SEMS SBC

Stefan Sayer
CEO, FRAFOS GmbH

FOSDEM 2012, 05.02.2012

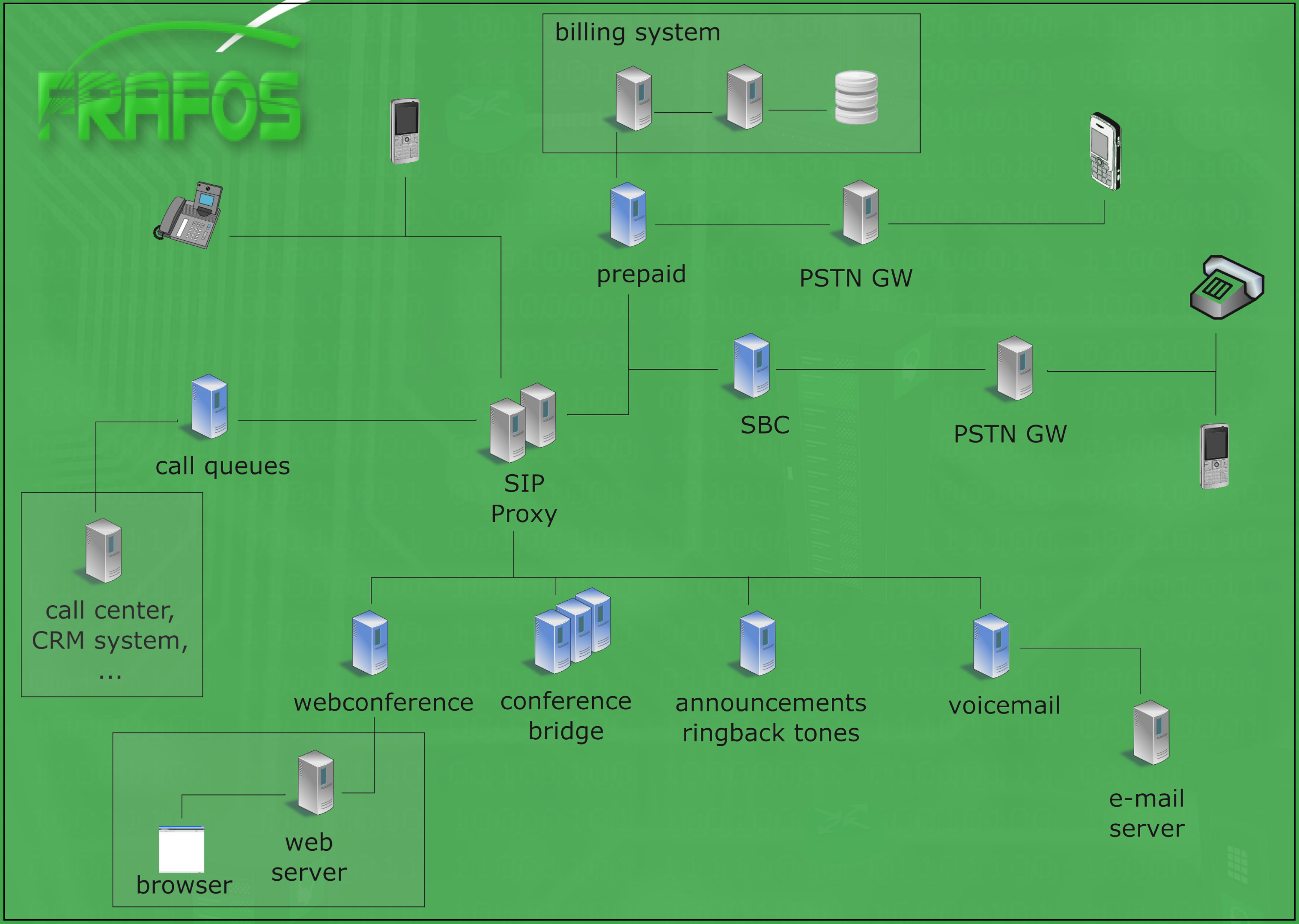
Contents

- SEMS project
- The flexible, open SBC
- SBC programmability

The SIP Express Media Server

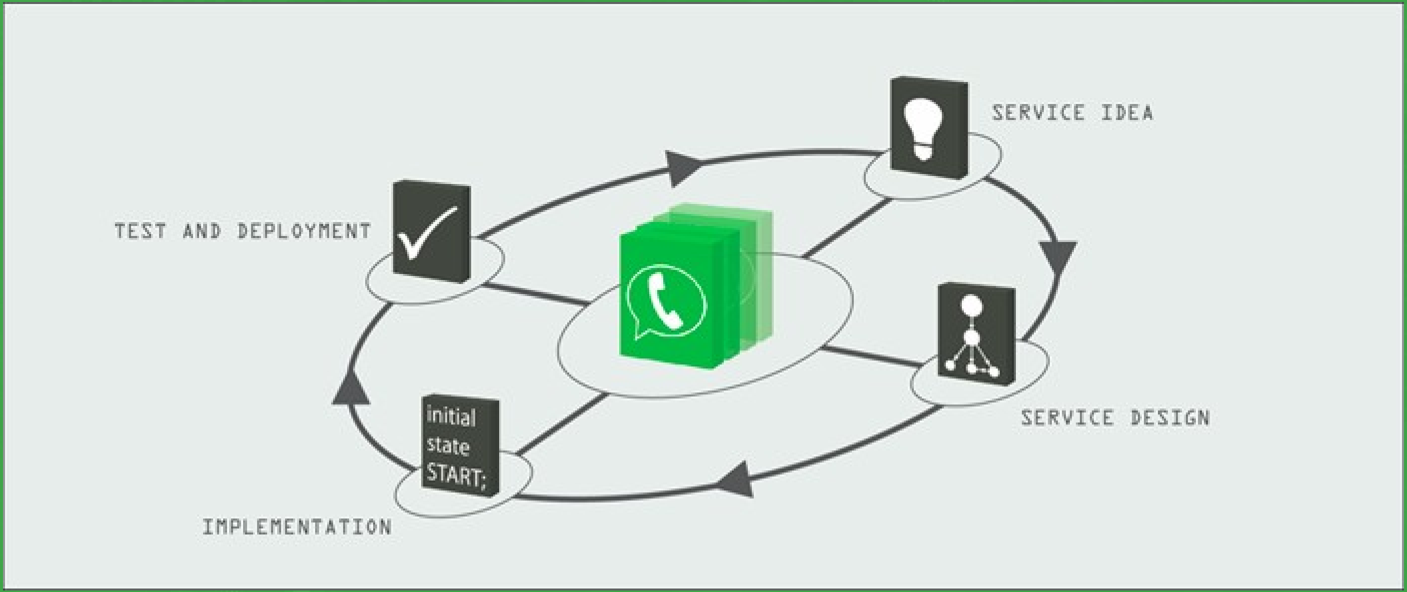
- Media, application server from iptel.org
- Only 1 year younger than SER (*2002)
- Widely used by carriers, ITSPs, OEMs, Universities, hobbyists

