Stream Sockets

- Typically used for client-server communications
  - **Client**: An application that establishes a connection to a server
  - **Server**: An application that receives connections from clients
  - Can also be used for other forms of communication like peer-to-peer

1) Establish connection:
   - client ----> server

2) Communicate:
   - client ----> server

3) Close connection:
   - client <---- server

IPv4 Address Structures

```
// IPv4 4-byte address
struct in_addr {
    uint32_t s_addr;  // Address in network byte order
};

// An IPv4-specific address structure
struct sockaddr_in {
    sa_family_t sin_family;  // Address family: AF_INET
    in_port_t sin_port;      // Port in network byte order
    struct in_addr sin_addr; // IPv4 address
    unsigned char sin_zero[8]; // Pad out to 16 bytes
};
```

```
struct sockaddr_in:

<table>
<thead>
<tr>
<th>family</th>
<th>port</th>
<th>addr</th>
<th>zero</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>17</td>
<td>9</td>
<td>17</td>
</tr>
</tbody>
</table>
```

Resolving DNS Names

- The POSIX way is to use `getaddrinfo()`
  - A complicated system call found in `<netdb.h>`

```
int getaddrinfo(const char* hostname,   // host name
                const char* service,   // service name
                const struct addrinfo* hints,  // hints
                struct addrinfo** res);
```

- Tell `getaddrinfo()` which host and port you want resolved
  - String representation for host: DNS name or IP address
  - Set up a "hints" structure with constraints you want respected
  - `getaddrinfo()` gives you a list of results packed into an "addrinfo" structure/linked list
    - Returns 0 on success; returns negative number on failure
  - Free the `struct addrinfo` later using `freeaddrinfo()`

DNS Lookup Procedure

```
struct addrinfo {
    int ai_flags;  // additional flags
    int ai_family;  // AF_INET, AF_INET6, AF_UNSPEC
    int ai_socktype;  // SOCK_STREAM, SOCK_DGRAM, 0
    int ai_protocol;  // IPPROTO_TCP, IPPROTO_UDP, 0
    size_t ai_addrlen; // length of socket addr in bytes
    struct sockaddr* ai_addr; // pointer to socket addr
    char* ai_canonname; // canonical name
    struct addrinfo* ai_next; // can form a linked list
};
```

1) Create a `struct addrinfo` hints
2) Zero out hints for "defaults"
3) Set specific fields of hints as desired
4) Call `getaddrinfo()` using `&hints`
5) Resulting linked list `res` will have all fields appropriately set

- See `dnsresolve.cc`