Building Applications using reSIProcate

Jason Fischl
Jason@purplecomm.com
• VOCAL / Vovida.org
• http://www.resiprocate.org
• Founded Summer 2002
• Part of SIPfoundry http://sipfoundry.org
• Informal organization
• Attended SiPit 12, 13 and 14
• Attending SiPit 15 in Taipei
• Part I
  – What is reSIProcate?
  – Who should use it?

• Part II
  – Programming with reSIProcate
  – Examples
  – Introduction to DialogUsageManager (DUM)
• Standards Compliant
• Easy to use
• Efficient: > 1000 tps
• Small footprint ( < 1 MB)
• Highly portable
• Microsoft Windows support
• Implement SIP Security
• Phones (e.g. embedded)
• Softphones (any platform)
• Gateways
• Proxies
• B2BUAs
• IM / Presence Servers or Clients
Current Features

- UDP, TCP, TLS
- Object-oriented interface
- Lazy Parser
- 3261 compliant
- Asynchronous DNS
- Single or multi-threaded
- DNS support (NAPTR/SRV)
- IPv6

- Multi-homed hosts
- Platforms: Windows, Linux, BSD, Solaris, QNX, OS/X
- MIME & S/MIME
- Extendable contents
- User Agent Library (DUM)
• Supported RFCs:
  – 2327: SDP
  – 2617: HTTP Digest
  – 2782: DNS SRV
  – 2915: DNS NAPTR
  – 2976: sip INFO
  – 3261: sip
  – 3263: Locating sip servers
  – 3265: subscribe/notify
  – 3420: sipfrag
  – 3325: network asserted id
  – 3428: sip MESSAGE
  – 3326: reason header
  – 3515: sip REFER
  – 3581: symmetric response

• In Development
  – 3323: Privacy mechanism
  – 3262: Reliable provisional
  – 3264: Offer/Answer
  – 3266: IPv6 in SDP
  – 3311: UPDATE
  – draft-simple-winfo
  – draft-sip-session-timer
  – draft-sip-caller-prefs

• Drafts: (partial list)
  – draft-impp-srv, draft-simple-presence
  – draft-sip-replaces, draft-sip-referredby
  – Draft-sip-connect-reuse
  – draft-sip-join, sip-gruu
  – sip-publish, sipping-mwi
• User Agent Library
• GAIM plug-in
• Documentation
• Presence Server
• SCTP
• Support for large numbers of TCP connections
• Performance improvements
• Footprint reduction
• Dialog Usage Manager (DUM)
- Windows
- Linux
- MAC
- QNX
- Others
• The Vovida Software License, Version 1.0
• BSD-like
• Free
• Can use it in commercial products
• Re-contribution not required
• PurpleComm (http://www.purplecomm.com)
  – Proxy, Registrar, Voicemail, Presence Server
  – Windows Softphone (http://teltel.com)
• Jasomi Networks (http://www.jasomi.com)
  – PeerPoint – Session Border Controller
• Xten Networks (http://www.xten.com)
  – eyeBeam
• CSP – Italy (http://www.csp.it)
  – IM/Audio/Video UA for Windows
  – Conference Server and h.323 gateway in development
• Computer Talk Technology (http://www.computer-talk.com)
  – Evaluating use of resiprocate for Contact Center product
• Adding Transports
  – TCP, UDP, TLS
  – Custom Transports
  – IPv6
  – Bind to specific interfaces (e.g. eth2)
• Use Helper class to help construct messages

```cpp
#include "resiprocate/SipStack.hxx"
#include "resiprocate/Helper.hxx"
#include "resiprocate/SipMessage.hxx"
#include "resiprocate/NameAddr.hxx"
using namespace resip;
...
SipStack stack;
stack.addTransport(TCP, 5060);
NameAddr from("<sip:jason@teltel.com>");
auto_ptr<SipMessage> reg(Helper::makeRegister(from, from);
stack.send(*reg);
...
```
Let the stack manage its own thread and select call

```c
SipStack stack;
StackThread stackThread(stack);
stackThread.run();

While(usleep(10)) {
    SipMessage* msg = stack.receive();
    if (msg) {
        // process the incoming sip message
        delete msg;
    }
}
```
• Single Threaded Mode
  – One thread for the entire application
  – Manage file descriptors at the app level