SEMS use cases

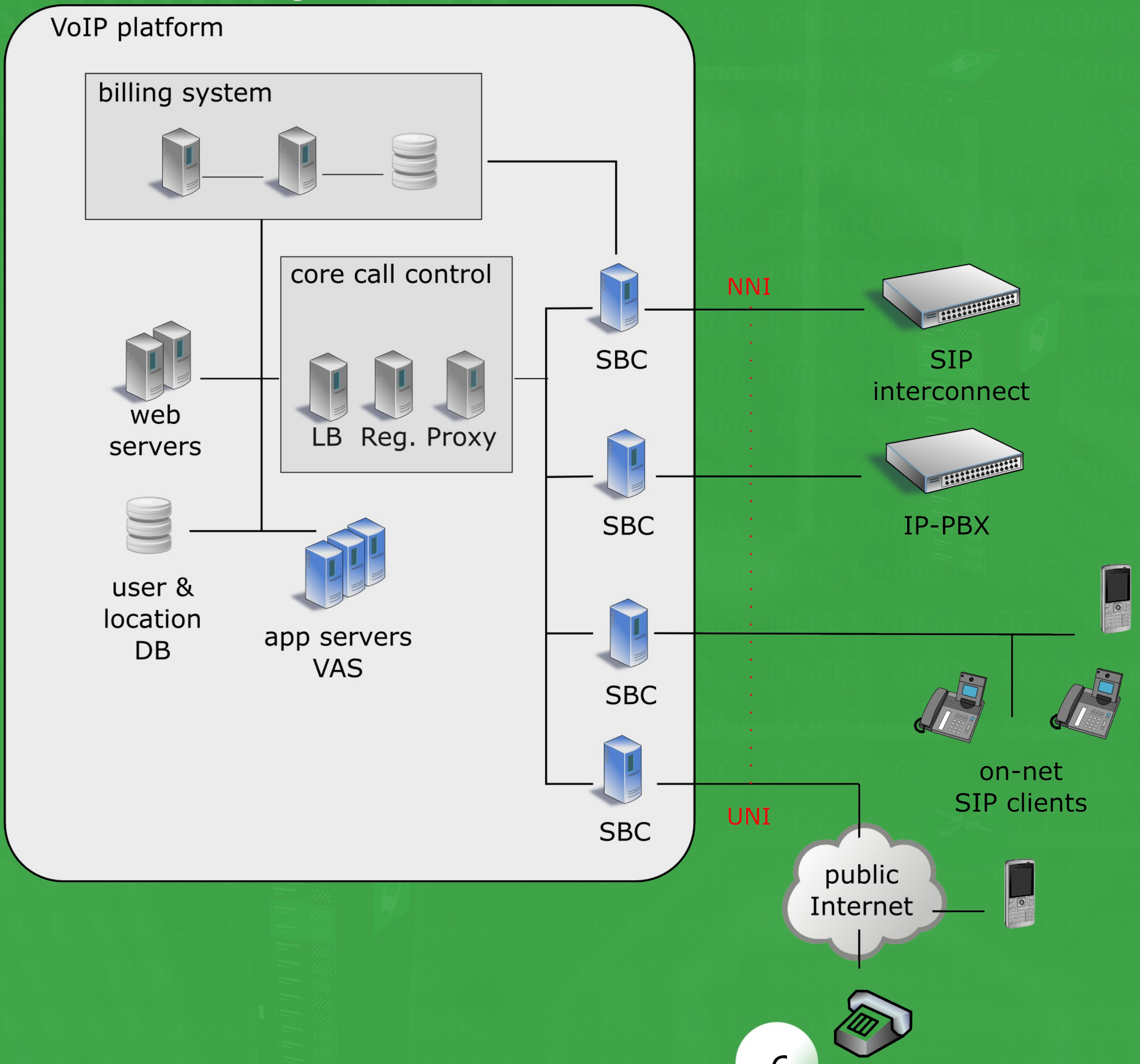


SEMS: The VAS platform

- C++, Python APIs
- DSM: State charts scripting engine



SBCs - what?



"SBCs are SIP application servers with focus on security and isolation"

SBCs - the need

- Security requirements are rising
  - → Policy enforcement and control at UNI/NNI
- Topology hiding is necessary
  - → NAT and security
- Core call controls become big and slow
  - → Routing and service management at NNI
- SIP implementations are buggy
  - → “SIP normalization”, translation

SBCs - the special case

- On signaling and media plane
- Call stateful – high requirements for availability and scalability
- Interworking with all “SIP dialects”

The SEMS SBC

- Widely deployed SIP technology
- Broad range of media capabilities
- Configurable transparency
- Policy programmability – “SBC platform”

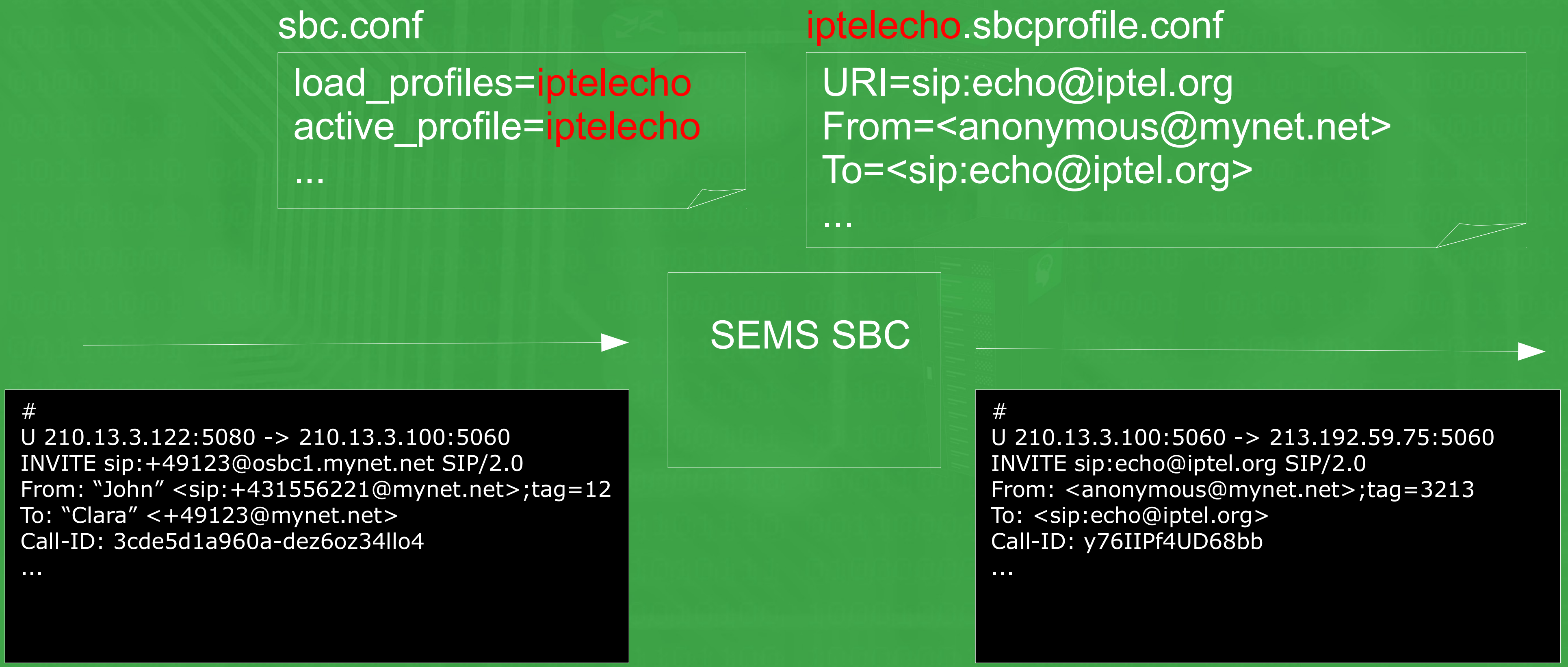
Signaling Features

- Topology hiding
- From, To, RURI, Contact, Call-ID manipulation
- Header and message filter
- Adding headers
- Reply code translation
- SIP authentication
- SIP Session Timer, Call Timer
- Prepaid accounting

Media features

- RTP anchoring / media steering
- Physical network separation
- NAT traversal, symmetric RTP (comedia style)
- Codec filter
- SDP normalization

