# SIP Express Media Server
# SEMS
# Workshop

KamailioWorld 2014

Stefan Sayer

VoIP Services Consulting and Development
email/xmpp:stefan.sayer@gmail.com
stefan.sayer@frafos.com

# Contents

- Intro: History, general info
- How to: install, use, provision, maintain
- Voice app programming
- SBC

# Contents

- Intro: History, general info
- How to: install, use, provision, maintain
- Voice app programming
- SBC

# History of SEMS

- Originates from the same team as SER (Kamailio/Openser/...) as Fraunhofer FOKUS, German public R&D institute
- Beginning: Answering machine add-on to SER
- Developed at various related companies (iptelorg, IPTEGO, ...)
- Since 2010 mainly at FRAFOS
- Open Source community since 2003

# FRAFOS, ABC SBC and SEMS

- FRAFOS Session Border Controller product ”ABC SBC” based on SEMS
- Much of FRAFOS' internal development being contributed to FOSS SEMS
- Best-effort support on mailing lists
- Sadly lately not much SEMS advertising, community efforts, website updates etc

# Sipwise sip:provider and SEMS

- SEMS as central B2BUA in sip:provider
- Several Sipwise-sponsored features/apps
  - e.g. PBX-type call flows, Mobile push app
- SPCE 100% open source
- sip:provider pro has closed source add-ons
  - e.g. replication

# F/OSS SEMS users

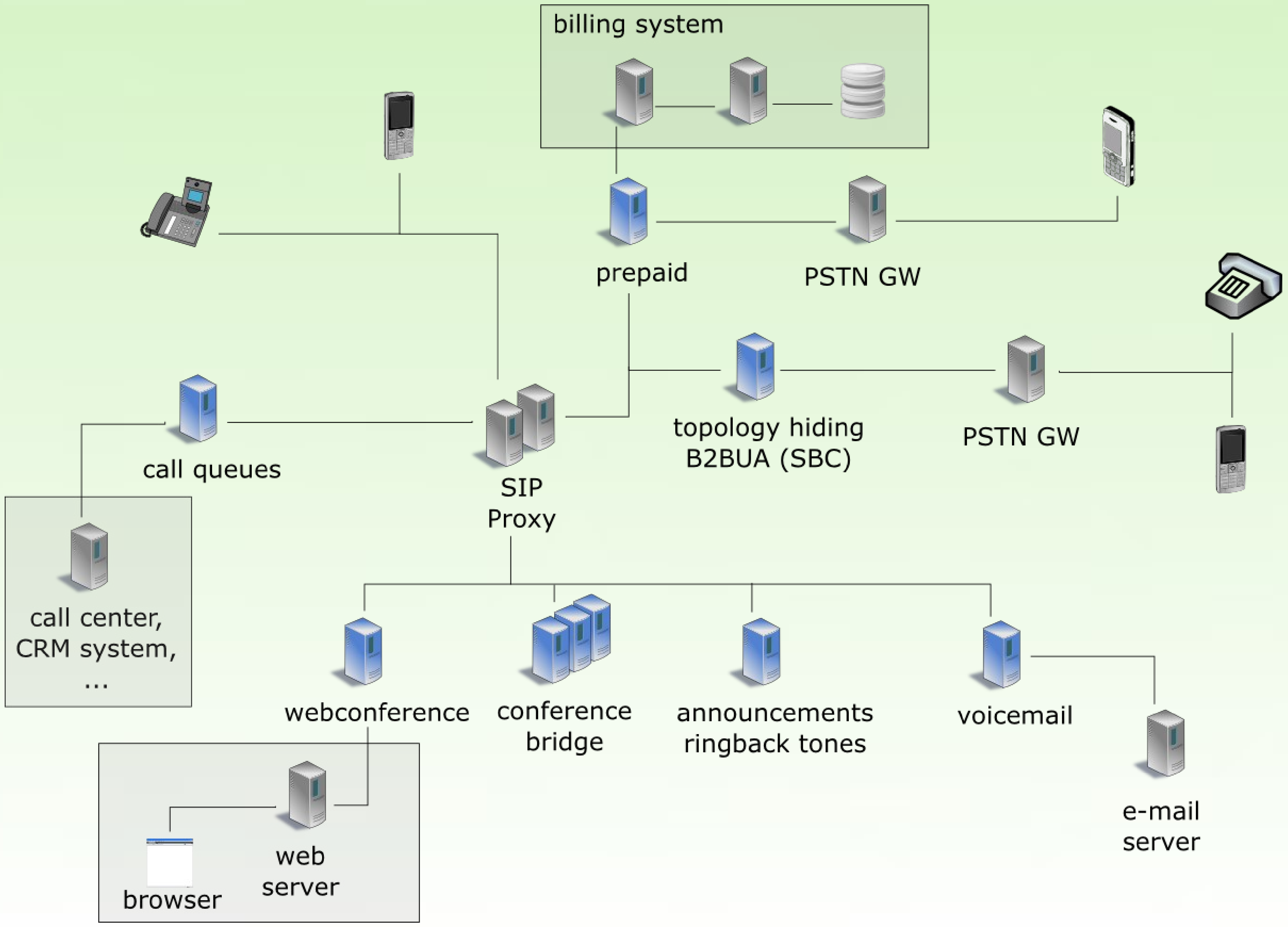
- Carriers
  - Deutsche Telekom
  - QSC
- VoIP service providers, telecoms services
  - TelTech, Millenicom, Gammatelecom, XConnect, …
- Universities
  - TU Berlin (D), FH Darmstadt (D), FELT (CZ), UPF (ES), ...
- Many more from FOSS community

