Internet Protocol

ComS 252 — Iowa State University

Barry Britt and Andrew Miner

What is IPv4?

- Version 4 of the Internet Protocol
- This is at the "Network Layer" in the OSI model
  - Deals with sending packets from one machine to another
  - Packets may go through several routers
- Let's see how it works

IP Packets

IP Packets consist of

1. The packet header
   - Like the envelope used to send postal mail
   - Contains information to send the data
   - Basically, a "`struct`" with several fields
   - Protocol specifies:
     - Order of the fields in the header
     - Number of bits for each field
2. Actual data
   - Interpreted based on the protocol used on top of IP
     - I.e., the protocol at the transport layer in the OSI model

What is in a packet header?

- Version number (should be 4)
  - First 4 bits of the packet

What is in a packet header?

- Version number (should be 4)
  - First 4 bits of the packet
- Total length (16 bits)
  - Total packet length, in bytes
  - Minimum is 20 bytes (20–byte header plus 0 bytes of data)
  - Maximum is

What is in a packet header?

- Version number (should be 4)
  - First 4 bits of the packet
- Total length (16 bits)
  - Total packet length, in bytes
  - Minimum is 20 bytes (20–byte header plus 0 bytes of data)
  - Maximum is $2^{16} - 1 = 65535$ bytes

What is in a packet header?

- Version number (should be 4)
  - First 4 bits of the packet
- Total length (16 bits)
  - Total packet length, in bytes
  - Minimum is 20 bytes (20–byte header plus 0 bytes of data)
  - Maximum is $2^{16} - 1 = 65535$ bytes
- Information for fragmenting packets
  - A packet may be broken into pieces and reassembled
  - Done so that packets will fit into frames

What is in a packet header?

- Version number (should be 4)
  - First 4 bits of the packet
- Total length (16 bits)
  - Total packet length, in bytes
  - Minimum is 20 bytes (20–byte header plus 0 bytes of data)
  - Maximum is $2^{16} - 1 = 65535$ bytes
- Information for fragmenting packets
  - A packet may be broken into pieces and reassembled
  - Done so that packets will fit into frames
- TTL: Time to Live (8 bits)
  - Specifies maximum lifetime of a packet
  - Keeps packets from going in circles indefinitely
  - Original spec was lifetime in seconds
  - In practice it is the maximum "hop count":
    - Each router decrements the TTL by one
    - Routers discard packets with TTL field of 0

What is in a packet header? (2)

- Protocol (8 bits)
  - What protocol is on top of this?
  - Can be ICMP, TCP, UDP, and others
  - ICMP: Internet Control Message Protocol
    - Used for query and error messages within IP

What is in a packet header? (2)

- Protocol (8 bits)
  - What protocol is on top of this?
  - Can be ICMP, TCP, UDP, and others
  - ICMP: Internet Control Message Protocol
    - Used for query and error messages within IP
- Source address (32 bits)
- Destination address (32 bits)

What is in a packet header? (2)

- Protocol (8 bits)
  - What protocol is on top of this?
  - Can be ICMP, TCP, UDP, and others
  - ICMP: Internet Control Message Protocol
    - Used for query and error messages within IP
- Source address (32 bits)
- Destination address (32 bits)
- Header Checksum
  - For detecting errors
- Other fields...
- IHL (4 bits): number of 32–bit words in the packet header
- Options: zero or more optional words (32 bits each)
  - Typically unused

Addresses in IPv4

- 32 bits long
  - Number of unique addresses in IPv4:

Addresses in IPv4

- 32 bits long
  - Number of unique addresses in IPv4: $2^{32} \approx 4.29$ billion
  - Number of devices "on the Internet" using IPv4:

Addresses in IPv4

- 32 bits long
  - Number of unique addresses in IPv4: $2^{32} \approx 4.29$ billion
  - Number of devices "on the Internet" using IPv4: $> 2^{32}$
    - Total number of devices on the Internet: ≈ 5 billion in 2010
  - How is this possible?