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
...
```

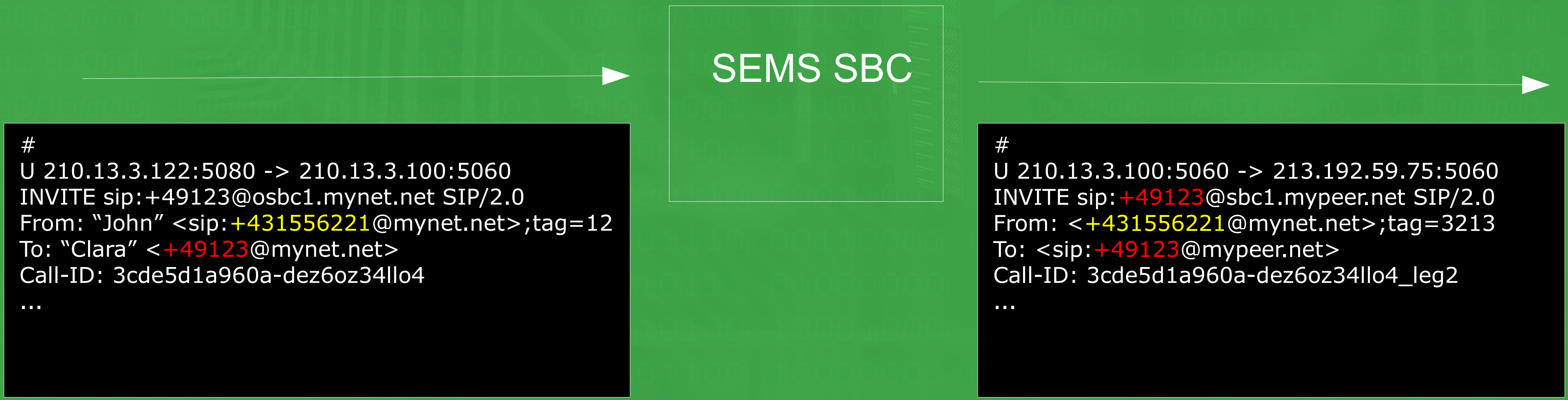
- define SBC behaviour in profiles

Set RURI, From, To, Call-ID ...

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



```
#
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
...
```

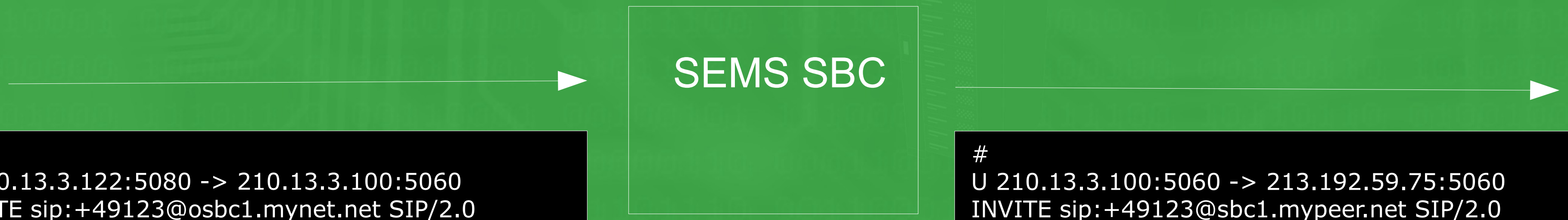
Replacement patterns

- RURI, From, To, PAI, PPI ($r, $f, $t, $a, $p)
- Call-ID ($ci)
- src, dst IP address/port ($si, $pi, $Ri, $Rp)
- P-App-Param hdr parameter ($P(myparam))
- Header value ($H(P-My-Header))
- Map any value via regexp ($M(val=>map))

Control SBC from proxy

dynamic_rtprelay_sst.sbcprofile.conf

```
...
enable_rtprelay=$H(P-Enable-RTPRrelay)
enable_session_timer=$H(P-Enable-SST)
...
```



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
P-Enable-RTPRelay: no
P-Enable-SST: yes
...
```

```
#
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
Session-Expires: 300
...
```

