? (in network)

- Parse packet
- Carry packet to the destination
- Deal with problems along the way
  - loops
  - corruption
  - packet too large
- Accommodate evolution
- Specify any special handling

What information do we need?

- Parse packet
- Carry packet to the destination
- Deal with problems along the way
  - loops
  - corruption
  - packet too large
- Accommodate evolution
- Specify any special handling

What information do we need?

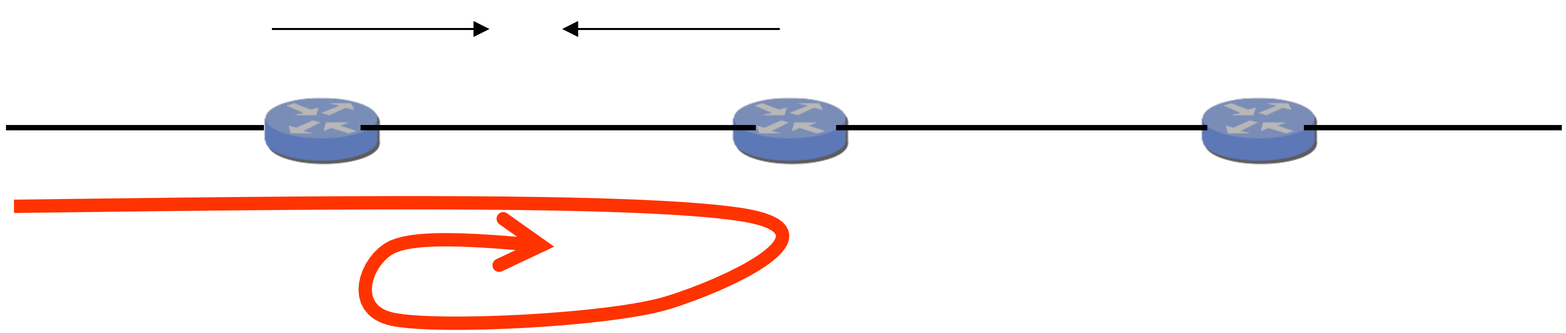
- Parse packet
  - *IP version number (4 bits), packet length (16 bits)*
- Carry packet to the destination
  - *Destination's IP address (32 bits)*
- Deal with problems along the way
  - loops:
  - corruption:
  - packet too large:

What information do we need?

- Parse packet
  - *IP version number (4 bits), packet length (16 bits)*
- Carry packet to the destination
  - *Destination's IP address (32 bits)*
- Deal with problems along the way
  - loops: *TTL (8 bits)*
  - corruption: *checksum (16 bits)*
  - packet too large: *fragmentation fields (32 bits)*

Preventing Loops (TTL)

- Forwarding loops cause packets to cycle for a looong time
  - left unchecked would accumulate to consume all capacity



- Time-to-Live (TTL) Field (8 bits)
  - decremented at each hop, packet discarded if reaches 0
  - ...and “time exceeded” message is sent to the source

Header Corruption (Checksum)

- Checksum (16 bits)
  - Particular form of checksum over packet header
- If not correct, router discards packets
  - So it doesn't act on bogus information
- Checksum recalculated at every router
  - Why?

Fragmentation

- Every link has a "Maximum Transmission Unit" (MTU)
  - largest number of bits it can carry as one unit

- A router can split a packet into multiple "fragments" if the packet size exceeds the link's MTU

- Must reassemble to recover original packet

- Will return to fragmentation shortly…

What information do we need?

- Parse packet
  - *IP version number (4 bits), packet length (16 bits)*
- Carry packet to the destination
  - *Destination's IP address (32 bits)*
- Deal with problems along the way
  - *TTL (8 bits)*, *checksum (16 bits), fragmentation (32 bits)*
- Accommodate evolution
  - *version number (4 bits)* *(+ fields for special handling)*
- Specify any special handling

Special handling

- “Type of Service” (8 bits)
  - allow packets to be treated differently based on needs
    - e.g., indicate priority, congestion notification
  - has been redefined several times
  - now called “Differentiated Services Code Point (DSCP)”

Options

- Optional directives to the network
  - not used very often
  - 16 bits of metadata + option-specific data
- Examples of options
  - Record Route
  - Strict Source Route
  - Loose Source Route
  - Timestamp
  - …..

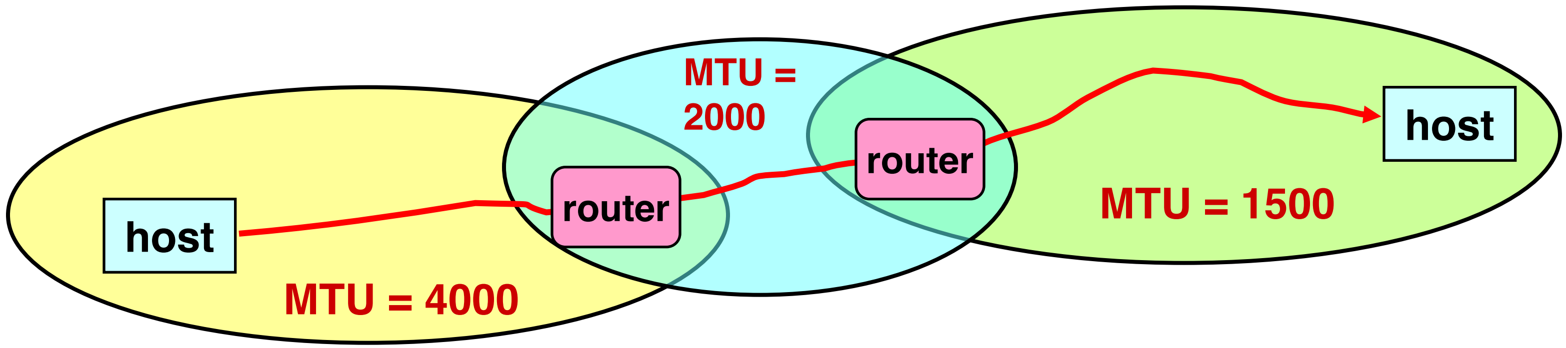
IP Router Implementation: Fast Path versus Slow Path

- Common case: Switched in silicon ("fast path")
  - Almost everything
- Weird cases: Handed to CPU ("slow path", or "process switched")
  - Fragmentation
  - TTL expiration (traceroute)
  - IP option handling
- Slow path is evil in today's environment
  - "Christmas Tree" attack sets weird IP options, bits, and overloads router
  - Developers cannot (really) use things on the slow path
    - Slows down their traffic – not good for business
    - If it became popular, they are in trouble!

What information do we need?

- Parse packet
  - *IP version number (4 bits), packet length (16 bits)*
- Carry packet to the destination
  - *Destination's IP address (32 bits)*
- Deal with problems along the way
  - *TTL (8 bits), checksum (16 bits), fragmentation (32 bits)*
- Accommodate evolution
  - *version number (4 bits) (+ fields for special handling)*
- Specify any special handling
  - *ToS (8 bits), Options (variable length)*

IP Fragmentation

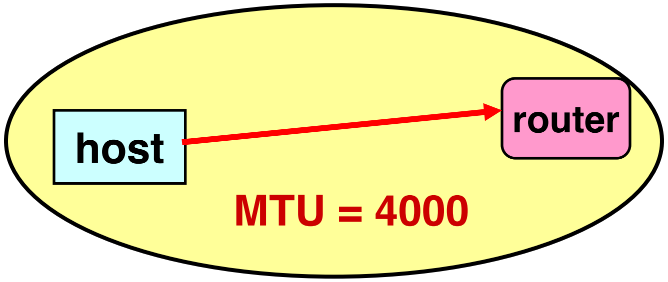


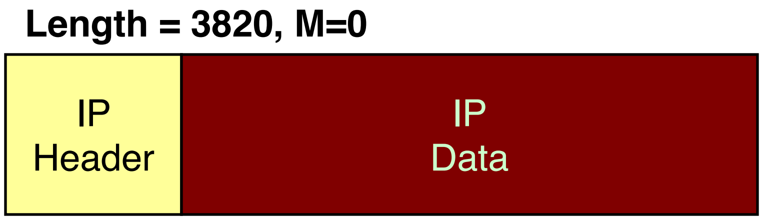
- Every network has own Maximum Transmission Unit (MTU)
  - Largest IP datagram it can carry within its own packet frame
    - E.g., Ethernet is 1500 bytes
  - Don't know MTUs of all intermediate networks in advance
- IP Solution
  - When hit network with small MTU, router fragments packet
  - Destination host reassembles the paper – why?

