



# ORACLE<sup>®</sup> Middleware Layer Net8<sup>™</sup> Performance Tuning Utilizing Underlying Network Protocol

---

*Gamini Bulumulle, Sun Professional Services*

*Sun BluePrints<sup>™</sup> OnLine—October 2002*



**<http://www.sun.com/blueprints>**

**Sun Microsystems, Inc.**  
4150 Network Circle  
Santa Clara, CA 95045 U.S.A.  
650 960-1300

Part No. 817-0370-11  
October 2002, Revision A  
Edition: October 2002

Copyright 2003 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved.

Sun Microsystems, Inc. has intellectual property rights relating to technology that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at <http://www.sun.com/patents> and one or more additional patents or pending patent applications in the U.S. and in other countries.

This document and the product to which it pertains are distributed under licenses restricting their use, copying, distribution, and decompilation. No part of the product or of this document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any.

Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and in other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, Sun BluePrints, SunDocs, and Solaris are trademarks, registered trademarks or service marks of Sun Microsystems, Inc. in the United States and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the US and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and in other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

ORACLE and Net 8 are trademarks or registered trademarks of the Oracle Corporation.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

U.S. Government Rights—Commercial use. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

---

Copyright 2003 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, Californie 95045 Etats-Unis. Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées des systèmes Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque enregistrée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company Ltd.

Sun, Sun Microsystems, le logo Sun, Sun BluePrints, SunDocs, et Solaris sont des marques de fabrique ou des marques déposées, ou marques de service, de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

Oracle et Net 8 sont des marques de fabrique ou des marques déposées de Oracle Corporation aux Etats-Unis et dans d'autres pays.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE. L'APTITUDE DE LA PUBLICATION A REPENDRE A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.



Please



Adobe PostScript

# ORACLE<sup>®</sup> Middleware Layer Net8<sup>™</sup> Performance Tuning Utilizing Underlying Network Protocol

---

This article provides a comprehensive analysis of Net8<sup>™</sup> performance tuning utilizing the underlying network protocols (UNPs). It compares Net8 performance with respect to three UNPs with conclusions and recommendations for tuning based on analytical test results. Secondly, it discusses other factors that impact Net8 performance in addition to the network.

This article covers the following topics:

- The Oracle Stack
- Net8 Performance and Tuning
- Tests
- Recommendations/Standards
- Conclusions
- References

The audience for this article is:

- Professionals who provide performance tuning in a client/server environment
- Network engineers
- Oracle database administrators
- IT architects

Client/server environment performance tuning is an imperative subject. Many publications discuss database and application performance tuning, but none discuss the Oracle middleware layer Net8 performance tuning.

This article discusses performance optimization and tuning of Net8 based on an arbitrary UNP that could be TCP/IP, SPX/IP or DECNet, among others. Net8 performance can be maximized by synchronization with the tunable parameters of the UNP, for example, buffer size. Further, it discusses the configurations of thin and thick JDBC (Java™ Database Configuration) drivers and performance impacts in the client/server environment due to these configurations.

The performance tuning concepts discussed in this article are applicable to network performance tuning too.

Oracle client/server systems can employ Net8 as an interface between the Oracle application software and the UNP. Net8 enables Oracle products to access, modify, share, and store data on heterogeneous computing platforms in a variety of networking environments.

Total Net8 transaction performance can be divided into components of connect time and query time:

*total Net8 transaction time = connect time + query time*

When designing and implementing networks, you can maximize connect time by calibrating the tunable parameters of Net8 and the UNP. Typically query time is affected by the database tuning parameters which are outside the scope of this article. However, this article discusses the database tuning parameters that impact network performance.

---

## The Oracle Stack

The Oracle client/server architecture separates a data processing system into two parts—client and server. The client executes the application software that issues data requests to the server. The server executes the database application software that responds to client requests and controls the database as required.

The performance of a client/server application can be optimized by expediting the connect and query times between the client and server and by reducing network traffic.

When configuring client/server applications, performance is impacted by:

- Configuration parameters of the application, that is, SQL\*Plus or Oracle server
- Net8 parameters
- UNP parameters
- Thin or Thick JDBC drivers

In FIGURE 1 and FIGURE 2 typical Upper Layer Protocols (ULPs) could be TCP/IP, IPX/SPX, or DECNet; Lower Layer Protocols (LLPs) could be Ethernet, Token Ring, or FDDI.

