Oracle Client Failover – Under The Hood

ITOUG Tech Day 2017
#ITOUGTD17

Robert Bialek
Principal Consultant

Ludovico Caldara
Oracle ACE Director, Senior Consultant

About Robert Bialek

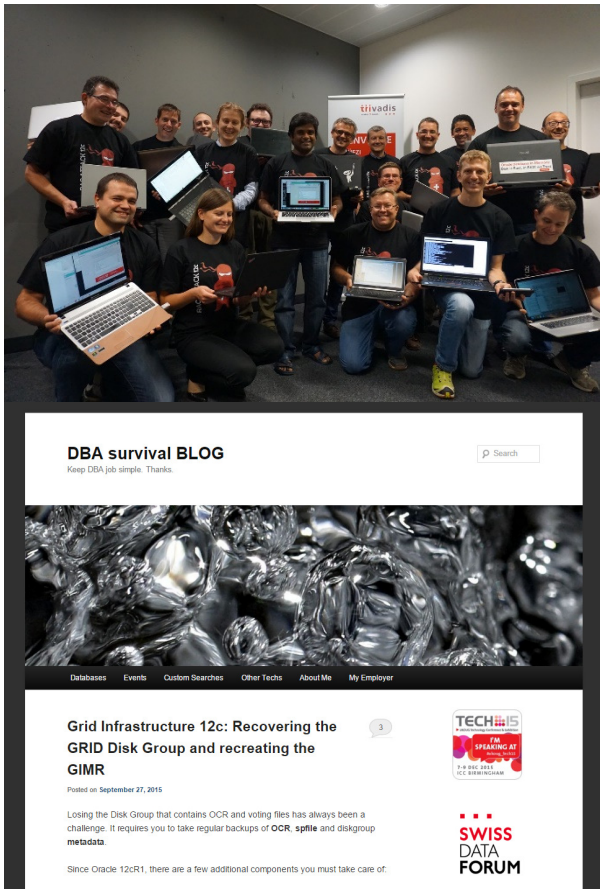
- Principal Consultant and Trainer at Trivadis GmbH in Munich
  - MSc in Computer Engineering
- Focus:
  - Oracle Database High Availability
  - Database Architecture/Internals
  - Backup/Recovery
  - Troubleshooting/Performance Tuning
  - Linux Administration
- Trainer for the following Trivadis courses
  - Oracle Grid Infrastructure, RAC, Data Guard

ORACLE
Certified Master

RED HAT CERTIFIED ENGINEER
redhat.

trivadis
makes IT easier.

About Ludovico Caldara



- 17 Years DBA (Not Only Oracle)
  - I do it everywhere (even Windows)
- RAC ATTACK Ninja & co-writer
- RACSIG President, SOUG & ITOUG Board
- OCP (11g, 12c, MySQL) & OCE
- Italian living in Switzerland

http://www.ludovicocaldara.net

@ludodba ludodba

ludovicocaldara

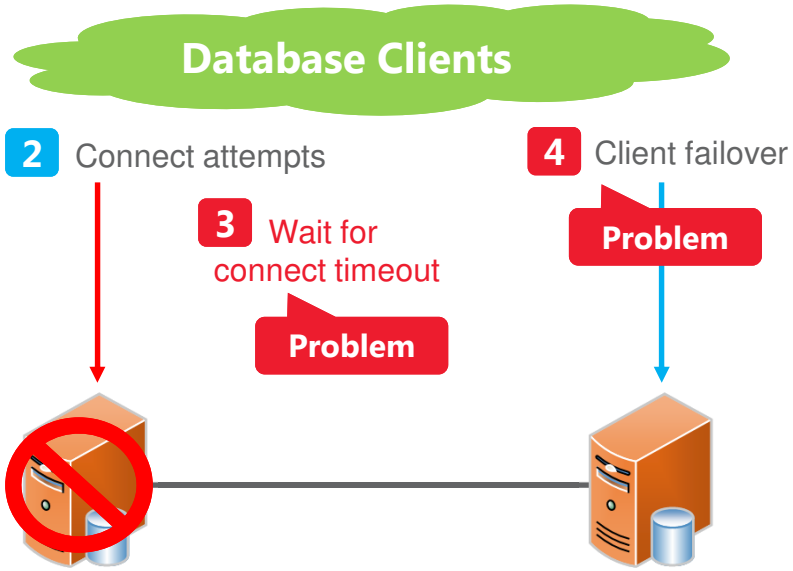
ORACLE ACE Director

trivadis makes IT easier.

Oracle Client Failover – Main Problems To Address

CASE 1

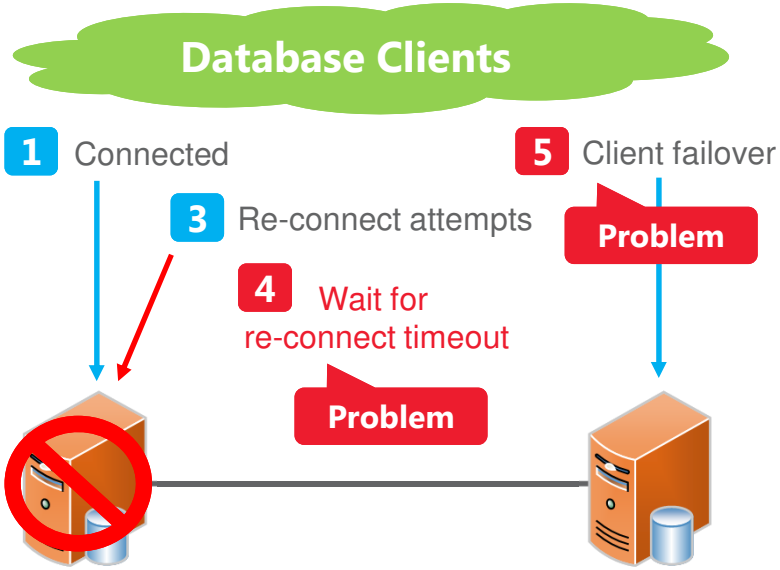
- **New** network session (connect)



1 IP not reachable (server/network/… issue)

CASE 2

- Already **established** network session



2 IP not reachable (server/network/… issue)

trivadis makes IT easier.

Oracle Client Failover – And The Solution?

- Depends strongly on many factors
  - Oracle client and database version.
  - Oracle database configuration, edition and available licenses.
  - Oracle client libraries/version (OCI, JDBC Thin,…).
  - Application design.
  - Network topology, latencies.
  - Operating system type, version and configuration.
  - With or without Virtual IP Addresses (VIP).