Addresses in IPv4

- 32 bits long
  - Number of unique addresses in IPv4: $2^{32} \approx 4.29$ billion
  - Number of devices "on the Internet" using IPv4: $> 2^{32}$
    - Total number of devices on the Internet: ≈ 5 billion in 2010
  - How is this possible?
    - There must be lots of devices using the same address
    - We will see why and how this works, later

Addresses in IPv4

- 32 bits long
  - Number of unique addresses in IPv4: $2^{32} \approx 4.29$ billion
  - Number of devices "on the Internet" using IPv4: $> 2^{32}$
    - Total number of devices on the Internet: ≈ 5 billion in 2010
  - How is this possible?
    - There must be lots of devices using the same address
    - We will see why and how this works, later
- Written as 4 integers separated by dots
  - Each integer is 8 bits
  - For example:

    `192.168.2.17` =
    `11000000.10101000.00000010.00010001`
- Governed by IANA (Internet Assigned Numbers Authority)
  - A department of ICANN
    (Internet Corporation for Assigned Names and Numbers)

Subnetworks

How are subnetworks handled in IPv4?

Subnetworks

How are subnetworks handled in IPv4?

- 1981 — 1993: Classful network
  - Introduced by RFC 791 — Internet Protocol
  - Splits the address space into classes
    - Class A, Class B, Class C, Class D, Class E
    - Classes D and E were not fully defined
- 1993 — today: Classless Inter–Domain Routing
  - Introduced by RFC 1518 and RFC 1519

Class A: Huge subnetworks

- First bit of 32–bit address is 0
- Next 7 bits: the subnetwork ID
  - At most $2^7 = 128$ Class A networks
- Final 24 bits: host ID within the subnetwork
  - At most $2^{24} = 16,777,216$ hosts in the subnet
- IP address has the form
  `0nnnnnnn.HHHHHHHH.HHHHHHHH.HHHHHHHH`

Example:

- Address `10.9.8.7` =
  `00001010.00001001.00001000.00000111`
- Class A subnetwork: `10.`
- Host within subnetwork: `9.8.7`

Class B: Large subnetworks

- First bit of 32–bit address is 1
  - Because it is not Class A
- Second bit of 32–bit address is 0
- Next 14 bits: the subnetwork ID
  - At most $2^{14} = 16,384$ Class B networks
- Final 16 bits: host ID within the subnetwork
  - At most $2^{16} = 65,536$ hosts in the subnet
- IP address has the form
  `10nnnnnn.nnnnnnnn.HHHHHHHH.HHHHHHHH`

Class C

- First bit of 32–bit address is 1
  - Because it is not Class A
- Second bit of 32–bit address is 1
  - Because it is not Class B
- Third bit of 32–bit address is 0
- Next 21 bits: the subnetwork ID
  - At most $2^{21} = 2,097,152$ Class C networks
- Final 8 bits: host ID within the subnetwork
  - At most $2^8 = 256$ hosts in the subnet
- IP address has the form

  `110nnnnn.nnnnnnnn.nnnnnnnn.HHHHHHHH`

Classes D and E

Class D

- First 4 bits are 1110
- IP address has the form
  `1110xxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx`

Class E

- First 4 bits are 1111
- IP address has the form
  `1111xxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx`

Why the switch away from Classful network in 1993?

Why the switch away from Classful network in 1993?

- Too “wasteful” of IP addresses
- E.g., suppose I want a subnet with 500 hosts
  - I need a Class B subnet with 65,536 addresses
- E.g., suppose I want a subnet with 70,000 hosts
  - I need a Class A subnet with 16 million addresses

Why the switch away from Classful network in 1993?

- Too "wasteful" of IP addresses
- E.g., suppose I want a subnet with 500 hosts
  - I need a Class B subnet with 65,536 addresses
- E.g., suppose I want a subnet with 70,000 hosts
  - I need a Class A subnet with 16 million addresses

- Classless Inter–Domain Routing is more flexible. . .

