Cost-based LCR for OpenSIPS using CGRateS

Dan Christian Bogos
dan.bogos@itsyscom.com

OpenSIPS Summit Amsterdam 2016

Our Background

Located in Bavaria/Germany, over 9 years of experience with architecting server side solutions in VoIP environment

Platform implementations covering both wholesale and retail business categories

Responsibly understanding real-time processing constrains and the seriousness of live system outages

About CGRateS

Charging/Billing engine
Plug-able into existing billing infrastructure
Accommodate new components into ISP/ITSP network (eg: add new VoIP switch, SMS Service, Data stream)
Non-intrussive into existing setups

Modular architecture
Easy to enhance by rewriting specific components - JSON/HTTP/GOB RPC API

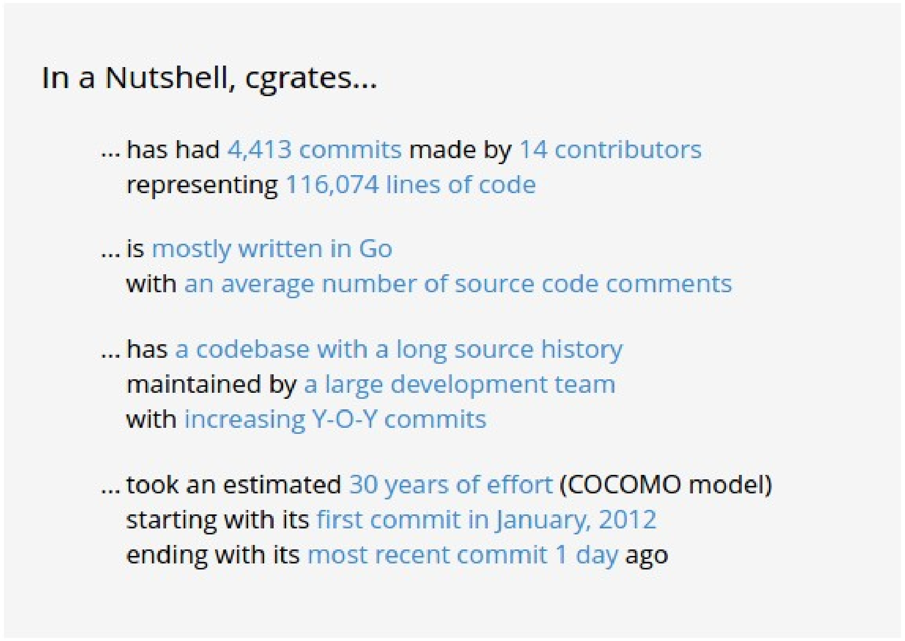
Performance Oriented
Built-in transactional cache system (data ageing, live counters)
Asynchronous processing with micro-threads

Feature-rich
Multi-tenancy, derived charging, account bundles, LCR, CDRStats, rates history, etc
Agile in developing new features

Test driven development
Aprox. 1200 tests as part of the build system

CGRATES
carrier grade realtime charging

About CGRateS (2)



In a Nutshell, cgrates...

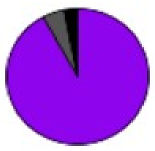
... has had 4,413 commits made by 14 contributors
representing 116,074 lines of code

... is mostly written in Go
with an average number of source code comments

... has a codebase with a long source history
maintained by a large development team
with increasing Y-O-Y commits

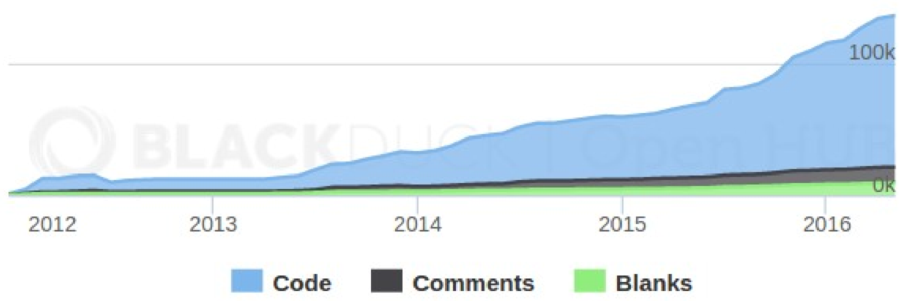
... took an estimated 30 years of effort (COCOMO model)
starting with its first commit in January, 2012
ending with its most recent commit 1 day ago