Fragmentation Related Fields

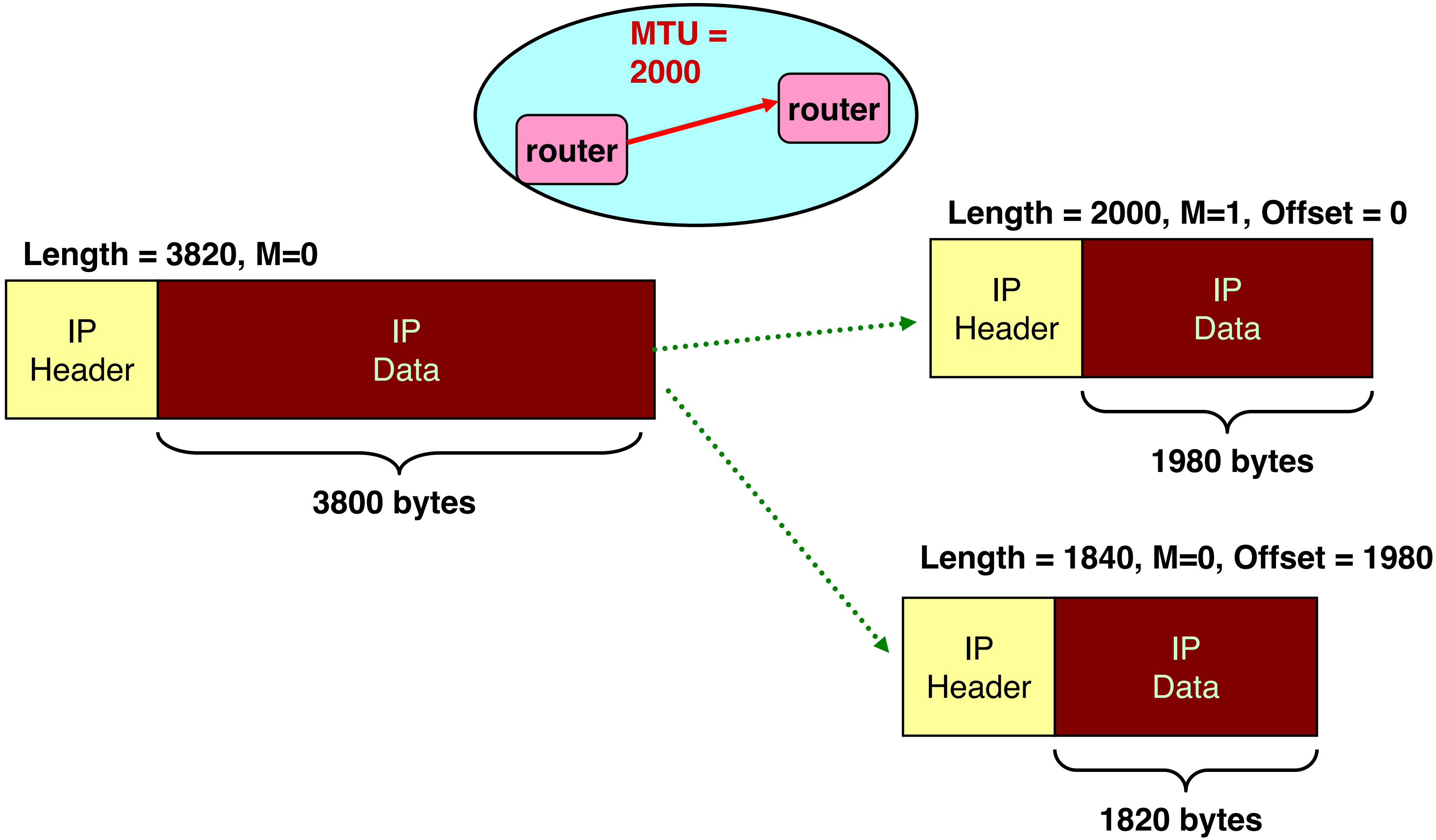
- Length
  - Length of IP fragment
- Identification
  - To match up with other fragments
- Flags
  - Don't fragment flag
  - More fragments flag
- Fragment offset
  - Where this fragment lies in entire IP datagram
  - Measured in 8 octet units (13 bit field)

IP Fragmentation Example #1





IP Fragmentation Example #2



Fragmentation is Harmful

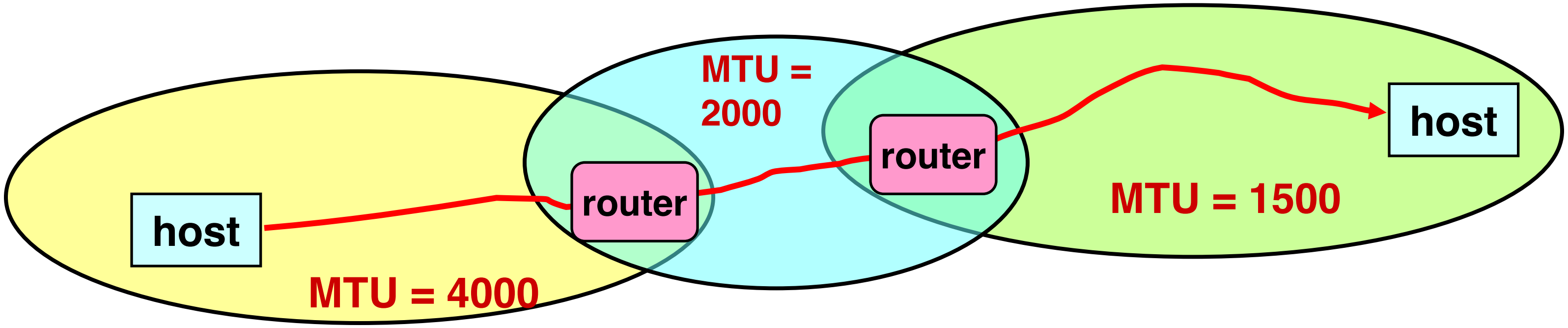
- Uses resources poorly
  - Forwarding costs per packet
  - Best if we can send large chunks of data
  - Worst case: packet just bigger than MTU
- Poor end-to-end performance
  - Loss of a fragment

- Path MTU discovery protocol → determines minimum MTU along route
  - Uses ICMP error messages
- Common theme in system design
  - Assure correctness by implementing complete protocol
  - Optimize common cases to avoid full complexity

Internet Control Message Protocol (ICMP)

