LISTENING IN: PASSIVE CAPTURE AND ANALYSIS OF ORACLE NETWORK TRAFFIC

Jonah H. Harris, EnterpriseDB Corporation

OVERVIEW

In this presentation we will discuss the Oracle wire-level protocol and demonstrate the methods for passively capturing, analyzing, and reporting the details of Oracle network traffic in real-time for use in end-to-end Oracle tuning and troubleshooting scenarios.

In cases where very short response time requirements must be met, or where sporadic spikes in response time occur, the most reliable way to tune and troubleshoot them is by capturing Oracle's Ethernet traffic, analyzing it, and reporting on various aspects of it. Throughout this session we will demonstrate the passive capture of SQL statements, their frequency, time spent in execution, number of roundtrips, and all relevant response times.

Using the data from these reports can not only assist DBAs in diagnosing network-related issues and in tuning Oracle's network settings, but also ensure that application developers are writing performant, network-friendly database access code.

INTRODUCTION

This paper introduces the concepts behind Oracle's Network Architecture as well as protocol descriptions, an example wire-level application, and an introduction to the SCAPE4O network monitoring utility.

As I've never been an Oracle insider, the material in this paper has been based on years of researching Oracle internals as well as analyzing network traffic and trace files. Likewise, in addition to similar research from Ian Redfern, the majority of this paper is based primarily on my own personal research and discussions with Tanel Põder; without their insight, this would've taken me significantly longer to rationalize.

THE ORACLE NETWORK ARCHITECTURE

Like several other databases, the Oracle network architecture is based on the Open Systems Interconnection (OSI) Basic Reference Model. The OSI model is a layered, abstract communications and computer network protocol architecture which consists of communications between separate systems being performed in a stack-like fashion; information passing from node-to-node through several distinct layers of code.

THE OSI MODEL

The OSI layers consist of the following:

1. Physical Layer
2. Data Link Layer
3. Network Layer
4. Transport Layer
5. Session Layer
6. Presentation Layer
7. Application Layer

The OSI Physical Layer (Layer 1)

The physical layer defines all the electrical and physical specifications for devices. In particular, it defines the relationship between a device and a physical medium. This includes the layout of pins, voltages, cable specifications, Hubs, repeaters, network adapters, Host Bus Adapters (HBAs used in Storage Area Networks) and more.

To understand the function of the physical layer in contrast to the functions of the data link layer, think of the physical layer as concerned primarily with the interaction of a single device with a medium, where the data link layer is concerned more with the interactions of multiple devices (i.e., at least two) with a shared medium. The physical layer will tell one device how to transmit to the medium, and another device how to receive from it (in most cases it does not tell the device how to connect to the medium). Obsolescent physical layer standards such as RS-232 do use physical wires to control access to the medium.

The OSI Data Link Layer (Layer 2)

The data link layer provides the functional and procedural means to transfer data between network entities and to detect and possibly correct errors that may occur in the physical layer. Originally, this layer was intended for point-to-point and point-to-multipoint media, characteristic of wide area media in the telephone system. Local area network architecture, which included broadcast-capable multiaccess media, was developed independently of the ISO work, in IEEE Project 802. IEEE work assumed sublayering and management functions not required for WAN use. In modern practice, only error detection, not flow control using sliding window, is present in modern data link protocols such as Point-to-Point Protocol (PPP), and, on local area networks, the IEEE 802.2 LLC layer is not used for most protocols on Ethernet, and, on other local area networks, its flow control and acknowledgment mechanisms are rarely used. Sliding window flow control and acknowledgment is used at the transport layers by protocols such as TCP, but is still used in niches where X.25 offers performance advantages.

The OSI Network Layer (Layer 3)

The network layer provides the functional and procedural means of transferring variable length data sequences from a source to a destination via one or more networks while maintaining the quality of service requested by the Transport layer. The Network layer performs network routing functions, and might also perform fragmentation and reassembly, and report delivery errors. Routers operate at this layer—sending data throughout the extended network and making the Internet possible. This is a logical addressing scheme – values are chosen by the network engineer. The addressing scheme is hierarchical.

The OSI Transport Layer (Layer 4)

The transport layer provides transparent transfer of data between end users, providing reliable data transfer services to the upper layers. The transport layer controls the reliability of a given link through flow control, segmentation/desegmentation, and error control. Some protocols are state and connection oriented. This means that the transport layer can keep track of the segments and retransmit those that fail.