- **Unfortunately no *one size fits all* solution…**

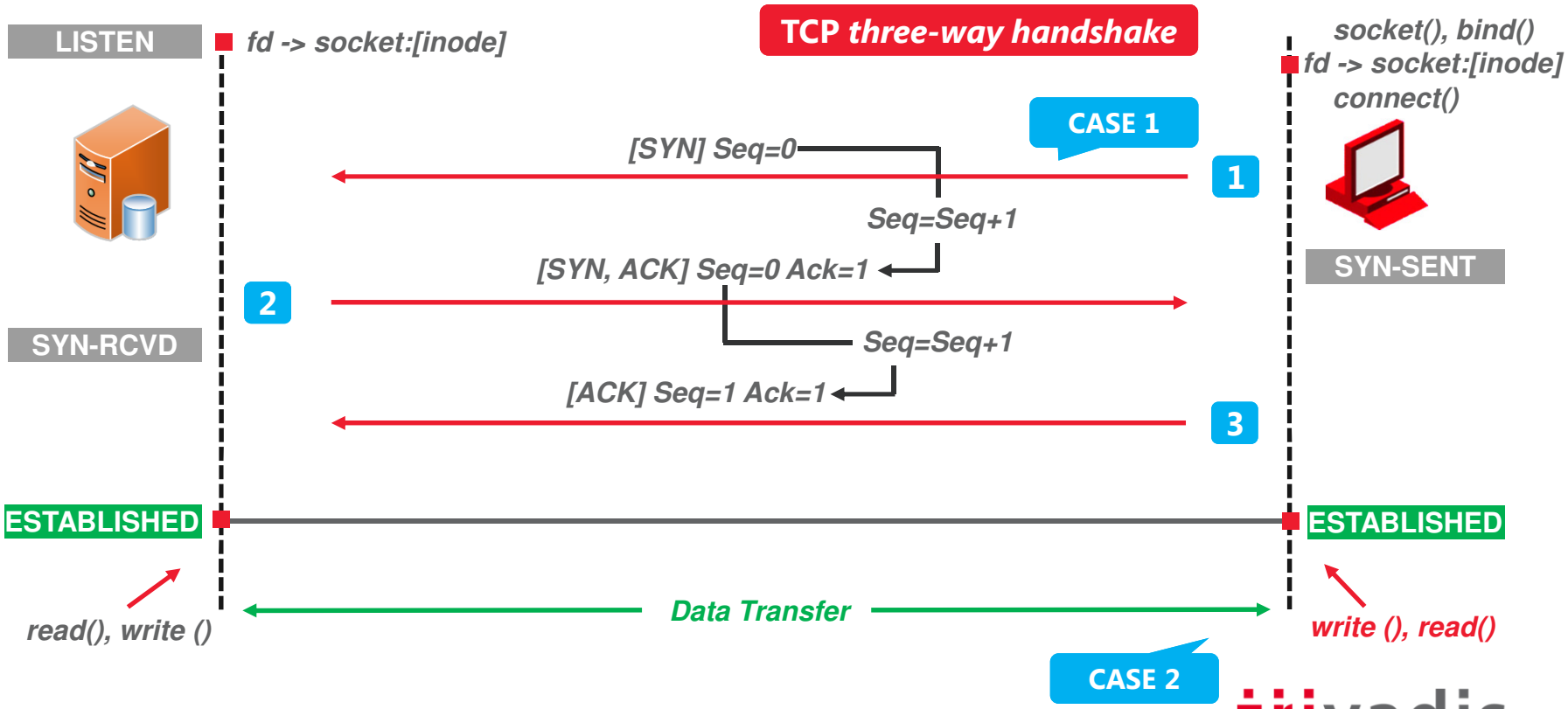
Agenda

1. **Operating System**
   - Introduction
   - Connect Timeouts/ARP Cache
   - Re-Connect Timeouts
   - Virtual IP Addresses
   - TCP Keepalive (DCD)
2. **Oracle Client Failover**
   - Database Services
   - Connect Timeouts
   - Re-Connect Timeouts
   - Transparent Application Failover
   - Fast Application Notification/Fast Connection Failover
   - Application Continuity
3. **Conclusions**

Operating System

Introduction

Operating System – Introduction



trivadis makes IT easier.

Operating System

Connect Timeouts/ARP Cache

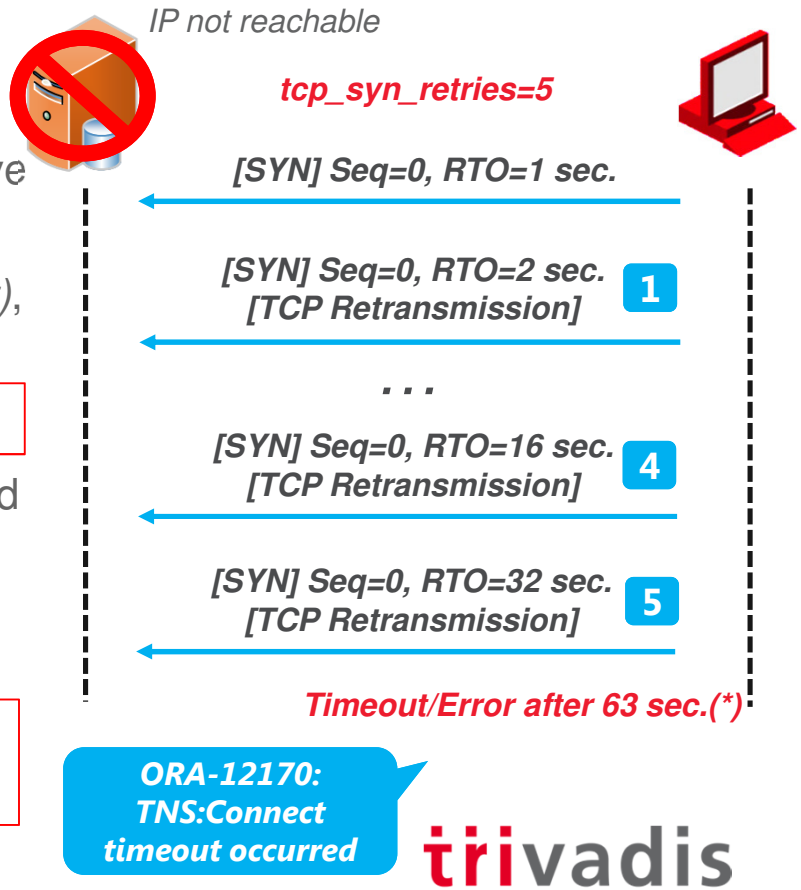
New Network Session – Connect Timeout

- Kernel parameter: *tcp_syn_retries*
  - Max. number of times initial SYNs for an active TCP connection attempt will be retransmitted.
  - Default value in OEL 5/7 is *5 (63 sec. timeout)*, in OEL 6 it is *6 (127 sec. timeout).*

```
Final Timeout = 2^(tcp_syn_retries+1)-1
```

  - Initial RTO is 1 sec., after each retry increased by 2 (*<RTO>=<RTO>*2*).
  - To change the value (not persistent).

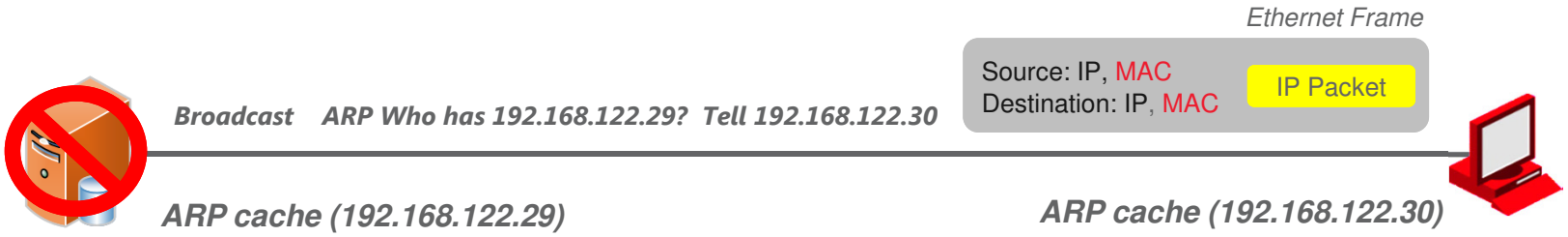
```
#Connect timeout after 15 sec.
sysctl -w net.ipv4.tcp_syn_retries=3
```



trivadis makes IT easier.

New Network Session – Connect Timeout/ARP

- Depending on the ARP cache entry status, the following two different scenarios are possible during client connection to an unresponsive server



	`IP:192.168.122.30 MAC:...:60:d4:0d REACHABLE`	`IP:192.168.122.29 MAC:...:1d:54:ec REACHABLE`
CASE 1	*Not refreshed yet*! Client connect timeout (*tcp_syn_retries*)	`IP:192.168.122.29 MAC:...:1d:54:ec REACHABLE`
CASE 2	*Refreshed*! Client connect timeout *~3sec* (the same network segment) ORA-12543: TNS: destination host unreachable	**ARP entry removed**

trivadis makes IT easier.

