More Network Addresses

Module Goals

At the conclusion of this module, students will be able to

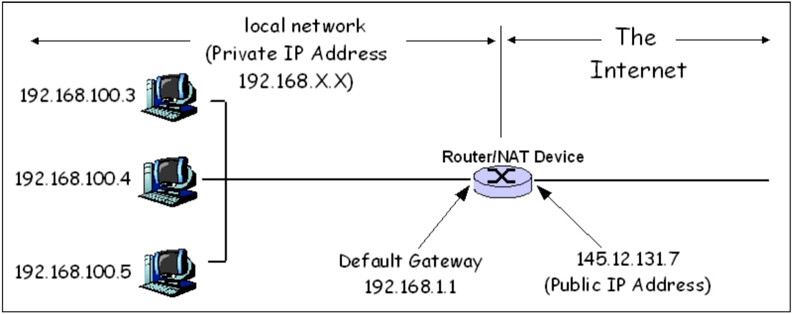
- understand the limitation of IPv4 addresses
- explain the behavior of Network Address Translation devices
- describe the differences between IPv4 and IPv6

The Perils of Only 32 Bits

- IPv7 addresses are starting to get scarce
- sometimes you might only get one IP address, but want to have many computers to connect to the Internet
- How can we address this?

  (**Network Address Translation**)

Network Address Translation (NAT)



How Does NAT Work?

- the router/NAT device has the IP address that was allocated
- other devices on that network are assigned generic IP address
- packets for all n devices on the network pass through the router
  (and they all have the same address!)
- a device on the “inside” of the network establishes a connection to an “outside” host
- the NAT device keeps track of the port and “inside” IP
- as incoming packets arrive, the NAT **rewrites** the destination address for the host that made the initial request

How Does NAT Work?

- What if an outside host tries to initiate the communication?
  - the NAT device won't have any information about the connection in its table!
  - the request gets thrown away
- alternatively, the router can be configured such that all connections coming in on some specific port should be forwarded to a specific host

The Future of NAT

- long-term?
  - with IPv6 on the way, NAT should be come less common
  - if there are more addresses to go around, why skimp and save?
- short-term?
  - with IPv6 still “coming soon”, NAT may become more common as ISPs try to delay or survive the transition
  - it’s difficult to put a precise number on what the adoption is for IPv6, but it’s well under 50%

IPv1? IPv2? Etc?

- IPv1: part of the TCPv1 standard
- IPv2: part of the TCPv2 standard
- IPv3: part of the TCPv3 standard
- IPv4: first time IP was separated from TCP
- IPv5: experimental protocol for streaming media
  (abandoned)

IPv6—The Next Generation

- motivations?
  - pretty much out of IPv4 addresses
  - reformat the header to facilitate processing, routing
  - add QoS header information
- datagram changes:
  - fixed-length 40 byte header
  - no more fragmentation by routers
    (fragmentation isn't gone—routers just don't do it anymore)

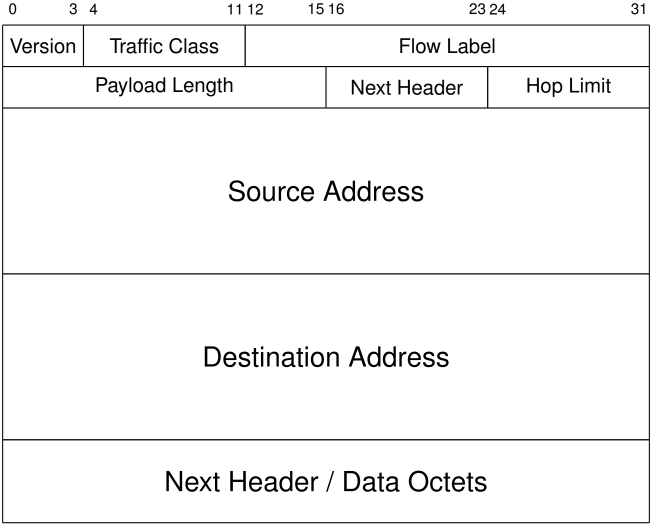
IPv6 Addresses

- classless addressing/routing (similar to CIDR)
- notation: `X:X:X:X:X:X:X:X` (16 bit hex entries)
  - contiguous 0s are compressed (`47CD::A456:0124`)
  - IPv6 compatible IPv4 addresses (`::128.42.1.87`)
- generally, the upper 64 bits are the network number and the lower 64 are the host number