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
```

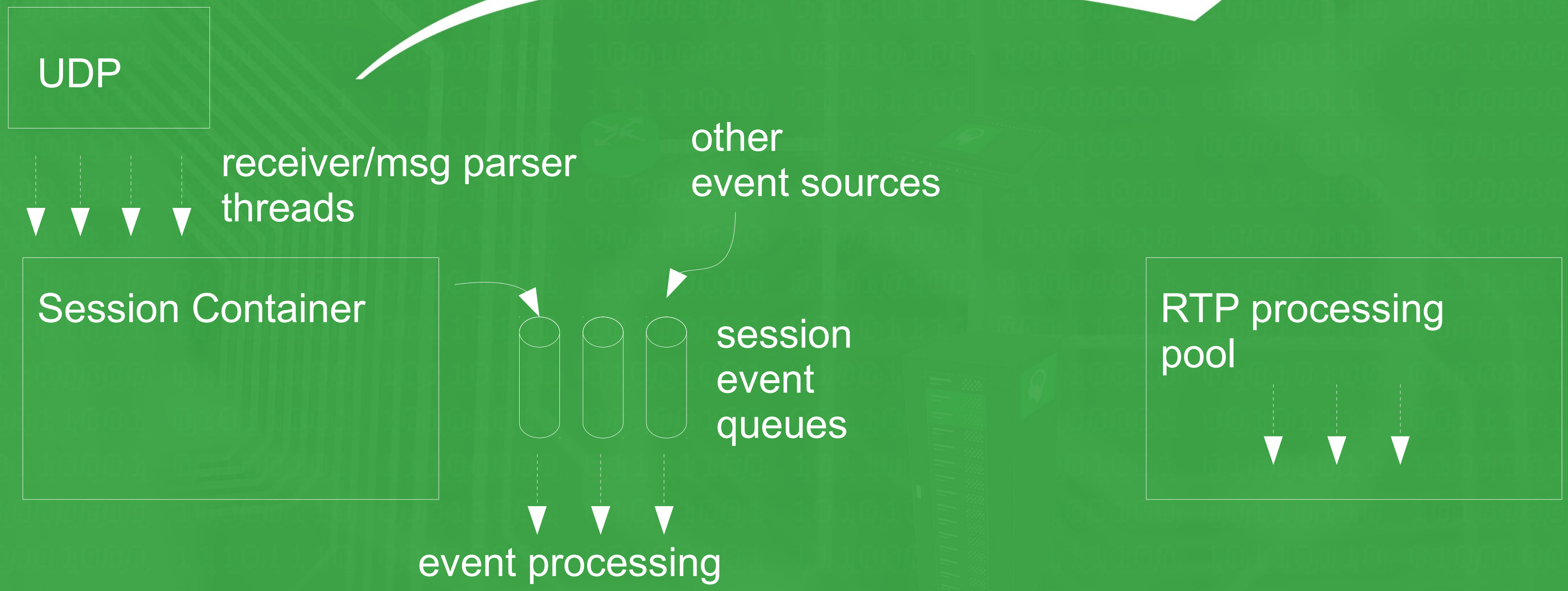
urimap.conf

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

Manage SBC

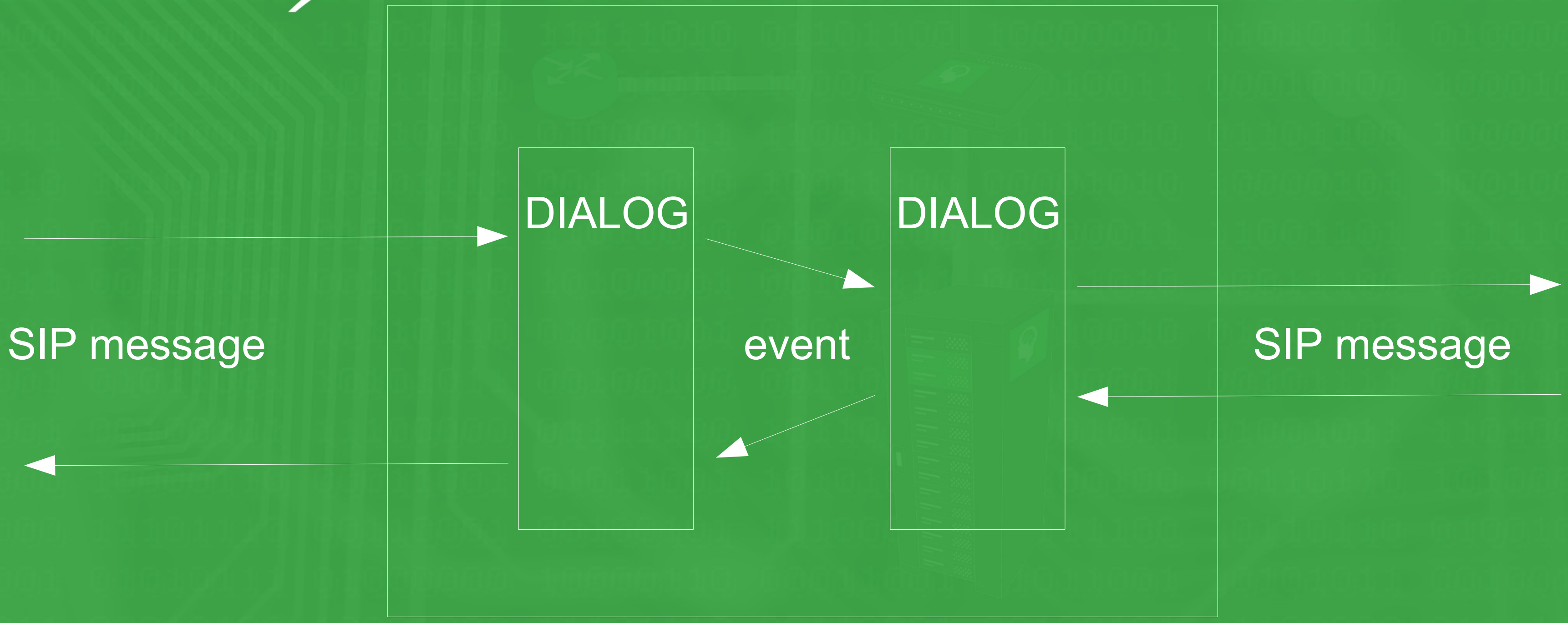
- sems-sbc-* tools
  - get and set active profile
  - load and reload profiles
  - load and reload mappings
- Track profile versions with MD5 hash
- Get statistics from monitoring

Processing model



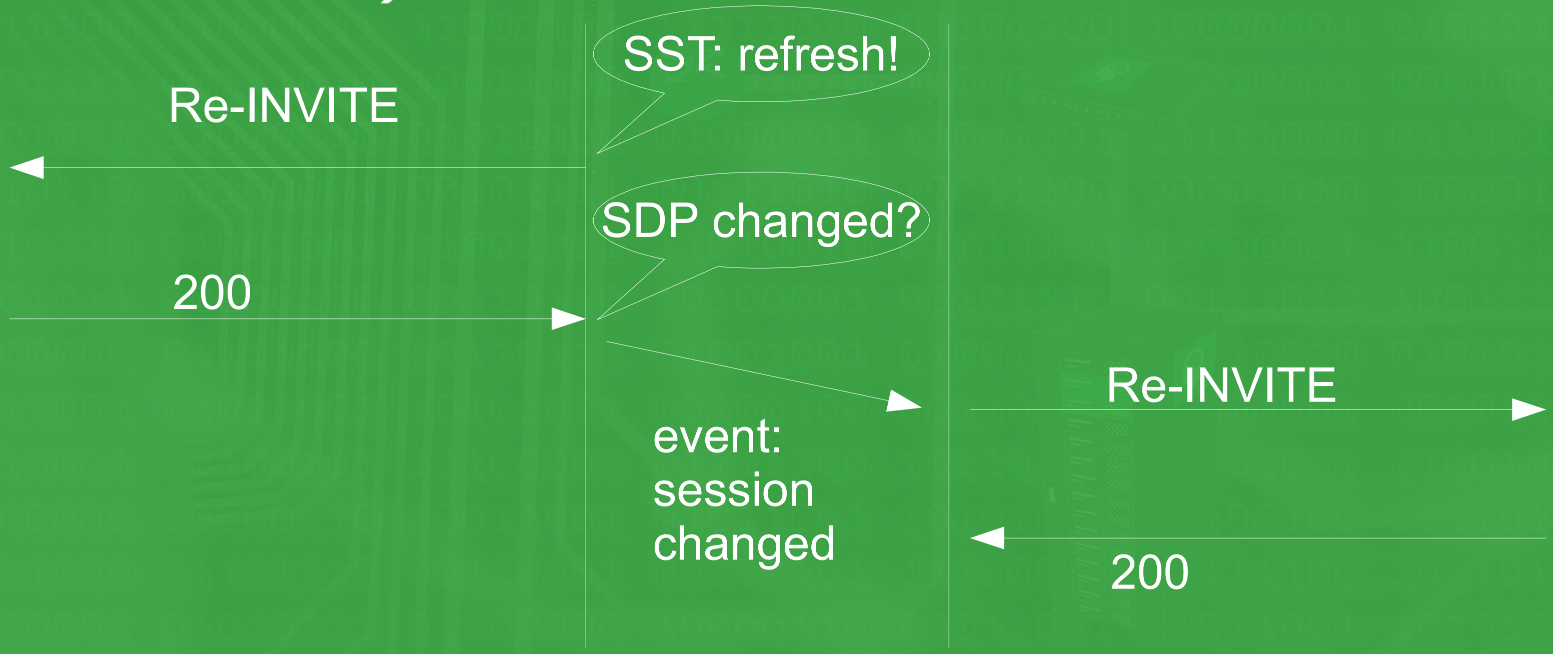
- Signaling: Async, one thread per call or thread pool
- Media: Sync, Thread pool

SEMS B2BUA architecture



- Two complete, separate instances of dialog handling: Locally SIP correct

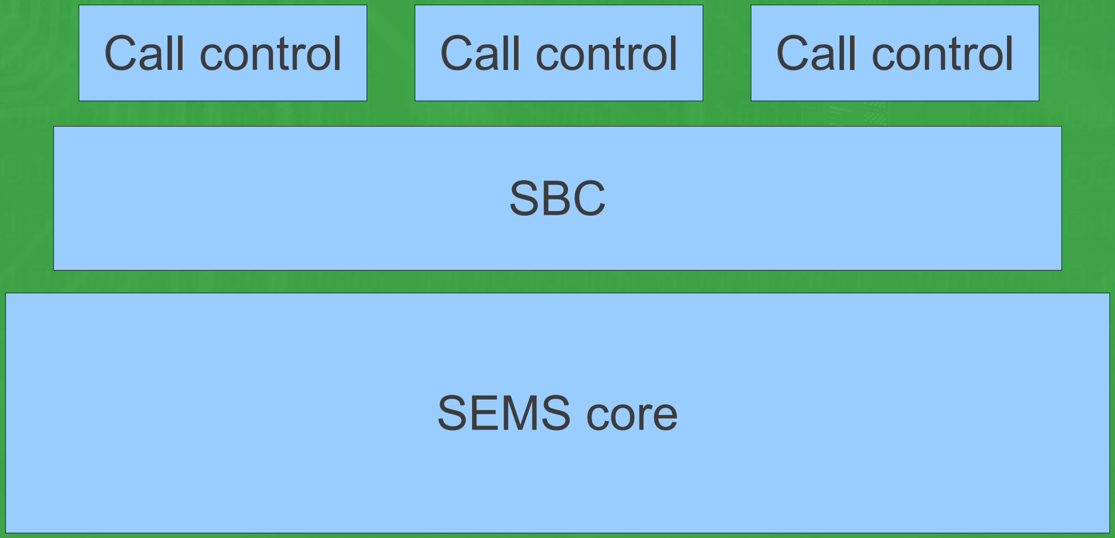
E.g.: Session Timers



- Use UPDATE or re-INVITE for refresh
