IPv6

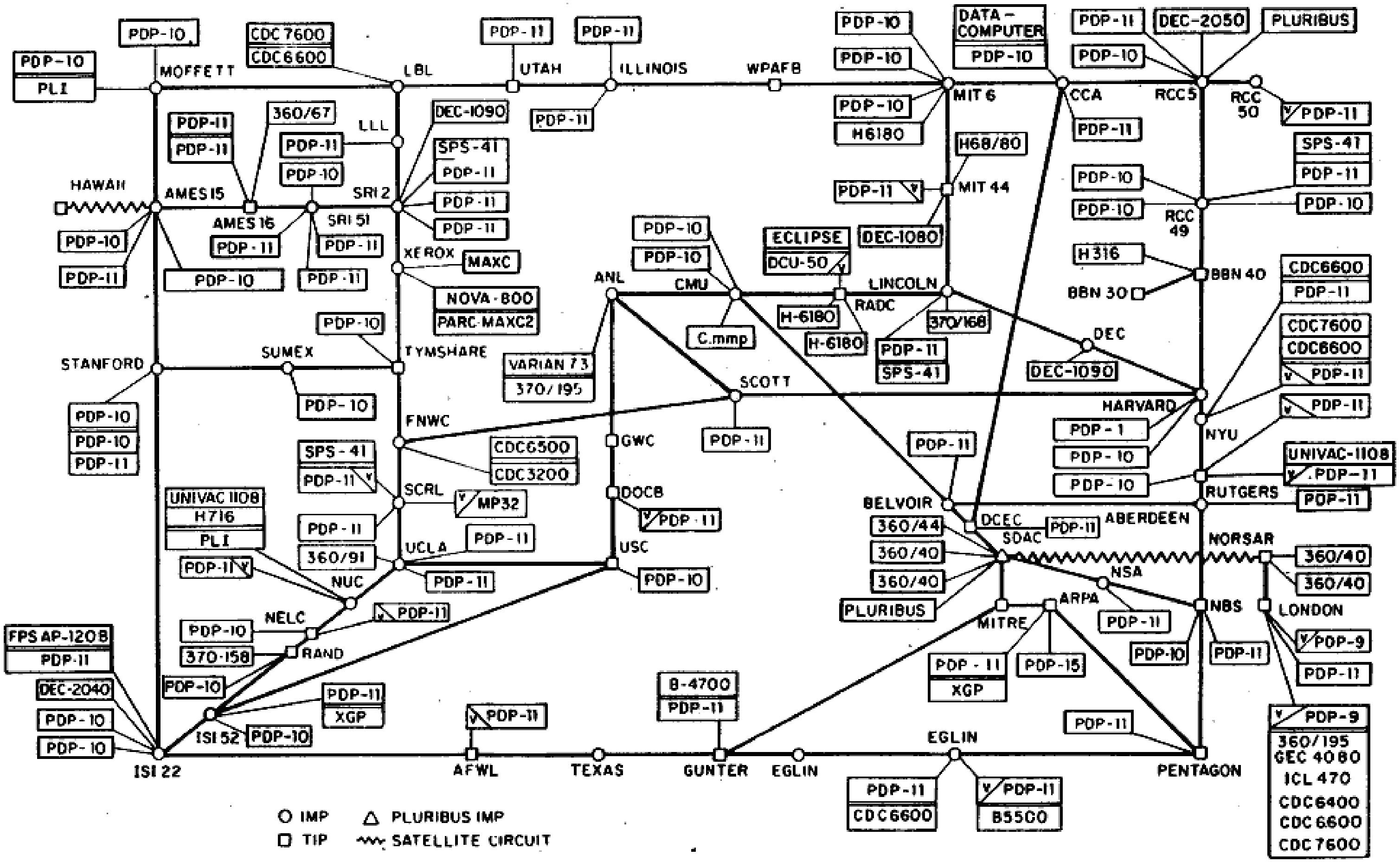
Network Programming

Motivation

- Internet grew too fast
- Recall from our first lecture…
- **"I think there is a world market for maybe five computers.” – TJ Watson, IBM president, 1943**