• Multi-Threaded Mode
  – Optional thread per transport
  – Optional StackThread
  – DUM + DUM client must be one thread or provide sync
• http://www.sipfoundry.org/reSIProcate/using.txt

SipMessage msg;
NameAddr to;
to.uri().user() = “jason”;
to.uri().host() = “teltel.com”;
msg.header(h_To) = to;
msg.header(h_To).displayName() = “Jason Fischl”;
assert(msg.exists(h_To));
msg.header(h_CSeq).sequence()++;

NameAddrs routes = msg.header(h_Routes);
NameAddrs reversed = routes.reverse();
for (NameAddrs::iterator i=routes.begin();
    i != routes.end(); ++i)
{
    std::cerr << *i << endl;
}
NameAddr firstRoute = routes.back();
routes.pop_back();
<table>
<thead>
<tr>
<th>RFC name</th>
<th>Access Token</th>
<th>resip Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept</td>
<td>h_Accepts</td>
<td>Mimes</td>
</tr>
<tr>
<td>Accept-Encoding</td>
<td>h_AcceptEncodings</td>
<td>Tokens</td>
</tr>
<tr>
<td>Accept-Language</td>
<td>h_AcceptLanguages</td>
<td>Tokens</td>
</tr>
<tr>
<td>Alert-Info</td>
<td>h_AlertInfos</td>
<td>GenericUris</td>
</tr>
<tr>
<td>Allow</td>
<td>h_Accells</td>
<td>Tokens</td>
</tr>
<tr>
<td>Authentication-Info</td>
<td>h_AuthenticationInfos</td>
<td>Auths</td>
</tr>
<tr>
<td>Authorization</td>
<td>h_Authorizations</td>
<td>Auths</td>
</tr>
<tr>
<td>Call-ID</td>
<td>h_CallID</td>
<td>CallID</td>
</tr>
<tr>
<td>Call-Info</td>
<td>h_CallInfos</td>
<td>GenericUris</td>
</tr>
<tr>
<td>Contact</td>
<td>h_Contacts</td>
<td>NameAddr</td>
</tr>
<tr>
<td>Content-Disposition</td>
<td>h_ContentDisposition</td>
<td>Token</td>
</tr>
<tr>
<td>Content-Encoding</td>
<td>h_ContentEncoding</td>
<td>Token</td>
</tr>
<tr>
<td>Content-Language</td>
<td>h_ContentLanguages</td>
<td>Tokens</td>
</tr>
<tr>
<td>Content-Length</td>
<td>h_ContentLength</td>
<td>IntegerCategory</td>
</tr>
<tr>
<td>Content-Type</td>
<td>h_ContentType</td>
<td>Mime</td>
</tr>
<tr>
<td>Content-Transfer-Encoding</td>
<td>h_ContentTransferEncoding</td>
<td>StringCategory</td>
</tr>
<tr>
<td>CSeq</td>
<td>h_CSeq</td>
<td>CSeqCategory</td>
</tr>
<tr>
<td>Date</td>
<td>h_Date</td>
<td>DateCategory</td>
</tr>
<tr>
<td>Error-Info</td>
<td>h_ErrorInfos</td>
<td>GenericUris</td>
</tr>
<tr>
<td>Expires</td>
<td>h_Expires</td>
<td>IntegerCategory</td>
</tr>
<tr>
<td>From</td>
<td>h_From</td>
<td>NameAddr</td>
</tr>
<tr>
<td>In-ReplyTo</td>
<td>h_InReplyTo</td>
<td>CallID</td>
</tr>
</tbody>
</table>
SipMessage request;
assert(request.header(h_To).exists(p_tag)); // exists
request.header(h_To).param(p_tag) = "jason"; // assign
request.header(h_To).remove(p_tag); // remove
request.header(h_To).uri().param(p_q) = 1.0; // FloatParam
request.header(h_Vias).front().param(p_branch).getTransactionId();

Auth auth;
auth.scheme() = "Digest";
auth.param(p_nonce) = "blah"; // QuotedDataParam
auth.param(p_algorithm) = "MD5";

