White-paper on Workload Management available for download:
http://www.pythian.com/library/papers_and_books/rac_workload/

Demystifying Oracle RAC Workload Management

North California Oracle User Group
20-May-2010
Alex Gorbachev

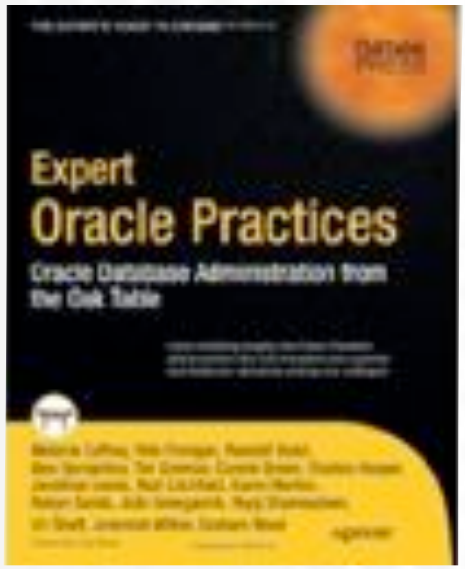
Pythian
love your data

Alex Gorbachev

ORACLE RACSIG

- CTO, The Pythian Group
- Blogger
- OakTable Network member
- Oracle ACE Director
- BattleAgainstAnyGuess.com
- Vice-president, Oracle RAC SIG

Sydney ORACLE Meetup



Oracle ACE

OakTable.net

Why Companies Trust Pythian

- Recognized Leader:
  - Global industry-leader in remote database administration services and consulting for Oracle, Oracle Applications, MySQL and SQL Server
  - Work with over 150 multinational companies such as Forbes.com, Fox Interactive media, and MDS Inc. to help manage their complex IT deployments
- Expertise:
  - One of the world's largest concentrations of dedicated, full-time DBA expertise.
- Global Reach & Scalability:
  - 24/7/365 global remote support for DBA and consulting, systems administration, special projects or emergency response

bridgewater SYSTEMS

TOYOTA

MDS Science advancing health

register.com

ORACLE Platinum Partner

Penn

FOX SPORTS

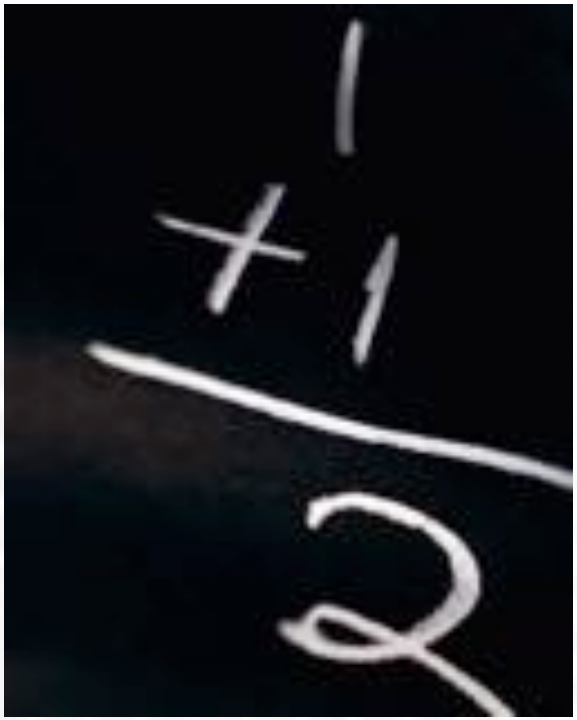
MySQL Enterprise Authorized Platinum REMOTE DBA SERVICES PARTNER

Where we are...

➡Client-side basics

- Oracle Cluster Services
- Server-side CLB
- FAN - HA & LBA events
- Thin JDBC RLB
- ONS Subscriber
- Your own LBA!

Simple example



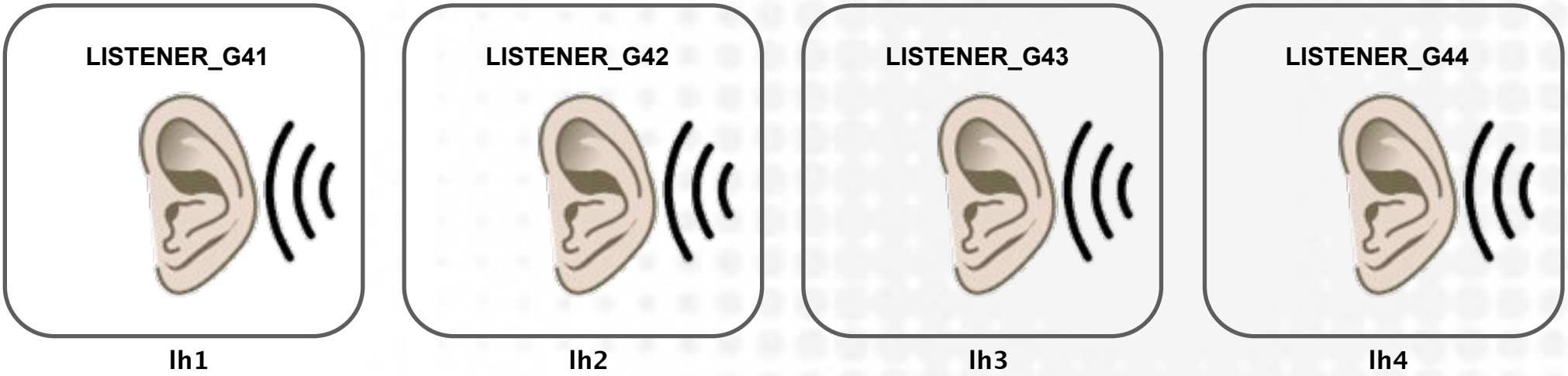
- Establishing one connection
- Dedicated server
- Several RAC nodes

Client-side CLB

```
(DESCRIPTION=
  (ADDRESS_LIST=
    (LOAD_BALANCE=ON)  (FAILOVER=ON)
    (ADDRESS=(PROTOCOL=TCP)(HOST=lh1-vip)(PORT=1521))
    (ADDRESS=(PROTOCOL=TCP)(HOST=lh2-vip)(PORT=1521))
    (ADDRESS=(PROTOCOL=TCP)(HOST=lh3-vip)(PORT=1521))
    (ADDRESS=(PROTOCOL=TCP)(HOST=lh4-vip)(PORT=1521))
   )
     (CONNECT_DATA=(SERVICE_NAME=service10g))
   )
```

Client-side connection balancing and connection-time failover

```
lh1-vip
lh2-vip
lh3-vip
lh4-vip
```



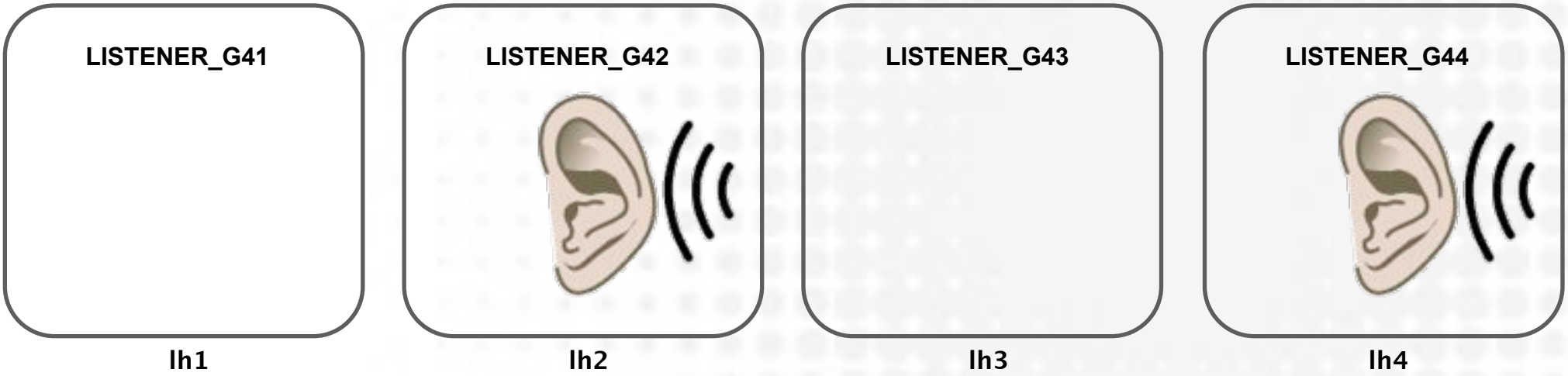
Client-side connection balancing and connection-time failover

```
lh1-vip
lh2-vip
lh3-vip
lh4-vip
```

Client-side connection balancing and connection-time failover

```
LOAD_BALANCE=OFF
FAILOVER=OFF
```

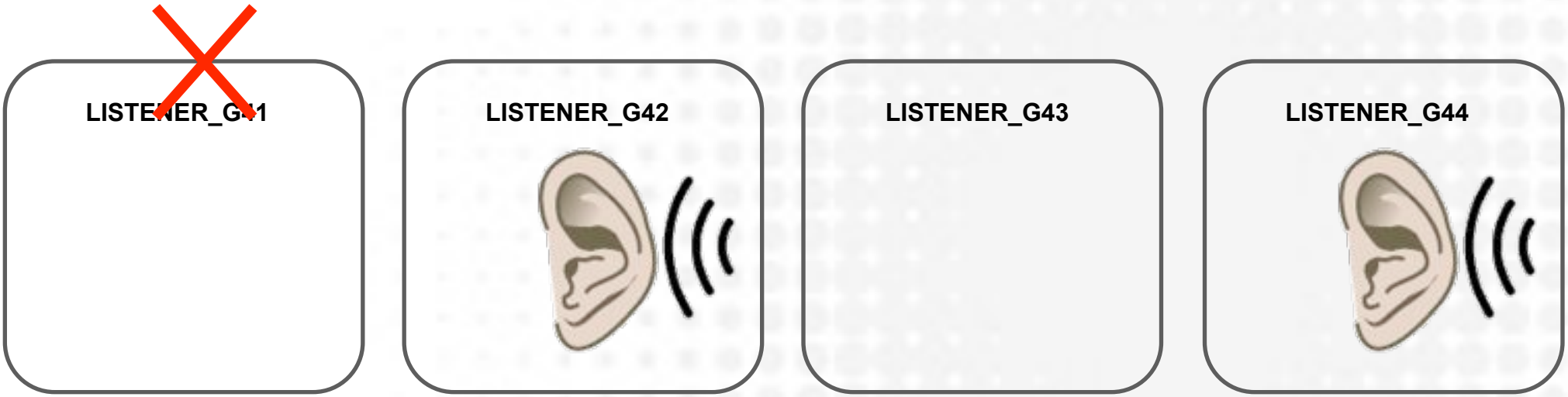
```
lh1-vip
lh2-vip
lh3-vip
lh4-vip
```



Client-side connection balancing and connection-time failover

```
LOAD_BALANCE=OFF
FAILOVER=OFF
```

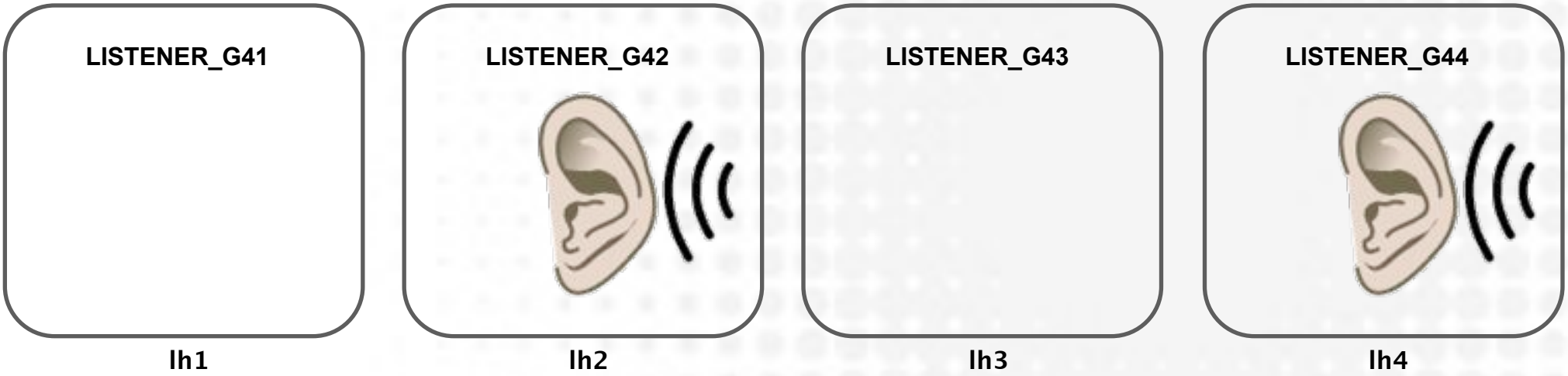
```
lh1-vip
lh2-vip
lh3-vip
lh4-vip
```



Client-side connection balancing and connection-time failover

```
LOAD_BALANCE=OFF
FAILOVER=ON
```

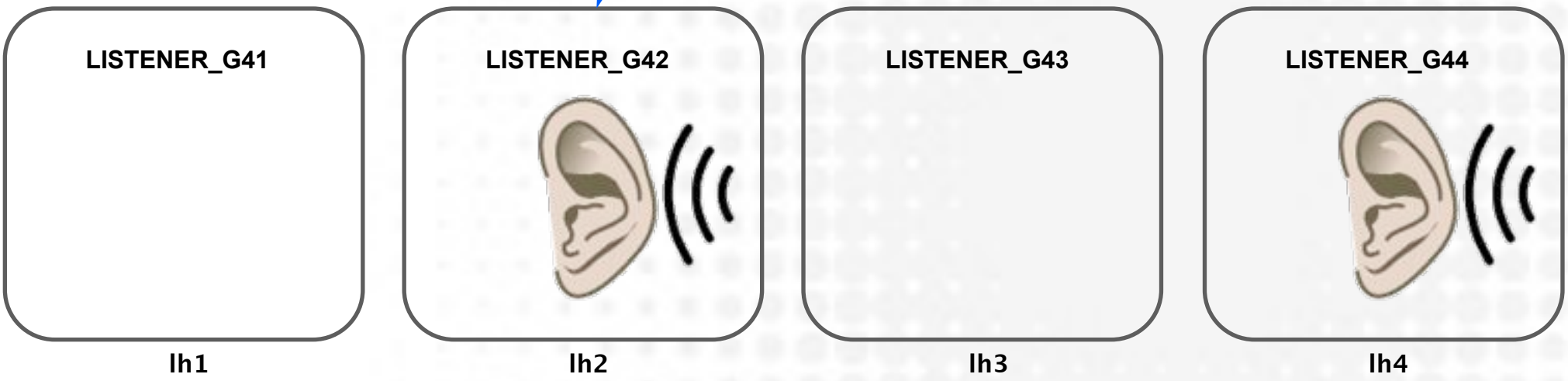
```
lh1-vip
lh2-vip
lh3-vip
lh4-vip
```



Client-side connection balancing and connection-time failover

```
lh1-vip
lh2-vip
lh3-vip
lh4-vip
```

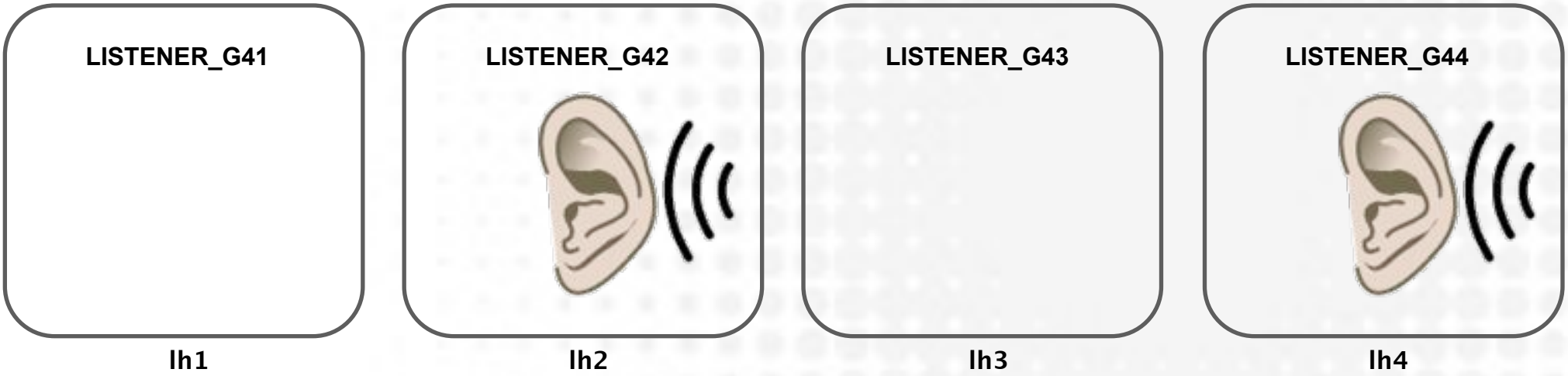
```
LOAD_BALANCE=OFF
FAILOVER=ON
```



Client-side connection balancing and connection-time failover

```
lh1-vip
lh2-vip
lh3-vip
lh4-vip
```

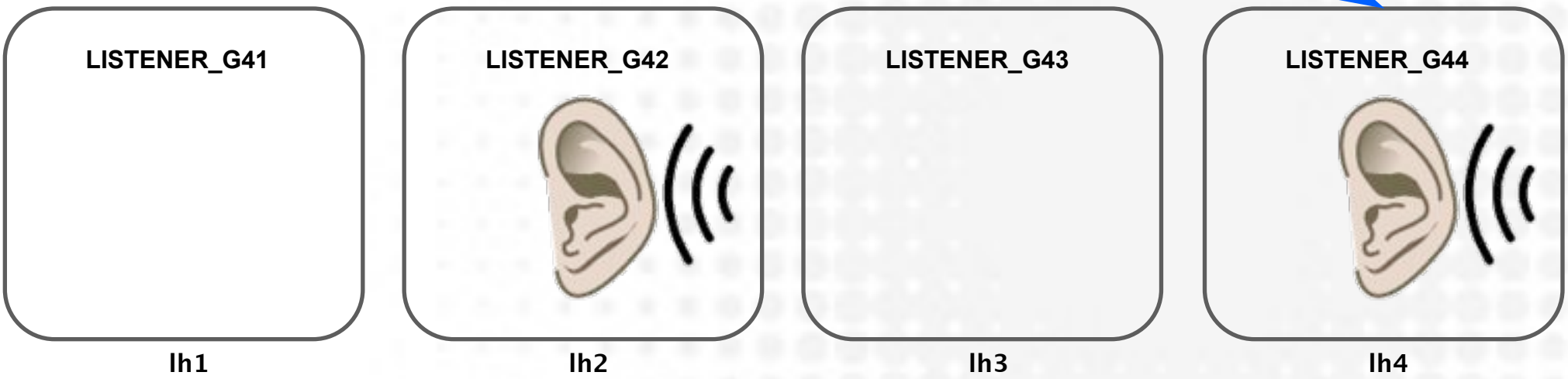
```
LOAD_BALANCE=ON
FAILOVER=ON
```



Client-side connection balancing and connection-time failover

```
lh1-vip
lh2-vip
lh3-vip
lh4-vip
```

```
LOAD_BALANCE=ON
FAILOVER=ON
```



Summary – client-side business

- Client chooses between several addresses
- Why
  - Failure tolerance
  - Distribution across several listeners
- How
  - Several connection points in connection descriptor
  - LOAD_BALANCE=ON
  - FAILOVER=ON

Listener – where to direct request?

lsnrctl status

……

```
Service "service10g" has 2 instance(s).
  Instance "g41", status READY, has 1 handler(s) for this service...
  Instance "g42", status READY, has 1 handler(s) for this service...
```

……

Where we are...

✓Client-side basics

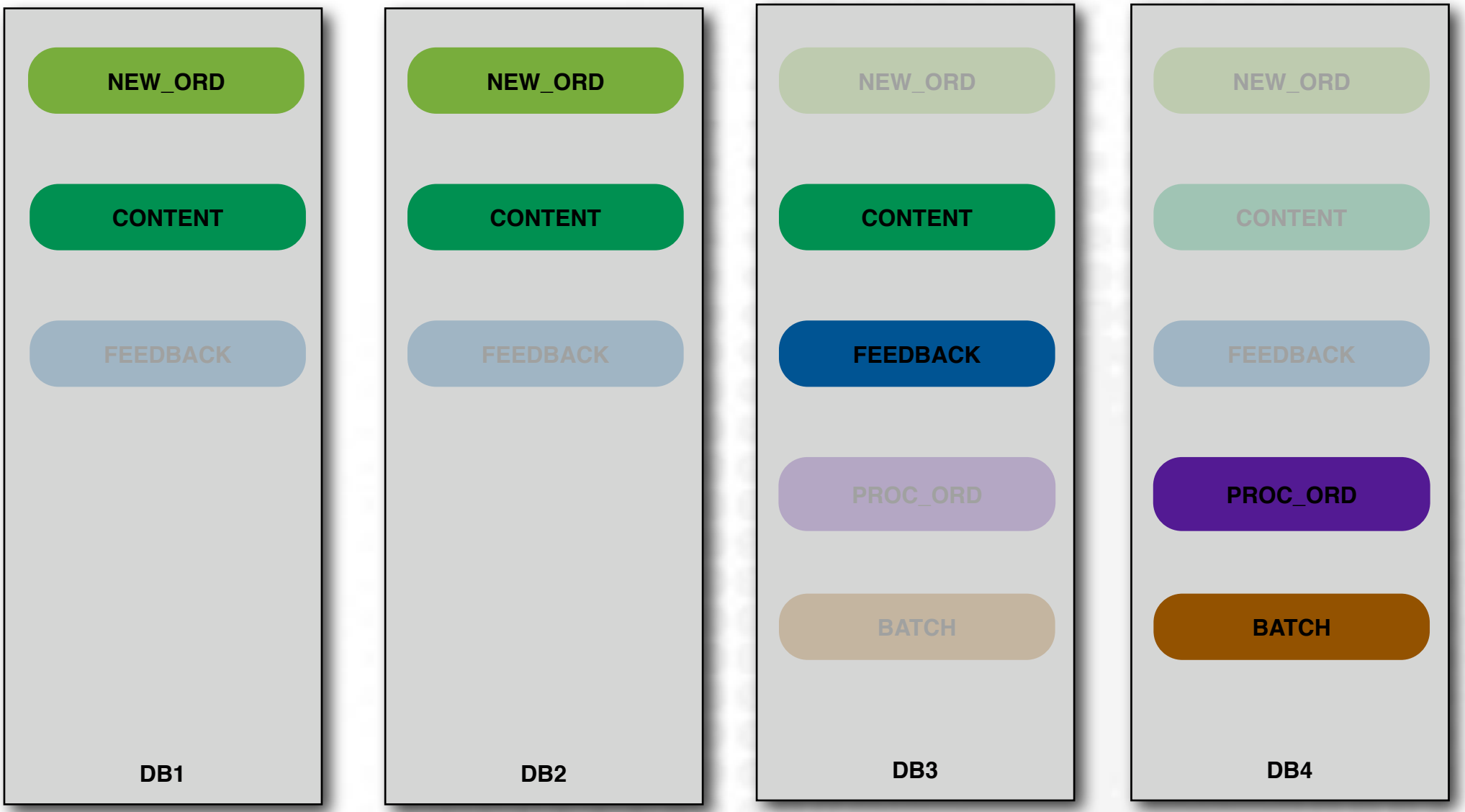
➡Oracle Cluster Services

- Server-side CLB
- FAN - HA & LBA events
- Thin JDBC RLB
- ONS Subscriber
- Your own LBA!

