

Porting and Deploying VoIP to IPv6: Lessons Learned

Marc Blanchet

Simon Perreault

Viagénie

<http://www.viagenie.ca>

Presented at ClueCon, Chicago, August 2008

ANNOUNCEMENT



Freeswitch now supports IPv6.

IPv6 port got integrated into 1.0.1

... BIG APPLAUSE... ;-)

Viagénie Team Credentials



- Consulting and R&D in IP networking
 - Customers such as providers, enterprises, manufacturers for IPv6 deployment, application porting, ...
- 20+ years in IP networking and Unix
- 10 years on IPv6
- Team wrote IETF drafts and RFCs. Co-chair of IETF WGs (idn, vcarddav)
- Wrote the “Migrating to IPv6” book, published by Wiley
- Gave IPv6 tutorials at many conferences. Authored and delivered the Cisco IPv6 course.
- Co-founder and member of the board, IPv6Forum
- Member of steering group of North American IPv6 Task Force
- VoIP developers, ported Asterisk and Freeswitch to IPv6. (Also ported NTP, Quake, ... to IPv6)

Plan



- IPv6
- Why IPv6 and VoIP
- New API
- Lessons learned while porting...
 - Asterisk (see <http://www.asteriskv6.org>)
 - FreeSWITCH
- Conclusion

IPv6?



- New version of IP:
 - fixes IPv4 issues
 - adds functionality
- Addresses:
 - 128 bits
 - written in hex with : as separator; method to compress the writing: all zeros = ::
 - **2001:db8:1:1::1**
 - In URL: enclose with []: **sip:jdoe@[2001:db8:1:1::1]:5060**
 - Loopback is **::1**
 - Link(Subnet,vlan,...) mask is fixed: **/64**
 - Unique private address space: no collision of private networks

IPv6?



- Addresses (cont):
 - Scoped addressing: link scope, site scope. An enabled IPv6 stack has already an IPv6 address (link scope) on each interface, even if no IPv6 external connectivity.
 - Multiple addresses per interface: link-scope, global, [site,...]
 - No NAT.
- Mobility: keep connections up even when host changes IP address
- Autoconfiguration: Stateless address allocation **without DHCP server**. Routers announce the link prefix on the link. Hosts use their MAC address for the host part of the address
- more...

IPv6 Market



- IPv4 address depletion: < 25% of remaining address space. Predictions of exhaustion for 2009-2011.
- Asia
 - Japan: see <http://www.v6pc.jp>
 - China: through NGN. Olympics is important milestone.
- US government:
 - Mandating IPv6 for 2008 in all agencies
 - DoD is leading
- VoIPv6: SNOM, PBXnSIP, leading vendors
- Providers (short list):
 - Teleglobe/VSNL/Tata, NTT, AT&T, GlobalCrossing,
 - Comcast: can't address all the devices (100M+) with IPv4. Deploying IPv6. (DOCSIS 3.0 is IPv6-ready).

IPv6 Support



- Support on OS (stack and API):
 - Same (new) API everywhere!!! ;-)
 - Since: Linux 2.4, FreeBSD 4.X, MacOSX 10.2, Windows XP, Solaris 8, ...
- Opensource Apps: Apache 2.0+ (1.3 with a patch), Sendmail, Postfix, OpenSSH, Xfree/Xorg, ...
 - Now Asterisk and *FreeSWITCH*... ;-)
- Support on network gear: Cisco, Juniper, Checkpoint, Quagga/Zebra, ...

Why IPv6 and VoIP?



- IPv6 and SIP
 - delivers direct end-2-end reachability between any host.
 - No NAT, no STUN, no TURN, no ICE, no MIDCOM, = no complexity, “just works”.
 - True end-2-end media path.
 - Much easier to deploy. A VoIP-IPv6 deployment in Japan found important cost reductions because of the ease of installation and support.
- To have an IPv6-enabled application, such as a PBX, need to convert to the new API.

