sip:provider
The VoIP Turnkey Solution

sip:provider CE:
Plugging in WebRTC and Other Use Cases

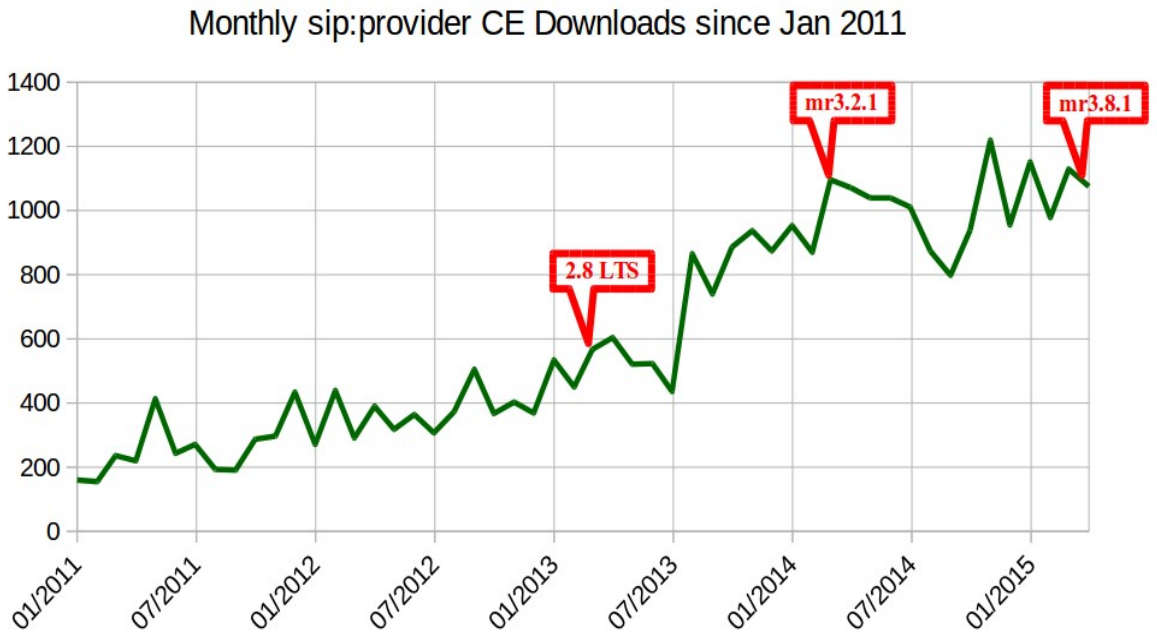
Daniel Grotti
<dgrotti@sipwise.com>
Andreas Granig
<agranig@sipwise.com>
Gernot Fuchs
<gfuchs@sipwise.com>
Andrew Pogrebennyk
<apogrebennyk@sipwise.com>
Victor Seva
<vseva@sipwise.com>

What's the sip:provider CE?

- A turn-key appliance for real-time communication (voice, video, presence, IM)
- using SIP and XMPP
- for carrier environments with 50k+ subscribers and 2k+ parallel calls
- based on Kamailio, Prosody and Sipwise projects

Some Statistics

- First public release in December 2010
- 16 releases so far (latest is mr3.8.1)
- 31k+ downloads total, ~1k per month



Agenda

- Set up your Virtual Machines
- Hook up SIP, WebRTC and XMPP Clients
- Use Rewrite Rules, Peering and Billing
- Know the Architecture
- Manage Configuration Files
- Tweaks for different Use Cases

Set up your VMs

- http://www.vagrantup.com
- http://www.virtualbox.org
- Grab the SPCE image from our USB Stick

```
$ vagrant init spce sip_provider_CE_mr3.8.1_vagrant.box
$ vagrant up
```

