Porting and Running Asterisk to IPv6

Presented at APRICOT, Bali, Feb 28th 2007

Marc Blanchet

Viagénie

http://www.viagenie.ca

Credentials

- 20+ years in IP networking and Unix, with 10 years on IPv6...
- IP engineering standards(IETF):
  - Wrote IETF drafts and RFCs.
  - Co-chaired internationalized domain names (idn) IETF wg
- Authoring:
  - Book: Migrating to IPv6, Wiley, 2006.
  - Cisco IPv6 course (co-author)
  - Tutorials on IP, security, Ipv6, etc... at many conferences, organisations
- IPv6forum: co-founder, board member. North American Ipv6 Task Force: steering group member.
- Asterisk developer, co-ported Asterisk to IPv6.
- President of Viagénie, consulting in advanced IP networking. Helping providers, enterprises, manufacturers and governments. IPv6, VoIP, Asterisk, Security, Internationalization, etc.

Plan

- Why IPv6 and Asterisk
- Asterisk architecture
- Challenges
- IPv6 in chan_sip
- Changes in Asterisk code
- Sip.conf
- Demo
- Running in production
- Lessons learned
- Next Steps
- Conclusion

VoIP today

- SIP-based VoIP:
  - Separate signaling and media path
  - Does not work well with NAT.
  - Multiple variations of NAT traversal solutions:
    - STUN, TURN, ICE, ...
    - showed complexity and brittleness
    - and lack of support in the implementations
  - User Agent may be behind a NAT with some efforts.
  - But it is very difficult to have a SIP server (proxy,registrar, ...) to be behind a NAT.

Consequences

- User consequence:
  - Calls do not go through
  - Audio is one-way
  - DTMF does not work
- Implementor consequence:
  - Very complex implementations.
  - Fragile. Difficult to debug.
  - Long cycle of development/testing.
- Deployment consequence:
  - careful planning
  - long time for deploying, testing, etc..

Asterisk

- http://www.asterisk.org
- “Asterisk® is a complete IP PBX in software. It runs on a wide variety of operating systems including Linux, Mac OS X, OpenBSD, FreeBSD and Sun Solaris and provides all of the features you would expect from a PBX including many advanced features that are often associated with high end (and high cost) proprietary PBXs. Asterisk's architecture is designed for maximum flexibility and supports Voice over IP in many protocols, and can interoperate with almost all standards-based telephony equipment using relatively inexpensive hardware.
- Asterisk® is released as open source under the GNU General Public License (GPL), meaning that it is available for download free of charge. Asterisk® is the most popular open source software available, with the Asterisk Community being the top influencer in VoIP.

Bridging everything together

- Asterisk:
  - bridges technologies together:
    - PSTN: analog, ISDN
    - Voice codings
    - VoIP: SIP/SDP/RTP, Skinny, H323, IAX, MGCP,
    - IP, linux, HTTP, DNS, ENUM
    - Messaging: Jabber, SMS, ...
    - Text to Speech
  - has a whole set of PBX features
  - all together creates a great framework and playground for innovative applications.
- Is an open-source project, supported by Digium, founded by Mark Spencer, author of Asterisk.

Some Asterisk Features

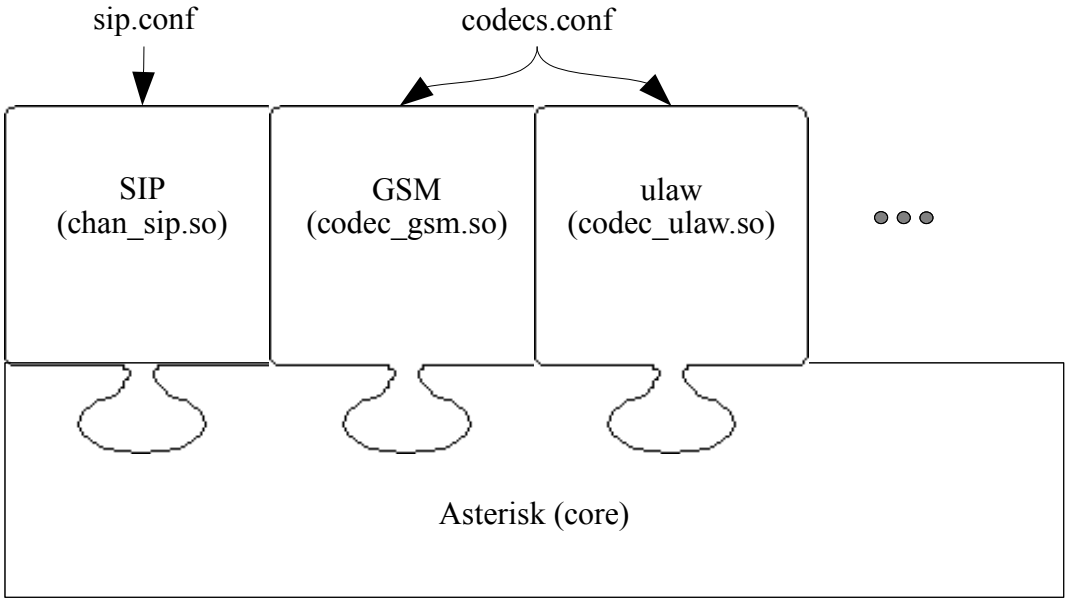
- Bridging between any channel (PSTN, VoIP, ...) using any technology.
  - Transcoding between any channel
- Automated Attendant, Interactive Voice Response, Directory, Music on Hold, Call Detail Records, Text-to-speech
- Call Forward, Call Monitoring, Call Parking, Call Queuing, Call Recording, Call Routing, Call Transfer, Call Waiting, Blind Transfer, Remote Call Pickup, Caller ID, Voicemail
- Conferencing, Follow-me, Trunking
- Call centers, Call queues, Call agents, Predictive Dialing
- Database Integration
- E911, ENUM
- Fax Transmit and Receive (3rd Party OSS Package), SMS
- All free! And relatively easy to configure and use.