- SST and timer values per leg
- Try to have e2e refresh

SBC programmability

- Pluggable Call Control modules for custom SBC application scenario
- e.g. policing with external data source

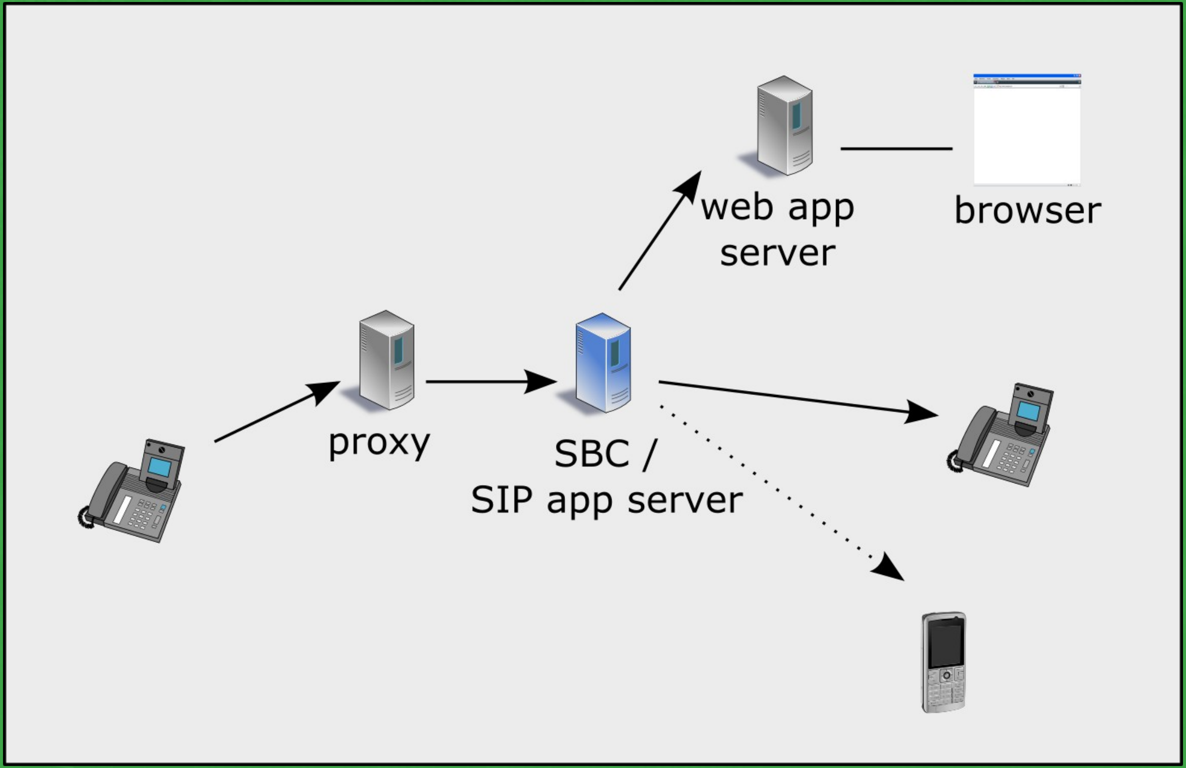


Call control SBC-API

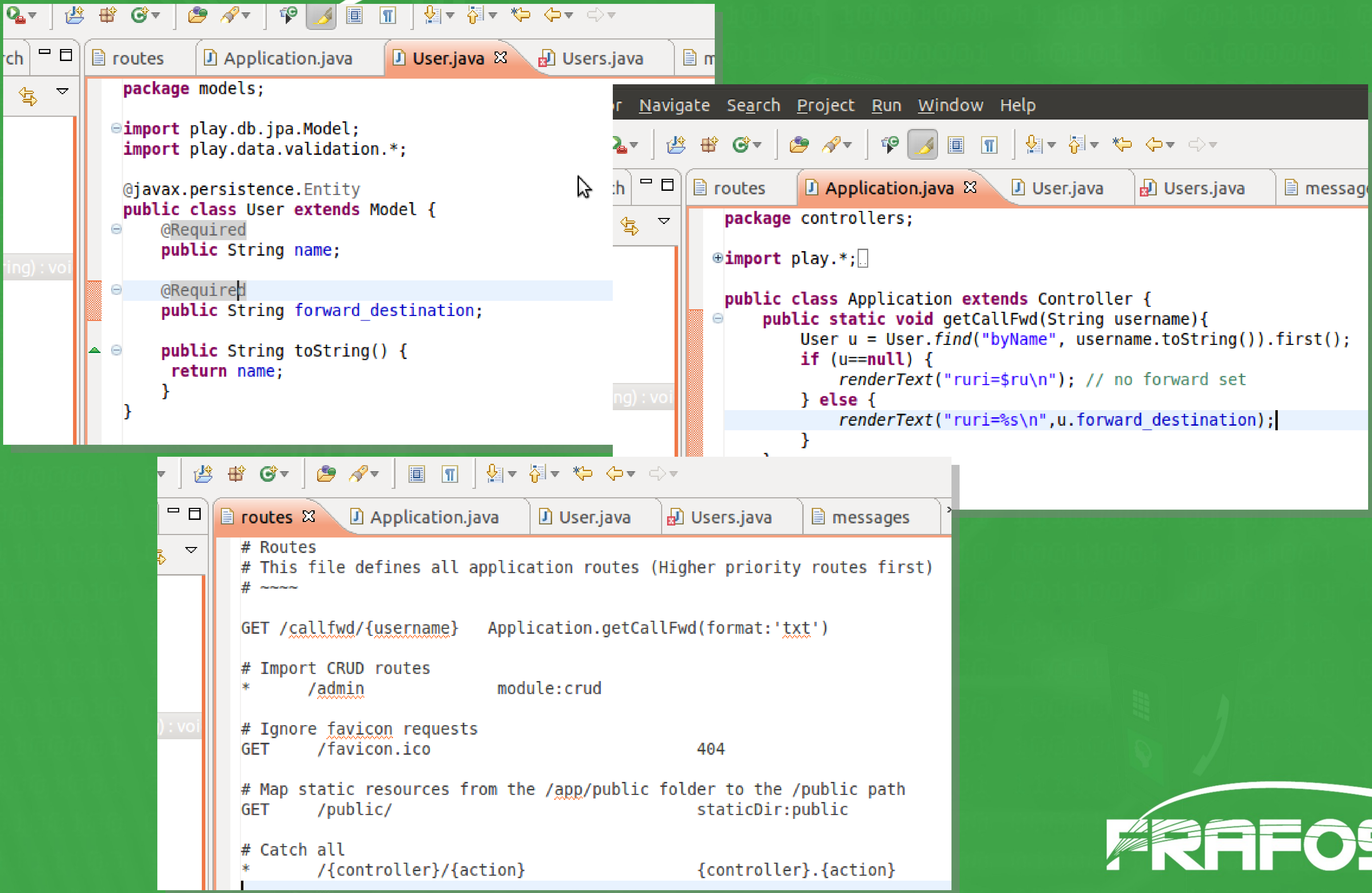
- V1: connect(...), start(...), end(...)
- Control SBC through call profile object
- Pattern replacements (provisioning) in both input and output
- Modules: CDR generation, call timer, prepaid, parallel calls limit, REST/http, REDIS blacklist

Programmability example (1)

- Call Forward settings via Web App
- Destination queried via REST interface



Minimal Play! Web app



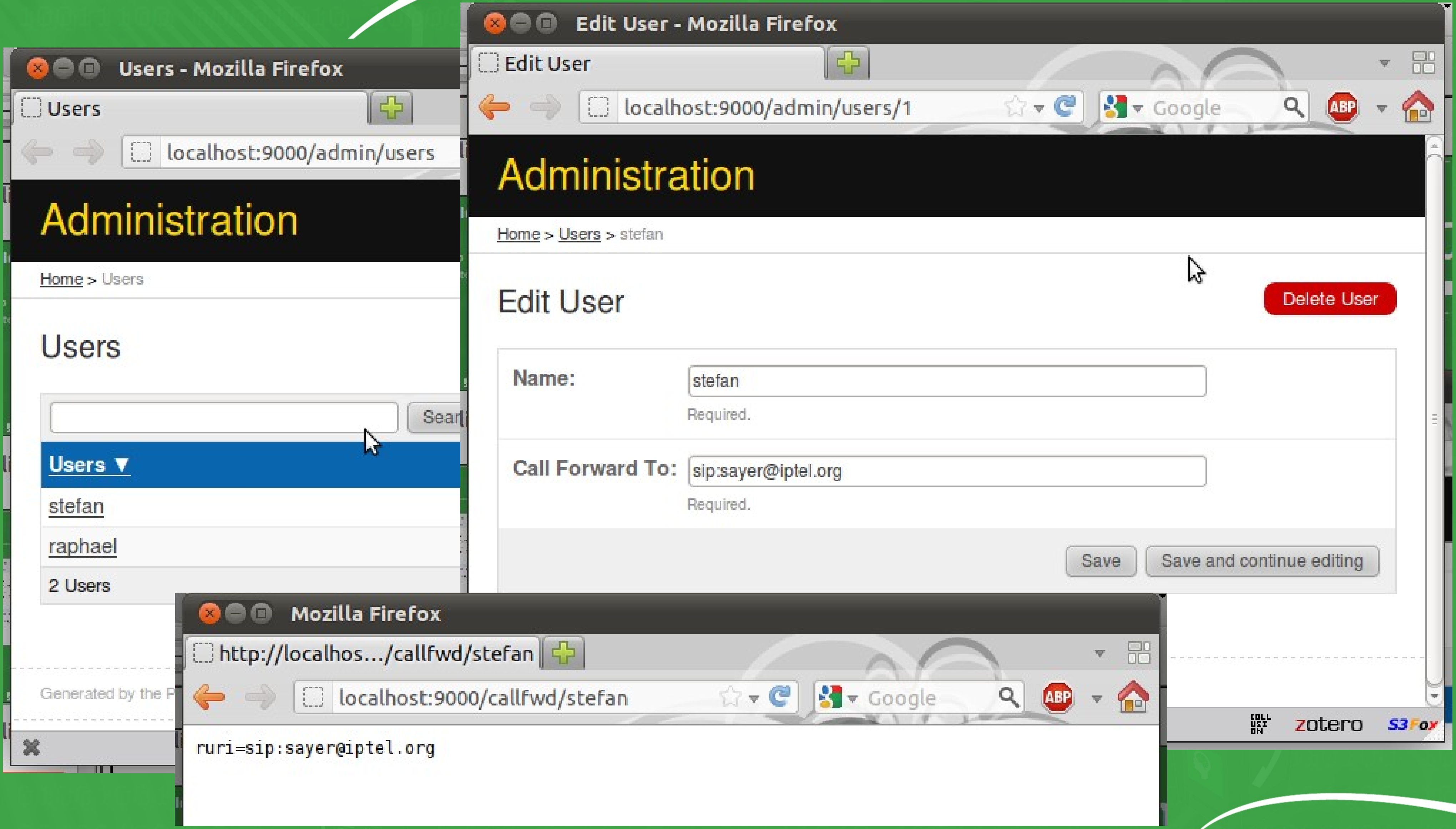
```
package models;