Accessing your SPCE

- https://your-ip:1443
  administrator/administrator

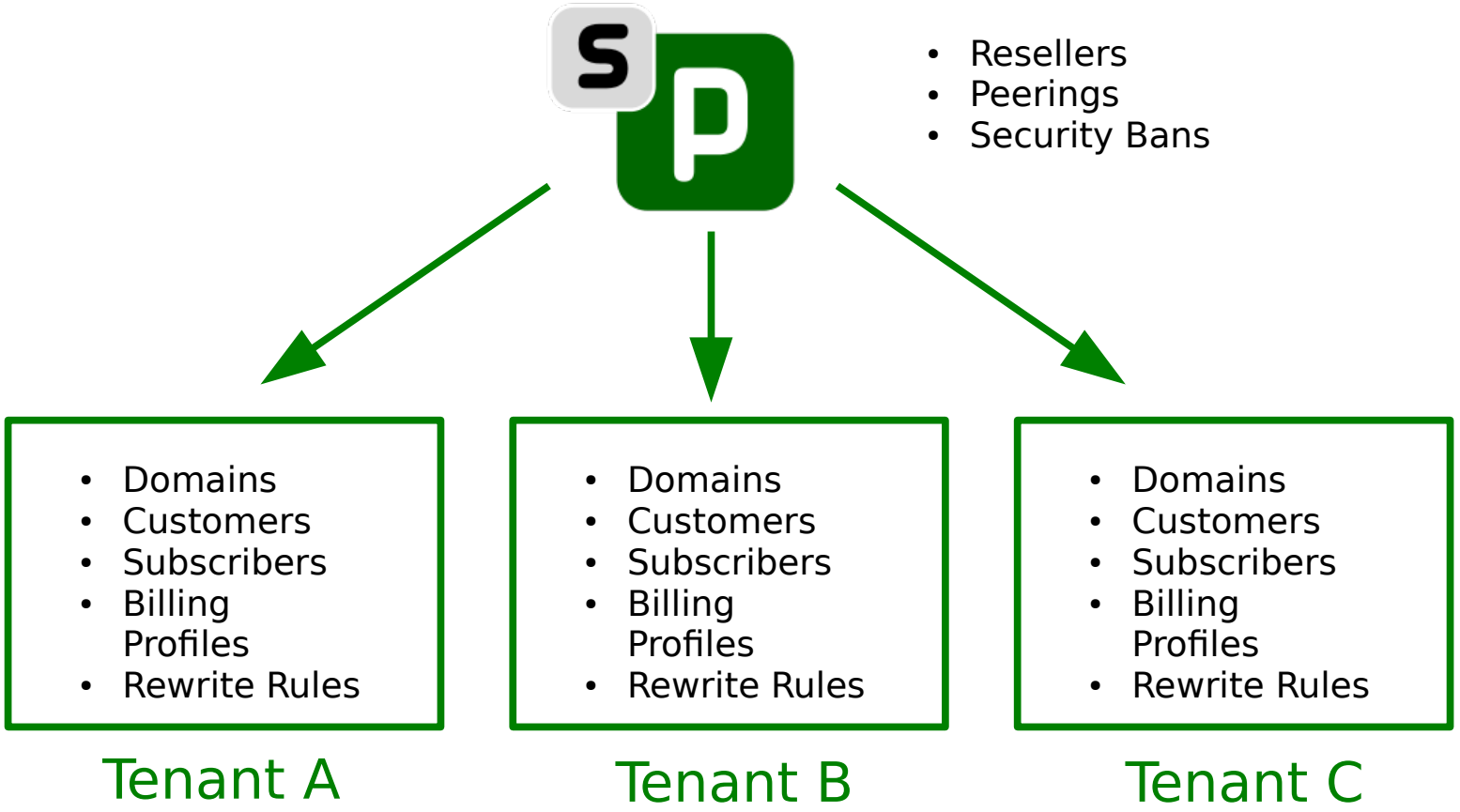
- vagrant ssh

  sudo -s

  root/sipwise

Creating a Reseller

- (Almost) Everything is Multi-Tenant!

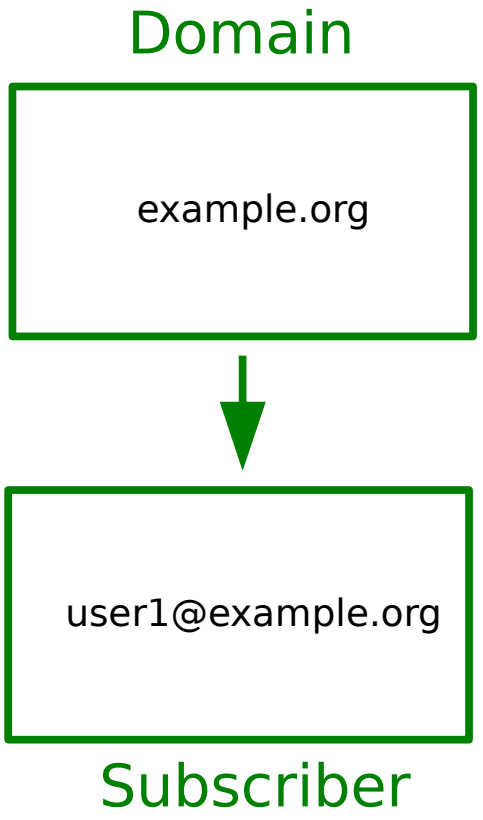


Creating a Reseller

- *default* Reseller, or your own:
- Settings → Resellers → Create with Default
- Adapt Default Values
  - Base Information
  - Contact Email
  - Admin Logins
- Create Billing Profile for Customers