ARPANET March, 1977

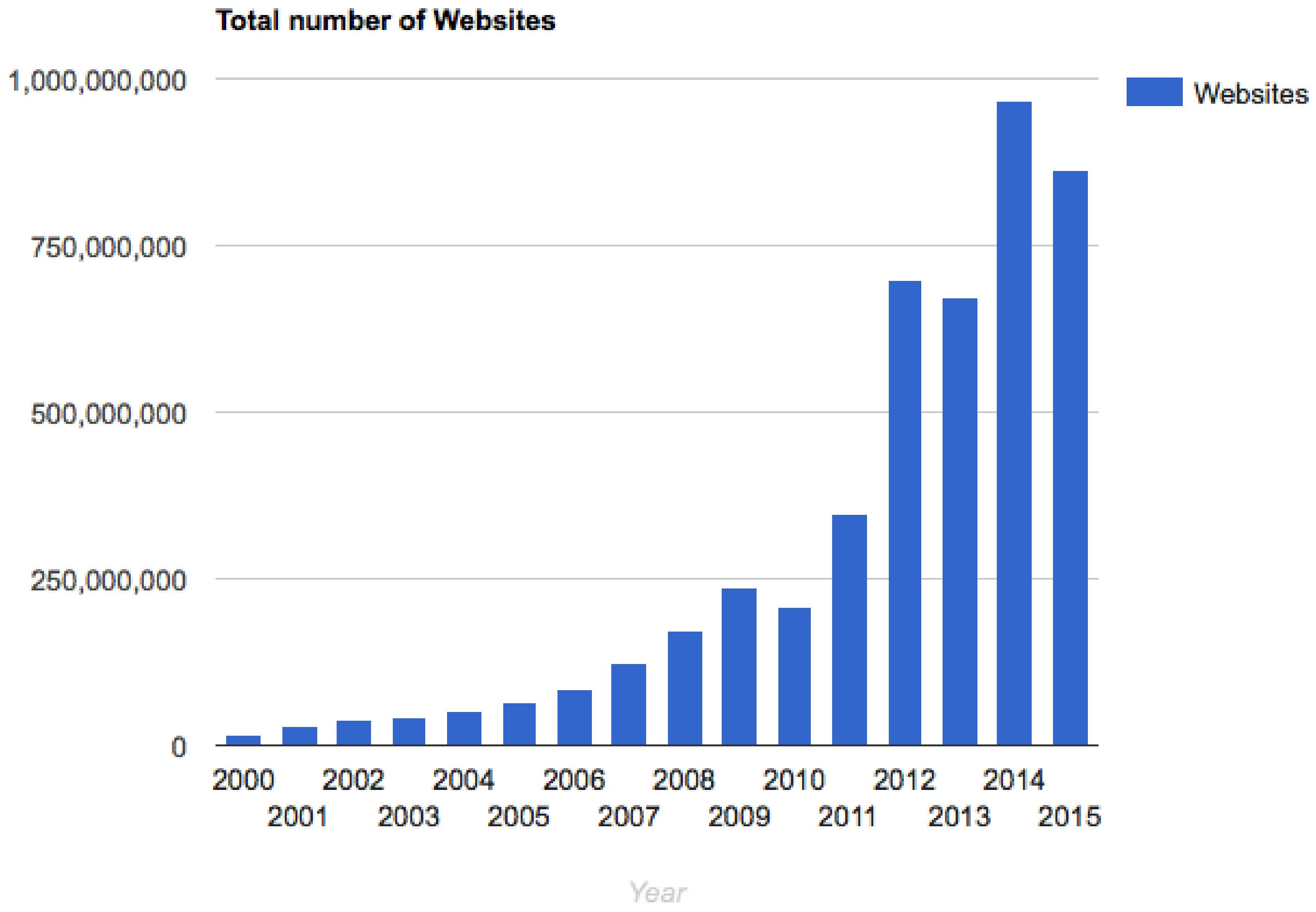
ARPANET LOGICAL MAP, MARCH 1977



(PLEASE NOTE THAT WHILE THIS MAP SHOWS THE HOST POPULATION OF THE NETWORK ACCORDING TO THE BEST INFORMATION OBTAINABLE, NO CLAIM CAN BE MADE FOR ITS ACCURACY)

NAMES SHOWN ARE IMP NAMES, NOT (NECESSARILY) HOST NAMES

Well That Escalated Quickly



- http://www.internetlivestats.com/total-number-of-websites/

IPv4 Problems

- Not enough addresses
    - Only 2^{32} or ~4.3 billion addresses
    - NAT can help…
- While we're at it, fix some other IPv4 problems
    - Fragmentation happens at routers
    - Lots of repetition in IPv4 stack
        - Layer 2 checksum, IPv4 header checksum, transport layer (TCP/UDP) checksum…

