PJSIP: Open Source Compact SIP and Media Stack

Perry Ismangil
and
Benny Prijono

Realtime Multimedia Communications

- Voice over IP (VoIP)

- Audio over IP: remote audio broadcasting

- Unified Communications

- Virtual world conversations

Realtime Multimedia Communications

- Signaling: SIP

- Negotiation: SDP

- Media transmission: RTP/RTCP

- Presence

- NAT traversal

Project Objectives

- Integrated SIP and Media stack package
  - Ease of use
- Compliance
  - Regression tests against compliant servers and clients
-
  - Test by community
-
  - Global interoperability events from SIP Forum

Project Objectives

- Very Very Portable

- One of the fastest (performance)

- One of the smallest (footprint & memory)

PJSIP History

Background:

- 2000: SLSIP (C++)

  ○ pc2call.com, first SIP web based dialer

- 2001-2003: prototyping

PJSIP History

- Feb 2006: PJSIP and PJMEDIA 0.5
  - pjsip.org, dual license

- Nov 2006: Incorporation of Teluu

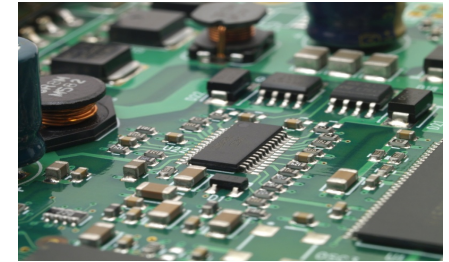- Nov 2007: Global SIP Interoperability Testing

PJSIP History

- Oct 2008
  - Release 1.0

  - More than twenty known projects/products using pjsip

  - More than 250,000 lines of code (source: ohloh.net)

PJSIP Community

- Downloads: 800 per month

- More than 500 mailing list members

- From the community

  - Various platform supports
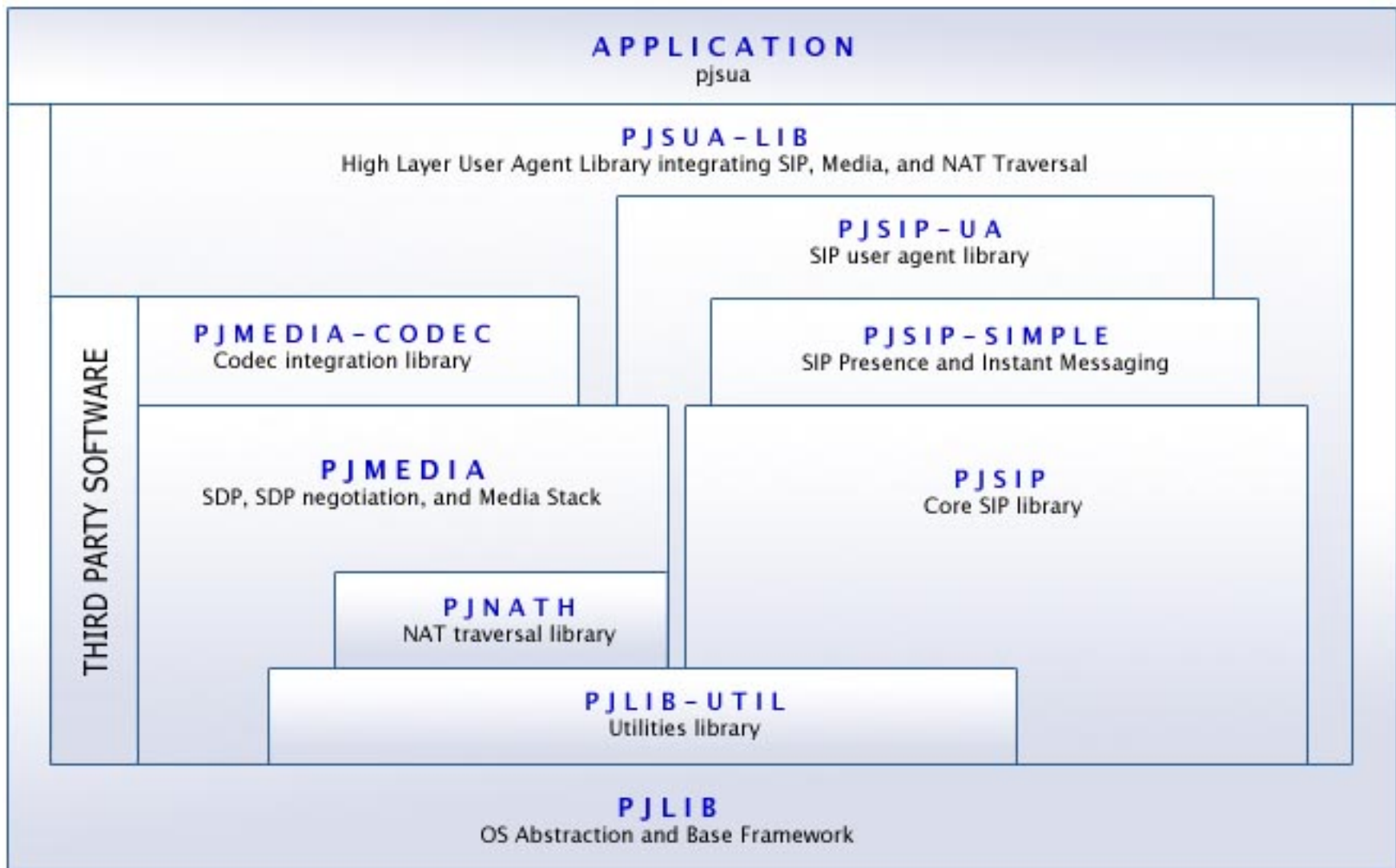
  - Additional language bindings