Creating a Domain

- Domain-Preferences are defaults for Subscriber-Preferences



- use_rtpproxy: *always with plain SDP*

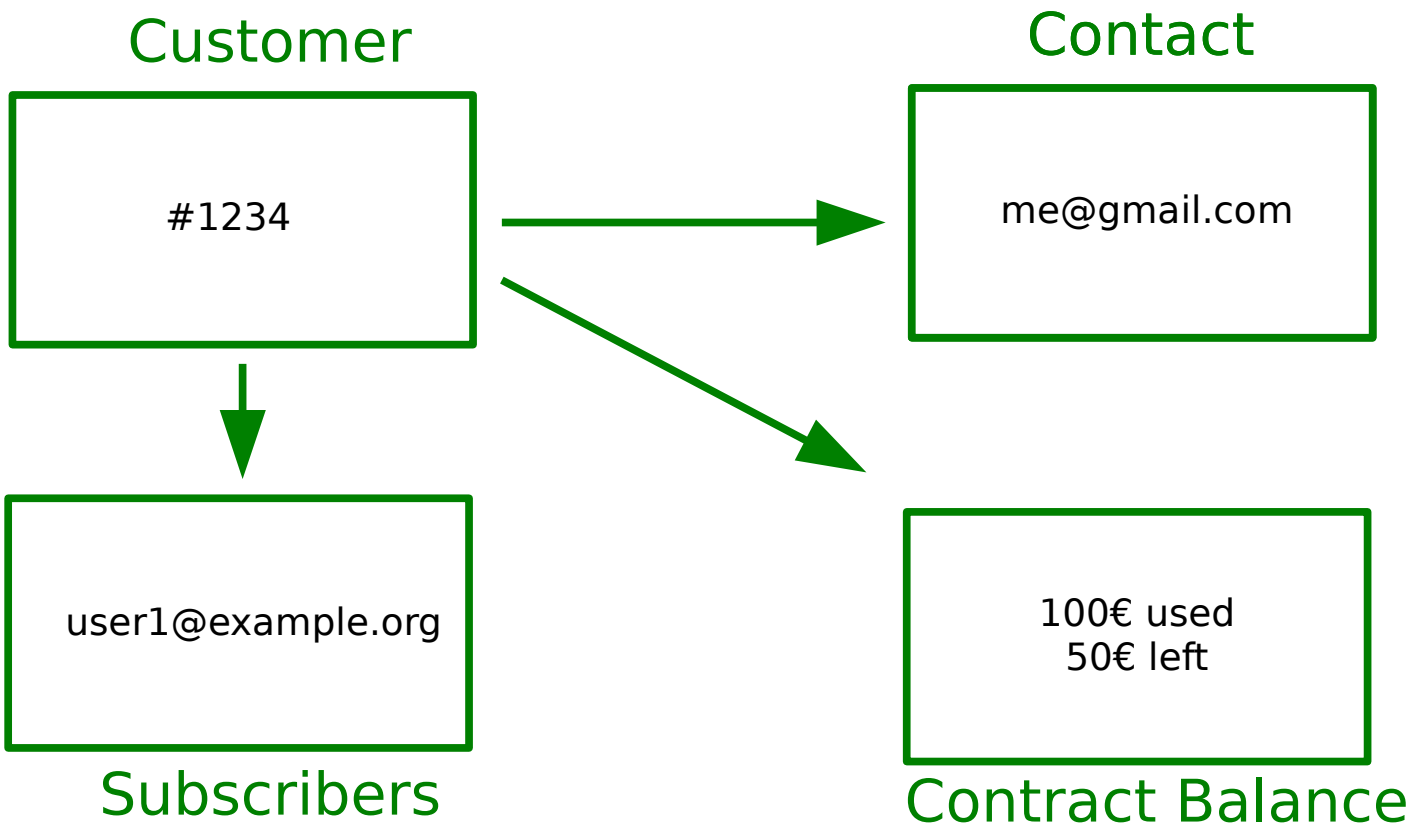
- use_rtpproxy: *domain default*
  - or -
- use_rtpproxy: *always with rtpproxy as only ICE candidate*

Creating a Domain

- *Your IP* as Domain, or your own:
- Settings → Domains → Create Domain
- Default Domain-Preferences are fine (for now)