The OSI Session Layer (Layer 5)

The session layer controls the dialogues/connections (sessions) between computers. It establishes, manages and terminates the connections between the local and remote application. It provides for full-duplex, half-duplex, or simplex operation, and establishes checkpointing, adjournment, termination, and restart procedures. The OSI model made this layer responsible for "graceful close" of sessions, which is a property of TCP, and also for session checkpointing and recovery, which is not usually used in the Internet protocols suite. Session layers are commonly used in application environments that make use of remote procedure calls (RPCs).

The OSI Presentation Layer (Layer 6)

The presentation layer establishes a context between application layer entities, in which the higher-layer entities can use different syntax and semantics, as long as the Presentation Service understands both and the mapping between them. The presentation service data units are then encapsulated into Session Protocol Data Units, and moved down the stack.

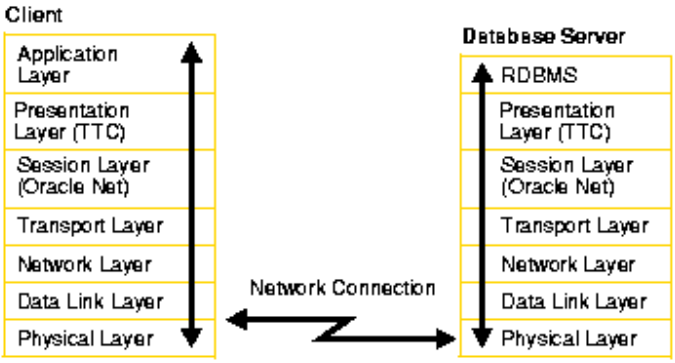
The OSI Application Layer (Layer 7)

The application layer interfaces directly to and performs application services for the application processes; it also issues requests to the presentation layer. Note carefully that this layer provides services to user-defined application processes, and

not to the end user. For example, it defines a file transfer protocol, but the end user must go through an application process to invoke file transfer. The OSI model does not include human interfaces. The common application services sublayer provides functional elements including the Remote Operations Service Element (comparable to Internet Remote Procedure Call), Association Control, and Transaction Processing (according to the ACID requirements).

Mapping OSI Layers to Oracle

Oracle Net Services starts at the OSI Session Layer.



Oracle to OSI Mapping

Oracle Protocol Support (Layer 5)

While Oracle Protocol Support sounds like it would map to Layer 4, it does not actually provide the network transport stack; making it Layer 5. Oracle Protocol Support is designed to map Transparent Network Substrate to industry-standard transport protocols using an existing protocol stack. More about this layer can be found below under Network Transport (NT).

Oracle Net Foundation Layer (Layer 5)

The Oracle Net foundation layer is responsible for establishing and maintaining the connection between the client application and database server, as well as exchanging messages between them. The Oracle Net foundation layer is able to perform these tasks because of a technology called Transparent Network Substrate (TNS). TNS provides a single, common interface functioning over all industry-standard protocols. In other words, TNS enables peer-to-peer application connectivity. In a peer-to-peer architecture, two or more computers (called nodes when they are employed in a networking environment) can communicate with each other directly, without the need for any intermediary devices.

Two Task Common Layer (Layer 6)

Character set differences can occur if the client and database server are running on different operating systems. The presentation layer resolves any differences. It is optimized for each connection to perform conversion only when required.

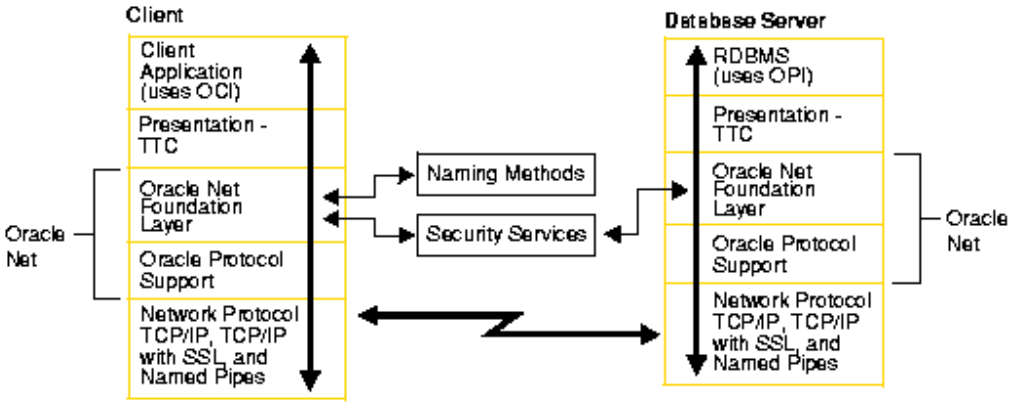
The presentation layer used by client/server applications is Two-Task Common (TTC). TTC provides character set and data type conversion between different character sets or formats on the client and database server.

