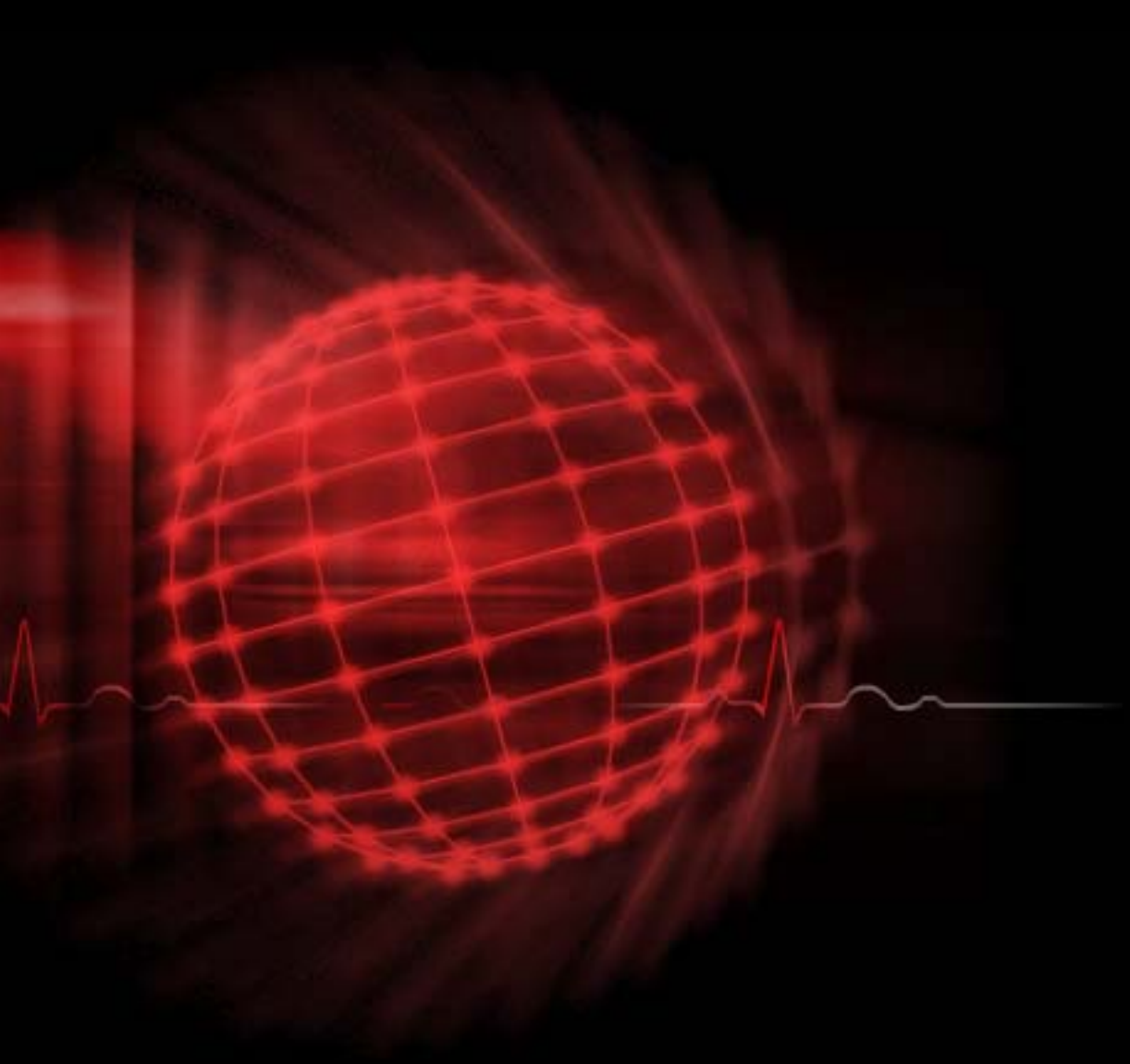


ORACLE 10g
GATEWAYS

Art Tumolo & Marc Connolly
Global Strategy and Solutions - IBM
Oracle Corporation

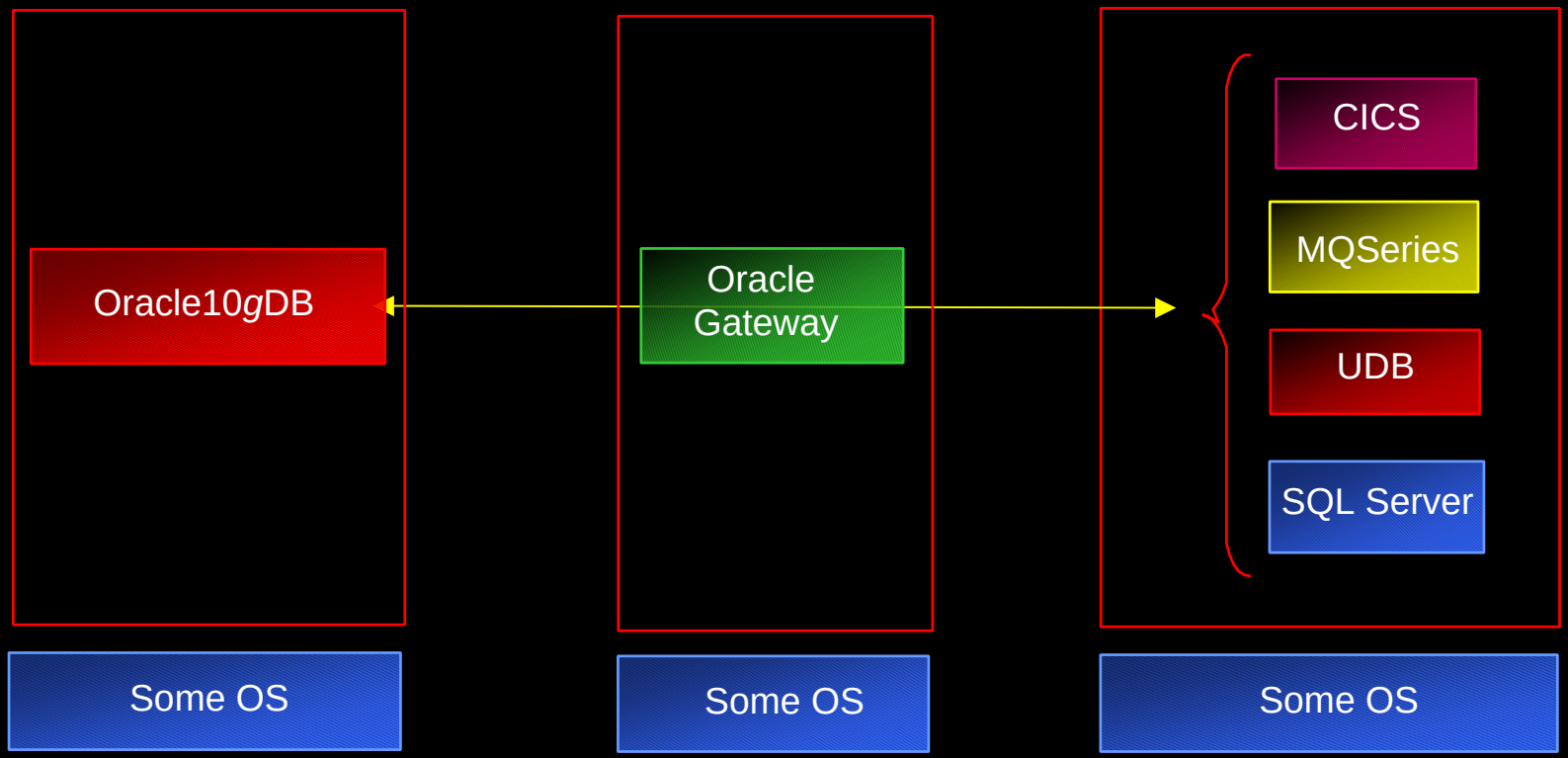
Agenda

- Configurations
- Architecture, Types, Implementations
- General Application & Use
- Gateways under zOS
  - OSDI
  - WLM Considerations
- Leveraging Oracle Gateways within 10*g* AS
- Dialog