import play.db.jpa.Model;
import play.data.validation.*;

@javax.persistence.Entity
public class User extends Model {
    @Required
    public String name;

    @Required
    public String forward_destination;

    public String toString() {
     return name;
    }
}
```

```
package controllers;

import play.*;

public class Application extends Controller {
    public static void getCallFwd(String username){
        User u = User.find("byName", username.toString()).first();
        if (u==null) {
            renderText("ruri=$ru\n"); // no forward set
        } else {
            renderText("ruri=%s\n",u.forward_destination);
        }
```

```
# Routes
# This file defines all application routes (Higher priority routes first)
# ~~~~

GET /callfwd/{username}   Application.getCallFwd(format:'txt')

# Import CRUD routes
*       /admin              module:crud

# Ignore favicon requests
GET     /favicon.ico                            404

# Map static resources from the /app/public folder to the /public path
GET     /public/                                staticDir:public

# Catch all
*       /{controller}/{action}                  {controller}.{action}
```

FRAFOS

...with CRUD module



Users - Mozilla Firefox

localhost:9000/admin/users

Administration

Home > Users

Users

Users ▼
stefan
raphael
2 Users

Generated by the P

Edit User - Mozilla Firefox

localhost:9000/admin/users/1

Administration

Home > Users > stefan

Edit User

Name: stefan
Required.

Call Forward To: sip:sayer@iptel.org
Required.

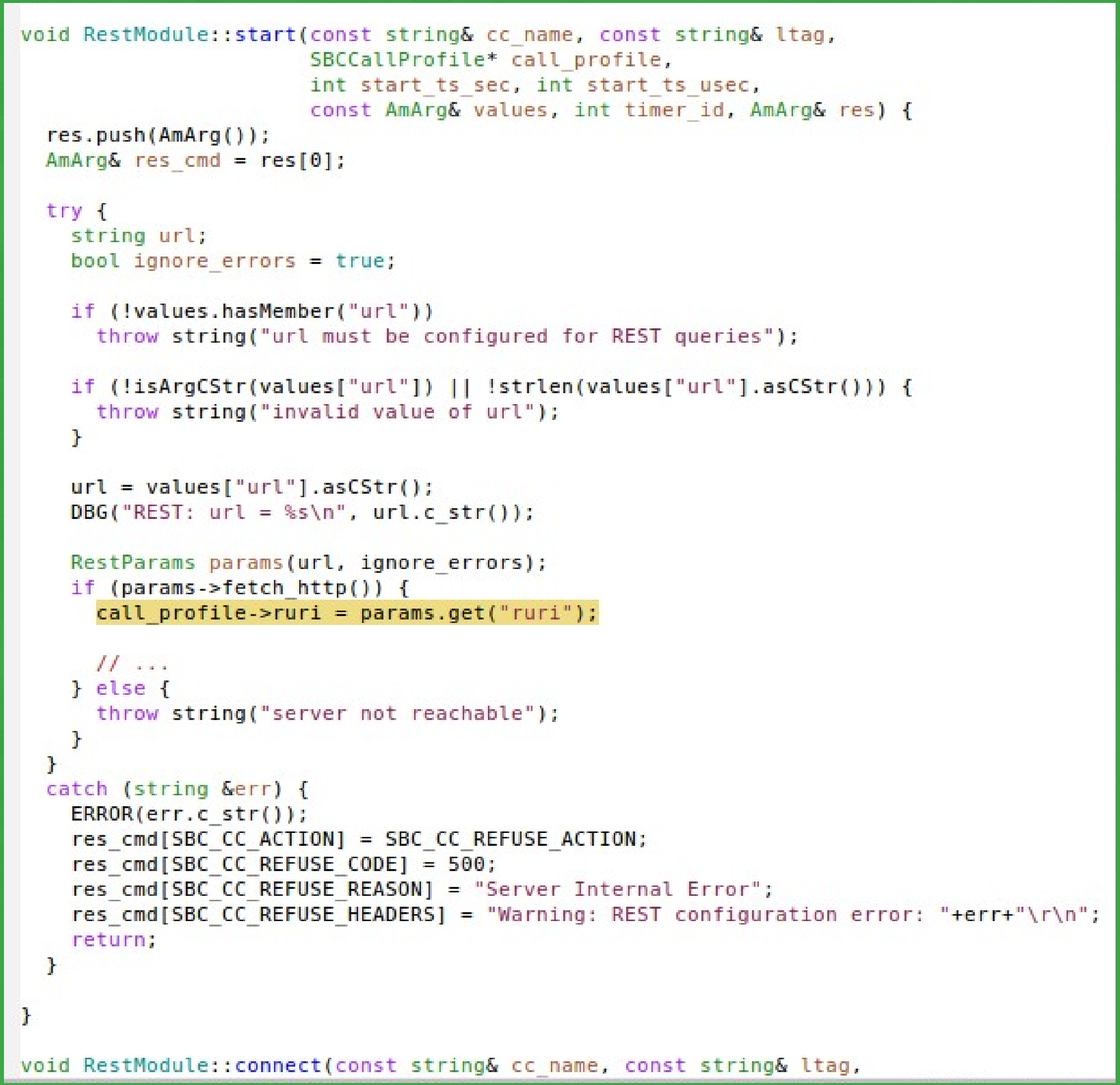
Mozilla Firefox

localhost:9000/callfwd/stefan