At the time of initial connection, TTC is responsible for evaluating differences in internal data and character set representations and determining whether conversions are required for the two computers to communicate.

Application and RDBMS Layer (Layer 7)

Information passed from a client application across a network protocol is received by a similar communications stack on the database server side. The process flow on the database server side is the reverse of the process flow on the client side, with information ascending through the communication layers.

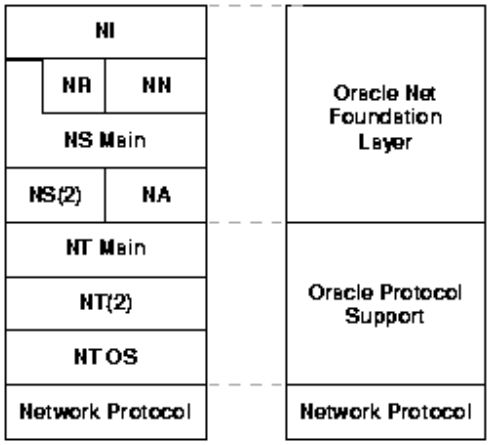
Instead of OCI, the database server uses Oracle Program Interface (OPI). For each statement sent from OCI, OPI provides a response. For example, an OCI request to fetch 25 rows would elicit an OPI response to return the 25 rows once they have been fetched.



Mapping Example

ORACLE NET COMPONENTS

The Oracle Net implementation stack is comprised of the following components:



Oracle Network Layer Components

NETWORK INTERFACE (NI)

This layer provides a generic interface for Oracle clients, servers, or external processes to access Oracle Net functions. The NI layer handles the "break" and "reset" requests for a connection.

NETWORK ROUTING (NR)

This layer routes the network session to the destination.

NETWORK NAMING (NN)

This layer resolves connect identifiers to connect descriptors.

NETWORK SESSION (NS)

The NS layer, which concists of the NR and NA layers, receives requests from NI, and settles all generic computer-level connectivity issues, such as: the location of the server or destination (open, close functions); whether one or more protocols will be involved in the connection (open, close functions); and how to handle interrupts between client and server based on the capabilities of each (send, receive functions).

NETWORK AUTHENTICATION (NA)

This layer negotiates authentication and encryption requirements.

NETWORK TRANSPORT (NT)

This layer maps the Oracle Net Foundation Layer functionality to industry-standard protocols.

THE ORACLE NETWORK PROTOCOL

The Oracle network protocol is comprised of the following components:

TRANSPARENT NETWORK SUBSTRATE (TNS)

A foundation technology, built into the Oracle Net foundation layer that works with any standard network transport protocol.

TWO TASK INTERFACE (TTI)

Encapsulated by TNS are Oracle's Two Task Interface sub-packets. TTI is the network-level interface to Oracle Database functionality.

TWO TASK COMMON (TTC)

A presentation layer type that is used in a typical Oracle Net connection to provide character set and data type conversion between different character sets or formats on the client and server.

ORACLE PROGRAMMATIC INTERFACE (OPI)

The Oracle Program Interface (OPI) is a server-side networking layer responsible for responding to each of the possible messages sent by the client interface. Interaction with OPI is handled through TTI functions.

TNS PACKET OVERVIEW

TNS consists of several packets which are described in the following sections.

TNS PACKET TYPES

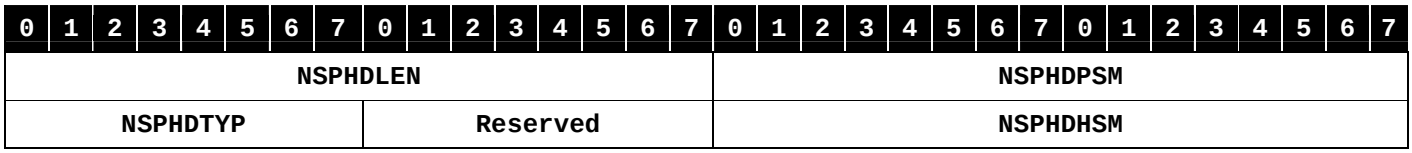
TNS packets have distinct types.