Why IPv6 and Asterisk?

- As any VoIP system, Asterisk does suffer NAT.
- Asterisk had no IPv6 support
- 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 Asterisk, need to convert to the new API.

Asterisk Architecture

- Channels: SIP, IAX, MGCP, ZAP(PSTN), etc..
- Each channel is implemented as a loadable module
- SIP Channel(chan_sip) is a “monolithic” channel that does SIP and SDP.



Challenges with IPv6 in Asterisk chan_sip

- Current architecture supports a single socket : 'sipsock'.
- The default source address is hardcoded to 0.0.0.0.
- The RTP socket is initialized from 'sipsock'
- Widespread use of sockaddr_in structures and short buffers (>256 bytes) to store hostnames and IP address strings.

- Many instances of similar code for parsing SIP url.

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

Best Practices for API usage

- Use sockaddr_storage for storing sockaddrs.
- Use sockaddr * for pointer to sockaddrs
- Always pass and carry the sockaddr length 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, old API calls (gethostbyname2(), getipnode*())

Design choices

- Use multiple sockets
  - Initial patch provides 1 socket per address family.
  - future work should include multiple sockets for each address family.
- Version independent when possible
  - Whenever possible, do not use sockaddr_in or sockaddr_in6 and never guess at the length of a sockaddr structure.
  - Only exception should be for setting socket options.

Code changes

- Replaced all use of sockaddr_in in data structures with sockaddr_storage.
- Associates a socklen_t element with each sockaddr_storage.
  - the socklen member is only initialized when a sockaddr_in of sockaddr_in6 structure is copied in the allocated memory... never when the memory is allocated.
- Created a ast_vinetsock API based on ast_netsock API
  - ast_netsock is IPv4-only. Used only in chan_IAX
  - Address string parsing.
  - Address structure handling.
  - Socket management.

New ast_vinetsock API

- ast_netsock (netsock.h) is currently used in chan_iax, not in chan_sip.
- ast_netsock has link lists to manage multiple sockets.
- the ast_netsock API was augmented to support IPv6.
- New and modified functions are in the new ast_vinetsock namespace (defined in netsock.c): no collision with ast_netsock.
- 3 types of functions are defined in ast_vinetsock:
  - Address string parsing.
  - Address structure handling.
  - Socket management.

String parsing functions

- Parse host:port and address strings in a version independent way.
- Used for:
  - Parsing and validation of configuration files.
  - Parsing SIP header fields such as 'contact' and 'via'.
-
- Db store uses ':' between fields. ':' is used in IPv6 address. Enclosing IPv6 address in []. Impact for other db readers.

Address structure handling functions

- Initialize sockaddr structures from strings.
- Extract data from sockaddr structures.
- Build host:port and address strings from sockaddr structures.
- Used for:
  - Selecting a source address.
  - Printing addresses and host:port strings to logs and console.
  - Building SIP/SDP fields from address structures.

