

This lesson describes the Oracle Net8*i* architecture.

Schedule:	Timing	Topic
	45 minutes	Lecture
	0 minutes	Lab
	45 minutes	Total

## Objectives

- After completing this lesson, you should have an understanding of:
  - Oracle Net8*i* Concepts
  - Connection methods
    - Two task architecture, MTS / Shared Server, Connection Manager
  - Database naming
    - Host naming, Local naming, Name Server

DBA12-2

Copyright © Jeremy Russell &amp; Associates, 2003. All rights reserved.



## Objectives

Net8*i* provides support for Oracle (and non-Oracle) software, across the entire breadth of current hardware platforms, protocols and operating systems. Design goals of Net8*i* include portability, scalability and manageability of the networking components.

Net8*i* offers several different connection methods

- single-task
- two-task
- multi-threaded server
- Oracle Connection Manager

several different database location methods and services

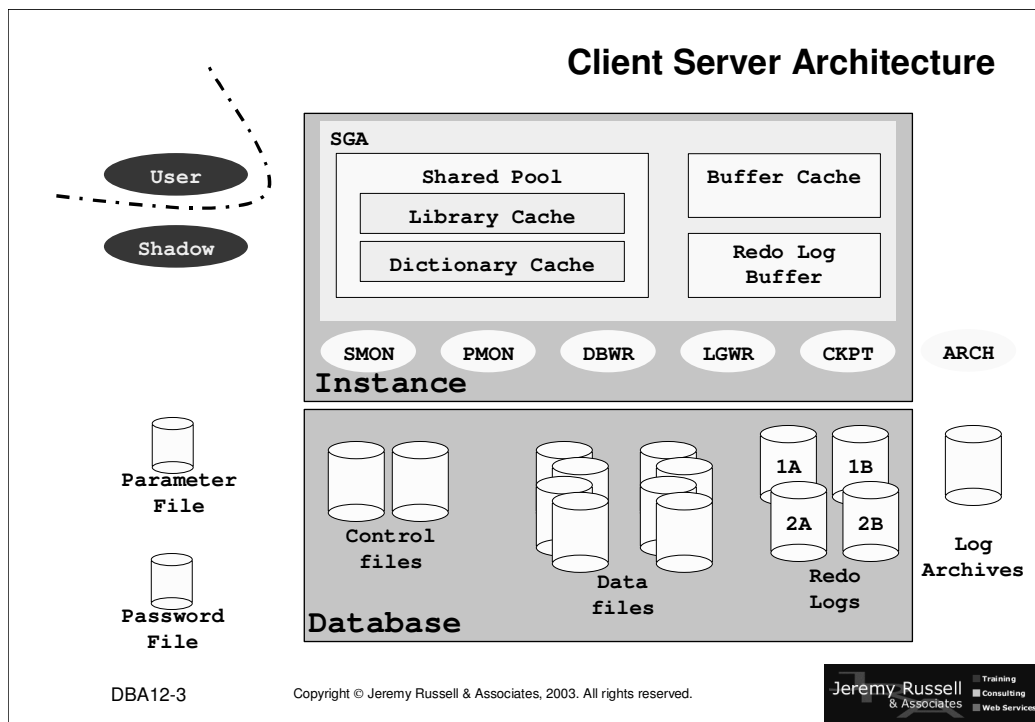
- Host naming
- Local Naming
- Oracle Name Server

and many different protocols, including

- TCP/IP, DecNet, SPX/IPX, LU6.2, HTTP, IIOP, SSL

**Oracle provides graphical network configuration programs, Network Manager and Net8 Assistant, to configure the networking control files. The Oracle documentation does state that they will not support any network that has not been configured with these tools.**

**Despite this, many DBA's configure the files with a text editor. The prudent DBA will ensure that copies of the files are taken before applying changes.**



## Client Server Architecture

Most Oracle database systems will use some form of “client/server” architecture. An exception is the single-task (single process) architecture, supported only on DEC VMS – this operating system provides secure partitioning between client and server components. Other operating systems are not capable of dealing with potential mischievous users.

For non-VMS systems, a **client** process, often running on a PC or workstation, accepts operator commands and passes them (in the form of SQL) to the **server** (shadow) process. The server process always runs on the database host. The intervening communication methodology forms one part of the Net8*i* architecture.

Net8*i* also uses other processes (listener, dispatchers and more) to facilitate the establishment of a connection.

Depending on the networking and database instance configuration, shadow processes may be *pre-spawned* to ensure that the client process achieves the fastest possible connection to the database. The overhead of starting up a new OS shadow process to service the client is minimised.