# Focus

- Telecoms applications, carrier environment
  - High volume prompts, voicemail, conferencing, …
  - B2BUA / SBC
- Speed and reliability
- Only SIP, not multi-protocol (almost)
- Versatile and easy to use app server for SIP networks

- *Built for purpose*

# Classic use cases



# SEMS Use Cases

- Endpoint apps
  - Prompts; RBT, pre-call prompts etc
  - Conference; meet-me, with web GUI, in/outbound
  - Voicemail; voicemail2email, mailbox
- App development
  - C++ apps
  - DSM
- SBC / B2BUA

# Practical / Technical

- Language: C++, C, some Python, DSM
- Sloccount: ~90k C++, 15k C
- Runs on Linux, Mac, BSD, ...
- Preferred OS: debian, RHEL/CentOS/EPEL
- Make files and CMake build system
- Links: http://iptel.org/sems, git.sip-router.org/sems
- Support/dev ML: http://lists.iptel.org
- Core / plugin architecture
- Docs: make doc ; or see doc/Readme.*

# SEMS Versions

- SPCE uses 1.4
- 1.5 with lots more SBC features
- Git master (to be 1.6) pretty stable for a while, recommended especially for SBC
  - Performance, registrations, sub/not, features, call control API etc
- EPEL/OBS still mostly at 1.4ish, use from source recommended (as of Apr 2014)

# Contents

- Intro: History, general info
- How to: install, use, provision, maintain
- Voice app programming
- SBC

# Getting practical...Installation!

- On fresh debian wheezy (netinstall)
  - su - ; apt-get install sudo; usermod -a G sudo semsadmin
  - sudo apt-get install debhelper devscripts
  - sudo apt-get install g++ make libspandsp-dev flite-dev libspeex-dev \
    
    libssl-dev python-dev python-sip-dev openssl libev-dev \
    
    libmysql++-dev libevent-dev libxml2-dev libcurl4-openssl-dev
  - sudo apt-get install git
  - git clone git://git.sip-router.org/sems
- Debian package creation
  - cd sems ; ln -s pkg/debian . ; dch -b -v `git describe --always` "sems git master" ; dpkg-buildpackage -rfakeroot -us -uc
- Install: dpkg -i ../sems_x.y.z.deb
- This is the VM image contents (+ small cfg chg)

# Basic configuration

- /etc/sems/sems.conf
  - Interfaces (default: first system interface)
  - Loaded plugins: load_plugins=...
  - App: application=...
- Example – default: webconference
  - load_plugins=wav;isac;l16;speex;g722;gsm;ilbc;webconference
  - application=webconference
- Application config files in /etc/sems/etc
  - e.g. /etc/sems/etc/webconference.conf

# App selection & provisioning

```
application= ...
#    $(ruri.user)       - user part of ruri is taken as application,
#                       e.g. sip:announcement@host
#    $(ruri.param)      - uri parameter "app", e.g.
#                       sip:joe@host.net;app=announcement
#    $(apphdr)          - the value of the P-App-Name header is used
#
#    $(mapping)         - regex=>application mapping is read from
#                       app_mapping.conf (see app_mapping.conf)
#    <application name> - application name configured here, e.g.
#                       application=announcement
```

Provisioning through SIP message:

- DB/profile access only once, in the proxy
- Simpler system design

```
route[SERVICES] {
        if (uri=~"sip:100.*") {
                force_send_socket(192.168.1.88:5060);
                t_relay_to_udp("192.168.1.112","5060");
                exit;
        }

        if (uri=~"sip:101.*") {
                remove_hf("P-App-Name");
                append_hf("P-App-Name: echo\r\n");
                force_send_socket(192.168.1.88:5060);
                t_relay_to_udp("192.168.1.112","5060");
                exit;
        }

        if (uri=~"sip:102.*") {
                remove_hf("P-App-Name");
                append_hf("P-App-Name: annrecorder\r\n");
                remove_hf("P-App-Param");
                append_hf("P-App-Param: usr=myuser1;dom=mydomain;typ=vm;lng=de;\r\n");
                force_send_socket(192.168.1.88:5060);
                t_relay_to_udp("192.168.1.112","5060");
                exit;
        }
```