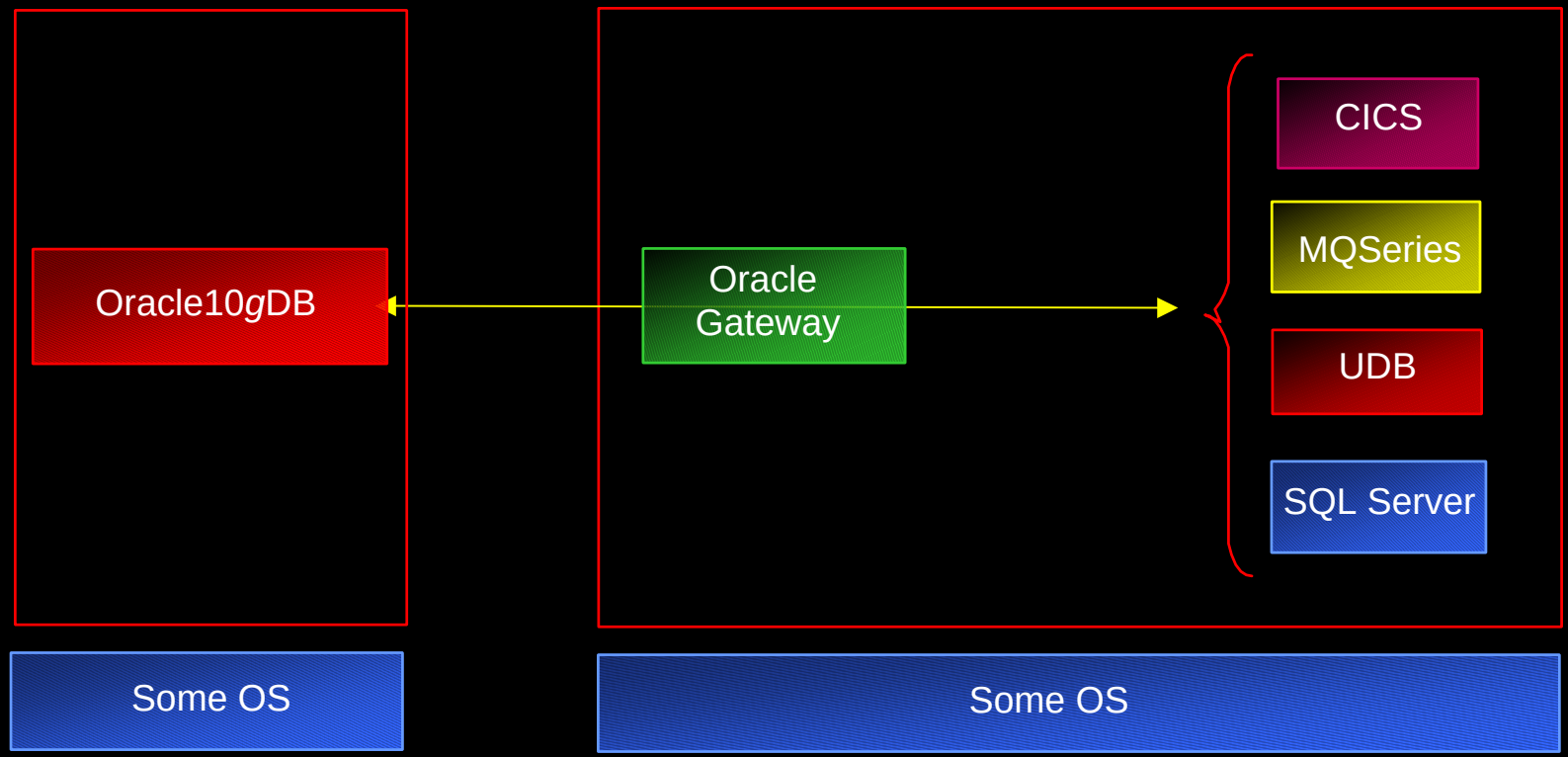
Oracle Gateways

Configurations, Architecture, Types
&
Implementation

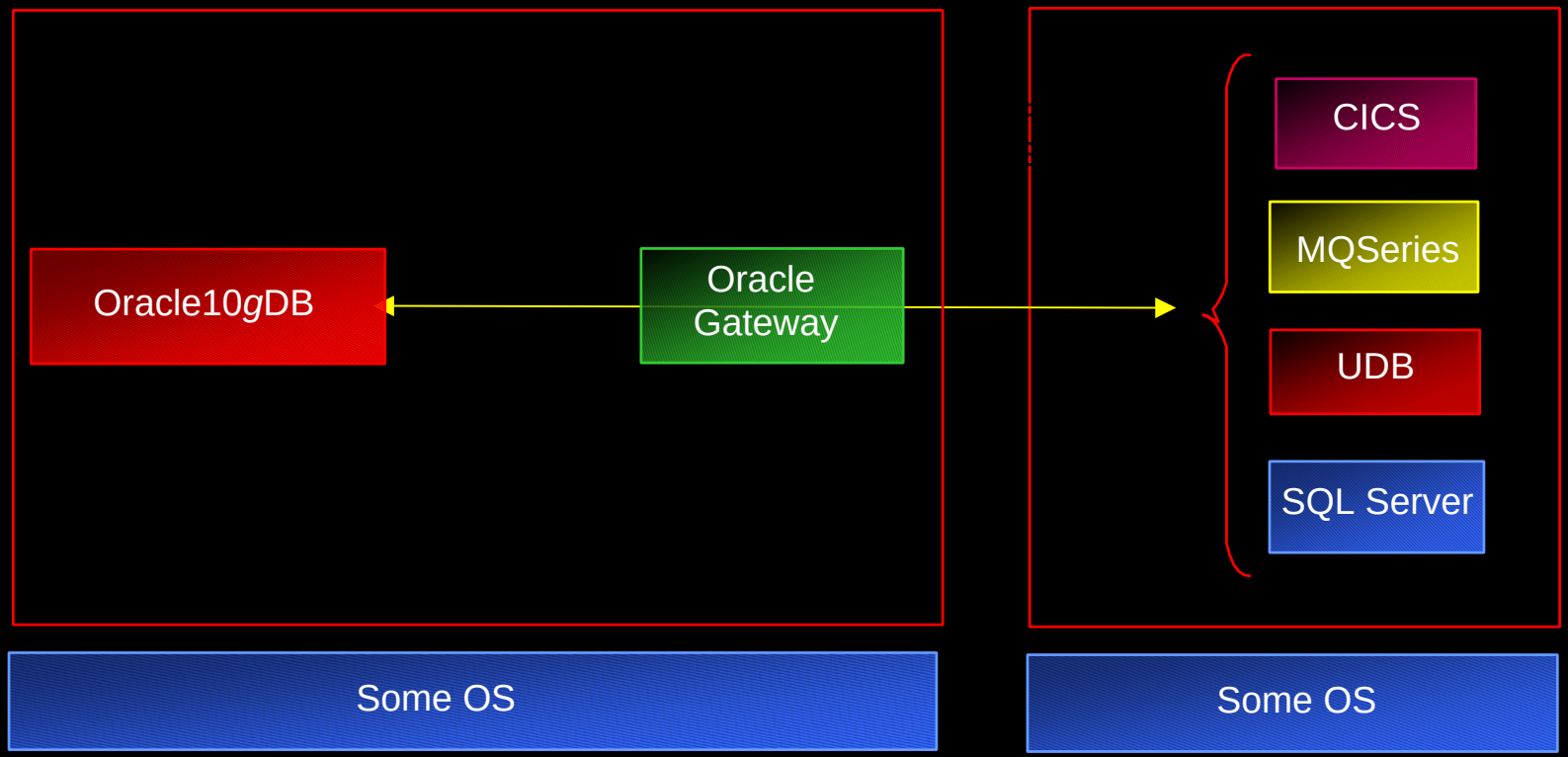
Oracle Gateway Configurations



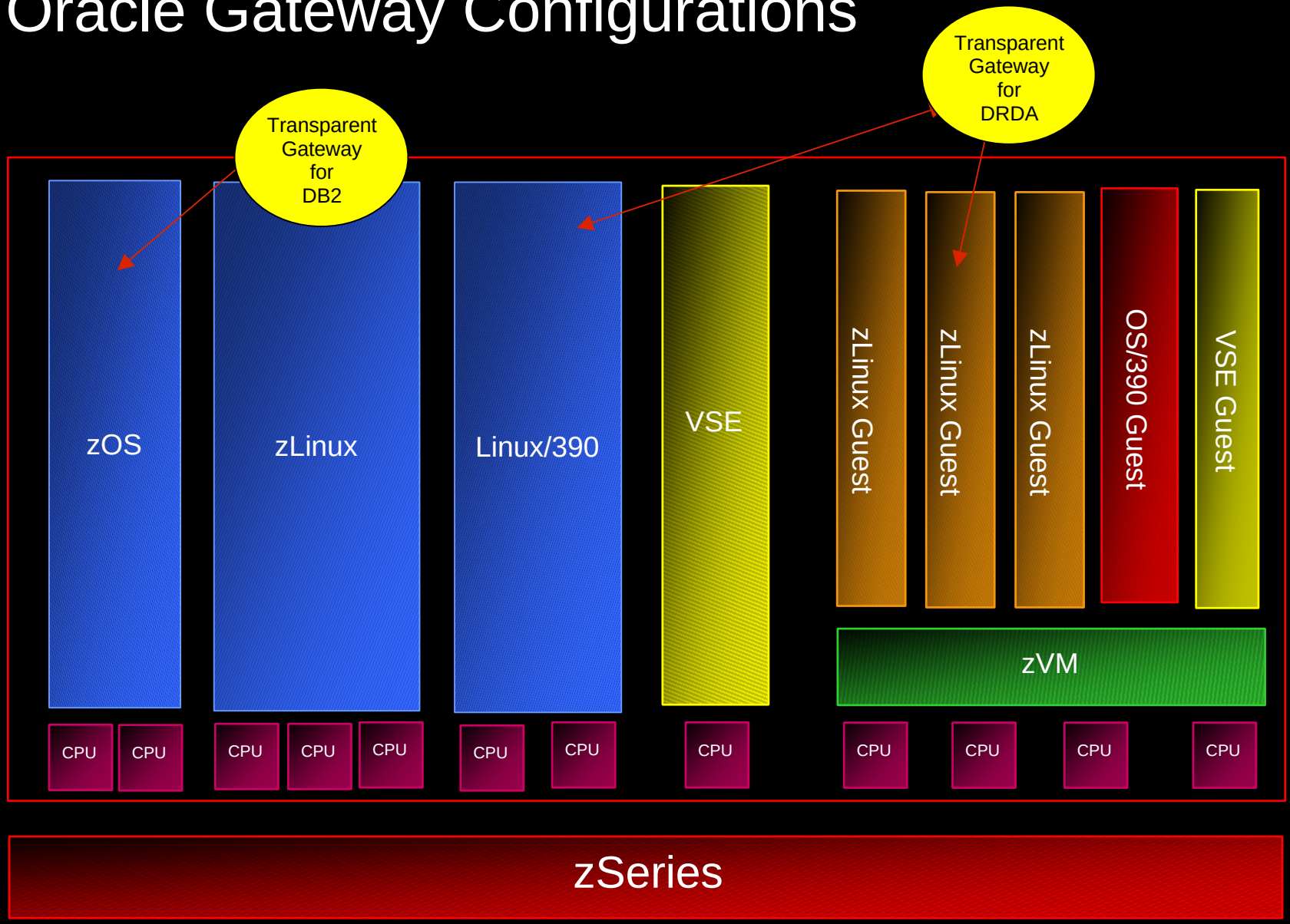
Oracle Gateway Configurations



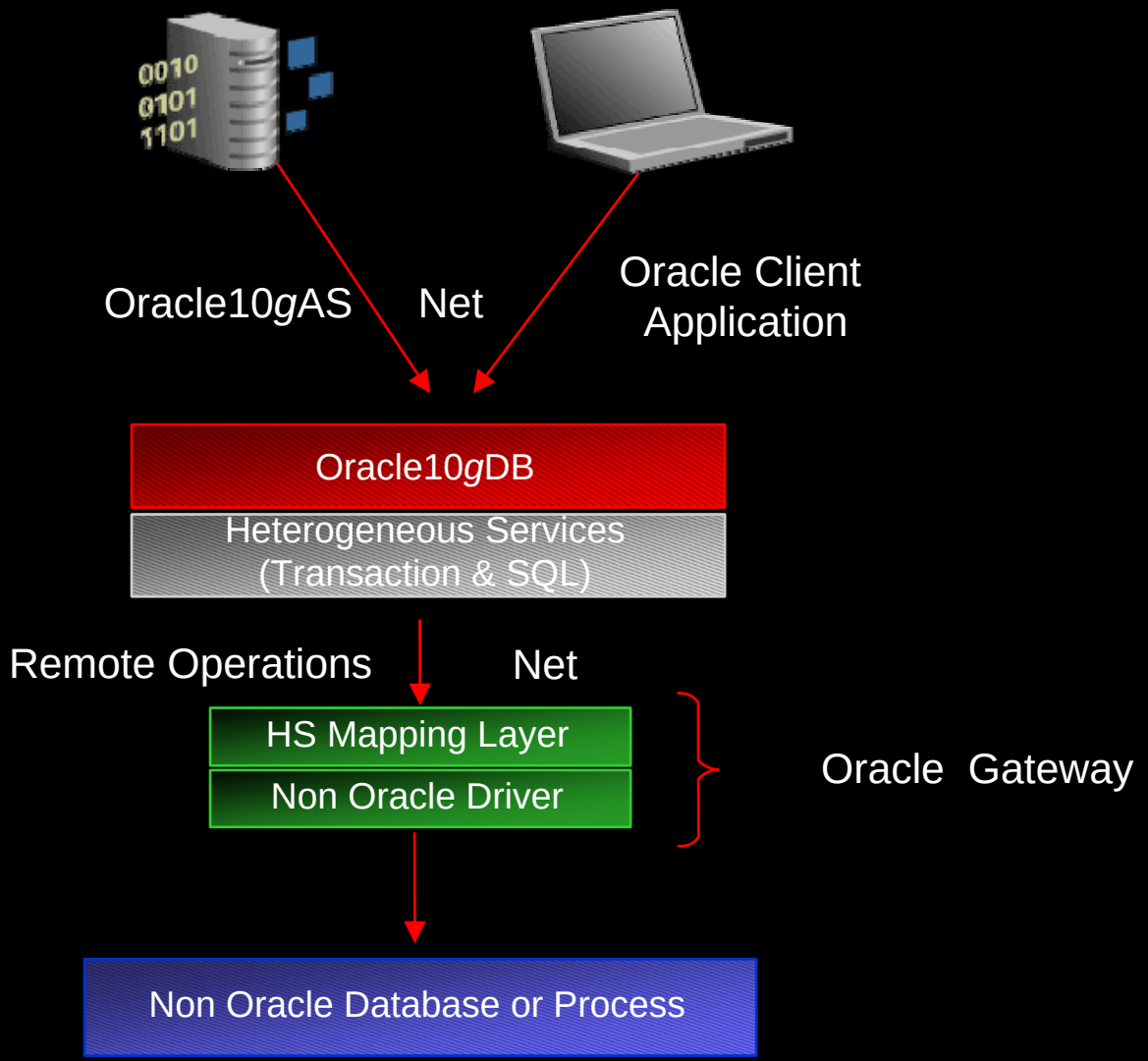
Oracle Gateway Configurations



Oracle Gateway Configurations



Oracle Gateways Architecture



Oracle Gateway Types

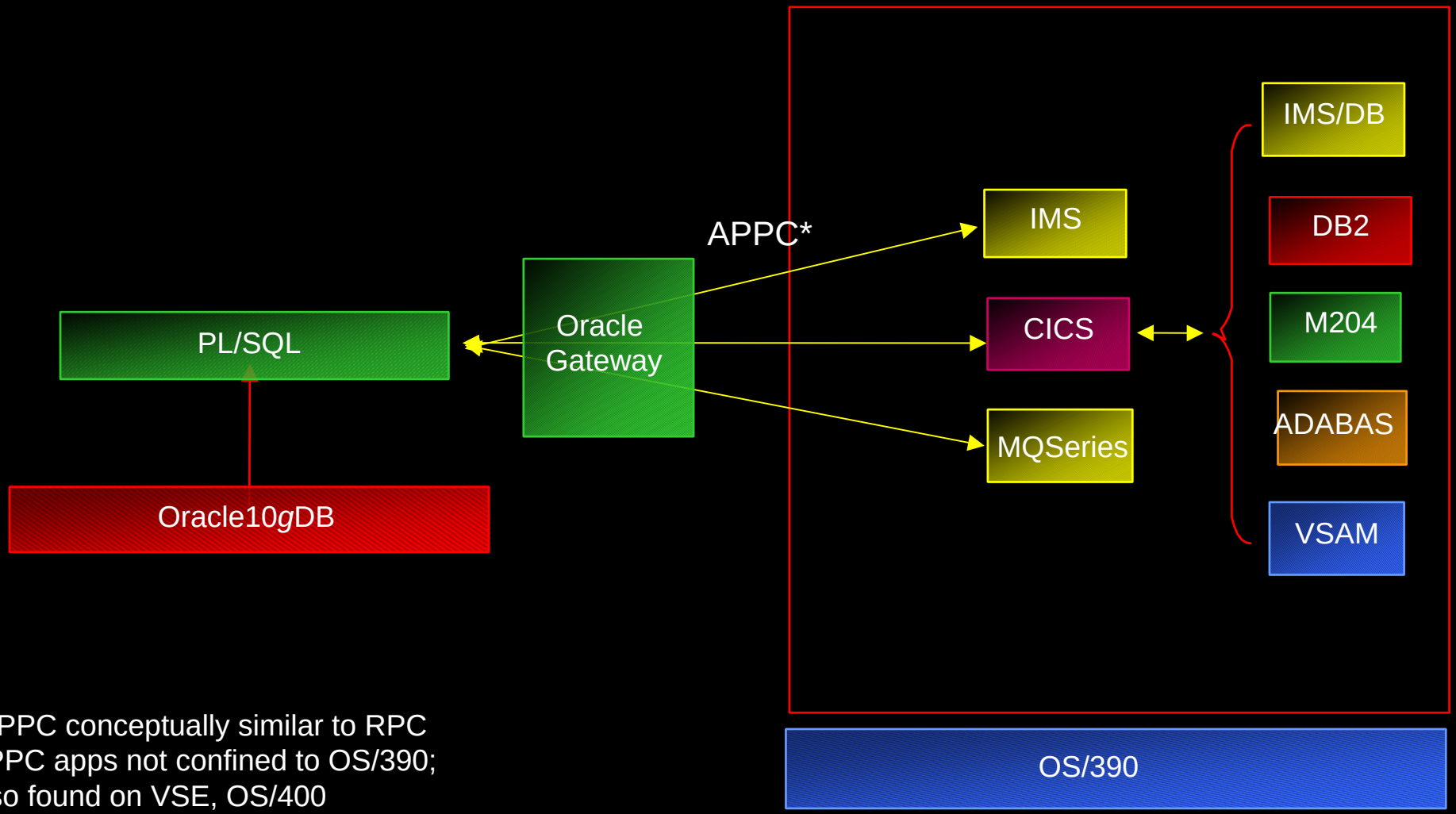
- Procedural Gateways
  - APPC
  - MQSeries
- Transparent Gateways
  - Informix, Sybase, Ingress, RDB, SQL Server
  - iWay (EDA/SQL Server)
  - DB2
  - DRDA
- Messaging Gateway
  - Websphere MQ, TIBCO

Oracle Procedural Gateway for APPC

APPC

- Advanced Program-to-Program Communications
  - IBM Designed and Architected
  - Part of IBM's Systems Network Architecture (SNA)
  - Distributed Process Protocol
- Comprised of System Software with an API
- Comprised of Set of Verbs
  - Management Verbs (Session Management, Activate Session, ...)
  - Transaction Program Verbs (Start, End, Get Properties,..)
  - Conversation Management (between TP's, Allocate, Send, Wait,..)