The Oracle client/server model can be mapped into the Open System Interconnection (OSI) reference model. Oracle client applications such as SQL\*Plus or SQL\*Forms, and server applications such as the Oracle relational database management system (RDBMS) are at layer seven of the OSI model; Net8 is at layers five and six; the ULP at layers three and four; the Lower Layer Protocol (LLP) at layer two; and the physical layer at one. In this discussion, the application software is at the top of the stack. Overall application performance is based on the performance of the lower three layers as well as variable external factors such as network traffic.

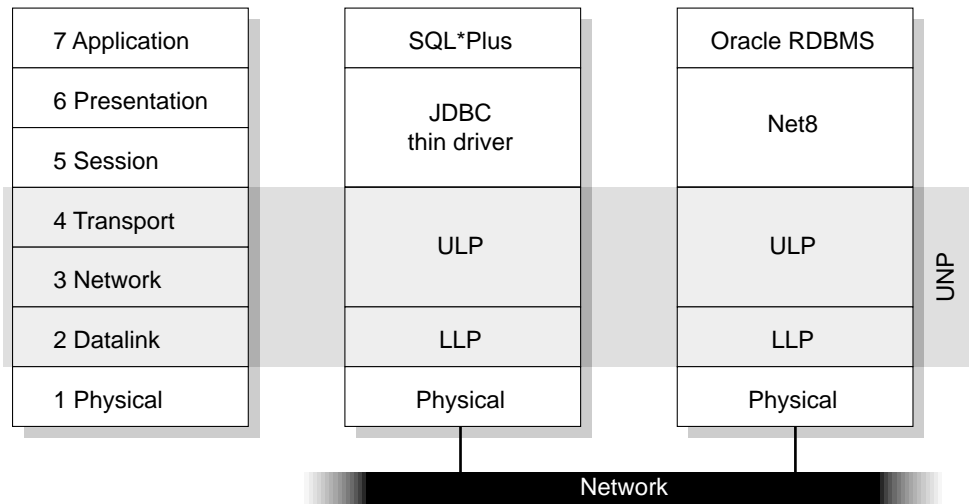
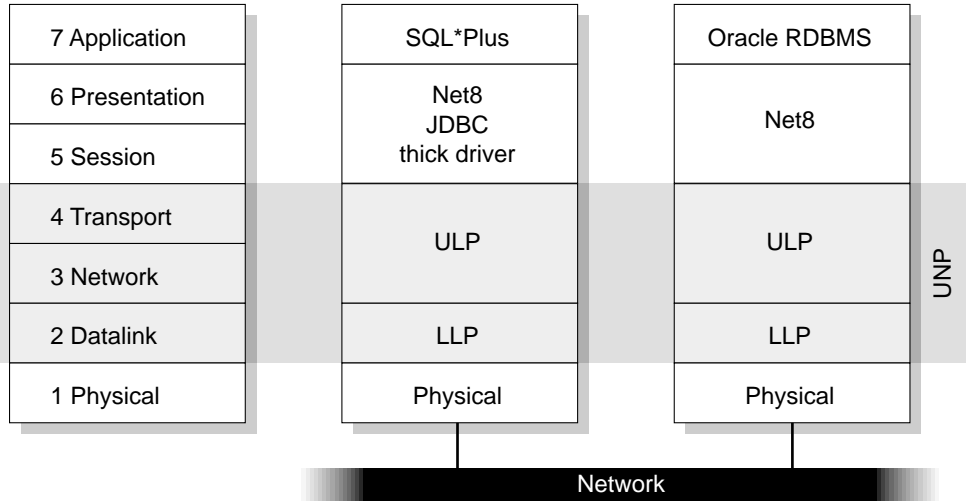


FIGURE 1 Oracle Client/Server Model With Thin JDBC Driver Configuration



**FIGURE 2** Oracle Client/Server Model With Thick JDBC Driver Configuration

The stack paradigm can be applied to Net8 performance, which depends to a great extent on the performance of the lower layers. Therefore, when designing or implementing Oracle client/server systems, taking into consideration the tunable parameters of the underlying layers is vital in order to optimize performance.

## Net8 Performance and Tuning

This section covers the following topics:

- Net8 Performance
- Net8 Tuning

# Net8 Performance

Net8 performance of is based on several factors. This section discusses these factors. Consider the data communication transaction resulting from a simple SQL\*Plus statement:

```
SQL> select * from dual;
D
-
X

SQL>
```

The SQL\*Plus client application initiates a network message as a result of the preceding statement. The message is received by the server, data is retrieved and returned through the network to the client.