# Applications

| | | |
|---|---|---|
| prompts | announcement | Prompt (lang, domain, ...) |
| | announce_transfer | Prompt and continue B2BUA |
| | early_announce | Ring Back Tones (183) |
| conference | conference | Simple meet-me conf |
| | webconference | Meet-me + GUI (XMLRPC) |
| echo | echo | test |
| voicemail | voicemail | Leave message; vm2email or mailbox |
| | annrecorder | Record prompt |
| | msg_storage | Store msg on FS |
| | mailbox | IMAP mailbox app |
| | mwi | Message waiting indication |
| click2dial | click2dial | Click 2 dial from webpage |
| | di_dial | Dial out (any app) |
| | | |
| SBC | sbc | See next chapter |

# Components / interfacing

| | | |
|---|---|---|
| interfacing | xmlrpc2di | XMLRPC interface and control |
| | jsonrpc | Json-rpc (v2) |
| | stats | Status, interface (simple UDP commands) |
| | diameter_client | DIAMETER (base protocol) client |
| | | |
| | | |
| | monitoring | In-mem DB (KV-store), detailed calls status |
| | | |
| auth | uac_auth | SIP client auth |
| | | |
| registration | registrar_client | SIP client registration |
| | reg_agent | SIP registration, from config file |
| | db_reg_agent | SIP registration, from DB |
| | | |
| sst | session_timer | SIP session timer |

# Monitoring, logging, tools

- sems-stats -c ”set_loglevel 3”
- sems-list-calls, sems-get-callproperties etc

```
semsadmin@sems-vm:~$ /usr/sbin/sems-list-calls
Active calls: 1
['348AE601-533AC7BE000CC43A-85DDF700']
semsadmin@sems-vm:~$ /usr/sbin/sems-get-callproperties
 348AE601-533AC7BE000CC43A-85DDF700
Active calls: 1
[   {   'dir': 'in',
        'from': '"baeresip" <sip:baresip@192.168.5.110
>',
        'ruri': 'sip:101@192.168.5.110',
        'to': '<sip:101@192.168.5.110>'}]
semsadmin@sems-vm:~$
```

- sems-logfile-callextract

# Contents

- Intro: History, general info
- How to: install, use, provision, maintain
- Voice app programming
- SBC