Platforms

Mac

Windows

ARM

intel

TEXAS INSTRUMENTS

Windows Mobile

iPhone

Embedded With RTEMS
www.rtems.com

S60

All trademarks belong to respective owners

Framework Architecture

APPLICATION
pjsua

PJSUA–LIB
High Layer User Agent Library integrating SIP, Media, and NAT Traversal

PJSIP–UA
SIP user agent library

PJMEDIA–CODEC
Codec integration library

PJSIP–SIMPLE
SIP Presence and Instant Messaging

PJMEDIA
SDP, SDP negotiation, and Media Stack

PJSIP
Core SIP library

THIRD PARTY SOFTWARE

PJNATH
NAT traversal library

PJLIB–UTIL
Utilities library

PJLIB
OS Abstraction and Base Framework

Operating System / Platfiorm Dependent

Technology Choice

- C instead of C++:
  - Size
  - Portability
- Build system:
  - Mixture of autoconf, Microsoft, and Symbian SDK
  - Many more request like Borland, Xcode etc.

Footprint (linux, i386)

`size -t:`

	.text	.data	.bss
libpjlib	58 KB	0.1 KB	0.7 KB
libpjlib-util	64 KB	~0 KB	~0 KB
libpjnath	90 KB	0.9 KB	~0 KB
libpjmedia	213 KB	0.5 KB	3 KB
libpjmedia-codec	29 KB	0.3 KB	0.3 KB
libpjsip	165 KB	1.6 KB	13 KB
libpjsip-ua	44 KB	0.2 KB	~0 KB
libpjsip-simple	35 KB	0.5 KB	~0 KB
libpjsua	96 KB	0.3 KB	0.1 KB
TOTAL	794 KB	4.4 KB	17.1 KB

Notes on Footprint

- Done with "size -t" on the static libraries:
  - this measures ALL components in libraries
  - with static linking, only part of the libraries that are used by application are included in the executable
- Lots of features
- Various settings to reduce footprint by up to 129 KB (http://trac.pjsip.org/repos/wiki/FAQ#footprint)
- Heap usage: around 150 KB for application with two active calls

Case Study: IP Phone

- Linux on ARM9

  - familiarity of developers

  - ability to meet feature demands

Case Study: IP Phone

- Third party frameworks criteria

    1. Functionality

    2. Cost

    3. Support

Case Study: IP Phone

- PJSIP deployment

  - Optimization on ARM platform

    - codecs

    - resampling filters

General Porting Challenges

- Compiler differences and quirks
  - e.g. inline modifiers, 64bit integer, warnings
- Endianness
  - network protocols in network byte order, WAV files are little endian
- OS abstraction
  - pick lowest common denominator, e.g. Win32 API SuspendThread()/ResumeThread() are not available in pthread
- Sound device abstraction
  - oss, alsa, jack, wmme, ds, wasapi, wdmks, asio, coreaudio, symbian streaming, aps, nds, iphone...
- Floating-point support
  - fixed-point alternative algorithms

Even More Porting Challenges

- Power consumption
  - avoid polling --> Symbian mobiles
- No heap memory management!
  - no malloc()?! --> use pjlib pool
- No OS!
  - deeply embedded systems, e.g. TI
- Everything is 16bit!
  - In TI DSP, sizeof(char) == sizeof(short) == sizeof(int) == 2 (!)
  - SIP is text based protocol!
  - Translator between network and application
- Broken compiler
  - Zilog z80 compiler doesn't support nested struct declaration

Roadmap

- Even faster for developers to pickup

- More interoperability testing

- Mobile

Thanks

- Any questions

- Please visit http://www.pjsip.org

- Join our mailing list or leave blog comments

Teluu - Communicate Everywhere

- The company behind pjsip project

- Provides
  - Flexible licensing
  - Professional support
  - Certified and optimised binaries
  - Network of third party consultants

http://www.teluu.com