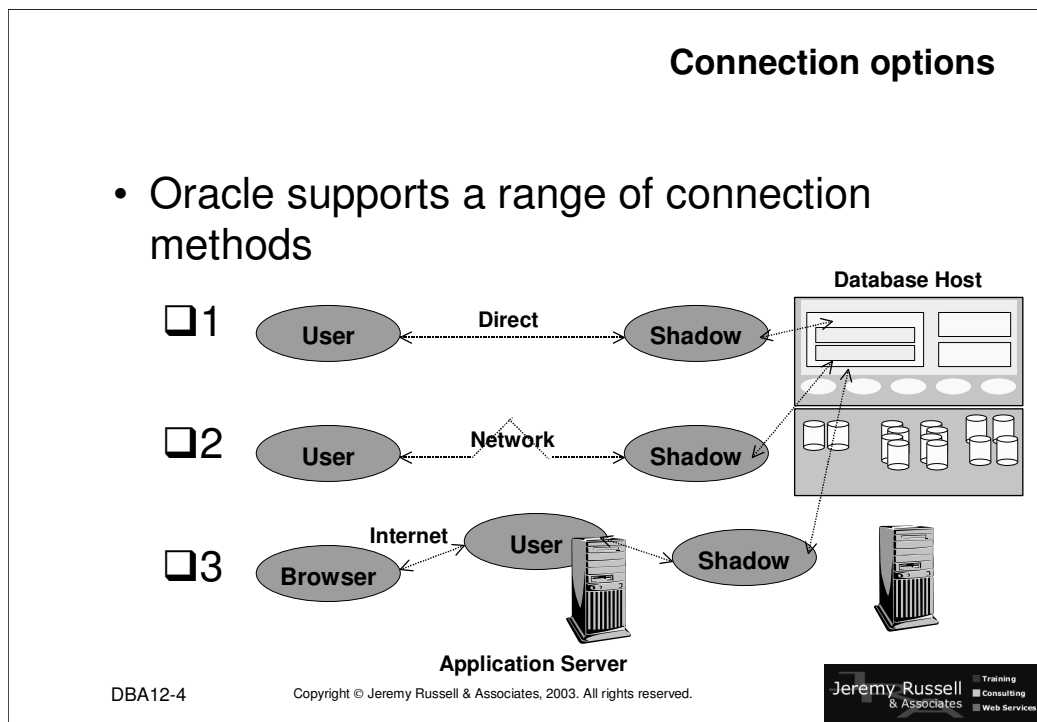
The number of pre-spawned processes can also be managed by the DBA (Network Administrator), providing further tuning opportunities.

## Connect String

The client process connects to the database service required, by specifying a “*connect string*” component of the initial connection command. For example:

```
sqlplus system/manager@dbService
```

In this example, “system” is the user account name, “manager” is the account password and “dbservice” – the **connect string** – is the name of the database service (or host) to receive the connection.



## Connection options

1. A direct connection, typical of a UNIX system with local “dumb” terminals

No client side configuration is required and databases are located using the host network name. This configuration only supports a single database on the host and is intended for use in smaller environments.

2. Client / server connections, with PC’s for presentation and a high capacity, high performance database server to manage data retrieval and updating.

Multiple database instances can be supported if required on the database server. Target connections are defined in configuration files resident on each client (`TNSNAMES.ORA`).

Distributed databases (DBMS to DBMS communication) can also be configured using this method i.e. each database acts as a pseudo-client for the other databases in the network.

3. Internet / Intranet applications, using an intermediate application server as a concentrator and HTML pre-processor.

This level of application may use several machines to act as application servers and communication concentrators to the network. Regionalised setups are possible, allowing many flexible opportunities for load management and configuration.

Oracle Application Server is not covered in this course.

## Net8 Components

- Multiple connection methods
  - Two task
  - Multi-threaded server
  - Connection Manager
- Naming methods
  - Host naming
  - Local naming
  - Oracle names

DBA12-5

Copyright © Jeremy Russell &amp; Associates, 2003. All rights reserved.



