NKSIP

“THE ERLANG SIP APPLICATION SERVER”

Why another SIP product?

- Existing products are either too complex to develop demanding SIP solutions (Kamailio, Mobicents) or too high-level to fully use all SIP's potential (Freeswitch, Asterisk).
- SIP is a very complex protocol, but most of this complexity has no added-value and should be hidden from the developer. On the other hand, SIP concepts are actually easy. No product clearly maps SIP concepts to the application.
- Implementing new RFCs is currently a painful process.
- SIP routing is at the very heart of any multimedia solution. Wouldn't it be nice to have full control of it?

NkSIP's vision

- An open source tool to develop any kind of SIP application, specially focused on highly available, distributed and scalable SIP applications.
- Each SIP application decides what callback functions to implement. Very few for a simple, standard behavior. Many of them for fine-tuned or low level behaviors. *If it is possible with SIP, it is possible with NkSIP*. Implement only what you need. Everything else is handled by NkSIP automatically, using sane defaults.
- All SIP applications are similar from NkSIP's point of view: UACs, UASs, proxy servers, B2BUAs, SBCs… why a different box for each one? You can write any SIP application (a softphone, a heavy load tester, a SBC, a stateless proxy…) but NkSIP already offers you common behaviors ready to use: authentication, proxies, dialogs, sessions, B2BUAs.
- NkSIP does NOT try to emulate a PBX again. SIP is much more than that. No dial plans. No extensions. Only pure SIP. You need to now what INVITE and BYE means to use NkSIP, but only a few details about transactions, ACKs, or generating the correct headers in every case. NkSIP takes care of the details. (But of course it's very easy to develop an old good PBX using NkSIP).
- Implementing new RFCs should be easy and natural. NkSIP offers a rock solid core, written from scratch to develop distributed, highly available and scalable systems. It will soon have a plug-in system to develop new functionality without having to modify the core.
- The Erlang ecosystem is the perfect fit for this kind of product. You can run any number of SIP applications on a NkSIP cluster, using Erlang but (soon) also using Javascript, Lua, Java, or any other language capable of receiving a JSON over a socket connection.

NkSIP's architecture

- A cluster made of any number of nodes, each one running the NkSIP application. Any number of user SIP applications are installed on all of them.
- When a SIP message arrives, it is sent to one specific node. All messages having the same Call-ID go to the same node.
- NkSIP starts calling callbacks defined in the corresponding SIP application. If they are not defined, a default behavior is used.
- NkSIP detects and manage transactions, dialogs and sessions. If not defined in the application response, it automatically generates Contact and any other mandatory or recommended header.
- The application can also initiate outgoing requests at any moment. NkSIP will again help it adding any necessary header or tag.
- If the node goes down, the transaction is lost, but another one resumes the dialog and sessions. The call is not affected.
- You can add or remove nodes at any time.

CURRENT STATUS

JUL 2014

- First version published on GitHub on July 2013, after about 12 months of private work.
- Full rewrite 3 months later to be more simple and scalable. Two more releases (0.2 and 0.3) on October and November 2013.
- Version 0.4 released on July 2014, with lots of new stuff, comprehensive documentation and the plugins mechanism.
- 0.5 will be the first distributed version.
- NkSIP has currently 105 *stars* and 33 *forks* in GitHub. +100 weekly unique visitors. 6 contributors.

Current **features**

- Full RFC3261 coverage, including SIP Registrar (using the RAM built-in store or any other external database).
- Robust and highly scalable, using all available processor cores. Currently is able to process 10.000 registrations/sec or 4.000 call setups/sec (INVITE+ACK+BYE) on a 4-core i7 machine.
- Full support for all defined SIP methods: PRACK, INFO, UPDATE, SUBSCRIBE, NOTIFY, REFER, PUBLISH and MESSAGE, as UAC, UAS and Proxy.
- Full support for reliable provisional responses (100rel), Session Timers, Path, Service Route, Outbound and GRUU.
- A written from scratch, fully typed Erlang code easy to understand and extend. Unit tests cover nearly all of the functionality. Few external dependencies. Hot, on the fly core and application code upgrade.
- UDP, TCP, TLS , SCTP and WEBSOCKETS transports, capable of handling thousands of simultaneous sessions. Full RFC4475 and RFC5518 Torture Tests passing.
- Stateful proxy servers with serial and parallel forking. Stateless proxy servers, even using TCP/TLS.
- Full IPv6 support. NkSIP can connect IPv4-only with IPv6-only hosts.
- Full support for NAPTR and SRV location, including priority and weights.
- Automatic outbound registrations and timed pings. Dialog and SDP media start and stop detection. SDP processing utilities. Powerful event support. Event State Compositor implementation.