Classless Inter–Domain Routing (CIDR)

- Every IP address is partitioned as follows:
  - The first n bits are the subnet
  - The last $32 - n$ bits are the host name within the subnet
  - n may vary by subnet
- A subnet name may be specified by giving
  1. The number of prefix bits n for the subnet
  2. A complete IP address with last $32 - n$ bits equal to zero
- n can be specified after the address, e.g.:

  `192.168.2.0/24`
- n can be specified by giving the subnet mask, e.g.:

  `192.168.2.0/255.255.255.0`
- The suffix of all zeroes is the subnet name
- The suffix of all ones is the broadcast address

16–bit subnet prefix example

Subnet mask:	255.255.0.0	=	11111111.11111111.00000000.00000000
Network:	129.186.0.0	=	10000001.10111010.00000000.00000000
First usable:	129.186.0.1	=	10000001.10111010.00000000.00000001
Last usable:	129.186.255.254	=	10000001.10111010.11111111.11111110
Broadcast:	129.186.255.255	=	10000001.10111010.11111111.11111111

16–bit subnet prefix example

Subnet mask:	255.255.0.0	=	11111111.11111111.00000000.00000000
Network:	129.186.0.0	=	10000001.10111010.00000000.00000000
First usable:	129.186.0.1	=	10000001.10111010.00000000.00000001
Last usable:	129.186.255.254	=	10000001.10111010.11111111.11111110
Broadcast:	129.186.255.255	=	10000001.10111010.11111111.11111111

24–bit subnet prefix example

Subnet mask:	255.255.255.0	=	11111111.11111111.11111111.00000000
Network:	192.168.2.0	=	11000000.10101000.00000010.00000000
First usable:	192.168.2.1	=	11000000.10101000.00000010.00000001
Last usable:	192.168.2.254	=	11000000.10101000.00000010.11111110
Broadcast:	192.168.2.255	=	11000000.10101000.00000010.11111111

16–bit subnet prefix example

Subnet mask:	255.255.0.0	= 11111111.11111111.00000000.00000000
Network:	129.186.0.0	= 10000001.10111010.00000000.00000000
First usable:	129.186.0.1	= 10000001.10111010.00000000.00000001
Last usable:	129.186.255.254	= 10000001.10111010.11111111.11111110
Broadcast:	129.186.255.255	= 10000001.10111010.11111111.11111111

24–bit subnet prefix example

Subnet mask:	255.255.255.0	= 11111111.11111111.11111111.00000000
Network:	192.168.2.0	= 11000000.10101000.00000010.00000000
First usable:	192.168.2.1	= 11000000.10101000.00000010.00000001
Last usable:	192.168.2.254	= 11000000.10101000.00000010.11111110
Broadcast:	192.168.2.255	= 11000000.10101000.00000010.11111111

28–bit subnet prefix example

Subnet mask:	255.255.255.240	= 11111111.11111111.11111111.11110000
Network:	172.1.1.176	= 10101100.00000001.00000001.10110000
First usable:	172.1.1.177	= 10101100.00000001.00000001.10110001
Last usable:	172.1.1.190	= 10101100.00000001.00000001.10111110
Broadcast:	172.1.1.191	= 10101100.00000001.00000001.10111111

Relationship between Classful and Classless

Old Class A :

- Subnets `0.0.0.0/8` — `127.0.0.0/8`
- Subnet mask is `255.0.0.0`

Old Class B :

- Subnets `128.0.0.0/16` — `191.255.0.0/16`
- Subnet mask is `255.255.0.0`

Old Class C :

- Subnets `192.0.0.0/24` — `223.255.255.0/24`
- Subnet mask is `255.255.255.0`

Private Networks

- Suppose I want to set up an IP network at home
- Network sits behind a gateway plus router plus firewall
- No reason to connect to any of my machines from "outside"
- Do I need to contact IANA for IP addresses?

Private Networks

- Suppose I want to set up an IP network at home
- Network sits behind a gateway plus router plus firewall
- No reason to connect to any of my machines from "outside"
- Do I need to contact IANA for IP addresses? NO
- Can I use whatever IP addresses I want?