Languages



Go	92%		XML	5%
7 Other	3%			

Lines of Code

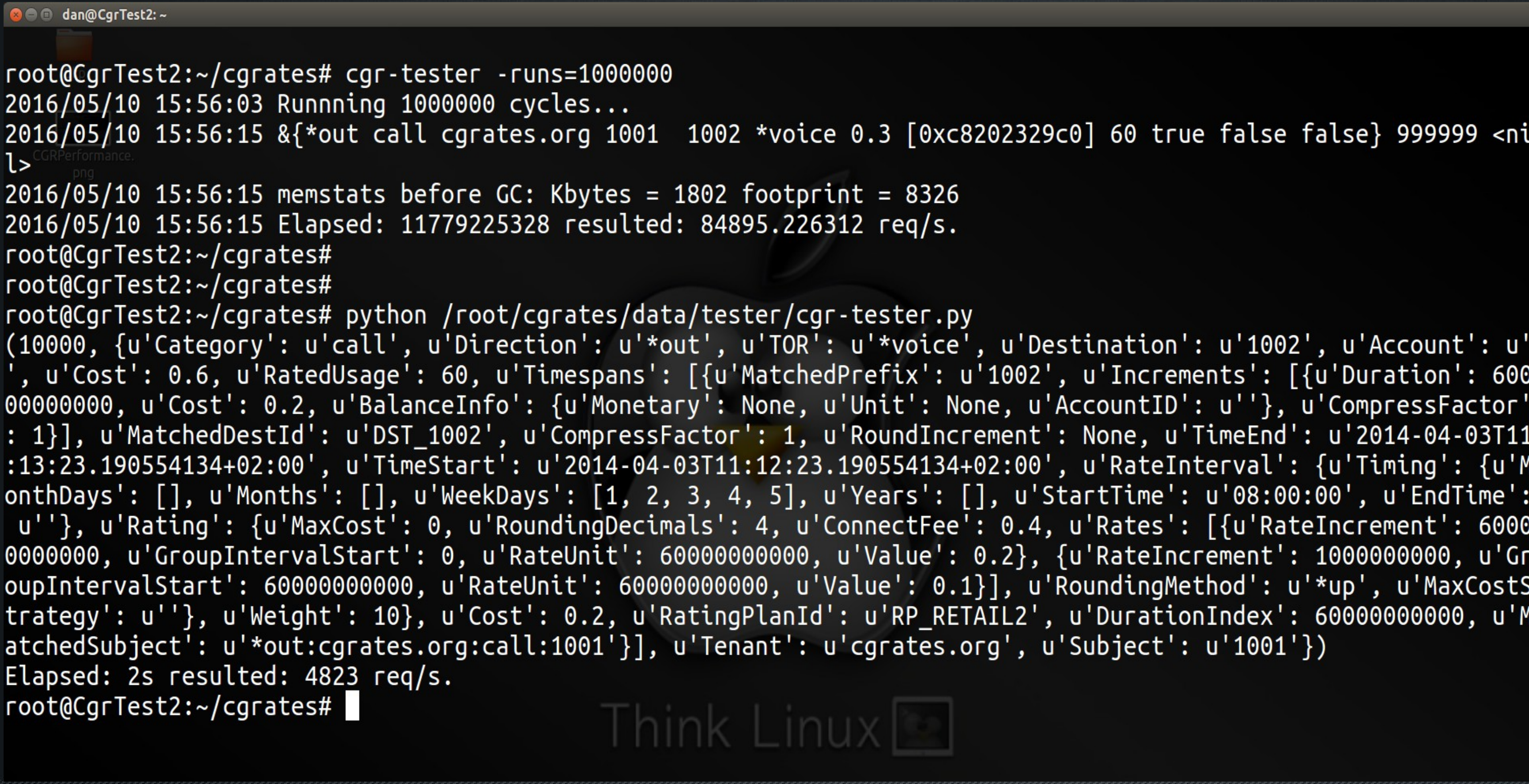


Actively maintained

*stats provided by openhub.net

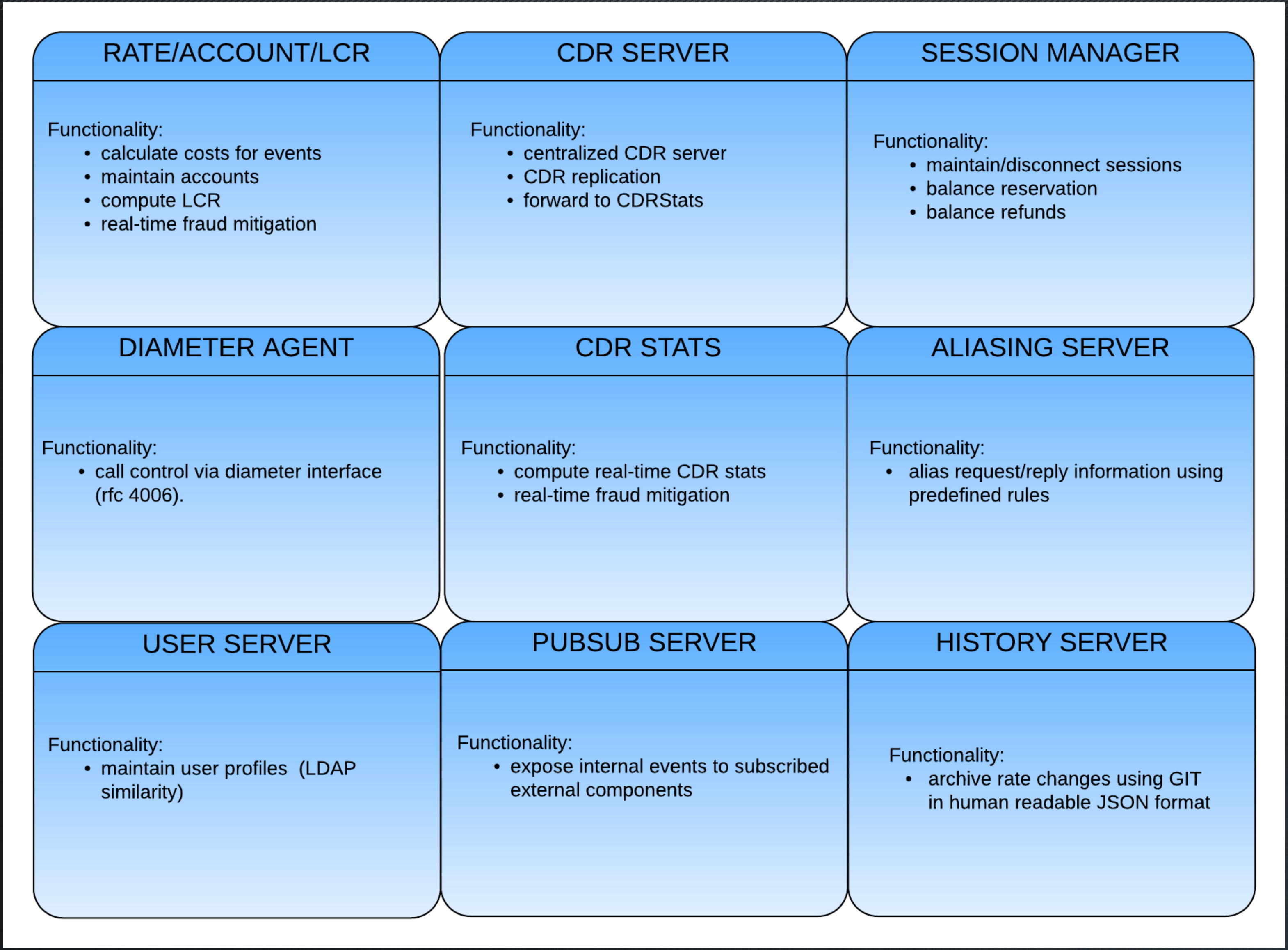
carrier grade realtime charging

About CGRateS (3)



```
root@CgrTest2:~/cgrates# cgr-tester -runs=1000000
2016/05/10 15:56:03 Runnning 1000000 cycles...
2016/05/10 15:56:15 &{*out call cgrates.org 1001  1002 *voice 0.3 [0xc8202329c0] 60 true false false} 999999 <ni
l>
2016/05/10 15:56:15 memstats before GC: Kbytes = 1802 footprint = 8326
2016/05/10 15:56:15 Elapsed: 11779225328 resulted: 84895.226312 req/s.
root@CgrTest2:~/cgrates#
root@CgrTest2:~/cgrates#
root@CgrTest2:~/cgrates# python /root/cgrates/data/tester/cgr-tester.py
(10000, {u'Category': u'call', u'Direction': u'*out', u'TOR': u'*voice', u'Destination': u'1002', u'Account': u'
', u'Cost': 0.6, u'RatedUsage': 60, u'Timespans': [{u'MatchedPrefix': u'1002', u'Increments': [{u'Duration': 600
00000000, u'Cost': 0.2, u'BalanceInfo': {u'Monetary': None, u'Unit': None, u'AccountID': u''}, u'CompressFactor'
: 1}], u'MatchedDestId': u'DST_1002', u'CompressFactor': 1, u'RoundIncrement': None, u'TimeEnd': u'2014-04-03T11
:13:23.190554134+02:00', u'TimeStart': u'2014-04-03T11:12:23.190554134+02:00', u'RateInterval': {u'Timing': {u'M
onthDays': [], u'Months': [], u'WeekDays': [1, 2, 3, 4, 5], u'Years': [], u'StartTime': u'08:00:00', u'EndTime':
 u''}, u'Rating': {u'MaxCost': 0, u'RoundingDecimals': 4, u'ConnectFee': 0.4, u'Rates': [{u'RateIncrement': 6000
0000000, u'GroupIntervalStart': 0, u'RateUnit': 60000000000, u'Value': 0.2}, {u'RateIncrement': 1000000000, u'Gr
oupIntervalStart': 60000000000, u'RateUnit': 60000000000, u'Value': 0.1}], u'RoundingMethod': u'*up', u'MaxCostS
trategy': u''}, u'Weight': 10}, u'Cost': 0.2, u'RatingPlanId': u'RP_RETAIL2', u'DurationIndex': 60000000000, u'M
atchedSubject': u'*out:cgrates.org:call:1001'}], u'Tenant': u'cgrates.org', u'Subject': u'1001'})
Elapsed: 2s resulted: 4823 req/s.
root@CgrTest2:~/cgrates#
```

Fast and … very fast



RATE/ACCOUNT/LCR

Functionality:
- calculate costs for events
- maintain accounts
- compute LCR
- real-time fraud mitigation

CDR SERVER

Functionality:
- centralized CDR server
- CDR replication
- forward to CDRStats

SESSION MANAGER

Functionality:
- maintain/disconnect sessions
- balance reservation
- balance refunds

DIAMETER AGENT

Functionality:
- call control via diameter interface (rfc 4006).

CDR STATS

Functionality:
- compute real-time CDR stats
- real-time fraud mitigation

ALIASING SERVER

Functionality:
- alias request/reply information using predefined rules

USER SERVER

Functionality:
- maintain user profiles (LDAP similarity)

PUBSUB SERVER

Functionality:
- expose internal events to subscribed external components

HISTORY SERVER

Functionality:
- archive rate changes using GIT in human readable JSON format

CGRateS subsystems

CGR-RALs (Rating)

Highly configurable rating

Connect fees, rate units, rate increments, rates grouping, a-number rating, various rounding methods, configurable decimals in costs, maximum cost per destination with hit strategy, rating profile scheduling

Multiple TypeOfRecord support

(eg: *voice, *data, *sms, *mms, *monetary, *generic)

Multiple Category filters for same TOR

(eg: calls, premium_calls, inbound_calls)

Derived Charging

Reseller/distributors chaining or inbound/outbound traffic charging

CGR-RALs (Accounting)

Prepaid, Postpaid, Pseudo-prepaid controller

Unlimited Balances per Account

*voice, *data, *sms, *mms, *monetary, *generic
Balance selection prioritisation through weights
Unlimited bundle combinations

Shared/Group Balances

Balance lifetime controls

Eg: balance expires or balance is active on specific time intervals

Concurrent sessions per account

Balance reservation in chunks of debit interval
Balance refunds
Debit sleep when needed

FRAUD MITIGATION

Part of Accounting

Tightly integrated, balance operations cannot avoid it
Minimum & maximum balance monitors
Minimum & maximum counter monitors

Part of CDRStats

Multiple metrics and stat queues thresholds

Scheduler integration

One-time, recurrent triggers

Synchronous & Asynchronous Actions

CDR SERVER

Realtime CDR Server
Accessible Internal, GOB, JSON, HTTP-JSON, HTTP-REST interfaces

Offline CDR Import (csv, xml, fwv)
Automated via Linux inotify or scheduled
Simultaneous folders monitored with multiple import templates per folder

Zero configuration CDR Sources
FreeSWITCH
Kamailio
OpenSIPS

CGRATES
carrier grade realtime charging

CDR SERVER (2)

Derived Charging support

Real-time CDR replication
Raw or Rated CDRs

CDR Exporter
CSV, Fixed Length Fields, Combined
Export templates

CDR STATS

Standalone component
Internally or remotely accessible
Performance oriented

RawCDR and RatedCDR sources

Multiple Stats Queues
Per server and individually configurable stat queues for same CDR

Highly configurable Stats Queues
QueueLength, TimeWindow, Metrics
CDR Field Filters

Individually configured ActionTriggers
One-time, recurrent triggers
Synchronous & Asynchronous Actions executed
Part of the Fraud Detection mechanism

CGR-RALs (LCR)