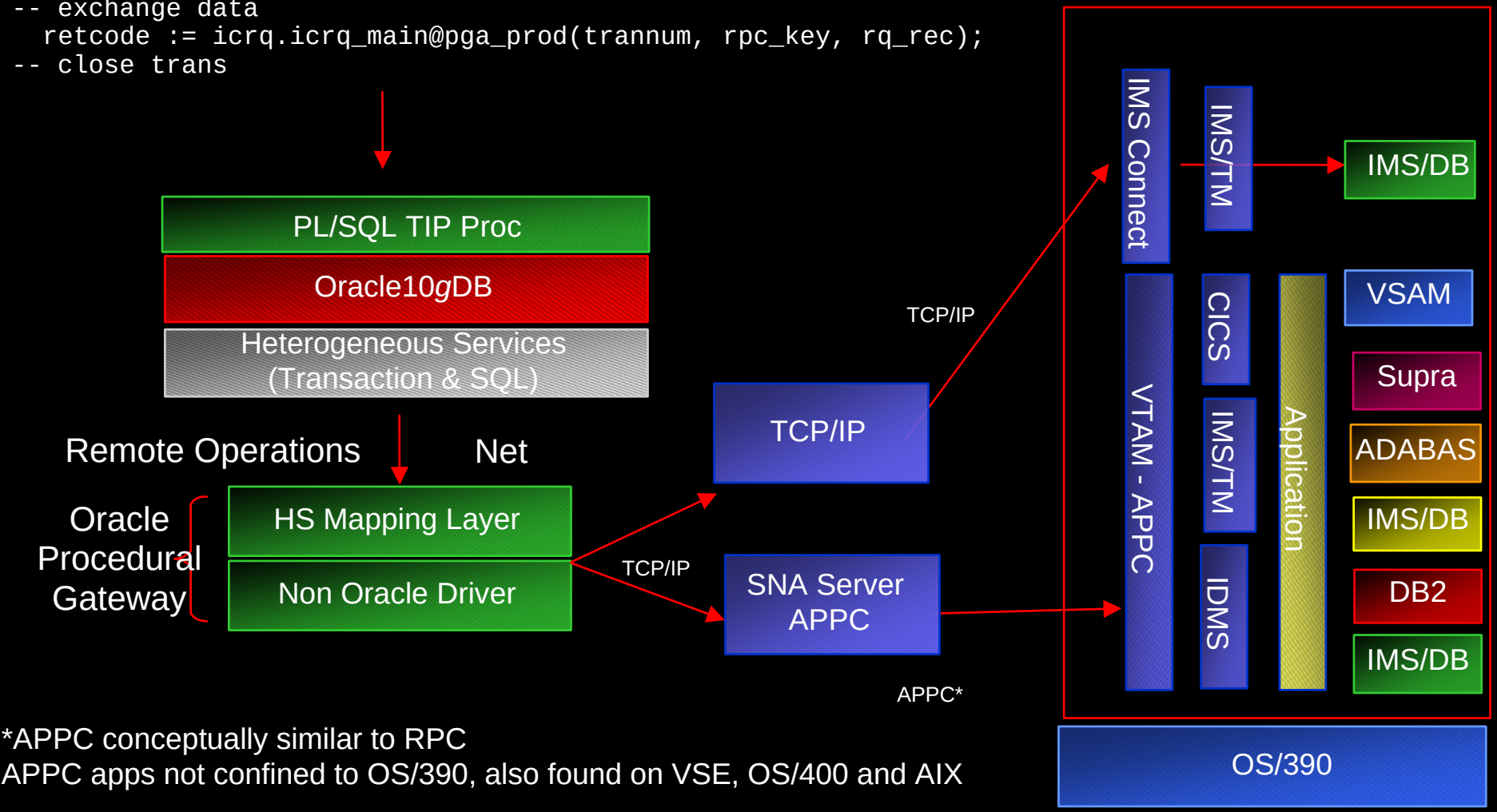
Oracle Procedural Gateway for APPC



*APPC conceptually similar to RPC
APPC apps not confined to OS/390;
also found on VSE, OS/400

Oracle Procedural Gateway for APPC

```
-- init trans
  retcode := icrq.icrq_init@pga_prod(trannum);
-- exchange data
  retcode := icrq.icrq_main@pga_prod(trannum, rpc_key, rq_rec);
-- close trans
```



*APPC conceptually similar to RPC
APPC apps not confined to OS/390, also found on VSE, OS/400 and AIX

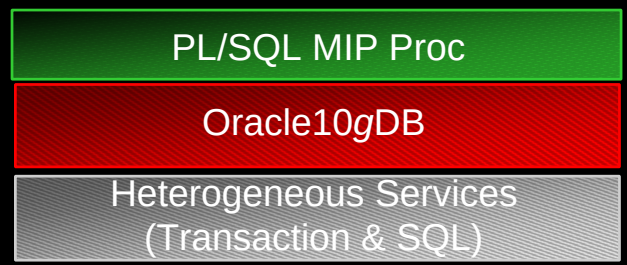
Oracle Procedural Gateway for MQSeries

MQSeries

- Advanced Program-to-Program Communications
  - Allows users to manipulate IBM MQSeries messages thru PL/SQL calls
  - Transaction control: Oracle commit-confirm protection is extended to the MQSeries environment without any special programming.
  - Provides pure JAVA based GUI front end Visual Workbench which generates PL/SQL code from COBOL copy books describing message format.
  - MQSeries Triggers are a feature that enables an application to be started automatically when a message event, such as the arrival of a message, occurs. Triggers can be used to invoke programs or transactions.

Oracle Procedural Gateway for MQSeries

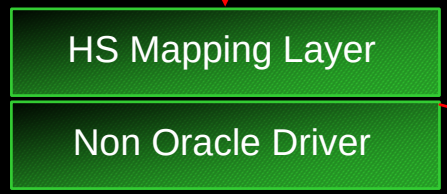
```
-- establish access to the queue
retcode := hire.qopen(queue_name)
-- put a employee message on the queue
retcode := hire.enqueue(payload)
-- get a employee message from the queue
retcode := hire.dequeue(payload)
-- relinquish access to the queue
retcode := hire.qclose(queue_name)
```