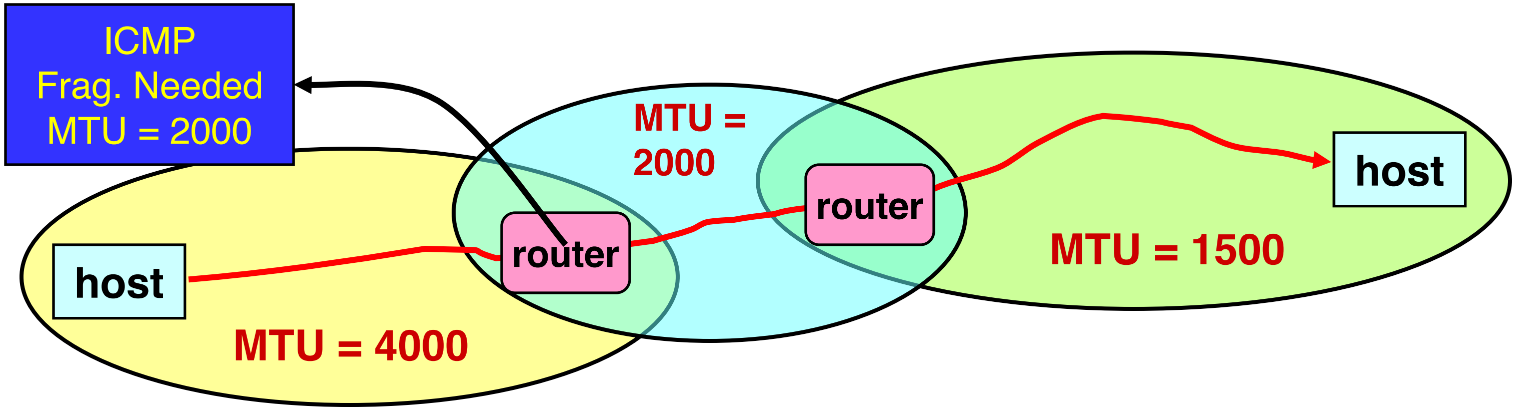
- Short messages used to send error & other control information
- Some functions supported by ICMP:
  - Ping request /response: check whether remote host reachable
  - Destination unreachable: Indicates how packet got & why couldn’t go further
  - Flow control: Slow down packet transmit rate
  - Redirect: Suggest alternate routing path for future messages
  - Router solicitation / advertisement: Helps newly connected host discover local router
  - Timeout: Packet exceeded maximum hop limit
- How useful are they functions today?

IP MTU Discovery with ICMP



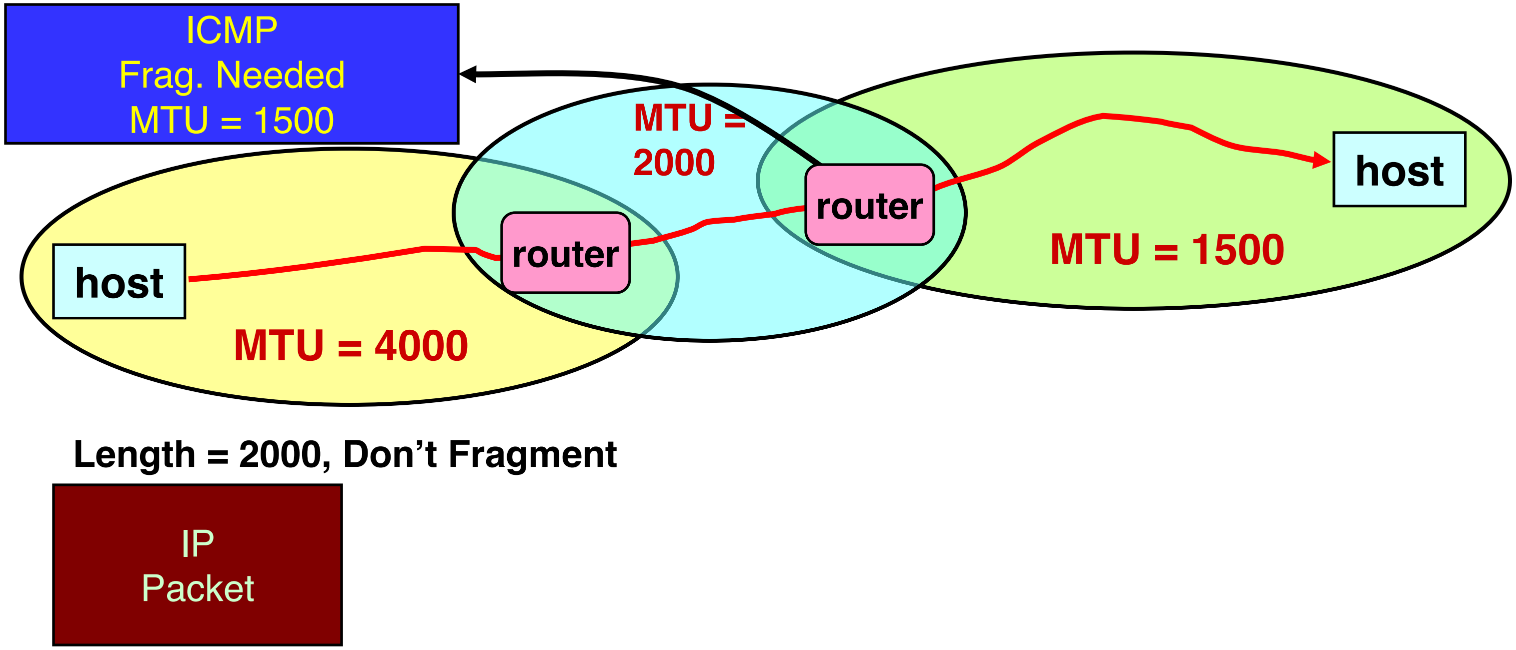
- Typically send series of packets from one host to another
- Typically, all will follow same route
  - Routes remain stable for minutes at a time
- Makes sense to determine path MTU before sending real packets
- Operation: Send max-sized packet with “do not fragment” flag set
  - If encounters problem, ICMP message will be returned
    - “Destination unreachable: Fragmentation needed”
    - Usually indicates MTU problem encountered
- ICMP abuse? Other solutions?

IP MTU Discovery with ICMP

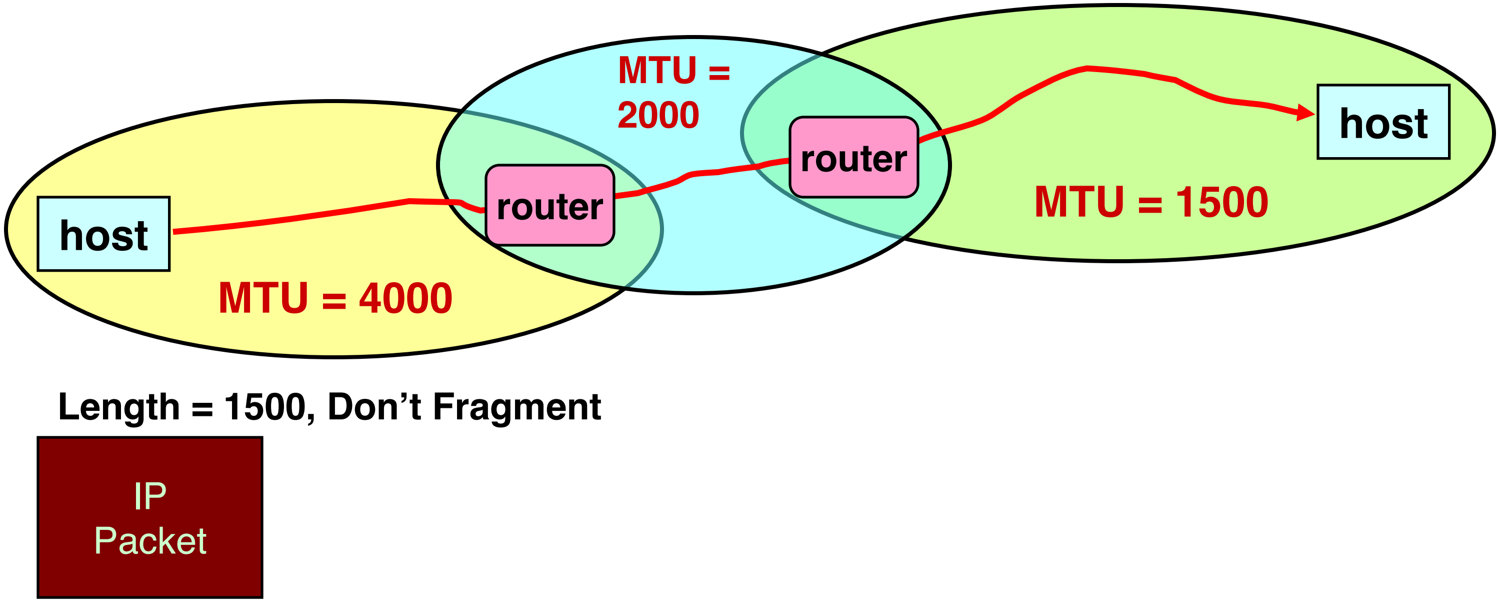


Length = 4000, Don't Fragment

IP MTU Discovery with ICMP



IP MTU Discovery with ICMP



- When successful, no reply at IP level
  - "No news is good news"
- Higher level protocol might have some form of acknowledgement

