Jan Janák, iptel.org
Outline

1. Introduction
   - About
   - History
   - Present

2. SER Overview
   - Features
   - Development Model
   - Architecture

3. Typical SIP Server Setup
   - Basic Setup
   - Scaling Up
   - High Availability

4. Summary
About iptel.org

- Started as a website focused on SIP and maintained by VoIP interest group in FhG FOKUS
- Established in 1999
- SIP Tutorial by Dorgham Sisalem and Jiri Kuthan
- Freely available SIP service based on 3rd party software
- Began developing SIP software for the SIP service in 2001, known under the name SIP Express Router
- Complemented later by SERWeb, RTPProxy, SEMS, MySTUN
- iptelorg GmbH spin-off founded by FhG FOKUS to provide commercial SER support, acquired by Tekelec in 2005.
iptel.org continues as non-profit site sponsored by FhG FOKUS

**Goals**

- Promote VoIP based on open standards (SIP, RTP).
- Promote use of open source VoIP software.
- Provide freely available reference SIP service.
- Maintain website with documentation and tutorials on SIP based technology.
- Develop open source SIP server software.
• First working SER version committed by Andrei Pelinescu-Onciul on 4th September 2001
• Originally intended as simple routing engine for Cisco PSTN gw
• Two weeks later initial version of the configuration language was done
• First SER modules followed in beginning of 2002, mysql driver, registrar, authentication, and record routing
• Attending first SIPIt in 2002 with SER running on an PDA
• May 2002: IPv6 support
• August 2002: Sipsak was created by Nils Ohlmeier
• September 2002: First public version and also SERWeb was born
• December 2002: First external contribution, ENUM support by Juha Heinanen
• January 2003: First SEMS version by Raphael Coeffic
Commercial Deployments

- Tens of thousands installations world-wide estimated
- One SER fork: http://www.openser.org
- Used by many SIP vendors as de-facto reference implementation
- Powering some of Largest SIP Setups:
  - EarthLink
  - FWD
  - sipgate
  - SIP PHONE
  - TELIO
  - T-Online

- Setups with 80k subscribers on single host exist
Used in Internet2 SIP.edu VoIP infrastructure
SIP.edu actively contributed to SER
SER got presence support from SIP.edu

Colorado State University
UCLA
University of Alaska
ETH Zurich
Columbia University
Yale University
MIT
Embedded Setups

Siemens Gigaset DD-WRT

Milkfish

http://www.milkfish.org
What is SER

- SIP Proxy Server
- Registrar
- Redirect server
- SIMPLE based presence server
- Transaction stateful
What is SER **NOT**

- Back-to-back User Agent
- Dialog Stateful
- PBX (Packet Branch Exchange)
- Media server
- PSTN Gateway
List of Features

- Written in ANSI C and optimized for speed
- Modular design
- Flexible configuration and routing language
- Supports MySQL, Postgres, LDAP, RADIUS
- Standard (RFC3261) compliant
- Web based administration interface
- NAT traversal capable
- Portable, runs on POSIX compliant systems
Development Model

- Licensed under GPL
- Currently about 20 developers
- For core and core modules FhG FOKUS is exclusive (c) owner
- Anyone can freely contribute extensions and modules
- Changes to existing modules are subject to approval
**Core Provides**
- Transport management
- Memory management
- Module interface
- Essential functionality

**Modules Provide**
- Script functions
- Module parameters
- Special variables
- Management functions
**Configuration File**

```
debug=3
echo = yes
log_stderr=no

loadmodule "/usr/lib/ser/modules/sl.so"
loadmodule "/usr/lib/ser/modules/tm.so"
loadmodule "/usr/lib/ser/modules/rr.so"
loadmodule "/usr/lib/ser/modules/usrloc.so"
loadmodule "/usr/lib/ser/modules/registrar.so"

modparam("usrloc", "db_mode", 0)
modparam("rr", "enable_full_rr", yes)

route {
    if (!method == "REGISTER") record_route();
    if (loose_route()) {
        t_relay();
        break;
    }
    if (uri == myself) {
        if (method == "REGISTER") {
            save("location");
            break;
        }
        if (!lookup("location")) {
            sl_reply(404, "Not Found");
            break;
        }
    }
    t_relay();
```

**Server Configuration**

- Inspired by perl and C, tells SER what to do with SIP messages
- Generic server settings
- Modules to load
- Module configuration
- SIP message processing
Configuration File

debug=3
fork = yes
log_stderr=no

loadmodule "/usr/lib/ser/modules/sl.so"
loadmodule "/usr/lib/ser/modules/tm.so"
loadmodule "/usr/lib/ser/modules/rr.so"
loadmodule "/usr/lib/ser/modules/usrloc.so"
loadmodule "/usr/lib/ser/modules/registrar.so"

modparam("usrloc", "db_mode", 0)
modparam("rr", "enable_full_rr", yes)

route {
    if (!method == "REGISTER") record_route();
    if (loose_route()) {
        t_reply();
        break;
    }
    if (uri == myself) {
        if (method == "REGISTER") {
            save("location");
            break;
        }
        if (!lookup("location")) {
            sl_reply("404", "Not Found");
            break;
        }
        t_reply();
    }
}
Configuration File