Creating a Customer

- Customers are Billing Containers

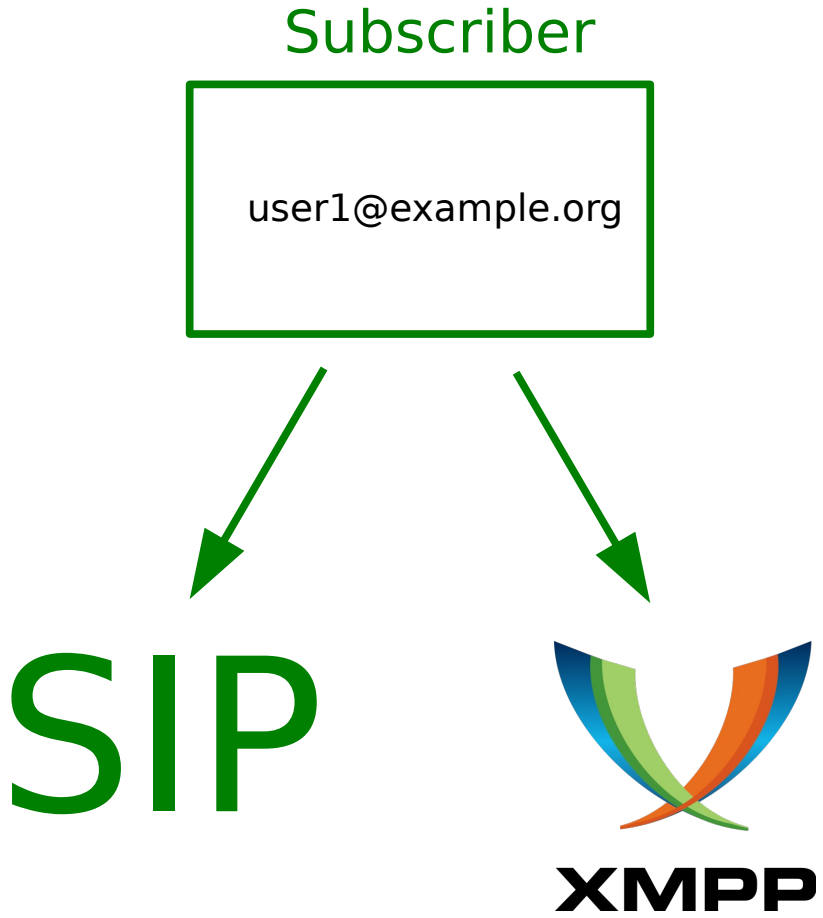


Creating a Customer

- Use an existing one, or your own:
- Settings → Customers → Create Customer
  - Select/Create a Contact
  - Select/Create a Billing Profile

Creating a Subscriber

- Now we're there!



Creating a Subscriber

- Use an existing one, or your own:
- Settings → Customers → *Your Customer* → Details → Create Subscriber
- or
- Settings → Subscribers → Create Subscriber

Connecting your Subscribers

- SIP and XMPP work out of the box
- SIP/TLS needs to be enabled
  - `vim /etc/ngcp-config/config.yml`
  - `kamailio → lb → tls → enable: 'yes'`
  - `ngcpcfg apply`

What about WebRTC?

`ws://your-ip:5060/ws, wss://your-ip:5061/ws, wss://your-ip:1443/wss/sip/`

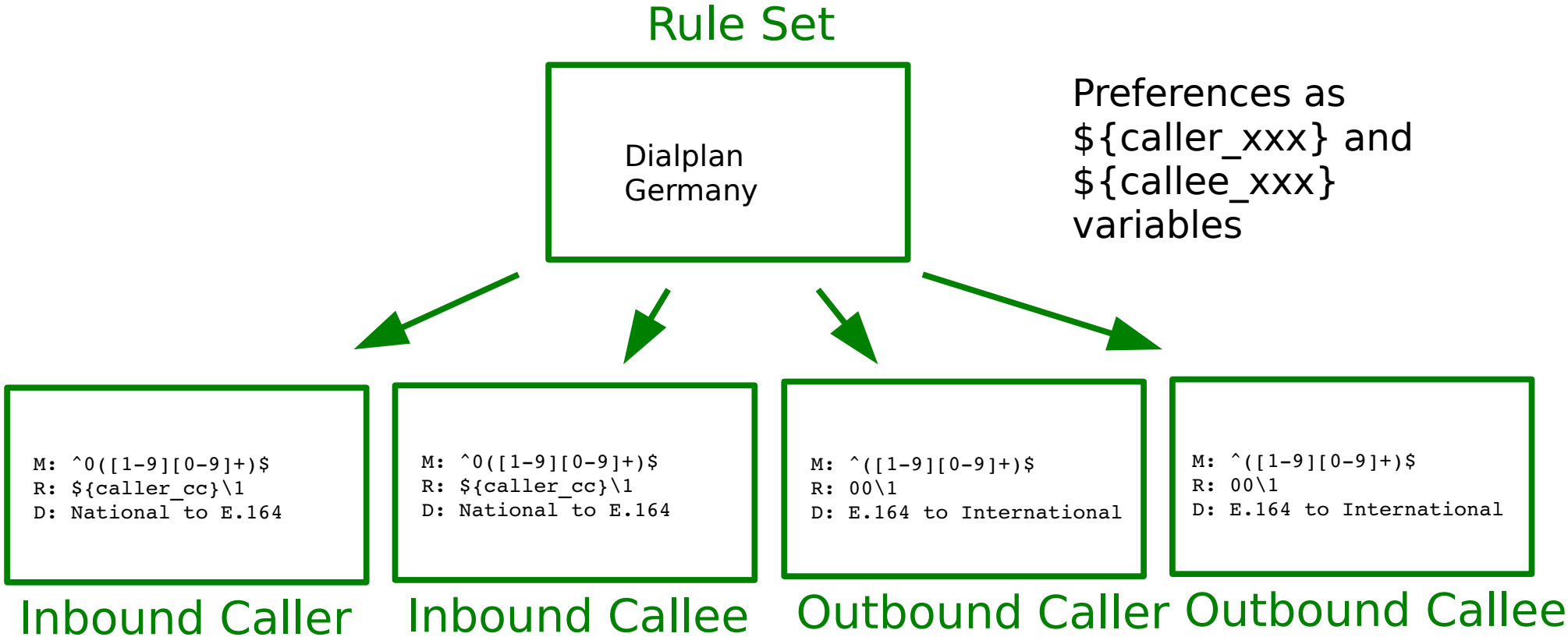
- Needs Preference Tweaking as of mr3.8.x
- WebRTC Subscribers → Details → Preferences