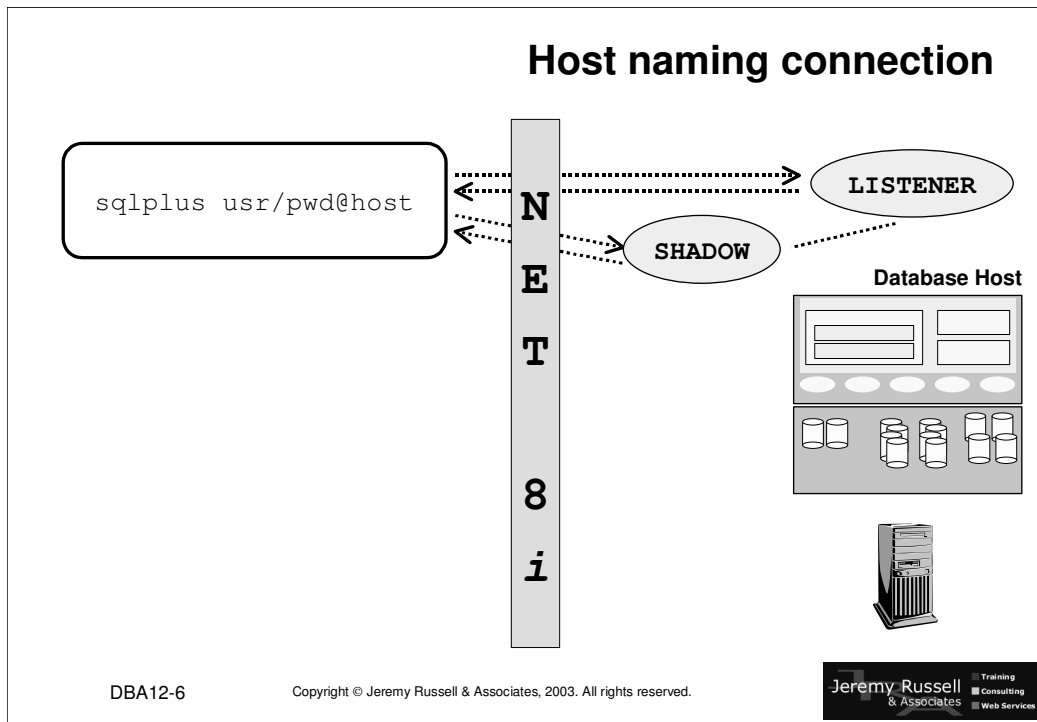
## Net8 Components

### Multiple connection methods

- Two-task                      Common client server connection method.  
Server (shadow) runs on the database host (which can be the same processor as the client). See page 22 in this lesson.
- Multi-threaded server      Shared, pre-spawned servers are generated when the instance starts.  
An additional dispatcher processor (or multiple dispatchers) also started by the instance will route requests to a free server and queue return results for the client.
- Connection Manager        An intermediate network node that routes requests from clients to the relevant database server. Connection Manager is also capable of protocol conversion and simple network access controls

## Naming methods

- **Host naming**                      Simplest level of database identification. The connection string is a network host name; the named host can only support a single networked database instance.
  
- **Local naming**                      Local naming requires a `TNSNAMES` file to reside on each client. If the network configuration changes, all the clients that require access to the new resource must be updated (or have access to a shared `TNSNAMES` file).
  - 📖 *TNS - Transparent Network Substrate - is Oracle's term for the networking layer that interfaces from the client hardware to the server hardware*
  
- **Oracle Names**                      The Name Server is a service that typically runs on one or more (dedicated) servers. Clients are configured to be aware of their nearest Name Server node; the Name Server is consulted by the client to determine the location of the database host. Once the Name Server identifies the host to the client, a connection is then set up using the relevant protocols.