Network Address Translation

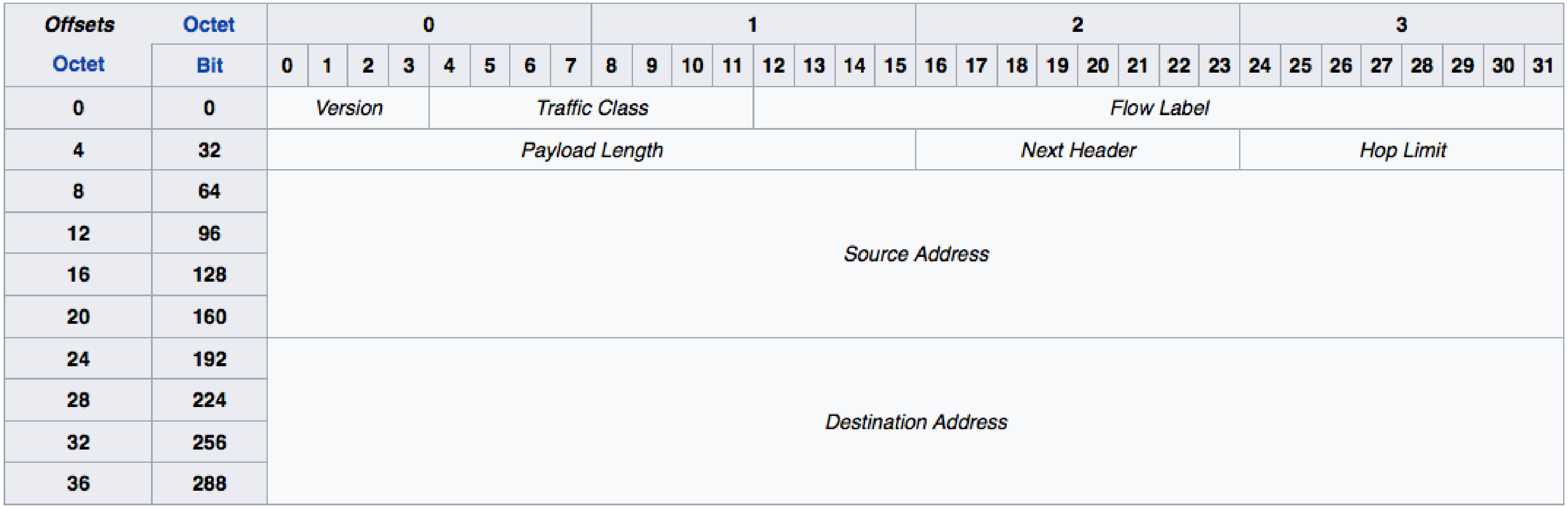
- A local network with N hosts doesn't need N public IP addresses
- We can use a private network such as 192.168.0/24 for *internal communication*
- Router with public 129.114.74.6 will do NAT - invisible to local and remote hosts
  - 192.168.0.1 makes a request to 128.113.0.2 on port 80, source port 9854
  - Router sees request, modifies headers to say source IP 128.113.0.2 source port 13590
  - When a reply is received, do lookup and change headers to say destination IP 192.168.0.1, destination port 9854

IPv6

- Benefits:
  - 128 bit address space
    - 667 sextillion (10^{21}) per square meter on Earth
    - We’re probably taking this intergalactic
  - DNS still works!
    - AAAA records instead of A records
  - IPv6 can tunnel via IPv4 networks
    - Endpoints must pack/extract IPv6 packets

IPv6 Header

Fixed header format



Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	*Version*				*Traffic Class*								*Flow Label*																			
4	32	*Payload Length*																*Next Header*								*Hop Limit*							
8	64	*Source Address*																															
12	96																																
16	128																																
20	160																																
24	192	*Destination Address*																															
28	224																																
32	256																																
36	288																																

Comparison

IPv4	IPv6
Address are 32 bits	Addresses are 128 bits
IPsec optional	IPsec required
No QoS flow control	QoS flow control provided by flow label
Fragmentation at routers	No fragmentation at routers
Checksum in header	No checksum
Header includes options	Options in IPv6 extension headers
ARP uses broadcast resolution	ARP uses multicast resolution via Neighbor solicitation messages

Addressing

- Example:
  2604:6000:111a:807e:14da:fa9e:3069:a300
  - See RFC 4291 §2.2 for full rules

- Unicast, anycast (set of nodes), multicast; no broadcast

- Unicast/anycast: broken into 64-bit network prefix, 64-bit interface identifier

IPv6 Security

- Must support IPsec
- Authentication and encryption separate but both are provided
  - MITM attacks more difficult for attacker
  - But IPsec usage is optional
- As with any major transition, bugs will reveal themselves

Happy Eyeballs

- Dual IPv4/IPv6 stacks
  - Check both but prefer IPv6 if available
  - RFC 6555

- Mac OS X El Capitan (and newer) gives IPv6 25 ms headstart
  - Adds a little delay but pushes IPv6 adoption
  - If you host a webserver, adding IPv6 could improve responsiveness (slightly)

When to plan the migration

- Yesterday
  - There is no “if” to this migration but rather “when”
- “World IPv6 Launch Day” happened June 6th, 2012
  - So your assignment is already late before you’ve even gotten it
- https://tools.ietf.org/html/rfc4038