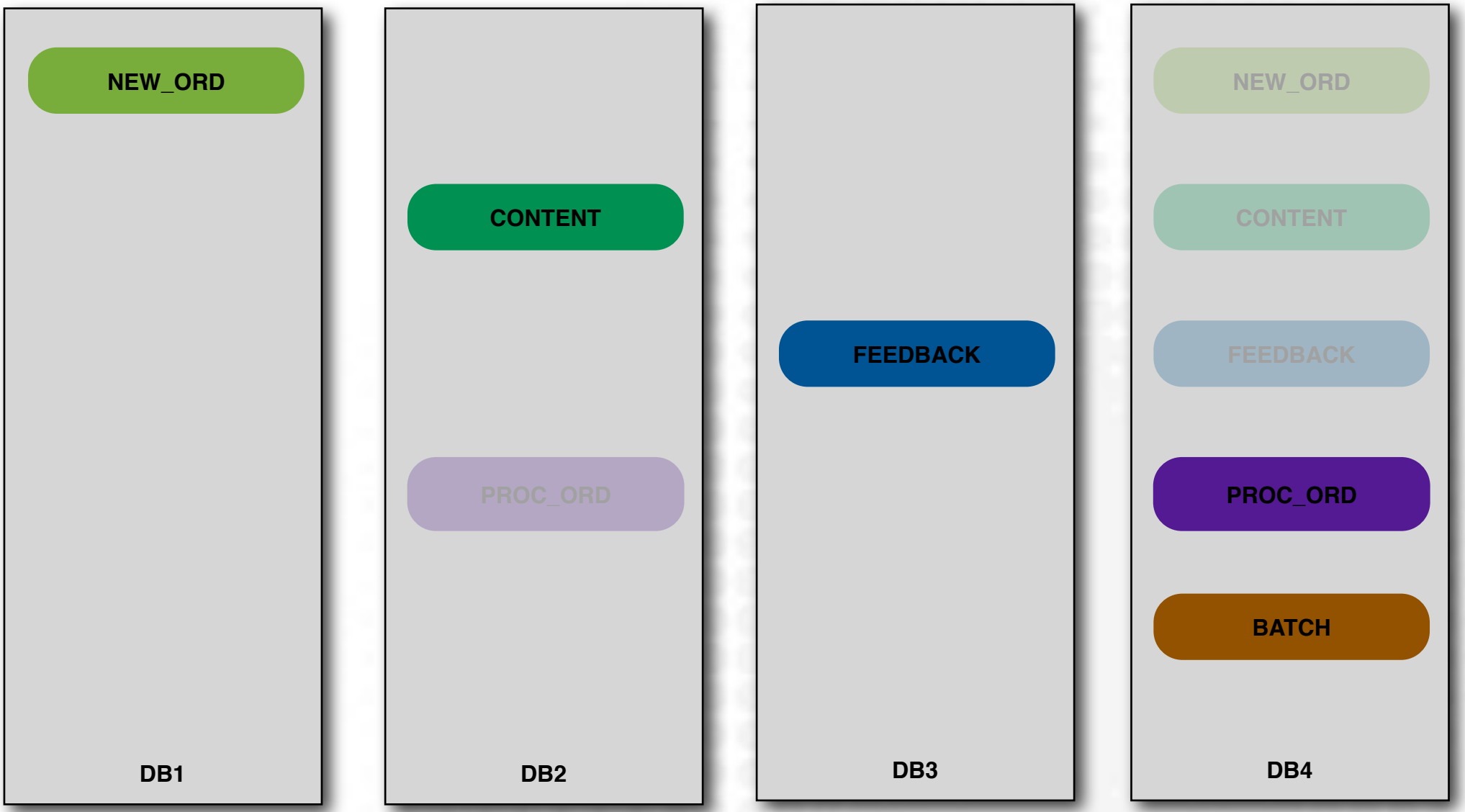
Orders processing example

- Business areas
  - Customers' order (web)
  - Content display (web cached on app-tier)
  - Feedback (web)
  - Orders fulfillment (back-office)
  - Data extraction batches

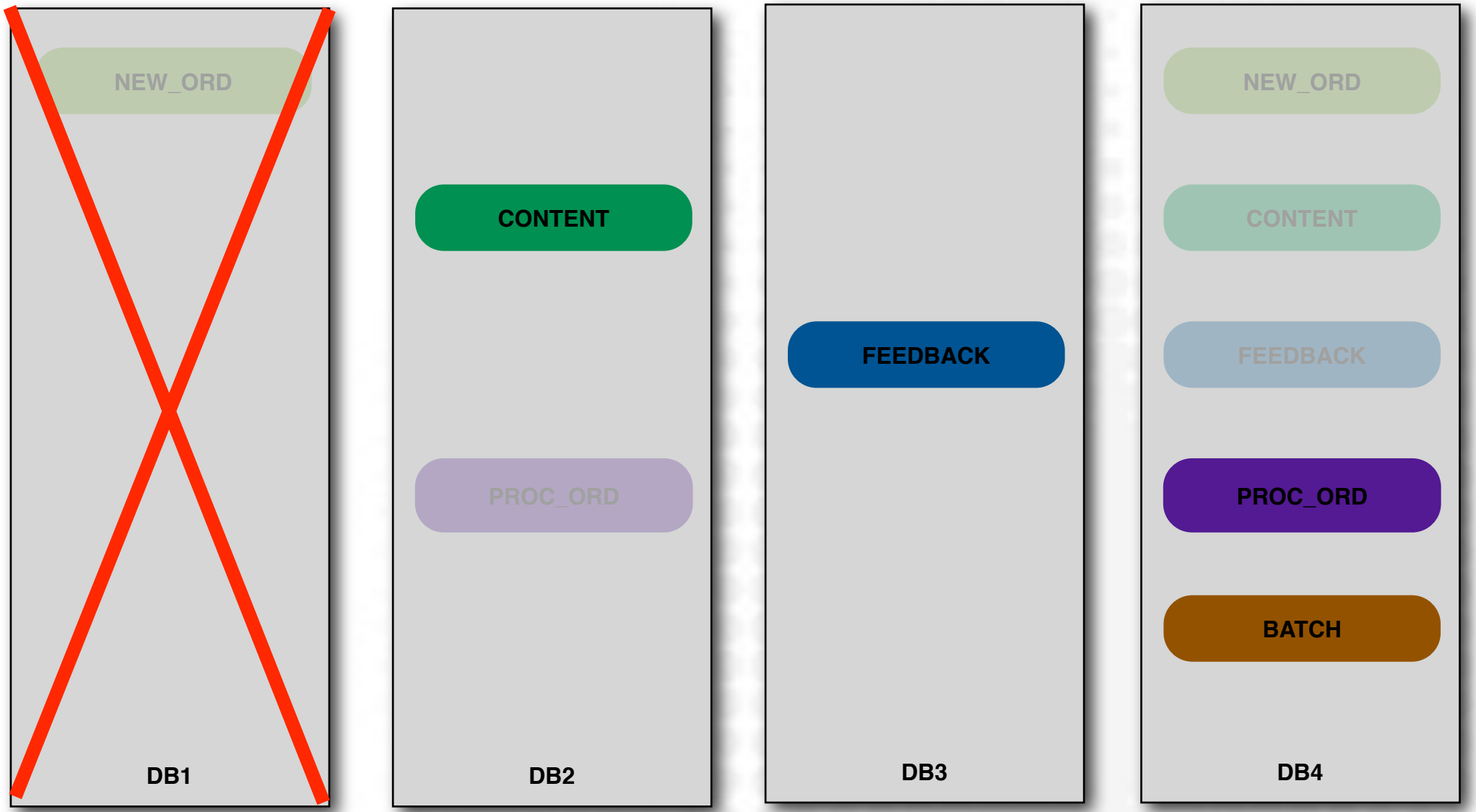
Services: preferred & available



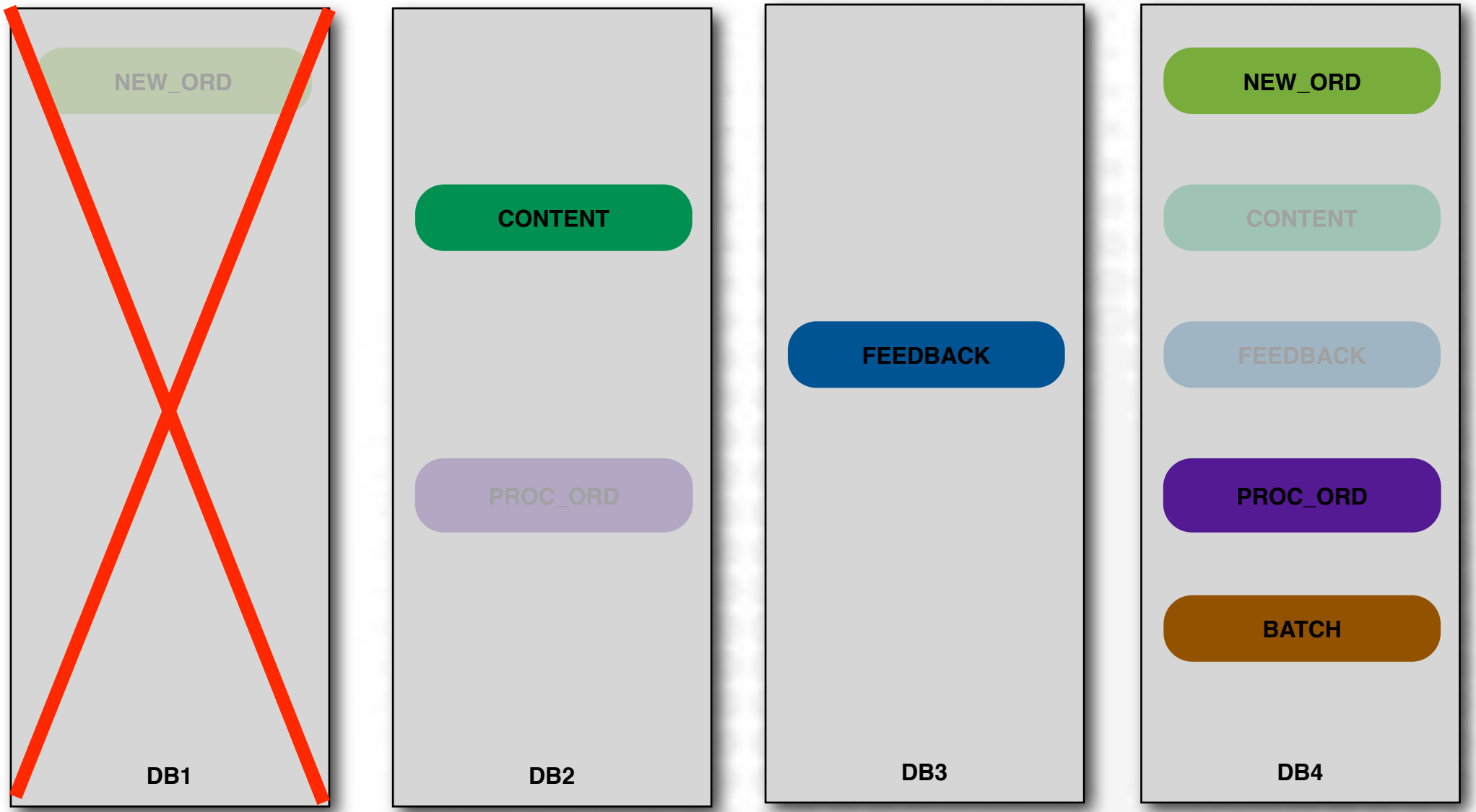
Node affinity



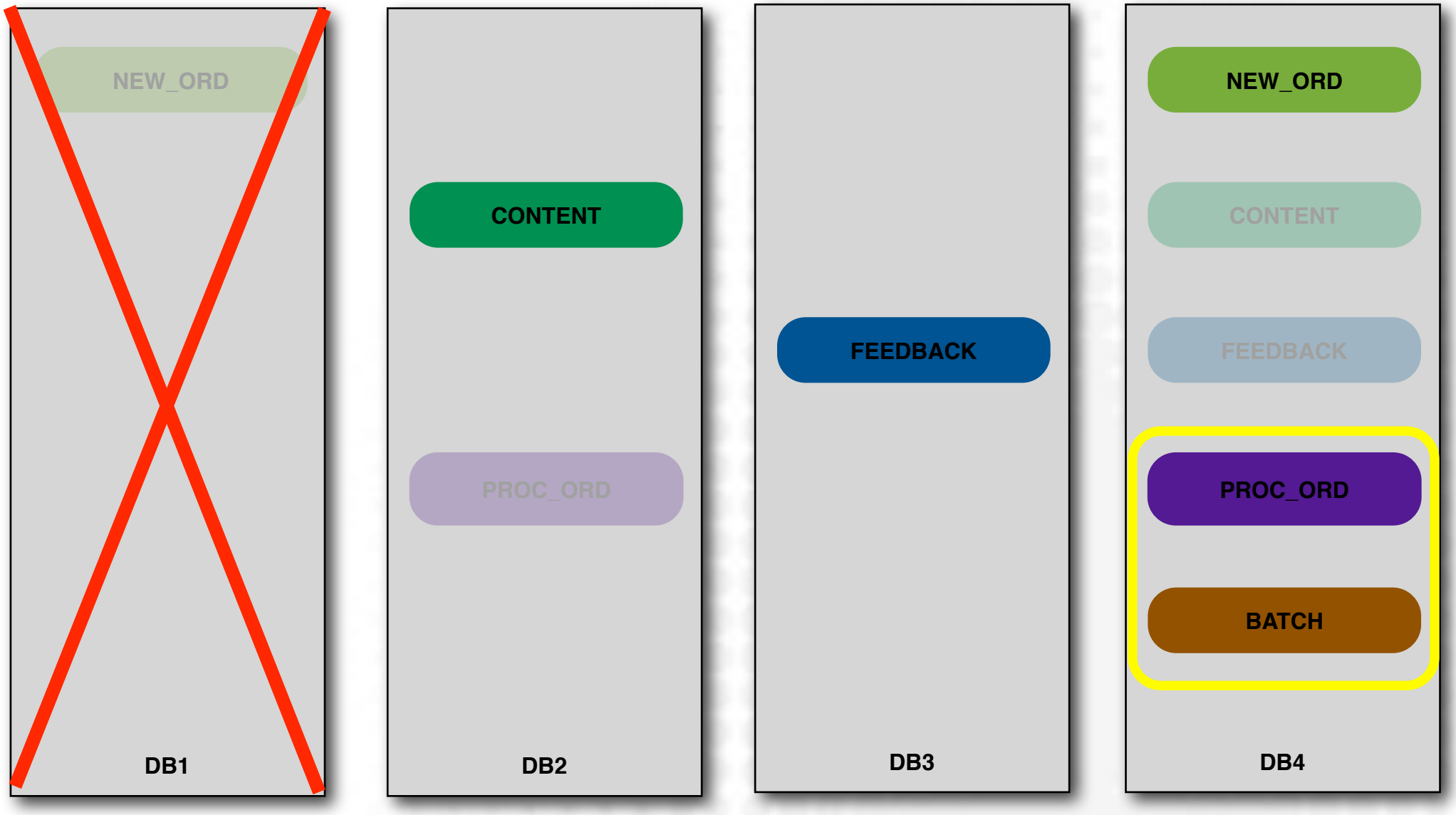
Service failover



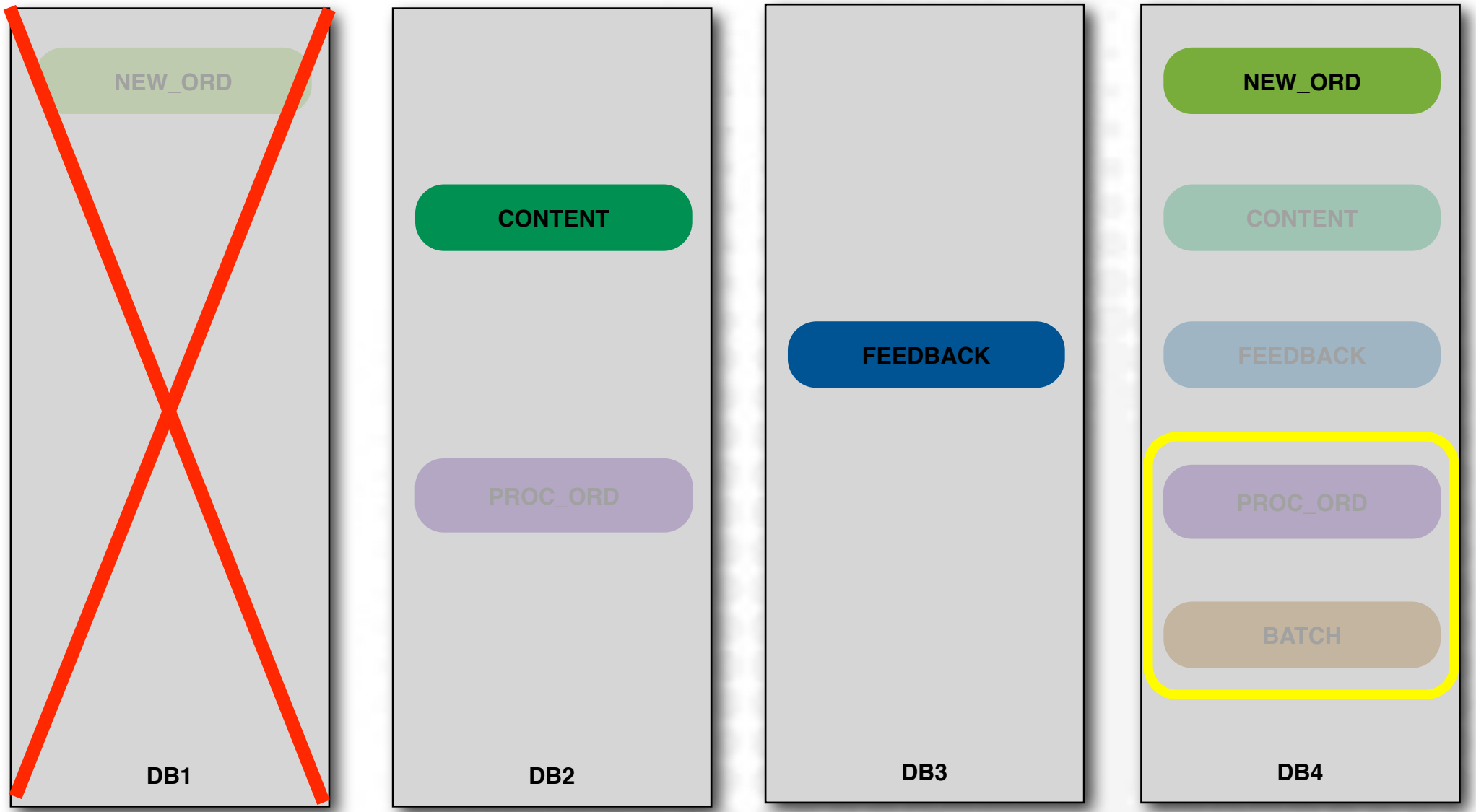
Service failover



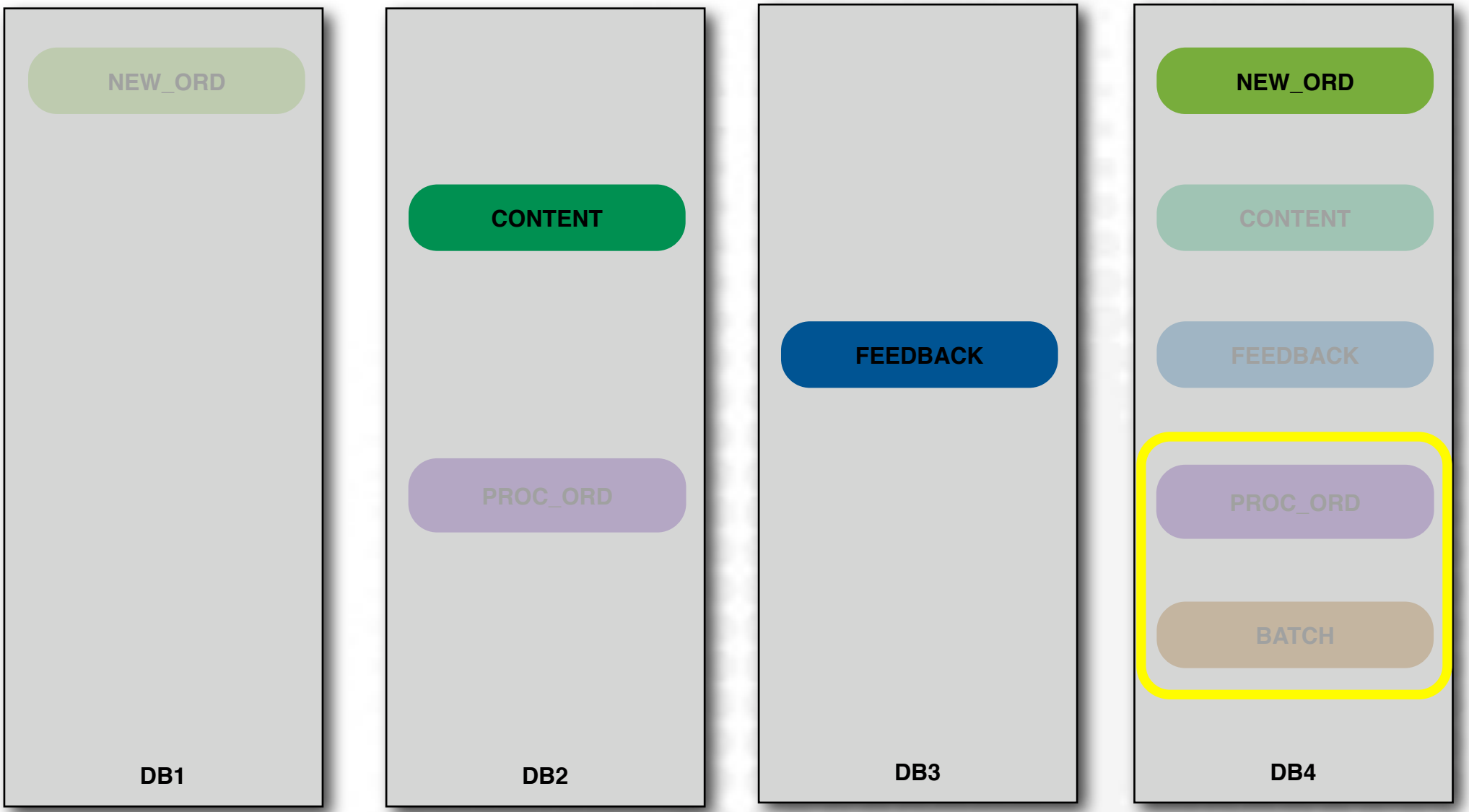
Service failover



Service failover



Service failover



Demo 1 - Services Automation

- HA events introduction
- Logging example
- Using server-side FAN HA events callout

Server-side FAN HA events

- Server-side callout can be used for:
  - Relocating services back to preferred nodes
  - Smart services handling
  - Notify and move sessions after service Down
  - Rebalance sessions on service up
  - Accommodate workload re-balancing for ***legacy applications***

Services & Resource Manager

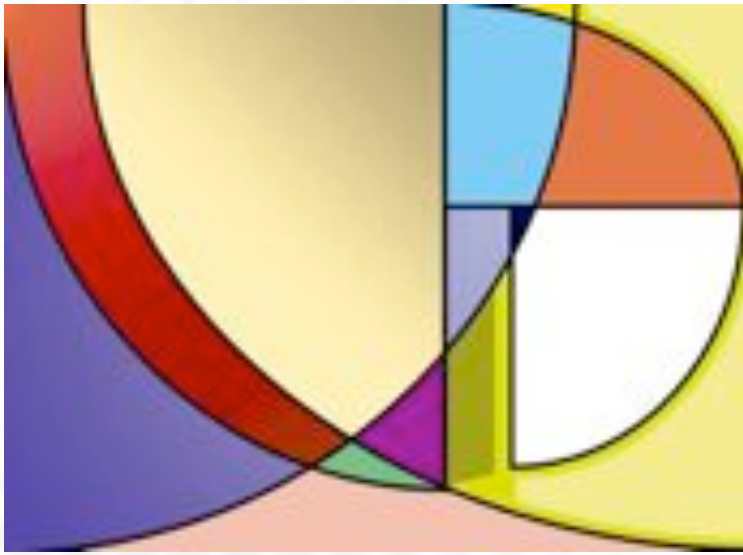
```
DBMS_RESOURCE_MANAGER.SET_CONSUMER_GROUP_MAPPING
    (DBMS_RESOURCE_MANAGER.SERVICE_NAME,
     'NEW_ORD', 'NEW_ORD_GROUP');

DBMS_RESOURCE_MANAGER.SET_CONSUMER_GROUP_MAPPING
    (DBMS_RESOURCE_MANAGER.SERVICE_NAME,
     'CONTENT', 'CONTENT_GROUP');

...
```

- Works on instance level – not RAC-aware
- Doesn't account for >1 instances per node

Services abstraction for apps



- Applications don’t need to know current services configuration!
- Listeners know services status
  - Listeners control which instance gets a new connection

Where we are...

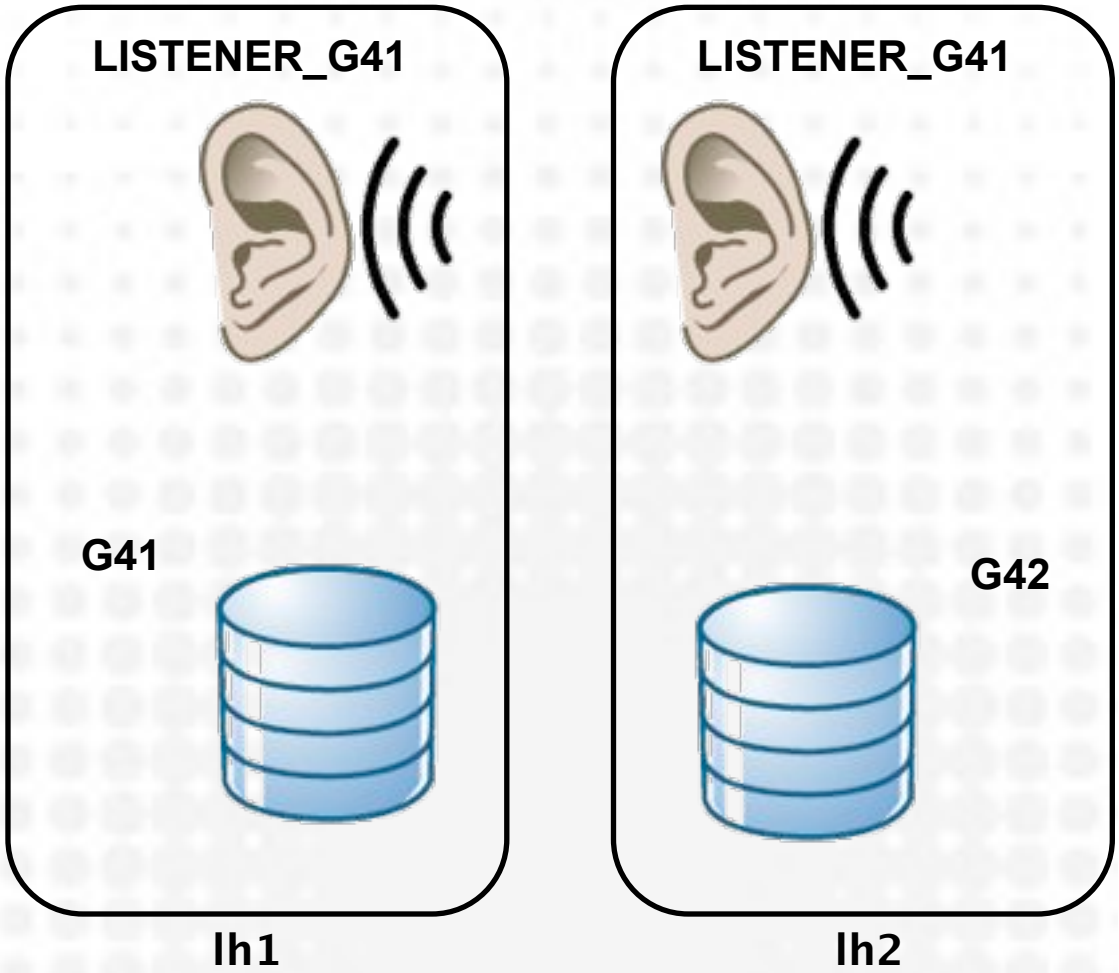
✓Client-side basics

✓Oracle Cluster Services

➡Server-side CLB

- FAN - HA & LBA events
- Thin JDBC RLB
- ONS Subscriber
- Your own LBA!