Private Networks

- Suppose I want to set up an IP network at home
- Network sits behind a gateway plus router plus firewall
- No reason to connect to any of my machines from "outside"
- Do I need to contact IANA for IP addresses? NO
- Can I use whatever IP addresses I want? YES but ...

Private Networks

- Suppose I want to set up an IP network at home
- Network sits behind a gateway plus router plus firewall
- No reason to connect to any of my machines from "outside"
- Do I need to contact IANA for IP addresses? NO
- Can I use whatever IP addresses I want? YES but ...
  - Suppose I choose an address that corresponds to `google.com`
  - When I try to send packets to `google.com` they will instead go to one of my machines

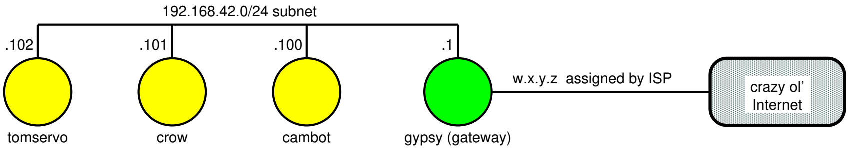
Private Networks

- Suppose I want to set up an IP network at home
- Network sits behind a gateway plus router plus firewall
- No reason to connect to any of my machines from "outside"
- Do I need to contact IANA for IP addresses? NO
- Can I use whatever IP addresses I want? YES but ...
  - Suppose I choose an address that corresponds to `google.com`
  - When I try to send packets to `google.com` they will instead go to one of my machines
- So, what addresses can I use?

Private Network Addresses

- RFC 1918 sets aside blocks of addresses for private use
- One "Class A" subnet: `10.`
  - Addresses `10.0.0.0` — `10.255.255.255`
- 16 "Class B" subnets: `172.16` — `172.31`
  - Addresses `172.16.0.0` — `172.31.255.255`
- 256 "Class C" subnets: `192.168.0` — `192.168.255`
  - Addresses `192.168.0.0` — `192.168.255.255`
- These addresses are guaranteed to remain private
- You may use these addresses without contacting IANA
  - That's why they are set aside
- Using CIDR, you may use whatever subnets you want
  - I can use `10.8.6.4/30` for a private subnet if I want ...

Example Home LAN

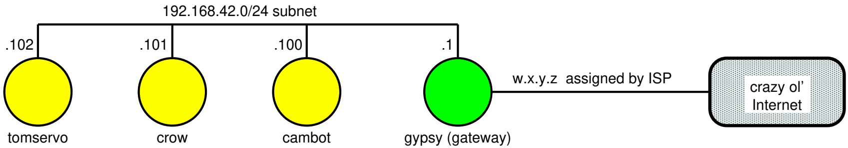


Suppose `gypsy` is a typical router

- Packets from one subnet are forwarded to the other

But what happens when I run a browser on `cambot`?

Example Home LAN



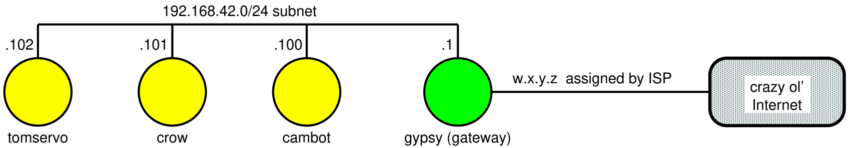
Suppose `gypsy` is a typical router

- Packets from one subnet are forwarded to the other

But what happens when I run a browser on `cambot`?

1. `cambot` (`192.168.42.100`) sends packets to `httpd` running on remote server (say, `129.186.23.166`)
2. How do packets get back to `192.168.42.100`?
   - Private IP address — cannot route packets to it

Example Home LAN



Suppose gypsy is a typical router

- Packets from one subnet are forwarded to the other

But what happens when I run a browser on cambot?

1. cambot (192.168.42.100) sends packets to httpd running on remote server (say, 129.186.23.166)
2. How do packets get back to 192.168.42.100?
   - Private IP address — cannot route packets to it

gypsy cannot be an "ordinary" router...