```
ruri=sip:sayer@iptel.org
```

REST call control module

- Using libcurl for http request
- Result expected as JSON or TEXT (key=value\n)



```
void RestModule::start(const string& cc_name, const string& ltag,
                       SBCCallProfile* call_profile,
                       int start_ts_sec, int start_ts_usec,
                       const AmArg& values, int timer_id, AmArg& res) {
  res.push(AmArg());
  AmArg& res_cmd = res[0];

  try {
    string url;
    bool ignore_errors = true;

    if (!values.hasMember("url"))
      throw string("url must be configured for REST queries");

    if (!isArgCStr(values["url"]) || !strlen(values["url"].asCStr())) {
      throw string("invalid value of url");
    }

    url = values["url"].asCStr();
    DBG("REST: url = %s\n", url.c_str());

    RestParams params(url, ignore_errors);
    if (params->fetch_http()) {
      call_profile->ruri = params.get("ruri");

      // ...
    } else {
      throw string("server not reachable");
    }
  }
  catch (string &err) {
    ERROR(err.c_str());
    res_cmd[SBC_CC_ACTION] = SBC_CC_REFUSE_ACTION;
    res_cmd[SBC_CC_REFUSE_CODE] = 500;
    res_cmd[SBC_CC_REFUSE_REASON] = "Server Internal Error";
    res_cmd[SBC_CC_REFUSE_HEADERS] = "Warning: REST configuration error: "+err+"\r\n";
    return;
  }