Remote listener registration



Remote listener registration

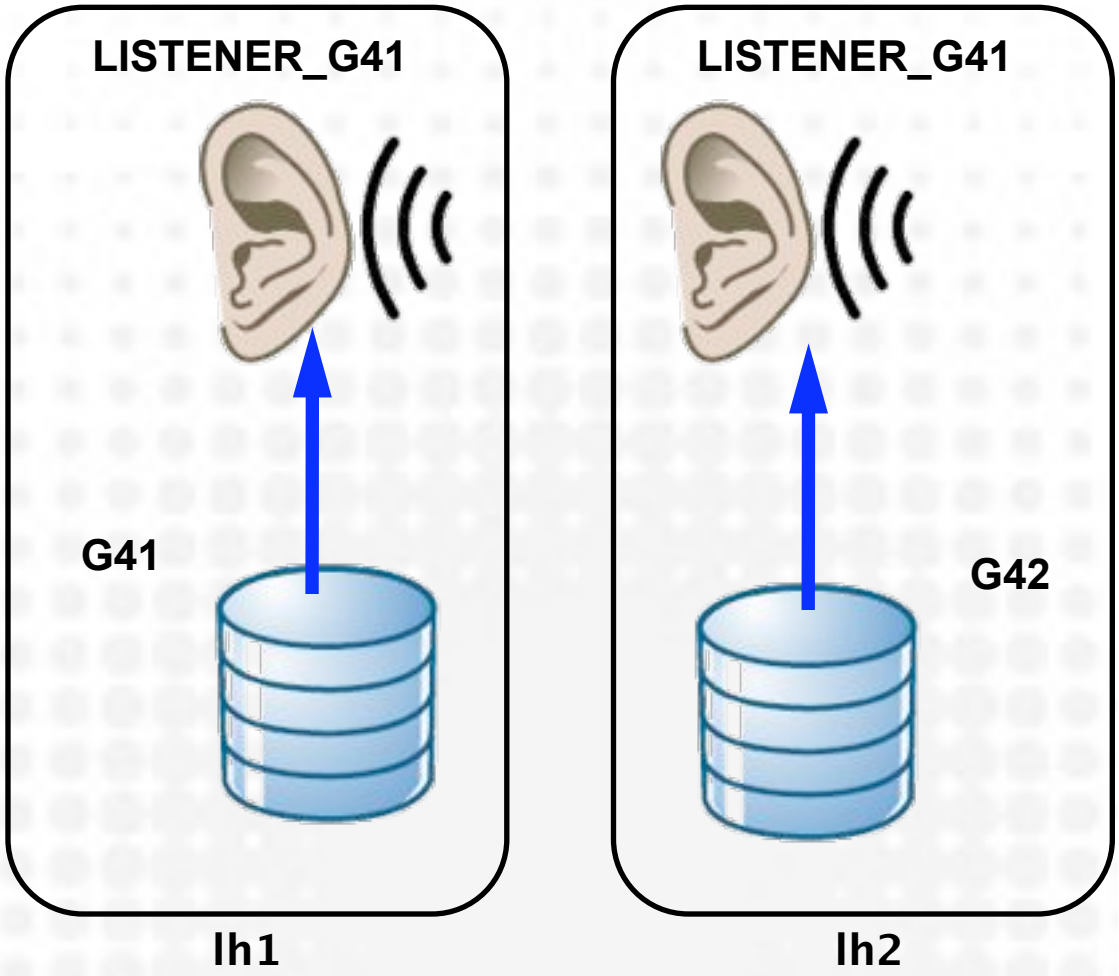
Local registration

```
G41.local_listener=LISTENER_G41

G42.local_listener=LISTENER_G42

G43.local_listener=LISTENER_G43

G44.local_listener=LISTENER_G44
```



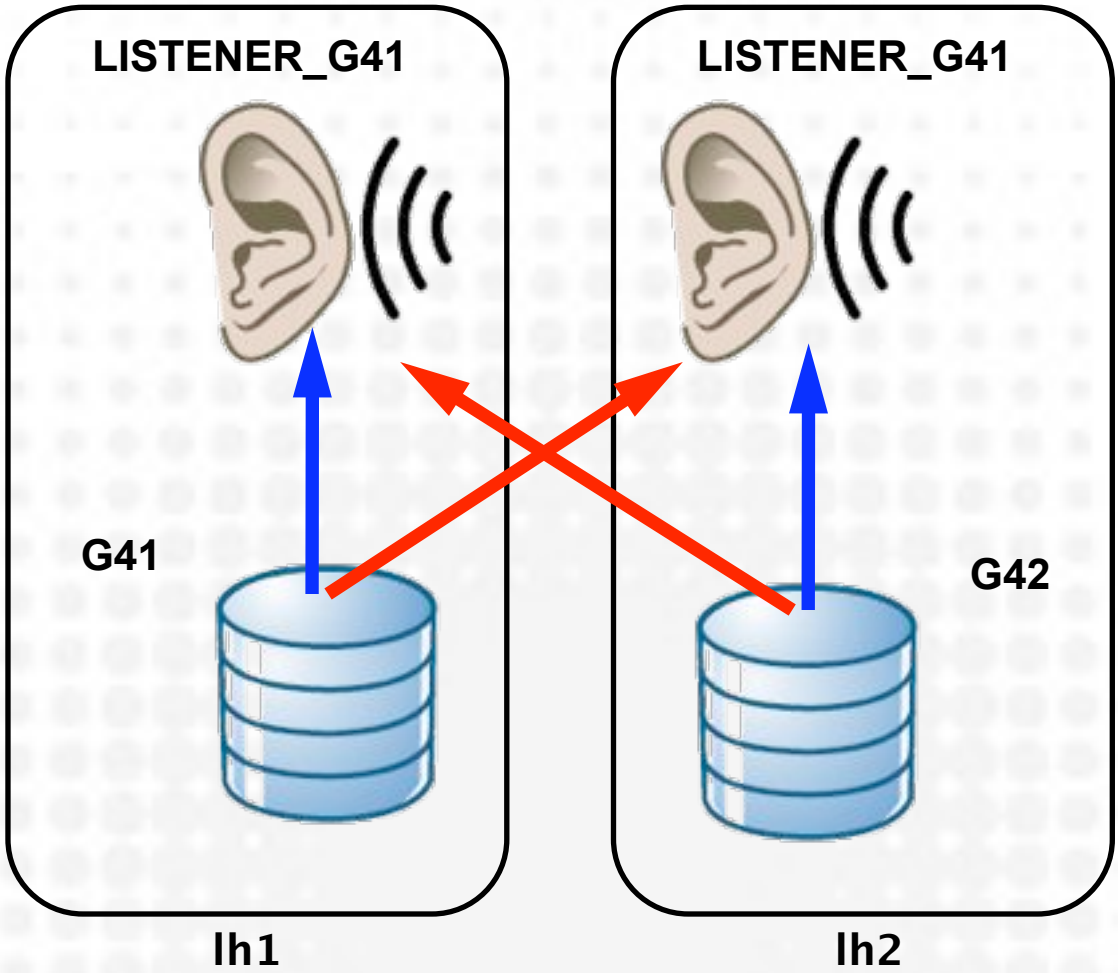
Remote listener registration

Local registration

```
G41.local_listener=LISTENER_G41
G42.local_listener=LISTENER_G42
G43.local_listener=LISTENER_G43
G44.local_listener=LISTENER_G44
```

Remote registration

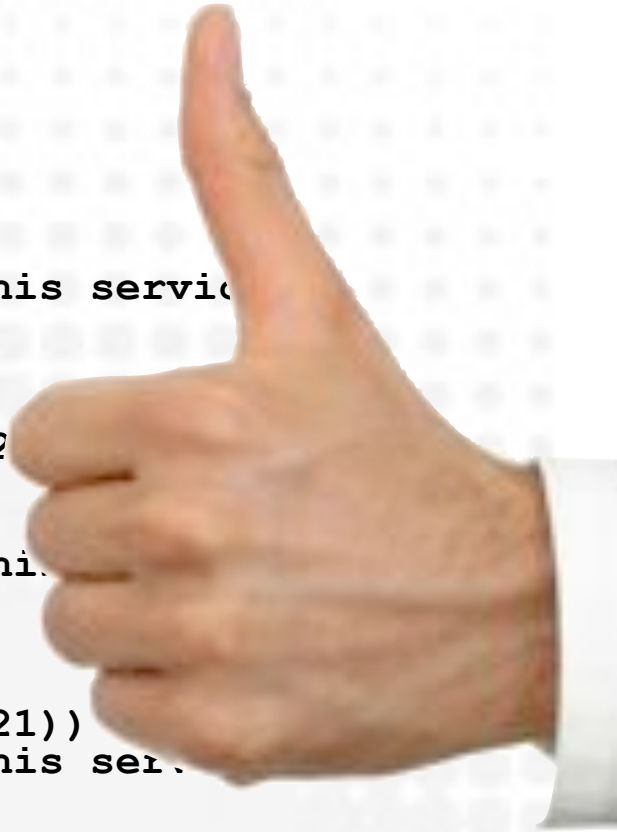
```
*.remote_listener=LISTENERS_G4
```



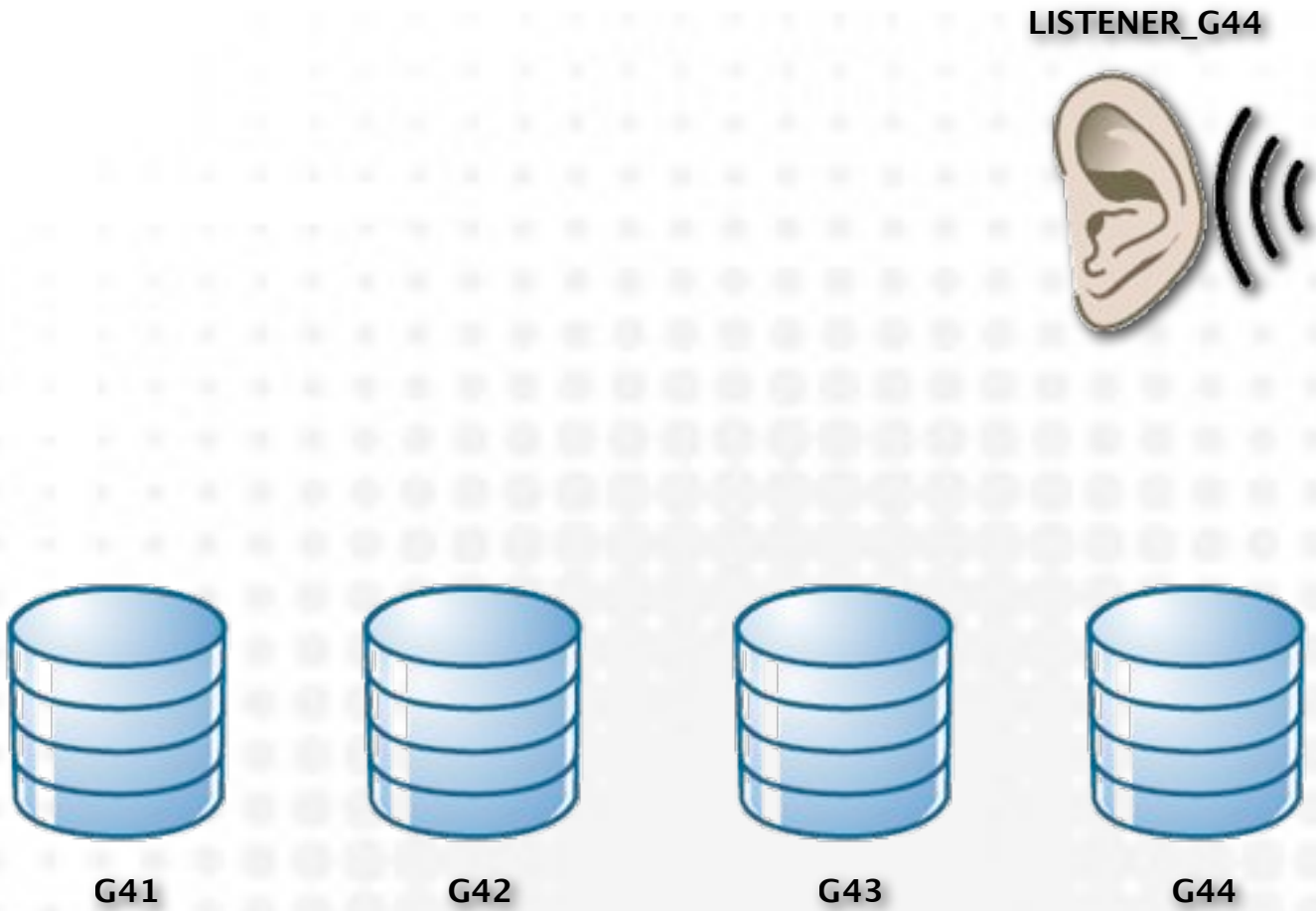
Correct listener configuration
(google for Pythian video VIP)

lsnrctl service

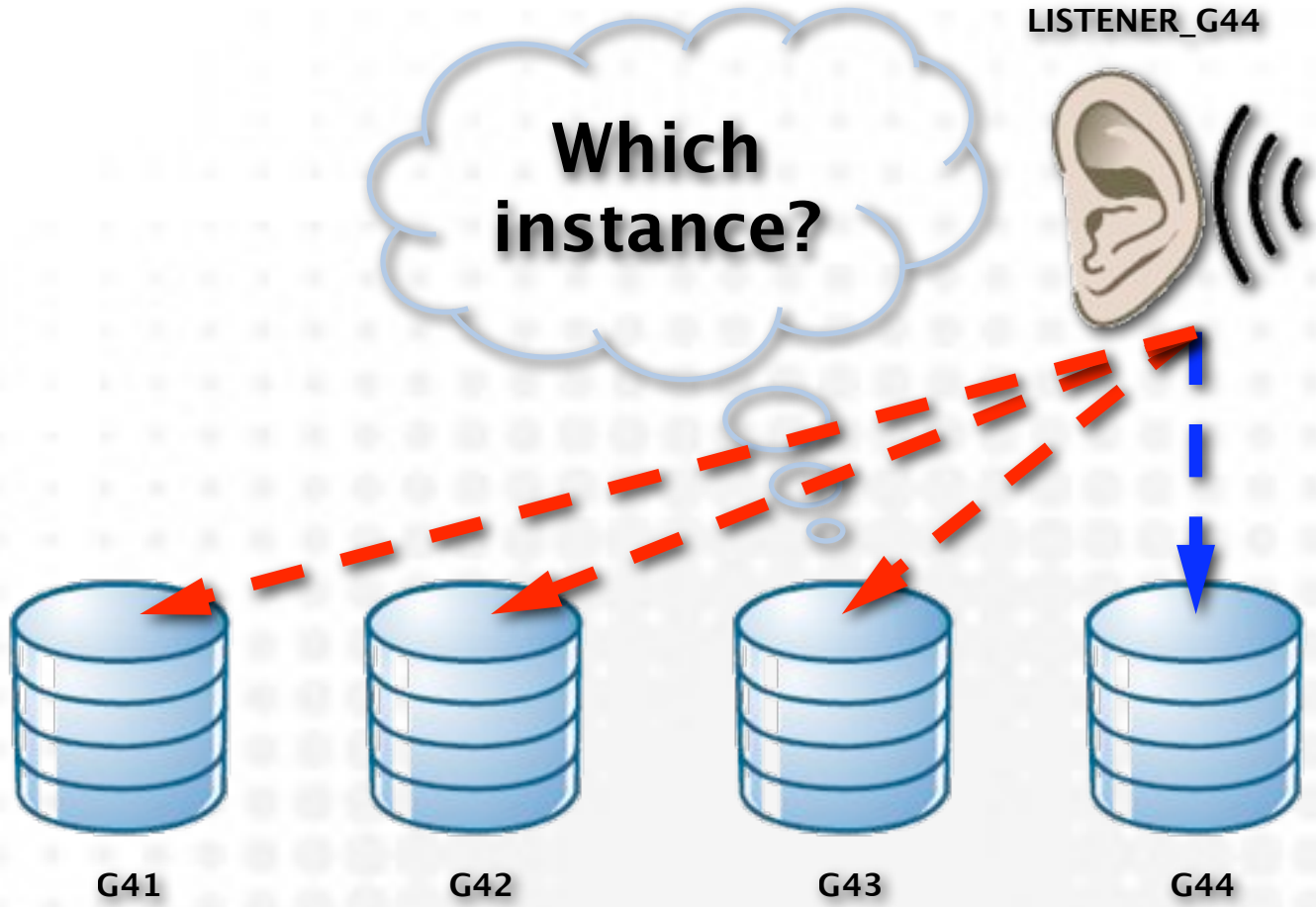
```
Service "service10g.oracloid.com" has 2 instance(s).
  Instance "g41", status READY, has 1 handler(s) for this servic
    Handler(s):
     "DEDICATED" established:0 refused:0 state:ready
         REMOTE SERVER
         (ADDRESS=(PROTOCOL=TCP)(HOST=lh1-vip)(PORT=152
      "DEDICATED" established:0 refused:0 state:ready
         LOCAL SERVER
  Instance "g42", status READY, has 1 handler(s) for thi
    Handler(s):
      "DEDICATED" established:0 refused:0 state:ready
         REMOTE SERVER
         (ADDRESS=(PROTOCOL=TCP)(HOST=lh2-vip)(PORT=1521))
  Instance "g43", status READY, has 1 handler(s) for this ser
    Handler(s):
      "DEDICATED" established:0 refused:0 state:ready
         REMOTE SERVER
         (ADDRESS=(PROTOCOL=TCP)(HOST=lh3-vip)(PORT=1521))
  Instance "g44", status READY, has 1 handler(s) for this
...
```



Server-side CLB

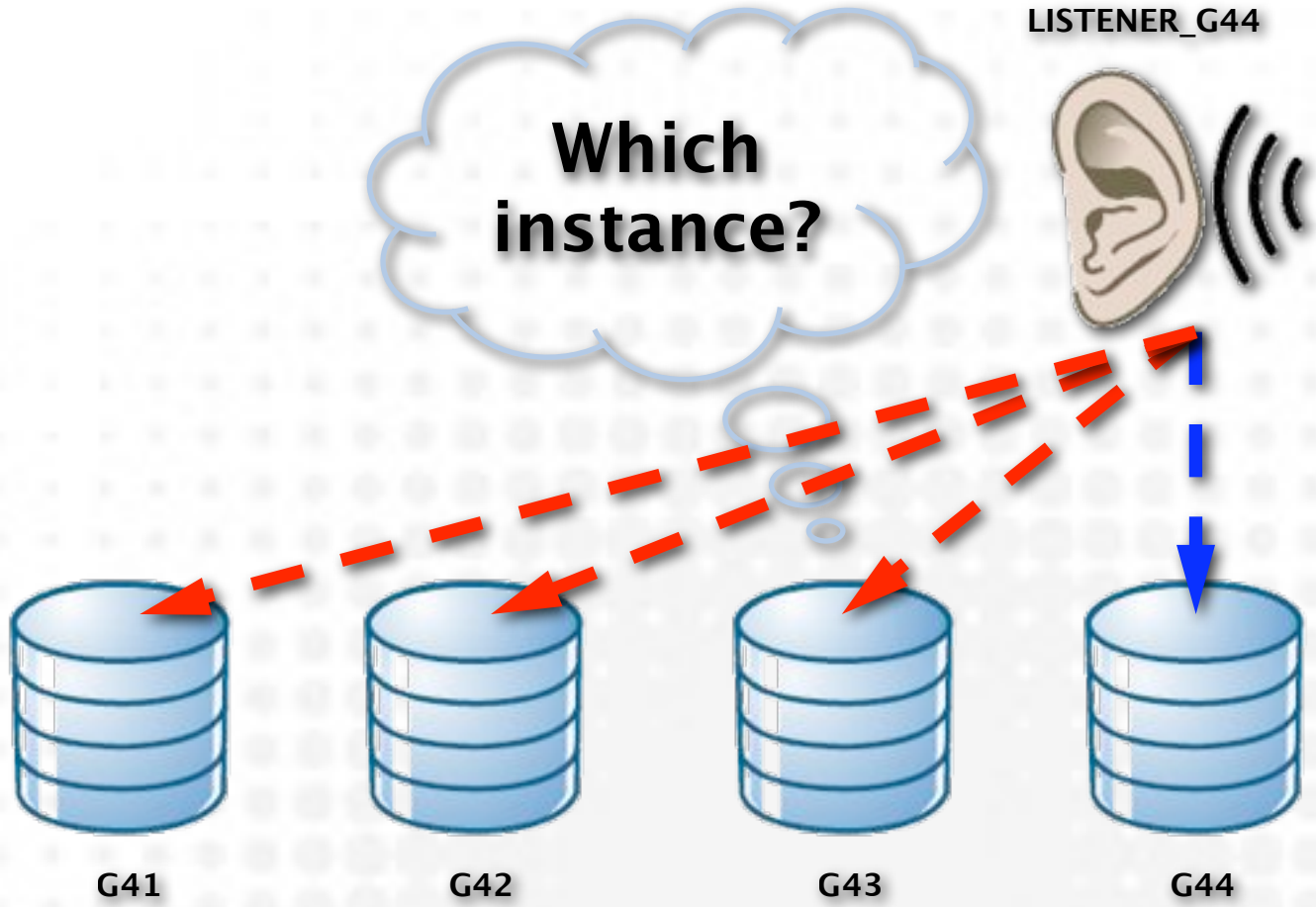


Server-side CLB



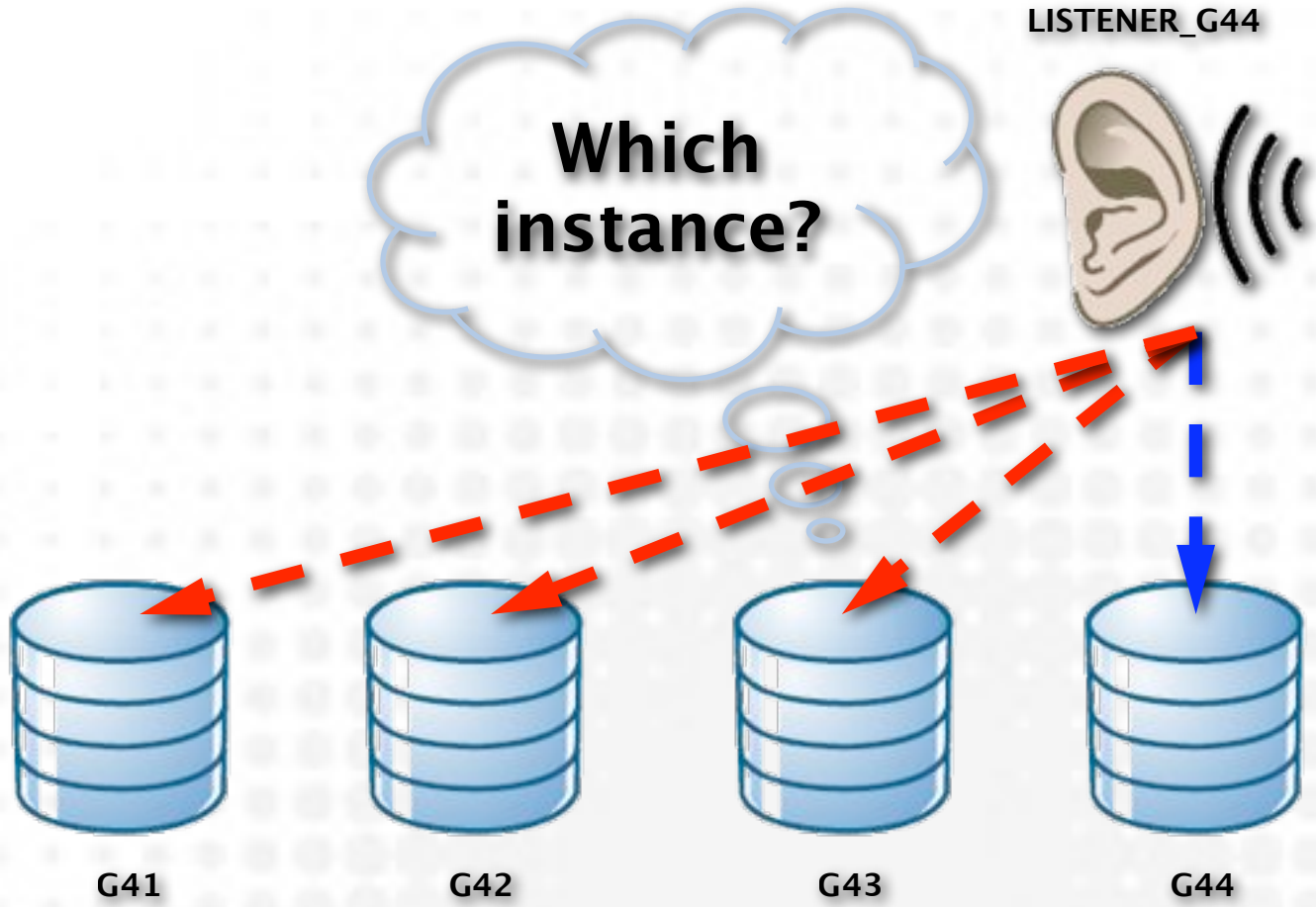
Server-side CLB

- <= 10gR1
  - Instance load
  - Node load



Server-side CLB

- <= 10gR1
  - Instance load
  - Node load
- 10gR2
  - Service Metrics
  - AWR



Demo helper
Tracing listener

lsnrctl trace **user**

```
nsglgrDoRegister: inst loads: ld1:145 mld1:5120 ld2:5 mld2:170
nsglgrDoRegister: inst loads: ld1:58 mld1:5120 ld2:4 mld2:170
```

- **No notion of instance name in trace ☹**

- **ld1 – Load Data 1, mld1 – Max Load Data 1**
- **ld2 – Load Data 2, mld2 – Max Load Data 2**
- **Load data is instance load and node load**

Demo helper

PREFER_LEAST_LOADED_NODE

- listener.ora parameter
  - PREFER_LEAST_LOADED_NODE_%
- ON (default)
  - ld1 => node, ld2 = > instance
- OFF
  - ld1 => instance, ld1 => node

Demo helper
Instance and node loads

- Instance load
  - Number of user sessions
  - TYPE<>BACKGROUND
- Max instance load
  - Max number of sessions
  - SESSIONS init.ora
- Node load
  - Index based probably on load average
- Max node load
  - Max CPU capacity
  - cpu_count*5K

Demo helper
Uneven connection distribution

- PREFER_LEAST_LOADED_NODE_...=ON
  - Default
- Listener uses CPU load to balance connections to the machine with more CPU capacity left
- Can work fine with differently sized nodes
- But what if connections are “permanent”?

Demo helper
"Even" connection distribution

```
PREFER_LEAST_LOADED_NODE_LISTENER_G41=OFF

PREFER_LEAST_LOADED_NODE_LISTENER_G42=OFF
```

- Load based on number of sessions left until maximum is reaches
  - init.ora parameter SESSIONS
- Can configure SESSIONS differently for every node and influence distribution

Demo 3 - 9i style CLB

- listener trace
- PREFER_LEAST_LOADED_NODE
  - instance load
  - node load
- PMON trace 10257 event

9i and 10gR1 CLB summary

- PMON reports to listener
  - instance load
  - node load
- PREFER_LEAST_LOADED_NODE
  - ON - node load (*default*)
  - OFF - instance load
- *lsnrctl trace user*
- PMON: *10257 trace name context forever, level 16*

DB_DOMAIN and bug 5593693

- db_domain init.ora parameter is set
- service_name is not FQDN
  - that's what srvctl does
- Listener appends default domain
- Service metrics updates are lost
  - on the way from PMON => listener
- Result: fallback to instance / node load
- Fixed - 10.2.0.4 & 11.1.0.6

10gR2 and 11g server-side CLB

- Service metrics for each service
- GOODNESS
  - Attractiveness of the instance for this service
  - Higher is worse (should be BADNESS)
- DELTA
  - Estimate of “badness” increase for additional session
- FLAGS (bit map)
  - 0 – all good
  - 1 – blocked
  - 2 – crossed threshold
  - 4 – goodness unknown (usually when no sessions connected)

Connection life-time

- Short living sessions
- Persistent / Oracle Forms / long connections
- Connection pool
  - Oracle standard connection pool
  - Custom connection pool
  - Application server connection pool

Connection classification & CLB

- Long connections
  - idle most of the time (!)
    - persistent connections (non-pooled)
    - over-allocated connection pools
- Short connections
  - active most of the time
  - dynamic connection pools
- Batch-like connection
  - long but always active
  - under-allocated connection pools (!)