NAT: Network Address Translation

A.k.a. "IP Masquerading"

- We have a gateway router where
  - One side is a private network
  - One side is a "real" address
- Packets coming from the private network:
  - Rewrite the packet header
  - Pretend the packet came from the gateway's "real" address
  - Remember that we did this
- Packets coming from the other end:
  - If this is a response to a modified packet, then rewrite the header and send it to the right private IP address
  - Otherwise, no change

NAT: Network Address Translation

A.k.a. "IP Masquerading"

- We have a gateway router where
  - One side is a private network
  - One side is a "real" address
- Packets coming from the private network:
  - Rewrite the packet header
  - Pretend the packet came from the gateway's "real" address
  - Remember that we did this
- Packets coming from the other end:
  - If this is a response to a modified packet, then rewrite the header and send it to the right private IP address
  - Otherwise, no change
  - Ok, wait, how does that work?

NAT: Network Address Translation

A.k.a. "IP Masquerading"

- We have a gateway router where
  - One side is a private network
  - One side is a "real" address
- Packets coming from the private network:
  - Rewrite the packet header
  - Pretend the packet came from the gateway's "real" address
  - Remember that we did this
- Packets coming from the other end:
  - If this is a response to a modified packet, then rewrite the header and send it to the right private IP address
  - Otherwise, no change
  - Ok, wait, how does that work? Magic
    - Uses TCP or UDP information also
    - And changes port numbers around

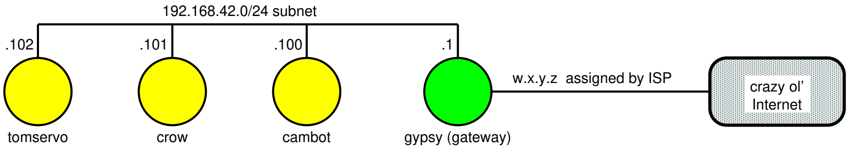
NAT: Network Address Translation

A.k.a. "IP Masquerading"

- We have a gateway router where
  - One side is a private network
  - One side is a "real" address
- Packets coming from the private network:
  - Rewrite the packet header
  - Pretend the packet came from the gateway's "real" address
  - Remember that we did this
- Packets coming from the other end:
  - If this is a response to a modified packet, then rewrite the header and send it to the right private IP address
  - Otherwise, no change
  - Ok, wait, how does that work? Magic
    - Uses TCP or UDP information also
    - And changes port numbers around

We will see how to set this up . . . later

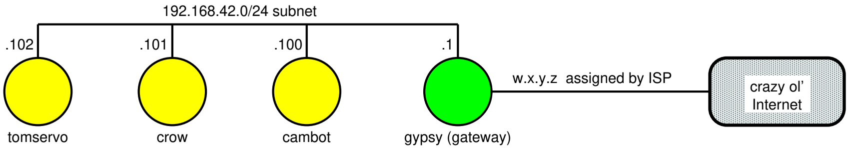
Example Home LAN (2)



Suppose `gypsy` is a typical home router (using NAT)

1. `cambot` sends packets to `129.186.23.166`
2. `gypsy` rewrites source address as `w.x.y.z` and forwards to ISP
3. ISP and Internet routers get packets to `129.186.23.166`
4. Return packets get to `w.x.y.z` (that's ISP's job)
5. `gypsy` rewrites destination address and forwards to `cambot`

Example Home LAN (2)



Suppose `gypsy` is a typical home router (using NAT)

1. `cambot` sends packets to `129.186.23.166`
2. `gypsy` rewrites source address as `w.x.y.z` and forwards to ISP
3. ISP and Internet routers get packets to `129.186.23.166`
4. Return packets get to `w.x.y.z` (that's ISP's job)
5. `gypsy` rewrites destination address and forwards to `cambot`

Fun fact: `w.x.y.z` might also be a private address

Benefits of NAT

- Make private networks useful
  - Can connect to "public" Internet
- Has significantly slowed consumption of IP addresses
  - Unique addresses necessary only for publicly–visible things
- Give us security and natural firewalling
  - Would–be crackers cannot attack a private IP address