IPv6

- " As of 2014, IPv4 still carried more than 99% of worldwide Internet traffic.[73][74] The Internet exchange in Amsterdam and Seattle are the only large exchanges that publicly show IPv6 traffic statistics, which as of October 2018 are tracking at about 2.9% and 7.7%, growing at about 1.9% and -2.6% per year, respectively.[75][76] As of 13 October 2018, the percentage of users reaching Google services with IPv6 reached 25.0% for the first time, growing at about 4.2% per year. This growth is down from 7.2% per year between July 2016 and July 2017.[77] As of April 2018 about 27% of Alexa Top 1000 web servers support IPv6.[78] "

  - https://en.wikipedia.org/wiki/IPv6#IPv6_readiness

Check your IPv6 support

- Linux? Mac?
  - ifconfig can tell you
- Windows 10?
  - http://technology.pitt.edu/support/enabling-and-disabling-ipv6-on-pcs
  - Enabled by default
- http://test-ipv6.com

IPv6 Programming

Network Programming

Finding Hosts

- **`gethostbyname()`** is tied to IPv4
- So is **`gethostbyaddr()`**
- We ~~might~~ *will* need to overhaul the API

getaddrinfo()

-
  ```
  int
  getaddrinfo(
    const char *node,
    const char *service,
    const struct addrinfo *hints,
    struct addrinfo **res);
  ```

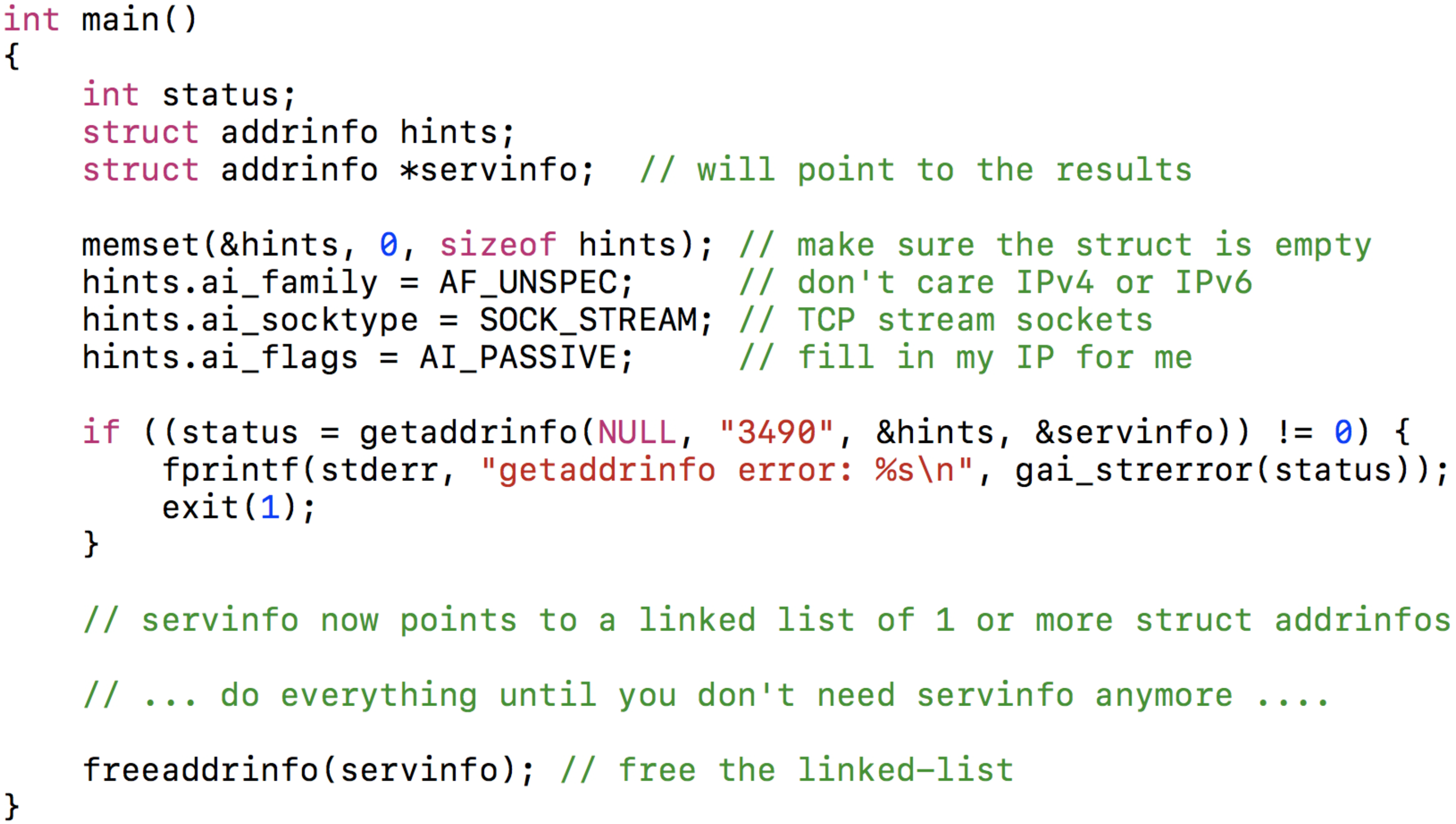
- See the man page for more details (especially on the hints!)