CLB goal

- CLB_GOAL_LONG
  - equal distributions of sessions
- CLB_GOAL_SHORT
  - based on Service Metrics and LBA goal (!)
  - GOAL_NONE
  - GOAL_SERVICE_TIME
  - GOAL_THROUGHPUT

Demo 4 - Service Metrics & CLB

- CLB_GOAL_LONG
- CLB_GOAL_SHORT
- listener trace
- PMON trace
- V$SERVICEMETRIC

10g services load balancing

- For each service defined:
  - Connection load balancing goal
    - CLB_GOAL_SHORT
    - CLB_GOAL_LONG
  - Load Balancing Advisory - LBA
    - GOAL_SERVICE_TIME
    - GOAL_THROUGHPUT

V$SERVICEMETRIC

- For each service and instance contains
  - Short interval (5 seconds)
  - Long interval (60 seconds)
  - INTSIZE_CSEC
  - CPUPERCALL
  - DBTIMEPERCALL
  - CPUPERSEC
  - DBTIMEPERSEC

V$SERVICEMETRIC (cont'd)

- GOODNESS
  - Attractiveness of the instance for this service
  - Higher is worse (should be BADNESS)
- DELTA
  - Estimate of "badness" increase for additional session
- FLAGS (bit map)
  - 0 – all good
  - 1 – blocked
  - 2 – crossed threshold
  - 4 – goodness unknown (usually when no sessions connected)

GOODNESS and CLB_GOAL

- LONG
  - GOODNESS=number of connected sessions
  - DELTA=1
- SHORT
  - Calculated based on
    - GOAL_SERVICE_TIME
    - GOAL_THROUGHPUT
    - GOAL_NONE

GOODNESS and CLB_GOAL_SHORT

- According to my observations
- GOAL_SERVICE_TIME
  - "per call" metrics used
    - CPUPERCALL
    - DBTIMEPERCALL
- GOAL_THROUGHPUT
  - "per second" metrics used
  - CPUPERSEC
  - DBTIMEPERSEC

Listener trace... again

- We can see goodness and delta when instance/service started

```
nsglgrDoRegister: Creating new service: service10g

nsglgrDoRegister: service:service10g
                                  flag:2 goodness:0 delta:0
```

- During service updates

```
nsglgrDoRegister: service:service10g what:4 value:1
nsglgrDoRegister: service:service10g what:2 value:47
```

  - what: 2 = goodness
  - what: 4 = delta

How does listener know about it?

- MMON is the “calculator”
  - V$SERVICEMETRIC
- PMON is the “messenger”
  - `10257 trace name context forever, level 16`

PMON trace example

- 11g trace

```
*** 2007-09-19 05:52:24.797
kmmlrl: update for session drop delta:
                    111 106 10 10 82
kmmlrl: service11g.oracloid.com goodness 8
kmmlrl: update for service goodness
kmmlrl: 55 processes
kmmlrl: node load 358
kmmlrl: instance load 19
kmmlrl: nsgr update returned 0
```

Service Metrics & CLB

- CLB_LONG

  Target - equal number of sessions per instance

  GOODNESS=number of session, DELTA=1

- CLB_SHORT + GOAL_NONE

  Target - equalize CPU utilization

  GOODNESS and DELTA => based on run-queue / load average

- CLB_SHORT + GOAL_SERVICE_TIME

  Target - minimize response time

  GOODNESS + DELTA <= LBA _PER_CALL (?)

- CLB_SHORT + GOAL_THROUGHPUT

  Target - maximize throughput

  GOODNESS + DELTA <= LBA _PER_SEC (?)

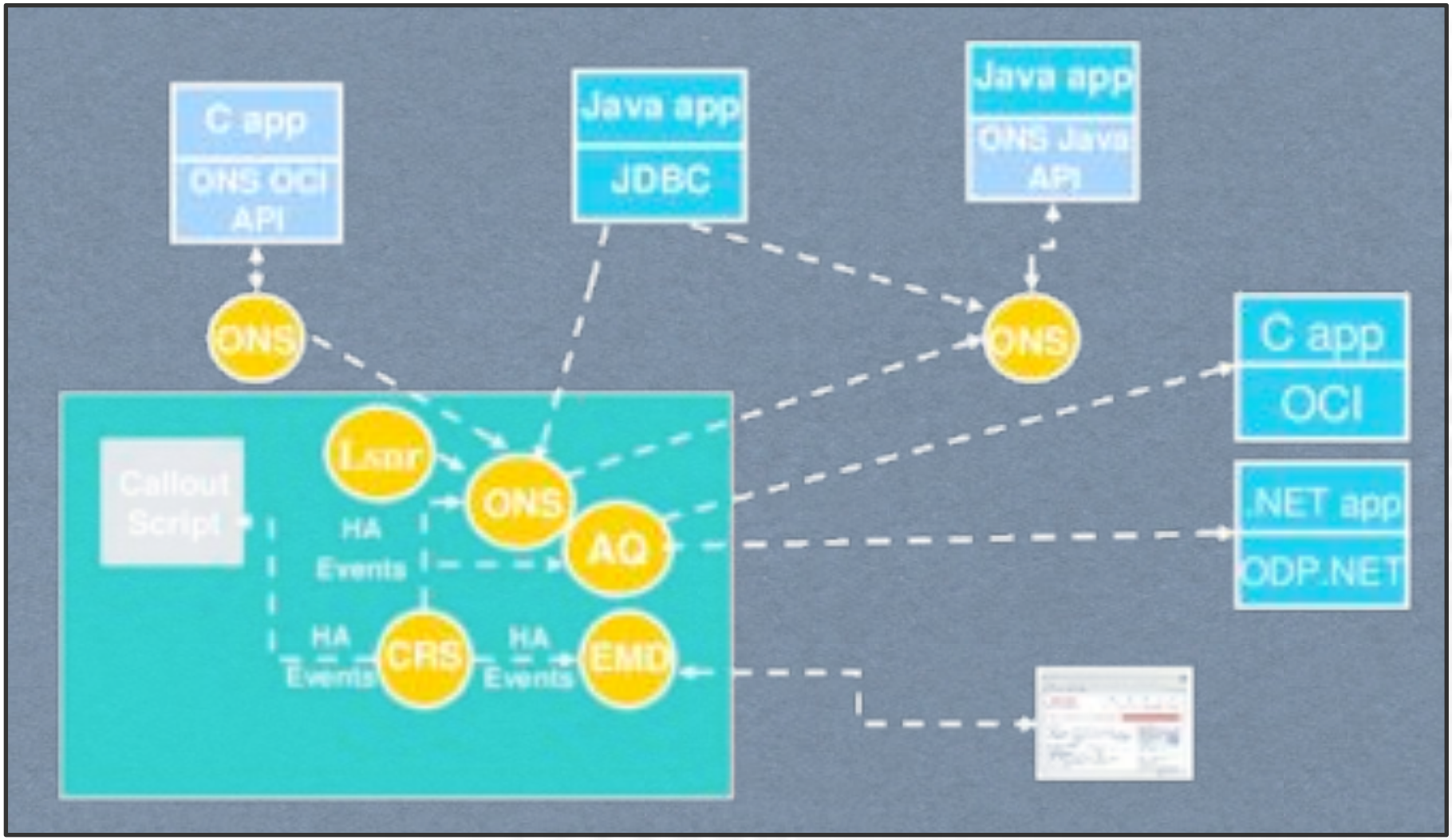
Nodes with different capacity

- Works with instance load
  - use different SESSIONS in init.ora
- Works with node load
  - based on CPU_COUNT
- Service Metrics?
  - bug 6613950
  - CLB_GOAL_SHORT is slower than node load *(my observations)*
  - CLB faster than RLB

Where we are...

- ✓Client-side basics
- ✓Oracle Cluster Services
- ✓Server-side CLB
- ➡FAN - HA & LBA events
- Thin JDBC RLB
- ONS Subscriber
- Your own LBA!

Clusterware processes



ONS Configuration

- $ORA_CRS_HOME/opmn/conf/ons.config

```
localport=6101
#10g
#remoteport=6201
loglevel=9
useocr=on
```

- $ORA_CRS_HOME/bin/onsctl start/stop/ping
- $ORA_CRS_HOME/bin/racgons add_config/remove_config

Demo 6 - LBA events

- ONS log

```
eventType: database/event/servicemetrics/ser
VERSION=1.0 database=g4 {
  {instance=g42 percent=34 flag=GOOD}
  {instance=g43 percent=33 flag=GOOD}
  {instance=g41 percent=33 flag=GOOD}
  } timestamp=2008-02-11 03:24:18
```

- SYS.SYS$SERVICE_METRICS_TAB

Using FAN events on app-tier

- ONC – Oracle Notification Client
- Java ONS API
- C ONS API
- Can be used to manage non-pooled connections or home-grown pools
- Can be standalone ONC – no JDBC/OCI
- – – – local ONS is required (at least with 10g)

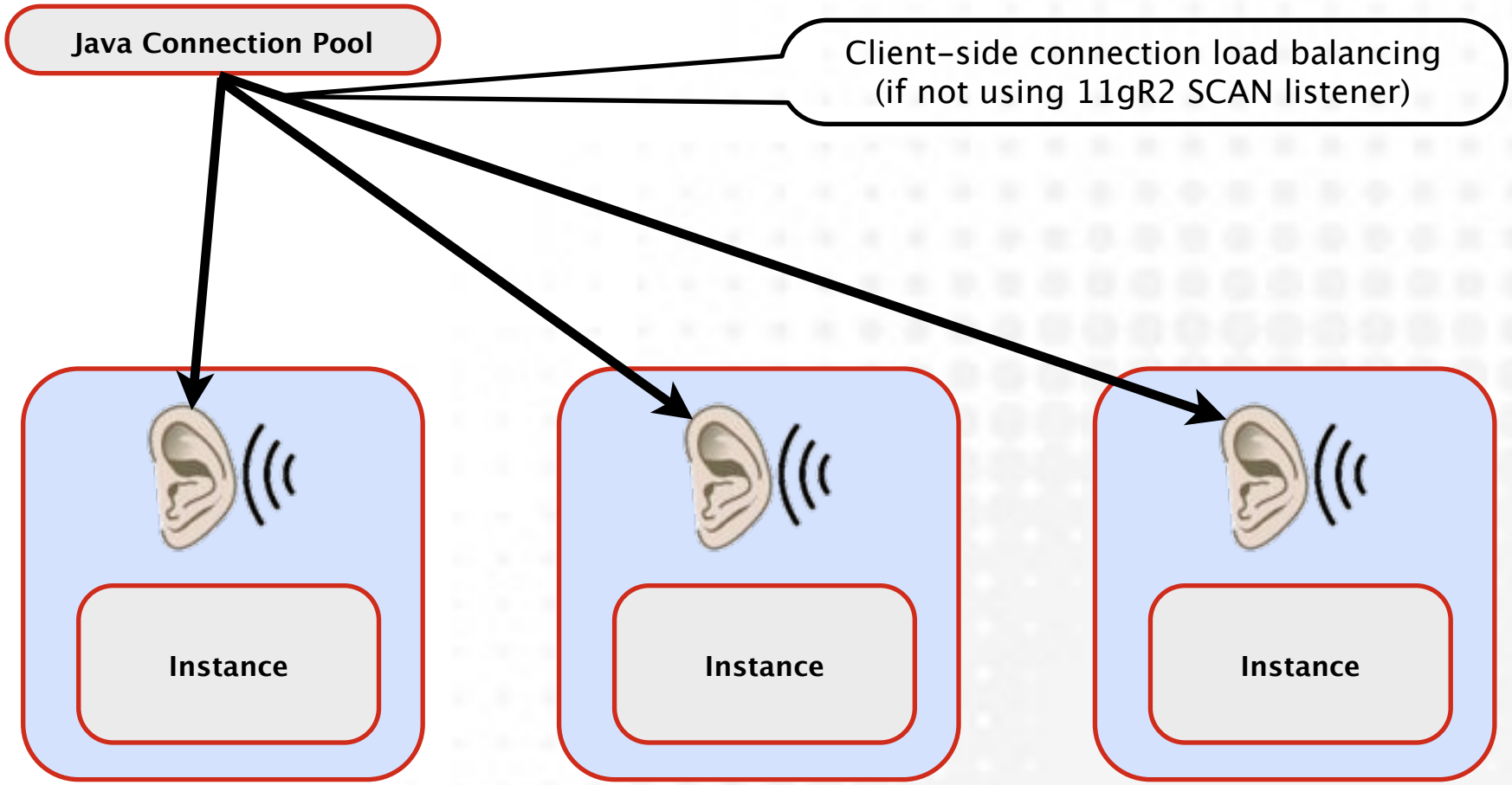
Where we are...

- ✓Client-side basics
- ✓Oracle Cluster Services
- ✓Server-side CLB
- ✓FAN - HA & LBA events
- ➡Thin JDBC RLB
- ONS Subscriber
- Your own LBA!