Benefits of NAT

- Make private networks useful
  - Can connect to “public” Internet
- Has significantly slowed consumption of IP addresses
  - Unique addresses necessary only for publicly–visible things
- Give us security and natural firewalling
  - Would–be crackers cannot attack a private IP address

- Most of Iowa State’s machines use private addresses
  - Including homework server
  - That’s why you need to go through the VPN from off campus

Other reserved addresses

The following blocks of addresses are also reserved:

- 0.0.0.0/8 : used for broadcast messages on “current” network
- 127.0.0.0/8 : used for “loopback” addresses
  - Address `127.0.0.1` refers to the local host itself
- 255.255.255.255/32 : “limited broadcast” destination address

IPv5

Internet Stream Protocol (ST and ST2)

- First specified in 1979
- Intended as connection–oriented complement to IPv4
- Second version (ST2) specified in 1990
- Final version is ST2+
  - RFC 1819
- ST2 uses packets with "5" in the version field
  - RFC 1700
- Never really caught on
- Technically, was never known as "IPv5"

IPv6: The new standard for Internet Protocol

- Not widely implemented yet
  - October 2011 estimate:

    3% of domain names have IPv6 support
    12% of networks have IPv6 support
- Described in RFC 2460 (December 1998)
- Main motivation: larger address space
  - Not the only change, but an important one

IPv6: The new standard for Internet Protocol

- Not widely implemented yet
  - October 2011 estimate:

    3% of domain names have IPv6 support

    12% of networks have IPv6 support
- Described in RFC 2460 (December 1998)
- Main motivation: larger address space
  - Not the only change, but an important one
- Why are we still using IPv4? Why so slow to adopt IPv6?

IPv6: The new standard for Internet Protocol

- Not widely implemented yet
  - October 2011 estimate:

    3% of domain names have IPv6 support
    12% of networks have IPv6 support
- Described in RFC 2460 (December 1998)
- Main motivation: larger address space
  - Not the only change, but an important one
- Why are we still using IPv4? Why so slow to adopt IPv6?
  - Because it still works
  - CIDR and NAT have greatly extended the life of IPv4

IPv6: The new standard for Internet Protocol

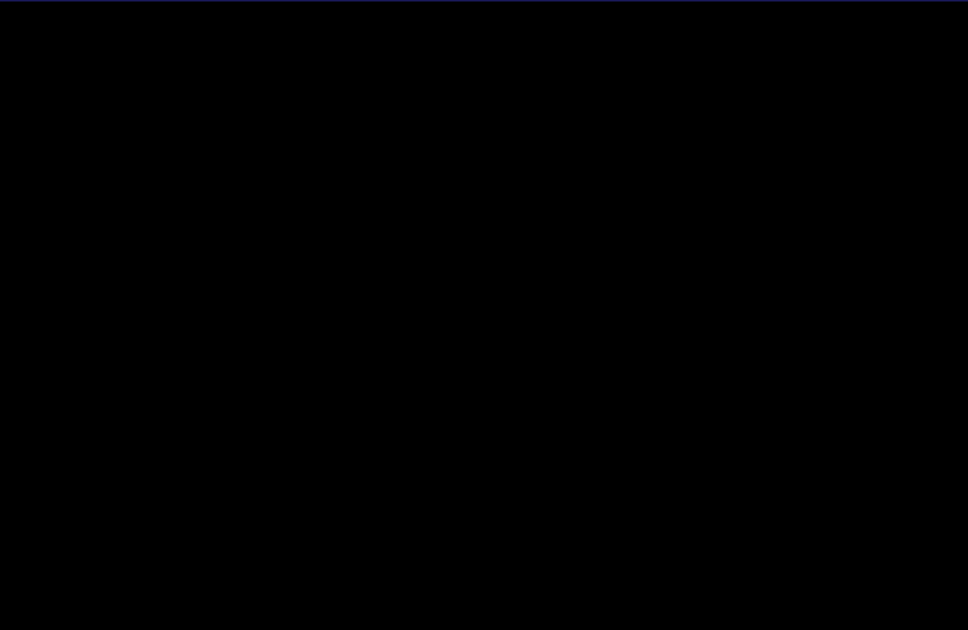
- Not widely implemented yet
  - October 2011 estimate:

    3% of domain names have IPv6 support

    12% of networks have IPv6 support
- Described in RFC 2460 (December 1998)
- Main motivation: larger address space
  - Not the only change, but an important one
- Why are we still using IPv4? Why so slow to adopt IPv6?
  - Because it still works
  - CIDR and NAT have greatly extended the life of IPv4
  - Need compatible hardware
  - Need to update applications (think: Y2K problem)
  - Routing is different
  - Not easy to "mix" IPv6 and IPv4
  - But the end is near for IPv4...

IPv4 address exhaustion

- IANA ran out of IPv4 addresses on 31 January 2011
- Asia's RIR (Regional Internet Registry) ran out 15 April 2011
- Europe's RIR ran out 14 September 2012



Vint Cerf discusses IPv6 in 2014. `https://www.youtube.com/watch?v=17GtmwyvmWE`

What is in an IPv6 packet header?

- They are different from IPv4
  - Larger (40 bytes vs. 20 or more bytes)
  - Simpler (8 fields vs. 13 or more fields)
- Version number (should be 6)
  - Still first 4 bits of the packet, so there is some hope...
- Payload length: 16 bits
- Next Header: 8 bits (replaces "Protocol" field)
  - Usually specifies the transport layer protocol
- Hop limit: 8 bits (replaces "Time to Live" field)
- Source address: 128 bits
- Destination address: 128 bits
- New in IPv6: "Flow label" field
- Missing in IPv6: checksum and fragmentation fields
  - IPv6 routers never fragment IPv6 packets

Addresses in IPv6

- 128 bits long
  - Number of unique addresses:

Addresses in IPv6

- 128 bits long
  - Number of unique addresses: $2^{128} \approx$

Addresses in IPv6

- 128 bits long
  - Number of unique addresses: $2^{128} \approx 3.40 \times 10^{38}$
  - Recall: IPv4 has $2^{32} \approx 4.29 \times 10^{9}$ unique addresses

Addresses in IPv6

- 128 bits long
  - Number of unique addresses: $2^{128} \approx 3.40 \times 10^{38}$
  - Recall: IPv4 has $2^{32} \approx 4.29 \times 10^{9}$ unique addresses
- Written as
  - 8 groups of 16–bit integers (in hex) with ":" separators
  - Can remove leading zeroes from hex values
  - Can replace one "block of zeroes" with empty string
  - E.g., the following refer to the same address:
    - `2001:0db8:0000:0000:0000:ff00:0042:8329`
    - `2001:db8:0:0:0:ff00:42:8329`
    - `2001:db8::ff00:42:8329`

Addresses in IPv6

- 128 bits long
  - Number of unique addresses: $2^{128} \approx 3.40 \times 10^{38}$
  - Recall: IPv4 has $2^{32} \approx 4.29 \times 10^{9}$ unique addresses
- Written as
  - 8 groups of 16–bit integers (in hex) with ":" separators
  - Can remove leading zeroes from hex values
  - Can replace one "block of zeroes" with empty string
  - E.g., the following refer to the same address:
    - `2001:0db8:0000:0000:0000:ff00:0042:8329`
    - `2001:db8:0:0:0:ff00:42:8329`
    - `2001:db8::ff00:42:8329`
- Special case: IPv4–mapped addresses
  - First 80 bits are 0
  - Next 16 bits are 1
  - Last 32 bits are the IPv4 address
  - Can write the IPv4 address in the usual dotted notation
    - E.g.: `::ffff:192.168.1.1`

Benefits of IPv6

- Larger address space (obviously)
- Simpler packet headers
  - Easier for routers to process
  - That translates to: speed
- NAT becomes unnecessary
  - Can still have private networks, but:
  - Private network machines can have a globally unique address
  - This is what a huge address space buys you
  - But there is some controversy about this . . .