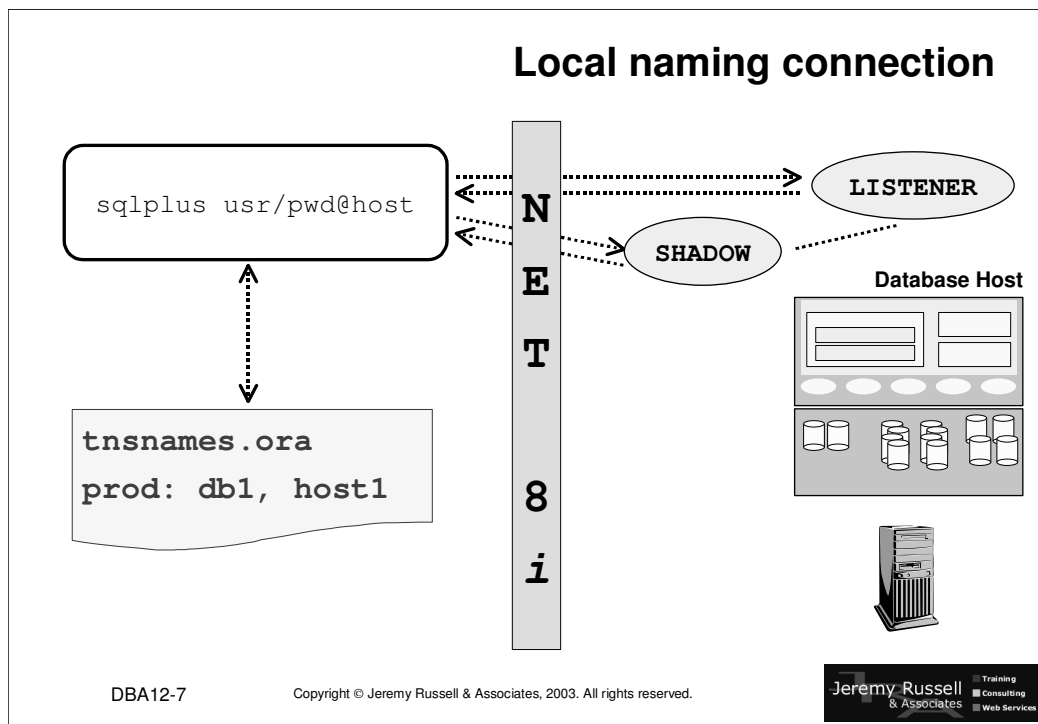


## Host naming connection

The host naming connection method is the simplest means of connecting remotely to a database.

The connect string is the network name of the database host.

The host can accept connections for a single database instance.



## Local naming connection

The local naming convention uses a TNSNAMES.ORA file located on each client in the network.

The TNSNAMES configuration file resides in Oracle's home directory/network/admin. This location can be overridden using the TNS\_ADMIN environment variable.

For each database, an entry is created that identifies:

- the local name for the database
- the database server network name, protocol and protocol specific parameters
- the instance name on the host.

More details on TNSNAMES can be found on page 10 in this lesson.

## SQLNET.ORA

- Resides on client and server
- Contains network setup information
- Optional diagnostic parameters
- Optional Oracle\*Names information

DBA12-8

Copyright © Jeremy Russell &amp; Associates, 2003. All rights reserved.



## SQLNET.ORA

To use Net8*i* to connect clients (or servers) to database hosts, each system in the network should have a `SQLNET.ORA` file. This file configures the network connection methods for the machine and controls logging and tracing of Oracle network operations.

Logging records error information to a log file; tracing records regular operational steps to a trace file.

```
# SQLNET.ORA Network Configuration File:
# $ORACLE_HOME/network/admin/sqlnet.ora

#SQLNET.AUTHENTICATION_SERVICES= (NTS) # Required for Windows systems

TRACE_LEVEL_CLIENT      = OFF          # Levels : USER < ADMIN < SUPPORT
TRACE_UNIQUE_CLIENT     = ON           # Generate unique trace file names
TRACE_FILE_CLIENT       = net8trace    # Client side trace file prefix
TRACE_DIRECTORY_CLIENT  = /oracle/trace # Client side trace file directory

LOG_FILE_CLIENT         = sqlnet.log    # Log file name (default)
LOG_DIRECTORY_CLIENT   = /oracle/log    # Log file directory

NAMES.DIRECTORY_PATH   = (TNSNAMES)    # Naming method(s) : ONAMES, HOSTNAME
```

## TNSNAMES.ORA

- Resides on client systems
- Identifies network services
  - Host name
  - Instance identifier
- Can also reside on distributed servers

DBA12-9

Copyright © Jeremy Russell &amp; Associates, 2003. All rights reserved.



## TNSNAMES.ORA

The TNSNAMES file identifies the services that can be used from the client.

For each service, the following information must be provided:

- Service name and domain name
- Description
- Address
- Protocol
- Host – protocol specific information for connection to the host
- Port for TCP/IP connections, to identify the listener port to be used (defaults to 1521).
- Service name/SID – the database identifier to receive connections

See the sample file on the next page.

**Sample TNSNAMES.ora file**

This file is configured for two separate connections, LIVE (via TCP/IP) and TEST (using the SPX protocol):

```
# TNSNAMES.ORA Network Configuration File:
# $ORACLE_HOME/network/admin/tnsnames.ora

LIVE.world =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS =
        (PROTOCOL = TCP)
        (HOST = DBHOST)
        (PORT = 1521))
      )
    (CONNECT_DATA = (SERVICE_NAME = production))
  )
)

TEST.world =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS =
        (COMMUNITY = spx.world)
        (PROTOCOL = SPX)
        (SERVICE = Server_lsnr)
      )
    (CONNECT_DATA = (SID = TEST))
  )
)
```

## Listener

- Process started separately on server
- Waits for incoming connection requests
- Spawns server (shadow) process
- Address of shadow returned to client
- Client communicates with shadow

DBA12-10

Copyright © Jeremy Russell &amp; Associates, 2003. All rights reserved.



## Listener

The listener process runs on the database host. The process is started and managed by the listener control utility (`LSNRCTL`). Several operational methods can be employed by the listener to establish a client/server connection.

### Bequeath session

The listener waits for incoming connection requests on the configured network protocol (and port). When a request is received, the listener will spawn a server (shadow) process. The session connection information is bequeathed to the newly spawned server process. Further communication between the client and the database instance does not require the listener to be active.