Establishing a Connection

Java Connection Pool

Establishing a Connection



Establishing a Connection

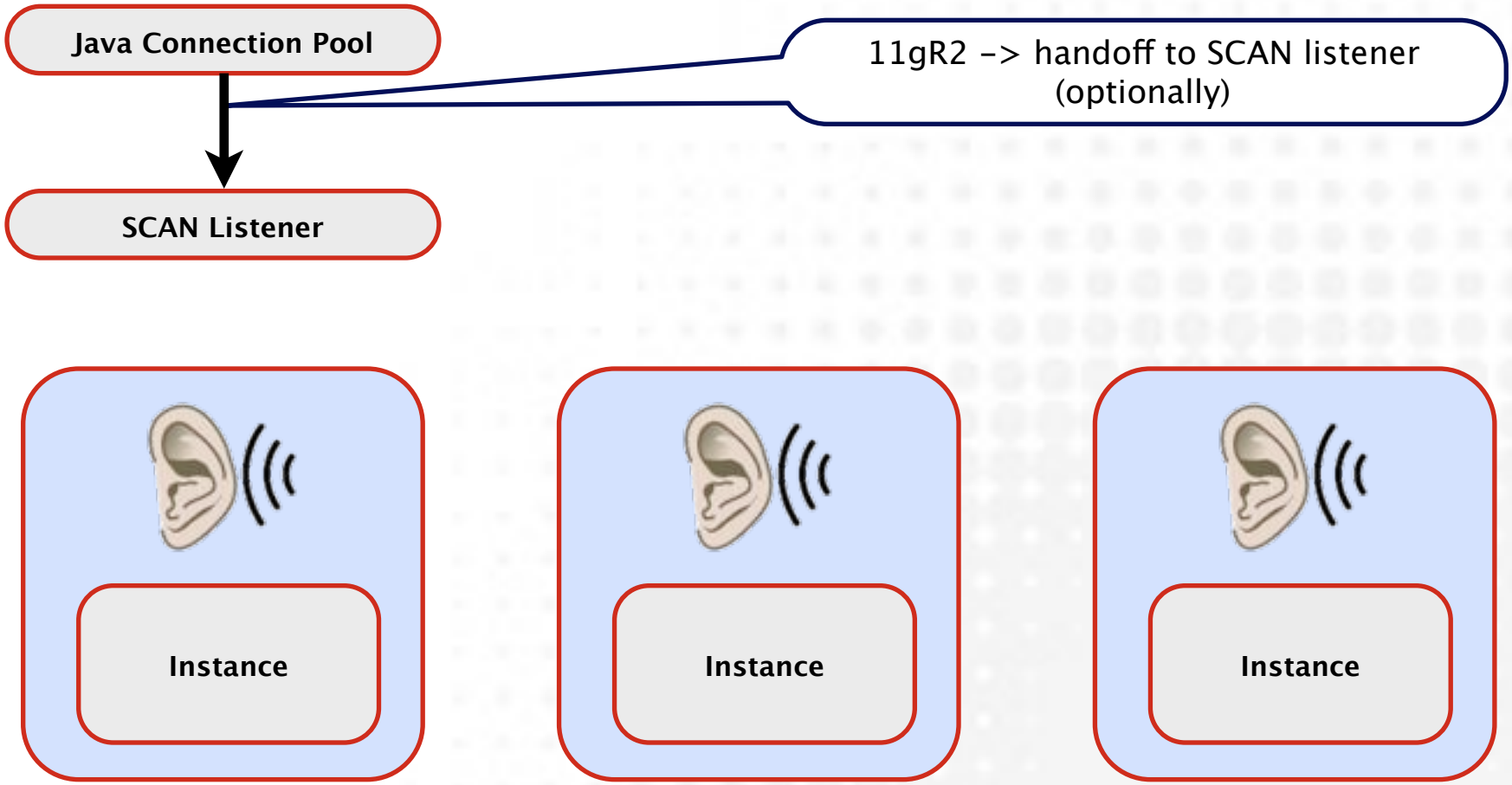
Java Connection Pool

Establishing a Connection

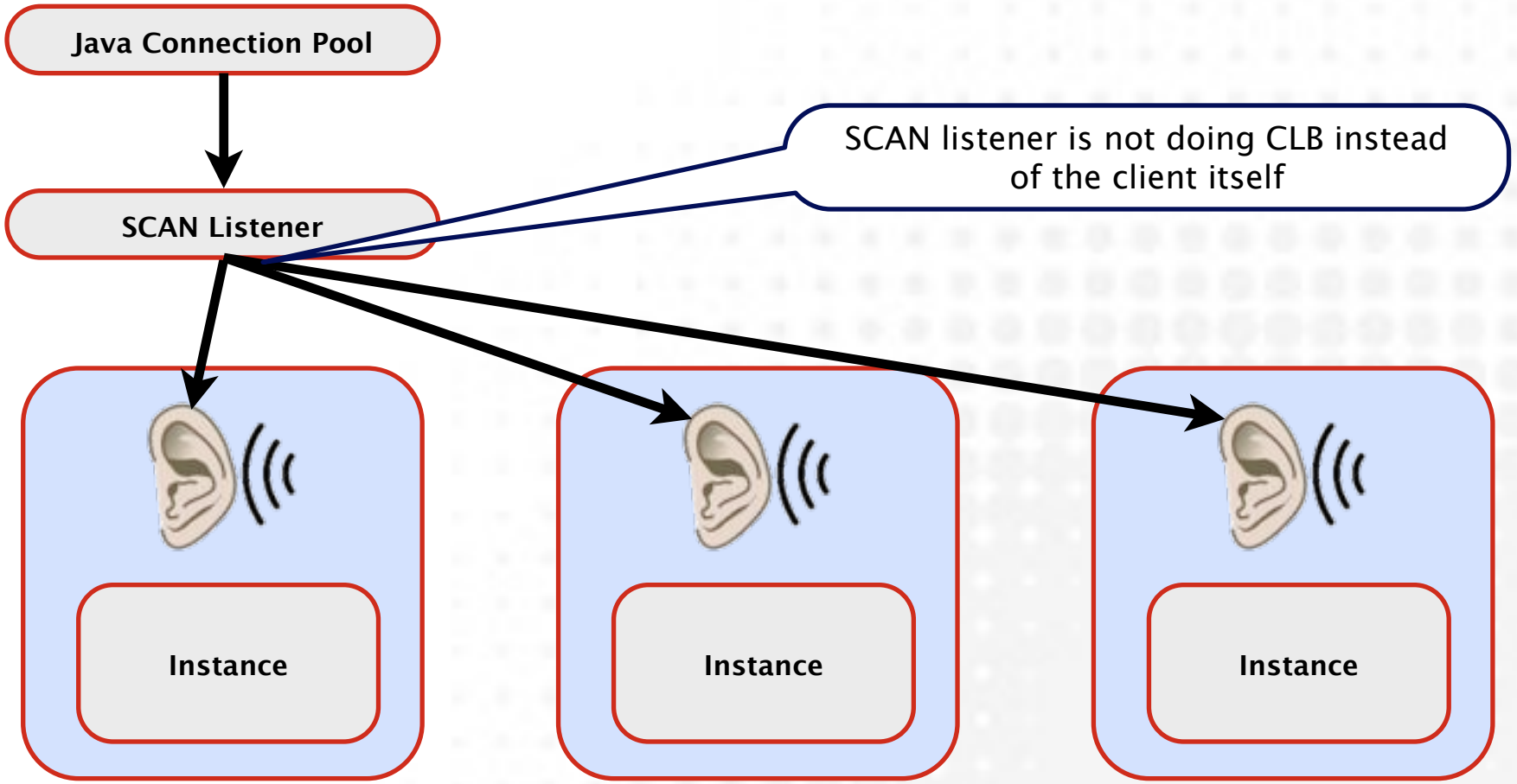
Java Connection Pool

SCAN Listener

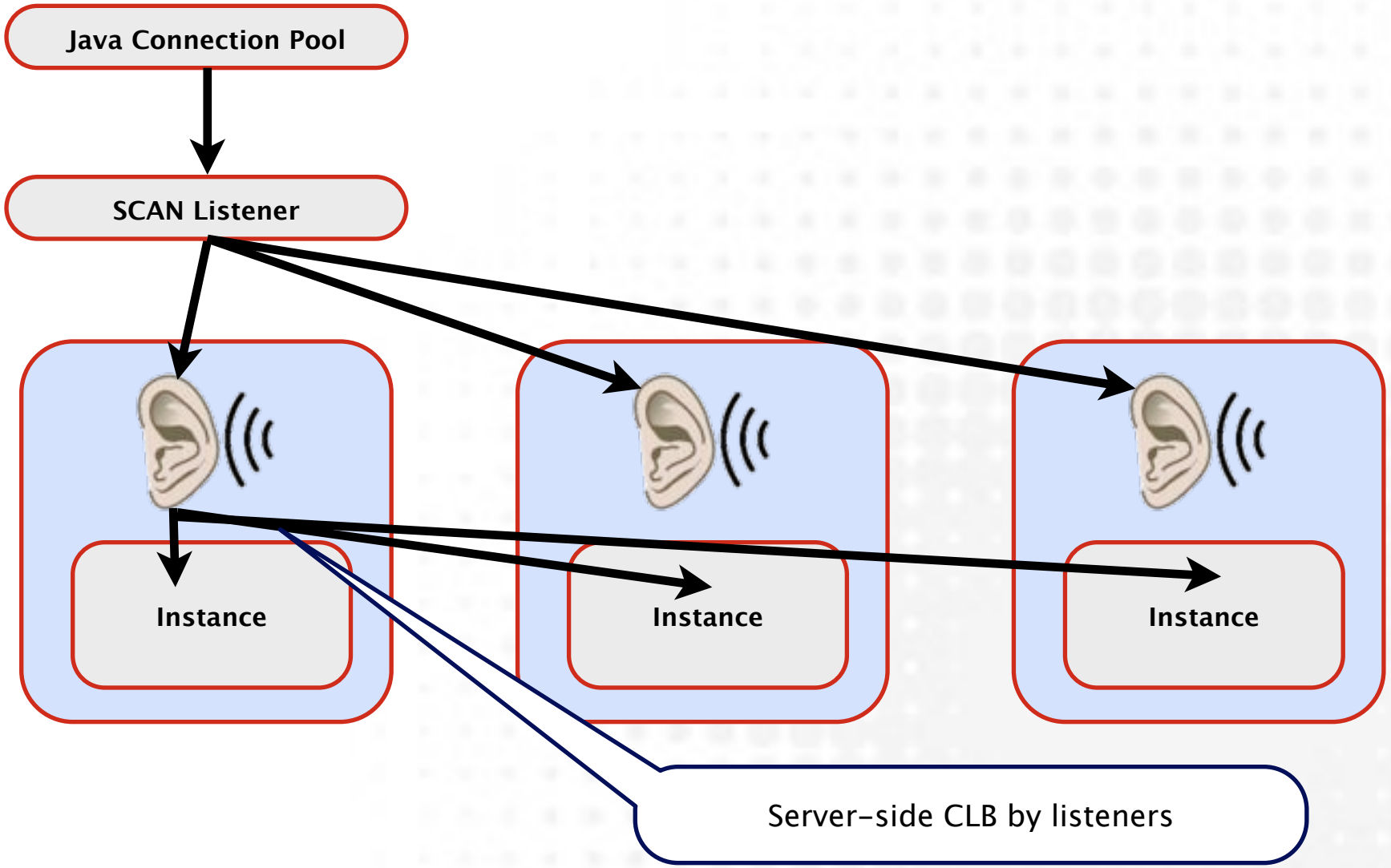
Establishing a Connection



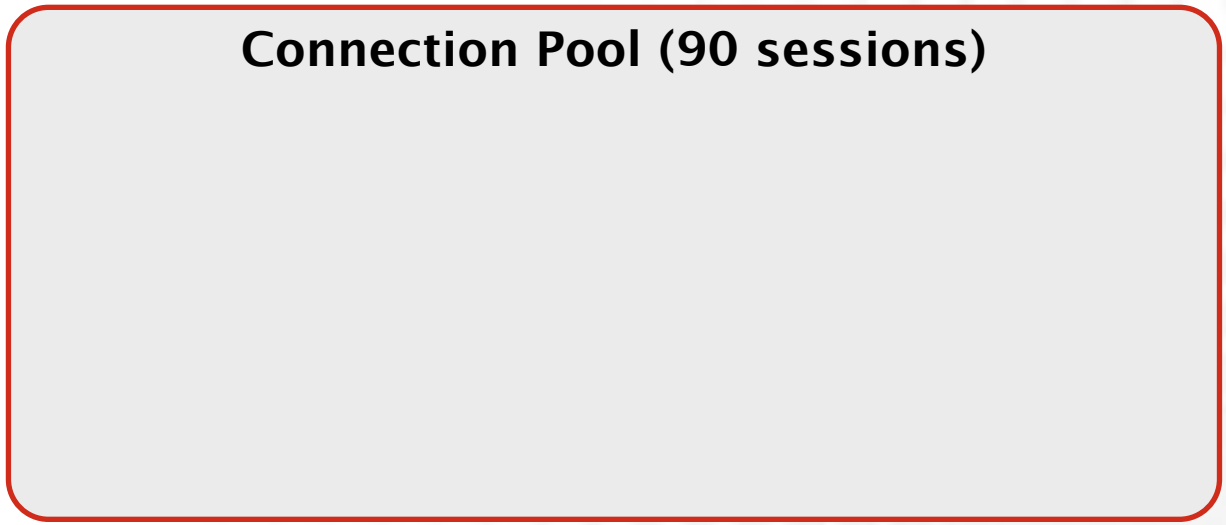
Establishing a Connection



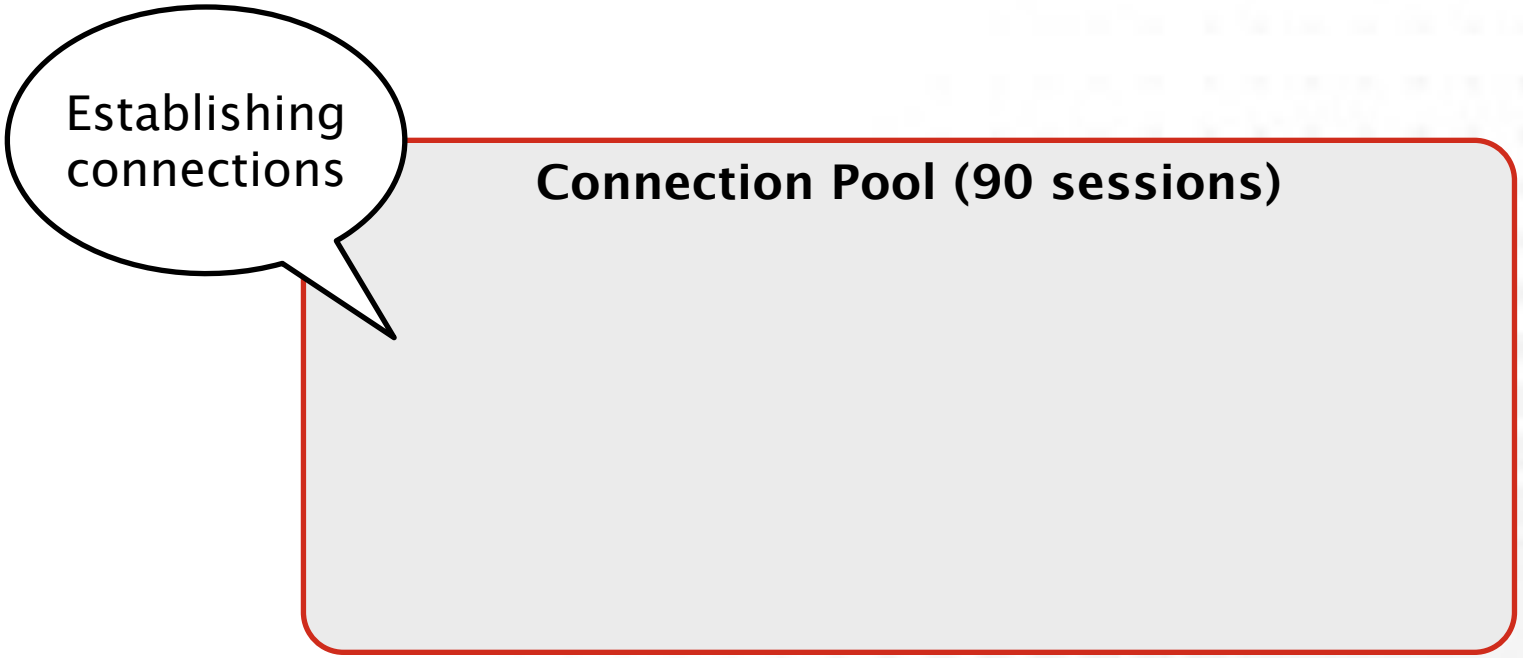
Establishing a Connection



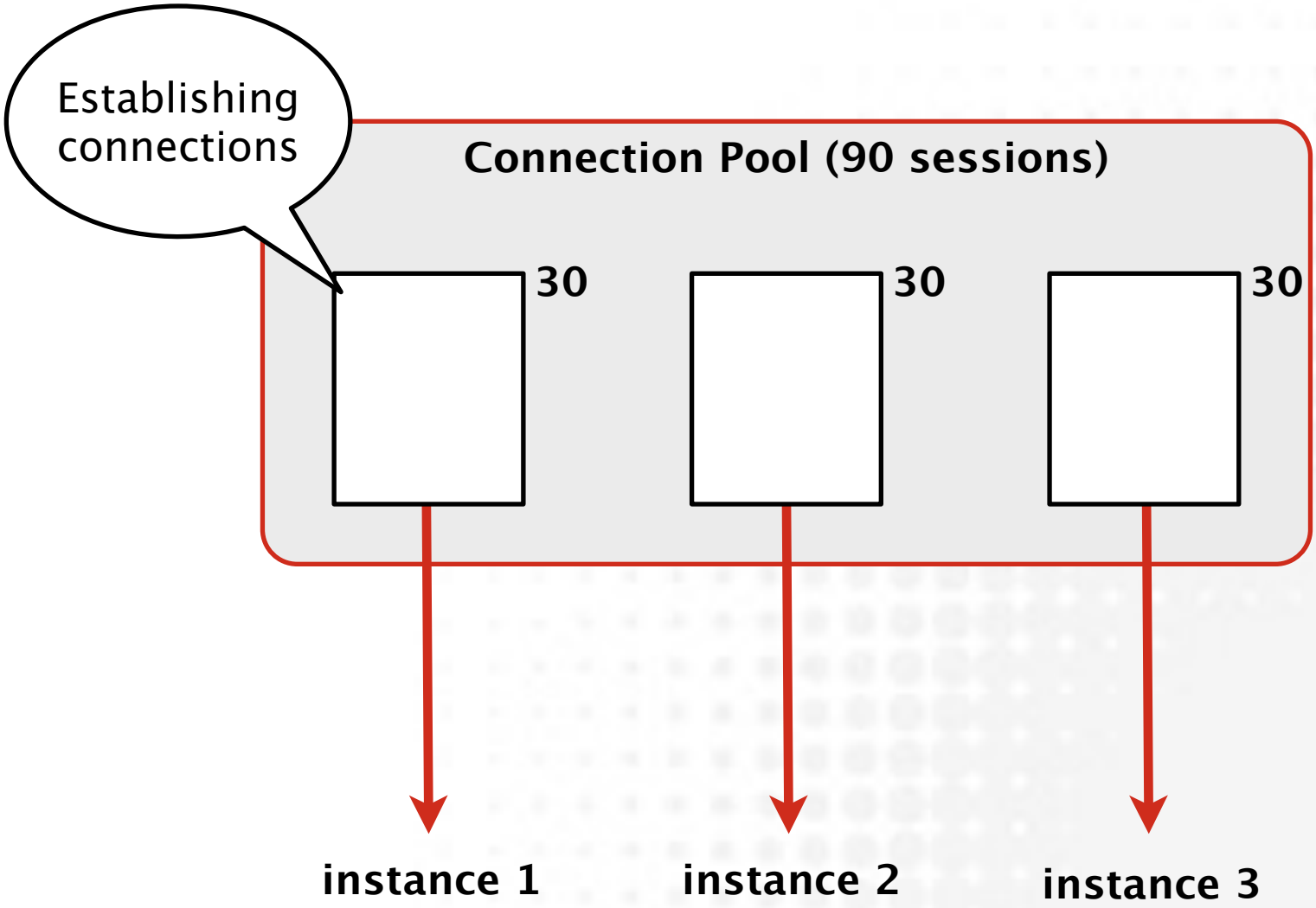
RLB - connection gravitation



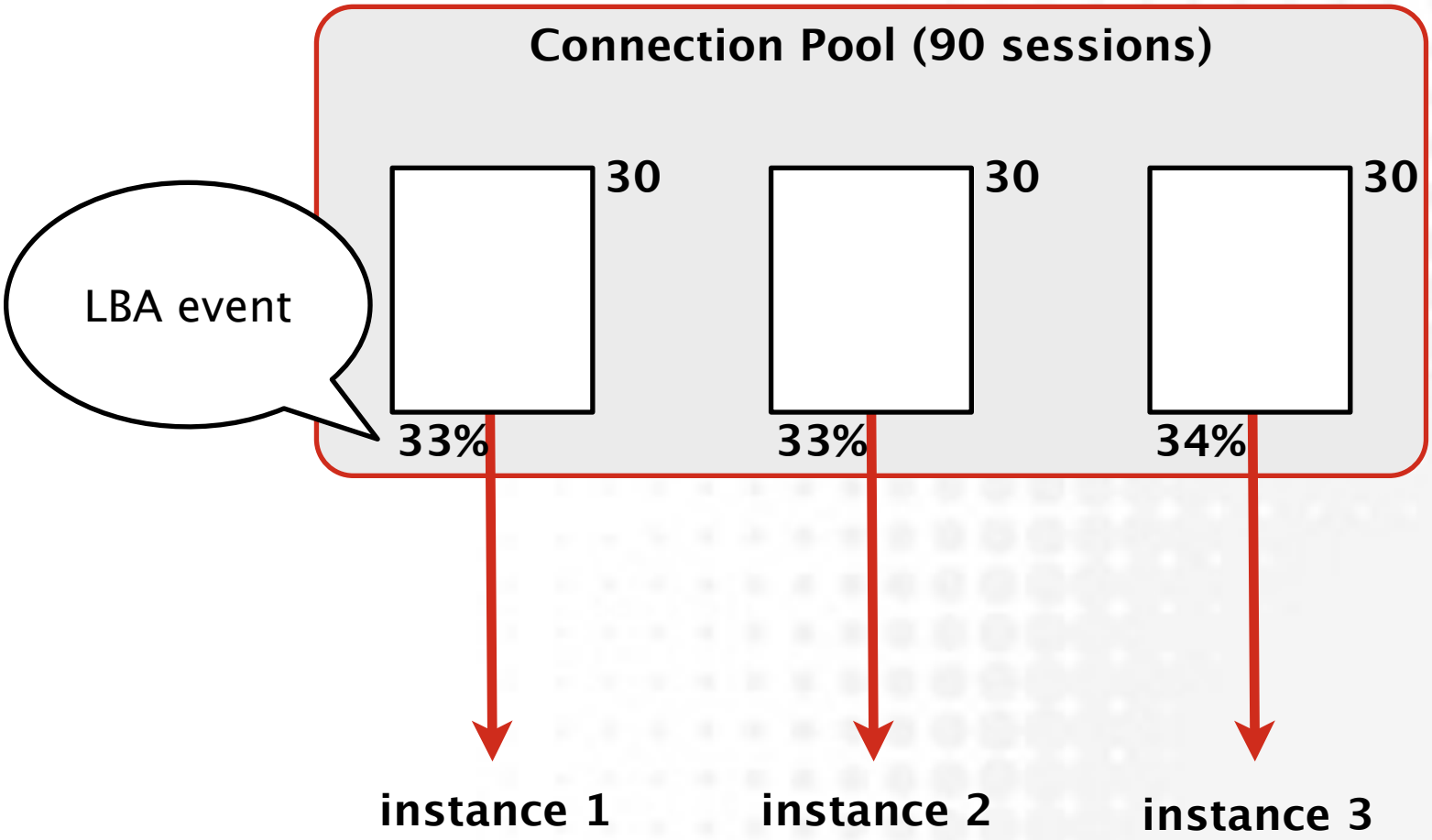
RLB - connection gravitation



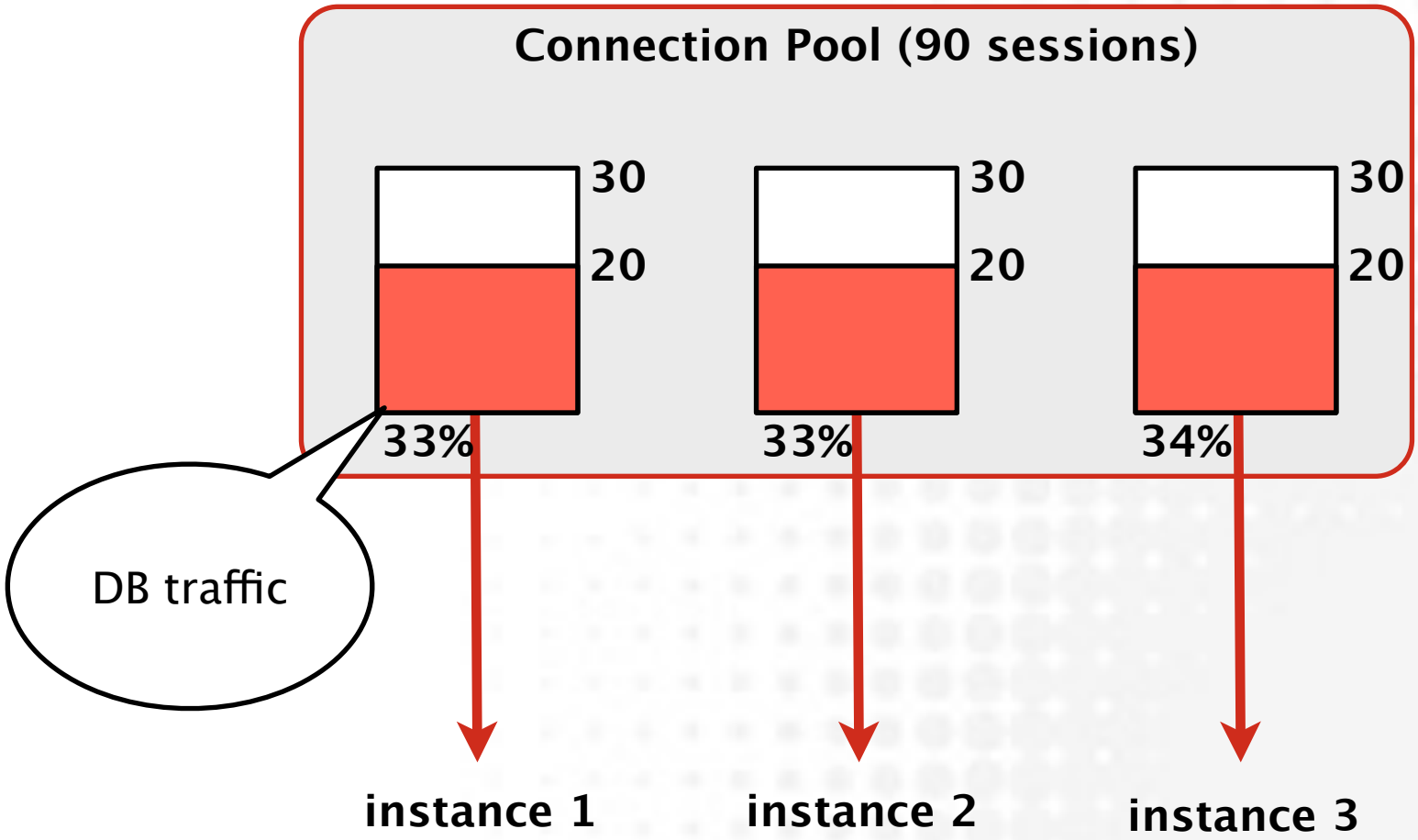
RLB - connection gravitation



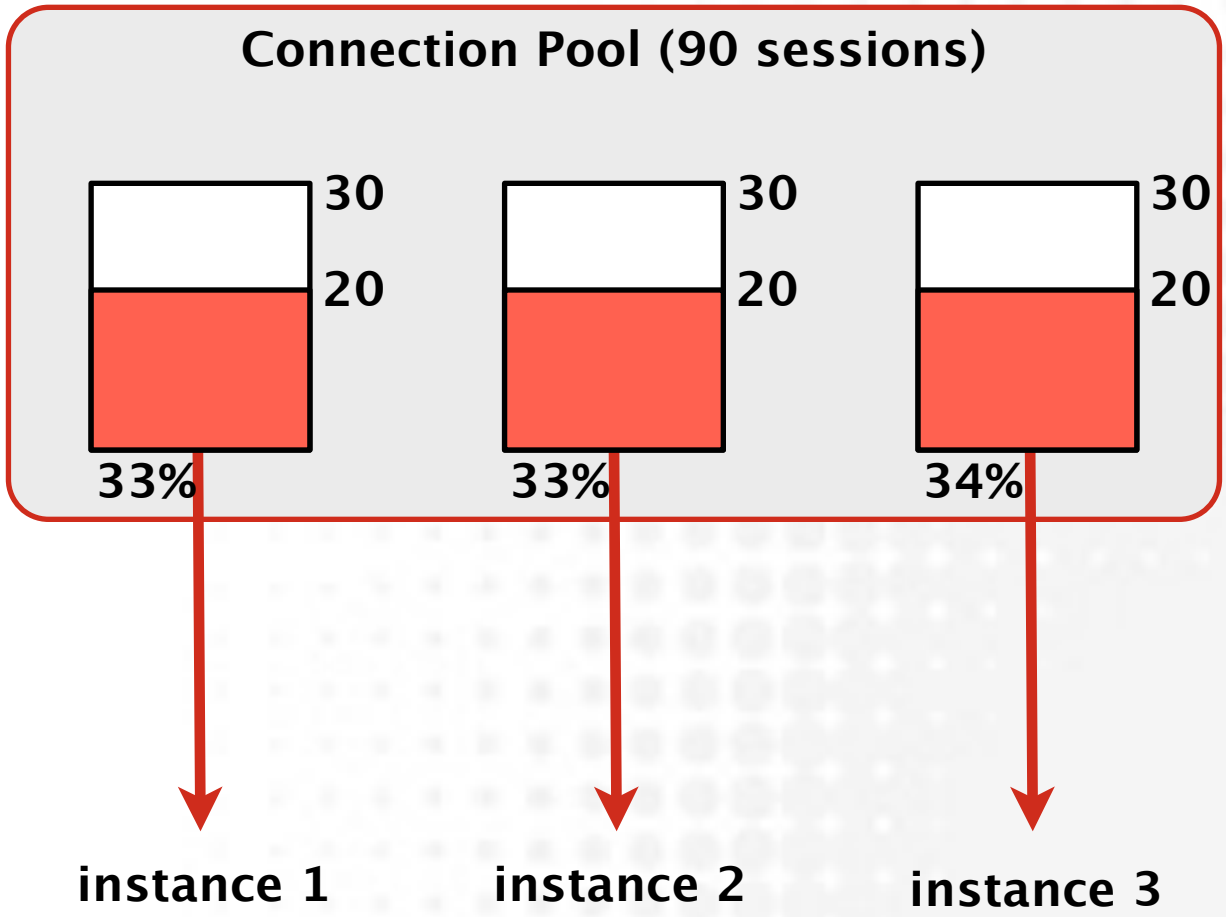
RLB - connection gravitation



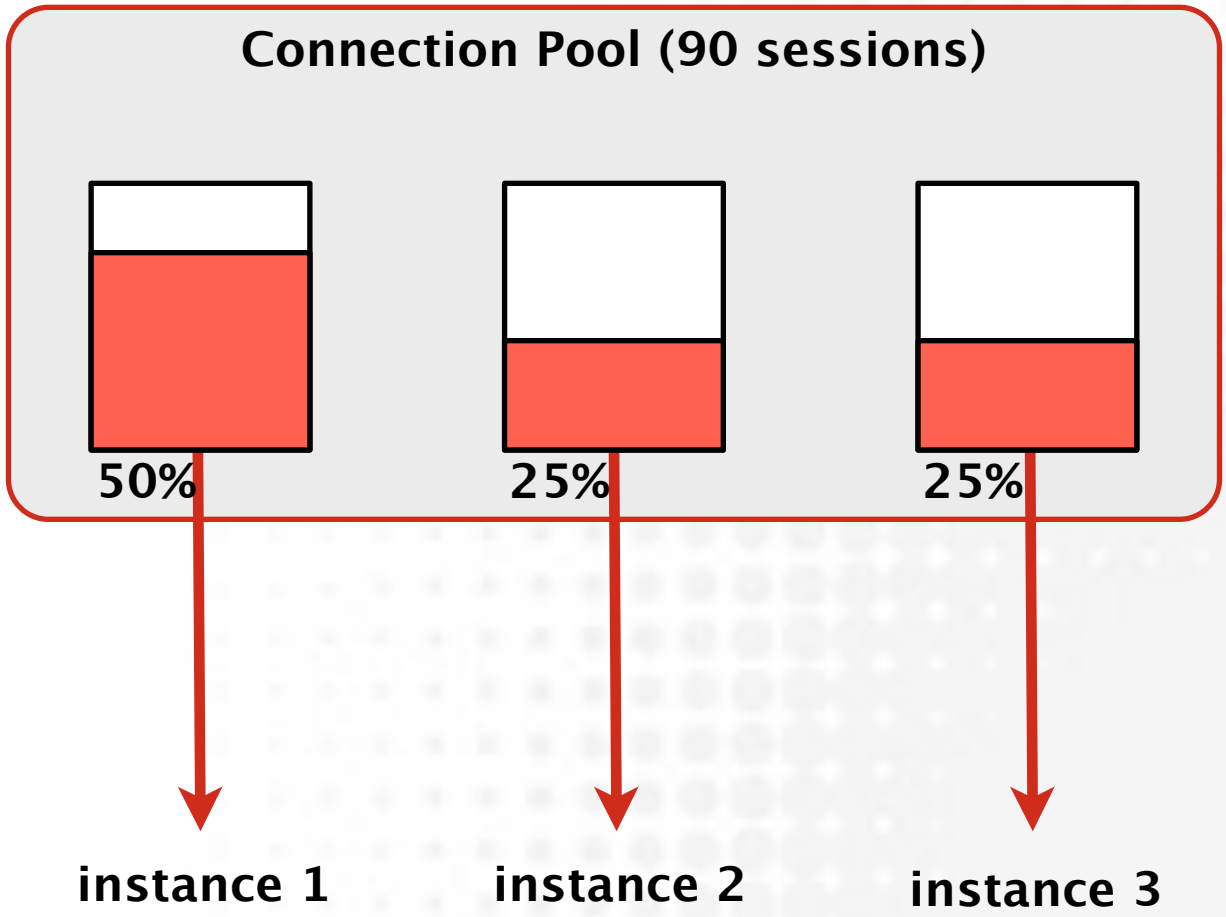
RLB - connection gravitation



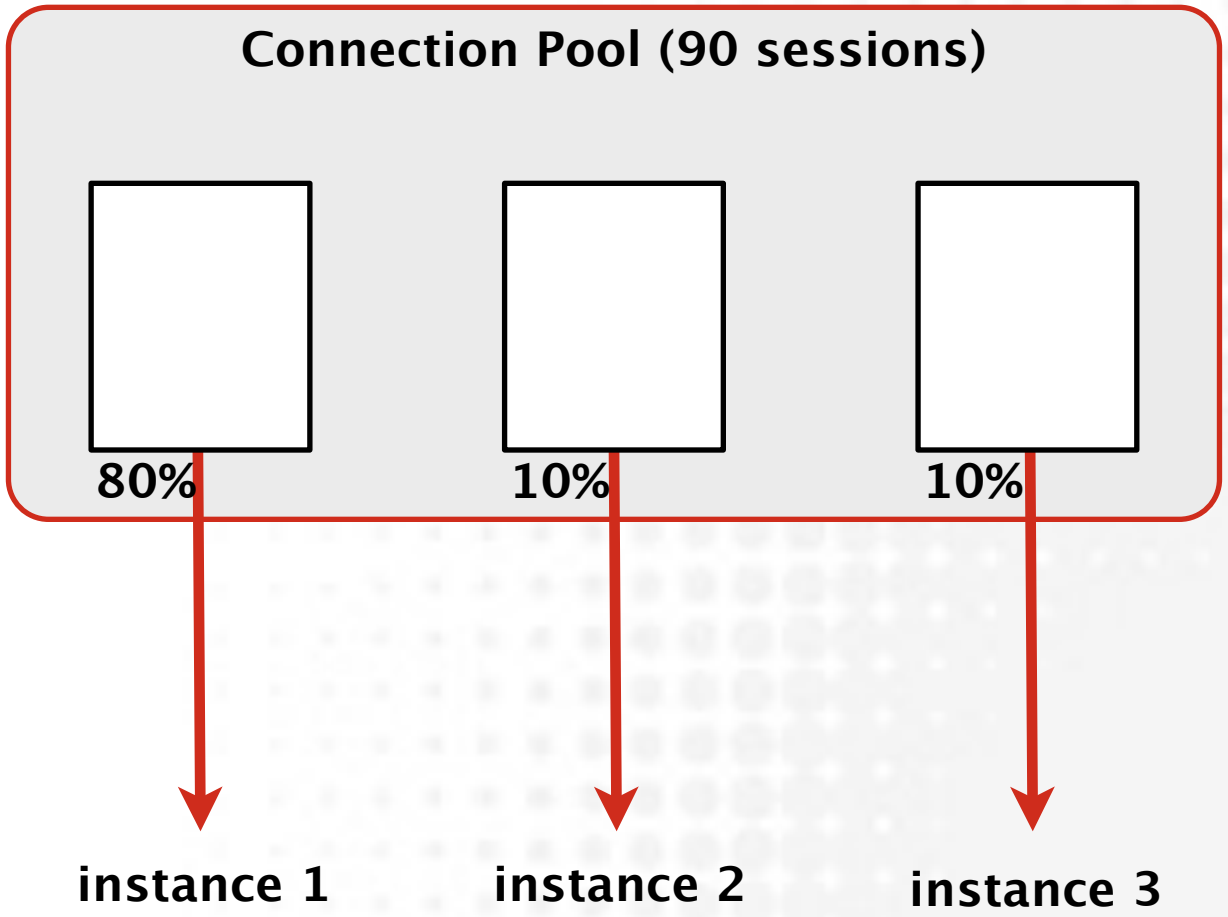
RLB - connection gravitation



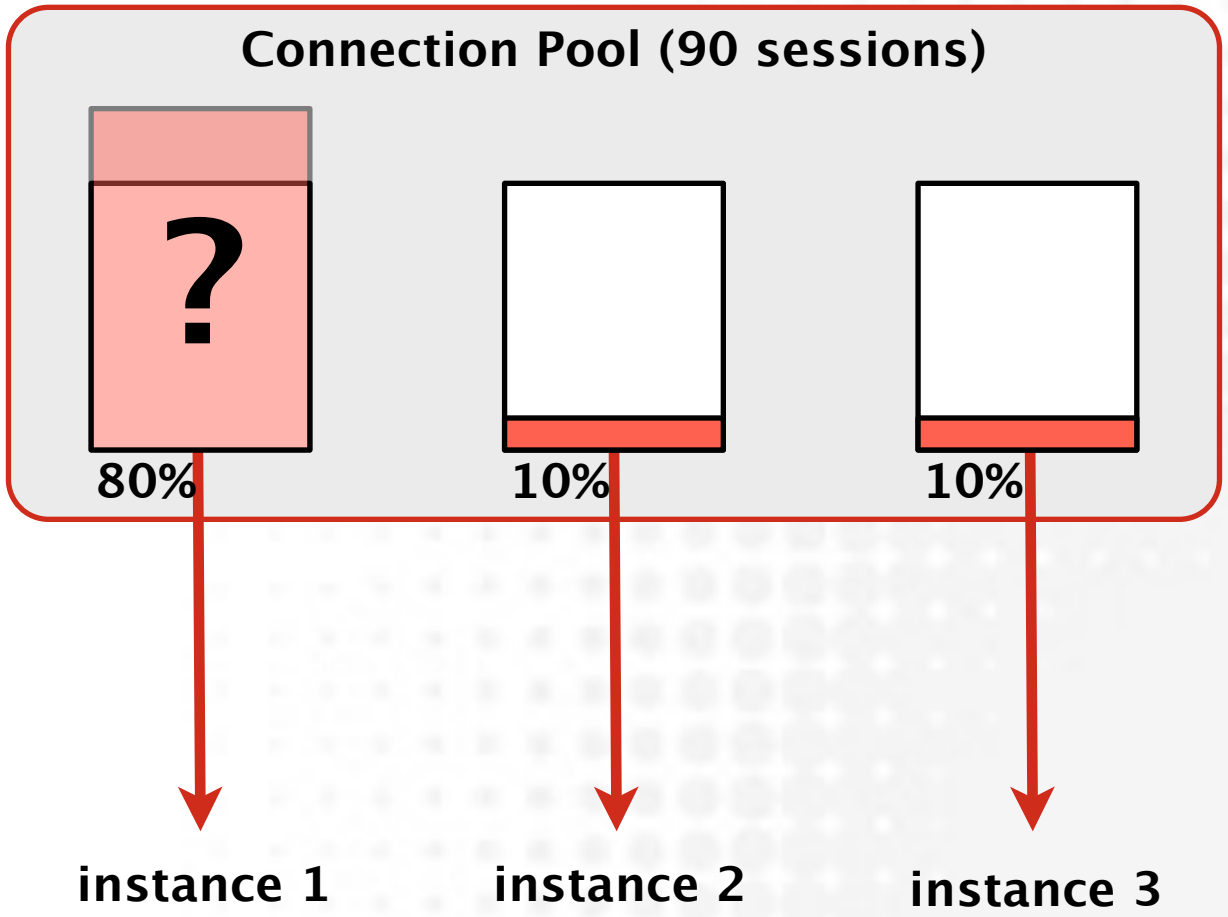
RLB - connection gravitation



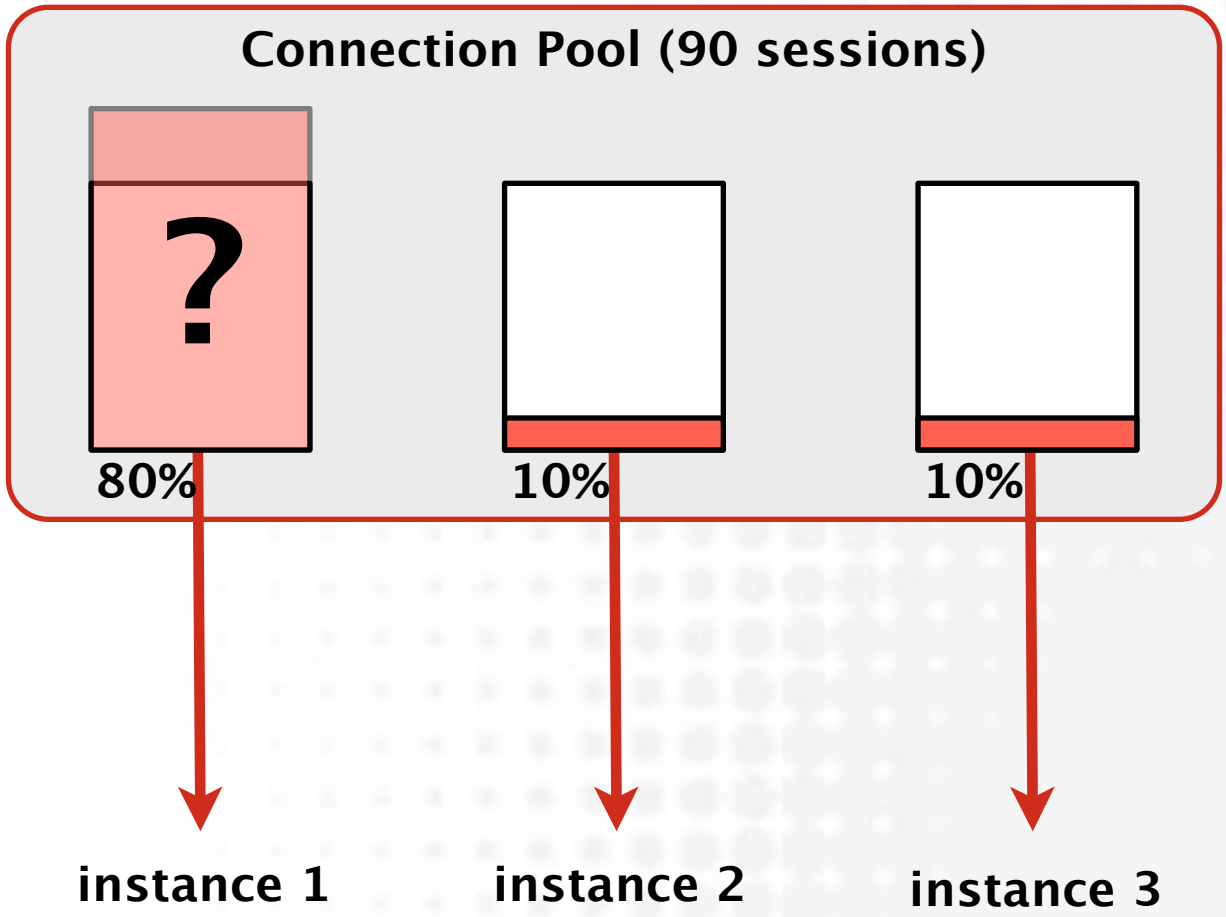
RLB - connection gravitation



RLB - connection gravitation

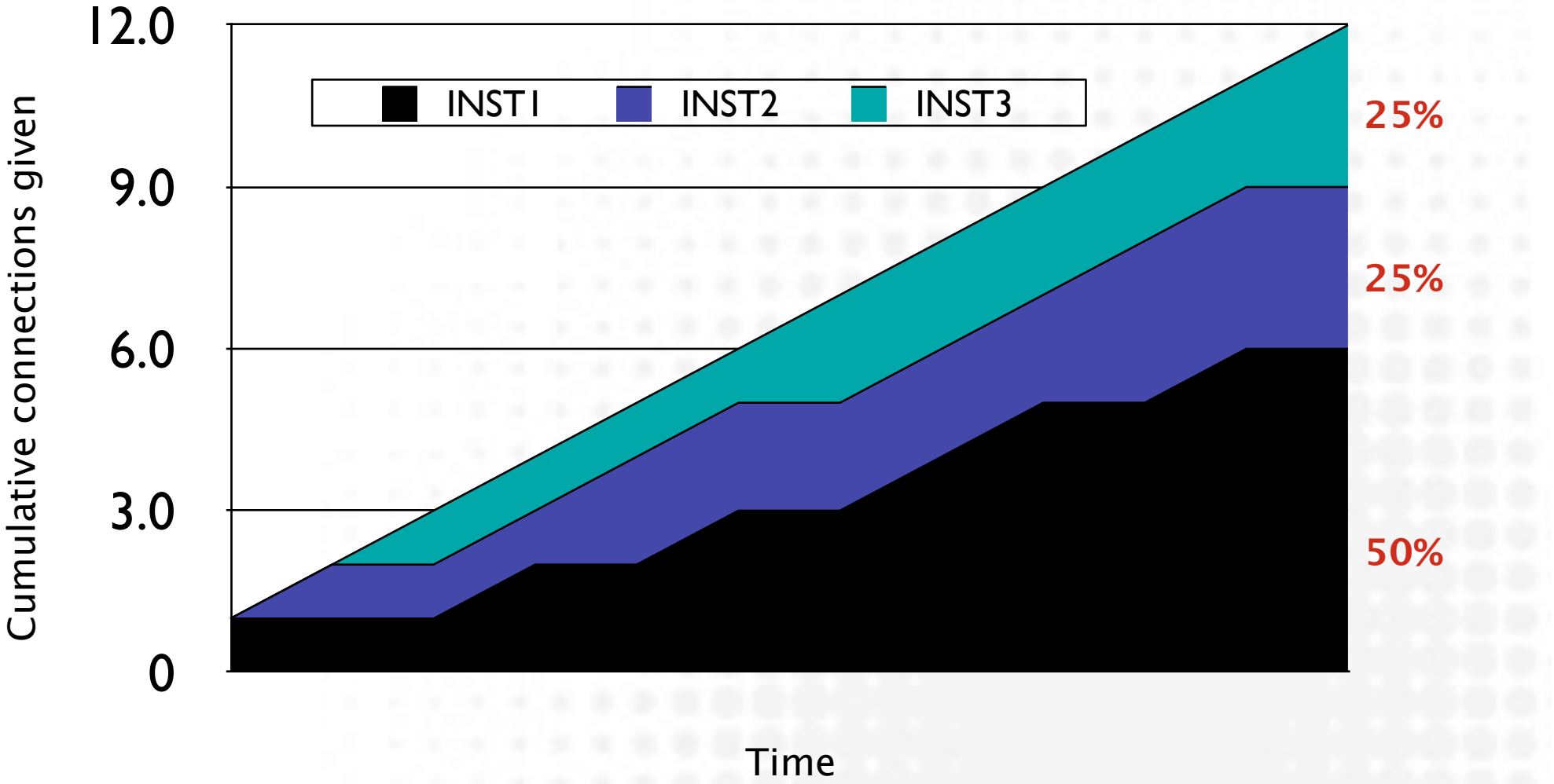


RLB - connection gravitation



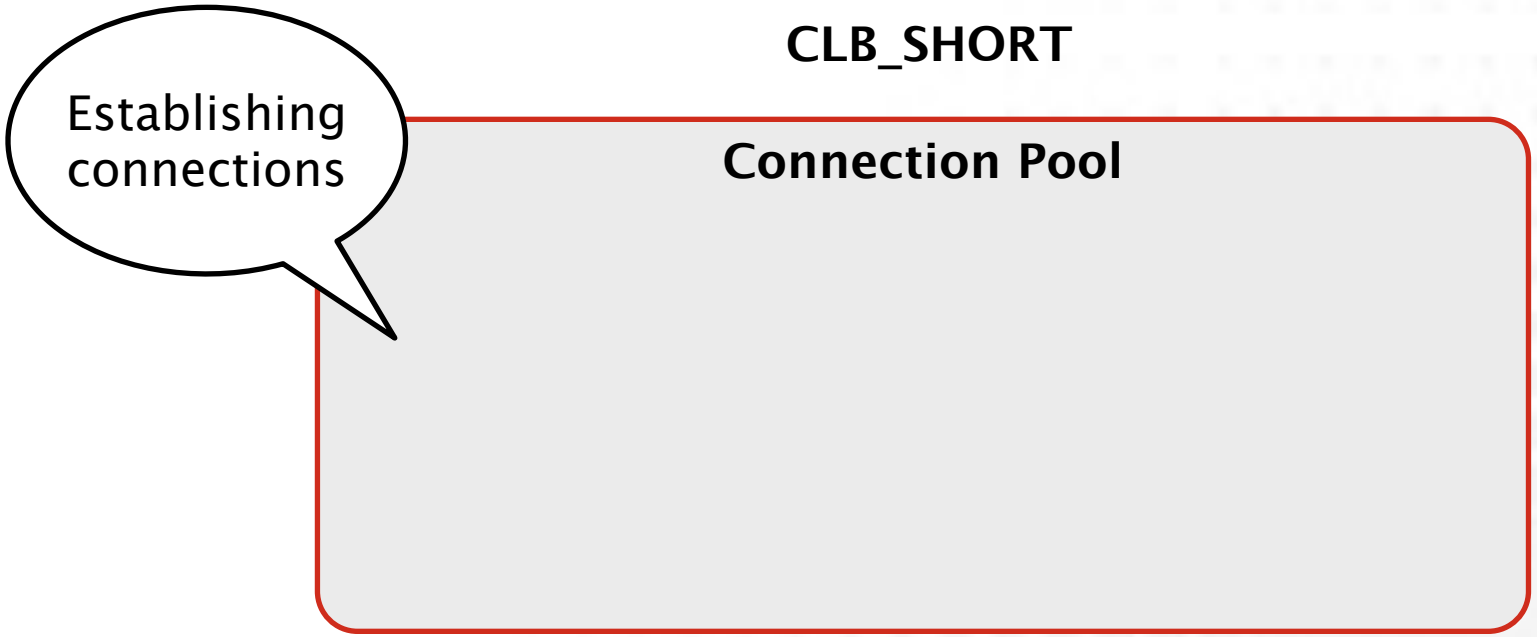
Connection pool must be over-allocated!

RLB - how is workload gravitated?

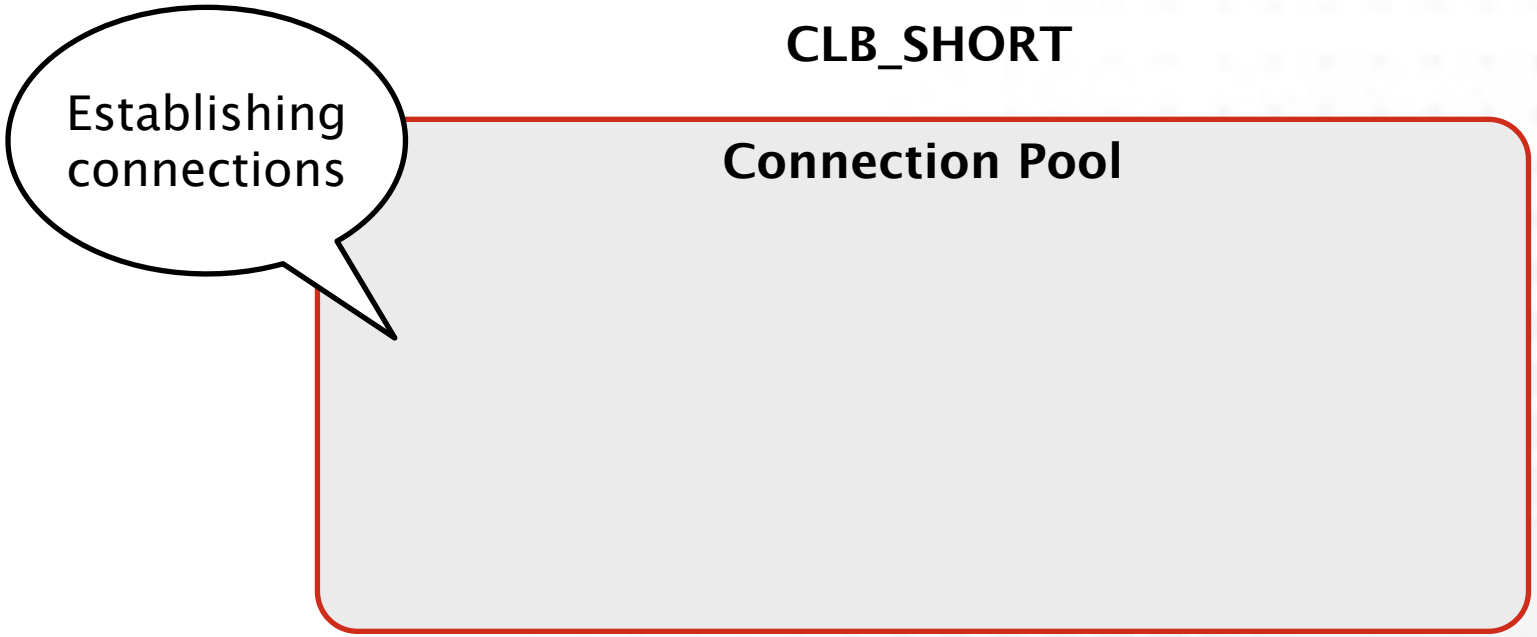


CLB options for RLB

CLB options for RLB

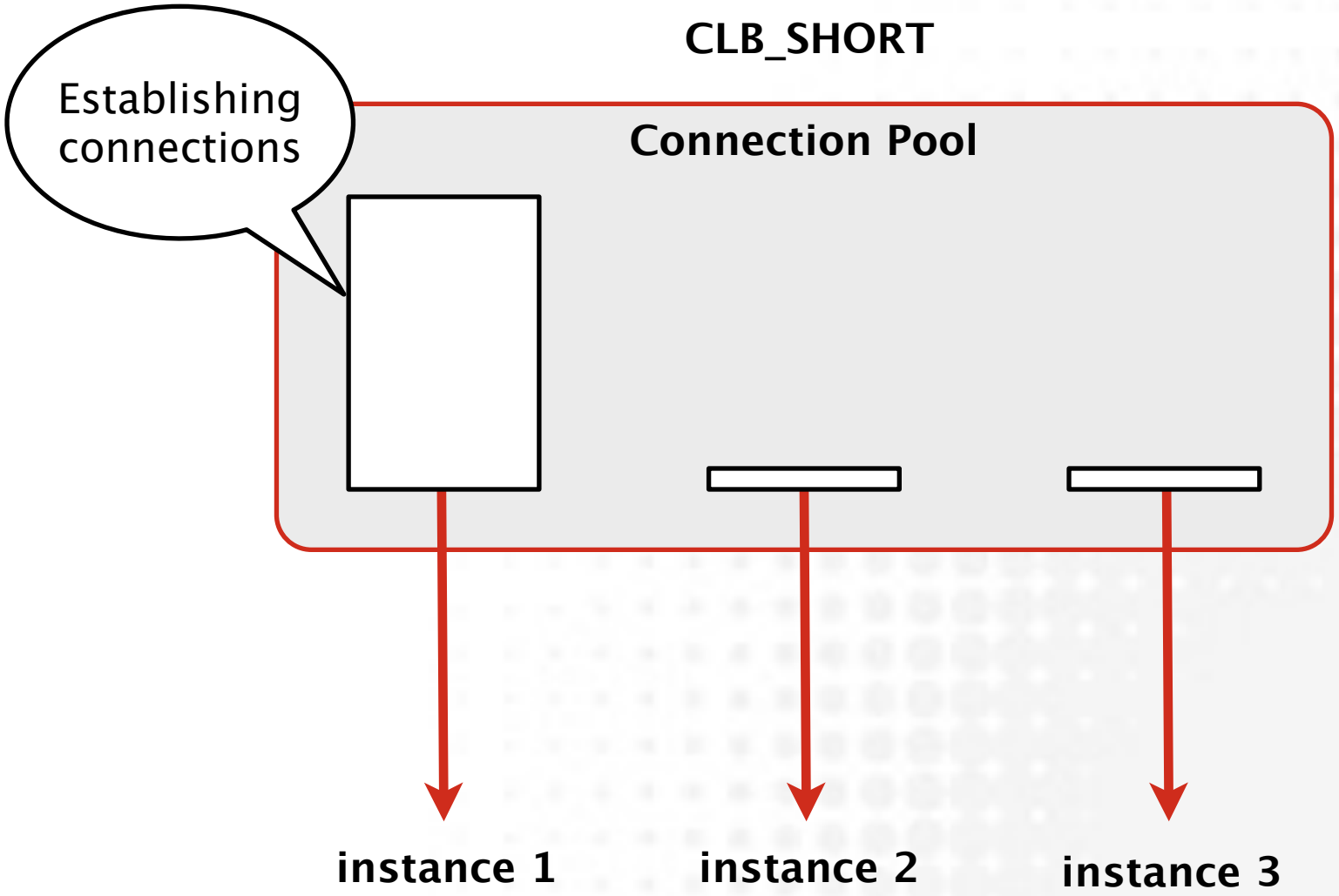


CLB options for RLB



Connections requests burst – most go to the node with the lowest goodness

CLB options for RLB



Connections requests burst – most go to the node with the lowest goodness

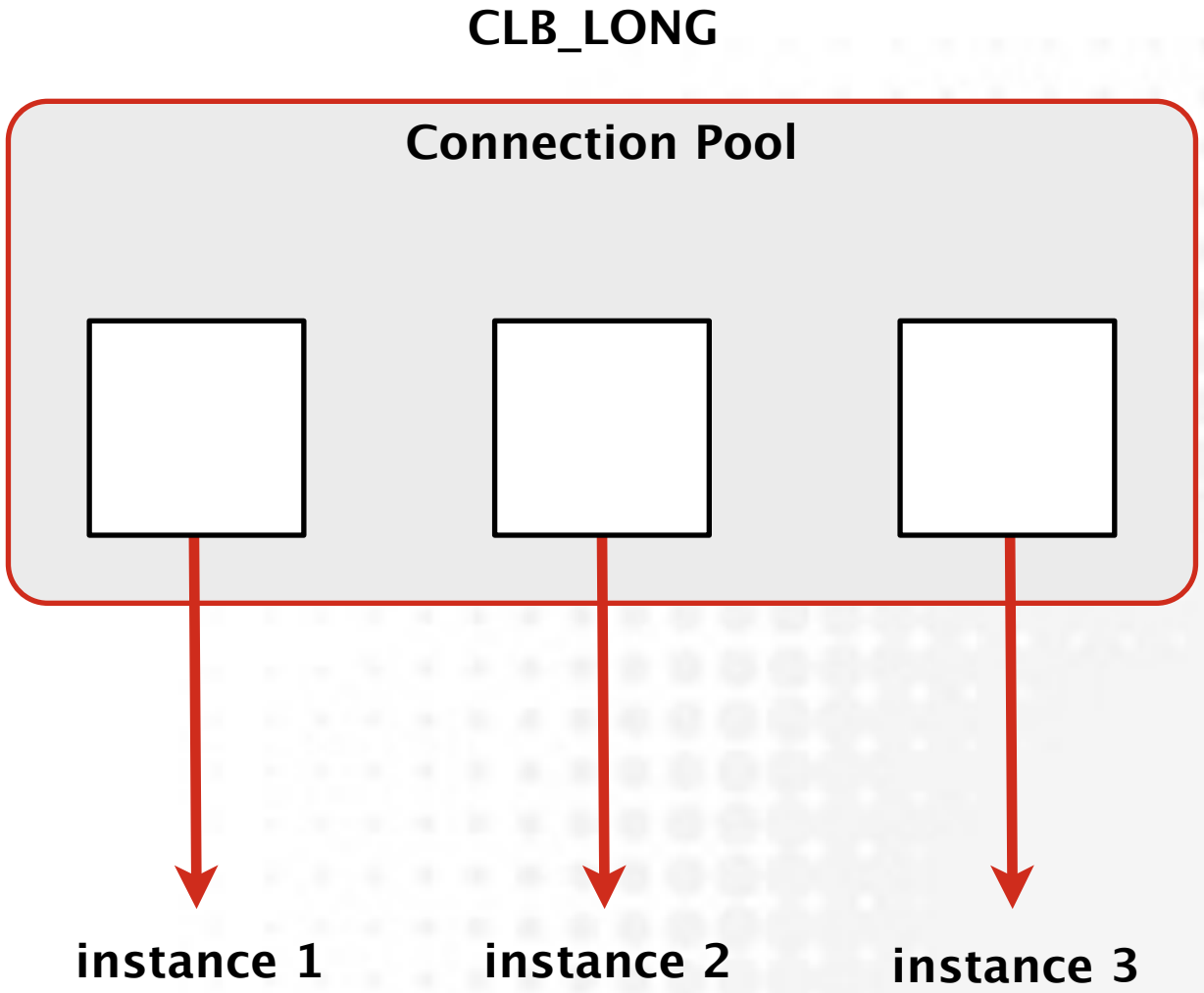
CLB options for RLB

CLB_LONG

instance 1 **instance 2** **instance 3**

Connections requests distributed equally across all instances running a service