debug=3
fork = yes
log_stderr=no

loadmodule "/usr/lib/ser/modules/sl.so"
loadmodule "/usr/lib/ser/modules/tm.so"
loadmodule "/usr/lib/ser/modules/rr.so"
loadmodule "/usr/lib/ser/modules/usrloc.so"
loadmodule "/usr/lib/ser/modules/registrar.so"

modparam("usrloc", "db_mode", 0)
modparam("rr", "enable_full_rr", yes)

route {
    if (!method == "REGISTER") record_route();
    if (loose_route()) {
        t_reply();
        break;
    }
    if (uri == myself) {
        if (method == "REGISTER") {
            save("location");
            break;
        }
        if (!lookup("location")) {
            sl_reply("404", "Not Found");
            break;
        }
        t_reply();
    }
}
SER Overview

Typical SIP Server Setup

Summary

Features

Development Model

Architecture

Configuration File

debug=3
fork = yes
log_stderr=no

loadmodule "/usr/lib/ser/modules/sl.so"
loadmodule "/usr/lib/ser/modules/tm.so"
loadmodule "/usr/lib/ser/modules/rr.so"
loadmodule "/usr/lib/ser/modules/usrloc.so"
loadmodule "/usr/lib/ser/modules/registrar.so"

modparam("usrloc", "db_mode", 0)
modparam("rr", "enable_full_rr", yes)

route {
    if (!method == "REGISTER") record_route();
    if (loose_route()) {
        t_relay();
        break;
    }
    if (uri == myself) {
        if (method == "REGISTER") {
            save("location");
            break;
        }
        if (!lookup("location")) {
            sl_reply("404", "Not Found");
            break;
        }
    }
    t_relay();

Server Configuration

- Inspired by perl and C, tells SER what to do with SIP messages
- Generic server settings
- Modules to load
- Module configuration
- SIP message processing
Configuration File

debug=3
fork = yes
log_stderr=no

loadmodule "/usr/lib/ser/modules/sl.so"
loadmodule "/usr/lib/ser/modules/tm.so"
loadmodule "/usr/lib/ser/modules/rr.so"
loadmodule "/usr/lib/ser/modules/usrloc.so"
loadmodule "/usr/lib/ser/modules/registrar.so"

modparam("usrloc", "db_mode", 0)
modparam("rr", "enable_full_rr", yes)

route {
    if (!method == "REGISTER") record_route();
    if (loose_route()) {
        t_relay();
        break;
    }
    if (uri == myself) {
        if (method == "REGISTER") {
            save("location");
            break;
        }
        if (!lookup("location")) {
            sl_reply("404", "Not Found");
            break;
        }
        t_relay();
    }
Overview of Operation

Server Processing

- SER is simple message forwarder
- SIP request arrives, execute route block
- Forwarding request 1st time, execute branch_route block
- Forwarding request 2nd time, execute branch_route block again
Overview of Operation

Server Processing

- SER is simple message forwarder
- SIP request arrives, execute route block
  - Forwarding request 1st time, execute `branch_route` block
  - Forwarding request 2nd time, execute `branch_route` block again
Overview of Operation

Server Processing

- SER is simple message forwarder
- SIP request arrives, execute route block
- Forwarding request 1st time, execute branch_route block
- Forwarding request 2nd time, execute branch_route block again

SER is simple message forwarder. When a SIP request arrives, execute the route block. When the request is forwarded for the first time, execute the branch_route block. If the request is forwarded again, execute the branch_route block again.
Overview of Operation

**Server Processing**

- SER is simple message forwarder
- SIP request arrives, execute `route` block
- Forwarding request 1st time, execute `branch_route` block
- Forwarding request 2nd time, execute `branch_route` block again
**Overview of Operation**

**Server Processing**
- Reply received, execute `onreply_route` block
- Reply received, execute `onreply_route` block again
- Reply forwarded upstream, execute `failure_route`
Overview of Operation

Server Processing
- Reply received, execute `onreply_route` block
- Reply received, execute `onreply_route` block again
- Reply forwarded upstream, execute `failure_route`
Overview of Operation

Server Processing

- Reply received, execute `onreply_route` block
- Reply received, execute `onreply_route` block again
- Reply forwarded upstream, execute `failure_route`
Adding More Boxes

- Necessary for 60k subscribers or more.
- SIP aware load balancer needed
- Each subscriber has a home proxy.
- Hashing based on From/Request-URI URI
- SER with dispatcher module can be used for this purpose.
- Provisioning applications can use the same load balancer to find out home proxy for subscriber.
Adding Yet More Boxes

- Master proxy sends heartbeat packets.
- Slave proxy is inactive while receiving heartbeat packets.
- Master proxy fails and stops sending heartbeat packets.
- Slave detects the failure and sends gratitious ARP responses to router.
- Gratitious ARP responses would map shared IP to slave.
- IP traffic to shared IP will be redirected to slave.
Summary

Highlights

- Works across NATs.
- No central back-end database required.
- Scalable up to millions of subscribers.
- Load balancing purely server side, no UA need to reconfigure UAs.
- No proprietary extensions, any RFC3261 conformant implementation works.
Thank You.