UnknownParameterType p_myparam("myparam");
request.header(h_RequestLine).param(p_myparam) = "myvalue";
• Flexible, extensible Content management
• Multi-part MIME

SipMessage invite;
const SdpContents* offer =
dynamic_cast<const SdpContents*>(invite.getContents);
if (offer) {
    SipMessage ok; // make from invite request
    SdpContents answer; // make from the offer
    ok.setContents(&answer);
    delete offer;
}
• 400 lines of C++
• Look in sip/stateless-proxy for source code
• Good example of header/parameter manipulation
• Uses the transaction layer only
• Runs as a thread
• Also look at sip/presSvr for presence server (SIMPLE)
• Soon to be contributed: load generator using DUM
• Compilers
  – g++, VS.NET, VC++ 6
  – Intel compiler

• Build environment
  – Windows
  – Linux/Unix
  – QNX
  – MAC OS X
  – Solaris

• Namespaces
  – All code in resip namespace

• Exceptions
  – BaseException
  – Must catch

• Assertions
  – Can be compiled out
#include "resiprocate/Logger.hxx"
#define RESIPROCATE_SUBSYSTEM resip::Subsystem::TEST
using namespace resip;

...  
// can use Log::SYSLOG, Log::COUT or Log::FILE
Log::initialize(Log::COUT, Log::toLevel("INFO"), argv[0]);
SipMessage msg;
InfoLog ("The message is " << msg.brief());
DebugLog ("detail: " << msg);
Log::setLevel(Log::Debug);
• By default, Contact and Via IP address, port and transport will be filled in by the TransportSelector
• Application can specify it and avoid stack population
• E.g.

```cpp
SipMessage request;
...
request.header(h_Contacts).clear();
NameAddr contact; // default nothing filled in
request.header(h_Contacts).push_back(contact);
stack.send(request);
```
• Example code

SipMessage request;

... 

Uri outboundProxy;

outboundProxy.host() = "10.1.1.1";

outboundProxy.port() = 9999;

Request.sendTo(request, outboundProxy);
• TLS
• S/MIME
• Specifying transport interfaces
• TCP connection reuse
• Introduction
  – DUM makes writing user agents easy
  – Hides the complex sip specific stuff
• Usages
  – ClientRegistration, InviteSession, ClientPublication, etc.
• Handlers
  – InviteSessionHandler, ClientRegistrationHandler
• What can you do with DUM?
  – Softphone
  – B2BUA
  – Load generator
• Manage refreshes for you
  – Just set up registration binding and keep it active
  – Set up a subscription and keep it active. DUM will let you know when new updates come in via NOTIFY requests
  – Keep an INVITE session active
  – Handles forking for you
  – Merged requests
  – GRUU
• Implement Offer/Answer (rfc 3264)
• Implements PRACK (rfc 3262)
• Manage Profile information such as authorization
• Implements UAC and UAS
• **Companies:**
  - Jasomi
  - TelTel
  - Xten
  - CSP
  - Talk

• **Organizations**
  - Vovida.org
  - SIPfoundry

• **Contributors:**
  - Adam Roach, adam@dynamicsoft.com
  - Alan Hawrylyshen, alan@jasomi.com
  - Bob Bramwell bob@jasomi.com
  - Bryan Ogawa bko@cisco.com
  - Cullen Jennings, fluffy@cisco.com
  - David Bryan, dave@bryan.cx
  - David Butcher, david@purplecomm.com
  - Derek MacDonald, derek@xten.com
  - Jacob Butcher, resip@nowhen.com
  - Jason Fischl, jason@purplecomm.com
  - Ken Kittlitz <ken@jasomi.com>
  - Kevin Chmilar, kevin@jasomi.com
  - Robert Sparks, rjsparks@nostrum.com
  - Rohan Mahy, rmahy@cisco.com
  - Ryan Kereliuk, ryker@ryker.org
  - Scott Godin, slgodin@icescape.com
• Subscribe to mailing list
• Get subversion access
• [http://www.resiprocate.org](http://www.resiprocate.org)
• Jason Fischl
  – jason@purplecomm.com
  – http://www.sipfoundry.org
  – http://www.resiprocate.org
Building Applications using reSIProcate

Jason Fischl
Jason@purplecomm.com