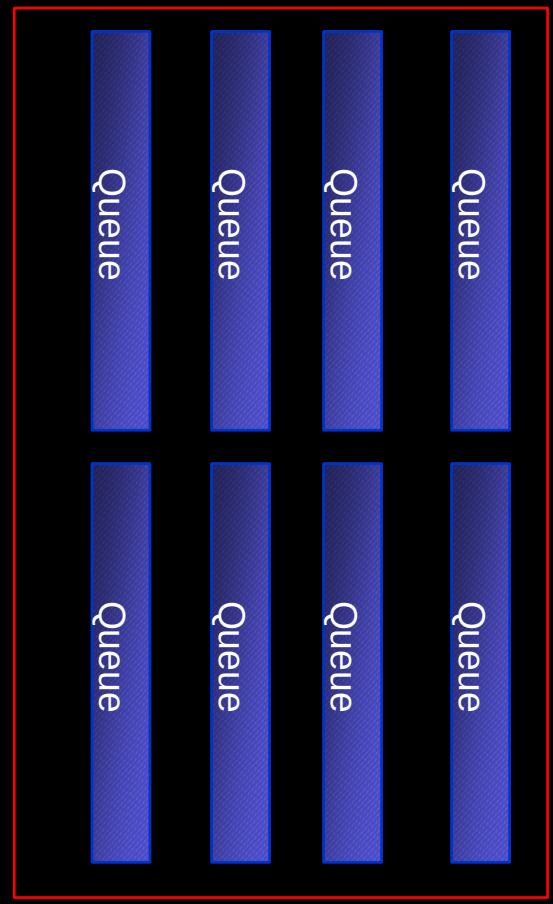
Remote Operations

Net

Oracle Procedural Gateway



TCP/IP



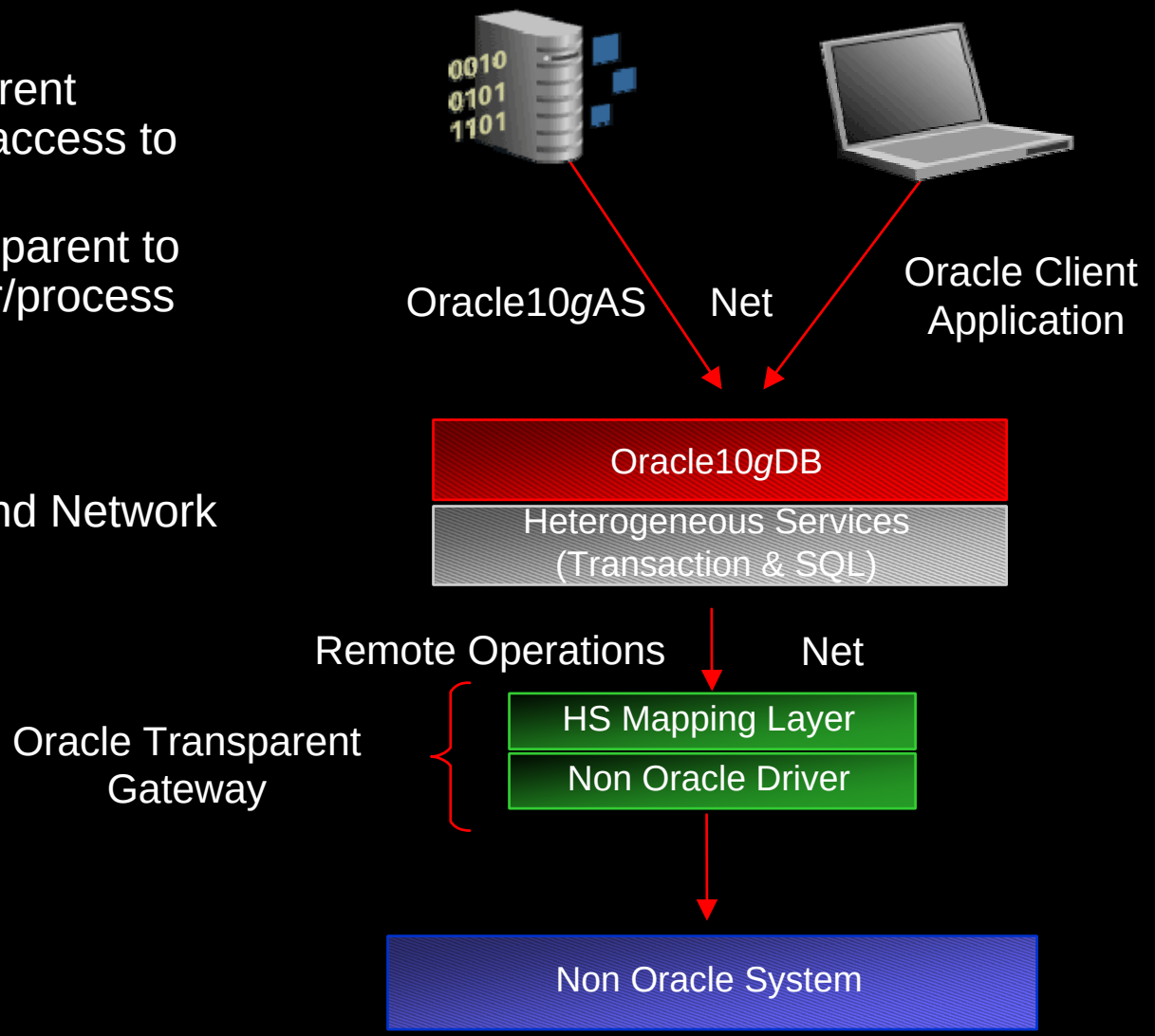
MQ Series Queue Manager

MQSeries apps not confined to OS/390, also found on VSE, OS/400 and AIX

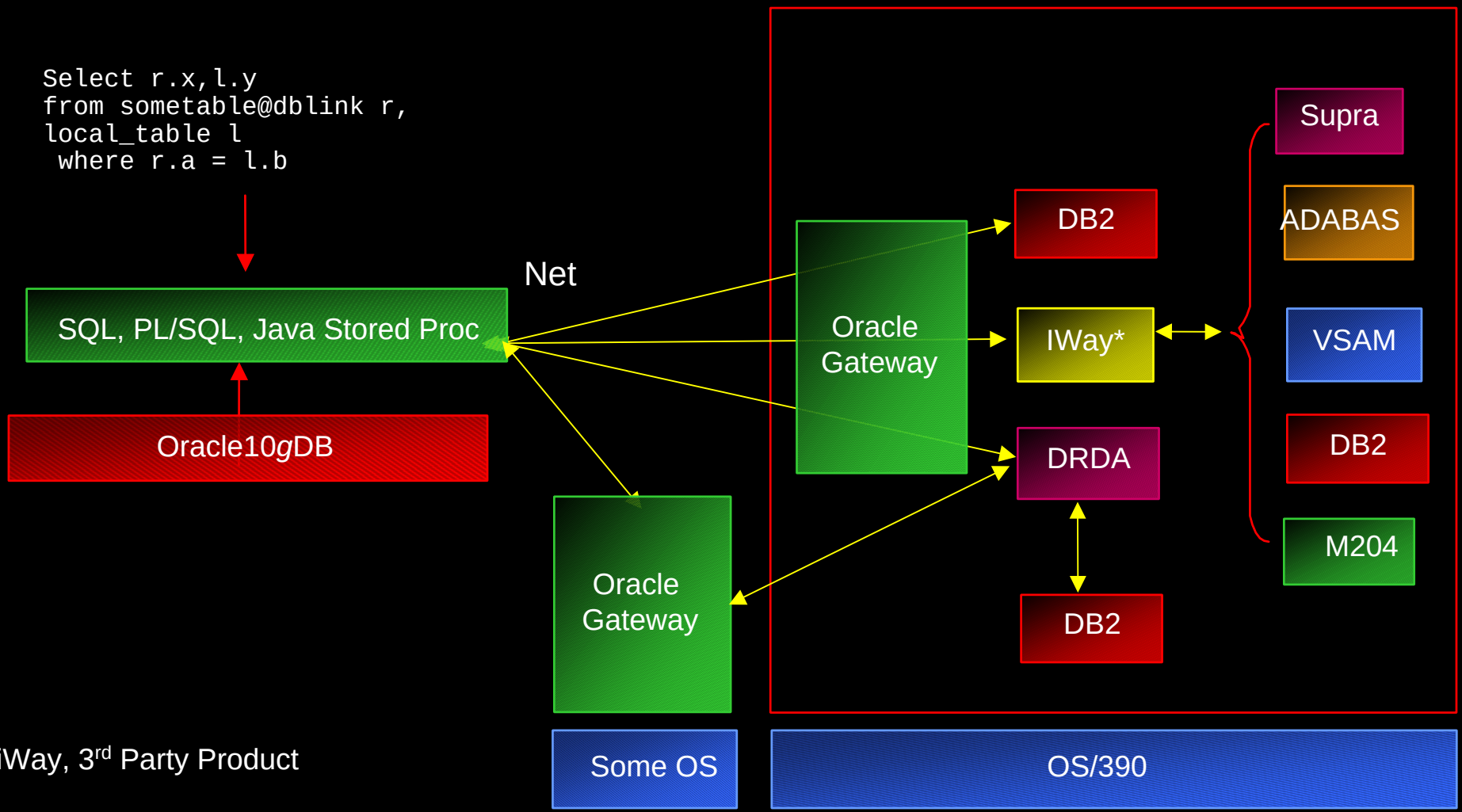
Oracle Transparent Gateways

Oracle Transparent Gateways

- Oracle10*g* Transparent Gateways provide access to non-Oracle system
- The access is transparent to the application user/process
  - Location
  - Commit
  - SQL
  - Platform, O/S and Network