### Redirect session (dedicated)

As the listener starts, a number of shadow processes are spawned (as configured in the `LISTENER.ORA` configuration file). The listener waits for incoming connection requests on the configured network protocol (and port). When a request is received, the listener redirects the session to one of the existing server processes. Further communication between the client and the database instance does not require the listener to be active.

Additional resources are required to create and maintain the pre-spawned servers but connection time improves by removing the overhead of starting a new process for each connection.

### Redirect session (dispatcher)

For a multi-threaded server environment, a dispatcher process (started by the database instance) receives the incoming request from the listener. The least loaded dispatcher is selected automatically. More details regarding the multi-threaded server environment are discussed later in this lesson.

## LISTENER.ORA

- Located on database host
- Controls listener configuration
- Specifies databases to accept incoming connections

DBA12-11

Copyright © Jeremy Russell &amp; Associates, 2003. All rights reserved.



## LISTENER.ORA

The LISTENER.ORA file configures the listener on a database host. The following information must be specified:

- Listener name Defaults to LISTENER.
- Listener address The list of addresses (protocols, host names and ports) that are using this listener.  
  
For IPC addresses, client processes run on the same host server. The KEY name is the database service or SID name.  
  
For TCP addresses, the host name and port number (default 1521) must match the client's TNSNAMES.ORA settings.
- Databases using the listener The SID\_LIST\_LISTENER block defines the databases for which requests are accepted by this listener.  
  
For each database, the corresponding ORACLE\_HOME directory and the System Identifier (SID) of the database must be specified.
- Additional parameters See page 15 for more details.

**Sample LISTENER.ora file**

```
# LISTENER.ORA Network Configuration File
# $ORACLE_HOME/network/admin/listener.ora
# Generated by Oracle configuration tools.

LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = IPC) (KEY = EXTPROC0))
      )
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP) (HOST = dbhost) (PORT = 1521))
      )
    )
  )

SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (GLOBAL_DBNAME = live)
      (ORACLE_HOME = /oracle81)
      (SID_NAME = prod)
    )
    (SID_DESC =
      (GLOBAL_DBNAME = test)
      (ORACLE_HOME = /oracle81)
      (SID_NAME = test)
    )
  )

TRACE_LEVEL_LISTENER = OFF
TRACE_DIRECTORY_LISTENER = /oracle/trace
TRACEFILE_LISTENER = listener1.trc
```

### Additional LISTENER.ORA parameters

- `LOG_DIRECTORY_listener_name`      Directory name for listener log
- `LOG_FILE_listener_name`            File name for listener log
- `LOGGING_listener_name`            Listener logging level – OFF or ON.
- `TRACE_DIRECTORY_listener_name`    Directory name for listener tracing
- `TRACE_FILE_listener_name`         File name for listener tracing
- `TRACE_LEVEL_listener_name`        Listener tracing level – OFF, USER, ADMIN, SUPPORT

Note – this is not a full list of parameters. Please consult the Oracle 8i Server Networking Guide.

## Listener Control - LSNRCTL

- Control program for listener service
- Starts and stops the listener process
- Other commands
  - SET trc\_level
  - SET trc\_file
- Actions are logged in LISTENER.LOG

DBA12-12

Copyright © Jeremy Russell &amp; Associates, 2003. All rights reserved.



### Listener Control – LSNRCTL

The listener program is started, managed and stopped using the listener control utility, LSNRCTL.

The utility operates either interactively or uses command line parameters.

## Starting the listener

The 'START' command will start the named listener (LISTENER by default). This example shows the command line mode of `lsnrctl`.

```
$ lsnrctl start

LSNRCTL for xxxxx 8.1.1.1.1
Production on 20-JAN-2003 10:04:40

Copyright (c) 1991, 2001, Oracle Corporation. All rights reserved.

Starting tnslnsr: please wait...

TNSLSNR for 32-bit Windows: Version 9.0.1.1.1 - Production
System parameter file is /oracle/network/admin/listener.ora
Log messages written to /oracle/log/listener.log
Trace information written to /oracle/trace/listener.trc

Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp) (HOST=dbhost) (PORT=1521)))

STATUS of the LISTENER
-----
Alias                LISTENER
Version              TNSLSNR for xxxx: Version 8.1.1.1.1 - Production
Start Date           20-JAN-2003 12:00:00
Uptime               0 days 0 hr. 0 min. 2 sec
Trace Level          user
Security             OFF
SNMP                 OFF
Listener Parameter File /oracle/network/admin/listener.ora
Listener Log File    /oracle/log/listener.log
Listener Trace File  /oracle/trace/listener.trc
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp) (HOST=dbhost) (PORT=1521)))
Services Summary...
Service "live" has 1 instance(s).
  Instance "prod", status UNKNOWN, has 1 handler(s) for this service...
The command completed successfully

$ _
```