Operating System

Re-Connect Timeouts

Established Network Session –Re-Connect Timeout

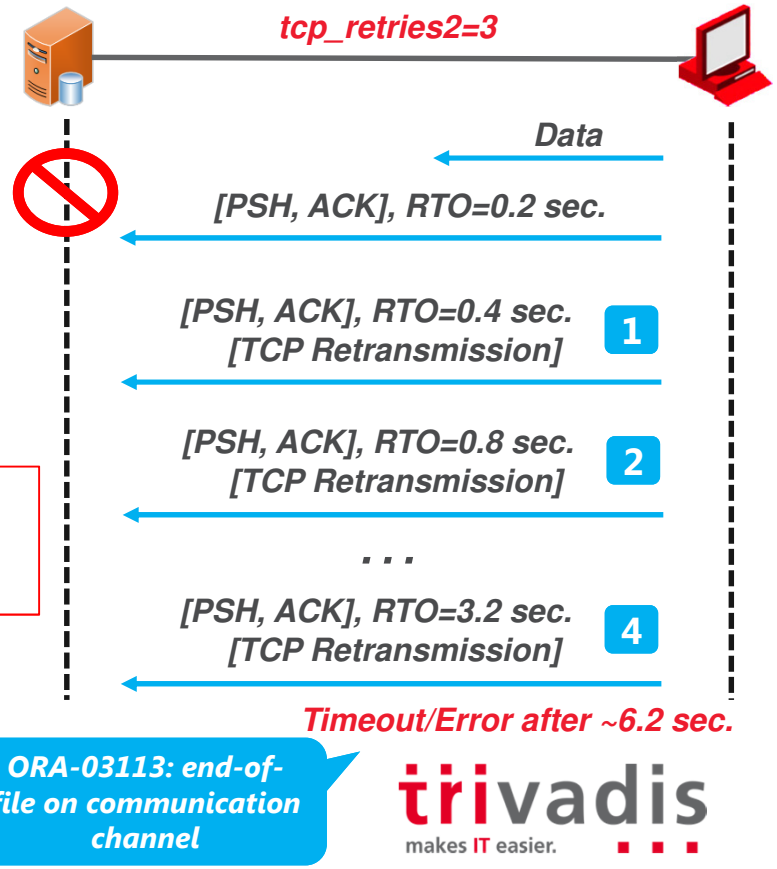
- Kernel parameter: *tcp_retries2*
  - Max. number of TCP packet retransmissions for established sessions plus 1.
  - Default value: *15*, Timeout range: *~15-30* min.
  - Initial RTO set 0.2 sec, increased by 2x during each re-try; the maximum value is 120 sec.
  - But, runtime *RTO* can be changed by kernel.

```
ss -ipo dst 192.168.122.29

  socket timer:(on,1min44sec,11)
  socket timer:(on,49sec,11)      #1 sec. later
```

  - To change the value (not persistent).

```
#Re-connect timeout after ~12 sec.
sysctl -w net.ipv4.tcp_retries2=4
```



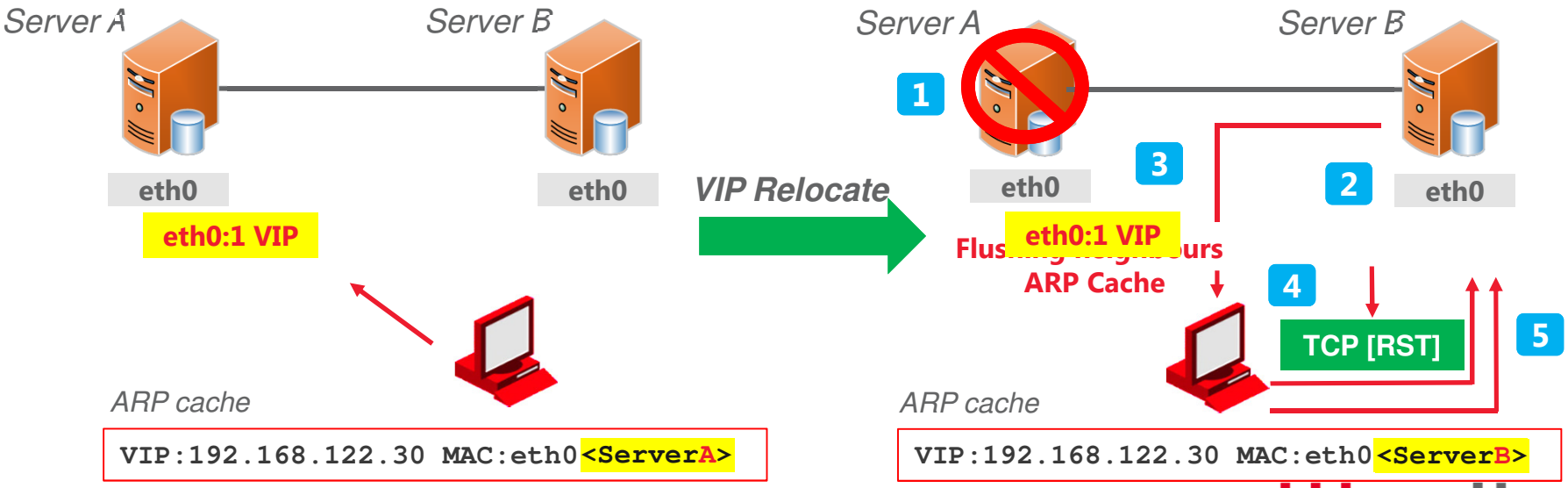
trivadis makes IT easier.

Operating System

Virtual IP Addresses

Virtual IP Addresses (VIP)

- IP addresses managed by a cluster which do not correspond persistently to physical NICs.
- Client connects to network socket: *<VIP>:<Port>.*

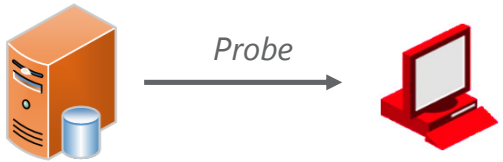


trivadis makes IT easier.

Operating System

TCP Keepalive (DCD)

Network – TCP Keepalive (DCD): Server



- TCP mechanism which helps to detect *broken* network connections.
- Kernel parameters.

Network socket closed after 7875 sec.

```
net.ipv4.tcp_keepalive_time = 7200   #keepalive probe every 2 hrs.
net.ipv4.tcp_keepalive_intvl = 75    #if not reachable probe every 75 sec.
net.ipv4.tcp_keepalive_probes =9     #close the connection after 9 failed probes
# 7200 seconds of the keepalive timer + 9 times 75 seconds = 7875 sec.
```

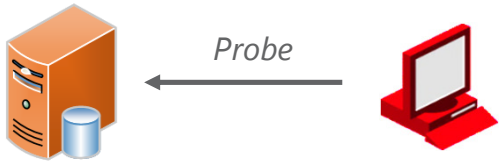
- For Oracle server (shadow) processes, automatically enabled on the network socket
  - Implementation changed in 12c (tcp socket timer instead of *Oracle Net* probes).

sqlnet.ora	OS Settings
SQLNET.EXPIRE_TIME=1	tcp_keepalive_time = 60 tcp_keepalive_intvl = 10 tcp_keepalive_probes = 6

Translates to 2 min. timeout

trivadis
makes IT easier.

Network – TCP Keepalive (DCD): Client



- For Oracle client processes not activated per default.

```
Local Address        Foreign Address      State       PID/Program name Timer
192.168.122.2:38814 192.168.122.3:15300 ESTABLISHED 5963/sqlplus     off(0.00/0/0)
```

  – Unless *ENABLE=BROKEN* specified in the connect descriptor.

```
NONCDB.TRIVADIS.COM =
  (DESCRIPTION =
     (ENABLE=BROKEN)
     (ADDRESS_LIST = ...
```

- The default client timeout is 7875 sec.!
  – Cannot be influenced by SQLNET.EXPIRE_TIME.
- To reduce the timeout, you need to reduce OS kernel parameters.

```
echo 60 > /proc/sys/net/ipv4/tcp_keepalive_time
echo 10 > /proc/sys/net/ipv4/tcp_keepalive_probes
echo 6 > /proc/sys/net/ipv4/tcp_keepalive_intvl
```

