Carrier-Friendly OpenSER

Welcome!

Henning Westerholt

OpenSER project

1&1 Internet AG

Spring Von.x 2008, 17.03.2008

Outline

1. *scenario overview*
2. *challenges*
3. *introducing carrierroute*
4. *recent developments*
5. *how to use it by yourself*

Scenario overview

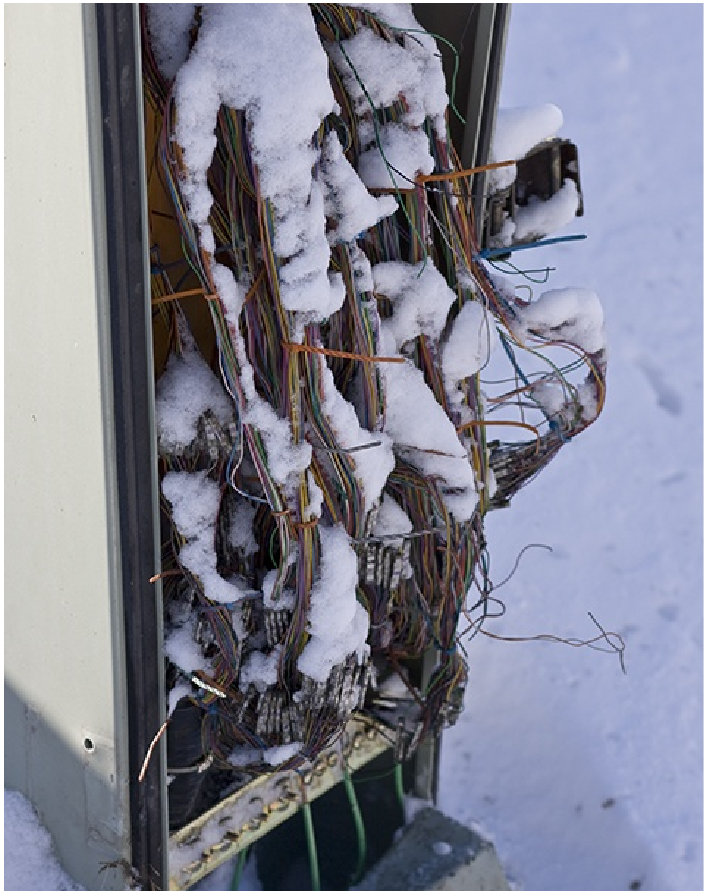
carrier infrastructure at 1&1

DSL reseller, no own network to the customer
uses network from Telekom, Telefonica, QSC..
interface to PSTN via Telefonica, Broadnet, BT..
different product lines and product flavours
(i.e. bitstream access vs. full PSTN line)

Routing problems

prefix based routing
customer specific routing
routing is different in case of failures
high number of routes

Not possible with plain OpenSER



Challenges

Flexibility

database supported routing for easier setup and maintenance
fine grained control over routing decisions
one solution for blacklisting, routing and balancing prefered

Performance

Caching of routing rules in memory
efficient datastructures for fast access

Scalability

usable with a huge number of routing rules
provide room for further extensions
move routing decisions out of the config script

Introducing carrierroute

History

originally written for OpenSER 0.9

generalized, ported and extended, contributed to 1.3

Further rework and bigger extensions to 1.4

Flexibility

Routing rules can be specified from console, config file and database

different hash sources usable, highly configurable

Performance

lookup delay under 10 µs

no real profiling has be done, its more than fast enough

short startup time (under 1 s), for big routing sets up to 20 seconds

Scalability

from a few routing rules (for balancers) up to several hundert thousand routing rules

Recent developments

Database supported failure routing

- matching to host, previous route, failure code..
- prevent re-routing to the same carrier
- handle all the little carrier differences

Better and flexible user interface

- only three functions to cover most of the use-cases
- use any data that is available as PV as hash source or user lookup

Extension of normal routing

- separation of user lookup and rewrite phase
- flag supported routing like in the failure route case

Internal structure updates and refactoring

Interface in config script

cr_route(carrier, domain, prefix_matching, rewrite_user, hash_source)

- *does the actual rewriting for the routing*
- *accept pseudo-variables for most parameters*
- *usable in combination with cr_user_carrier to get user specific routing*

cr_user_carrier(user, domain, dstavp)

- *loads the carrier for an user and stores it in an AVP*
- *not needed for a uniform user base*

cr_next_domain(carrier, domain, prefix, host, reply, flags, dstavp)

- *loads the next route after a failure has happened*
- *matches to prefix, host, reply code, flags..*
- *usage is optional, failure routing by other means still possible*

Database structure

tables

carrierroute - normal routing rules
carrierfailureroute - failure routing
route_tree - carrier to id mapping

carrieroute colums

carrier, domain, prefix, host, probability, strip
new: flags, mask

user lookup

subscriber table holds mapping from user to carrier

carrierfailureroute colums

similiar to carrierroute table
reply code, flags, mask, next domain

End of part one

More informations:

Database and configuration snipets can be found in the module documentation.
Contact: henning.westerholt@1und1.de, Project user and developer mailing list.

The next talk will focus on concepts how its possible to make the server more usable, better maintainable and increase the flexibility.

Pictures:

slide 1: Trey Ratcliff, http://www.flickr.com/people/stuckincustoms/
slide 3: Pygo, http://www.flickr.com/people/pygo/
slide 6: Marko Kivelä, http://www.flickr.com/people/marko_k/
slide 8: Jeff Kubina, http://www.flickr.com/people/kubina/

Licence of the slides: cc BY NC SA