Important Concepts

- Base-level protocol (IP) provides minimal service level
  - Allows highly decentralized implementation
  - Each step involves determining next hop
  - Most of the work at the endpoints
- ICMP provides low-level error reporting

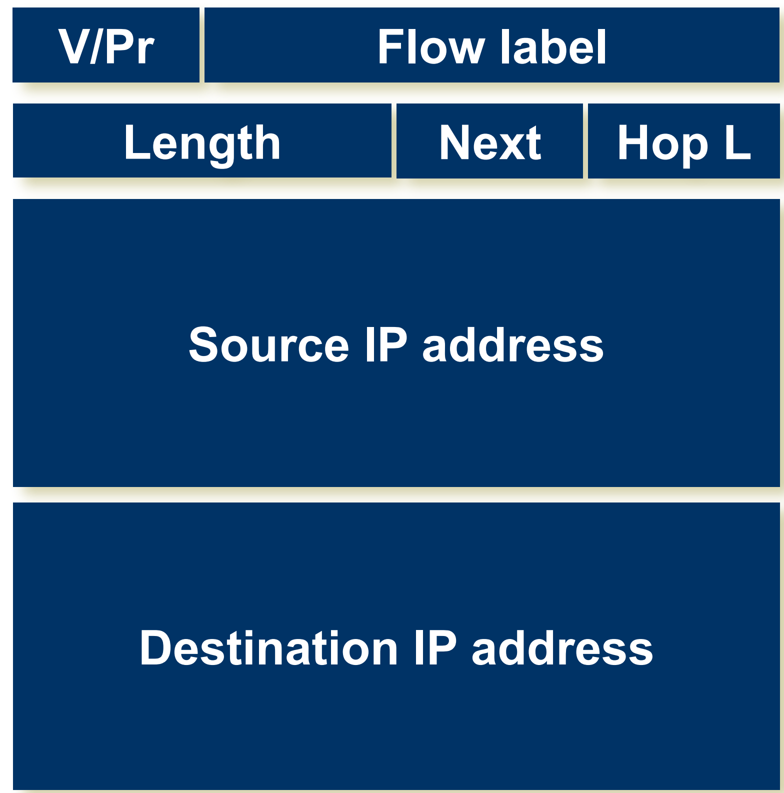
- IP forwarding → global addressing, alternatives, lookup tables
- IP addressing → hierarchical, CIDR
- IP service → best effort, simplicity of routers
- IP packets → header fields, fragmentation, ICMP
  - Interface to higher layers

Outline

- The IP protocol
  - IPv4
  - IPv6
- IP in practice
  - Network address translation
  - Address resolution protocol
  - Tunnels

IPv6

- "Next generation" IP.
- Most urgent issue: increasing address space.
  - 128 bit addresses
- Simplified header for faster processing:
  - No checksum (why not?)
  - No fragmentation (really?)
- Support for guaranteed services: priority and flow id
- Options handled as "next header"
  - reduces overhead of handling options



IPv6 Address Size Discussion

- Do we need more addresses? Probably, long term
  - Big panic in 90s: "We're running out of addresses!"
  - Big worry: Devices. Small devices. Cell phones, toasters, everything.
- 128 bit addresses provide space for structure (good!)
  - Hierarchical addressing is much easier
  - Assign an entire 48-bit sized chunk per LAN – use Ethernet addresses
  - Different chunks for geographical addressing, the IPv4 address space,
  - Perhaps help clean up the routing tables - just use one huge chunk per ISP and one huge chunk per customer.

010	**Registry**	**Provider**	**Subscriber**	**Sub Net**	**Host**

IPv6 Header Cleanup: Options

- 32 IPv4 options → variable length header
  - Rarely used
  - No development / many hosts/routers do not support
    - Worse than useless: Packets w/options often even get dropped!
  - Processed in “slow path”.
- IPv6 options: “Next header” pointer
  - Combines “protocol” and “options” handling
    - Next header: “TCP”, “UDP”, etc.
  - Extensions header: Chained together
  - Makes it easy to implement host-based options
  - One value “hop-by-hop” examined by intermediate routers
    - E.g., “source route” implemented only at intermediate hops

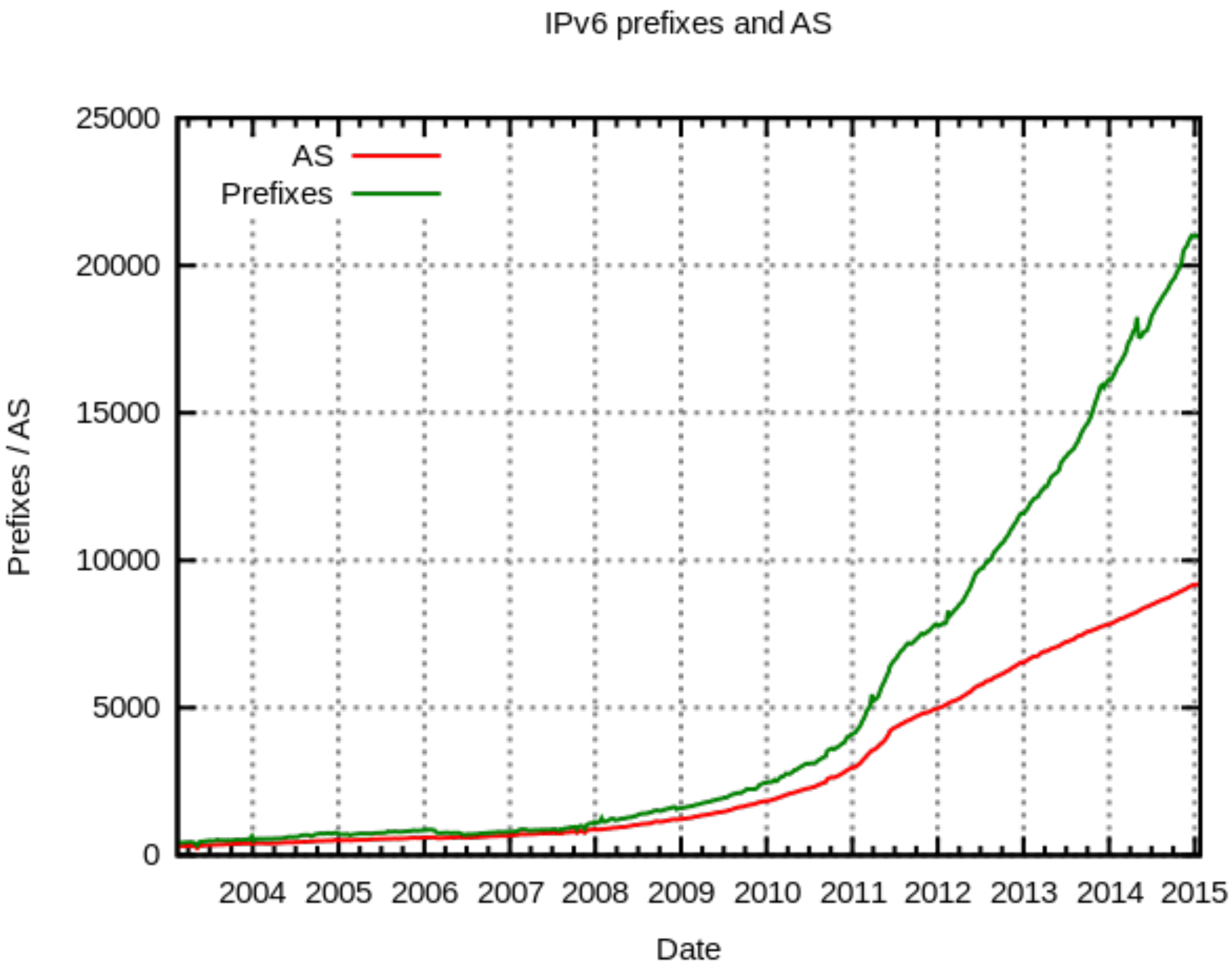
IPv6 Header Cleanup: “no”