getnameinfo()

-
  ```
  int
  getnameinfo(
     const struct sockaddr* sa,
     socklen_t salen, char* host,
     size_t hostlen, char* serv,
     size_t servlen, int flags);
  ```

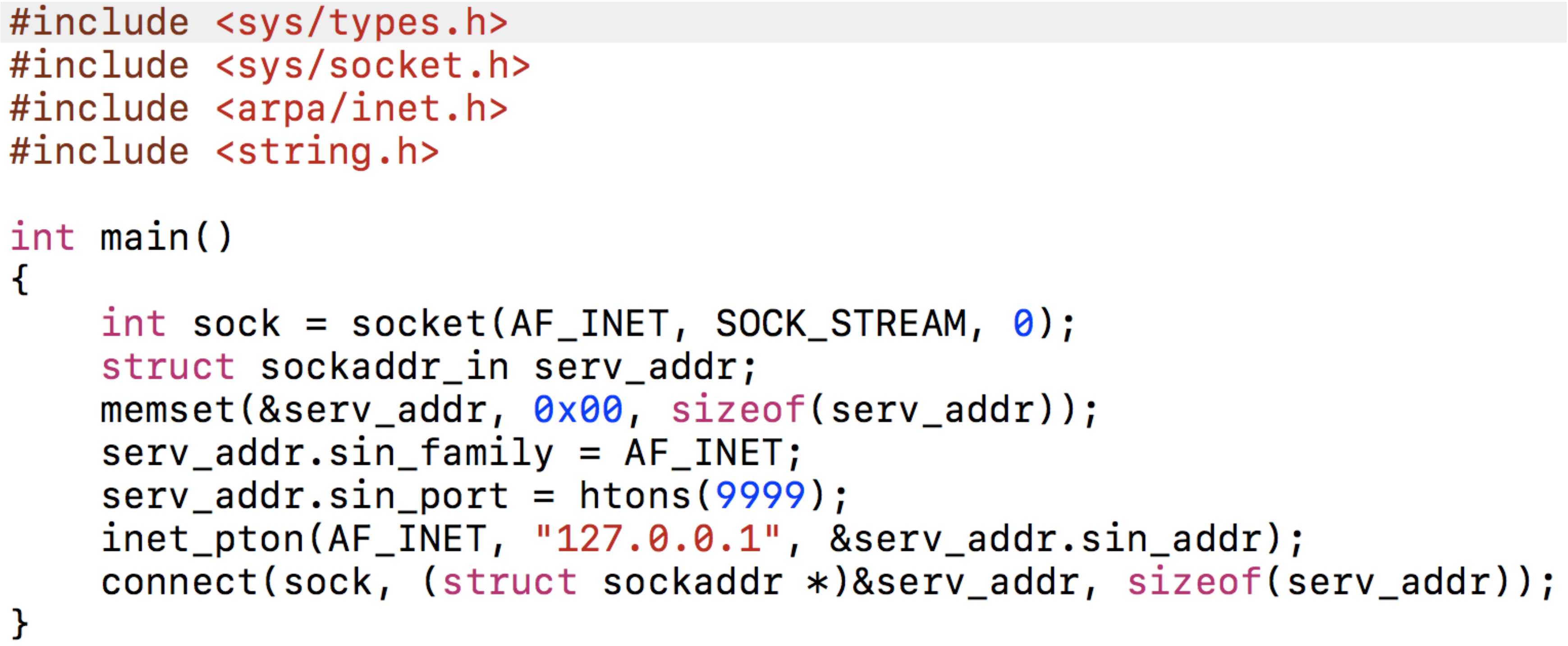
- See the man page for more details

IPv6



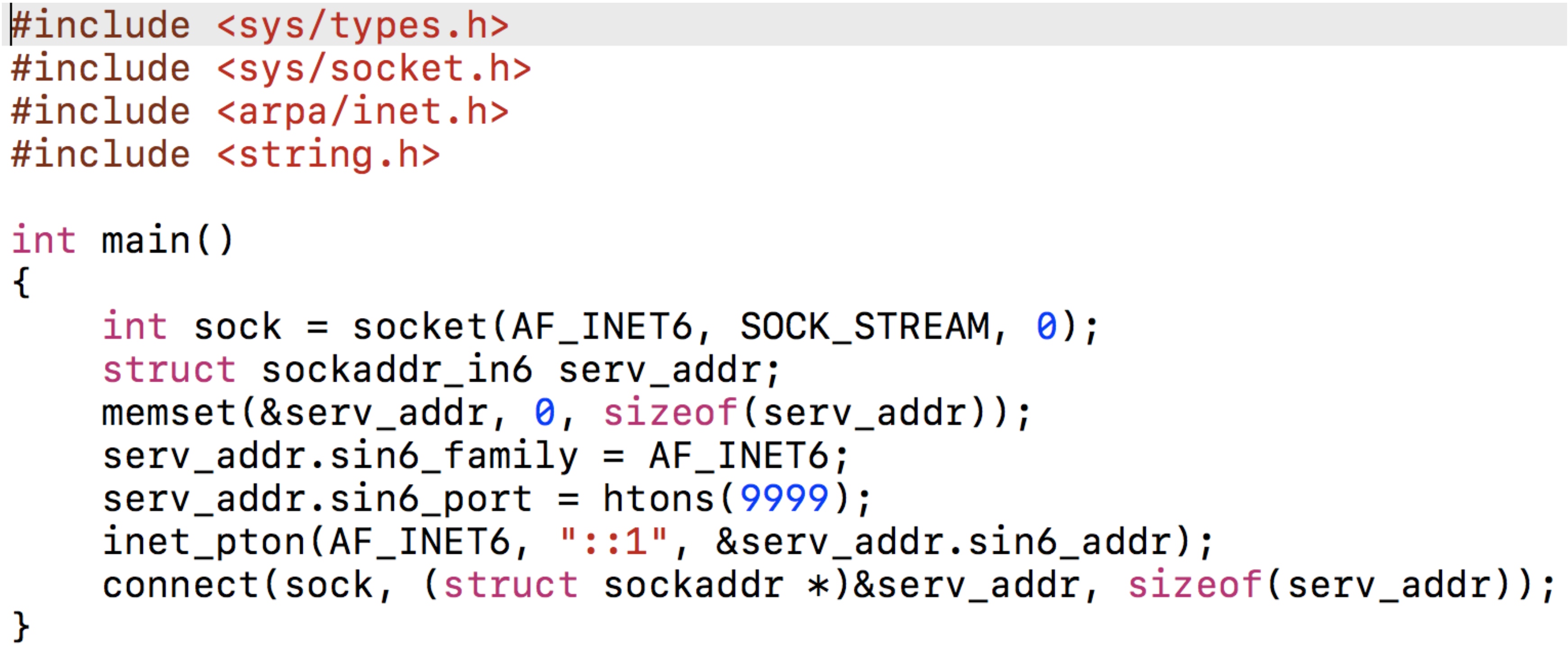
```c
int main()
{
    int status;
    struct addrinfo hints;
    struct addrinfo *servinfo;  // will point to the results

    memset(&hints, 0, sizeof hints); // make sure the struct is empty
    hints.ai_family = AF_UNSPEC;     // don't care IPv4 or IPv6
    hints.ai_socktype = SOCK_STREAM; // TCP stream sockets
    hints.ai_flags = AI_PASSIVE;     // fill in my IP for me

    if ((status = getaddrinfo(NULL, "3490", &hints, &servinfo)) != 0) {
        fprintf(stderr, "getaddrinfo error: %s\n", gai_strerror(status));
        exit(1);
    }

    // servinfo now points to a linked list of 1 or more struct addrinfos

    // ... do everything until you don't need servinfo anymore ....

    freeaddrinfo(servinfo); // free the linked-list
}
```