New API



- New API for IPv6 [RFC3493, RFC3542]
 - Makes the application version independent. The stack chooses which IP version will be used for that connection.
 - A ported application becomes IP version unaware.
 - No change to `socket()`, `bind()`, `listen()`, `accept()`, `connect()`, `recv()`, `send()`, `close()`...
- Changes:
 - Struct **hostent** replaced by struct **addrinfo**
 - Addrinfo is a linked list of addresses
 - It contains everything needed to initialize a socket.

New API



- Changes:
 - sockaddr record
 - **sockaddr_in** : IPv4
 - **sockaddr_in6** : IPv6 only. Do not use.
 - **sockaddr_storage**: version independent for memory allocations.
 - **sockaddr ***: for casting
 - **gethostbyname()** replaced by **getaddrinfo()**
 - **gethostbyaddr()**, **inet_addr()**, **inet_ntoa()** replaced by **getnameinfo()**
- More considerations:
 - Parsing URLs: need to take care of the IPv6 syntax (i.e. [])
 - Parsing and storing IP addresses

New API



- History:
 - New API had multiple revisions, based on feedback of porting, deployment and engineering.
 - Documentation and “old” code still uses old API calls.
 - Old ways:
 - IPv4-mapped addresses: important security issues.
 - Old calls (deprecated, nowadays no more available in some OS):
 - `gethostbyname2()`
 - `getipnodebyname()`
 - `getipnodebyaddr()`

FreeSWITCHv6

FreeSWITCHv6



- FreeSWITCH is IPv6-enabled since 1.0.1
- Running in production as our main telephony switch for one month
- And there was much rejoicing...

FreeSWITCHv6



- SIP stack is Sofia-SIP, and is IPv6-enabled.
- Needed work:
 - mod_sofia glue
 - Uses address as string for registrar key. (Good!)
 - Some IPv4-specific URI building logic.
 - Some IPv4-specific SDP building logic.
 - Core: `$$local_ip_v6` now contains useful data.
 - RTP:
 - Used a single port for input and output. Couldn't transcode network protocols.
 - Now opens a second port of other family when needed.

FreeSWITCHv6 (2)



– ACLs

- Was completely IPv4-specific.
- Redesigned for IPv4 and IPv6.
- New in IPv6: scope ID must match.
- Potential for optimization with SSE2 (anyone interested?)
- Not contributed yet, needs more testing.

Lessons Learned

Use Addresses Sparingly



- Call connect() or bind(), then discard the address.
- Anti-pattern:
 - Have a host name resolving function return an address.
 - Later, use that address.
- Better:
 - Have a host name resolving function return **a list** of addresses.
 - Later, use these addresses.
- Best:
 - Combine the connecting/binding with the resolving.

Prepare for Multiplicity



- With version-independent programming, addresses are never encountered alone.
- Binding to **localhost** binds an IPv4 socket to **0.0.0.0** and an IPv6 socket to **::** (depends on OS).
- Hosts often have A as well as AAAA records. Try all of them when calling `connect()`.

Banish Old APIs



- You should never use these:
 - `inet_addr()`, `inet_aton()`, `inet_ntoa()`
 - `inet_pton()`, `inet_ntop()`
 - `gethostbyname()`, `gethostbyaddr()`
- Not even these: (at least not for addresses)
 - `htonl()`, `htons()`, `ntohl()`, `ntohs()`
- All you need is:
 - **`getaddrinfo()`** (string to address)
 - **`getnameinfo()`** (address to string)

An Address is Atomic



- Do not separate address components.

– Anti-pattern:

```
if ( sa->sa_family == AF_INET ) {
    addr = ((sockaddr_in*)sa)->sin_addr.s_addr;
    port = ((sockaddr_in*)sa)->sin_port;
} else if ( sa->sa_family == AF_INET6 ) {
[...]
```

```
snprintf( uri, sizeof(uri), "sip:%s@%s:%hu",
    user, host, port );
```

– Why it is bad:

- Repeated logic for brackets in URL.
- Not version-independent.
- What about IPv6 scope ID?