Performance can be rated by the difference between the time the client application presents a communication request to the client Net8 ( $t1$ ) to the time the client Net8 returns the response to the client application ( $t2$ ). Referring to FIGURE 1, ( $t2 - t1$ ) is the time required for data to be propagated through client layers 6 through 1, transported across the network medium, propagated through server layers 1 through 6, plus the symmetric return trip.

The time ( $t2 - t1$ ) can be further divided into connect time and query time. Connect time is the round-trip time taken to communicate data between client and server application layers; query time is the time taken by the server to process the data.

Thus,

$$\Delta t = t2 - t1 = \text{connect time} + \text{query time} \quad (1)$$

## Connect Time Factors

Connect time is based on various external factors as well as the statuses of certain Oracle runtime options and helper utilities.

**TABLE 1** Connect Time Factors

External Factors	Oracle Options and Utilities
Use of domain name service	Prespawn processes
Network topology	Multithreaded server (MTS) versus dedicated connections
Network throughput (data rate)	Size of Tnsnames.ora file
Number of hops (bridges and routers) between client and server	Status of Net8 tracing
Network contention, if applicable	Status of security features
Response time	Thin JDBC driver
Heterogeneous network protocols	Thick JDBC driver

## Prespawn Processes

Prespawn dedicated server processes provide a faster connection to the database by eliminating the time required to spawn a process for each connection request.

## MTS Versus Dedicated Connections

The MTS has its own dispatcher. A dedicated environment must create processes. This creation makes it a little slower.

## Size of the Tnsnames.ora File

The Tnsnames.ora file, which is on the client, is significant for applications using Net8. The size of this file can be directly related to connect time. When a client application initiates a transaction to retrieve data from a server, the entire Tnsnames.ora file is read.

Example:

```
$ sqlplus uid/passwd@alias_name
```

The alias name is stored in the `Tnsnames.ora` file. Thus, the size of `Tnsnames.ora` determines a portion of the connect time. Instead of reading the entire file and scanning for the relevant entry, it is better to implement an indexing method.

## Net8 Tracing

If Net8 tracing is turned on, every client/server connection generates a trace file. These files are usually large. The size of the file depends on the level of tracing. Since tracing generates a trace file, it increases the connect time.

## Security Features

Implementation of security features such as encryption/decryption algorithms increase processing time at both ends of each secure transaction.

## JDBC Drivers

There are two types JDBC drivers—Thin and Thick. Employing Thin or Thick drivers in the Oracle client/server stack could impact configuration and performance in the client/server environment.

## Configurations—Thin Versus Thick JDBC Drives

For example, Thin JDBC driver configuration there is no Net8 layer in the client stack (FIGURE 1), where as configuring the Thick driver, the Net8 layer exists in the client stack (FIGURE 2), but in either configuration the Net8 layer exists in the database server. In the case of Thin driver, the socket information is configured in the application itself, where as having the Thick driver, the Net8 layer still exists, therefore the socket configuration is done in the `Tnsname.ora` file. Therefore, in the heterogeneous application environment from the configuration and manageability point of view employing the Thick driver is easier because the only thing to do is populate the `Tnsnames.ora` file to every client. In the Thin driver environment the configuration becomes a bit tedious as Oracle Names are required.

## Performance Considerations—Thin Versus Thick JDBC Drives

**Connect\_Time**—The question is how do these two drivers (Thin and Thick) impact performance other than configuration. Employing the Thin driver, the client/server stack becomes thinner, there is no `Tnsname.ora` file to scan, and the “connect string” is provided on the command line. In the Thick driver environment the stack becomes thicker and parsing is required in the `Tnsname.ora` file. Therefore, compared to the Thick drive `connect_time` could be faster having the Thin drive.

**Query\_Time**—In either case, `Query_Time` should be the same unless the encryption is turned on (Thick driver) using the Advanced Networking Option (ANO).

### Factors Affecting Query Time

Once the connection is made, query time is the amount of time required to retrieve data from the database. Query time is impacted by the following factors:

- Indexing
- Array size

### Indexing

Such factors affect performance at the database level. Since this article focuses on network performance, discussion is limited to array size.

### Array Size

The size of the `array_size` parameter impacts performance. For example, in `SQL*Plus`, the `array_size` parameter is defined by the `SET` command:

```
SQL> set array_size value
```

The `value` parameter determines the number of rows (called a *batch*) that `SQL*Plus` fetches from the database at one time. The `value` parameter can range from 1 to 5000. A large value increases the efficiency of queries that fetch many rows, but requires more host memory.