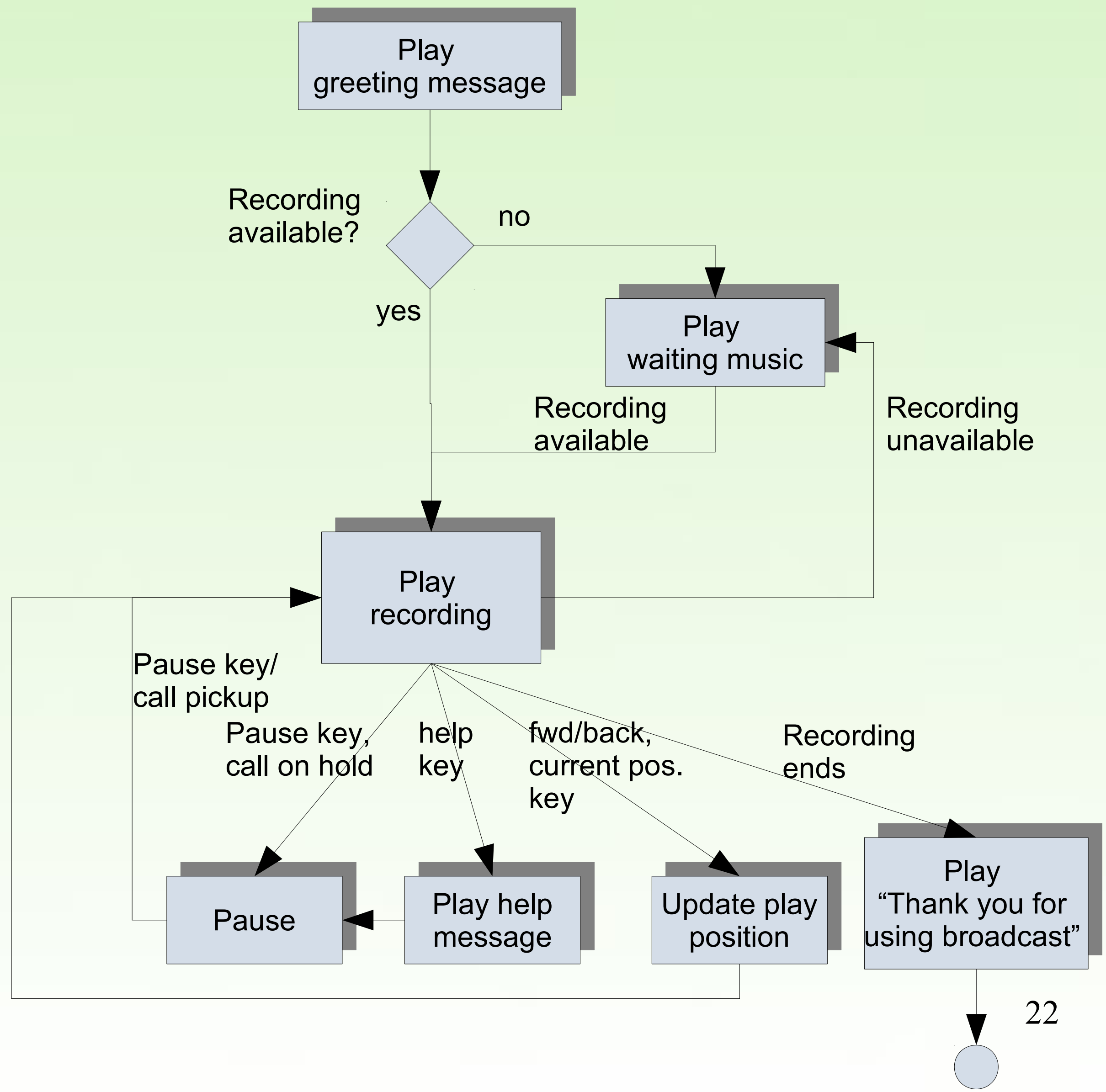
# Voice app programming

- C++ API
- Python API
- DSM
  - Simple + ~intuitive script language
  - Interpreter and environment in SEMS
  - Use for
    - Rapid Prototyping
    - Complex apps
    - Code structuring and re-use (modules)