Creating IPv6 Addresses

- one of the easiest ways to create an IPv6 address is using the link layer identifier
  (e.g., the Ethernet MAC address)
- the "link-local" or auto configured IPv6 address is `fe80::/10` plus the MAC address
- the IPv6 address space provides enough addresses for 1500 devices per square foot of the Earth's surface

IPv6 Header


0
3
4
11
12
15
16
23
24
31
Version
Traffic Class
Flow Label
Payload Length
Next Header
Hop Limit
Source Address
Destination Address
Next Header / Data Octets

IPv6 Header

- the 40-byte “base” header simplifies routing
- extension headers (fixed order, mostly fixed length)
  - fragmentation
  - source routing
  - authentication and security
  - other options

IPv6 Headers

- Version (4 bits): currently 6
- Traffic Class (8 bits): still not really used aside from ECN
- Flow Label (20 bits): a router can feel free to ignore this or treat it specially
- Payload Length (16 bits): not including the header
- Next Header (8 bits): either indicates the protocol running on top of IP or indicates the presence of more headers
- Hop Limit (8 bits): replaces TTL

IPv6 Extension Headers

- destination options, intended only for the ultimate destination
- hop-by-hop options, intended for the routers between the source and destination
  (routing)
- security options
- fragmentation

Minutia

- any link layer aiming to carry IPv6 data **must** be able to handle 1280 byte IP datagrams in the payload

  (remember, Ethernet's payload is 1500 bytes)

- IPv6 does describe a **jumbogram**, which is a 4 GiB (minus 1 byte) payload for link layers that support such a thing

Advanced Routing Capabilities

- in the **routing header**, a packet can contain a list of IPv6 addresses that the packet should traverse
  - in other words, picking its own rout
  - why? security, cost, throughput...
- sometimes you might want to pick a topological entity, so an **anycast** address is used

  (say you're on a mobile device and sending data to the nearest router for your carrier's network)
- an anycast address is actually a list of addresses that a "normal" routing protocol selects from

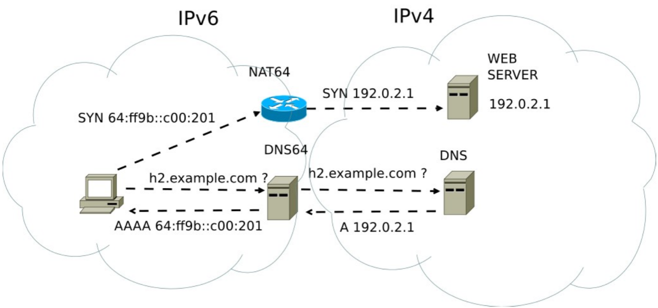
IPv4/IPv6 Coexistance

- option 1: dual stacks
  - the operating system can support both versions, so we'll do as much as we can over IPv6 when we can
  - when all else fails, fall back on IPv4
  - not really a long term solution—remember, the fundamental reason for the transition is address exhaustion

IPv4/IPv6 Coexistance

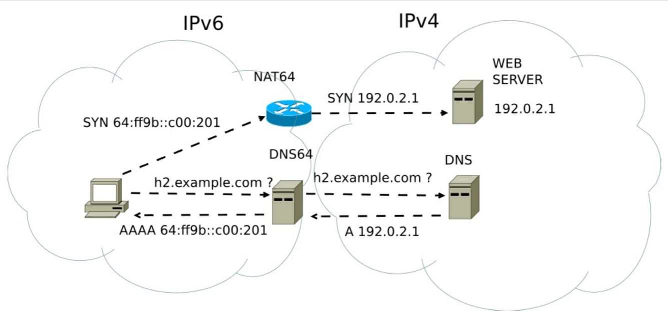
- option 2: translation
    - hardware could generate an IPv4 address, send to a special network device that transforms it into an IPv6 address (and vice versa on the way back in)
    - NAT64: multiple IPv6 machines share on IPv4 address and the NAT translates v6 addresses to v4
    - DNS64: a hostname that only resolves to an IPv4 address can be transformed into a special IPv6 address for the device

NAT64 and DNS64



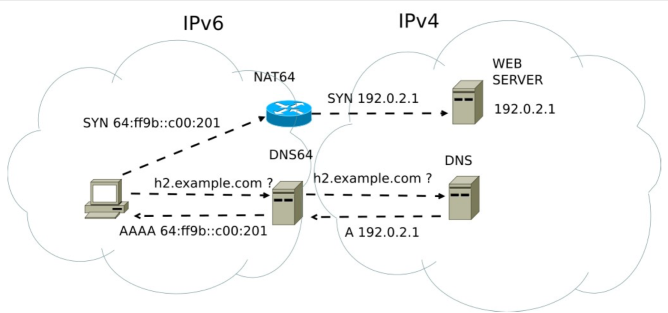
the host asks, “what’s the IP of `h2.example.com`?