}

void RestModule::connect(const string& cc_name, const string& ltag,
```

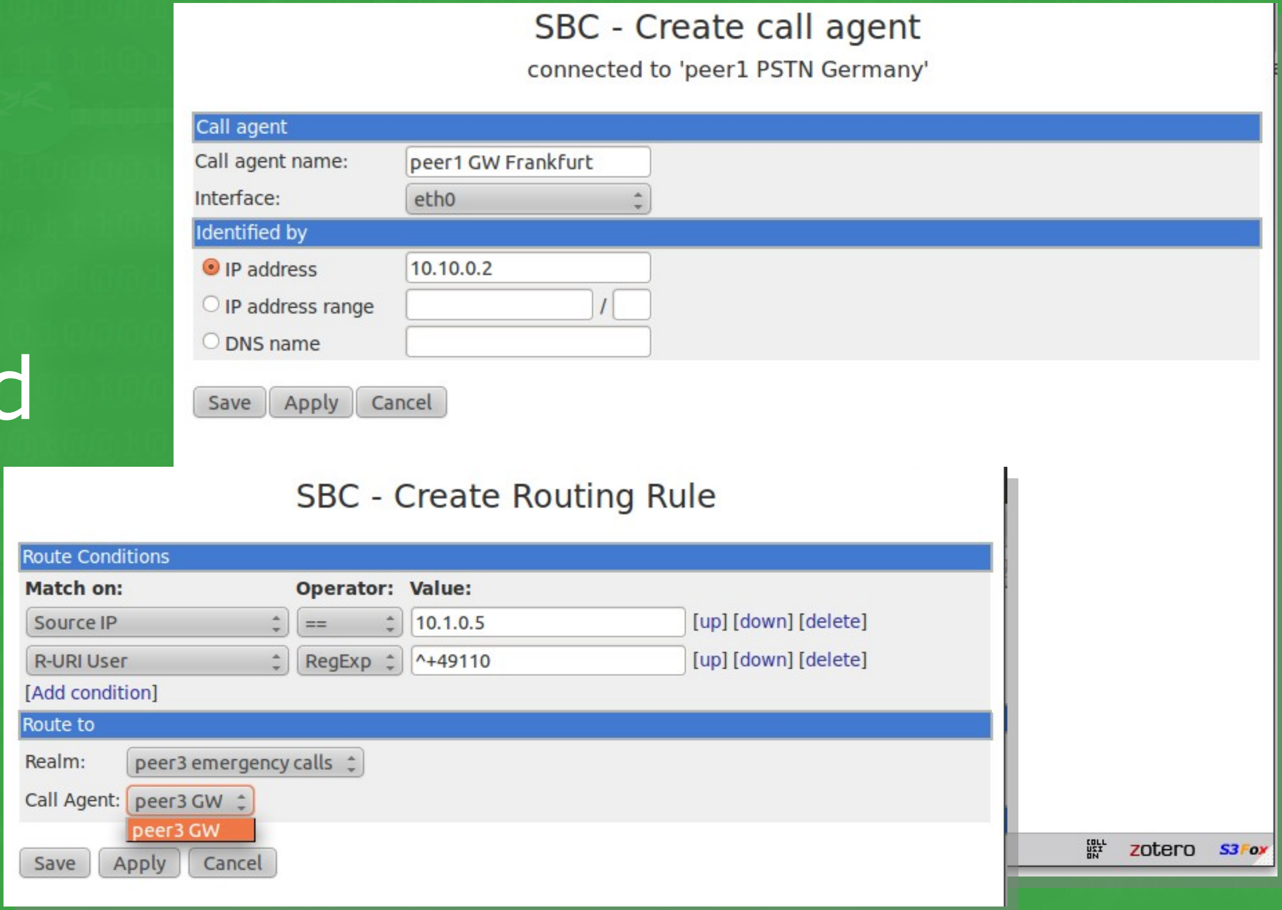
Programmability example (2)

- In-memory Blacklist DB: REDIS
- Connection pool
- Configurable command
  - SMEMBER blacklist $fU
  - rate limiting with ZRANGE
  - More complex logic in lua
  - …

Programmability example (3)

FRAFOS:

SBC with Provisioning and OAM GUI



SBC - Create call agent

connected to 'peer1 PSTN Germany'

Call agent

Call agent name:	peer1 GW Frankfurt
Interface:	eth0

Identified by

● IP address	10.10.0.2
○ IP address range	/
○ DNS name	

SBC - Create Routing Rule

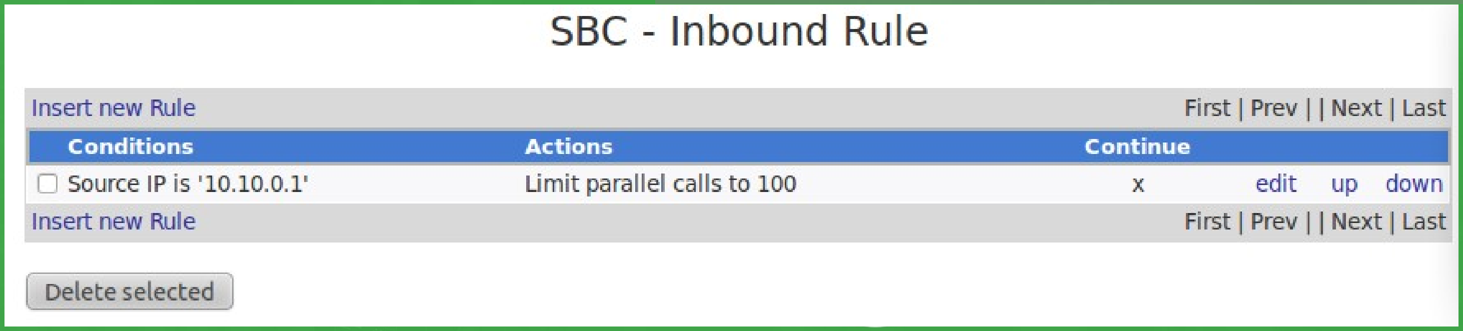
Route Conditions

Match on:	Operator:	Value:	
Source IP	==	10.1.0.5	[up] [down] [delete]
R-URI User	RegExp	^+49110	[up] [down] [delete]

[Add condition]

Route to

Realm:	peer3 emergency calls
Call Agent:	peer3 GW



SBC - Inbound Rule

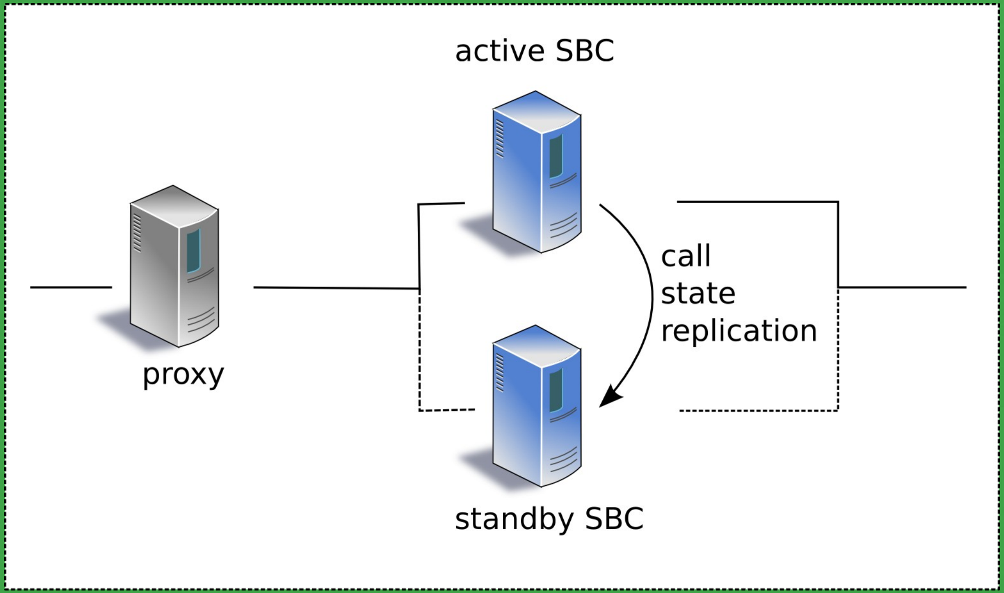
Insert new Rule — First | Prev | | Next | Last

Conditions	Actions	Continue			
□ Source IP is '10.10.0.1'	Limit parallel calls to 100	x	edit	up	down

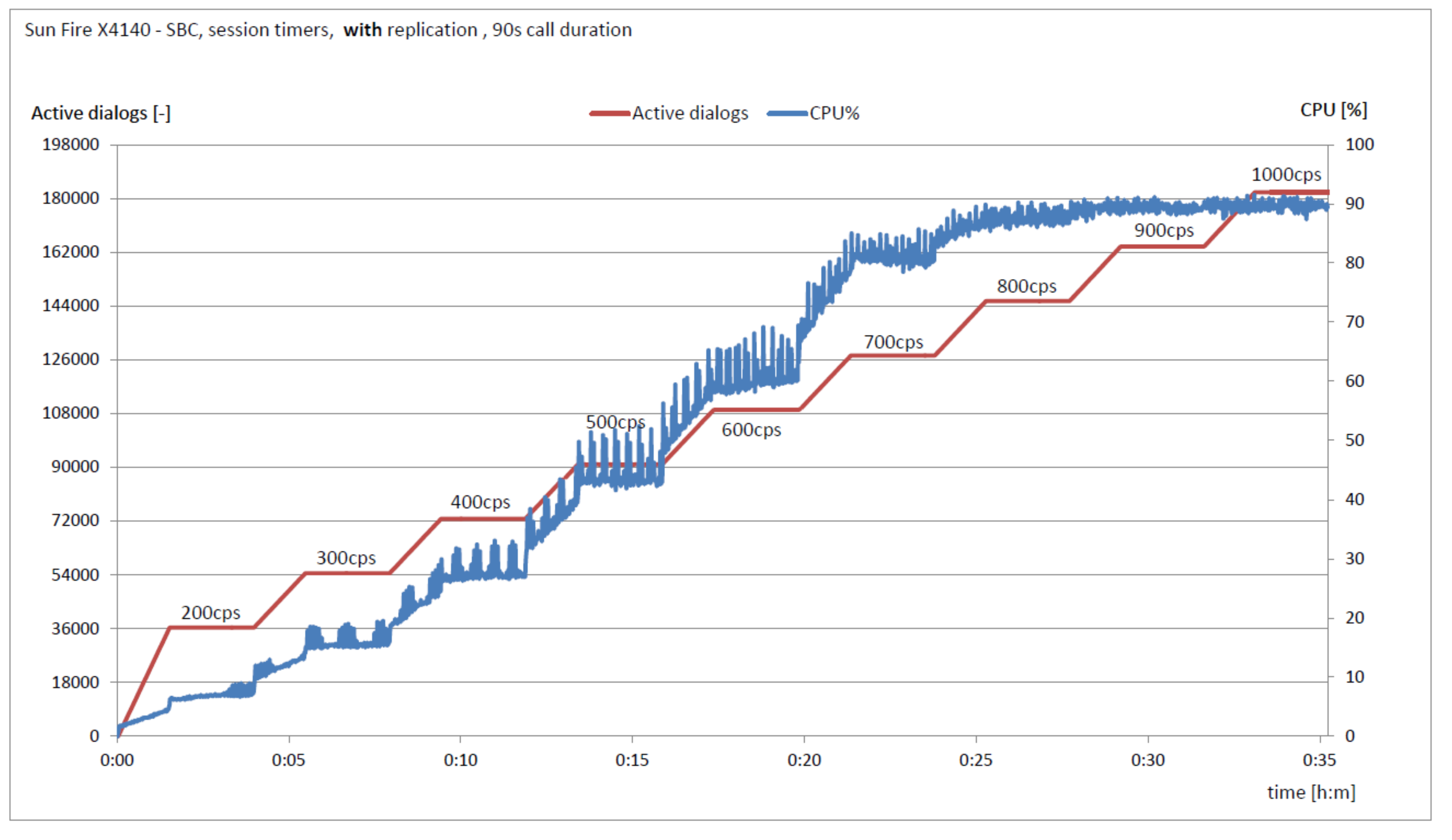
Insert new Rule — First | Prev | | Next | Last

FRAFOS: SBC high availability

- Replication of call state to hot standby
- Transparent fail-over



SBC performance



FRAFOS

Thank You.

http://iptel.org/sems