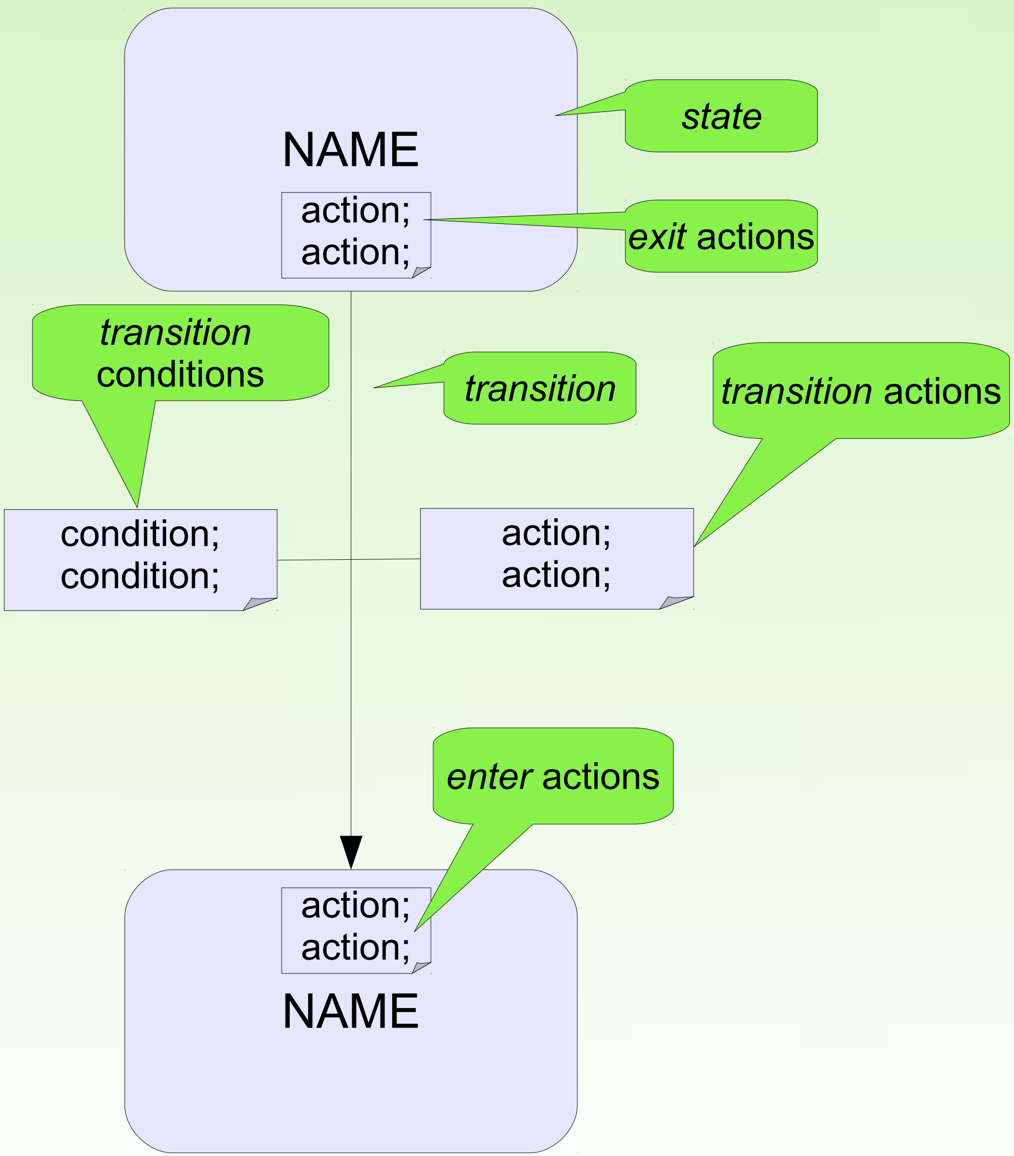
# Services: how to define

- State diagrams are the “natural” way to correctly define and document service logic
- Do actions and process events, while transitioning from state to state

Example:



# DSM: State machine notation



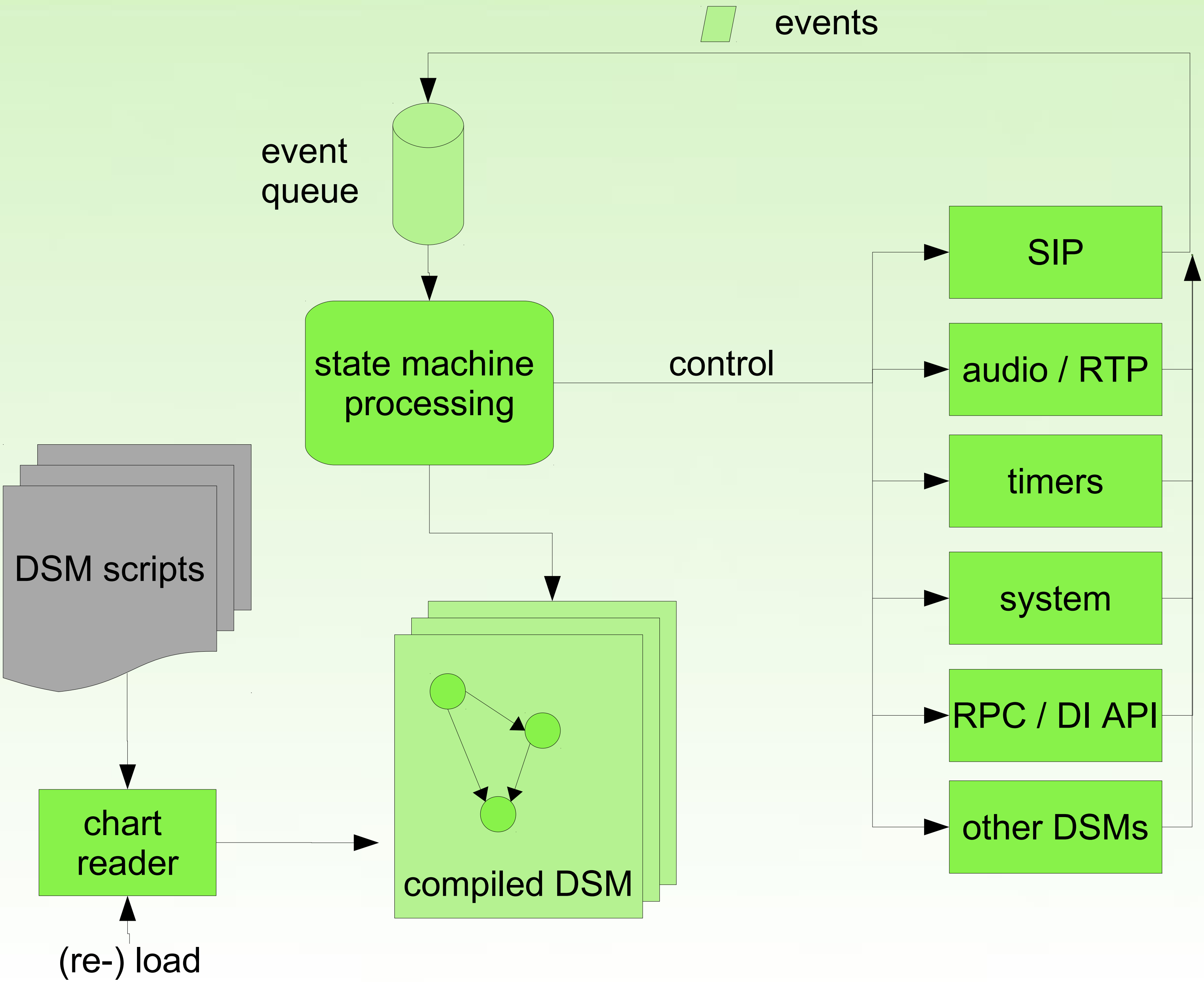

DSM state diagram definition language:

```
-- comment
import(mod_name);

[initial] state name
        [ enter {
            action;
            action;
            ...
          } ]
        [ exit {
            action;
            action;
            ...
          } ]
    ;

transition name s1 - [ { condition; condition; ... } ]
      [/ { action; action; ...} ]  -> s2;
```