Oracle10*g* Transparent Gateways



*iWay, 3rd Party Product

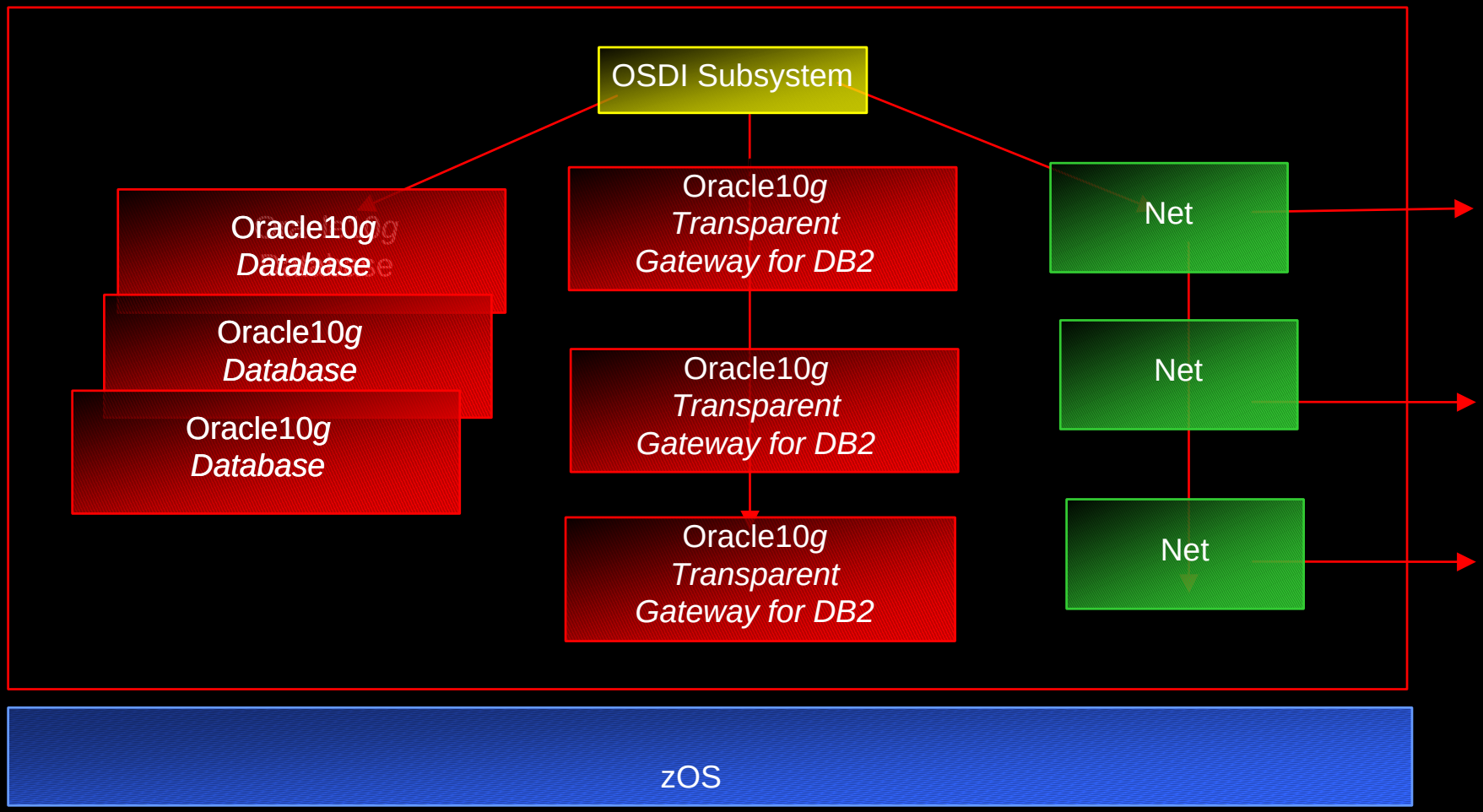
Oracle10*g* Gateways under zOS

OSDI

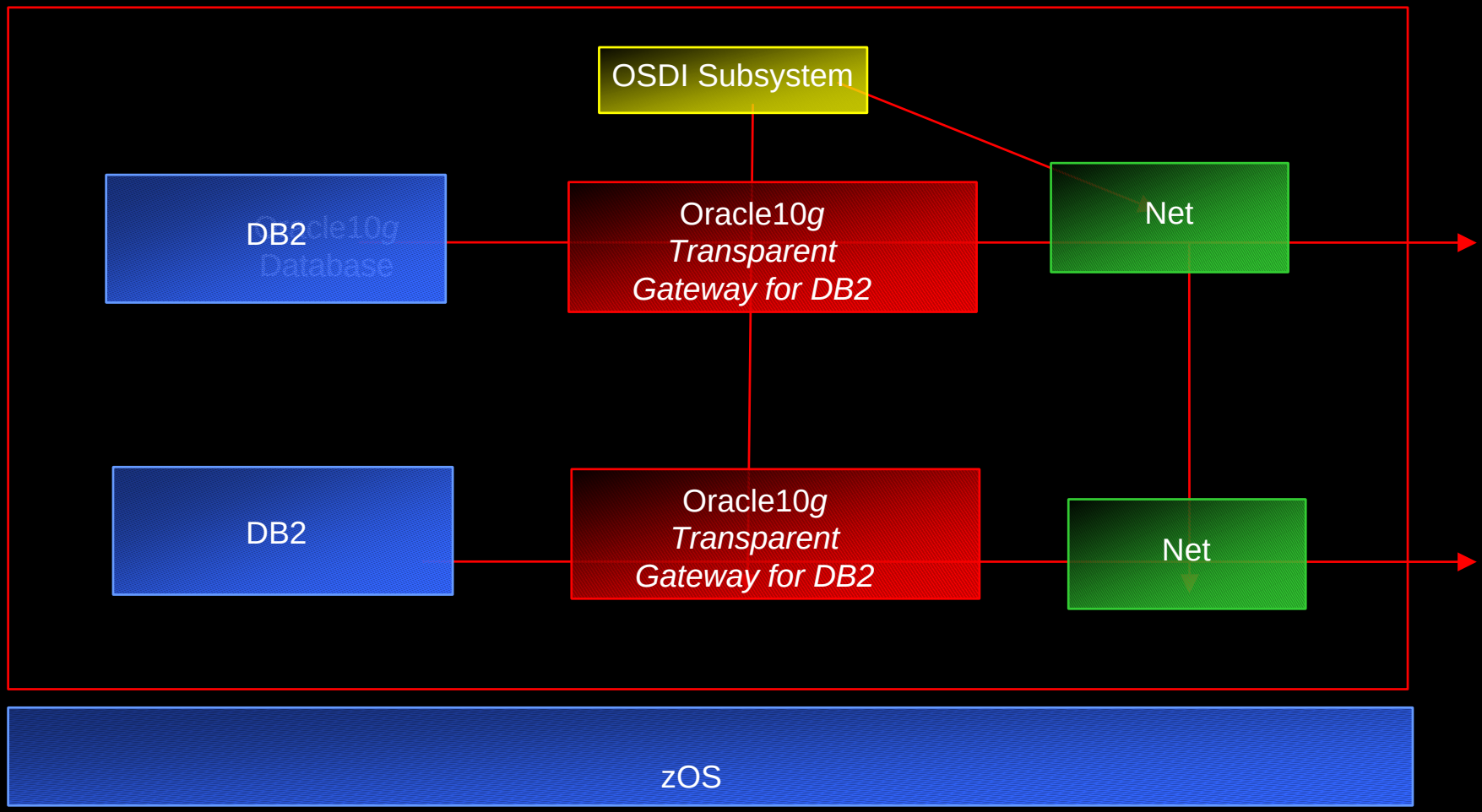
OSDI – Operating System Dependent Interface

- Subsystem - Common management layer shared by all Oracle products on OS/390
  - Primary role is to manage services
  - Manage connections
- Services (Database, Network & Gateways)
- Security - Authenticates connections to Services via SAF (using RACROUTE)

OSDI – Operating System Dependent Interface



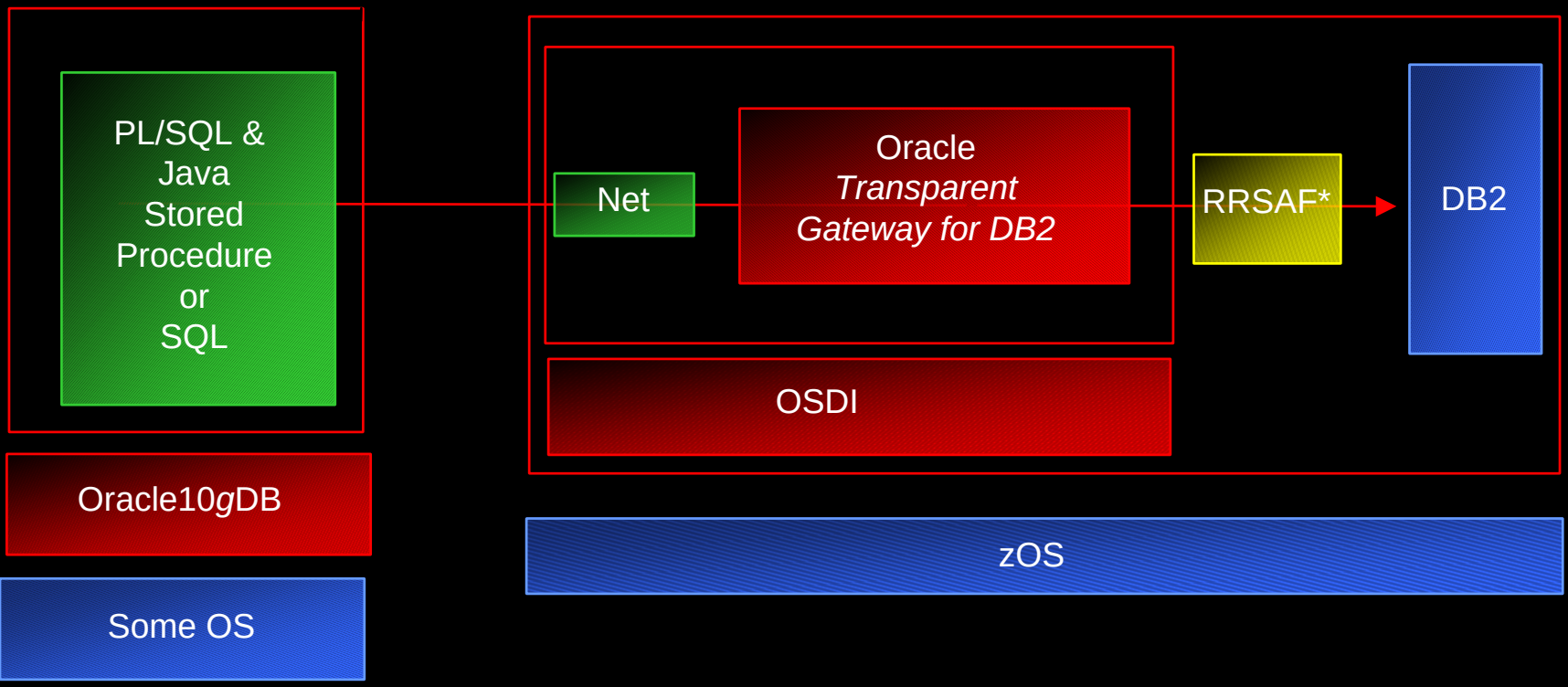
OSDI – Operating System Dependent Interface



Oracle Gateways

zOS Considerations

Oracle10*g* Transparent Gateway for DB2



*In *Oracle9i Transparent Gateway, the DB2 Call Attach Facility (CAF) is used*

Transparent Gateway for DB2

- TG4DB2 in 8*i* was based on MPM
  - All work in TG4DB2 ran at priority of TG4DB2
- TG4DB2 in 9*i* is based on OSDI
  - Work from network comes through Oracle NET
    - Runs in an enclave until it gets to TG4DB2 A/S
    - TCB per session, uses Call Attach Facility (CAF)
  - You *must* classify this enclave
    - Velocity goal with high importance is appropriate