Translates to 2 min. timeout

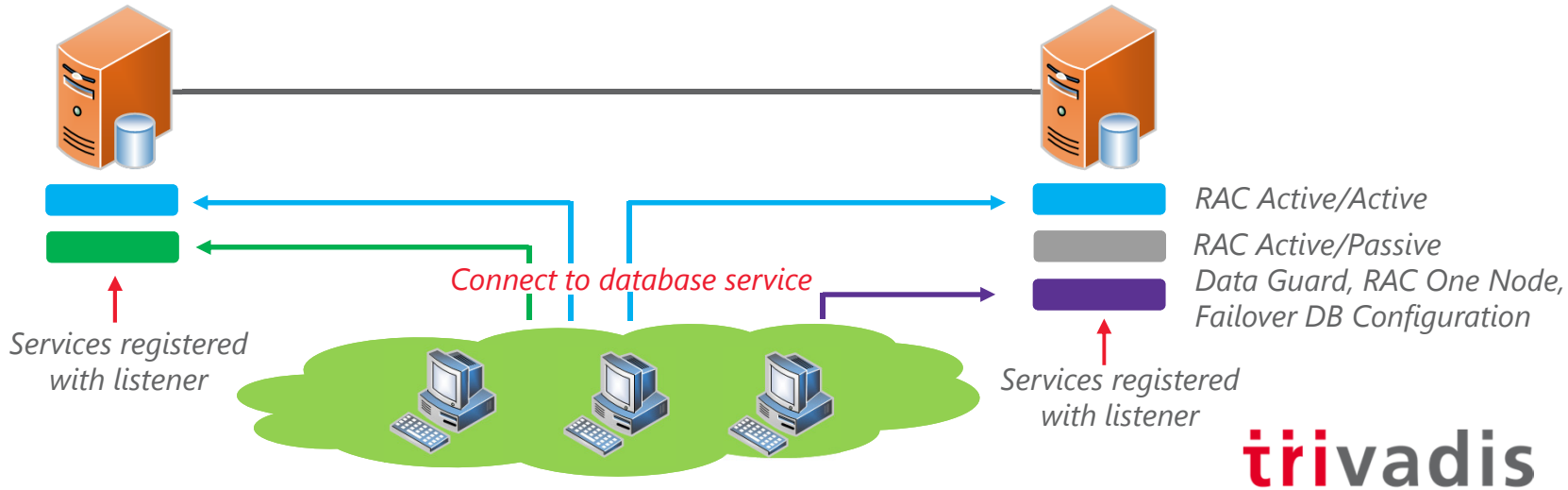
Oracle Client Failover

Database Services

trivadis
makes IT easier.

The Foundation – Database Services

- A named representation of one or more running Oracle database instances
  - Introduced with the Oracle 8i version.
  - Part of the Oracle client connect descriptor.
  - Basis of Oracle database high availability and workload management.



trivadis
makes IT easier.

Database Services (1)

- Database services can be created with:
  - *srvctl* (*Grid Infrastructure*), gdsctl (Global Data Services).
  - *dbms_service.create_service*() PL/SQL procedure.
- Different high availability and workload management attributes can be defined

```
srvctl add service
  -db          <db_unique_name>
  -service     <service>
  -preferred   "<preferred_list>"
  -available   "<available_list>"
  -serverpool  <pool_name>
  -cardinality [UNIFORM | SINGLETON]
  -tafpolicy   [NONE | BASIC | PRECONNECT]
  -role        [PRIMARY, PHYSICAL_STANDBY, LOGICAL_STANDBY, SNAPSHOT_STANDBY]
  -clbgoal     [SHORT | LONG]
  -rlbgoal     [SERVICE_TIME | THROUGHPUT | NONE]
  ...
```

Some attributes applicable only for specific configurations

Not available with Oracle Restart

Database Services (2)

- Example service creation with PL/SQL

```
BEGIN
  DBMS_SERVICE.CREATE_SERVICE (
     service_name     => 'OLTP.TRIVADIS.COM',
     network_name     => 'OLTP.TRIVADIS.COM',
     failover_method  => 'BASIC',
     failover_type    => 'SELECT',
     failover_retries => 180,
     failover_delay   => 3);
END;
/
```

- Database service created with the above method needs to be started after opening a database

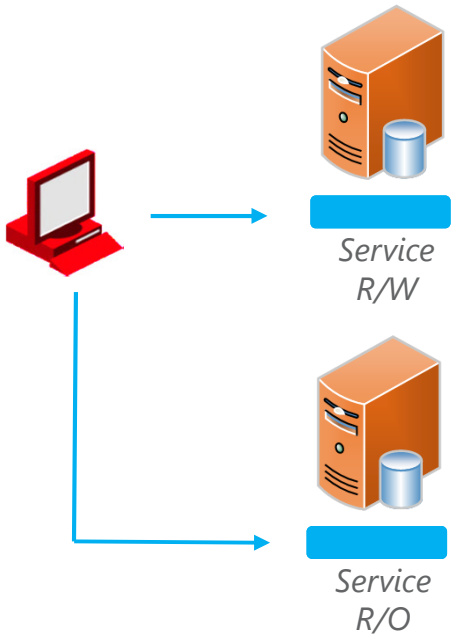
```
EXECUTE DBMS_SERVICE.START_SERVICE('OLTP.TRIVADIS.COM')
```

- Create your own TRIGGER firing AFTER STARTUP ON DATABASE

Role-Based Services (1)

- For a Data Guard system, we need a role-based service, that is running only if database has a specific role
  - Read-write service on a primary database.
  - Optionally, a service on standby databases for reporting.
  - Optionally, a service on snapshot standby databases.

- To accomplish this:
  - Use Oracle Grid Infrastructure role-based services.
  - Create your own AFTER STARTUP ON DATABASE trigger.



trivadis
makes IT easier.

Role-Based Services (2)

- Example role-based services with Grid Infrastructure.

```
srvctl add service -db DG_SITE1 -service OLTP_RW.trivadis.com \
       -role PRIMARY
srvctl add service -db DG_SITE1 -service OLTP_RO.trivadis.com \
       -role PHYSICAL_STANDBY
srvctl add service -db DG_SITE1 -service OLTP_SNAP.trivadis.com \
       -role SNAPSHOT_STANDBY
```

- Services are started, only if database and service role match.

```
SvcAgent::start 680 query_db_role
SvcAgent::start 710 not starting service oltp Role mismatch - Service
role:PRIMARY, current DB role:PHYSICAL_STANDBY
```

Role-Based Services (3)

- Example role-based services without Grid Infrastructure.

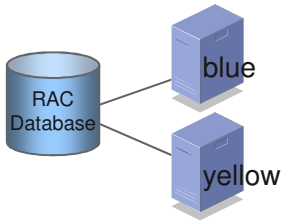
```
CREATE OR REPLACE TRIGGER service_trigger
AFTER STARTUP ON DATABASE
DECLARE
  v_service_ro   VARCHAR2(64) := rtrim(sys_context('userenv','db_name')||'_RO.'
                                 || sys_context('userenv','db_domain'), '.');
  v_service_rw   VARCHAR2(64) := rtrim(sys_context('userenv','db_name')||'_RW.'
                                 || sys_context('userenv','db_domain'), '.');
  v_service_snap VARCHAR2(64) := rtrim(sys_context('userenv','db_name')||'_SNAP.'
                                 || sys_context('userenv','db_domain'), '.');
  v_ro_service_count  NUMBER;
BEGIN
  SELECT count(*) INTO v_ro_service_count FROM v$active_services WHERE name = v_service_ro;
  IF sys_context('userenv','database_role') IN ('PRIMARY','SNAPSHOT STANDBY')
     AND v_ro_service_count = 1 THEN
    dbms_service.stop_service(v_service_ro);
    dbms_service.disconnect_session(v_service_ro,dbms_service.immediate);
  END IF;
  IF sys_context('userenv','database_role') = 'PRIMARY' THEN
    dbms_service.start_service(v_service_rw);
  ELSIF sys_context('userenv','database_role') = 'SNAPSHOT STANDBY' THEN
    dbms_service.start_service(v_service_snap);
  ELSE
    IF v_ro_service_count = 0 THEN
      dbms_service.start_service(v_service_ro);
    END IF;
  END IF;
END;
/
```

Database Services – Application Client Configuration (1)

- Configuration for RAC (without DG): use Single Client Access Name (SCAN).

```
OLTP.trivadistraining.com =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP )(HOST = sweden)(PORT = 1521 ))
    (CONNECT_DATA =
      (SERVICE_NAME = OLTP.trivadistraining.com )
    )
  )
```

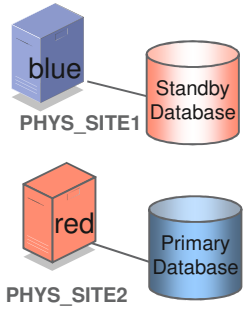
3 Virtual IP Addresses



- Configuration for Data Guard (without RAC).

```
PHYS.trivadistraining.com =
  (DESCRIPTION =
    (FAILOVER = ON)(LOAD_BALANCE = OFF)
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = blue)(PORT = 1521))
      (ADDRESS = (PROTOCOL = TCP)(HOST = red)(PORT = 1521)))
    (CONNECT_DATA = (SERVICE_NAME = PHYS_RW.trivadistraining.com)
))
```

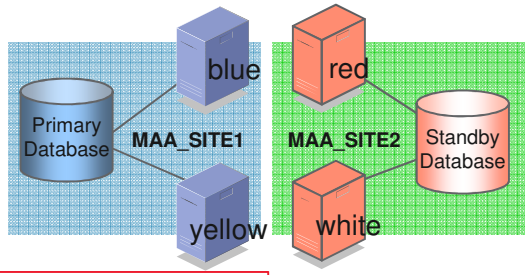
Potential issue with TCP Timeouts



trivadis
makes IT easier.