Old remote login utilities

- `rcp`: remote file copy
  - Copy a file from a remote host to the local one or from the local host to a remote one
- `rsh`: run a shell, remotely
- `rlogin`: remote login
- `telnet`: run a shell, remotely
  - A little smarter than `rsh` and `rlogin`

Old remote login utilities

- `rcp`: remote file copy
  - Copy a file from a remote host to the local one or from the local host to a remote one
- `rsh`: run a shell, remotely
- `rlogin`: remote login
- `telnet`: run a shell, remotely
  - A little smarter than `rsh` and `rlogin`

- NEVER USE THESE

Why is it bad to use `rcp`, `rsh`, `rlogin`, `telnet`?

Why is it bad to use `rcp`, `rsh`, `rlogin`, `telnet`?

- Usernames and passwords are transmitted in "plain text"
  - I.e., not encrypted in any way
  - Anyone watching the network traffic can see
    - But that's not easy do do, right?
- Allowed `.rlogin` configuration files in user's home directory
  - These can be misused and abused . . .
  - . . . potentially allowing anyone to login without a password
- `rlogin` protocol has no way to authenticate the client
  - Corrupt clients can fake it and gain access

Why is it bad to use `rcp`, `rsh`, `rlogin`, `telnet`?

- Usernames and passwords are transmitted in “plain text”
  - I.e., not encrypted in any way
  - Anyone watching the network traffic can see
    - But that’s not easy do do, right?
- Allowed `.rlogin` configuration files in user’s home directory
  - These can be misused and abused ...
  - ... potentially allowing anyone to login without a password
- `rlogin` protocol has no way to authenticate the client
  - Corrupt clients can fake it and gain access

- `scp` and `ssh` are much better choices

How hard is it to watch network traffic?

- Incredibly easy if you have physical access to the network
- Use a packet analyzer
  - A.k.a.: *network analyzer*, *packet sniffer*, *protocol analyzer*
- These intercept and log network frames
- How to intercept frames?
  - Ethernet cards can run in promiscuous mode
    - Copies all frames, not just ones to the card
    - In shared–media network, we can watch all traffic
  - Network tap
    - Hardware device that copies network traffic
    - . . . sometimes on just one link
  - Wireless ethernet: just listen
  - *None of these* can be detected
- These all have legitimate uses for network debugging

Secure replacements

`scp`

- Securely copy files between remote and local hosts

`ssh`

- Securely run a shell on a remote host
- You can do (secure) X forwarding
  - Remote X client connects to local X server
  - Which means the graphics are "forwarded" through `ssh`

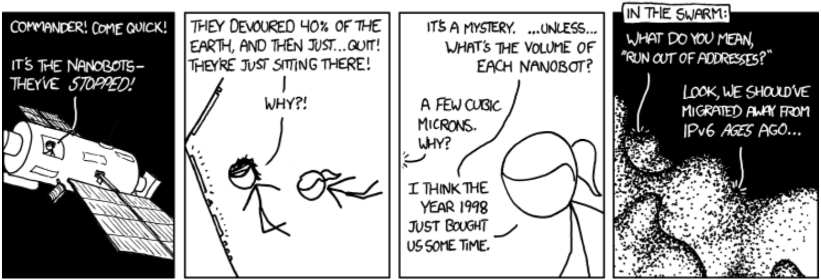
Both of these run on top of SSL

- Secure Sockets Layer
- Cryptographic protocols on top of IP

Generating an `ssh` key

1. Run `ssh-keygen` to generate a key
   - Can associate a passphrase with the key
   - Key will have two parts
     (i) A private part (keep this secret)
     (ii) A public part
   - How it works is quite technical, but:
     - The public key is used to encrypt messages
     - Only the private key can decrypt the message
     - You cannot determine the private key from its public one
2. Add the public key to the appropriate file on the remote host
   - Usually `~/.ssh/authorized_keys`
3. You should be able to `ssh` into the remote host using the key
   - You will be prompted for the key's passphrase
   - You will not be prompted for the password on the remote host

An appropriate xkcd comic: `http://xkcd.com/865`



End of lecture