CLB options for RLB



Connections requests distributed equally across all instances running a service

Rule of thumb

use CLB_LONG with connection pools

JDBC Implicit Connection Cache

- Standard JDBC drivers + ons.jar

```
ods.setConnectionCachingEnabled(True);
ods.setFastConnectionFailoverEnabled(True);
ods.setONSConfiguration
      ("nodes=lh1:6201,lh2:6201");
  Properties prop = new Properties();
  prop.setProperty("MinLimit", MIN_CONN);
  prop.setProperty("MaxLimit", MAX_CONN);
  prop.setProperty("InitialLimit", INIT_CONN);
  prop.put
     (oracle.net.ns.SQLnetDef.TCP_CONNTIMEOUT_STR,
     "1000")); // 1 second
  ods.setConnectionCacheProperties(prop);
```

Demo 7 - JDBC RLB

- Implicit Connection Cache
- Enabling FCF
- Subscribe with ONS
- Tracing LBA events in JDBC

ONS JDBC bugs

- Bug 6151350 - RESTRICTION ON SETONSCONFIGURATION

- Bug 6315760 - SETONSCONFIGURATION ONLY REGISTERS A MAXIMUM OF ***THREE*** NODES
  - *-Doracle.ons.maxconnections=5*

Are we there yet?

- ✓Client-side basics
- ✓Oracle Cluster Services
- ✓Server-side CLB
- ✓FAN - HA & LBA events
- ✓Thin JDBC RLB
- ➡ONS Subscriber
- Your own LBA!

Demo 8 - ONC JDBC API

- Local ONS is required on client
- Subscribing to events
  - How to filter only required events
- Parsing events

Are we there YET???

- ✓Client-side basics
- ✓Oracle Cluster Services
- ✓Server-side CLB
- ✓FAN - HA & LBA events
- ✓Thin JDBC RLB
- ✓ONS Subscriber
- ➡Your own LBA!

Demo 9 - your own LBA

- Maximum flexibility of LBA
- ONS Publisher
- Oracle doesn't know how to balance you workload better than you!

- This is not documented anywhere and likely unsupported

Are we there yet?

NO :)

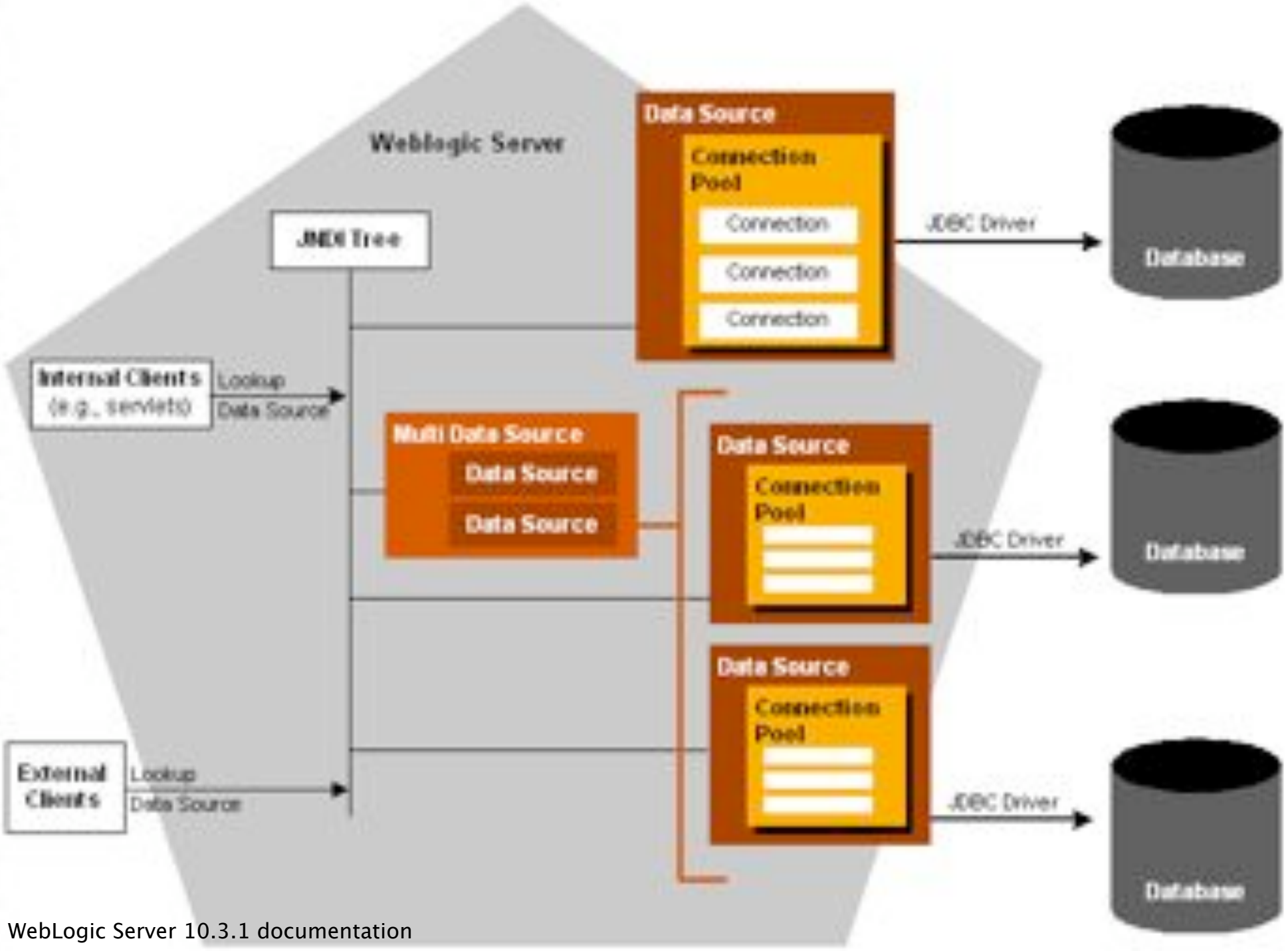
- cle Clu er Ser
- lishi a co ction
- AN A 8 BA e ts
- hin J B
- NS Sub er
- ✓Your own LBA!