Database Services – Application Client Configuration (2)

- Configuration for Maximum Availability Architecture
  - RAC with Data Guard.




```
MAA.trivadistraining.com =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (LOADBALANCE = OFF )
      (ADDRESS = (PROTOCOL = TCP )(HOST = sweden )(PORT = 1521 ))
      (ADDRESS = (PROTOCOL = TCP )(HOST = italy )(PORT = 1521 ))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = MAA_RW.trivadistraining.com )
    )
  )
```

3 Virtual IP Addresses

Potential issue with TCP Timeouts

trivadis makes IT easier.

Oracle Client Failover

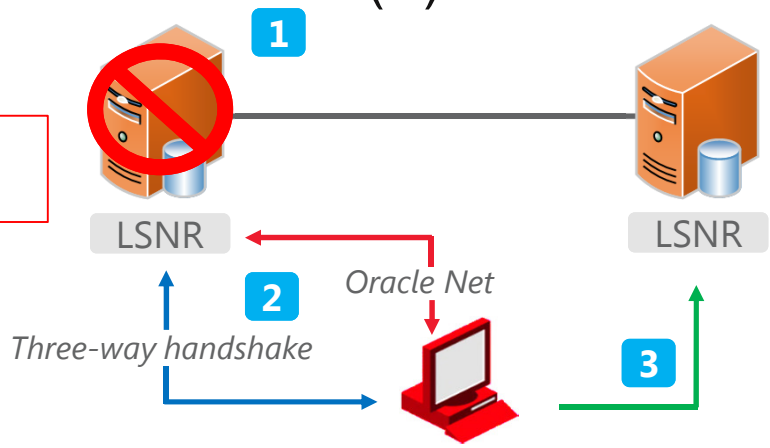
Connect Timeouts

New Oracle Net Session – Connect Timeout (1)

- sqlnet.ora parameters (*OCI*, *ODP.net*)

```
TCP.CONNECT_TIMEOUT=3                #default 60 sec.
SQLNET.OUTBOUND_CONNECT_TIMEOUT=5 #no default
```

With SCAN, a client might wait 3 x <timeout_value>



- Address description parameters (>=11gR2)
  - Override sqlnet.ora parameters

Introduced in 12.1.0.2

```
OLTP.trivadis.com =
  (DESCRIPTION =
    (FAILOVER=ON)  (LOAD_BALANCE=OFF)
    (CONNECT_TIMEOUT=5)(RETRY_COUNT=3)(RETRY_DELAY=1)(TRANSPORT_CONNECT_TIMEOUT=3)
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP )(HOST = italy )(PORT = 1521))
      (ADDRESS = (PROTOCOL = TCP )(HOST = sweden )(PORT = 1521)))
    (CONNECT_DATA = (SERVICE_NAME = OLTP.trivadis.com)))
```

  - Parameters can be used for OCI, ODP.net

trivadis makes IT easier.

New Oracle Net Session – Connect Timeout

- *JDBC Thin* driver
  - *TRANSPORT_CONNECT_TIMEOUT* is available beginning with 12.2 version
  - To use *RETRY_COUNT* with 12.1.0.2, patch is required (*BUG 19154304)*

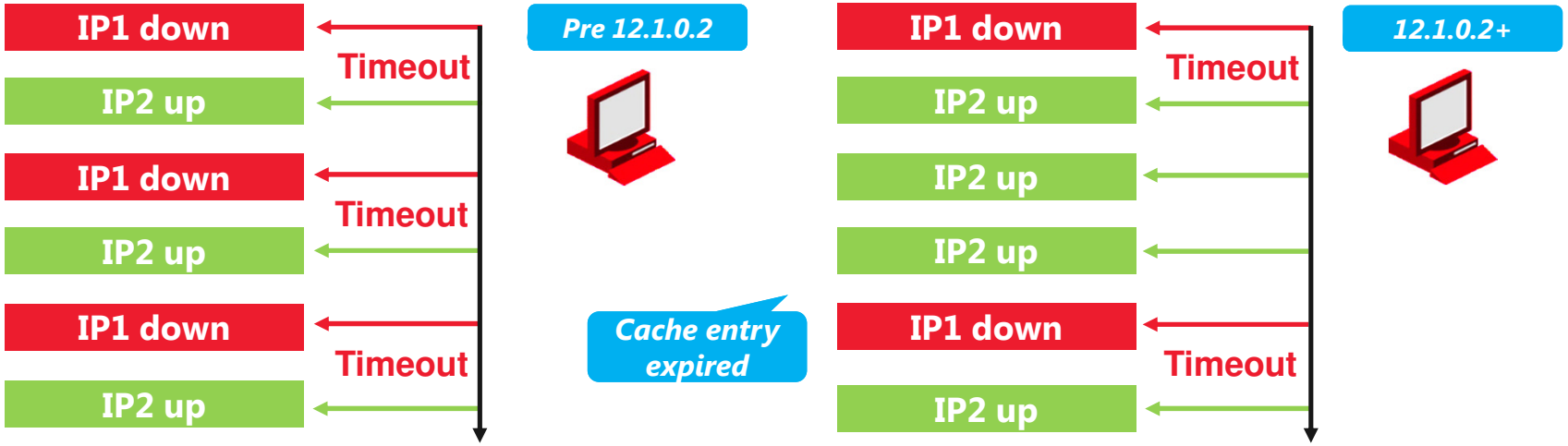
```
pds.setURL("jdbc:oracle:thin:@(DESCRIPTION =(FAILOVER=ON)(LOAD_BALANCE=OFF)" +
  "(CONNECT_TIMEOUT=3)(RETRY_COUNT=10)(RETRY_DELAY=1)" +
  "(ADDRESS_LIST = " +
  "(ADDRESS = (PROTOCOL = TCP )(HOST = blue.trivadis.com )(PORT = 1521)) " +
  "(ADDRESS = (PROTOCOL = TCP )(HOST = brown.trivadis.com )(PORT = 1521))) " +
  "(CONNECT_DATA = (SERVICE_NAME = sales_rw.trivadis.com)))");
```

- *JDBC Thin* clients can alternatively use the following driver property (*ms*)
  - Overrides *CONNECT_TIMEOUT* from address description parameters

```
Properties prop = new Properties();
prop.put(oracle.net.ns.SQLnetDef.TCP_CONNTIMEOUT_STR, ""+3000);
ods.setConnectionProperties(prop);
```

New Oracle Net Session – Connect Timeout

- De-prioritization of down database nodes (activated by default – 600 sec.)

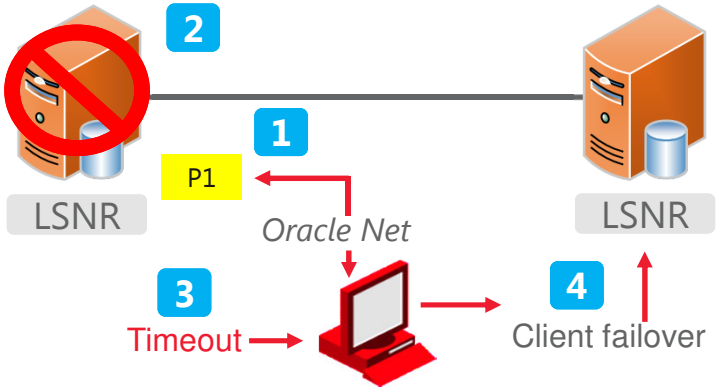


- Down state of a server is kept in client process cache
  - For *OCI* driver introduced with 12.1.0.2: *SQLNET.DOWN_HOSTS_TIMEOUT*
  - For *JDBC Thin* driver with 12.2.0.1: *oracle.net.DOWN_HOSTS_TIMEOUT*

Oracle Client Failover

Re-Connect Timeouts

Established Oracle Net Session – Re-Connect Timeout



- Break established network connection without waiting for long TCP timeouts (>15 min.)
- sqlnet.ora parameters (OCI & ODP.net)

```
SQLNET.RECV_TIMEOUT=30   #no default value
SQLNET.SEND_TIMEOUT=30   #no default value
```

  – The actual wait time is 2 x timeout value (wait for timeout -> switch into break and reset mode -> wait for timeout)!

