# Writing IPv6 Network Applications

Trent 'Lathiat' Lloyd
trent@ztsoftware.net

NextGenCollective.net


# Introduction

- IPv6 is a rapidly growing technology
- The need for IPv6-Compatible applications is in demand
- Many essential applications (squid, sendmail, apache) have already been ported
- Ideally, everythign wants to be IPv6 Compatible, so we must start now.

# Issues at hand

- 128 bit addresses (compared to 32bit)
- DNS Resolution
- IPv4 Fall Back
- IP Header Size Dependance
- Address dependancies (i.e. 127.0.0.1)

# Implementation Changes

- New API calls (inet_pton, inet_ntop, getnameinfo etc...)
- Different paramters (AF_INET6)
- IPv6 Protocol handles IPv4 (::ffff:203.134.64.66) and uses IPv4 transport

# API Change Chart

| API | IPv4 | IPv6 |
|---|---|---|
| Address Conversion | inet_addr, inet_ntoa | inet_pton()<br>inet_ntop() |
| Data Structures | AF_INET<br>in_addr<br>sockaddr_in | AF_INET6<br>in6_addr<br>sockaddr_in6 |
| DNS | gethostbyname()<br>gethostbyaddr()<br>getnameinfo()<br>getaddrinfo() | getipnodebyname(<br>getipnodebyaddr |

# Issues at hand

- Handling larger address size
- Handling different format address
- Resolving new DNS type
- Handling IPv4 and IPv6 Simultaneously

# The Client Side

# Step 1: Resolving the name

- The preffered function is getaddrinfo()

```
struct addrinfo hints, *res, *ressave;
bzero(&hints, sizeof(hints));
hints.ai_socktype = SOCK_STREAM;
hints.ai_family = PF_UNSPEC;
getaddrinfo("hostname", "service", &hints, &res);

while (res) {
  // process
  res = res->ai_next;
}
```

# Determining who we are connecting

- The getnameinfo() function can convert our resolved address to text

```
char addr[INET6_ADDRSTRLEN];
getnameinfo(res->ai_addr, res->ai_addrlen, addr, sizeof(addr),
NULL, 0, NI_NUMERICHOST);
```

- Simply printf("Trying %s\n", addr);

# Connecting to the host (Knock. Kno

- Connect() is used to connect to the remote host

```
int sock;
connect(sock, res->ai_addr,res->ai_addrlen);
```

# Reading and Writing

- recv(socket, buffer, maxsize, flags)

```
char buf[4096];
int recvd;
recvd = recv(sock, buf, sizeof(buf), flags);
buf[recvd] = '\0';
```

# The Server Side

# Finding a listening address

- We also use gethostbyname() to find a listening port

```
struct addrinfo hints, *res, *ressave;
int n,listenfd;
memset(&hints,0,sizeof(struct addrinfo));
hints.ai_flags = AI_PASSIVE;
hints.ai_family = PF_UNSPEC
hints.ai_socktype = SOCK_STREAM
n = getaddrinfo(NULL, PORT, &hints, &res);
ressave=res;
```

# Binding to an address

```
while (res) {
listenfd = socket(res->ai_family, res->ai_socktype, res->ai_protocol);
if (listenfd) {
  if (bind(listenfd, res->ai_addr, res->ai_addrlen))
break;
close(listenfd);
}
}
res = res->ai_next;
}
```

# Questions?

Thanks to:
hs247.com
Abdul Basit
NextGenCollective.net
David Coulson
William Stearns
Grahame Bowland