# DSM service environment



# DSM: example

```
-- just a small demo
import(mod_mysql);
import(mod_utils);

initial state START
  enter {
     playFile(hello.wav);
     mysql.connect(mysql://user:pwd@domain/db);
     mysql.queryGetResult(select count(id) as new_messages from messages where userid=@user);
   };
transition “we have some messages” START – test($new_messages != 0) -> PLAY_MESSAGES;
transition “no messages” - test($new_messages == 0) / playFile(no_messages.wav) -> PLAY_AND_BYE;

state PLAY_AND_BYE;
transition “BYE received” PLAY_AND_BYE – hangup / stop(false) -> end;
transition “files finished” PLAY_AND_BYE – noAudioTest() / stop(true) -> end;

state PLAY_MESSAGES
    enter {
      playFile(you_have.wav);
      utils.playCountRight($new_messages);
      playFile(messages.wav);
    };
transition “key to listen to the messages”  PLAY_MESSAGES – keyPress(1) –> PLAY_MSG;
transition “key to main menu”  PLAY_MESSAGES – keyPress(2) –> MENU;
...
```

# DSM language: Identifiers

- Variables: $var
  - e.g. set($userid=”stefan”)
  - test(len($pin)==5)
- Event Parameters: #param
  - e.g. test(#key==1)
- “Selects”: @select
  - e.g.playFile(@user); @callid, @local_tag

# DSM: Control

- Event types
  - invite, key, timer, noAudio, event (generic), ...
- Hierarchical DSMs
  - jumpFSM(), callFSM(), returnFSM()
- repost()
  - mark event as not processed (transitional states)
- stop()
- exceptions (exception transitions)

# DSM: Functionality

- Core module: basic actions and conditions
  - playFile, setPromptSet, playPrompt, recordFile, set, append, log, setTimer, …
- Modules: more actions and conditions
  - import(mod_name)
  - mod.action();

# Trying it out: IVR app

VM: /usr/lib/sems/dsm/connect_extension.dsm  src: doc/dsm/examples/b2b_connect_extension

```
-- This small DSM collects the extension via key input and
-- connects to the extension in B2B mode.
--
-- Set run_invite_event=no in dsm.conf. Make necessary prompts by make in wav/
-- The domain part of the extension to call is set below.

initial state START
  enter {
     -- directory for prompts, may also be e.g. in $config.prompts_dir
     set($prompts_dir="/usr/lib/sems/audio/connect_extension/");

     -- the prompt we play at the beginning
     sets($prompt_name=$(prompts_dir)enter_extension.wav);
     playFile($prompt_name);

     -- 60 seconds to enter extension, if no
     setTimer(1, 60);

     -- evaluate directly transition, to go
     repost();
  };
```

```
-- state for collecting extension digits
state COLLECTING
  enter {
    -- set 10 seconds 'hint' timer
    setTimer(2, 10);
  };

-- digit key pressed
transition "key press" COLLECTING - key(#key<10) / {
    -- break any possibly playing prompt
    flushPlaylist();
    -- add key to extension
    append($extension, #key);
} -> COLLECTING;

transition "hint timer hit" COLLECTING - timer(#id==2) / {
     -- 'please enter the extension'
     sets($prompt_name=$(prompts_dir)enter_extension.wav);
     playFile($prompt_name);
} -> COLLECTING;
```

# IVR app: connecting B2B

- Example connects to sip:000777xyz@iptel.org

```
-- connect (# or *) pressed
transition "connect key press" COLLECTING - key(#key>9) / {
    -- break any possibly playing prompt
    flushPlaylist();

    -- 'now connecting'
    sets($prompt_name=$(prompts_dir)connecting.wav);
    playFile($prompt_name);

    -- prefixing with 000777 (iptel.org webconference direct dialin)
    set($remote="sip:000777");
    -- room
    append($remote, $extension);
    -- at domain
    append($remote, @);
    append($remote, "iptel.org");

    log(3,$remote);

    -- connect in B2BUA mode
    B2B.connectCallee($remote, $remote);
} -> CONNECTING;

state CONNECTING;
```