NAT64 and DNS64



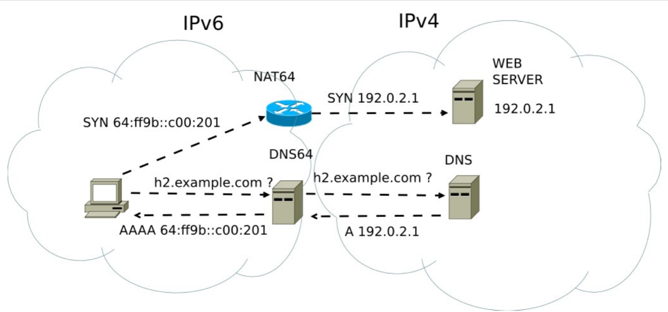
the DNS64 server asks, “what’s the v4 IP of `h2.example.com`?

NAT64 and DNS64



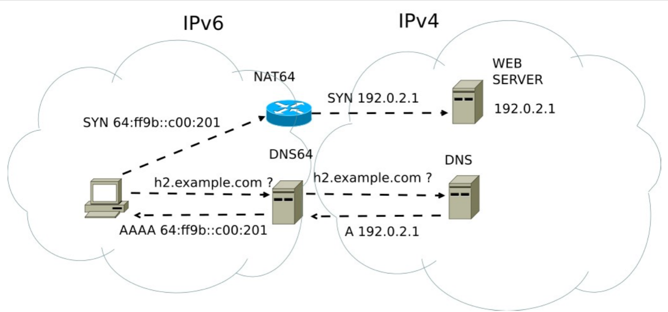
the DNS replies “192.0.2.1”

NAT64 and DNS64



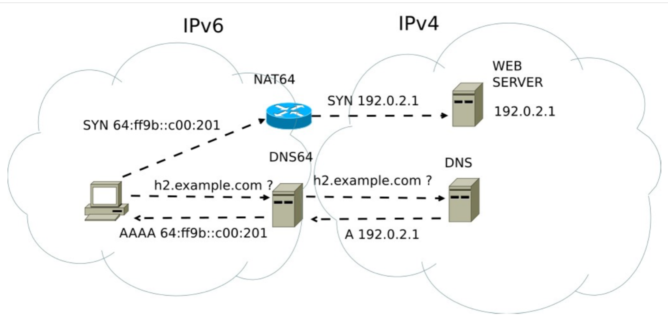
the DNS64 translates into a v6 address

NAT64 and DNS64



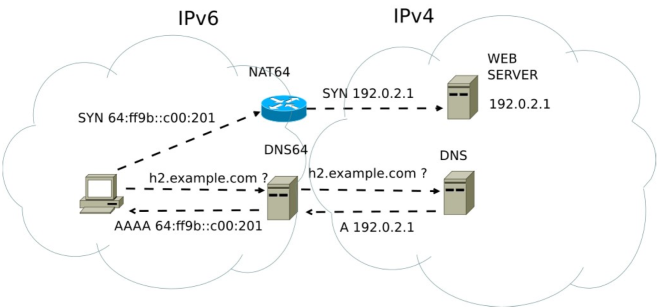
the host says “Host at v6 address: give me your website!”