- No checksum
  - Motivation was efficiency: If packet corrupted at hop 1, don’t waste b/w transmitting on hops 2..N.
  - Useful when corruption frequent, b/w expensive
  - Today: corruption is rare, bandwidth is cheap
- No fragmentation
  - Router discard packets, send ICMP “Packet Too Big” → host does MTU discovery and fragments
  - Reduced packet processing and network complexity.
  - Increased MTU a boon to application writers
  - Hosts can still fragment - using fragmentation header. Routers don’t deal with it any more.

Migration from IPv4 to IPv6

- Interoperability with IP v4 is necessary for incremental deployment.
  - No “flag day”
- Fundamentally hard because a (single) IP protocol is critical to achieving global connectivity across the internet
- Process uses a combination of mechanisms:
  - Dual stack operation: IP v6 nodes support both address types
  - Tunnel IP v6 packets through IP v4 clouds
  - IPv4-IPv6 translation at edge of network
    - NAT must not only translate addresses but also translate between IPv4 and IPv6 protocols
  - IPv6 addresses based on IPv4 – no benefit!
- 20 years later, this is still a major challenge!

Things are looking up?



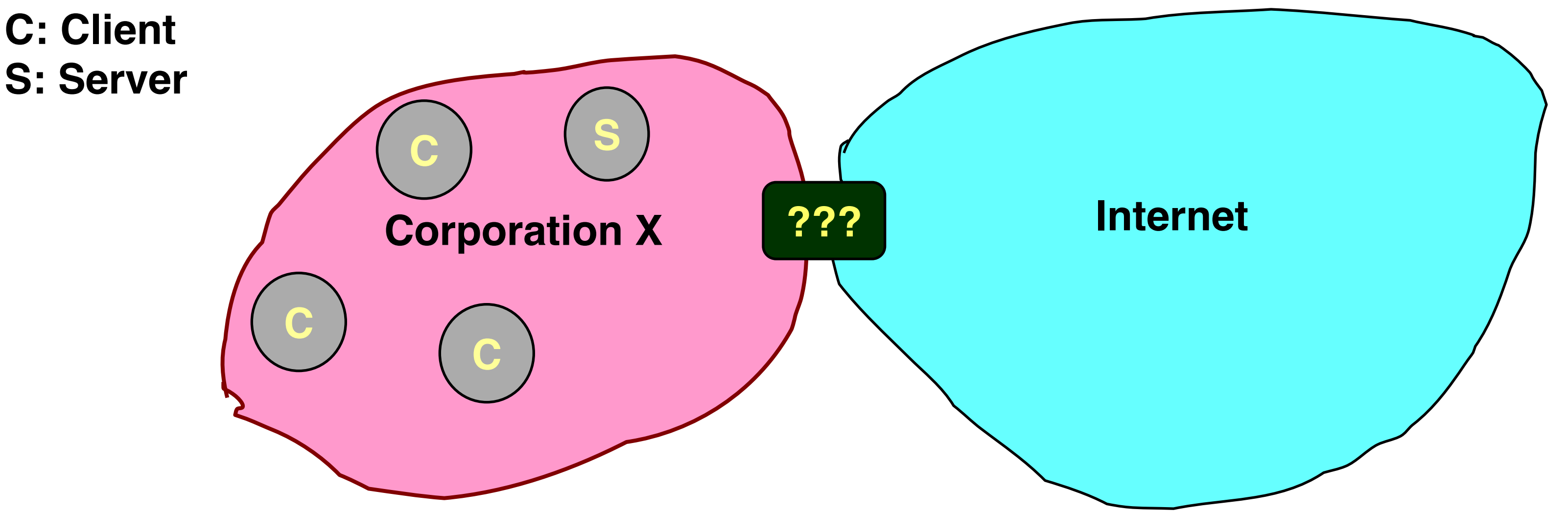
Outline

- The IP protocol
  - IPv4
  - IPv6
- IP in practice
  - Network address translation
  - Address resolution protocol
  - Tunnels

How have we made it so far with IPv4, though?

- Original IP Model: Every host has unique IP address
- This has very attractive properties …
  - Any host can communicate with any other host
  - Any host can act as a server
    - Just need to know host ID and port number
- … but the system is open – complicates security
  - Any host can attack any other host
  - It is easy to forge packets
    - Use invalid source address
- … and it places pressure on the address space
  - Every host requires "public" IP address

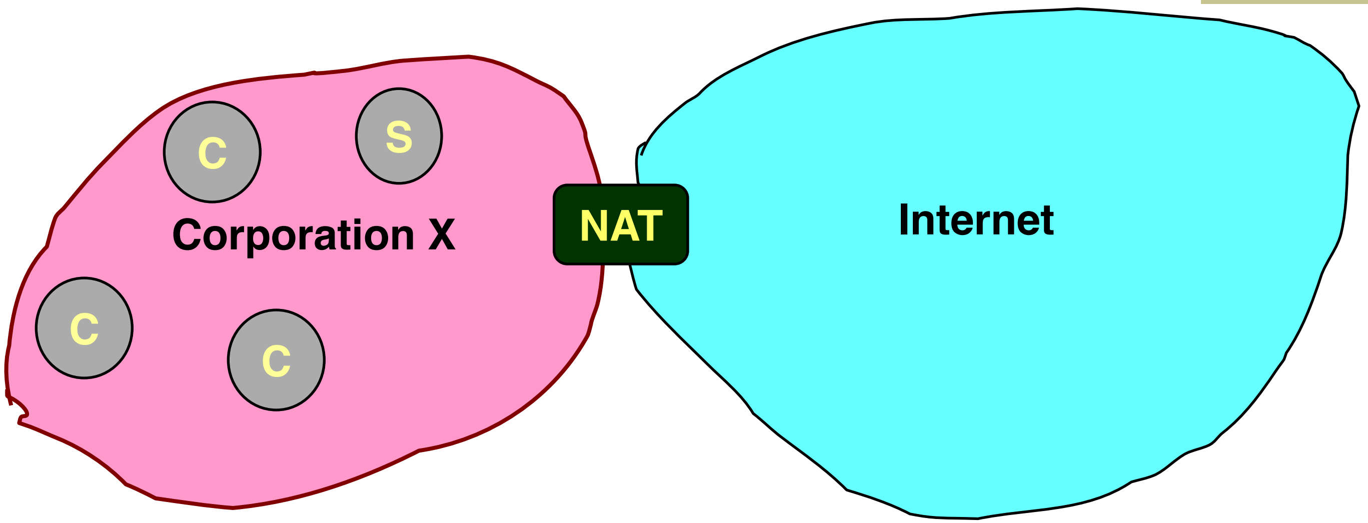
Challenges When Connecting to Public Internet



- Not enough IP addresses for every host in organization
  - Increasingly hard to get large address blocks
- Security
  - Don't want every machine in organization known to outside world
  - Want to control or monitor traffic in / out of organization

But not All Hosts are Equal!