- Connection Packet—The initial request packet used to connect to a database.
- Accept Packet—A response packet from the server accepting the connection.
- Acknowledge Packet—An acknowledgement packet.
- Refuse Packet—A response packet from the server refusing connection.
- Redirect Packet—A response packet from the server redirecting the client to connect to another host/port.
- Data Packet—The most commonly used packet which encapsulates
- NULL Packet—An empty packet generally used as a keepalive.
- Abort Packet—An abort packet.
- Resend Packet—A request for resend packet.
- Marker Packet—A packet used to indicate that multiple packets were required to transmit a single message.
- Attention Packet—A special type of marker packet.
- Control Packet—A packet used to send control (trace) information.

TNS PACKET HEADER (NSPHD)

All Oracle network packets are encapsulated by a TNS packet. The header of the TNS packet declares its type and information required to access the sub-packet (if any).

1	2	3	4



Offset	Bytes	Type	Name	Description
00	2	UB2	NSPHDLEN	Packet length - number of bytes in the entire packet.
02	2	UB2	NSPHDPSM	Packet checksum - the 16-bit ones complement of the 16-bit ones complement sum of the entire packet.
04	1	UB1	NSPHDTYP	Packet type (see below)
05	1	UB1	RESERVED	
06	2	UB2	NSPHDHSM	Header checksum - the 16-bit ones complement of the 16-bit ones complement sum of the packet header.

Two Task Interface Packet Overview

Encapsulated by a standard TNS data packet, there are several TTI sub-packets which are often used.

TTI Protocol Negotiation

The TTI protocol version sub-packet informs the server of the protocol versions it is compatible with, and requests similar information back from the server.

TTI Data Types

The TTI data type sub-packet informs the server of the character set and data type representations it is using, and requests similar information back from the server.

TTI Version

This function requests a textual representation of the server version information; the result of which, is the text from V$VERSION.

TTI Function Call

TTI function call sub-packets are commonly used to request data from the server.

Oracle Network Tracing & Analysis Methods

The following methods can be used to analyze, monitor, trace, and detect Oracle networking issues.

SQLNET.ORA Tracing

Inherently, Oracle provides the ability to trace the client and server Oracle Net stack. The weakness of this method is that, because it is very verbose, and because it is not passive, if it is enabled, it can greatly affect network performance. While not especially good at network monitoring, it is quite good at detecting issues related to network naming.

Generic Network Monitoring

Using utilities such as Wireshark, or operating system utilities like tcpdump, you can passively capture and analyze Oracle Ethernet traffic. The downside here is that these utilities are fairly generalized and are not able to dissect the most important Oracle TTI packets.

Oracle-Specific Network Monitoring

Using utilities designed specifically to monitor and analyze Oracle network traffic provides the best and most detailed data. The downside in this case is that, because Oracle's network protocol is proprietary, very few people have spent the time researching it enough to write useful utilities. As such, there are very few of these utilities available.

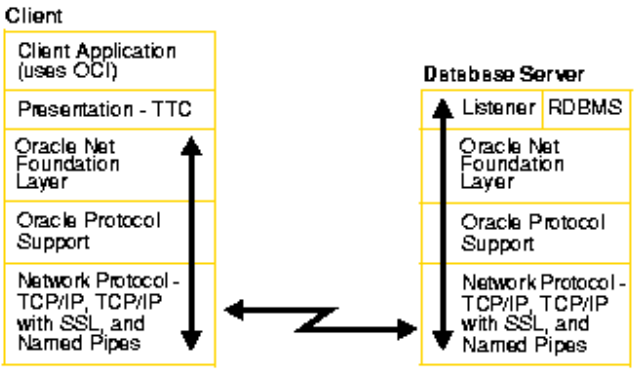
ORACLE NETWORK CONVERSATIONS

The following are basic examples of Oracle network conversations.

CONNECTION CONVERSATION

The connection process consists of:

- Client requests an OCI connection to TNS entry ORCL.
- Network Naming finds ORCL in TNSNAMES.ORA.
- Client builds and sends a TNS Connect Packet (NSPTCN) to the listener.
- The Listener responds with a TNS Resend (NSPTRS) or Redirect [to another port] (NSPTRD) packet.
- Client acts accordingly.
- Server responds with an Accept (NSPTAC) or Refuse (NSPTRF) packet
- Client requests additional services (ANO)
  - Authentication
  - Encryption
  - Data Integrity
  - Supervisor



AUTHENTICATION CONVERSATION

After connection, the client requests authentication from the server using the following process:

- Client & server negotiate protocol version
- Client & server negotiate data types
- Client sends server basic information
  - User Name
  - Terminal Name
  - Machine Name
  - Program Name
  - …
- Server responds with challenge/response…

QUERY & FETCH CONVERSATION

After being authenticated, the client (generally) requests data as follows:

- Open a cursor
- Parse the query
- Execute the query
- Fetch the data
- Cancel the cursor
- Close the cursor

SCAPE4O

SCAPE4O, SQL Capture and Analysis by Passive Evaluation for Oracle, is a utility which passively captures Oracle TCP/IP packets and provides the user with a detailed analysis of Oracle connections, statistics, query activity, and relevant response times.