- For *JDBC Thin* clients you can set the following connection property

```
Properties prop = new Properties();
prop.put ("oracle.jdbc.ReadTimeout", "30000"); //30 sec.
ods.setConnectionProperties(prop);
```


trivadis makes IT easier.

Established Oracle Net Session – Re-Connect Timeout

- **Important**: be very careful with re-connect timeouts!
  - You might encounter unwanted side effects, like dropping still valid Oracle Net connections!
  - Deploy them **only** if strictly necessary after **careful** testing! Better, **don't use them**!
  - Tuning OS kernel parameter *tcp_retries2* might be a better choice!
  - Tuning re-connect timeouts is not necessary, in case you use Fast Connection Failover (FCF) with Fast Application Notification (FAN).

Oracle Client Failover

Transparent Application Failover

Transparent Application Failover – Overview

- TAF is a feature of the client *OCI* driver introduced in Oracle 8
  - Masks many failures from the end users.
  - Allows for automatic re-connection.
  - In many cases allows for resumable queries.
  - Useful for session migration between RAC instances during some planned downtimes.
- Failover process can only be initiated, after receiving an error for the established connection.

Waiting too long for an error slows down the failover process!

```
(ADDRESS_LIST =
  (ADDRESS = (PROTOCOL = TCP )(HOST = italy )(PORT = 1521))
  (ADDRESS = (PROTOCOL = TCP )(HOST = sweden )(PORT = 1521)))
```

- TAF properties can be set on the client or server side (recommended, higher priority).

TAF – Parameters (Server/Client Side)

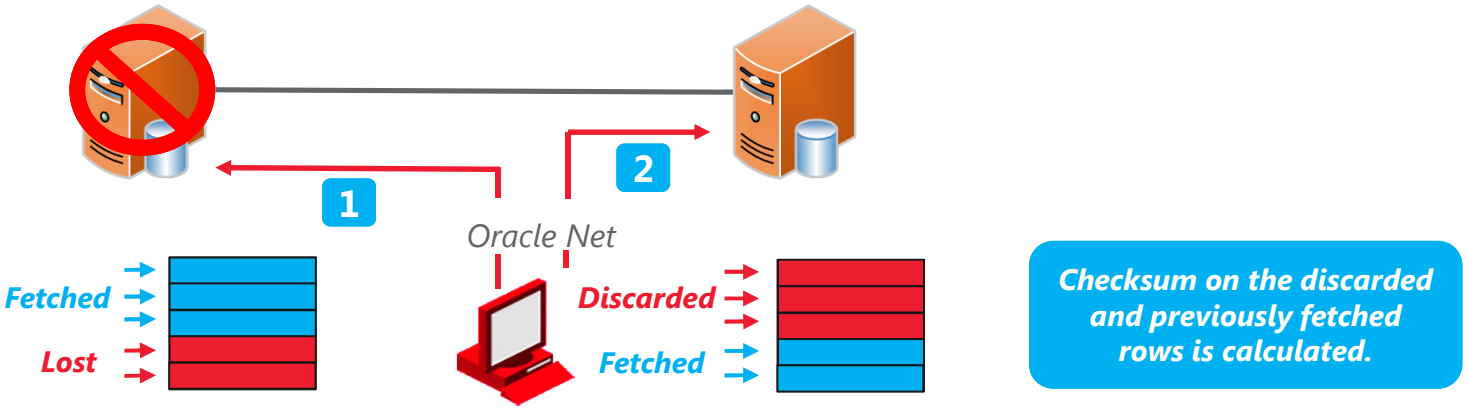
- Comparison of the server and client side TAF parameters.

TAF Parameter	Grid Infrastructure srvctl Parameters	PL/SQL DBMS_SERVICE.CREATE_SERVICE	Client tnsnames.ora/LDAP
Policy/Method	**-tafpolicy** [NONE | BASIC | PRECONNECT]	**FAILOVER_METHOD** [FAILOVER_METHOD_NONE | FAILOVER_METHOD_BASIC]	**METHOD** [BASIC | PRECONNECT]
Type	**-failovertype** [NONE | SESSION | SELECT | TRANSACTION]	**FAILOVER_TYPE** [FAILOVER_TYPE_NONE | FAILOVER_TYPE_SESSION | FAILOVER_TYPE_SELECT]	**TYPE** [SESSION | SELECT | NONE]
Backup Service (Preconnect)			**BACKUP** <SERVICE_NAME>
Failover Delay	**-failoverdelay**	**FAILOVER_DELAY**	**DELAY**
Failover Retry	**-failoverretry**	**FAILOVER_RETRIES**	**RETRIES**

trivadis makes IT easier.

Transparent Application Failover – Types

- Two types of TAF – *SESSION* or *SELECT*
  - With *SESSION*, client connection/session is re-created to a surviving database instance.
  - *SELECT* supports query re-executions that where in progress at the time of a failure.



trivadis makes IT easier.

Transparent Application Failover – Server Side Example

- Example server side TAF BASIC method configuration.

```
srvctl add service
  -db              OLTP_SITE1
  -service         OLTP_RW
  -preferred       OLTP1,OLTP2
  -tafpolicy       BASIC
  -failovertype    SELECT
  -failoverdelay   1
  -failoverretry   180
```

```
BEGIN
  DBMS_SERVICE.CREATE_SERVICE (
     service_name     => 'OLTP.TRIVADIS.COM',
     network_name     => 'OLTP.TRIVADIS.COM',
     failover_method  => 'BASIC',
     failover_type    => 'SELECT',
     failover_retries => 180,
     failover_delay   => 3);
END;
/
```

Transparent Application Failover – Client Side Example

- Example client side TAF BASIC method configuration.

```
OLTP.trivadis.com =
  (DESCRIPTION =
    (FAILOVER=ON) (LOAD_BALANCE=OFF)
    (CONNECT_TIMEOUT=5)(RETRY_COUNT=3)(RETRY_DELAY=1)(TRANSPORT_CONNECT_TIMEOUT=3)
    (ADDRESS=(PROTOCOL=TCP)(HOST=cl121.trivadis.com)(PORT=1521))
    (CONNECT_DATA =
       (SERVICE_NAME = OLTP.trivadis.com)
       (FAILOVER_MODE =
           (TYPE = SESSION)
           (METHOD = BASIC)
           (RETRIES = 180)
           (DELAY = 1)
        )
     )
   )
```

Single Client Access Name

Transparent Application Failover – Failover Behavior

- Example client failover behavior with TAF BASIC method
  - Query *GV$SESSION* before failover.

```
INST_ID USERNAME  SID FAILOVER_TYPE FAILOVER_METHOD FAILED_OVER
------- -------- ---- ------------- --------------- -----------
1            BIR   44        SELECT           BASIC          NO
```

  - Kill PMON process on the RAC instance 1 and query GV$SESSION after failover again.

```
INST_ID USERNAME  SID FAILOVER_TYPE FAILOVER_METHOD FAILED_OVER
------- -------- ---- ------------- --------------- -----------
2            BIR  277        SELECT           BASIC         YES
```

TAF – Session Migration

- TAF might be used, to transparently migrate client sessions between RAC nodes during planned downtimes.
- Method 1.

```
srvctl stop service -db <db_unique_name> -instance <instance> -service <service>
EXEC DBMS_SERVICE.DISCONNECT_SESSION('<service>', DBMS_SERVICE.POST_TRANSACTION)
```

- Method 2.

```
srvctl stop service -db <db_unique_name> -instance <instance> -service <service>
srvctl stop instance -db <db_unique_name> -service <service> \
                     -stopoption "TRANSACTIONAL LOCAL"
```

Resource ACTION_TIMEOUT=600

All ongoing transactions running in the local instance for more than 10 min. are aborted!