C: Client
S: Server

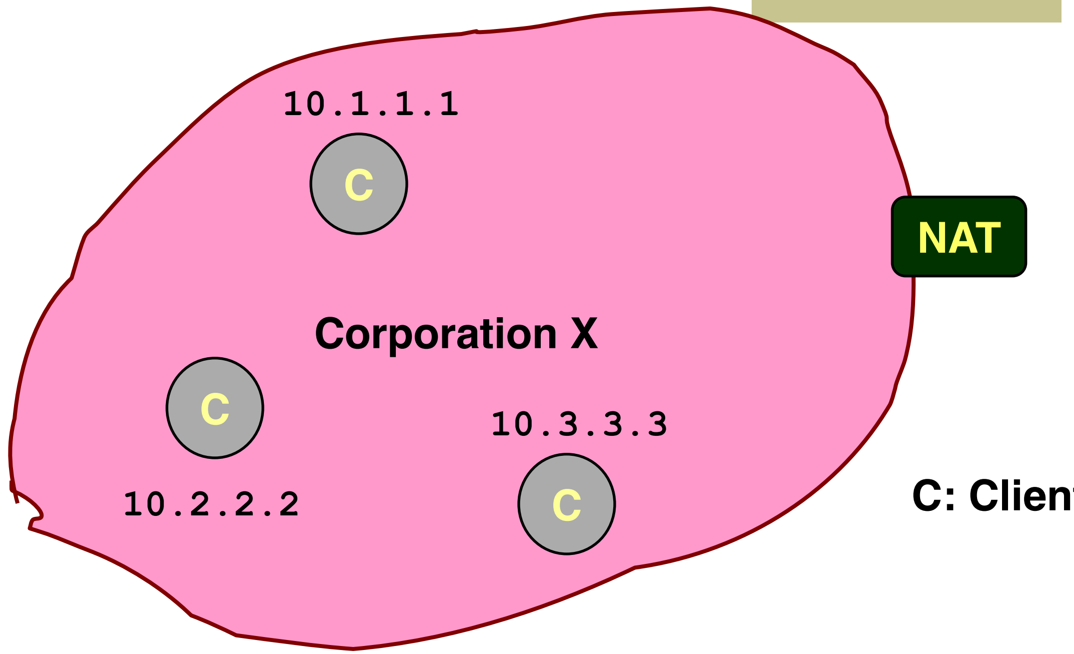


- Most machines within organization are used by individuals
  - For most applications, they act as clients
- Only a small number of machines act as servers for the entire organization
  - E.g., mail server, web, ..
  - All traffic to outside passes through firewall

(Most) machines within organization do not need public IP addresses!

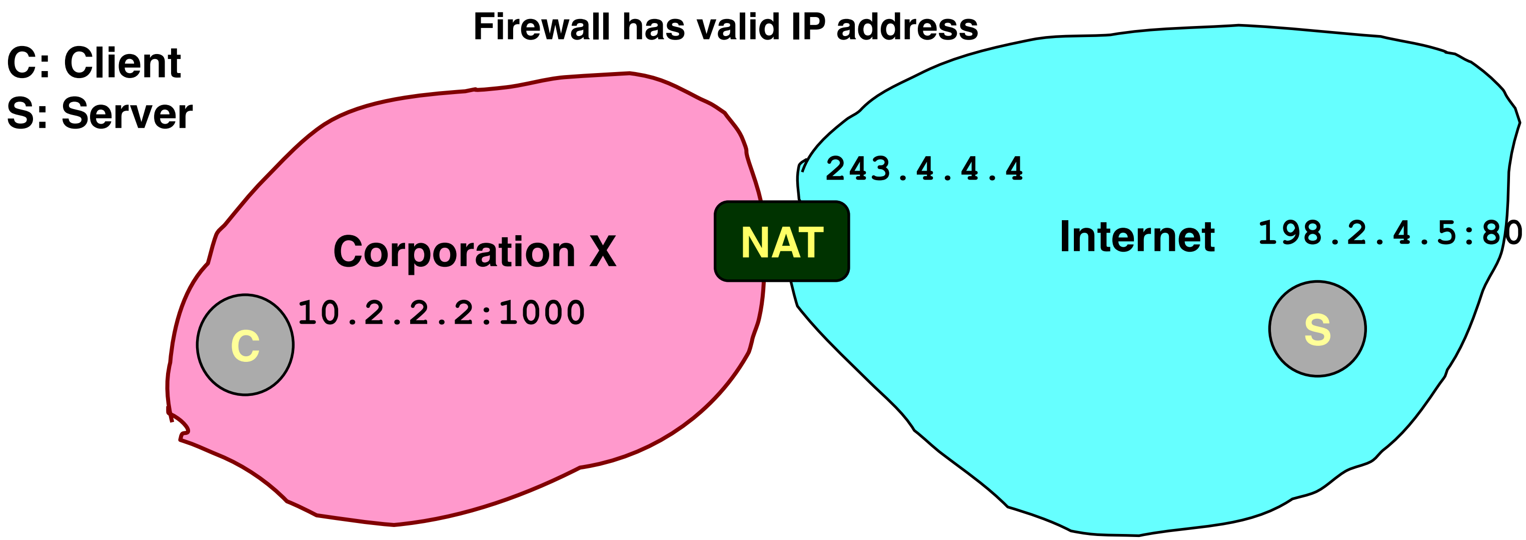
Reducing Address Use: Network Address Translation

- Within organization: assign each host a private IP address
  - IP addresses blocks 10/8 & 192.168/16 are set aside for this
  - Route within organization by IP protocol
  - Can do subnetting, ..



- The NAT translates between public and private IP addresses as packets travel to/from the public Internet
  - It does not let any packets from internal nodes “escape”
  - Outside world does not need to know about internal addresses