Core component logic
Internally or remotely accessible through APIer or RATER components
Non-intrusive, injects supplier information into Telecom Switch

Tightly coupled with ACCOUNTING subsystem
Provides LCR over bundles

Integrates traffic patterns
Computes LCR for specific call duration

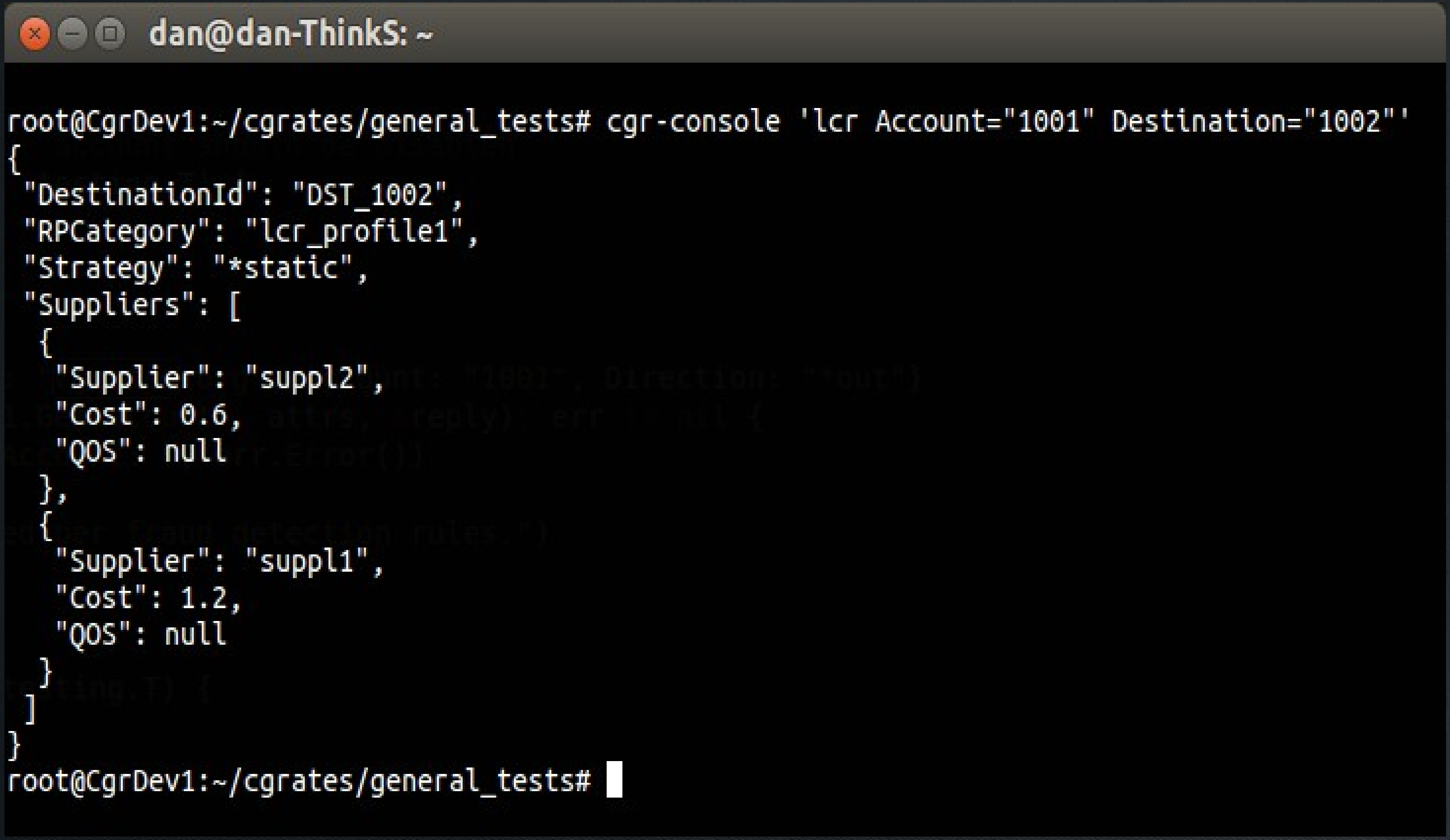
Advanced profile selection mechanism
Filter on Direction,Tenant,Category,Account,Subject/CLI prefix,Destination
Weight based prioritization
Activation time

Extended functionality through multiple strategies
*static, *least_cost, *highest_cost, *qos_thresholds, *qos, *load_distribution
Flexible strategy parameters

*static

Classic way of LCR, suppliers ordered based on configured rule parameters

“*out,cgrates.org,call,1001,*any,DST_1002,lcr_profile1,*static,suppl2;suppl1,2014-01-14T00:00:00Z,10”