Socket management functions

- Initialize sockets through ast_vinetsock structures.
- Set socket options.
- Bind on sockets and register callback functions.
- Used for:
  - Initializing IP listeners

Impacts on Asterisk Code

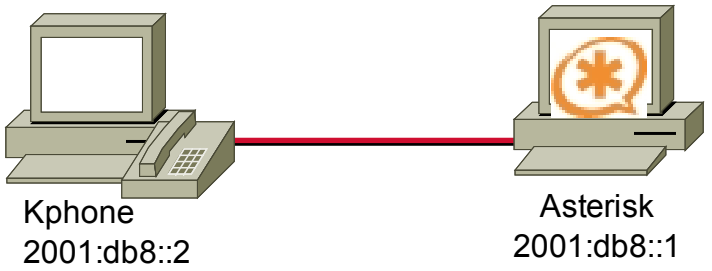
- Files touched:
  - netsock.c/.h
  - chan_sip.c
  - rtp.c
  - Few others
- Some numbers:
  - ~25% of functions were changed/touched
  - many thousand lines changed/touched.
  - “Everywhere” in chan_sip, because: networking, logging (printing addresses) and sip url parsing.

Modifications to sip.conf

- 'bindaddr' now supports the address:port syntax such as:
  - 10.1.1.1
  - 10.1.1.1:5060
  - [2001:db8::1]
  - [2001:db8::1]:5060
- If no 'bindaddr' is specified for an address family, the wildcard is used (0.0.0.0 AND [::]).
- 'host' contains only the address, therefore no brackets.
- 'bindport' is still supported for backward compatibility.

'Hello World' demo

- Uses Kphone as IPv6 SIP UA.
- Register to Asterisk.
- Make a call to play the 'Hello world' sound file.

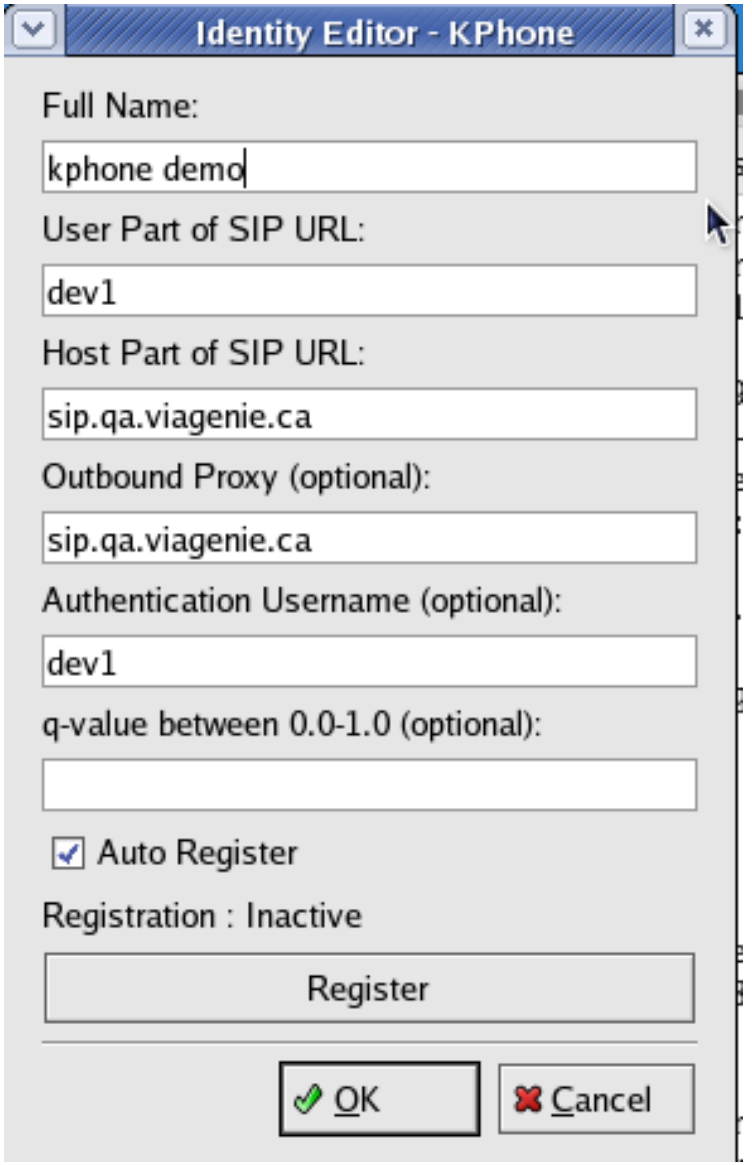


'Hello World' demo (cont.)