IPv4 echo client



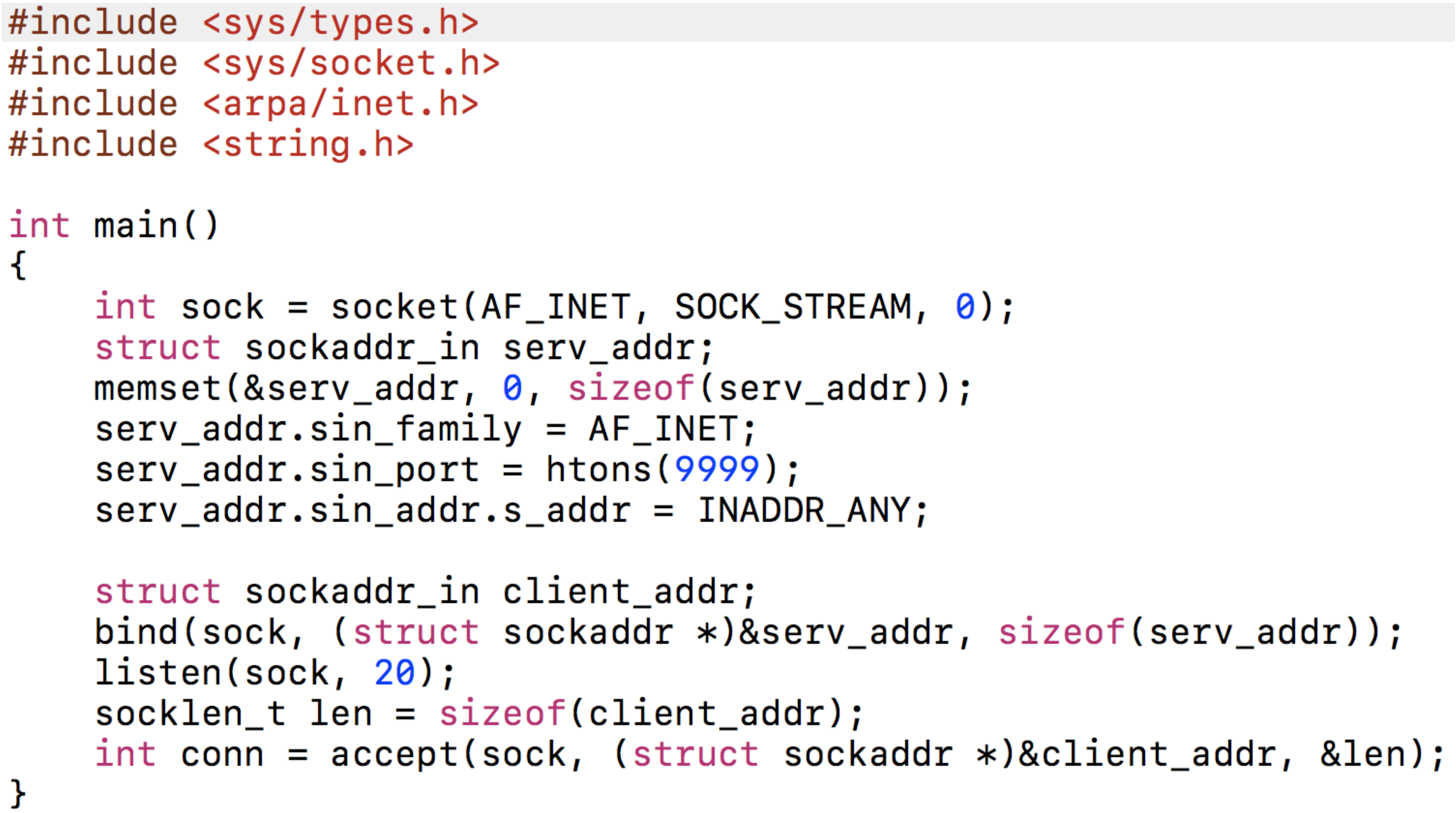
```
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <string.h>

int main()
{
    int sock = socket(AF_INET, SOCK_STREAM, 0);
    struct sockaddr_in serv_addr;
    memset(&serv_addr, 0x00, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(9999);
    inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr);
    connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr));
}
```