dan@dan-ThinkS: ~

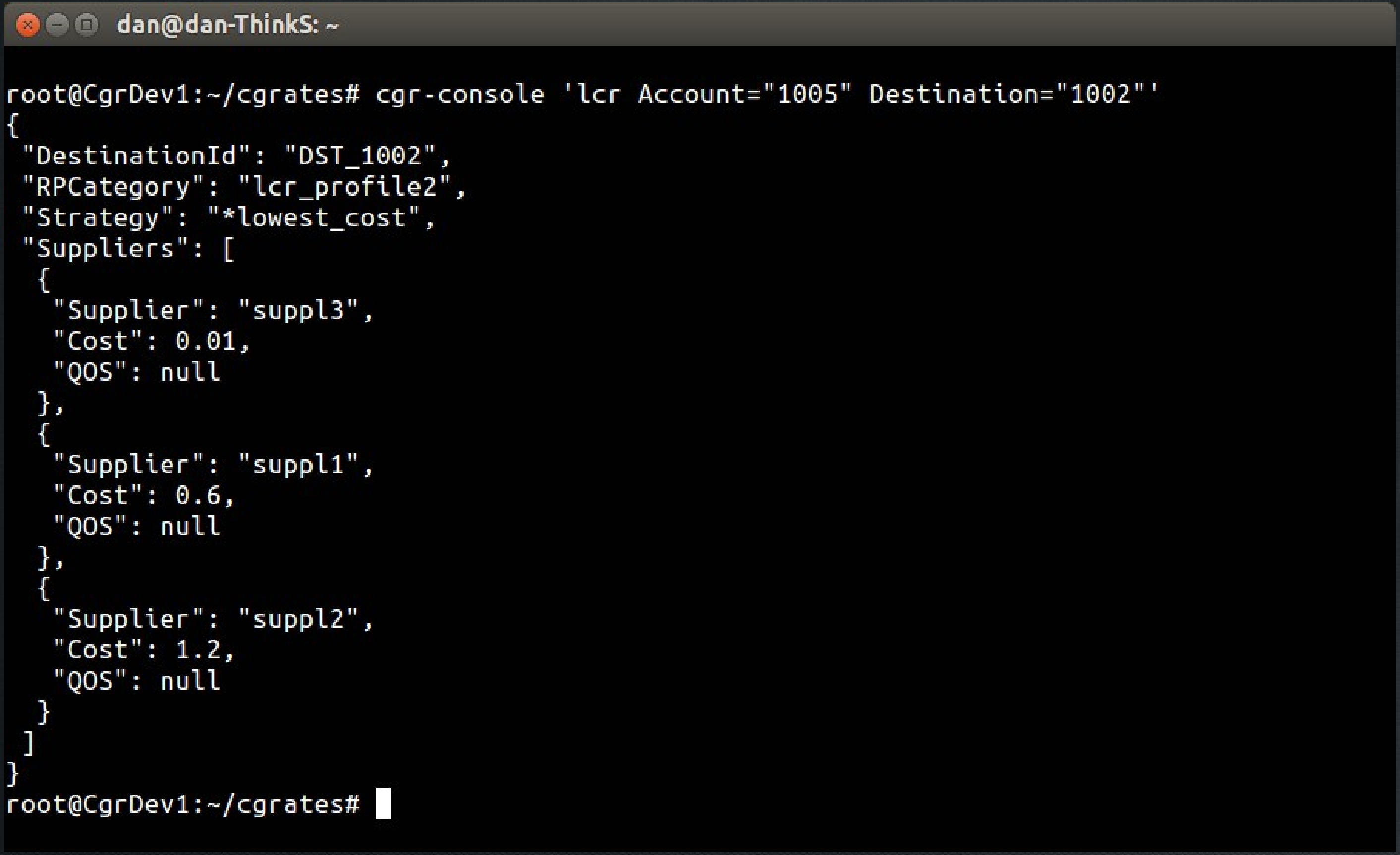
```
root@CgrDev1:~/cgrates/general_tests# cgr-console 'lcr Account="1001" Destination="1002"'
{
 "DestinationId": "DST_1002",
 "RPCategory": "lcr_profile1",
 "Strategy": "*static",
 "Suppliers": [
  {
   "Supplier": "suppl2",
   "Cost": 0.6,
   "QOS": null
  },
  {
   "Supplier": "suppl1",
   "Cost": 1.2,
   "QOS": null
  }
 ]
}
root@CgrDev1:~/cgrates/general_tests#
```

LCR Strategies (1)

*lowest_cost

Use supplier with least cost

“*out,cgrates.org,call,*any,*any,*any,lcr_profile1,*lowest_cost,,2014-01-14T00:00:00Z,10”



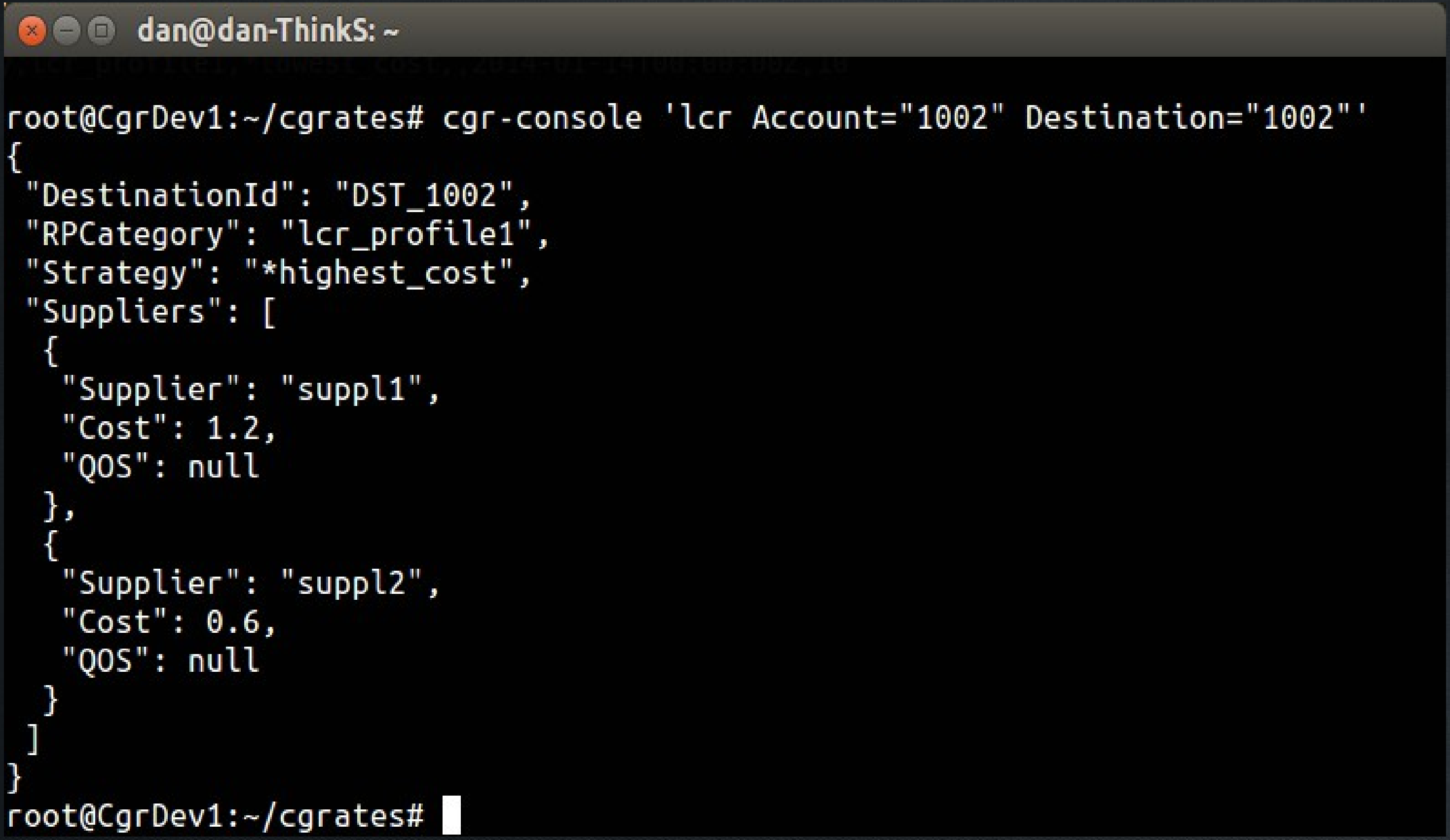
```
root@CgrDev1:~/cgrates# cgr-console 'lcr Account="1005" Destination="1002"'
{
 "DestinationId": "DST_1002",
 "RPCategory": "lcr_profile2",
 "Strategy": "*lowest_cost",
 "Suppliers": [
  {
   "Supplier": "suppl3",
   "Cost": 0.01,
   "QOS": null
  },
  {
   "Supplier": "suppl1",
   "Cost": 0.6,
   "QOS": null
  },
  {
   "Supplier": "suppl2",
   "Cost": 1.2,
   "QOS": null
  }
 ]
}
root@CgrDev1:~/cgrates#
```