11g New Features

- SCAN Listener
- Universal Connection Pool
- Connection Affinity in LBA (web sessions, XA)
  - Reduces Cache Fusion overhead
  - Requires UCP
- Data Guard Broker integration with FAN events
- Policy-Based Cluster and Capacity Management

Are we there yet?

Bonus - WebLogic Server Connection Management



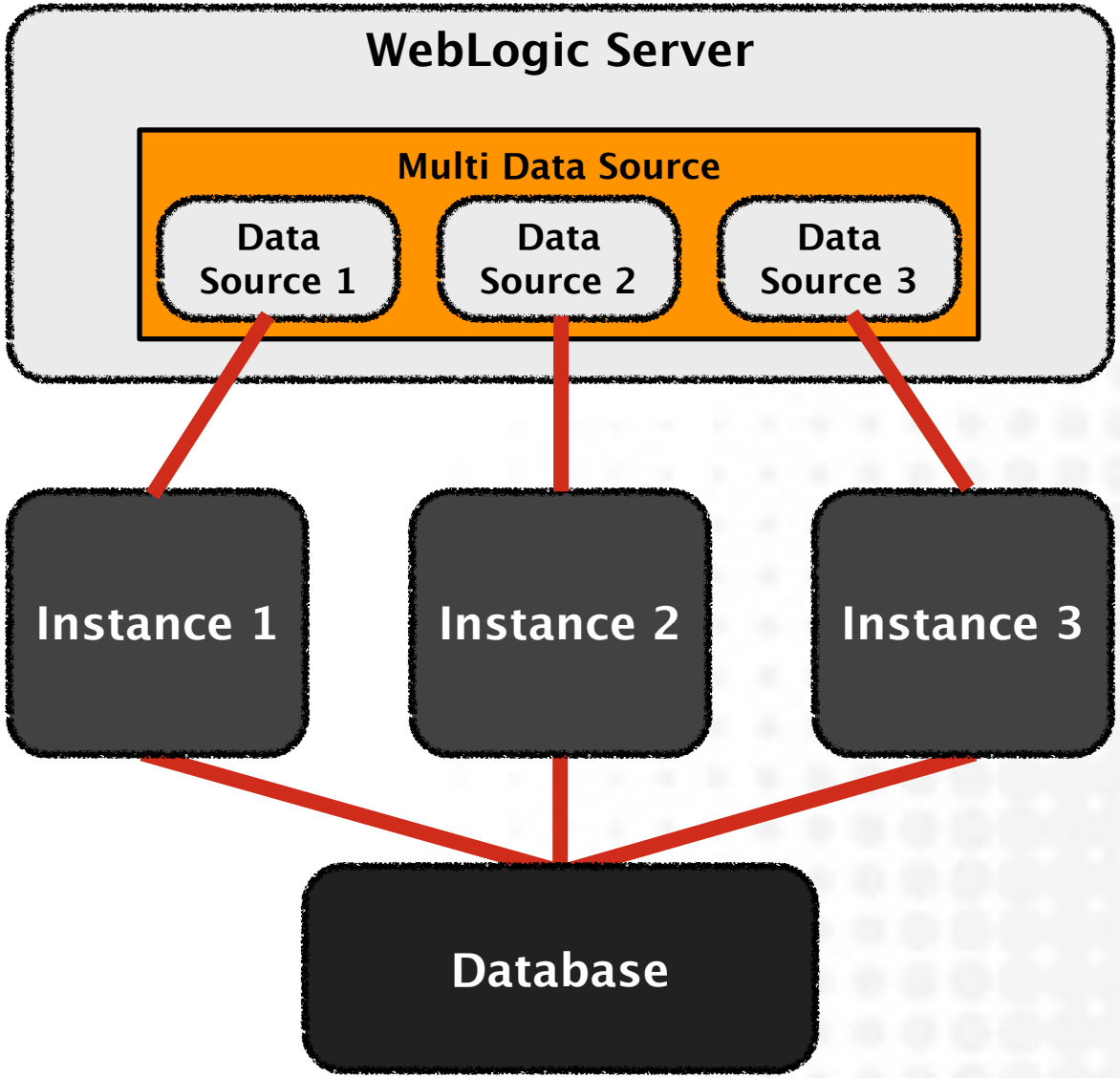
Source – WebLogic Server 10.3.1 documentation

WebLogic Configurations Options for Oracle RAC

- 4 Supported Data Source Configurations

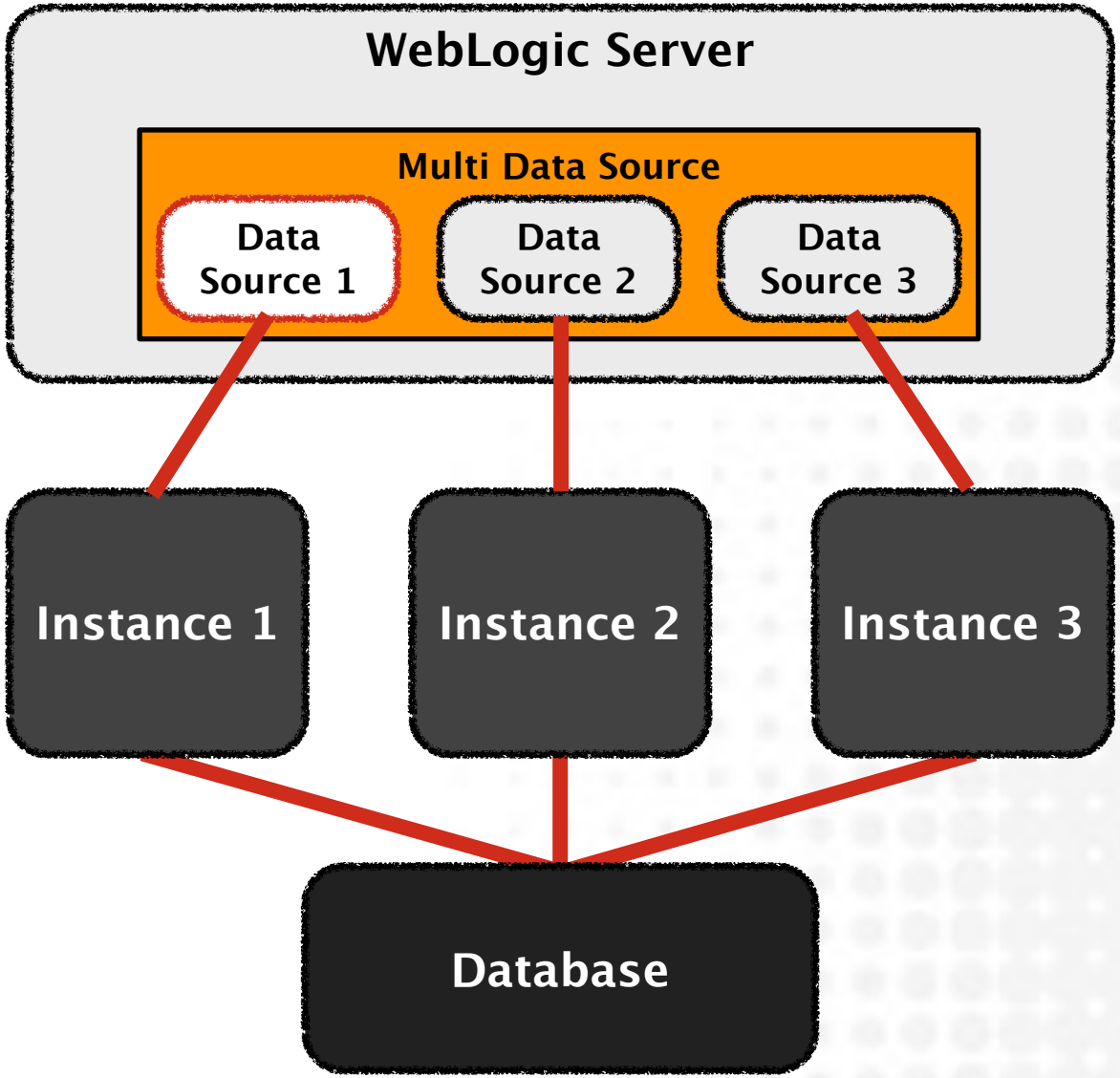
	MDS XA	MDS no-XA	RAC Services	Oracle native
Load balancing	Y	Y	Y	N
Failover	Y	Y	Y	Y
XA support	Y	N	Y	N
JDBC Store	N	Y	Y	Y
RAC Services	N	N	Y	Y

WebLogic Multi Data Source with Oracle RAC



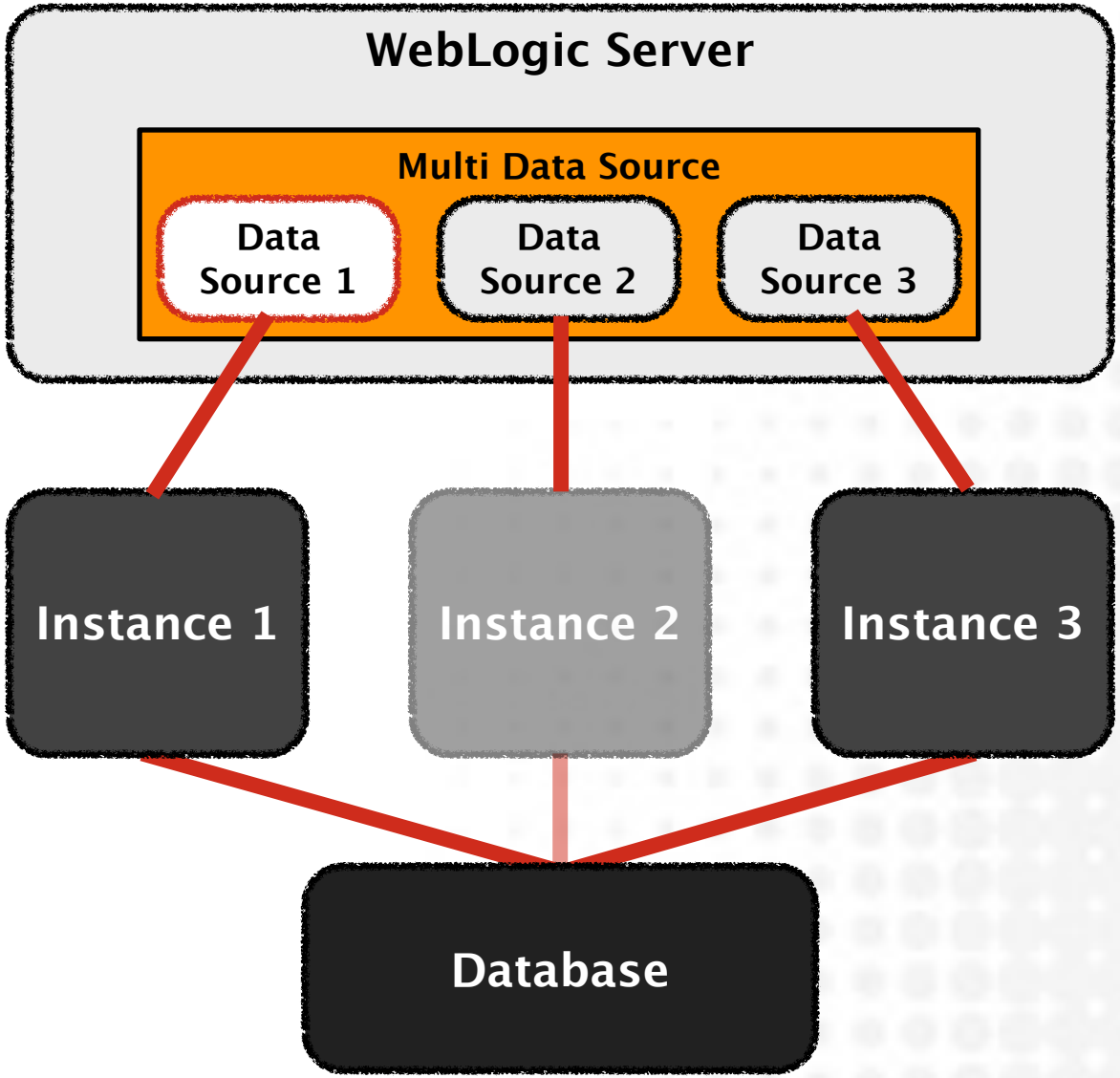
- App uses MDS
- MDS routes connection to one of its DS
  - failover or load balancing
- DS online management
  - Suspend
  - Shutdown
  - Add
  - Remove

WebLogic MDS for Failover with Oracle RAC



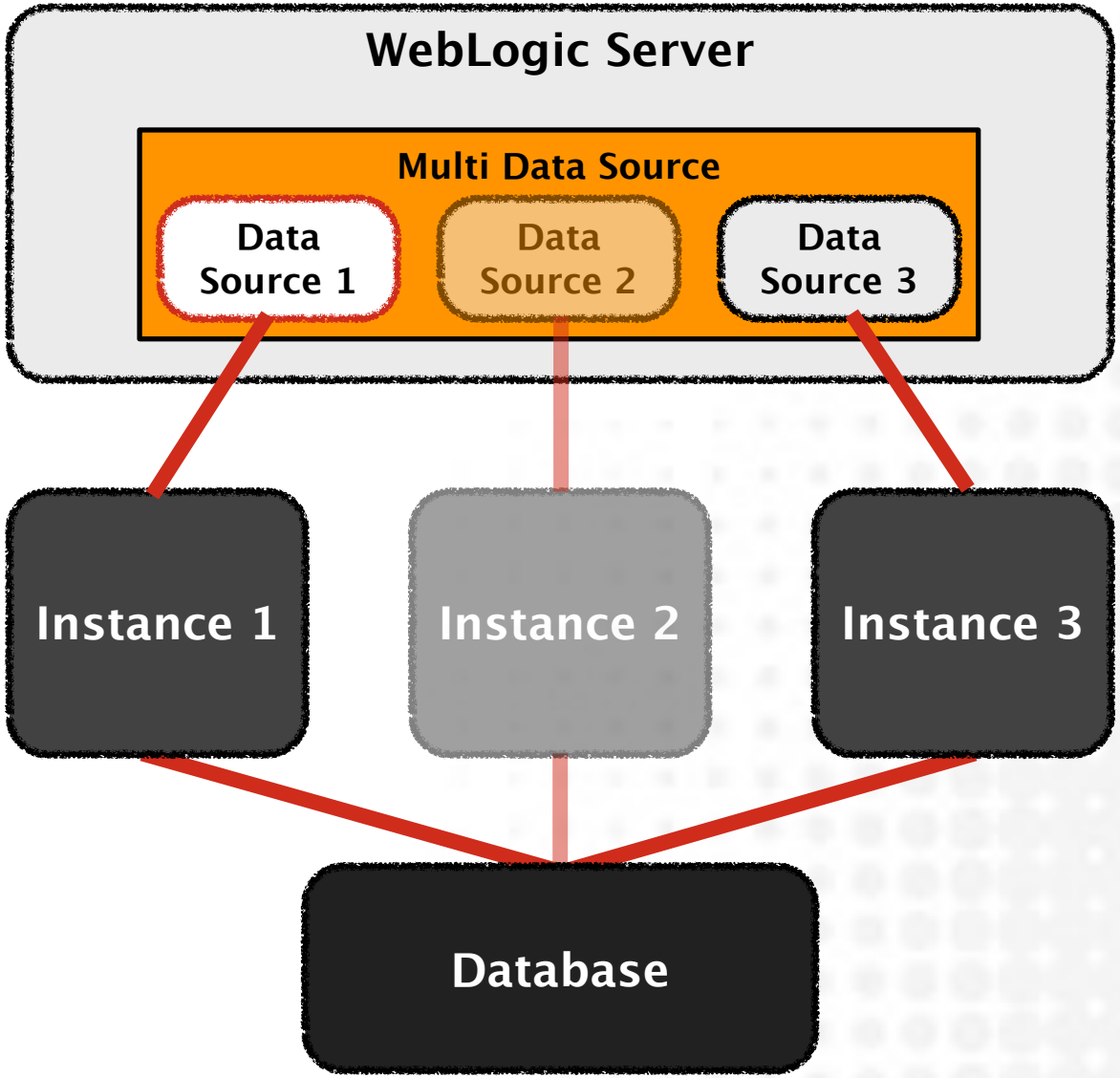
- Failover MDS
  - DS health checks
    - failure => DS dead
  - Can manually disable DS
  - Dead DS are re-tested
    - default 120 seconds

WebLogic MDS for Failover with Oracle RAC



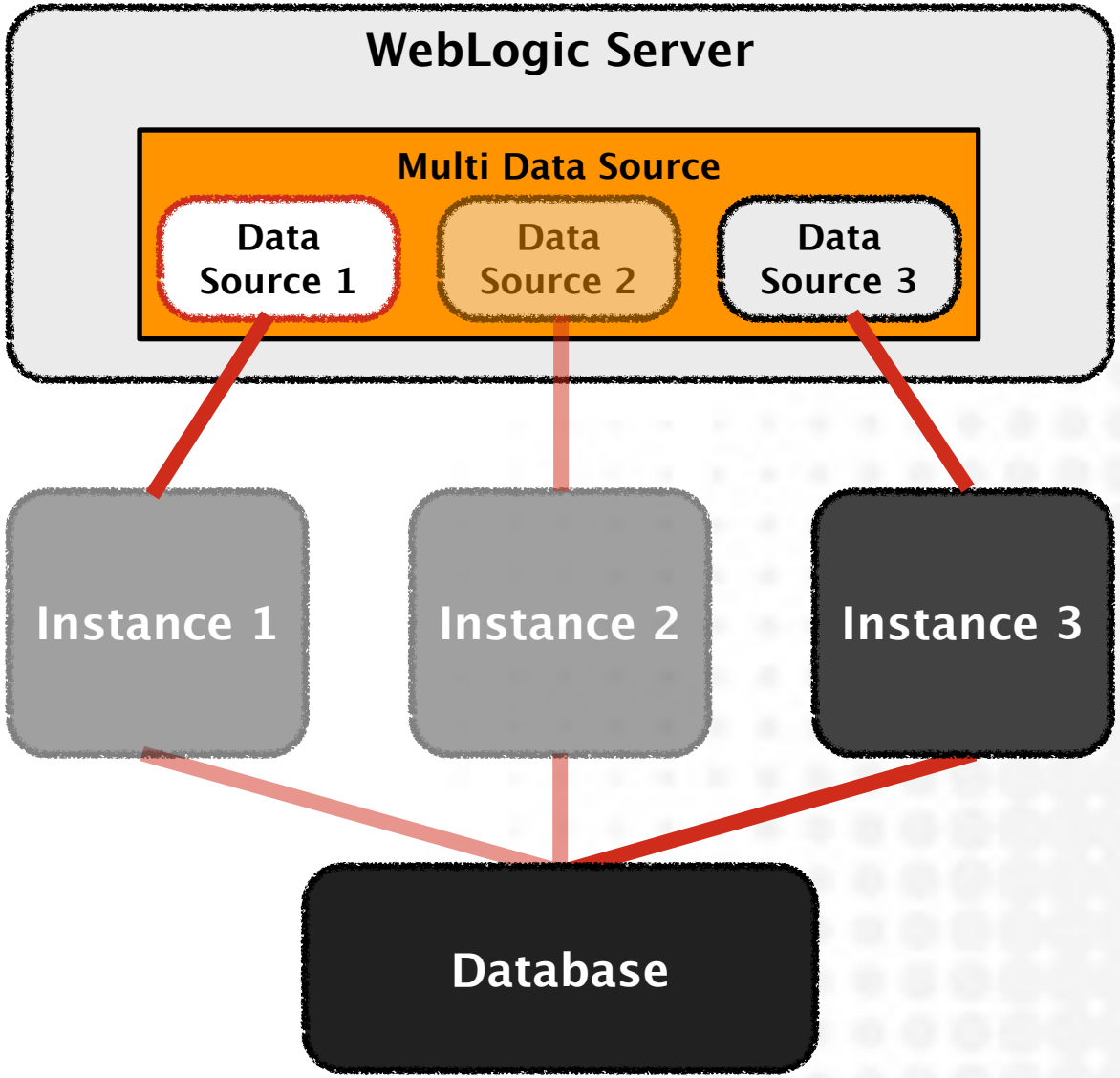
- Failover MDS
  - DS health checks
    - failure => DS dead
  - Can manually disable DS
  - Dead DS are re-tested
    - default 120 seconds

WebLogic MDS for Failover with Oracle RAC



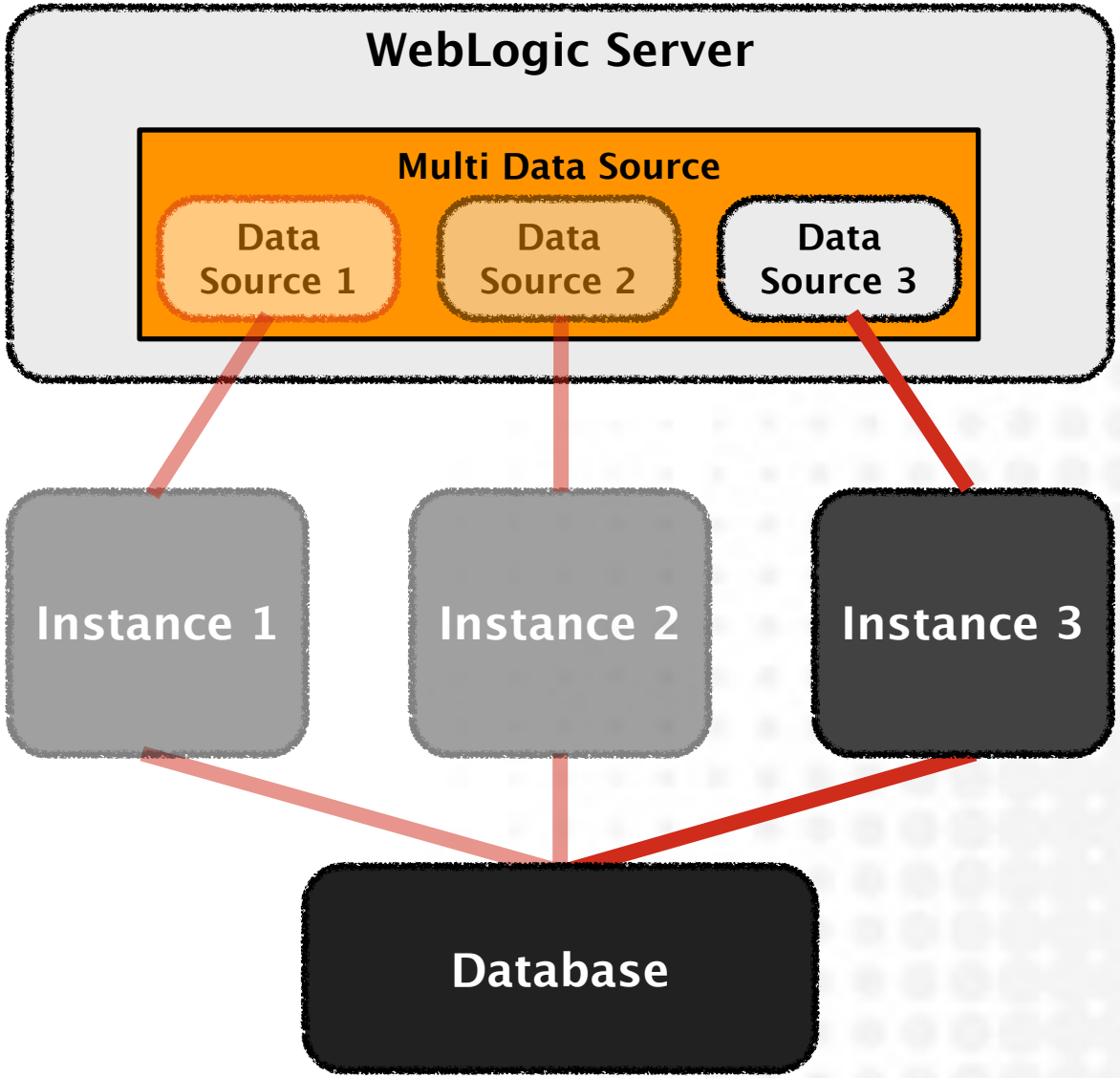
- Failover MDS
  - DS health checks
    - failure => DS dead
  - Can manually disable DS
  - Dead DS are re-tested
    - default 120 seconds