NAT: Opening Client Connection

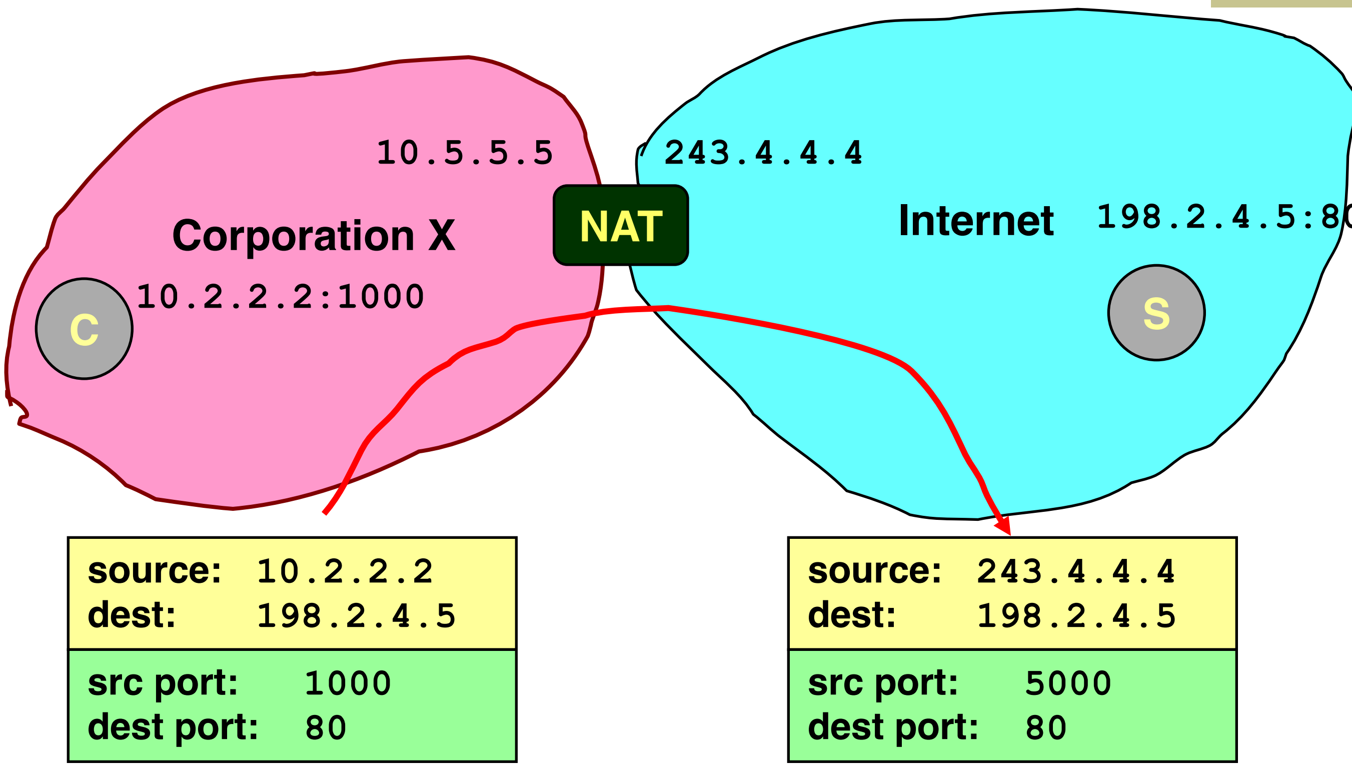


- Client 10.2.2.2 wants to connect to server 198.2.4.5:80
  - OS assigns ephemeral port (1000)
- Connection request intercepted by firewall
  - Maps client to port of firewall (5000)
  - Creates NAT table entry

Int Addr	Int Port	NAT Port
10.2.2.2	1000	5000

NAT: Client Request

C: Client
S: Server

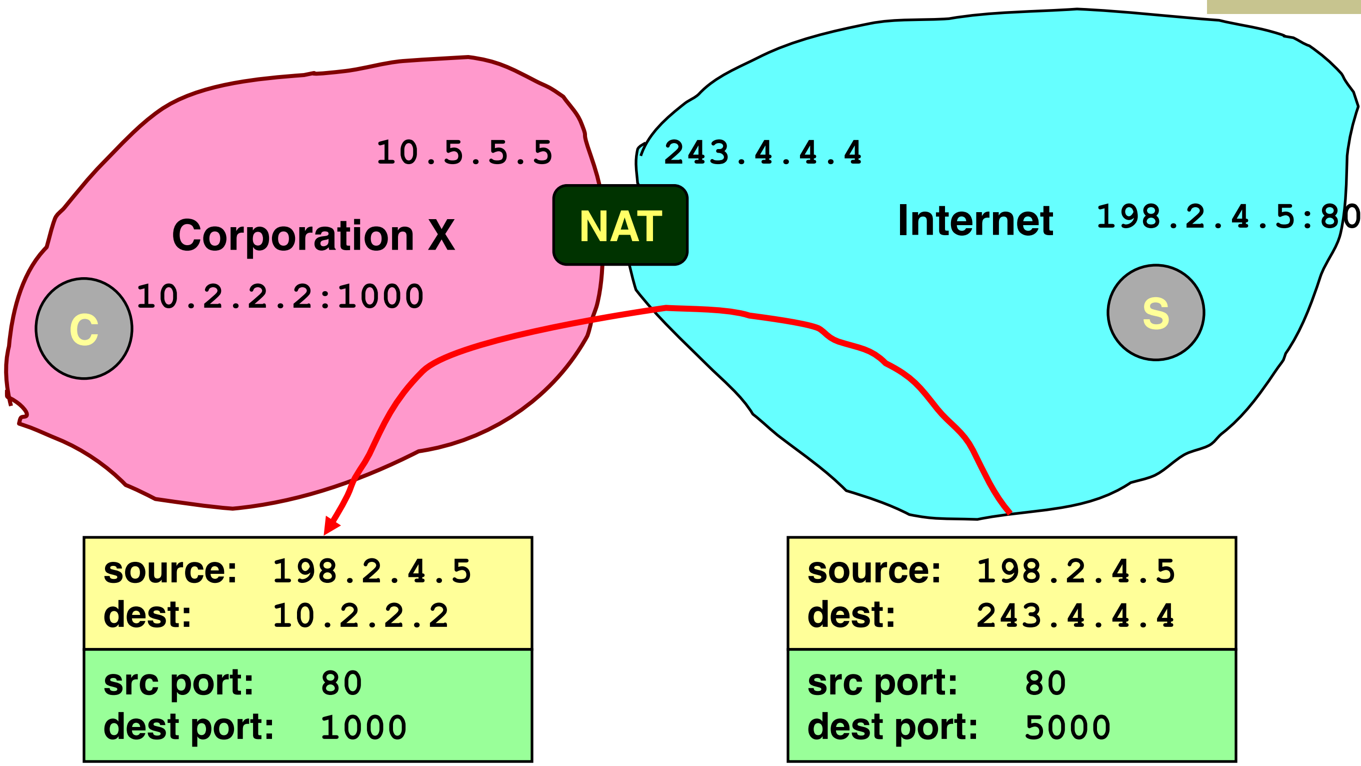


Int Addr	Int Port	NAT Port
10.2.2.2	1000	5000

- Firewall acts as proxy for client
  - Intercepts message from client and marks itself as sender

NAT: Server Response

C: Client
S: Server

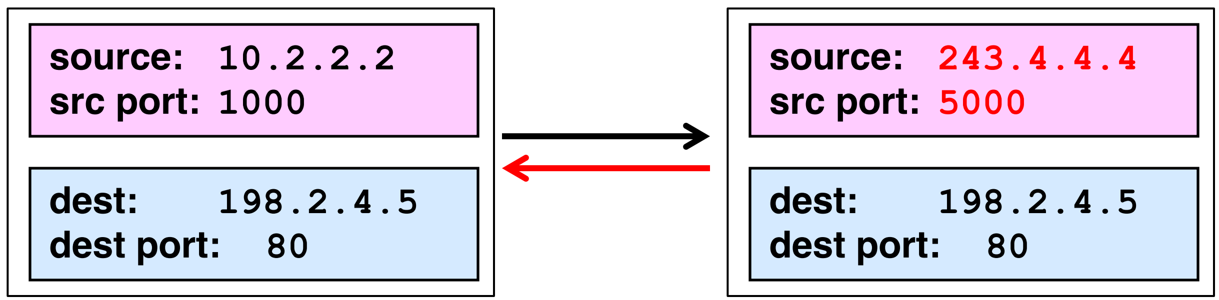


Int Addr	Int Port	NAT Port
10.2.2.2	1000	5000

- Firewall acts as proxy for client
  - Acts as destination for server messages
  - Relabels destination to local addresses