Transparent Gateway for DB2

- With latest OSDI patches the work in TG4DB2 TCB is joined to the enclave
  - Some of DB2 code runs in the enclave
  - Background DB2 processes charged to DB2
  - Response time goals possible
- In 10g, CAF removed, RRSAF used to connect to DB2
  - **All** DB2 work runs in the enclave
  - Response time goals are recommended

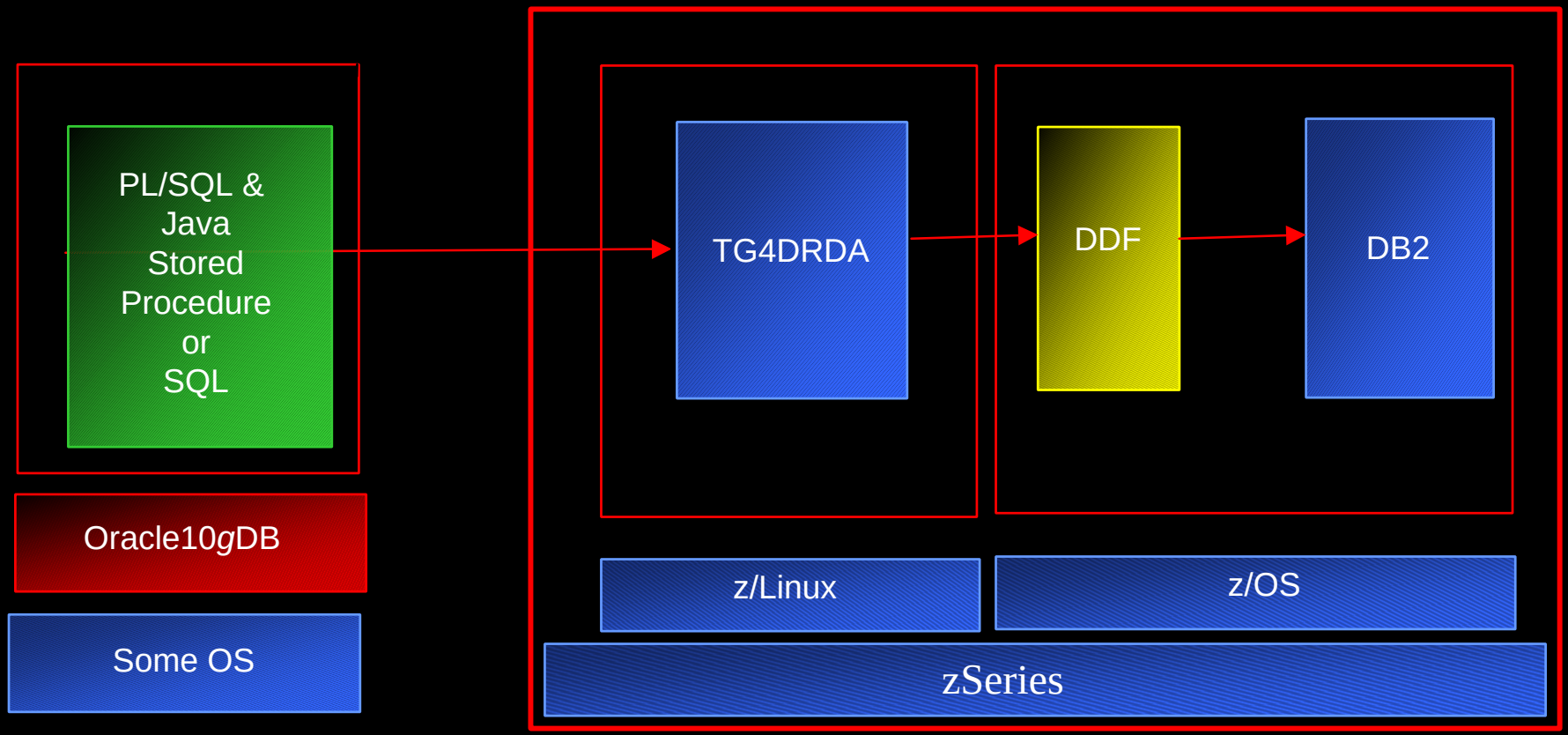
Oracle10*g* Transparent Gateways

General Application & Use
(DRDA & DB2 as Example)

IBM's DRDA

- Distributed Relational Database Architecture
  - Designed and Developed By IBM
  - Database to Database & Application Comm Protocol
  - AR – Application Requestor, AS – Application Server
  - Levels of Certification and Compliance
  - Conceptually the same as Oracle Net
- Oracle's DRDA Gateway
  - Connectivity is via TCP/IP or SNA
  - Acts as a DRDA Application Requestor (AR)

Oracle10*g* Transparent Gateway for DRDA with Linux on zSeries



Symatics & General Utilization

- DB2 accessible via DBLINK:
  - *Select a, b, c from z,y@db2_link*
- Synonyms and Views can be used to facilitate use
- SQL/DDL Pass Through
  - Any “preparable” DB2 statement
  - Exceptions: CONNECT, COMMIT, ROLLBACK
- Native DB2 Stored Procedures Supported
- Two Phase Commit Supported
- Distributed Join/Union
- Sqlplus describe command for foreign objects in TG

General Application – SQL

- Create DBLINK:

```
CREATE DATABASE LINK dblink
    CONNECT TO userid identified by password
    USING 'tns_name_entry';
```

- Use Synonyms for greater transparency:

```
Create synonym some_name for sometable@dblink;
select * from some_name;
```

- SQL SELECT:

```
Select x,y,z from sometable@dblink where a = b order by a;
```

- SQL DISTRIBUTED JOIN:

```
Select r.x,l.y from sometable@dblink r, local_table l
 where r.a = l.b
```

General Application

- SQL REMOTE UPDATE:

```
create or replace trigger some_trigger
 after update of some_column on some_local_table
 for each row
begin
     update some_db2_table@dblink
     set some_db2_col = :new.some_col
     where some_other_db2_col = :new.some_local_col;
end;
```

- SQL Insert (not supported) :

```
INSERT INTO some_db2_table@dblink SELECT *
FROM some_local_table;
```

- SQL Insert (supported) :

```
COPY FROM local_userid/password@localInstance
INSERT some_db2_table@dblink
USING SELECT * FROM some_local_table;
```

General Application

- SQL PASS THROUGH:

```
rows_affected = dbms_hs_passthrough.execute_immediate@dblink
('delete from some_table where x = 1');
```

--or--

```
declare
  rows_affected integer;
begin

rows_affected:=dbms_hs_passthrough.execute_immediate@dblink
  ('create table some_table (col_1 integer, col_2
decimal(6,2), col_3 char(14)))');
end;
```

General Application

- CURSORS via PASS THROUGH:

```
 declare
  cursor_id binary_integer;
  return_code binary_integer;
  some_column_val varchar2(10)
begin
  cursor_id:=dbms_hs_passthrough.open_cursor@dblink;
  dbms_hs_passthrough.parse@dblink(cursor_id,'select some_column from
some_table');
begin
  return_code:=0;
  while (true)
  loop
    return_code:=dbms_hs_passthrough.fetch_row@dblink
(cursor_id,false);
    dbms_hs_passthrough.get_values@dblink
(cursor_id,1,some_column_val);
    insert into some_local_table values (some_column_val);
  end loop;
  exception
    when no_data_found then
      begin
        dbms_output.put_line('result set end');
        dbms_hs_passthrough.close_cursor@dblink(cursor_id);
      end;
    end;
end
```

General Application

- BIND & HOST VARIABLE USAGE:

```
declare
  cursor_id integer;
  rows_affected integer;
begin
  cursor_id := dbms_hs_passthrough.open_cursor@dblink;

  dbms_hs_passthrough.parse@dblink(cursor_id,
        'update employee set salary=salary*2 where employee_ssn=?');

  dbms_hs_passthrough.bind_variable@dblink(cursor_id,1,'240286734');


rows_affected:=dbms_hs_passthrough.execute_non_query@dblink(cursor_id
);

  dbms_output.put_line(rows_affected||' rows updated');

  dbms_hs_passthrough.close_cursor@dblink(cursor_id);
end;
```

General Application

- DB2 STORED PROCEDURES:

```
declare
  ssn number(9,0);
  bonus number(8,2);
begin
  ssn := 241928435;

  sysproc.get_employee_bonus@dblink(ssn, bonus);
  update local_employee set current_bonus = bonus where
employee_ssn = ssn;
end;
```

- Notes:
  - SYSPROCEDURES queried with authid, luname and procedure columns
  - SYSPROC qualifier only req'd in v5.1
  - AUTHID from user id of dblink
  - LUNAME from 'CURRENT SERVER'
  - Gateway looks for non-blank authid/luname entries first

General Application

- Two Phase Commit Guidelines
  - Gateway is commit coordinator
  - Only one gateway per 2PC UW
  - 2PC UW state stored in ORACLE2PC DB2 table
  - ORACLE2PC DB2 table needs to be available

General Application – Replication

- Oracle STREAMS Replication
- Rules based Application of Oracle DML Activity
- Oracle to DB2 Only at this time
- Simple Set Up
  - Create Capture/Apply Process
  - Define Replication Rules
  - Schedule / Start Capture/Apply Process
- Functions the same as it would in Oracle-only:
  - Changes are identified in the redo log, captured, and enqueued as logical change records.
  - Uses Heterogeneous Services and Transparent Gateway.
  - A single capture process may capture changes that will be applied at both Oracle and non-Oracle databases.