WebLogic MDS for Failover with Oracle RAC



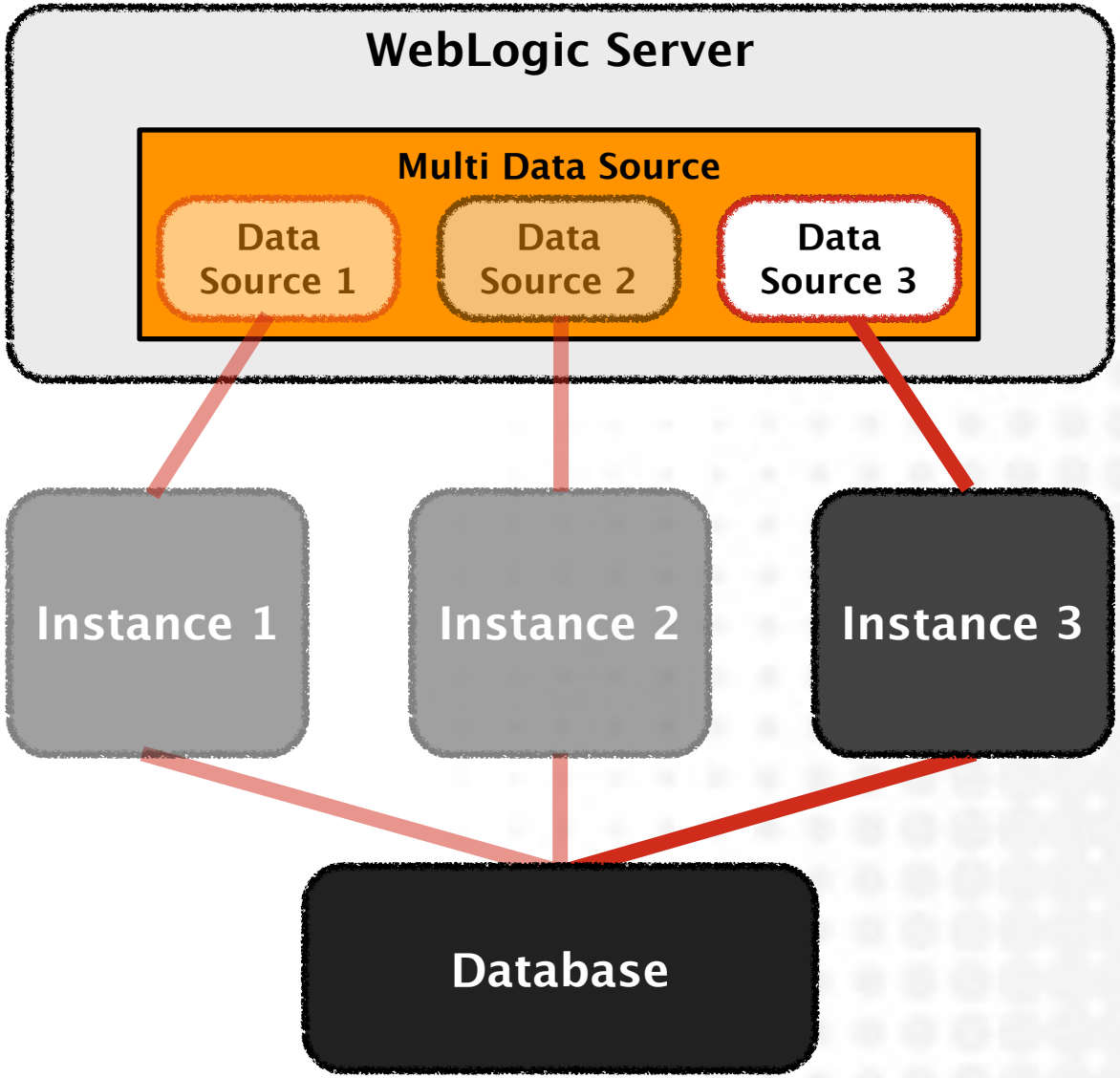
- Failover MDS
  - DS health checks
    - failure => DS dead
  - Can manually disable DS
  - Dead DS are re-tested
    - default 120 seconds

WebLogic MDS for Failover with Oracle RAC



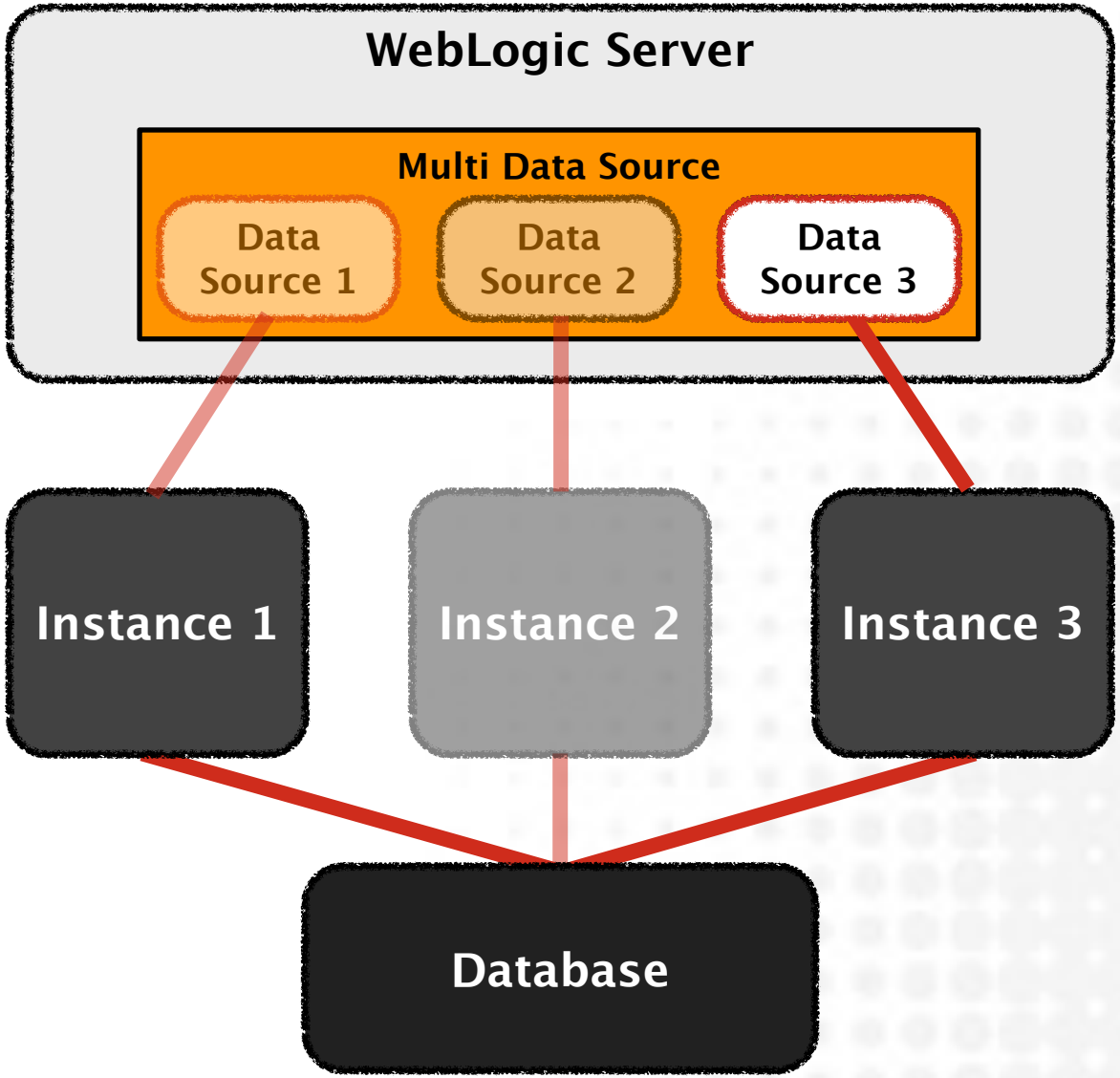
- Failover MDS
  - DS health checks
    - failure => DS dead
  - Can manually disable DS
  - Dead DS are re-tested
    - default 120 seconds

WebLogic MDS for Failover with Oracle RAC



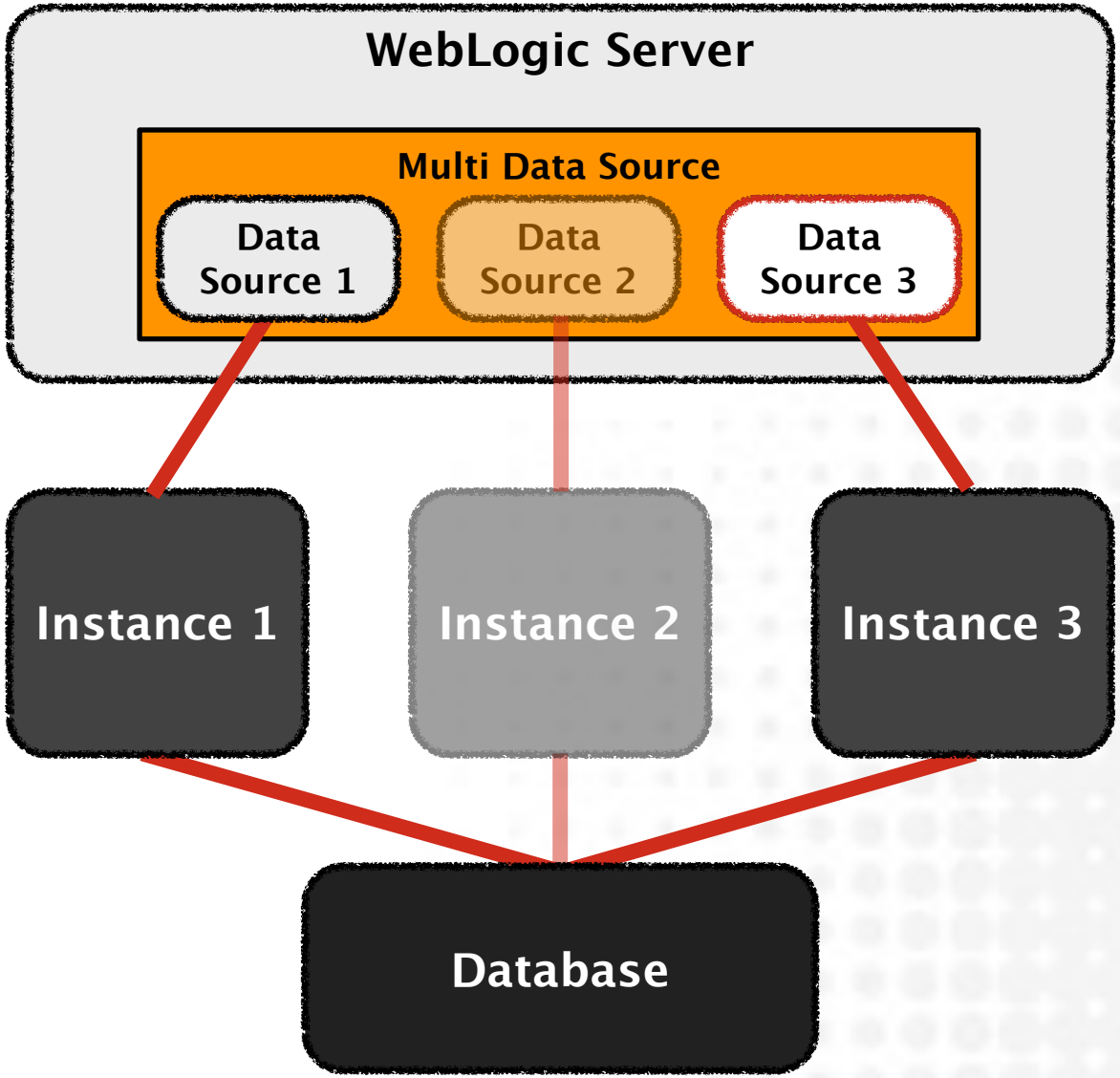
- Failover MDS
  - DS health checks
    - failure => DS dead
  - Can manually disable DS
  - Dead DS are re-tested
    - default 120 seconds

WebLogic MDS for Failover with Oracle RAC



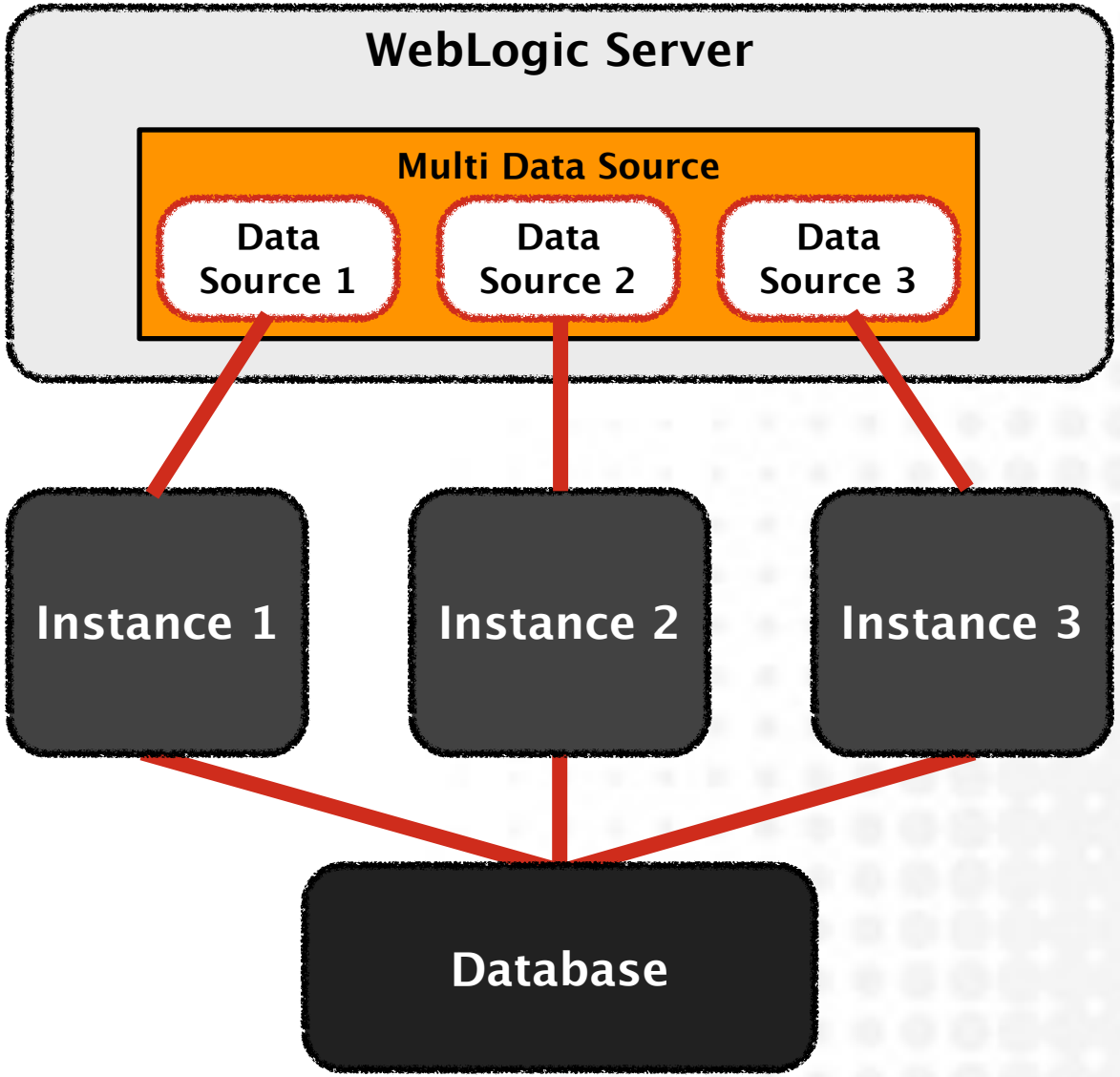
- Failover MDS
  - DS health checks
    - failure => DS dead
  - Can manually disable DS
  - Dead DS are re-tested
    - default 120 seconds

WebLogic MDS for Failover with Oracle RAC



- Failover MDS
  - DS health checks
    - failure => DS dead
  - Can manually disable DS
  - Dead DS are re-tested
    - default 120 seconds

WebLogic Multi Data Source with Oracle RAC

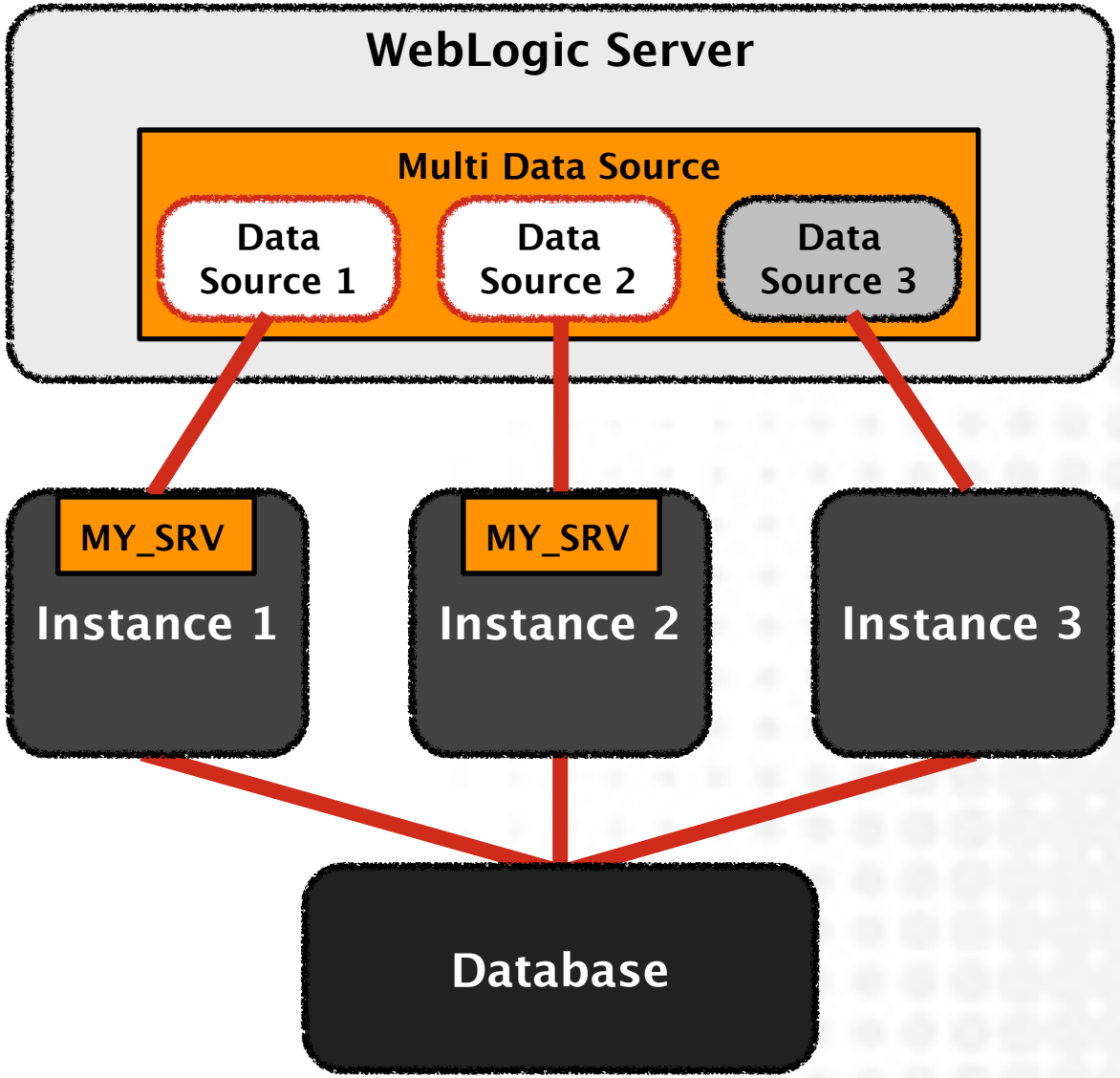


- Load Balancing
- MDS routes connection to all active DS
  - round-robin
  - failover is supported
  - no performance base load balancing like Oracle RLB
  - DS size doesn't affect LB

Oracle RAC: JDBC Pool vs WebLogic Multi DS

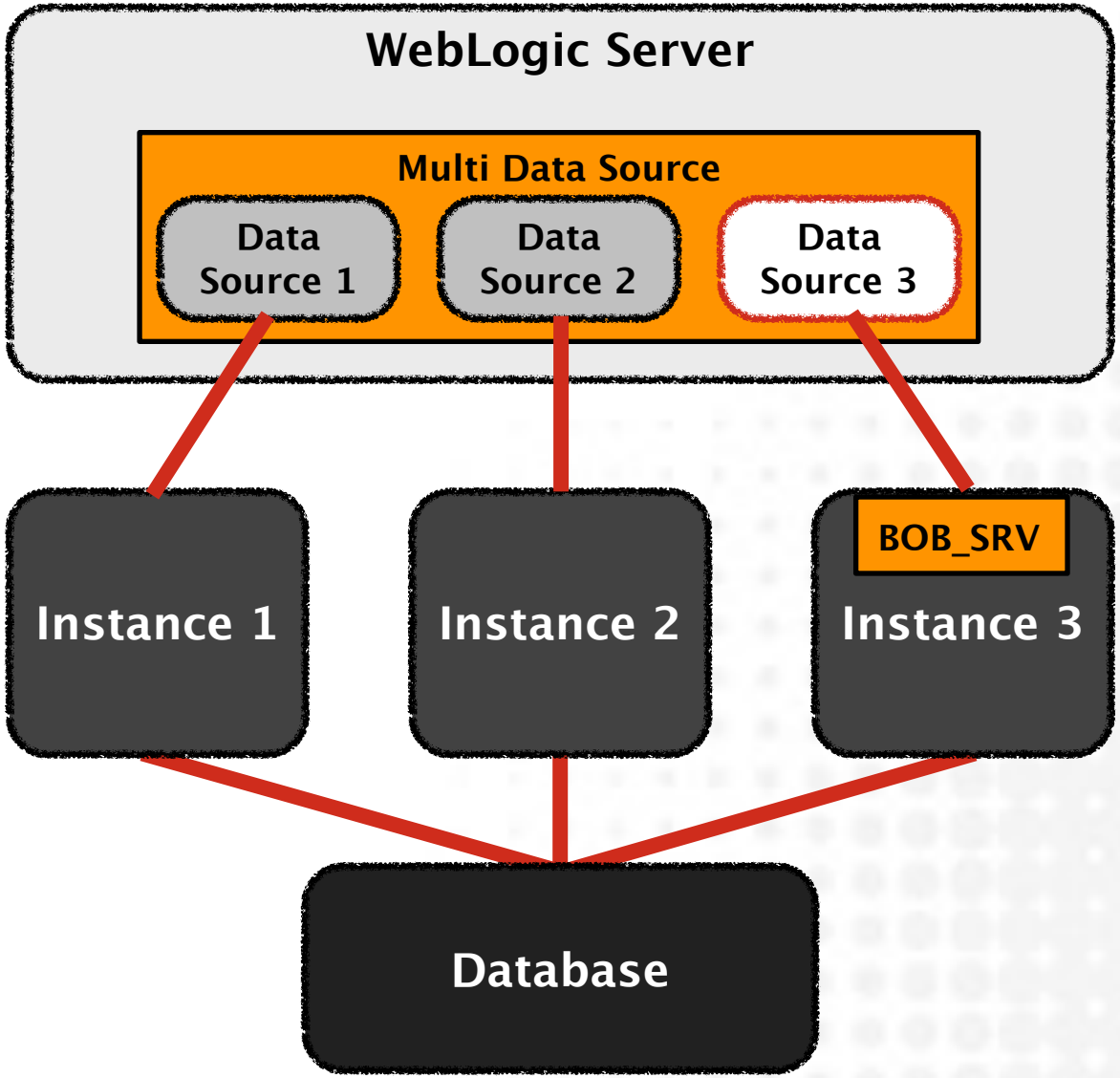
	Oracle JDBC	WLS Multi-DS
Connect failover	connect time failover	MDS failover
Connection load balancing	JDBC CLB	none
Server-side CLB based on service metrics / performance	Listener	none
Control connections layout	RAC Services	MDS config / members
Reconnect on failure	FCF	MDS managed
Run-time load balancing based on service performance	RLB / LBA	MDS LB round-robin
Use new (or recovered) instance	not until reconnect	automated by MDS

WebLogic Connections to RAC Services

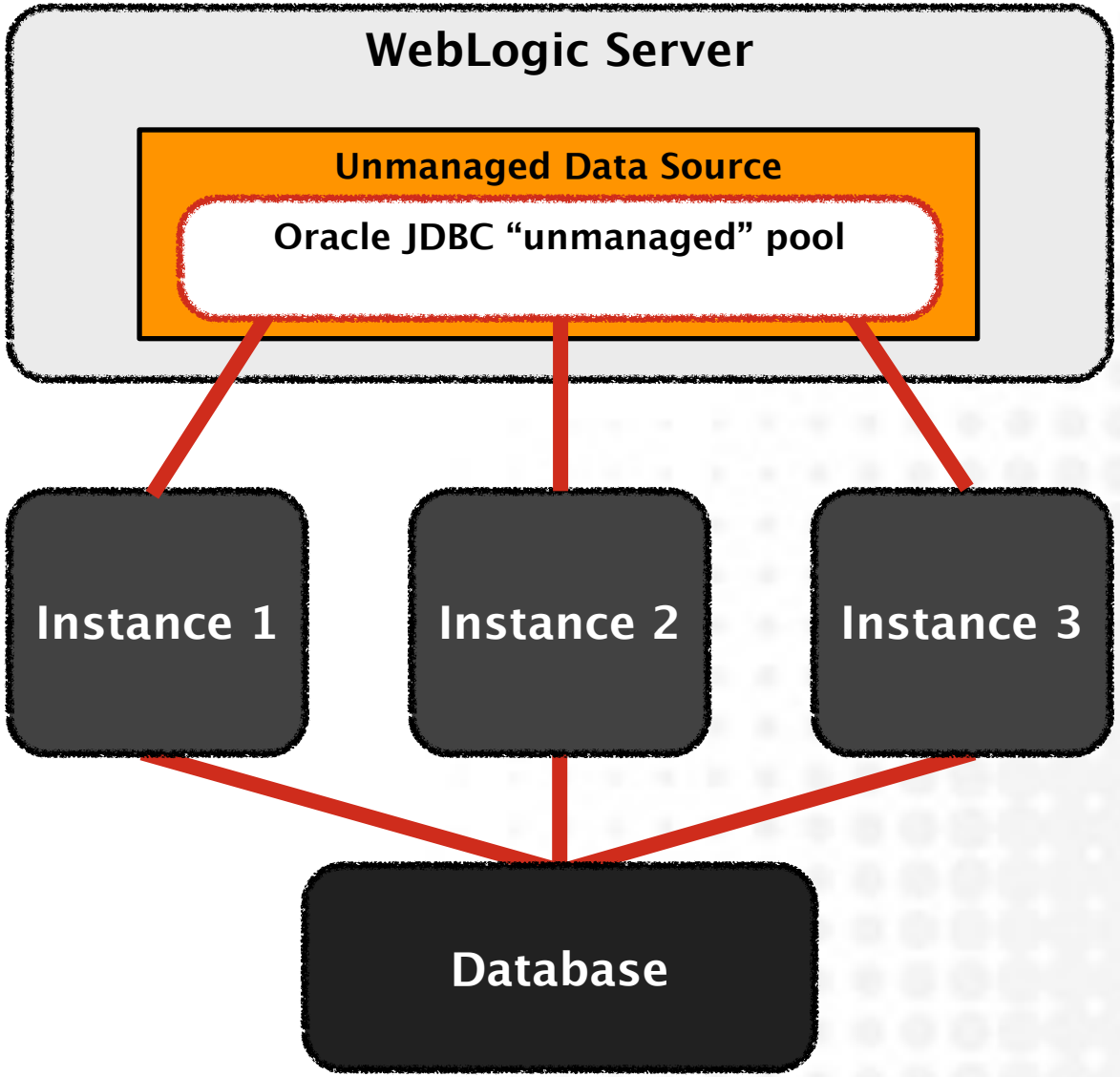


- Initial Capacity = 0
- DS must exist for every node that can run the service
- Services layout change requires WLS reconfiguration
- XA supported
  - branches of the same XA transactions are routed to the same instance
- MDS for load balancing
  - Failover makes little sense

WebLogic Connections to RAC Services



WebLogic Unmanaged DS with Oracle RAC



- JDBC native CLB
- DB Server - CLB_GOAL_LONG
- Failover - FCF
  - Based on HA events
- RLB does not work
- Connection testing not required

What's next?

- Book draw
  - Leave me your business cards

Q & A

Email me - gorbachev@pythian.com

Read my blog - http://www.pythian.com

Follow me on Twitter - @AlexGorbachev

Join Pythian fan club on Facebook & LinkedIn