## Stopping the listener

The 'Stop' command will start the named listener (`LISTENER` by default). This example shows the interactive mode of `LSNRCTL`.

```
$ lsnrctl
LSNRCTL for xxxxx: Version 8.1.1.1.1 - Production on 20-JAN-2003 12:01:00
Copyright (c) 1991, 2001, Oracle Corporation. All rights reserved.
Welcome to LSNRCTL, type "help" for information.
LSNRCTL> stop listener
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=EXTPROC)))
The command completed successfully
LSNRCTL> exit
$ _
```

## Listener Management

- Automatic instance registration
- Listener load balancing
- Failover and troubleshooting

DBA12-13

Copyright © Jeremy Russell &amp; Associates, 2003. All rights reserved.



## Listener Management

### Automatic instance registration

A new feature within Oracle 8i provides automatic instance registration, the identification of an instance to the listener to minimise configuration.

The listener.ora file does not therefore require the SID\_LIST\_LISTENER parameter to list the databases served by the listener.

The database parameter file (init.ora) specifies the instance name and service names that the database wishes to use. For example:

```
instance_name = prod
service_names = prod.world
```

### Listener load balancing

When using automatic instance registration and multi-threaded servers, load balancing can also be implemented using multiple listeners.

The client request will choose a listener from the list identified in the client TNSNAMES.ORA file. If the listener cannot be contacted, an attempt is made to contact the next listener in the list (see Failover below).

The successfully contacted listener identifies the least loaded node (by CPU usage). On that node, the least loaded dispatcher process (number of connections) is identified and the connected routed accordingly.

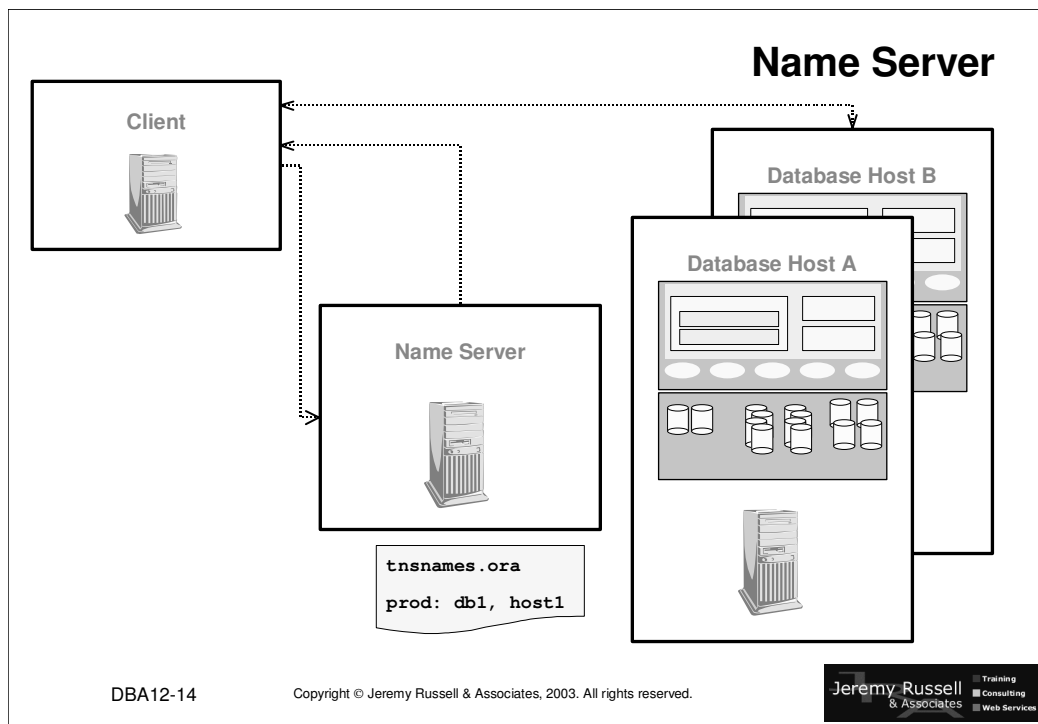
## Failover

By setting the parameters in the TNSNAMES.ORA file as illustrated below, a listener list and automatic failover can be implemented.

```
# TNSNAMES.ORA Network Configuration File:
# $ORACLE_HOME/network/admin/tnsnames.ora

LIVE.world =
  (DESCRIPTION =
    (FAILOVER = on)
    (LOAD_BALANCE = on)
    (ADDRESS_LIST =
      (ADDRESS =
        (PROTOCOL = TCP)
        (HOST = DBHOSTa)
        (PORT = 1521))
      (ADDRESS =
        (PROTOCOL = TCP)
        (HOST = DBHOSTb)
        (PORT = 1521))
    )
    (CONNECT_DATA = (SERVICE_NAME = production)
  )
)

TEST.world =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS =
        (COMMUNITY = spx.world)
        (PROTOCOL = SPX)
        (SERVICE = Server_1snr)
      )
    )
    (CONNECT_DATA = (SID = TEST)
  )
)
```