IPv6 echo client



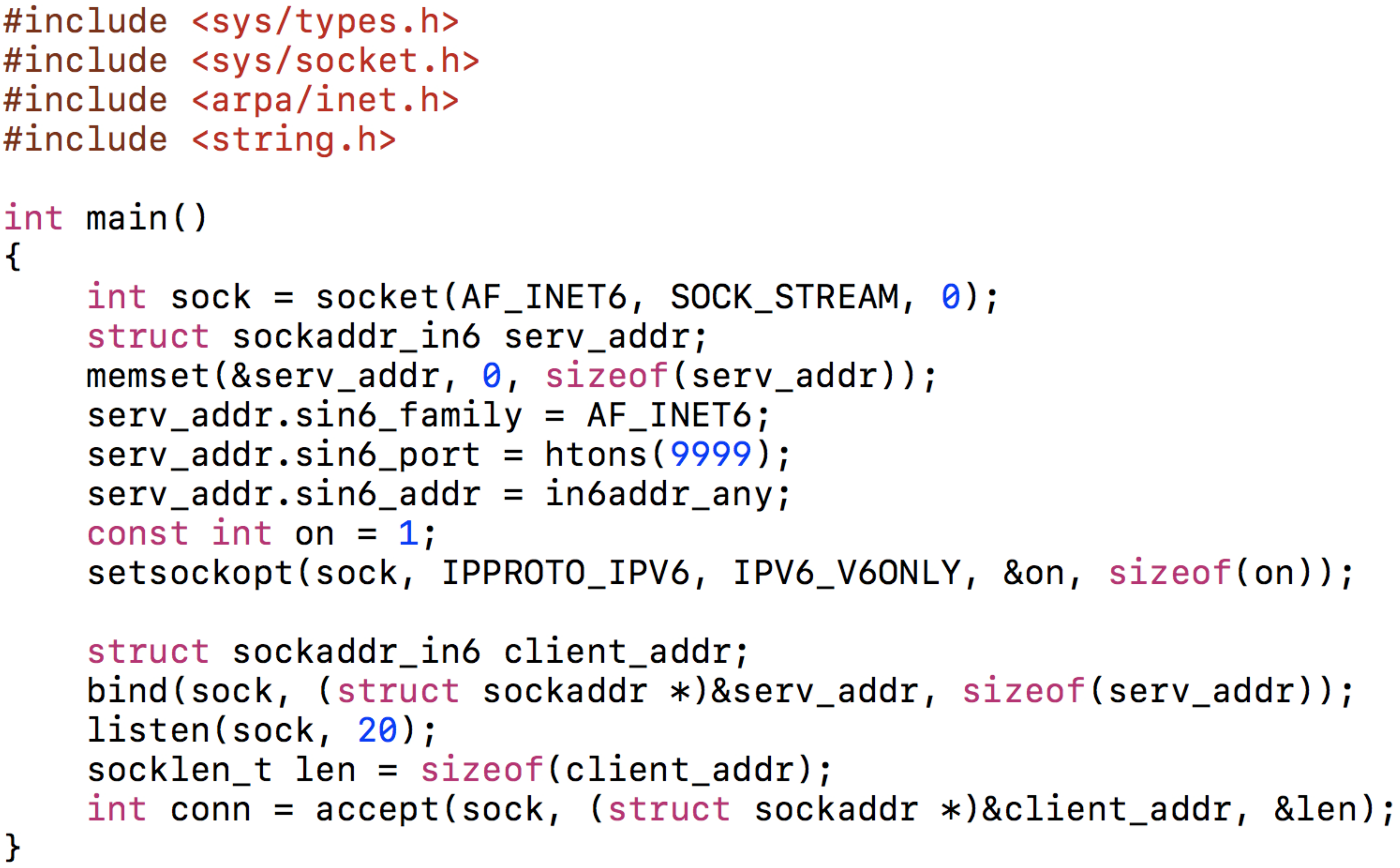
```
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <string.h>

int main()
{
    int sock = socket(AF_INET6, SOCK_STREAM, 0);
    struct sockaddr_in6 serv_addr;
    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin6_family = AF_INET6;
    serv_addr.sin6_port = htons(9999);
    inet_pton(AF_INET6, "::1", &serv_addr.sin6_addr);
    connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr));
}
```

IPv4 echo server



```c
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <string.h>

int main()
{
    int sock = socket(AF_INET, SOCK_STREAM, 0);
    struct sockaddr_in serv_addr;
    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(9999);
    serv_addr.sin_addr.s_addr = INADDR_ANY;

    struct sockaddr_in client_addr;
    bind(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr));
    listen(sock, 20);
    socklen_t len = sizeof(client_addr);
    int conn = accept(sock, (struct sockaddr *)&client_addr, &len);
}
```

IPv6 echo server



```c
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <string.h>

int main()
{
    int sock = socket(AF_INET6, SOCK_STREAM, 0);
    struct sockaddr_in6 serv_addr;
    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin6_family = AF_INET6;
    serv_addr.sin6_port = htons(9999);
    serv_addr.sin6_addr = in6addr_any;
    const int on = 1;
    setsockopt(sock, IPPROTO_IPV6, IPV6_V6ONLY, &on, sizeof(on));

    struct sockaddr_in6 client_addr;
    bind(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr));
    listen(sock, 20);
    socklen_t len = sizeof(client_addr);
    int conn = accept(sock, (struct sockaddr *)&client_addr, &len);
}
```

Transition from IPv4 to IPv6

- Run in dual-stack mode
  - See “Happy Eyeballs” from slide 11 (or Wikipedia)
  - Probably will result in writing more code…

- See Beej’s guide!

- http://www.beej.us/guide/bgnet/html/multi/ip4to6.html

Lab 6

- *getaddrinfo()* lookup program
  - The man page/one of the slides has almost all the code you need
- Due today for full credit
  - Half credit Monday or Tuesday
  - Could change to be full credit at instructor discretion, but don't rely on it