  → NAT and Media Flow Control
  - `use_rtpproxy: Always with rtpproxy as additional/only ICE candidate`
  - `transport_protocol: RTP/SAVPF (encrypted SRTP with RTCP feedback) - (for Chrome Version 43.0.2357.65)`
- Domain → Details → Preferences

  → NAT and Media Flow Control
  - `transport_protocol: RTP/AVP (Plain RTP)`
- Depends on your Use Case (SIP ↔ WebRTC Bridging)

Rewrite Rules

- Your Dial-Plans in Perl Regex



Creating Rewrite Rules

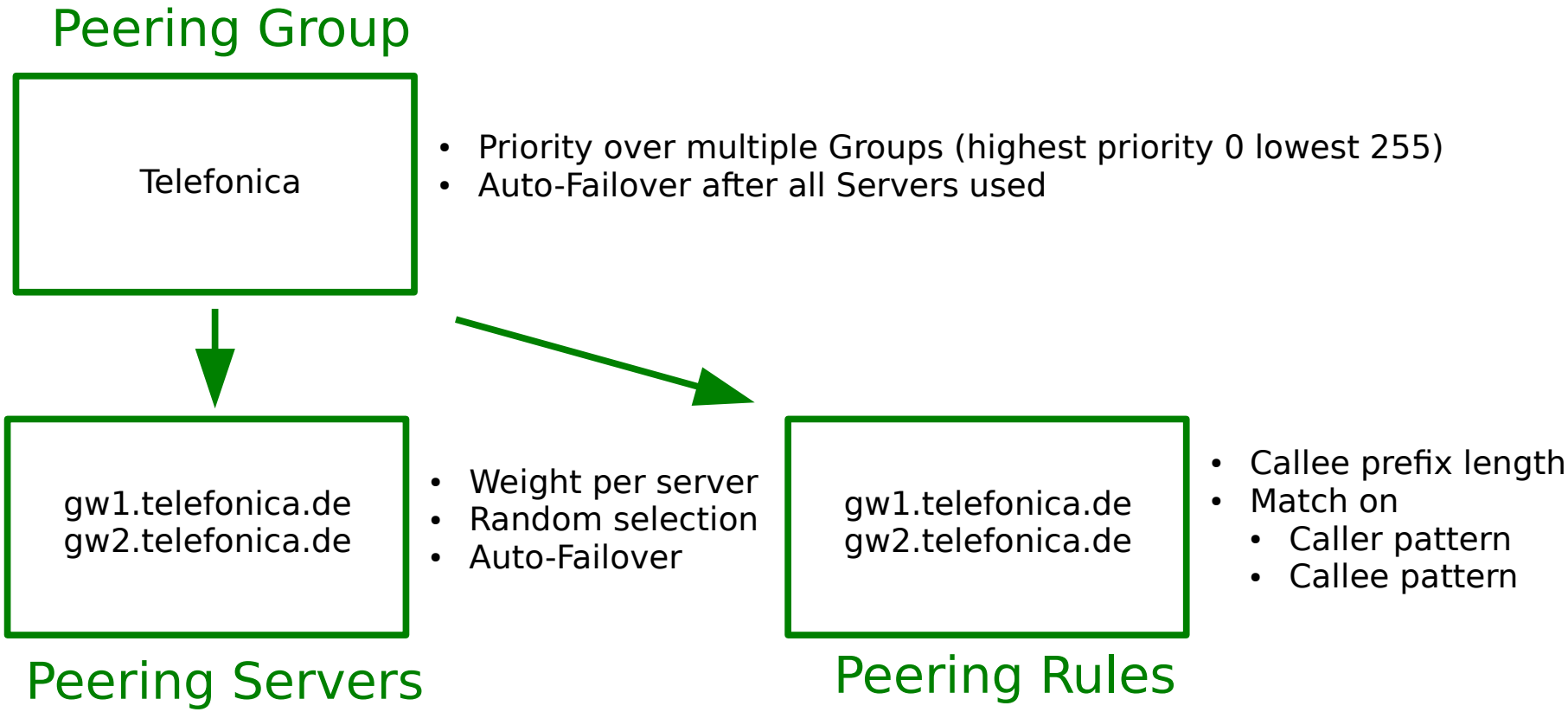
- Settings → Rewrite Rules
- Processing stops on first match (order matters)
- Assign to Subscribers, Domains and Peer
  - Preferences → rewrite_rule_set