General Application – Replication

- Copying DB2 to Oracle:

```
create  local_table as select * from remote_db2@dblink;
                          --or--
insert into local_table select * from remote_db2@dblink;
```

- SNAPSHOT :

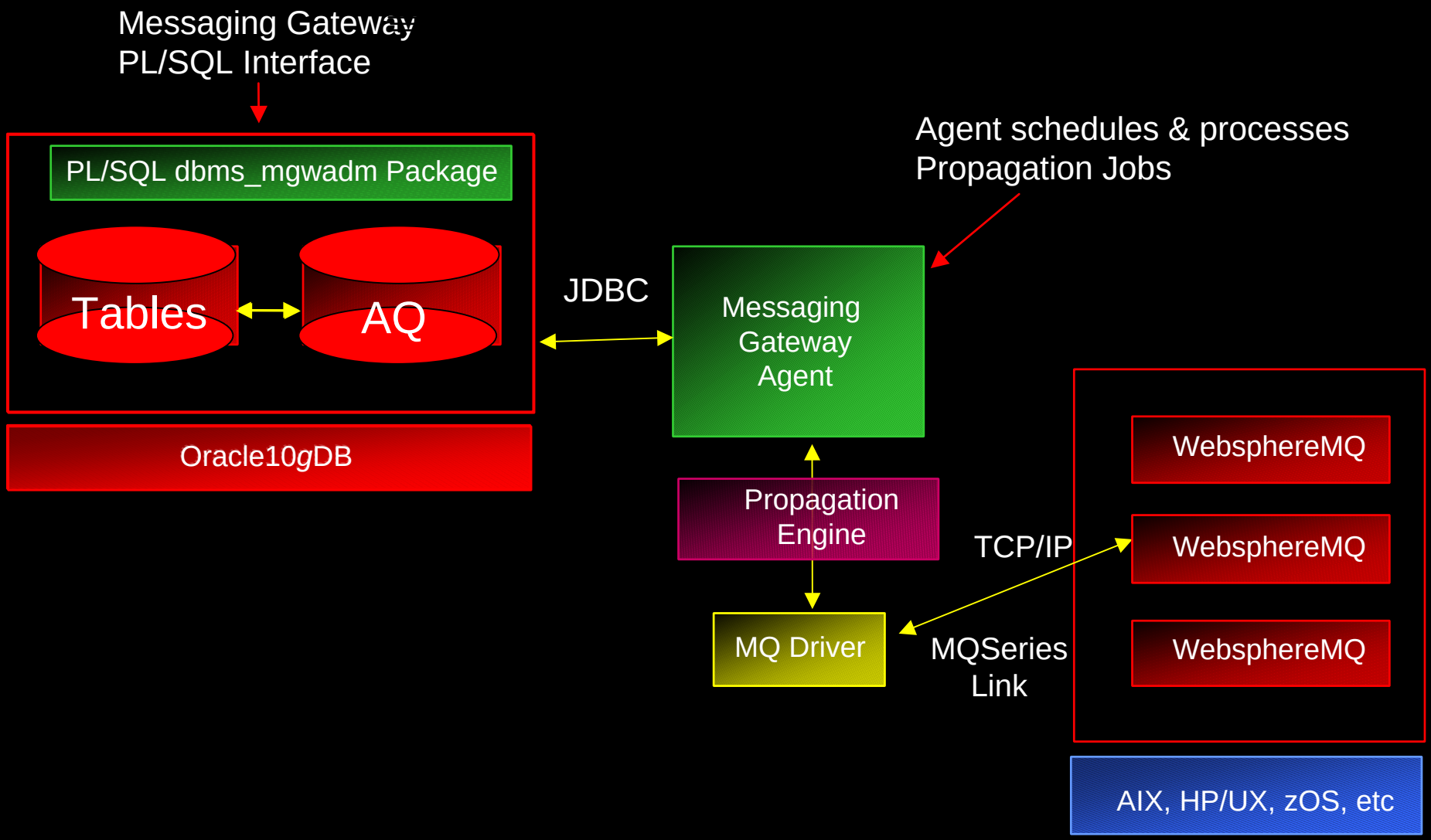
```
create snapshot local_table
  pctfree 10 pctused 50
  tablespace users
  storage (initial 50k next 50k)
  refresh complete next sysdate + 1
  with rowid
  as
    select * from remote_db2@dblink
  where some_remote_col = 'whatever';
```

Oracle Messaging Gateway

Oracle10*g* Messaging Gateway Architecture

- Gateway Agent
  - External C routine
  - Schedules and processes propagation jobs
- Administration Package
  - Configure the gateway
  - Manage the gateway
  - Monitor the propagation process

Oracle10*g* Messaging Gateway



Oracle10*g* Messaging Gateway Functionality

- Extends AQ Message Propagation
  - Outbound Message Propagation & Inbound Message Consumption
- Native Message Format
  - Oracle – RAW or ADT (abstract data type)
  - MQ - TEXT or byte messages of any type
- Messaging Conversion
  - Automatic or Customized message transformation functions

Oracle10*g* Messaging Gateway Functionality

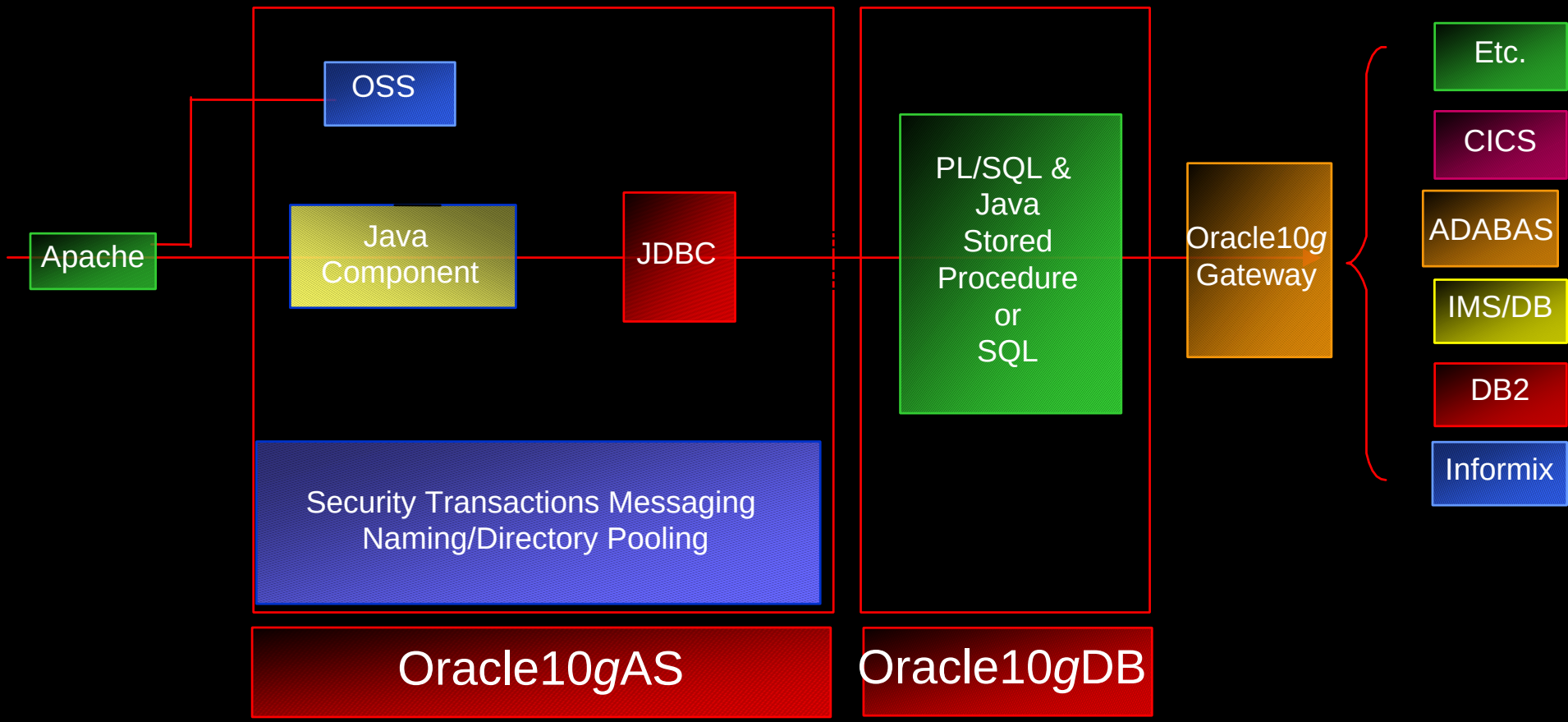
- Message Delivery Guarantees
  - With transaction support, *exactly-once* delivery of persistent messages
  - Without transaction support and/or non-persistent messages, *at-most-once delivery*
- Integrated in 10g DB
  - *Internal Process*
  - *Managed via AQ-like PL/SQL interface*
  - *Configuration stored in database*
  - *Documented in the Oracle® Streams Advanced Queuing User's Guide and Reference Release 10.1*

Oracle10*g* Messaging Gateway and Procedural Gateway Contrasted

- PG4MQ allows single message manipulation and transaction support.
  - Eg, Update Oracle table, Get a Message, rollback/commit
- MGW is designed to be bulk messages mover
  - Between Oracle AQ and non-Oracle queue systems like MQseries or Tibco
  - MGW not Available on zOS

Leveraging Oracle Gateways within Oracle10*g*AS

Oracle10*g*AS & Oracle10*g* Gateways



Open Dialog


ORACLE®