LCR Strategies (2)

*highest_cost

Use supplier with highest cost

`“*out,cgrates.org,call,1002,*any,DST_1002,lcr_profile1,*highest_cost,,2014-01-14T00:00:00Z,10”`


dan@dan-ThinkS: ~

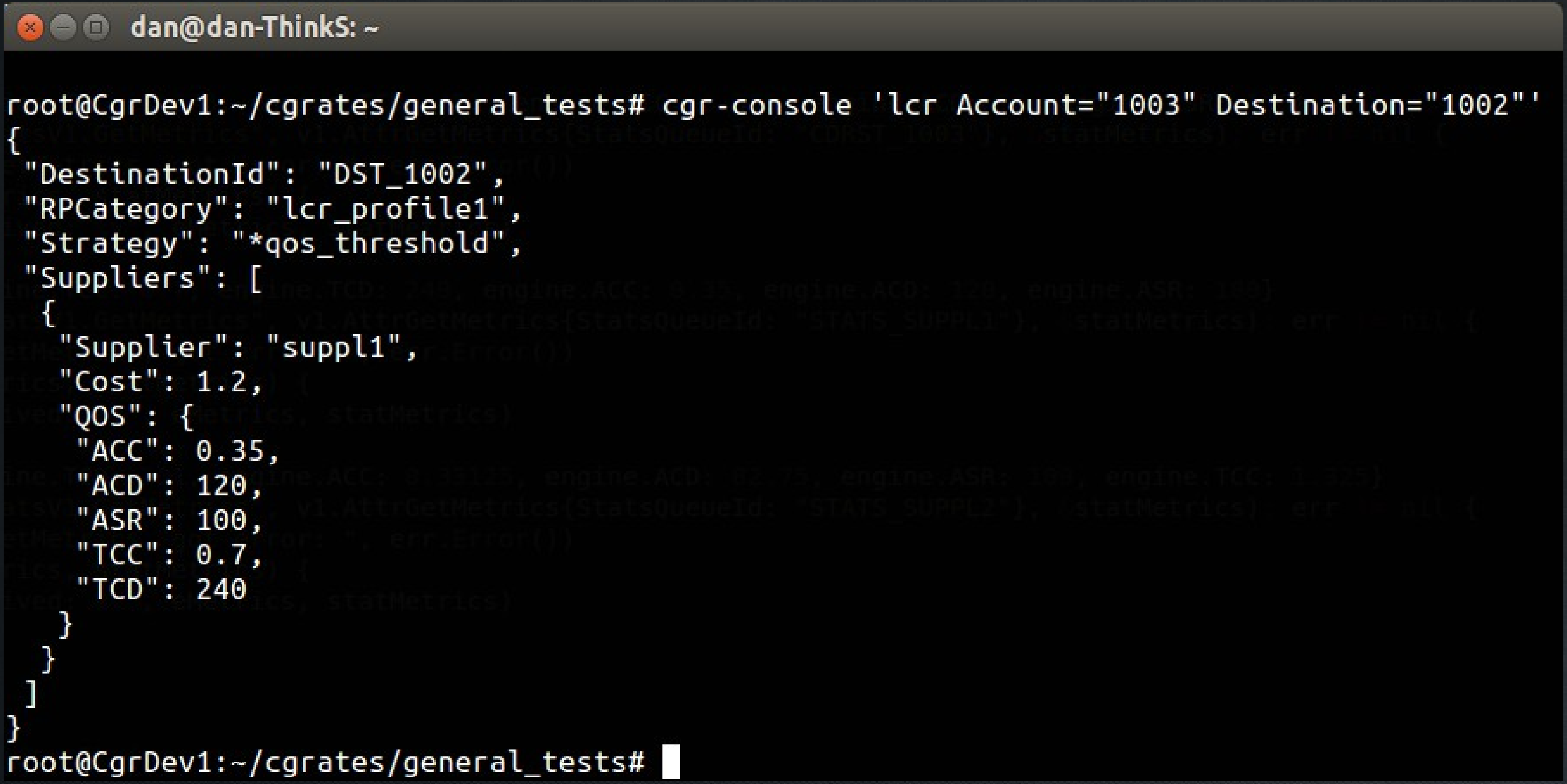
```
root@CgrDev1:~/cgrates# cgr-console 'lcr Account="1002" Destination="1002"'
{
 "DestinationId": "DST_1002",
 "RPCategory": "lcr_profile1",
 "Strategy": "*highest_cost",
 "Suppliers": [
  {
   "Supplier": "suppl1",
   "Cost": 1.2,
   "QOS": null
  },
  {
   "Supplier": "suppl2",
   "Cost": 0.6,
   "QOS": null
  }
 ]
}
root@CgrDev1:~/cgrates#
```