Oracle Client Failover

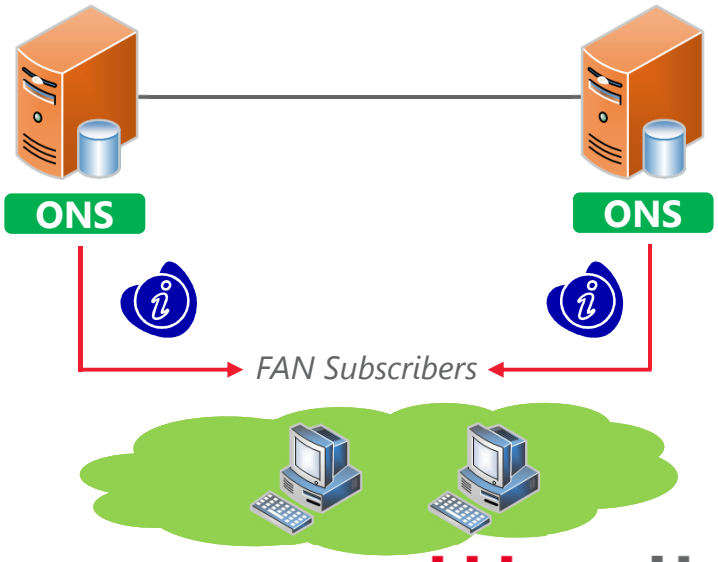
Fast Application Notification

Fast Connection Failover

Fast Application Notification – Overview

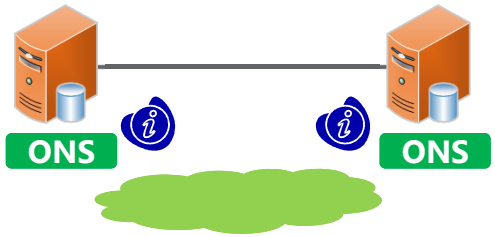
- Provides rapid notification about status changes (up/down events) for database services, instances and nodes.
- Delivers workload information about services (*runtime* load balancing).
- Starting with Oracle 12c ONS is used as the FAN transport for all client types
- FAN event consists of header and payload:

```
** Event Header **
Notification Type: database/event/service
Delivery Time: Mon Oct 10 21:56:43 CEST 2016
Generating Node: cldb01.trivadis.com
Event payload: VERSION=1.0 event_type=SERVICEMEMBER
service=soe_app1.TRIVADIS.COM instance=RAC2
database=rac_site1 db_domain=TRIVADIS.COM
host=cldb02 status=down reason=FAILURE
timestamp=2016-10-10 21:56:43 timezone=+02:0
```



trivadis makes IT easier.

Requirements For Using FAN Events (1)

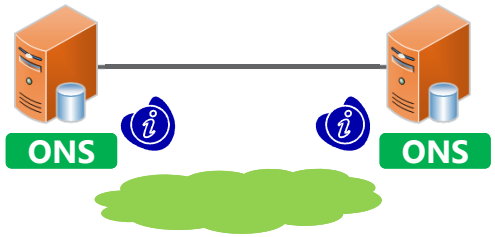


- Oracle Grid Infrastructure is necessary to register with ONS
  - ONS default ports – local: *6100*, remote: *6200* (firewall).
  - Configured and started automatically for GI cluster installations.
  - For GI standalone systems needs to be activated and configured manually (e.g. Data Guard).

```
srvctl enable ons
srvctl modify ons -remoteservers <remote_node> -verbose
srvctl start ons
```

- Database needs to be registered in OCR/OLR with the *ora.database.type* type
  - Does not work for user defined resources (failover databases).
- Can be used with different client types: JDBC, OCI, ODP.net
  - Integrated with UCP, starting with 11gR2 FAN API can be used (SimpleFan.jar)

Requirements For Using FAN Events (2)



- Correct database service configuration is necessary
  - *rlbgoal* needs only to be set to receive runtime load balancing advisory events.

```
srvctl add service                                        #The same for GDS (gdsctl)
  -clbgoal       [SHORT|LONG] #LONG is the default
  -rlbgoal       [SERVICE_TIME | THROUGHPUT | NONE]
  -notification [TRUE | FALSE]          #To enable FAN for OCI/ODP.net connections
```

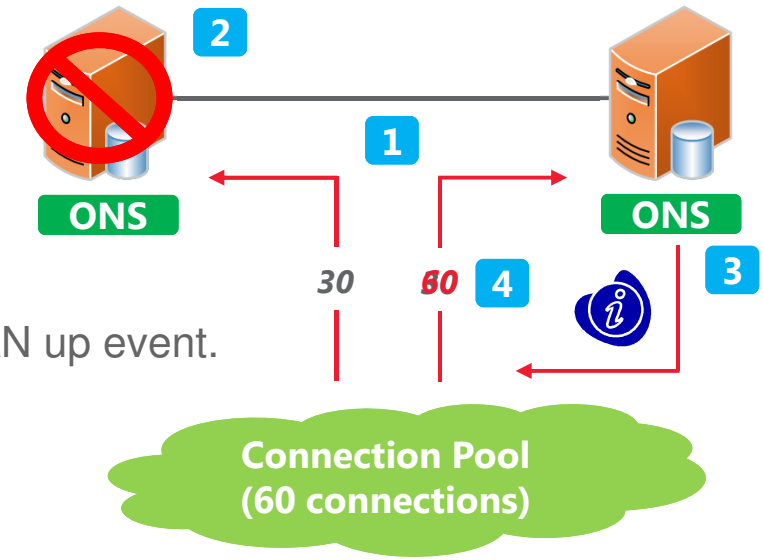
- Beginning with the 12c version (client and server), FAN-enabled clients can use FAN *auto-configuration*
  - For older versions you need to specify the ONS endpoints manually.

```
pds.setONSConfiguration("nodes=blue.trivadis.com:6200,brown.trivadis.com:6200");
```


trivadis makes IT easier.

Fast Connection Failover – Overview

- Pre-configured client side FAN integration for JDBC clients
  - It works in combination with connection pooling mechanism, so the Universal Connection Pool (UCP) or WebLogic Server Active GridLink is necessary.
- Reacts to up/down FAN events
  - Remove dead connections from connection pool after receiving FAN down event and redistributing them, if applicable, to other available nodes.
  - Connection re-distribution after receiving FAN up event.
- Do not configure TAF with FCF for JDBC thick (OCI) clients.



trivadis makes IT easier.

Fast Connection Failover – FAN Event

- Example Fast Connection Failover processing information after a service down FAN event received by a client application (switchover in a Data Guard environment).

```
Oct 11, 2016 10:52 AM SUCCESS <Reason:user> <Type:SERVICE_DOWN>
<Service:"sales_rw.trivadis.com"> <Instance:"dg2"> <Db:"dg2_site2">
Connections:(Available=20 Affected=20 FailedToProcess=0 MarkedDown=20
Closed=20)(Borrowed=0 Affected=0 FailedToProcess=0 MarkedDown=0
MarkedDeferredClose=0 Closed=0)
```

- The FCF information can be processed in the application exception code.

```
catch (SQLException ex) {
   if (conn == null || !((ValidConnection) conn).isValid()) {
     String fcfInfo =
((OracleJDBCConnectionPoolStatistics)pds.getStatistics()).getFCFProc
essingInfoProcessedOnly();
```


trivadis makes IT easier.

Fast Connection Failover – Setup

- Example how to use FCF with *Universal Connection Pool* (UCP)
  - Configure ONS and database service.
  - Include UCP and ONS libraries in your *CLASSPATH* (not part of the *Oracle Instant Client* installation).

```
CLASSPATH=.:/usr/lib/oracle/12.1/client64/lib/ojdbc7.jar:/usr/lib/oracle/12.1/client
64/lib/ons.jar:/usr/lib/oracle/12.1/client64/lib/ucp.jar
```

- To subscribe to *FAN* events and use HA *UCP* features you need to activate *FCF* first.

```
...
PoolDataSource  pds = PoolDataSourceFactory.getPoolDataSource();
pds.setConnectionFactoryClassName("oracle.jdbc.pool.OracleDataSource");
pds.setURL(dbURL);
...
pds.setFastConnectionFailoverEnabled(true);  //not activated per default!
```

Setting connection pool properties

Fast Connection Failover – Restrictions

- *Fast Connection Failover* restrictions
  - In-flight transactions are lost as well as calls in the middle of execution.
  - As with TAF, FAN is not designed to hide server process failure (*No more data to read from socket*).
  - Application exception handling is absolutely necessary!