# DSM: Modules

| Module | Functionality |
|---|---|
| mod_sys | System commands (sys.mkdir, ...) |
| mod_dlg | Dialog related (dlg.bye, ...) |
| mod_uri | URI processing and operations |
| mod_conference | Conferencing functions (conference.join,...) |
| mod_utils | Utilities (utils.getNewId, utils.spell, ...) |
| mod_monitoring | Monitoring functions |
| mod_aws | Amazon AWS functions |
| mod_mysql | MySQL (mysq.query) |
| mod_py | Python (py(print 'hello world')) |

# DSM: Writing modules

- Implement actions and conditions
- Simplified by macros

```
DEF_ACTION_1P(SCMkDirAction);
…
DEF_CMD("sys.mkdir", SCMkDirAction);
…
EXEC_ACTION_START(SCMkDirAction) {
  string d = resolveVars(arg, sess, sc_sess, event_params);
  DBG("mkdir '%s'\n", d.c_str());
  if (sys_mkdir(d.c_str())) {
    sc_sess->SET_ERRNO(DSM_ERRNO_OK);
  } else {
    sc_sess->SET_ERRNO(DSM_ERRNO_FILE);
  }
} EXEC_ACTION_END;
```

# DSM benefits

- eases development
- enforces modularity, code reusability
- check service consistency (e.g. BYE handled)
- create service directly from specification
- reduce time to market
  - adapt services on the fly
  - speed up implementation/customization – test - product cycle

# Contents

- Intro: History, general info
- How to: install, use, provision, maintain
- Voice app programming
- SBC

# SBC application

- B2BUA, completely transparent to fully opaque
- Network separation
- SIP and (optional) RTP
- Mediation (SIP message, codecs etc)
- Registration handling (reg caching) etc

- ”The Swiss Army Knife of call stateful SIP processing”

# Flexible profile based control

sbc.conf

```
load_profiles=iptelecho
active_profile=iptelecho
...
```

iptelecho.sbcprofile.conf

```
URI=sip:echo@iptel.org
From=<anonymous@mynet.net>
To=<sip:echo@iptel.org>
...
```

SEMS SBC

```
#
U 210.13.3.122:5080 -> 210.13.3.100:5060
INVITE sip:+49123@osbc1.mynet.net SIP/2.0
From: "John" <sip:+431556221@mynet.net>;tag=12
To: "Clara" <+49123@mynet.net>
Call-ID: 3cde5d1a960a-dez6oz34llo4
...
```

```
#
U 210.13.3.100:5060 -> 213.192.59.75:5060
INVITE sip:echo@iptel.org SIP/2.0
From: <anonymous@mynet.net>;tag=3213
To: <sip:echo@iptel.org>
Call-ID: y76IIPf4UD68bb
...
```

# control SBC from proxy

set_fromto.sbcprofile.conf

```
URI=$tU@sbc1.mypeer.net
From=<$fU@mynet.net>
To=<sip:$tU@mypeer.net>
Call-ID=$ci_leg2
...
```

known
SER
pseudo-variables

SEMS SBC

```
#
U 210.13.3.122:5080 -> 210.13.3.100:5060
INVITE sip:+49123@osbc1.mynet.net SIP/2.0
From: "John" <sip:+431556221@mynet.net>;tag=12
To: "Clara" <+49123@mynet.net>
Call-ID: 3cde5d1a960a-dez6oz34llo4
...
```

```
#
U 210.13.3.100:5060 -> 213.192.59.75:5060
INVITE sip:+49123@sbc1.mypeer.net SIP/2.0
From: <+431556221@mynet.net>;tag=3213
To: <sip:+49123@mypeer.net>
Call-ID: 3cde5d1a960a-dez6oz34llo4_leg2
...
```

# SBC example: auth_b2b

- Identity change
- SIP auth upstream
- Set e.g. In headers
  - $P(*name*) selects *name* from P-App-Param

auth_b2b.sbcprofile.conf

```
RURI=sip:$rU@$P(d)
From="\"$P(u)\" <sip:$P(u)@$P(d)>"
To="\"$rU\" <sip:$rU@$P(d)>"

enable_auth=yes
auth_user=$P(u)
auth_pwd=$P(p)
```

Test:

```
        if (uri=~"^sip:\+49.*") {
                remove_hf("P-App-Name");
                remove_hf("P-App-Param");
                append_hf("P-App-Name: sbc\r\n");
                append_hf("P-App-Param: u=8708138;d=sipgate.de;p=mypasswd\r\n");
                force_send_socket(192.168.2.32:5060);
                t_relay_to_udp("192.168.2.34","5060");
                exit;
        }
```