LCR Strategies (3)

*qos_threshold

Supplier with lowest cost, matching QoS thresholds min/max for ASR, ACD, TCD, ACC, TCC

“*out,cgrates.org,call,1003,*any,DST_1002,lcr_profile1,*qos_threshold,20;;2m;;;;;;;,2014-01-14T00:00:00Z,10”



dan@dan-ThinkS: ~

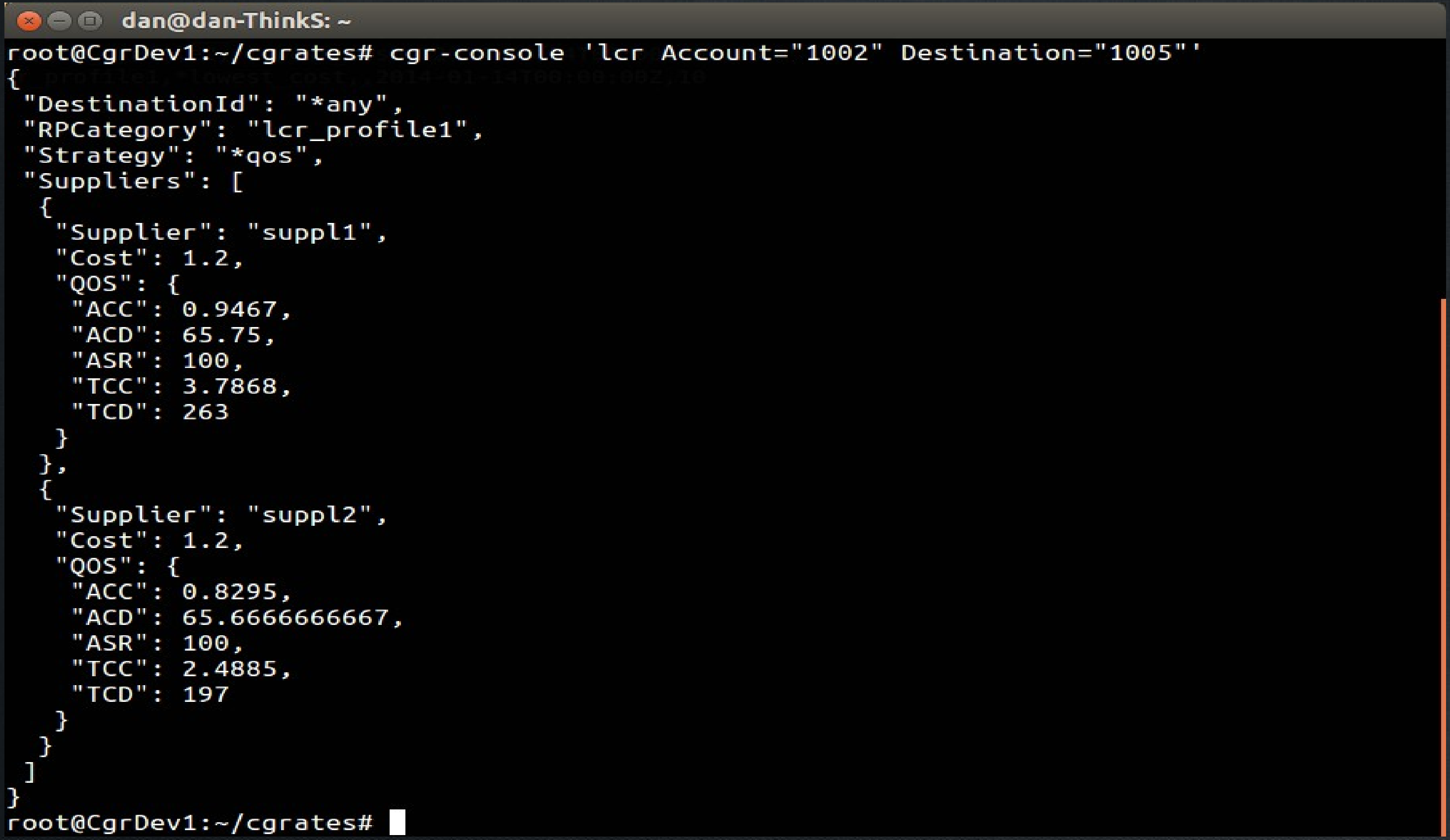
```
root@CgrDev1:~/cgrates/general_tests# cgr-console 'lcr Account="1003" Destination="1002"'
{
 "DestinationId": "DST_1002",
 "RPCategory": "lcr_profile1",
 "Strategy": "*qos_threshold",
 "Suppliers": [
  {
   "Supplier": "suppl1",
   "Cost": 1.2,
   "QOS": {
     "ACC": 0.35,
     "ACD": 120,
     "ASR": 100,
     "TCC": 0.7,
     "TCD": 240
   }
  }
 ]
}
root@CgrDev1:~/cgrates/general_tests#
```

LCR Strategies (4)

*qos

Supplier with best quality, independent on cost

“*out,cgrates.org,call,1002,*any,*any,lcr_profile1,*qos,,2014-01-14T00:00:00Z,10”