## Name Server

The Oracle Name Server is a centralised service, running on a dedicated system, that offers the following benefits over local naming:

- Easier administration
  - Increased Efficiency
  - Removes redundancy
  - Location transparency
- Database hosts and service names are defined once only. Changing the database networking configuration does not require updating each client's `TNSNAMES.ORA` file. New addresses are immediately accessible to all clients.

Use of Oracle Name Server is recommended when the network continually changes structure and size.

Multiple Oracle Name Servers can be configured on a network. The Name Servers can be regionalised for local use and can be configured to automatically exchange information between themselves.

The client `SQLNET.ORA` includes a `NAMES.PREFERRED_SERVER` entry to identify the nearest Name Server, which communicates on TCP port 1621. The `NAMES.DIRECTORY_PATH` entry in the `SQLNET.ORA` file should be set to `ONAMES` to use Oracle Name Server for database address resolution.

```
...
NAMES.PREFERRED_SERVERS =
  (ADDRESS_LIST =
    (ADDRESS =
      (PROTOCOL = TCP)
      (HOST = ONAMES1)
      (PORT = 1621)
    )
  )
NAMES.DIRECTORY_PATH = (ONAMES)
```

## Two-task architecture

- Most common connection method (client / server)
- Server (shadow) runs on database host
- Bequeath or redirect session
- TCP/IP, IPC or other protocols

DBA12-15

Copyright © Jeremy Russell &amp; Associates, 2003. All rights reserved.



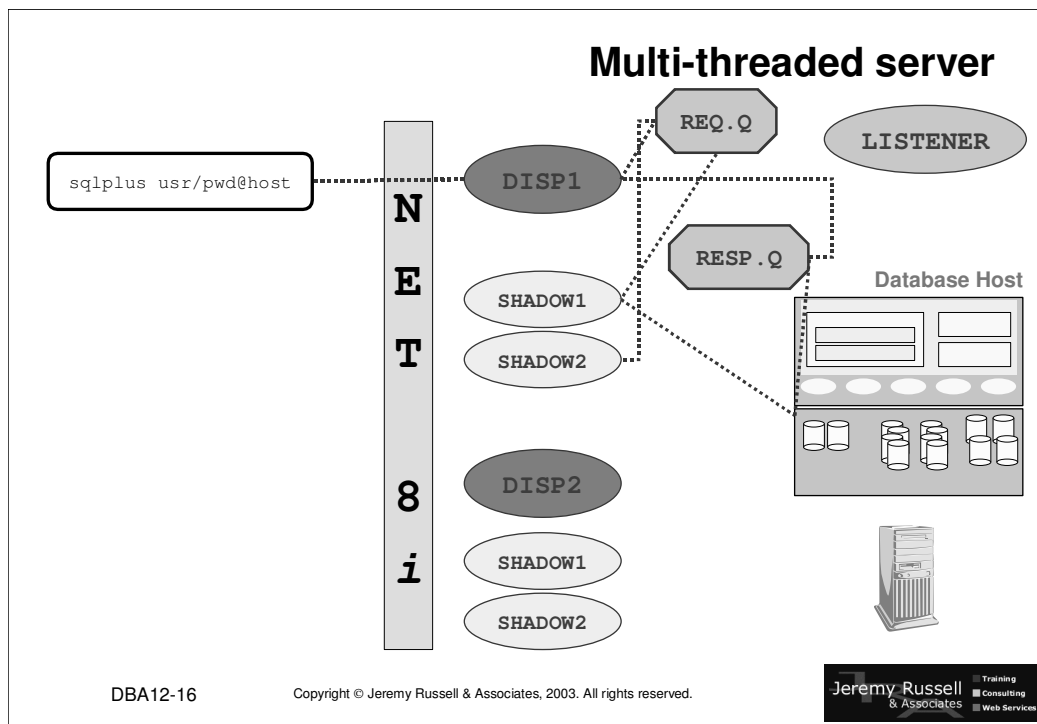
## Two-task architecture

The listener waits for incoming connection requests on the configured network protocol (and port). When a request is received, the listener will spawn a server (shadow) process. The session connection information is bequeathed to the newly spawned server process. Further communication between the client and the database instance does not require the listener to be active.