# Profile selection

- Static
  - active_profile=static_config
- Pseudo-var
  - active_profile=$rU
- Mapping
  - active_profile=$M(val=>map)
- Select first matched
  - active_profile=$M($si=>ipmap),$M($ru=>urimap),$H(P-SBCProfile),refuse

ipmap.conf

```
^10\.0\..*=>internal1
^10\.1\..*=>internal2
```

urimap.conf

```
iptel.org=>iptel
fliptel.com=>fliptel
```

# Some profile options

```
RURI=$r
From=$f
To=$t
Contact=<sip:$Ri>
Call-ID=$ci_leg2

outbound_proxy=sip:192.168.5.106:5060
force_outbound_proxy=yes
next_hop=192.168.5.106:5060
outbound_interface=extern

enable_reg_caching=yes
min_reg_expires=3600
max_ua_expires=60

dlg_nat_handling=yes

enable_rtprelay=yes
rtprelay_force_symmetric_rtp=yes
aleg_rtprelay_interface=intern
rtprelay_interface=default
```

```
header_filter=blacklist
header_list=P-App-Param,P-App-Name
sdp_filter=whitelist
sdpfilter_list=g729,g723,ilbc,speex,gsm

append_headers="P-Src-IP: $si\r\n"

enable_session_timer=yes
session_expires=120
minimum_timer=90

enable_auth=yes
auth_user=$P(u)
auth_pwd=$P(p)
```

# Manage SBC

- sems-sbc-* tools
  - get and set active profile
  - load and reload profiles
  - load and reload mappings

```
sems-sbc-get-activeprofile
sems-sbc-get-regex-map-names
sems-sbc-list-profiles
sems-sbc-load-callcontrol-modules
sems-sbc-load-profile
sems-sbc-reload-profile
sems-sbc-reload-profiles
sems-sbc-set-activeprofile
sems-sbc-set-regex-map
sems-sbc-teardown-call
```

- Track profile versions with MD5 hash
- Get statistics from monitoring

# SBC modules

| | | |
|---|---|---|
| SBC | sbc | sbc application |
| | | |
| | uac_auth | For SIP authentication |
| | xmlrpc2di | Control, e.g. Profile reload, regex maps reload |
| | | |
| Call control | cc_ctl | Control settings from SIP msg headers |
| | cc_rest | Read settings from http API |
| | | |
| | cc_prepaid | Prepaid, internal balances |
| | cc_prepaid_xmlrpc | Prepaid, balances external, queried via XMLRPC |
| | cc_call_timer | Max call duration timer |
| | cc_pcalls | Parallel call limit |
| | cc_syslog_cdr | CDR generation |
| | | |
| | cc_dsm | SBC apps in DSM script |

# SBC programming

- Simple call_control API
  - start(), connect(), end()
- Extended call_control API
  - SIP Message events, like sipRequest, sipReply, ...
  - B2B events: B2B.otherRequest, B2B.otherReply, …
  - Disconnect and reconnect call legs (PBX style)
  - Program with DSM script

# SBC DSM example

```
--
-- simple DSM/SBC example: disconnect B leg after a timer
-- play a file in the A leg after that
--

import(mod_dlg);
import(mod_sbc);
import(mod_utils);

initial state START enter {
  log(3, "entering START state");
};

transition "init event" START - start / {
  log(3, "initializing");
  logAll(3);

  if sbc.isALeg() {
    log(3, "this is an A leg");
    setTimer(1, 10);
  } else {
    log(3, "this is a B leg");
  }
} -> RUN;

state RUN;
```

# SBC DSM example (cont)

```
transition "state changed" RUN - legStateChange / logParams(3) -> RUN;

transition "timer hit" RUN - timer(#id == 1) / {
  -- save other leg's ltag
  dlg.getOtherId($b_ltag);

  -- don't send hold, keep media session
  sbc.disconnect(false, true);

  -- instruct other leg to hang up
  set($cmd="hangup");
  set($call_id=@local_tag);
  postEvent($b_ltag, cmd;call_id);

  setInputPlaylist();
  connectMedia();
  playFile("wav/default_en.wav");
  sbc.streamsSetReceiving(false, false);

} -> PLAYING_FILE;

state PLAYING_FILE;
```

# SBC DSM example (cont)

```
transition "file ended" PLAYING_FILE - noAudio / {
  -- use sbc.stopCall, otherwise RTP relay may still be active
  sbc.stopCall("normal-call-clearance");
} -> END;

-- B leg side
transition "got disconnect cmd" RUN - event(#cmd=="hangup") / {
  -- disconnect our leg from the other, too
  sbc.disconnect(false, true);
  -- stop our call leg
  sbc.stopCall("normal-call-clearance");
} -> END;

state END;
```

# Questions?

Thanks for your attention.

# Links and References

- SEMS homepage: http://iptel.org/sems
- Code: sems repo at git.sip-router.org
- DSM documentation

  http://git.sip-router.org/cgi-bin/gitweb.cgi?p=sems;a=tree;f=doc/dsm
- FRAFOS website: www.frafos.com