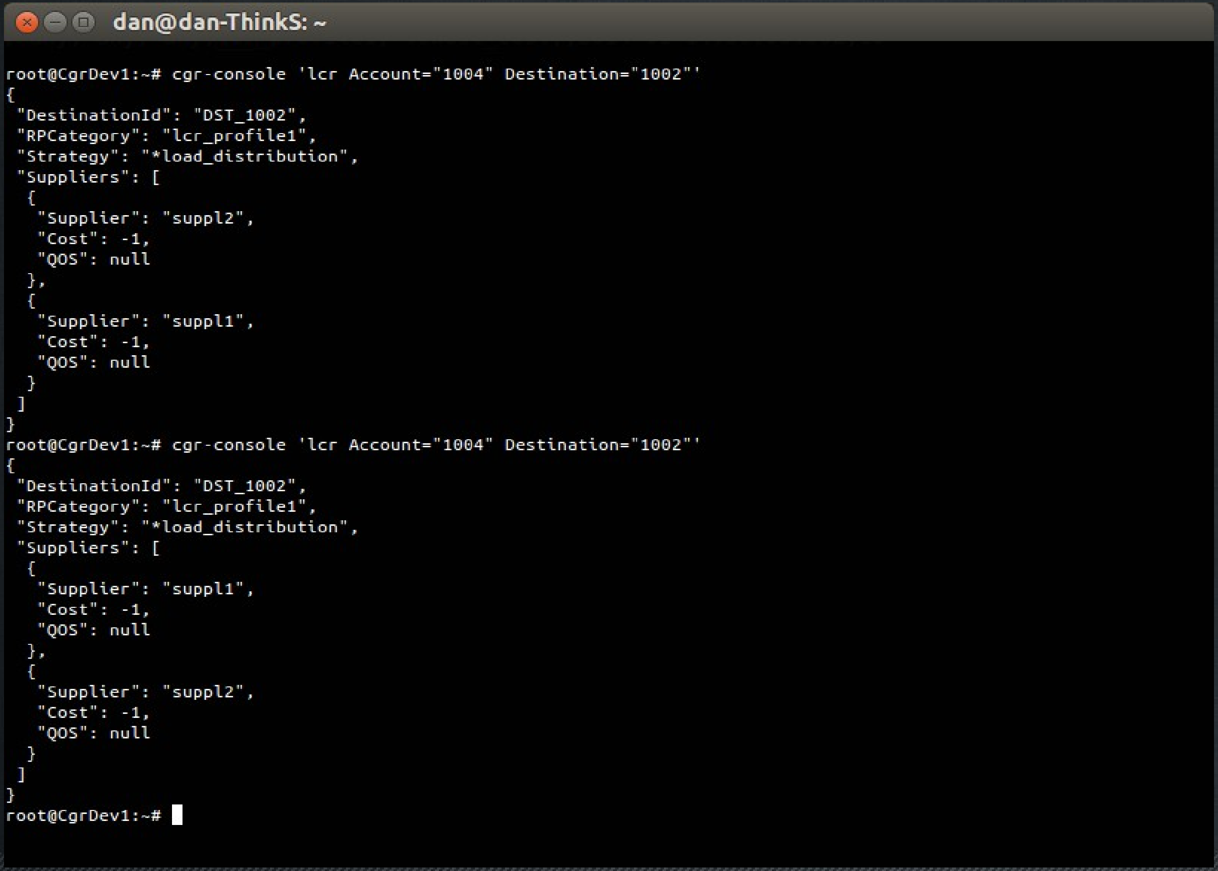
```
root@CgrDev1:~/cgrates# cgr-console 'lcr Account="1002" Destination="1005"'
{
 "DestinationId": "*any",
 "RPCategory": "lcr_profile1",
 "Strategy": "*qos",
 "Suppliers": [
  {
   "Supplier": "suppl1",
   "Cost": 1.2,
   "QOS": {
    "ACC": 0.9467,
    "ACD": 65.75,
    "ASR": 100,
    "TCC": 3.7868,
    "TCD": 263
   }
  },
  {
   "Supplier": "suppl2",
   "Cost": 1.2,
   "QOS": {
    "ACC": 0.8295,
    "ACD": 65.6666666667,
    "ASR": 100,
    "TCC": 2.4885,
    "TCD": 197
   }
  }
 ]
}
root@CgrDev1:~/cgrates#
```

LCR Strategies (5)

*load_distribution

Load based results resulted, configurable supplier ratios, individually or server defaults

“*out,cgrates.org,call,1004,*any,DST_1002,lcr_profile1,*load_distribution,supplier1:5;supplier2:3;*default:1,2014-01-14T00:00:00Z,10”



dan@dan-ThinkS: ~

```
root@CgrDev1:~# cgr-console 'lcr Account="1004" Destination="1002"'
{
 "DestinationId": "DST_1002",
 "RPCategory": "lcr_profile1",
 "Strategy": "*load_distribution",
 "Suppliers": [
  {
   "Supplier": "suppl2",
   "Cost": -1,
   "QOS": null
  },
  {
   "Supplier": "suppl1",
   "Cost": -1,
   "QOS": null
  }
 ]
}
root@CgrDev1:~# cgr-console 'lcr Account="1004" Destination="1002"'
{
 "DestinationId": "DST_1002",
 "RPCategory": "lcr_profile1",
 "Strategy": "*load_distribution",
 "Suppliers": [
  {
   "Supplier": "suppl1",
   "Cost": -1,
   "QOS": null
  },
  {
   "Supplier": "suppl2",
   "Cost": -1,
   "QOS": null
  }
 ]
}
root@CgrDev1:~# 
```

LCR Strategies (6)

CGRATES carrier grade realtime charging

OpenSIPS Integration

Multiple integration mechanisms
Based on traffic profile
Shared data through pseudovariables

REST_CLIENT for call authorization, LCR
HTTP-JSON RPC Request/Answer

EVI ACC_ACCOUNTING
*prepaid, *pseudoprepaid, *postpaid, *rated

EVI E_ACC_CDR
*pseudoprepaid, *postpaid, *rated

OpenSIPS Integration (2)

MI_DATAGRAM
Ability to tear-down calls in real-time via *dlg_end_dlg*

CDR.csv processing (db_flatfile)
*pseudoprepaid, *postpaid, *rated
caching module for CDR split over multiple files

Integration tutorial available
http://cgrates.readthedocs.io/en/latest/tut_opensips.html

Questions?

Website
http://www.cgrates.org

Documentation
http://cgrates.readthedocs.org

Code + issues tracker
https://github.com/cgrates/cgrates

Support
Google group: CGRateS
IRC Freenode: #cgrates

carrier grade realtime charging