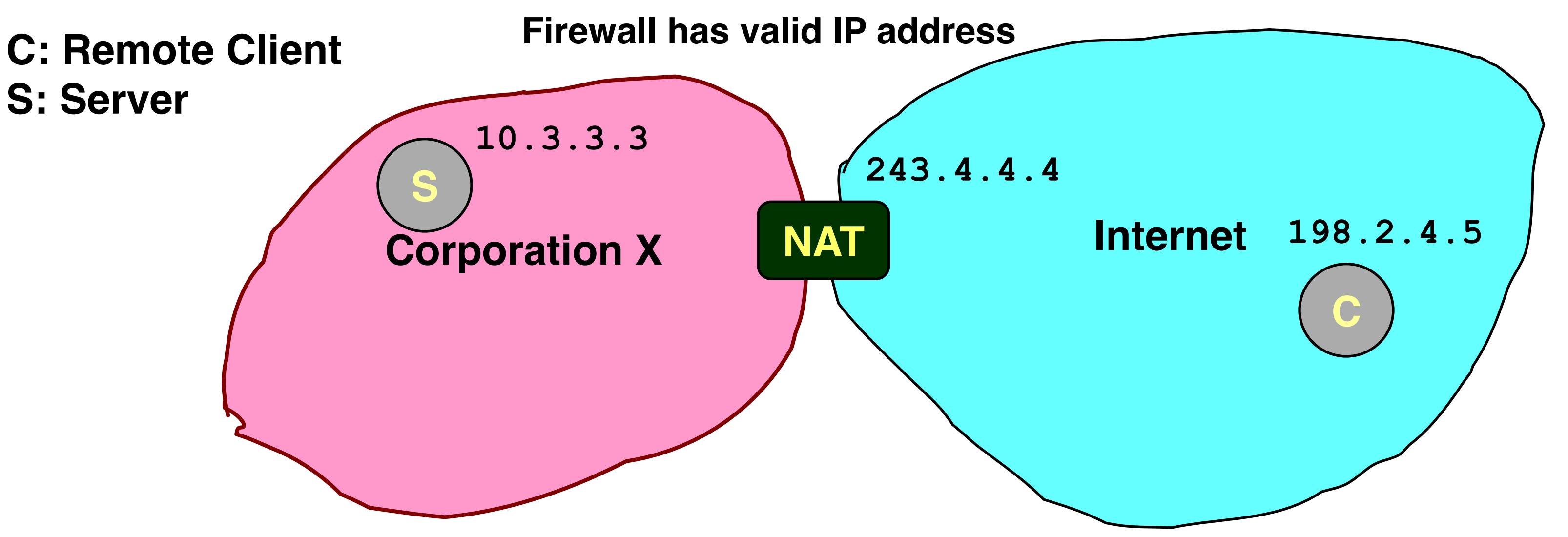
Client Request Mapping

Private network: Public Internet:



- NAT manages mapping between two four-tuples
- Mapping must be unique: one to one
- Must respect practical constraints
  - Cannot modify server IP address or port number
  - Client has limited number of IP addresses, often 1
  - Mapping client port numbers is important!
- Mapping must be consistent
  - The same for all packets in a communication session

NAT: Enabling Servers



- Use *port mapping* to make servers available

Int Addr	Int Port	NAT Port
10.3.3.3	80	80

  - Manually configure NAT table to include entry for well-known port
  - External users give address 243.4.4.4:80
  - Requests forwarded to server

Additional NAT Benefits

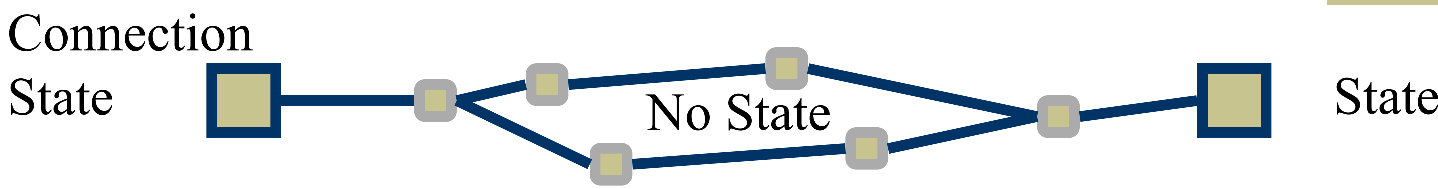
- They significantly reduce the need for public IP addresses
- NATs directly help with security
  - Hides IP addresses used in internal network
    - Easy to change ISP: only NAT box needs to have IP address
    - Fewer registered IP addresses required
  - Basic protection against remote attack
    - Does not expose internal structure to outside world
    - Can control what packets come in and out of system
    - Can reliably determine whether packet from inside or outside

- And NATs have many additional benefits
  - NAT boxes make home networking simple
  - Can be used to map between addresses from different address families, e.g, IPv4 and IPv6

NAT Challenges

- NAT has to be consistent during a session.
  - Mapping (hard state) must be maintained during the session
    - Recall Goal 1 of Internet: Continue despite loss of networks or gateways
  - Recycle the mapping after the end of the session
    - May be hard to detect
- NAT only works for certain applications.
  - Some applications (e.g. ftp) pass IP information in payload - oops
  - Need application level gateways to do a matching translation
- NATs are a problem for peer-peer applications
  - File sharing, multi-player games, …
  - Who is server?
  - Need to “punch” hole through NAT

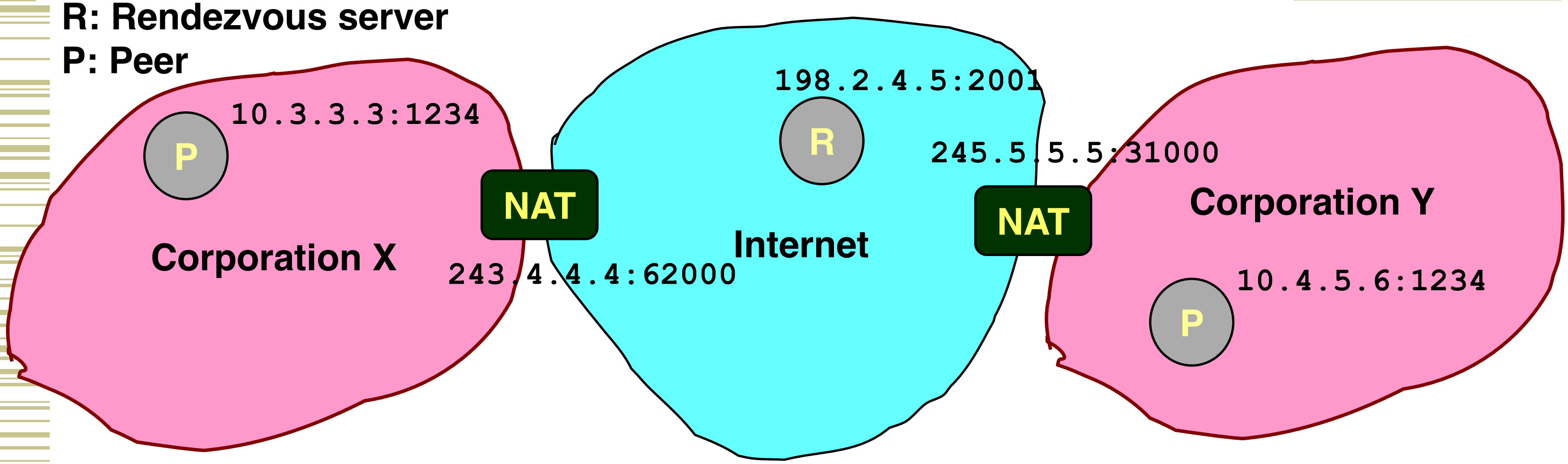
Principle: Fate Sharing



- "You can lose state information relevant to an entity's connections if and only if the entity itself is lost"
  - Example: OK to lose TCP state if either endpoint crashes
  - The TCP connection is no longer useful anyway!
- It is NOT okay to lose it if an unrelated entity goes down
  - Example: if an intermediate router reboots
- NATs violate this principle: if a NAT goes down, all communication session it supports are lost!
  - Unless you add redundancy and put state in persistent storage
- Bad news: many stateful "middleboxes" violate this rule
  - Firewalls, mobility services, … - more on this later
- Good news: today's hardware is very reliable

Many Options Exist for Peer-Peer

R: Rendezvous server
P: Peer



- NAT recognizes certain protocols and behaves as a application gateway
  - Used for standard protocols such as ftp
- Applications negotiate directly with NAT or firewall – need to be authorized
  - Multiple protocols dealing with different scenarios
- Punching holes in NAT: peers contact each other simultaneously using a known public (IP, port), e.g. used with rendezvous service
  - Use publicly accessible rendezvous service to exchange accessibility information
  - Assumes NATs do end-point independent mapping
- But remains painful!