Peerings

- Dynamic peering via ENUM
  - config.yml: kamailio → proxy → use_enum
- Dynamic peering via Foreign Domains
  - Dom-Preference: allow_out_foreign_domain
  - Dom-Preference: unauth_inbound_calls
- Static peering via Peering Groups
  - Everything not local goes to peers
  - Force inbound/outbound to peer via Preferences

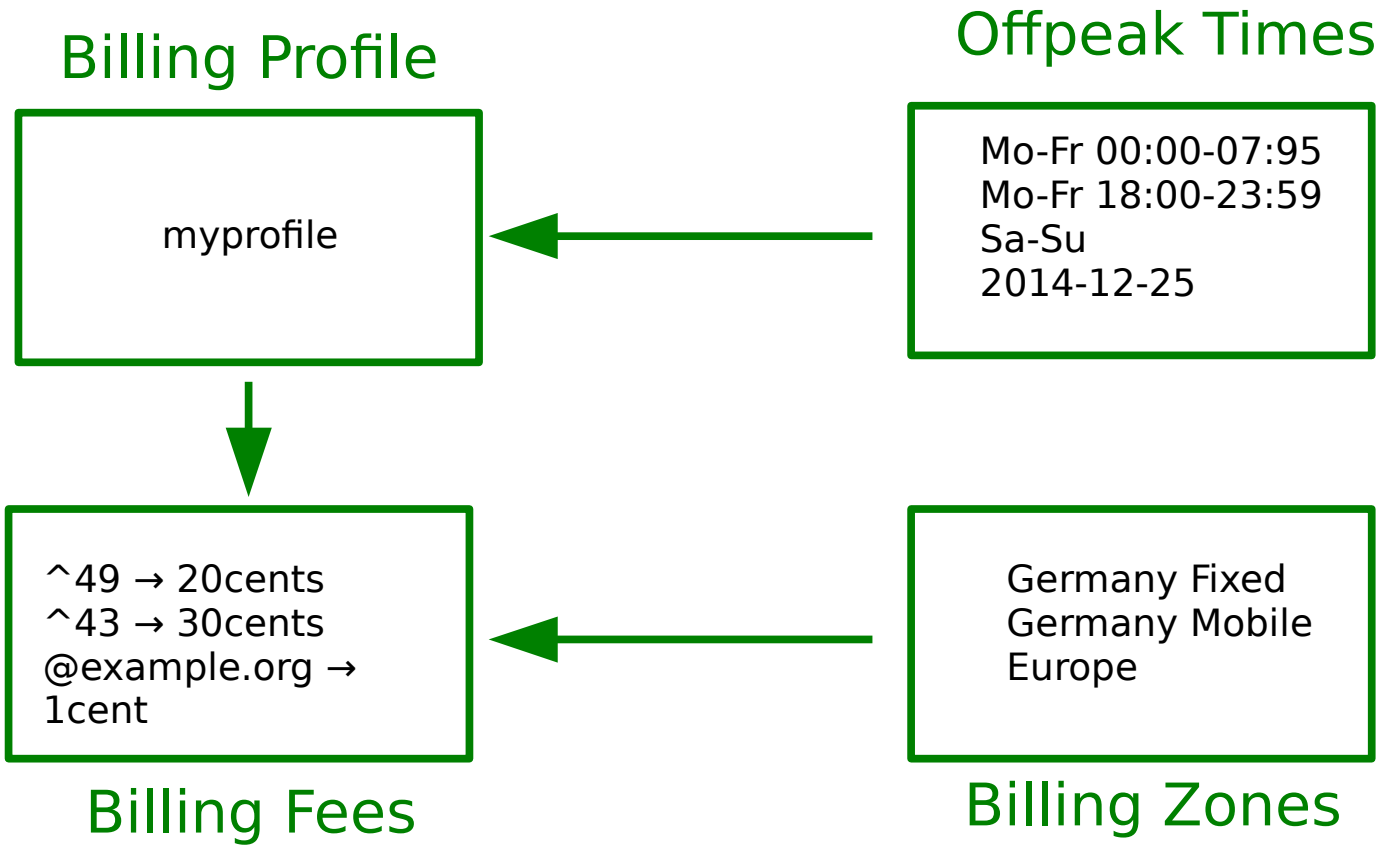
Static Peerings

- Settings → Peerings → Create Peering Group



Billing Profiles

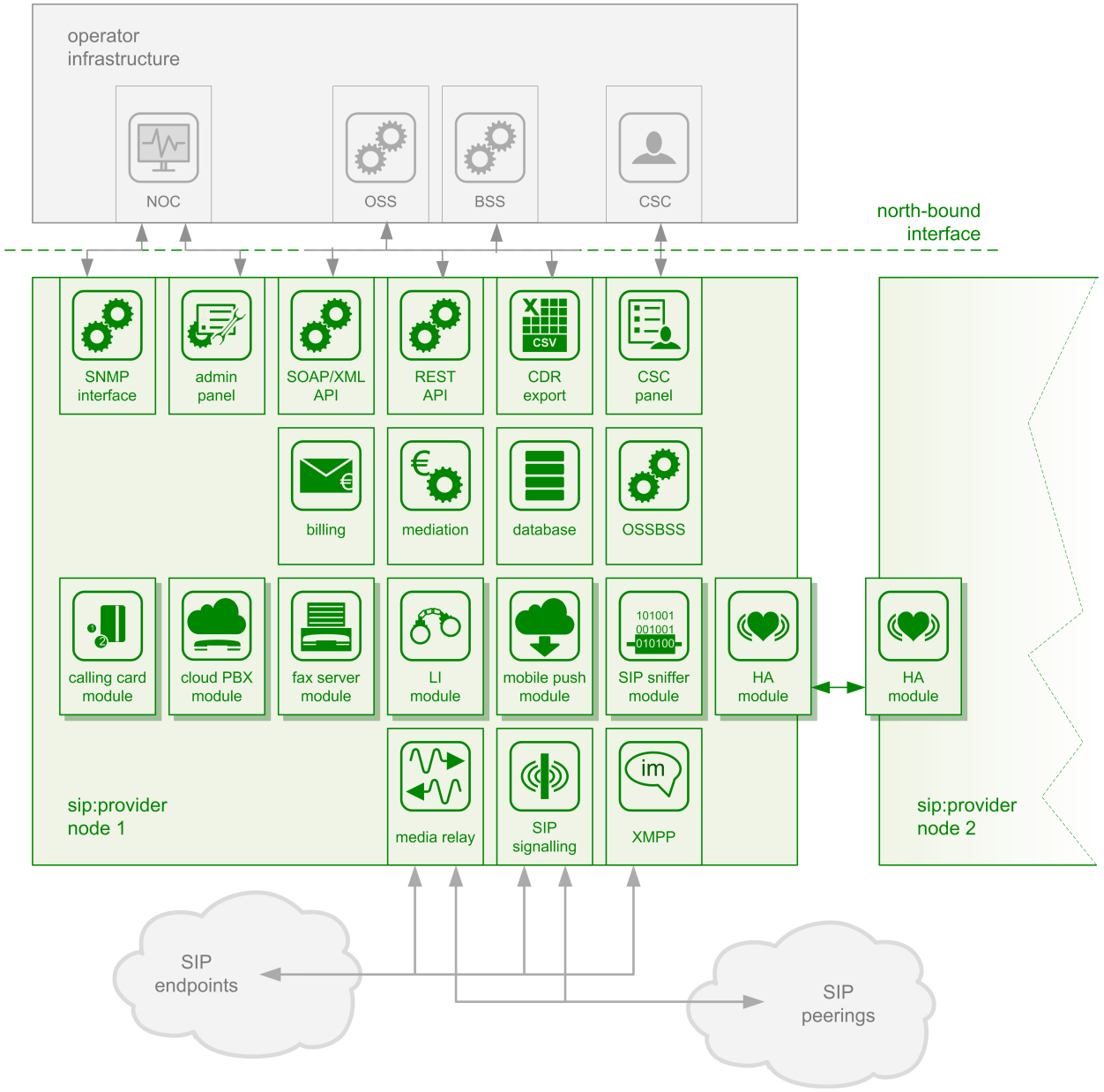
- Settings → Billing → Create Billing Profile