By calibrating the array size, it is possible to distribute the time required to query the records rather than fetching them all at once, thus decreasing the *perceived* query time. Note that the total time to query the records in smaller groups may be greater than the total time to query the records all at once. Computational overhead to access the database is repeated for each call to the database when the array size is less than the number of records required to be fetched. If the `array_size`

parameter is large, the impact of the overhead is minimal, but additional time is required to retrieve the batch. If the `array_size` parameter is small, the frequency that the overhead impacts the database is greater, but data retrieval time per batch is smaller.

Put another way, when retrieving an arbitrary number of rows, a smaller array size reduces fetch time but increases overhead, whereas larger array size increases fetch time but reduces overhead. Overall, a larger array size produces better results.

Referring to expression (1), there are tradeoffs between connect time and query time. Using a larger array size might optimize query time, at the expense of connect time and overall performance. It is important to determine the optimum batch size, which is a product of array size and row length. Row length, in turn, is a function of the type and amount of data (for example, `VARCHAR2`, `LONG`) in a table.

## SDU Parameter

If the array size is set to a higher value based on row data type, the application passes a large amount of data to Net8. The Net8 buffer size determines the amount of data that can be processed by Net8 (FIGURE 1). The session data unit (SDU) parameter defines the Net8 buffer. For Net8 version 2.3.x and above, the default size of the SDU parameter is 2 kilobytes (configurable up to 32 kilobytes); for versions 2.3 and below, the default SDU is also 2 kilobytes (the maximum configurable size). As an Net8 connection is established, the client and server negotiate the size of the SDU to be used. If the SDUs of the client-side and server-side differ, the smaller of the two is selected. This “decision” is made by the server-side Net8.

If the SDU parameter is smaller than the application fetch size, fragmentation could occur. If SDU is larger than the application fetch size, there is no fragmentation, and the entire packet can be sent across the network (assuming ULP and LLP buffer sizes are large enough to handle it).

Again, the array size is the number of rows that Oracle fetches before it passes them to the server Net8 to be returned to the client. This action affects Net8 packet sizes throughout the communication stream.

### CODE EXAMPLE 1 Syntax SDU in Tnsnames.ora File

```
EOUG=
  (DESCRIPTION=
    (SDU=2048)service layer buffer size
    (TDU=1024)transport layer size
    (ADDRESS=
      (PROTOCOL=TCP)
      (HOST=ORLSUN9)
      (PORT=4446)
    )
    (CONNECT_DATA=
      (SID=V7321)
    )
  )
```

### CODE EXAMPLE 2 Syntax SDU in Listener.ora File

```
LISTENER=
  (ADDRESS_LIST=
    (ADDRESS=
      (PROTOCOL=TCP)
      (HOST=ORLSUN9)
      (PORT=4446)
    )
  )
STARTUP_WAIT_TIME_LISTENER=0
CONNECT_TIMEOUT_LISTENER=10
TRACE_LEVEL_LISTENER=OFF
SID_LIST_LISTENER=
  (SID_LIST=
    (SID_DESC=
      (SDU=8192)
      (SID_NAME=V7321)
      (ORACLE_HOME=ORACLE/7321)
    )
  )
```

#### Example:

Assume the SDU is 2 kilobytes, the array\_size parameter is set to 3 and the first 6 rows of data are the following sizes (in bytes): 1511, 410, 730, 300, 200, 500.

The Oracle server first requests the server side Net8 to send 2651 bytes (the first three rows), then 1000 bytes (the last three rows). The Oracle server sends the following datagrams:

Datagram	Size (bytes)	Data (bytes)	Net8 header (bytes)
1	2048 (SDU)	2038	10
2	623	613 remaining	10
3	1010	1000 requested	10

## Relationship Between SDU and MTU Parameters

The maximum transfer unit (MTU) defines the buffer size of UNP, specifically with TCP/IP. The following statements summarize the relationship between SDU and MTU parameters:

- If SDU = MTU This is the ideal situation; no fragmentations occur.
- else if SDU > MTU Fragmentation occurs.
- else SDU < MTU Performance does not increase.

---

**Note** – The three preceding conditions are met if there is enough space left for the UNP header information.

---

Example:

Assume the ULP is TCP/IP and the MTU parameter (buffer size) is set to 1500. Packet 1 is 2048 bytes (condition: SDU > MTU), which cannot be “absorbed” by the ULP because of ULP buffer size limitations. As a result, fragmentation occurs and performance suffers.

