Networking, Part 6: Creating a UDP server

jakebailey edited this page on Dec 16, 2014 · 2 revisions

How do I create a UDP server?

There are a variety of function calls available to send UDP sockets. We will use the newer getaddrinfo to help set up a socket structure.

Remember that UDP is a simple packet-based ('data-gram') protocol ; there is no connection to set up between the two hosts.

First, initialize the hints addrinfo struct to request an IPv6, passive datagram socket.

```
memset(&hints, 0, sizeof(hints));
hints.ai_family = AF_INET6; // INET for IPv4
hints.ai_socktype =  SOCK_DGRAM;
hints.ai_flags =  AI_PASSIVE;
```

Next, use getaddrinfo to specify the port number (we don't need to specify a host as we are creating a server socket, not sending a packet to a remote host).

```
getaddrinfo(NULL, "300", &hints, &res);

sockfd = socket(res->ai_family, res->ai_socktype, res->ai_protocol);
bind(sockfd, res->ai_addr, res->ai_addrlen);
```

The port number is <1024, so the program will need `root` privileges. We could have also specified a service name instead of a numeric port value.

So far the calls have been similar to a TCP server. For a stream-based service we would call `listen` and accept. For our UDP-serve we can just start waiting for the arrival of a packet on the socket-

```
struct sockaddr_storage addr;
int addrlen = sizeof(addr);

// ssize_t recvfrom(int socket, void* buffer, size_t buflen, int flags, struct so

byte_count = recvfrom(sockfd, buf, sizeof(buf), 0, &addr, &addrlen);
```

The addr struct will hold sender (source) information about the arriving packet. Note the `sockaddr_storage` type is a sufficiently large enough to hold all possible types of socket addresses (e.g. IPv4, IPv6 and other socket types).

Clone this wiki locally

https://github.com/angrave/SystemPr

Clone in Desktop