RFC	Description	Notes
RFC2617	Digest authentication	
RFC2782	DNS SRV	
RFC2915	DNS NAPTR	
RFC2976	INFO	
RFC3261	SIP 2.0	
RFC3262	Reliable provisional responses	
RFC3263	Locating SIP Services	
RFC3264	Offer/Answer Model	
RFC3265	Event Notification	
RFC3311	UPDATE	
RFC3326	Reason	
RFC3327	Registering Non-Adjacent Contacts	path
RFC3428	MESSAGE	
RFC3515	REFER	
RFC3581	RPort	
RFC3608	Service-Route	
RFC3903	PUBLISH	
RFC4168	SCTP Transport	No TLS-SCTP
RFC4475	Torture Tests	Included in unit tests
RFC4566	SDP	Only parser and generator
RFC5057	Multiple Dialogs	
RFC5118	IPv6 Torture Tests	Included in unit tests
RFC5389	STUN	Basic STUN client and server (no IPv6)
RFC5626	Outbound	
RFC6026	2xx responses	
RFC6157	IPv6 Transition	
RFC6665	Event Notification	Obsoletes 3265. GRUU support

ROADMAP

JUL 2014

v0.5: **NkCore**

- *Extraction* of all non-SIP stuff from NkSIP, into a new open source product: NkCore (configuration, database, logging, transports…)
- NkCore will be distributed from the start, based on Riak Core.
- NkSIP will be a module on top of NkCore. v0.5 will be the first *distributed* version of NkSIP.
- Powerful plugin mechanism, with virtually no overhead.

v0.6: **Non-Erlang apps**

- *NkCore* will allow to write apps using other languages.
- For javascript, a *node.js* instance is started in each node for each application. Each callback function can be implemented in javascript, using NkCore's javascript client library.
- Possibility to use other languages: Lua, Java, Python, Ruby.
- Two ways to develop logic on top of NkCore:
  - Application: external to NkSIP, easy, any language.
  - Plug-in: internal, must use Erlang, full power. Interesting examples: new RFCs, event support packages, etc.

NKCORE

“THE ERLANG DISTRIBUTED APPLICATION SERVER”

Another application server?

- NkSIP offers a fresh approach to the development of applications, but currently is only focused on SIP.

- Many concepts and technology solutions from NkSIP are applicable to many other scenarios: Diameter, REST, MSRP, XMPP, DNS, Mail…

- There is no easy to use, fully distributed, open source application server out there.

NkCore's vision

- You can start any number of nodes, and install any number of user applications on all of them.
- Incoming messages (in a *broad* sense: SIP, REST request…) are sent to a specific node and triggers the call of specific application callbacks. Applications are written using Javascript, Lua, Java, etc.
- Many *modules* are available, offering tools and semi automatic processing of messages: SIP (NkSIP), Web, Websockets, Database access, Diameter, DNS, etc. Each one adds new callbacks and APIs available to the application. Some of them offer additional optional plugins (i.e. event support packages for SIP).
- Many common functionality is included: configuration, cluster management, in-memory simple database, dns, logging, authentication, roles, etc.

Expected **features**

- Hot code loading (NkCore itself and modules and user applications).
- Allows applications to become highly available and scalable very easily.
- Embedded distributed database. Web server. All transports (tcp/tls, udp, sctp, ws/wss). Roles. External database abstractions.
- Hybrid-cloud model. Multi-cluster management.
- Launching and control of external processes: Freeswitch, DNS server, Riak database…
- The only tool needed for many critical applications: emergency call centers, message-centric applications, single-page HTML5 applications, mobile.