An Address is Atomic (2)



- Better:

```
enum {
    URI_NUMERIC_HOST = 1,
    URI_NUMERIC_PORT = 2,
    URI_IGNORE_SCOPE = 4,
    [...]
};

int build_uri( char *uri, size_t size,
               const char *user,
               const sockaddr *sa, socklen_t salen,
               int flags );
```

Deployment Considerations

IPv4 - IPv6 Interoperability

- IPv4 and IPv6 UAs can communicate via a relay.
- Usually relay is a B2BUA (e.g. FreeSWITCH)
- Relaying media may cause unwanted load.
- Consider using a cross-protocol TURN server instead.
- A TURN server is designed for this task.
- Reliability and scalability provided by anycast + load balancing mechanism.
- Details on this, and more, in Wednesday presentation on STUN / TURN / ICE.

Conclusion



- Discussed:
 - Benefits of IPv6 and why open-source B2BUA benefit from being IPv6-enabled.
 - How to port an application to IPv6
 - Changes to FreeSWITCH
 - Lessons learned
 - VoIPv6 deployment
- Try IPv6 now!
 - <http://freenet6.net>
 - <http://asteriskv6.org>
 -

Questions?



Contact info:

Marc.Blanchet@viagenie.ca

Simon.Perreault@viagenie.ca

This presentation is available at <http://www.viagenie.ca/publications/>

References

- <http://www.asteriskv6.org>
- [RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, February 2003.
- [RFC3542] Stevens, W., Thomas, M., Nordmark, E., and T. Jinmei, "Advanced Sockets Application Program Interface (API) for IPv6", RFC 3542, May 2003.
- IPv6 Network Programming, Junichiro itojun Hagino, Elsevier, 2004, ISBN 1555583180.
- Migrating to IPv6, Marc Blanchet, Wiley, 2006, ISBN 0-471-49892-0, <http://www.ipv6book.ca>

Backup Slides



Best Practices for API usage

- Use **sockaddr_storage** for storing sockaddrs.
- Use **sockaddr *** for pointer to sockaddrs
- Always pass and carry the sockaddr length (in a **socklen_t**) to be fully portable across OS platforms.
- After the **getaddrinfo()** call, go through the link list of addrinfo to connect.
- Parse addresses and URL to support both IPv4 and IPv6 addresses (with port numbers) syntax.
- Do not use IPv4-mapped addresses or old API calls (**gethostbyname2()**, **getipnode*()**)

Eliminate Timeouts



- Many users already have an IPv6 address that is not reachable globally. (Local router, zombie Teredo, etc.)
- When connecting to results of `getaddrinfo()` sequentially, IPv6 connections will timeout.
- Reordering results so that IPv4 is tried first is a bad idea because the reverse may also be true.
- **Solution:** connect in parallel. (harder to implement)
- Even worse: DNS servers may timeout when queried for AAAA records. Cannot use `getaddrinfo()`.
- **Solution:** single-family `getaddrinfo()` calls in parallel.

Eliminate Timeouts (2/2)



- Combine the two previous solutions within a single API for resolving and connecting.

```
int fd = resolve_connect( "example.com", "80" );
```

- Use worker threads for resolving and connecting in parallel. (Better: a single thread with nonblocking sockets and a DNS resolving library.)
- Connect to each address as soon as it is received. Do not wait for all address families to finish resolving.
- Cancel other connections once one succeeds.
- Disadvantage: this wastes packets. May be significant in some cases (e.g. lots of short connections).

For Protocol Designers



- Protocols that transport addresses are harder to implement in a version-independent way.
- SIP, RTSP, and SDP do transport addresses **very much**.
- Many ways to encode addresses make it hard:
 - By themselves (e.g. **c=IN IP6 2001:db8::1**)
 - With brackets and port (e.g. **Via: SIP/2.0/UDP [2001:db8::1]:5060**)
 - Implicitly as part of any URI (e.g. **From: <sip:jdoe@example.com>**)