Oracle Client Failover

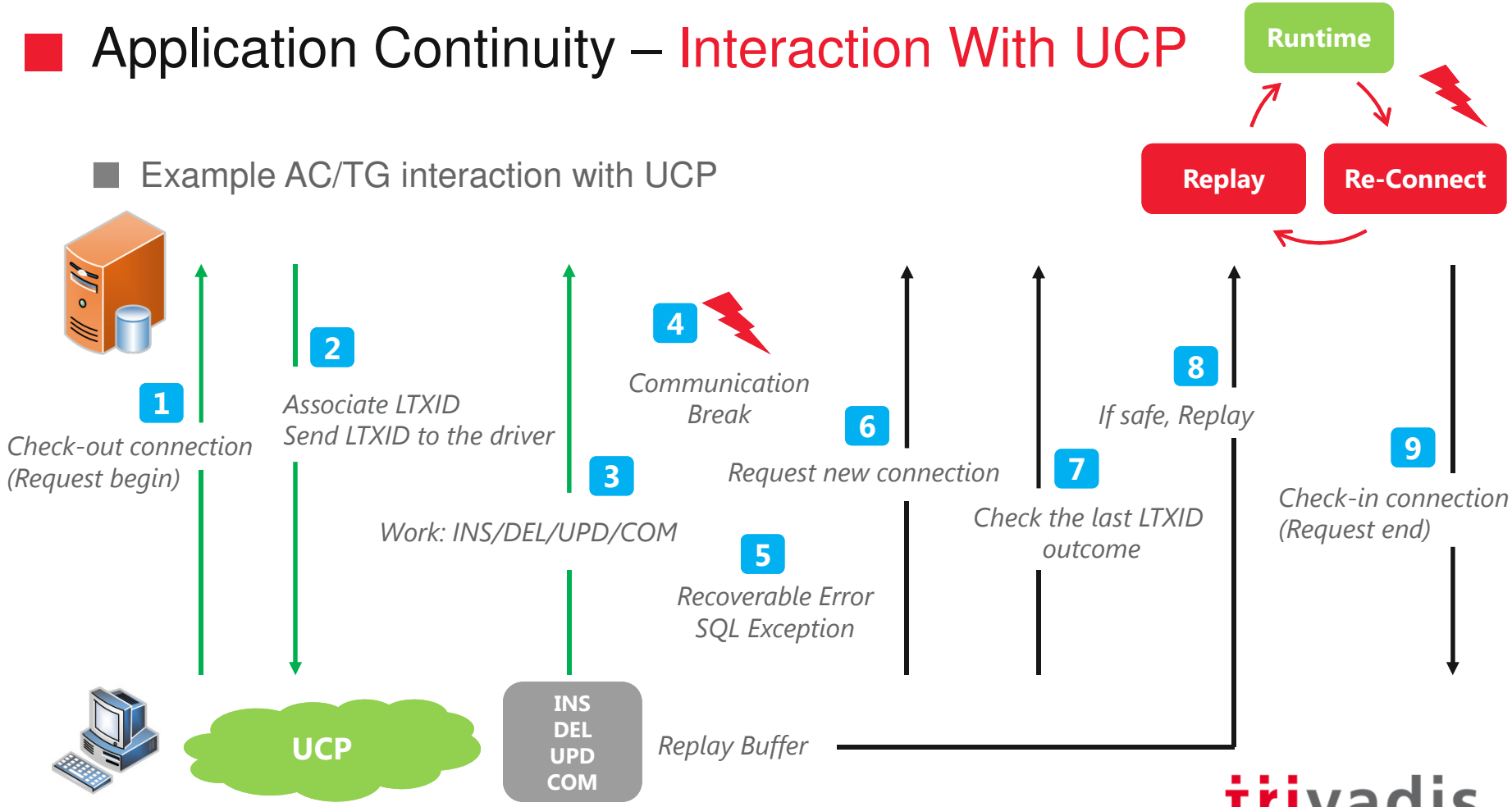
Application Continuity

Application Continuity – Overview

- Addresses temporary recoverable outages of instances, databases and network communications.
- Transaction Guard – server side component
  - Transaction state is recorded and retrievable within database in order to ensure idempotent execution on replay (*DBMS_APP_CONT.GET_LTXID_OUTCOME).*
  - Can be used standalone using Oracle Client 12c for *JDBC thin*, *OCI* and *ODP.net.*
  - Available with Oracle 12c Enterprise Edition.
- Oracle 12c JDBC Replay Driver – client side component
  - Replays the failed request so that the client may simply continue.
  - As of 12.1 implemented only for *JDBC thin* client, in 12.2 also OCI and *ODP.net.*
- Application Continuity requires *RAC* or *RAC One Node* or *ADG* (*GG*) option.

Application Continuity – Interaction With UCP

- Example AC/TG interaction with UCP



trivadis
makes IT easier.

Application Continuity – Application Changes

- Application Continuity with *UCP*

```
PoolDataSource  pds = PoolDataSourceFactory.getPoolDataSource();
pds.setConnectionFactoryClassName("oracle.jdbc.replay.OracleDataSourceImpl");
...
conn = pds.getConnection();                // Implicit database request begin
  // JDBC calls protected by Application Continuity
conn.close();                              // Implicit database request end
```

- Application Continuity without connection pool

```
OracleDataSourceImpl ods = new OracleDataSourceImpl();
conn = ods.getConnection();
...
((ReplayableConnection)conn).beginRequest(); // Explicit database request begin
  // JDBC calls protected by Application Continuity
((ReplayableConnection)conn).endRequest();   // Explicit database request end
```

Application Continuity – Requirements (1)

- Database service attributes need to be correctly specified for AC and TG

```
srvctl add service
 -failovertype TRANSACTION    # to enable Application Continuity
 -commit_outcome TRUE         # to enable Transaction Guard
 -retention 86400             # the number of seconds the commit outcome is retained
 -replay_init_time 900        # seconds after which replay will not be initiated
 -failoverretry 20
 -failoverdelay 2
 -notification TRUE    # with Oracle Restart, to avoid ORA-44781 during service start
```

Application Continuity – Requirements (2)

- Mutable Values
  - Replay is aborted whenever a data divergency is encountered between original and replay requests and answers.
  - Sequences can be configured to keep their values on replay.

```
GRANT KEEP SEQUENCE ON <SEQUENCE> TO USER <USER>;
```

  - SYSDATE/SYSTIMESTAMP can be configured to keep their values on replay.

```
GRANT KEEP DATE TIME TO <USER>;
```

  - SYS_GUID can be configured to keep values on replay.

```
GRANT KEEP SYSGUID TO <USER>;
```

Application Continuity – Deactivating Replay

- Killing/Disconnecting a session without replay

```
ALTER SYSTEM KILL SESSION 'sid, serial#, @inst' NOREPLAY;
ALTER SYSTEM DISCONNECT SESSION 'sid, serial#, @inst' NOREPLAY;
EXECUTE DBMS_SERVICE.DISCONNECT_SESSION('[service_name]', DBMS_SERVICE.NOREPLAY);
```

- Stopping a service without replay

```
srvctl stop service -db RAC_SITE1 -instance RAC2 -service OLTP -force \
                    -stop_option immediate -noreplay
```

- Some restrictions:
  - Autonomous transactions, XA, ADG with read/write DB links, GoldenGate or Logical Standby databases not supported
- Error handling still necessary (non-recoverable errors, replay not possible, etc.)

Conclusions

Conclusions

- To achieve high availability, correct client-side configuration for failover is crucial.
- Tuning OS kernel parameters is not the preferred way to go.
- At least Oracle client connect timeouts should be set.
- Be careful with Oracle re-connect timeouts (undesired side effects).
- VIP addresses are very useful in cluster environments and solve many problems out of the box.
- Dynamic database services are key to client high availability.
- TAF/FAN/FCF are very powerful
  - But with some limitations – and exception handling is still necessary!
- Application Continuity helps to transparently replay in-flight transactions.
  - Exception handling is still necessary!