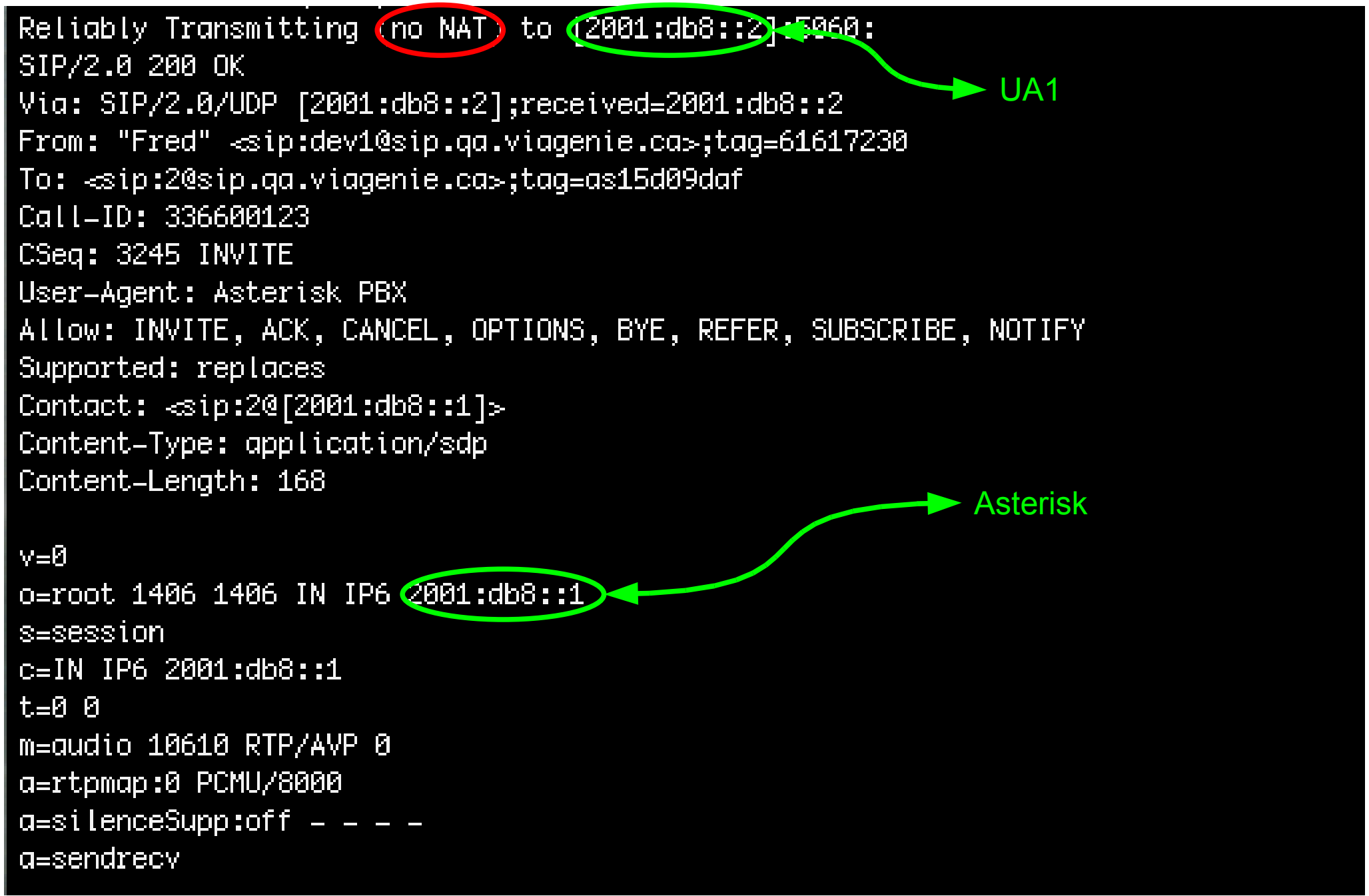
```
[general]
context=internal
bindaddr=[2001:db8::1]
allow=ulaw

[dev1]
type=friend
host=dynamic
context=internal
disallow=all
allow=ulaw

[dev2]
type=friend
host=dynamic
context=internal
disallow=all
allow=ulaw
```

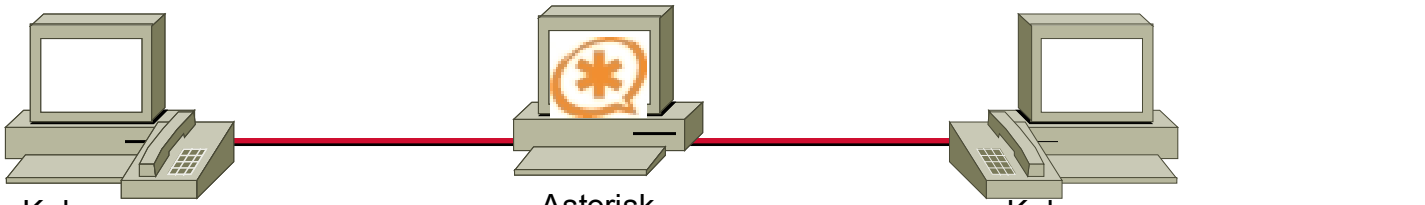


'Hello World' demo (cont.)