The separation of user and server processes, even on the same system, provides a level of additional security. For each user process, there is a dedicated server process started. Even when the user is not making requests, the server process remains in memory and uses resources.

Using the two-task architecture allows the user to have immediate access through their dedicated server process to the database host.

Some operating system overhead is used in managing a large number of (potentially) idle processes.



## Multi-threaded server

The Multi Threaded Server (MTS) option overcomes some of the operating system load in managing server processes for each user, by sharing a smaller number of processes amongst multiple user processes.

For OLTP applications, MTS may be more efficient, since users are not communicating with the server continuously. More users may be able to supported (effectively) within a specified server hardware architecture than with the two-task architecture.

## Configuring for MTS

A database instance must be configured to use MTS, by setting the init.ora parameters below:

- `MTS_DISPATCHERS`      Number of dispatchers to start when the instance starts, for a specified protocol.
- `MTS_MAX_DISPATCHERS`    Maximum number of dispatchers allowed on this instance.
- `MTS_SERVERS`            Number of servers to start when the instance starts.
- `MTS_MAX_SERVERS`        Maximum number of servers allowed on this instance.

```
mts_dispatchers = "(PROTOCOL=TCP) (DISPATCHERS=3) "
mts_max_dispatchers=10
mts_servers = 8
mts_max_servers = 20
```

The `MTS_DISPATCHERS` and `MTS_SERVERS` parameters are dynamic and can be altered by the DBA for load balancing:

```
ALTER SYSTEM SET MTS_SERVERS = 12
```

### Making a connection and processing requests

A connection is made using the listener in the standard way – the listener then routes the connection to a DISPATCHER process for the instance. This selected dispatcher then controls the connection for the duration of the user session.

The dispatcher adds subsequent requests to a single dispatcher request queue. The request will be dequeued by *any* available server for processing. The results are placed onto a response queue for the appropriate dispatcher (one queue is maintained for each dispatcher). When the dispatcher finishes it's current processing, it will check it's queue and process any pending responses by routing them back to the relevant client.

### Tuning the MTS

The DBA can monitor the length of request and response queues, by examining the relevant MTS tables (V\$DISPATCHER and V\$SHARED\_SERVER). If these table indicate, the DBA can either add or stop dispatcher and server processes to improve response times or reduce the OS load on the system.

## Connection Manager

- Connection Manager provides
  - Concentrator
  - Protocol converter
  - Network access control
- Configured by CMAN.ORA
- Managed by CMCTL
- Requires multi-threaded server

DBA12-17

Copyright © Jeremy Russell &amp; Associates, 2003. All rights reserved.



## Connection Manager

The Oracle Connection Manager is a service that provides the following features

- Scalable use for MTS systems
- Access Control
- Protocol Conversion

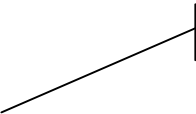
The client TNSNAMES.ORA (for local naming) or the Name Server file (for Oracle Naming) is configured for using Connection Manager.

On the Connection Manager server, a CMAN.ORA file can be configured to manage access control.


## TNSNAMES.ORA

```
# TNSNAMES.ORA Network Configuration File:
# $ORACLE_HOME/network/admin/tnsnames.ora

LIVE.world =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS =
        (PROTOCOL = TCP)
        (HOST = CMANHOST)
        (PORT = 1610))
      (ADDRESS =
        (PROTOCOL = TCP)
        (HOST = DBHOST)
        (PORT = 1521))
    )
    (CONNECT_DATA = (SERVICE_NAME = production)
      (SOURCE_ROUTE = YES)
    )
  )
)
```



From client to connection manager



From connection manager to database

## INIT.ORA

To use Connection Manager, the database instance must be properly configured:

```
mts_dispatchers = "(PROTOCOL=TCP) (DISPATCHERS=3) (MULTIPLE=ON) "
```

## CMAN.ORA

The CMAN.ORA file on the Connection Manager host is configured to accept or reject connections between client and server. A sample cman\_rules section is illustrated below.

```
cman_rules = (rules_list =
  (rule = (src = clientPC)
    (dst = dbhost)
    (srv = live)
    (act = accept))
  (rule = (src = 192.168.0.x)
    (dst = dbhost)
    (srv = live)
    (act = reject))
)
```

## Summary

- In this lesson, you learned about:
  - Oracle Net8*i* Concepts
  - Connection methods
    - Two task architecture, MTS / Shared Server, Connection Manager
  - Database naming
    - Local naming, TNSNAMES, Name Server

DBA12-18

Copyright © Jeremy Russell &amp; Associates, 2003. All rights reserved.



**Practice 12 - Questions**

There are no practice questions for this lesson.