Example:

TCP/IP-level fragmentation:

Net8 buffer size 2048

TCP/IP buffer size 1500

This combination generates two Net8 packets. Packet 1a is 1500 (1460+40) bytes and packet 1b is 628 (588 + 40) bytes. As a result of this fragmentation, the amount of traffic passed to the LLP increases by a multiple of two. When these packets go through the datalink layer, more data is prepended (for example, Ethernet, 14 bytes). Theoretically, at the bottom of the client stack, the size of the two packets is:

$1500 + 14 = 1514$  packet 1a

628 + 14 = 642 packet 1b

Now consider packet 2 (SDU < MTU). Since the size of this packet is 623 bytes, less than the MTU size (1500 bytes), there is no fragmentation. However, increasing the Net8 packet size can increase performance as a larger packet transforms across the network.

packet 2 → 623 (data) + 40 (TCP/IP header) + 1 padding (byte) + 14 (Ethernet header) = 678 data (bytes)

Now consider the ideal condition where SDU equals MTU. In this situation, there is no fragmentation as the buffer sizes are synchronized. This is the optimum situation.

## Net8 Tuning

As discussed previously, performance optimization means reducing network traffic, which can be achieved through the *tuning* process. Referring to FIGURE 3 >, the Oracle server application passes a *batch* of data to Net8, where a 10-byte control header ( $H_S$ ) is prepended, forming a *frame* that is passed to the ULP. The ULP prepends its header  $H_{ULP}$  the size of which depends on the protocol used. TCP/IP<sup>1</sup>, for example, uses a 40-byte header<sup>2</sup>; IPX/SPX, a 30-byte header<sup>3</sup>, forming a *datagram* that is passed to the LLP. The LLP prepends its header  $H_{LLP}$  the size of which again depends on the protocol used. Ethernet, for example, uses a 14-byte header<sup>4</sup>, forming a *packet* that is passed to the physical layer for transmission.

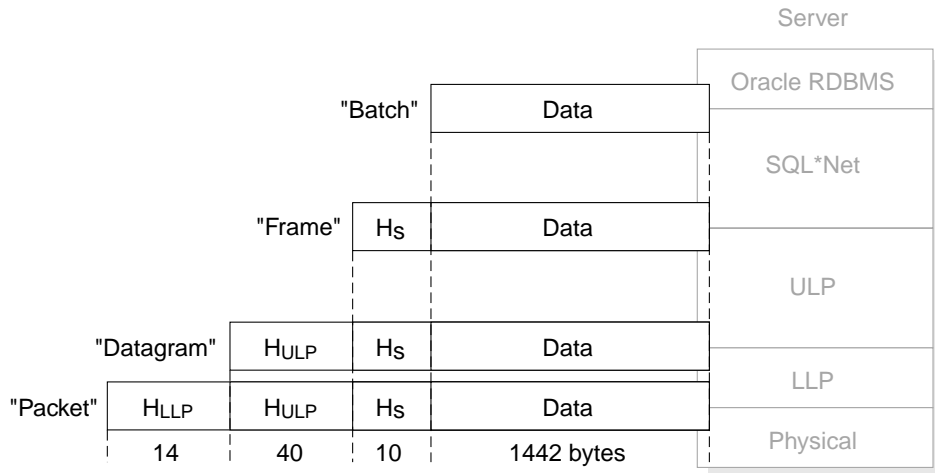


FIGURE 3 > Data Flow Through the Server Network Stack

Ideally, if the data buffers of Net8, the ULP, and the LLP are synchronized, fragmentation is minimized or eliminated as data flows from the application layer to the LLP.

Example:

Assume the SDU is 2 kilobytes, the ULP (TCP/IP) MTU is 1500 bytes, and the LLP (Ethernet) buffer is 1506 bytes. The application passes 1442 bytes of data to Net8, which prepends a 10-byte header, producing a frame of 1452 bytes. Net8 in turn passes the frame to the ULP, which prepends a 40-byte header, producing a datagram of 1492 bytes. ULP then passes the datagram to the LLP, which prepends a 12-byte header, producing a packet of 1506 bytes. The batch has successfully passed through the client stack without fragmentation.

In this example, note that because each succeeding lower layer buffer is large enough to absorb the data received from its respective upper layer, there is no fragmentation. This is the ideal situation. In practice, this is seldom possible due to incompatibilities between buffer sizes of the layers. When data flows between layers of incompatible buffer sizes, fragmentation occurs, and as a result, extra network traffic is generated. With this in mind, components of the stack can be tuned to minimize fragmentation, which reduces network traffic and thereby increases performance.