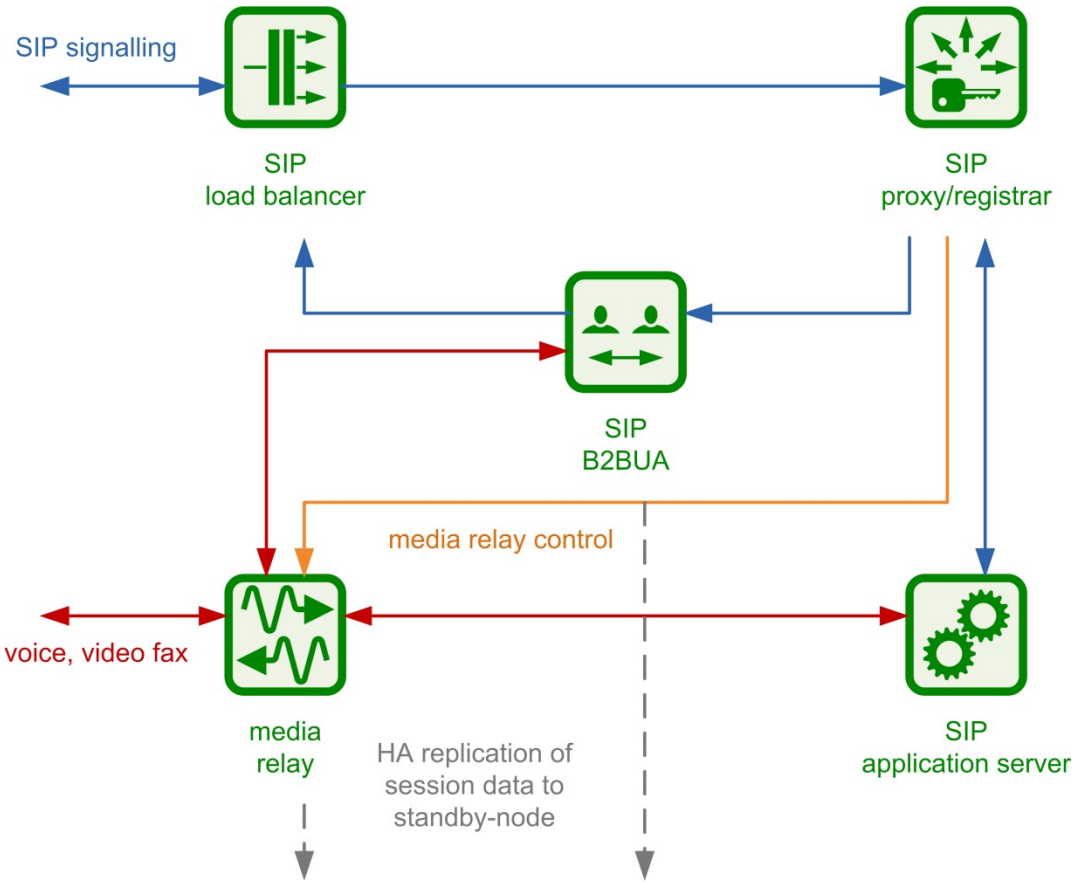
That's it for the operational part

Easy, eh?

sip:provider Architecture



SIP Components



Configuration Framework

- Templates!
- Controlled by:
  - `/etc/ngcp-config/config.yml`
  - `/etc/ngcp-config/constants.yml`
  - `/etc/ngcp-config/network.yml`
- Template sources:
  - `/etc/ngcp-config/templates/...`
- Backed by Git (but you don't use it directly)
  - `ngcpcfg framework`

Changing Configs

- Use customtt-Files!

```
$ cp \
    /etc/ngcp-config/templates/etc/kamailio/lb/kamailio.cfg.tt2 \
    /etc/ngcp-config/templates/etc/kamailio/lb/kamailio.cfg.customtt.tt2
```

- Apply changes

```
$ ngcpcfg apply
```

  - Commits your changes, generates configs and restarts affected services (neat, eh?)
- Automatically tracks everything you drop into /etc/ngcp-config/templates/

Your Use Cases?

- If we wanted to do $this...
- then you have to do $that

Your turn!

More Questions?

@spce-user mailing list

- http://lists.sipwise.com/listinfo/spce-user

@spce documentation

- https://www.sipwise.org/products/spce/documentation/