2 Phones call demo

- 2 Kphone IPv6 SIP User Agents register to an Asterisk server.
- Establish a SIP call between the two user agents through an extension on Asterisk.



Bidirection call demo (cont.)

VIAGĒNIE

```
Reliably Transmitting (no NAT) to [2001:db8::3]:5060:
INVITE sip:lefebvre@[2001:db8::3];transport=udp SIP/2.0
Via: SIP/2.0/UDP [2001:db8::1]:5060;branch=z9hG4bK1dc90af0;rport
From: "Fred" <sip:dev1@[2001:db8::1]>;tag=as38e677
To: <sip:lefebvre@[2001:db8::3];transport=udp>
Contact: <sip:dev1@[2001:db8::1]>
Call-ID: 5351c608290f3c9d03ab0e346ed44a80@2001:db8::1
CSeq: 102 INVITE
User-Agent: Asterisk PBX
Max-Forwards: 70
Date: Wed, 18 Oct 2006 19:38:06 GMT
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY
Supported: replaces
Content-Type: application/sdp
Content-Length: 224

v=0
o=root 1406 1406 IN IP6 2001:db8::2
s=session
c=IN IP6 2001:db8::2
t=0 0
m=audio 32770 RTP/AVP 0 101
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16
a=silenceSupp:off - - - -
a=sendrecv
```

UA1

Asterisk

UA2

IPv6 SIP user agents

- Few open source IPv6 SIP user agents are available at this time. Many pretend to be Ipv6-ready, but they were never tested or with very low number of tests. We have been sending patches to some of them.
- Makes testing and especially interop testing more limited.
- We tested 2 softphones with a 'working' ipv6 implementation:
  - kphone 3.1.1 with IPv6 patch.
  - Linphone 1.3.5
  - Both implementations contains (IPv6) bugs.
- Testing 3 commercial SIP UA IPv6-enabled: both hard and softphones. One that worked well: Counterpath Eyebeam (Windows version) (not yet released public)

Running Asterisk-v6 in Production

- User point of view:
  - no difference. Same quality of voice, etc...
- Infrastructure point of view:
  - Dual-stack network.
  - Some phones are v4, others are dual-stack, some are (by config) restricted to v6.
- Deployment point of view:
  - Much easier: easier to deploy phones in home networks, for road-warriors, etc..
  - Easier to define firewall rules, since one can filter based on the source and destination addresses/prefixes (not possible with NAT)
  - Easier to troubleshoot, since easy to trace

Lessons Learned

- IPv4-IPv6 SIP in production is challenging
  - Found without trying to do:
    - IPv6 SIP signaling but media path is established using IPv4.
    - Troubleshooting is more difficult?
    - Need to investigate
- Conformance support for IPv6 SIP implementations.
- Difficult to find other implementations to test with.
- Based on deployment experience, should write a BCP paper on IPv4-IPv6 SIP deployments.

Next Steps

- Code is based on august 2006 trunk. Need to remerge to 1.4 and trunk.
- Running in production in our office and remote sites with IPv6 **and** IPv4 phones.
- Discuss with community how to integrate code into trunk (ongoing)
- Add a startup flag to Asterisk to disable IPv6.
- More testing! Especially Interop tests.
  - test with other implementations (SER, ...)
  - test with other IPv6 SIP UAs.... if you have one, please contact us.
- Improve IPv6 support in chan_sip to better handle complex scenarios.
  - implement ANAT [RFC4091, RFC4092].
  - IPv6 <-> IPv4
- Add IPv6 support to chan_iax (work in progress) and chan_*.
- Fix bugs

Conclusion

- Discussed:
  - the benefits of IPv6 and Why Asterisk benefits of being IPv6-enabled.
  - How to port an application to IPv6
  - Changes to Asterisk
  - Demo
  - Next Steps
- Information on this Asterisk-IPv6 project is available at:
  - http://www.asteriskv6.org .
  - We will be posting progress, tests with IPv6 UA, code, ...

Questions?

Contact info:

Marc.Blanchet@viagenie.ca

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

Information on this Asterisk-IPv6 project: http://www.asteriskv6.org

References

- [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