NAT64 and DNS64



the NAT64 recognizes a special v6 address and translates to v4

NAT64 and DNS64

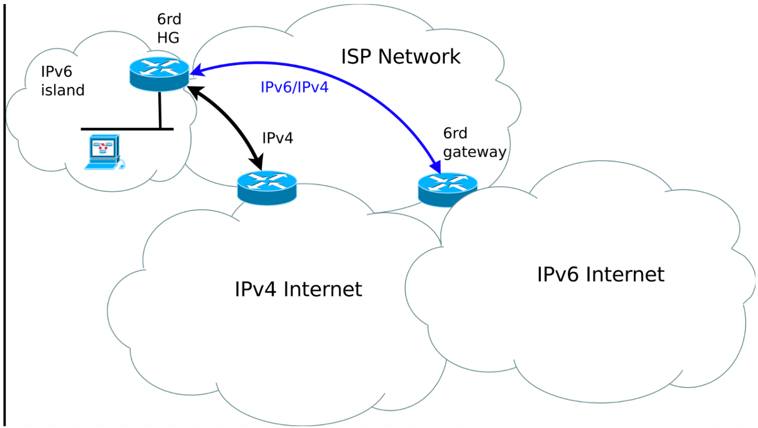


finally, the request arrives

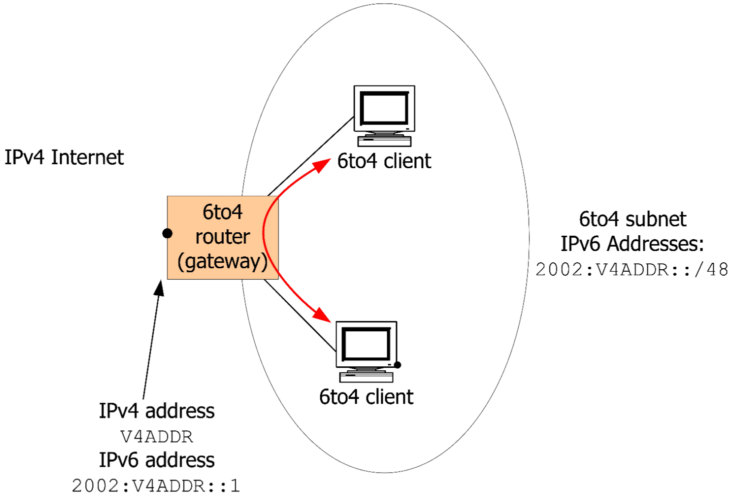
IPv4/IPv6 Coexistance

- option 3: tunneling
  - cram an IPv6 packet into an IPv4 packet into link-layer frame
  - this is often just referred to as **6in4**
    (an IPv**6** packet **in** an IPv**4** packet)
  - full implementations that do the routing are called **6to4** and include the 6in4 mechanism
  - example:
    - your IPv6 address is `2002:<IPv4 address>`
    - when a router sees such an address and cannot forward the packet using IPv6, it extracts the IPv4 address and forwards it
  - never meant to be a permanent solution; only a stopgap

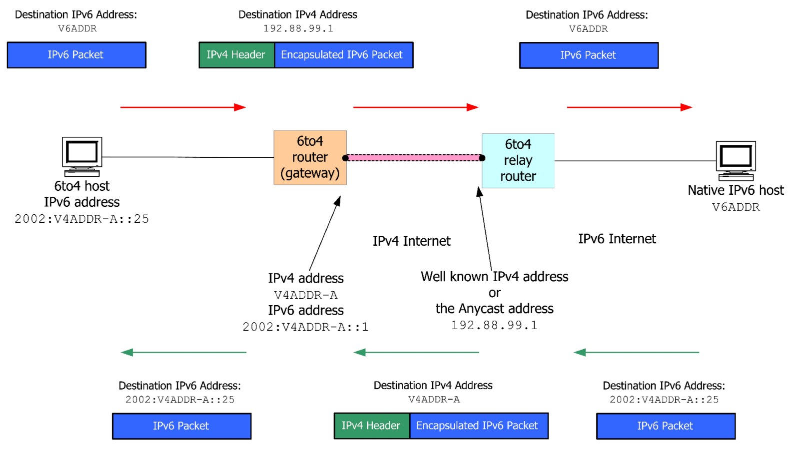
6to4 Big Picture



6to4 Scenario A



6to4 Scenario B



6rd (Rapid Deployment)

- 6to4 advertises common IPv4 and IPv6 prefixes to networks they are prepared to provide relay/translation services for
  - but... there is no guarantee that all native IPv6 hosts have a working route towards such a relay
  - therefore, a 6to4 host is not guaranteed to be reachable by all native IPv6 hosts, because 6rd views the IPv4 network as a link layer for IPv6
- 6rd makes each ISP use one of its own IPv6 prefixes

  (see RFC5569)
- pretty much everything else is the same, however, ISPs get more control over everything
  - customers are happy because of quality of service
  - ISPs are happy because they have control

Summary

- 32 bit addresses are a limiting factor for IPv4
- NAT helps translate addresses between a public facing IP address and private facing IP addresses on the sub-network
- IPv6 has plenty of addresses, but transitioning has been painfully slow