Name based sockets

Javier Ubillos

Swedish Institute of Computer Science

July 25th, 2010

http://www.ietf.org/id/draft-ubillos-name-based-sockets-01.txt
Name Based Sockets!

Making application development easier
The general problem

All IP (locator) management is done by the application.

There for, all interesting features need to be implemented by the application.

- Mobility
- Multi-homing
- IPv4/IPv6 interoperability
- NA(P)T traversal
- Path diversity exploitation
- Etc...

```c
addr = gethostbyname( someString );
...
connect( ..., addr, ... );
write( ... );
close( ... );
connect( ..., addr, ... );
write( ... );
close( ... );
```
Two typical approaches

Surrogate addresses
- HIP
- Shim6
- MobileIP
- Provider Independent addresses (e.g. with BGP)

Socket abstractions
- .NET
- Java
- Python
- ... Practically any language or framework

Application
Transport
IP
Ethernet

2010-07-26 IETF 78 Maastricht
Surrogate addresses

“Application transparency gives backwards compatibility (API)”

• Extra name spaces.
• Extra resolutions (more indirections)
• Applications are not aware, hence still might try to solve issues in app-space.
Socket abstractions

Developers seem to like them...

• One implementation for every framework
• More often than not
  – Resolve once
  – Reuse IP
  – Reuse IP
  – Reuse IP
  – Reuse IP
  – Reuse IP
What do we want?

• No new indirections
• No new delays (e.g. first packet delay)
• Address management
  – Mobility
  – Multi-homing
  – Renumbering
  – IPv4/IPv6 interoperability
  – NAT penetration
• Backwards compatibility
Components

- API
- Initial name exchange
- Address updates
- Backwards compatibility (on the road map)
API

```c
fd = socket(AF_NAME, SOCK_STREAM, 0);
struct sockaddr_name name_sock;

// Initialize name_sock with remote name
bind(fd, name_sock, sizeof(name_sock));
connect(fd, name_sock, sizeof(name_sock));

write(fd, send_buffer, len);
read(fd, recv_buffer, len);
```
The components (API)

- **listen()** - Prep for incoming session
  ```
  fd = listen( local_name, peer_name, service, transport );
  ```
- **open()** - Initiate outgoing session
  ```
  fd = open( local_name, peer_name, service, transport );
  ```
- **accept()** - Receive incoming session
  ```
  accept( peer_name, fd );
  ```
- **read()** - Receive data
  ```
  data = read( fd );
  ```
- **write()** - Send data
  ```
  write( fd, data );
  ```
- **close()** - Close session
  ```
  close( fd );
  ```
Initial name exchange
Traditional

Name: host.left.org
IP: a.b.c.d

Name: host.right.org
IP: w.x.y.z
Traditional

1. Send to host.right.org

Resolve to w.x.y.z

From: a.b.c.d
To: w.x.y.z

From: w.x.y.z
To: a.b.c.d

I like this!
I'll reply.

Name: host.left.org
IP: a.b.c.d

Name: host.right.org
IP: w.x.y.z
Name exchange

From: a.b.c.d
To: w.x.y.z
Resolve to w.x.y.z

From: w.x.y.z
To: a.b.c.d

From: host.left.org
To: host.right.org

From: host.right.org
To: host.left.org

From: host.left.org
To: host.right.org

From: host.right.org
To: host.left.org

I like this!
I'll reply.

Name in the reply = Upgraded hosts!
Name exchange

Name: `host.left.org`
IP: `a.b.c.d`

Name: `host.right.org`
IP: `w.x.y.z`

Resolve to `w.x.y.z`

From: `a.b.c.d`
To: `w.x.y.z`

From: `host.left.org`
To: `host.right.org`

From: `w.x.y.z`
To: `a.b.c.d`

From: `host.right.org`
To: `host.left.org`
Backwards compatibility

Name: host.left.org
IP: a.b.c.d

Name: host.right.org
IP: w.x.y.z

1. Send to host.right.org
Listen
From: a.b.c.d
To: w.x.y.z
Resolve to w.x.y.z
From: w.x.y.z
To: a.b.c.d
From: host.left.org
To: host.right.org
Ignore that weird extension. But reply to the packet.
No name-header sent back. I'll try N more times and then give up.
The current prototype

- Supports TCP
  - Uses TCP semantics
    - socket(), listen(), open(), accept(), read(), write()
- Supports Shim6
  - Well, to a certain extent, we are working on it :)
- Exchanges names
- Linux
  - Ubuntu (client/server)
  - Android (client)

Implementation by Juan Lang (UC Davis)
and by Zhongxing Ming (Tsinghua University)
Current development

• Support for UDP
  – Using TCP-like semantics

• Mobility/Multi-homing
  – Shim6

• Collaboration between
  – Ericsson
  – Tsinghua University
  – Swedish Institute of Computer Science
The road map

• IPv4/IPv6 Interoperability
• NAT penetration
• Path diversity utilization
• Naming resolution (depth)
  – Host
  – Application
  – Etc…

• And more... Do you have any suggestions? Please let us know!
Questions?