OpenSIPS 2.0

a programmable SIP framework

Bogdan-Andrei Iancu
CEO Voice System
Founder OpenSIPS Project
Once upon a time there was a powerful and flexible SIP Server ......

....there was OpenSIPS doing tens of thousands CPS.

BUT....
What to be addressed

SIP Low level awareness

• you still need to be aware of and handle low level SIP bits and pieces (transactions, dialogs, NAT, etc) to make it work

• you cannot focus only on service creation
What to be addressed

Configuration skills

- you are required to learn the custom OpenSIPS scripting language
- you are limited to what OpenSIPS script language has to offer
- the script language is not integration friendly
What to be addressed

Horizontal Scalability

- cannot scale with a single instance, no matter how powerful it is
- clustering must be naturally achieved
- traffic and data sharing across all nodes in cluster
OpenSIPS 2.0 == 42
STEP 1 – separation of SIP stack and routing logic

monolithic & mixed implementation

Routing Engine

Core SIP Stack
STEP 2 – decouple routing & core to get separated & independent applications

Routing Engine
Core SIP Stack

Routing Engine
Core SIP Stack

OpenSIPS
OpenSIPS

2.0 Genesis
Solved:

- no need to deal SIP low level, just to control and interact with it
- you can focus on service creation without taking care of SIP specific details
- achieve vertical scalability (routing logic and core may be on different machines)
- optimize the processing – the Core part (SIP stack) may automatically and transparent handle certain SIP events (like CANCEL, ACK, etc)
STEP 3 – programmable routing logic (Perl, Python, Java, etc)

Routing Engine

Python & modules

Routing Engine

Perl & libs

Routing Engine

Java & classes

Network API
Solved:

- no more custom language for scripting – you can use your own favorite language (any)

- scripting is no more limited – you can take full advantages of the capabilities (as scripting) and already existing functionality from the high-level programming languages

- integration (with whatever other apps in whatever other languages) become trivial as what language is used is no more a limitation, but rather an advantage

- routing logic can be actually part of other larger application
STEP 4 – horizontal scalability for both Core and Logic
Solved:

• horizontal scalability – each part (core and logic) may individually scale across several machines.

• the logic will be responsible for clustering (service and data) by providing to Core part data storage support

• it is cheaper to scale (for same number of CPUs) with several machines, rather than only one
Examples
Intensive traffic, simple logic

Java application for SIP trunking

Routing Engine

Network

Core SIP Stack

Core SIP Stack

Core SIP Stack
Heavy logic and integration

Routing Engine

Routing Engine

Routing Engine

Python service controller

Core SIP Stack

Network
Specialized logics

- Call Routing Engine
- Presence Routing Engine
- Messages Routing Engine

Network

Core SIP Stack
Geographical Clustering

Java abstract layer for clustering with distributed DB (P2P)

Routing Engine
Routing Engine
Routing Engine
Routing Engine

USA
Europe
Asia

Core SIP Stack
Core SIP Stack
Core SIP Stack
Thank you for your attention
You can find out more at www.opensips.org
bogdan.iancu@voice-system.ro

Questions are welcome