ARCHITECTURE

SCAPE4O is a multi-threaded application based on libpcap which can capture and analyze data directly from the network or from a stored packet capture.

ANALYSIS

In addition to dissecting TNS, SCAPE4O is able to collect the following data for each SQL query found over the wire:

- Top 10 Queries (By Time/Transfers/etc)
  - Counters for
    - INSERT
    - UPDATE
    - DELETE
    - SELECT
    - COMMIT
    - ROLLBACK
    - PL/SQL (Anonymous Blocks)
    - DDL
  - Response Time

ADVICE

In addition to helping identify network-related issues, SCAPE4O can recommend whether a more optimal SDU could be set or whether the application fetching method could be improved.

REFERENCE MATERIAL

- OSI Text—Significant portions of the OSI model description was taken from Wikipedia under the terms of the GNU Free Documentation License.
- TNS Packet Structures—The TNS packet header and structure definitions can be found in Note:1007807.6, SQL*NET PACKET STRUCTURE: NS PACKET HEADER.

SIMILAR ORACLE-SPECIFIC NETWORK MONITORING UTILITIES

The following are utilities I know of which are similar to SCAPE4O:

- WireCache (http://www.wirecache.com/)— Transparent Database Accelerator and SQL Query Analyzer

GENERIC NETWORK PROTOCOL ANALYZERS

The following are good generic protocol analyzers:

- Wireshark (http://www.wireshark.org/)—Transparent Database Accelerator and SQL Query Analyzer
- Microsoft Network Monitor (http://support.microsoft.com/kb/933741/en-us)—Transparent Database Accelerator and SQL Query Analyzer

OTHER ORACLE WIRE-LEVEL SOFTWARE

If you're looking for other software which uses Oracle's wire-level protocol directly, the following is a list of ones I'm aware of:

- DataDirect (http://www.datadirect.com/)—Well-known wire-level ODBC, JDBC, .NET drivers.
- CoreLab (http://www.crlab.com/)—Wire-level OraDirect .NET driver and Oracle Class Library for C++.
- OraCmd (http://www.withdata.com/)—A Windows-only alternative to SQL*Plus written in Delphi by Shiji Pan.

APPENDIX A: SOURCE CODE EXCERPTS

```
/* ------------------------------------------------------------------------ */

/* -------------------- Network Substrate Packet Types --------------------- */

/*
 * Network Substrate Packet Header
 */
struct nsphd
{
    ub2     nsphdlen;                               /* Packet Length (in bytes) */
    ub2     nsphdpsm;                                       /* Packet Checksum */
    ub1     nsphdtyp;                                          /* Packet Type */
    ub1     nsphdrsv;                               /* Reserved for Future Use? */
    ub2     nsphdhsm;                                 /* Packet Header Checksum */
};
typedef struct nsphd nsphd;

/*
 * Network Substrate Connection Packet
 */
struct nspcn
{
    ub2     nspcnvsn;                                        /* Packet Version */
    ub2     nspcnlov;                                /* Lowest Compatible Version */
    ub2     nspcnopt;                           /* Supports Global Service Options */
    ub2     nspcnsdu;                         /* Session Data Unit Size (in bytes) */
    ub2     nspcntdu;                       /* Transport Data Unit Size (in bytes) */
    ub2     nspcnntc;                             /* NT Protocol Characteristics */
    ub2     nspcntna;                                /* Line Turnaround Value */
    ub2     nspcnone;                          /* The number 1 in Host Byte Order */
    ub2     nspcnlen;                        /* Length of Connect Data (in bytes) */
    ub2     nspcnoff;                             /* Byte Offset to Connect Data */
    ub2     nspcnmxc;                                   /* Maximum Connect Data */
    ub2     nspcnfl0;                                        /* Connect Flags 0 */
    ub2     nspcnfl1;                                        /* Connect Flags 1 */
    ub2     nspcndat;                                           /* Connect Data */
};
typedef struct nspcn nspcn;
```