---

## Tests

The following sections describe the test conditions and methodology.

### Test Conditions

1. LAN and WAN.
2. Defaulted parameters by varying values of parameters such as SDU, TDU, and array size.
3. Set TDU to 1024 and changed SDU to 1 kilobyte, 2 kilobytes, 4 kilobytes, and 8 kilobytes.
4. Set TDU to 2048 and changed SDU to 1 kilobyte, 2 kilobytes, 4 kilobytes, and 8 kilobytes.
5. Changed the `array_size` parameter to 1, 2, 5, 10, 25, 50, 100, 150, and 1000
6. Client application—SQL\*Plus 3.3.2.0.2

7. Client-side—Net8 2.3.2.1.4; server-side—Net8 2.3.2.1.0.
8. Server operating system—Solaris™ 2.5 operating environment
9. Client operating system—Windows NT 4.0
10. RDBMS version 7.3.2.1.0
11. Created three tables:
  - stats1 (4096 rows, 20 bytes/row, VARCHAR2)
  - stats2 (4096 rows, 100 bytes/row, VARCHAR2)
  - eoug (410 rows, LONG).
12. Disabled screen output with `set termout off;`
13. Enabled timing with `set timing on;`
14. Turned on SQL tracing with `alter session set sql_trace=true;`
15. Network protocols—TCP/IP, SPX/IPX, and DECNet.
16. Testing on local area network (LAN)—client/server on the same subnet.
17. Testing on wide area network (WAN)—client/server on a different subnet.
18. Repeated each test three times and averaged the results.
19. Used Ethernet as the LLP for all tests.

## Test Methodology

Since the size of the SDU parameter is negotiated between the server and the client, the size of the parameter on the server side was set to 8 kilobytes and did not change as the client SDU was varied from 1 kilobyte, 2 kilobytes, 4 kilobytes and 8 kilobytes.

1. Initiate the following command from the client and capture the transmitting time ( $t_{c \rightarrow s}$ ) from client to server using the LAN analyzer.

```
$ sqlplus system/manager@test1;  
SQL>
```

2. Fetch/retrieve data from the server table using the following command.:

```
SQL>select * from stats1;
```

**3. Capture the data transmission from server to client and the transmitting time ( $t_{s \rightarrow c}$ ) using a LAN analyzer. The total connect time is as follows:**

$$total\_connect\_time = t_{c \rightarrow s} + t_{s \rightarrow c}$$

On the client, the array size and TDU parameters are constants and the SDU parameter is changed to 1 kilobyte, 2 kilobyte, 4 kilobyte, and 8 kilobyte. The following section describes the results obtained.

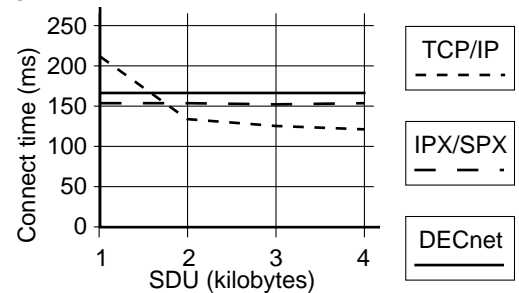
## Results/Observations

When the TDU and array size parameter were set to default values, the following results were observed with respect to connect time and packet size.

Changing the SDU parameter significantly improved connect time ( $t_{s \rightarrow c}$ ). This improvement can be attributed to the facts that the amount of data being retrieved remained constant, but the size of the packets was varied.

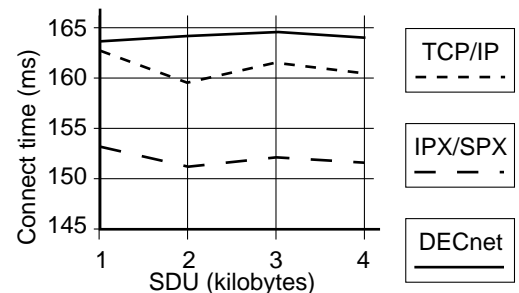
**TABLE 2** Table Stats2 Row size = 100, Batch size = 1500 bytes, TDU = default size, Array size = 15 (default), and client SDU

SDU (kilobytes)	Connect time		
	TCP/IP	IPX/SPX	DECnet
1	211.9547	152.177	164.813
2	133.3079	152.093	164.501
4	128.0889	151.997	164.721
8	124.5137	152.321	165.008



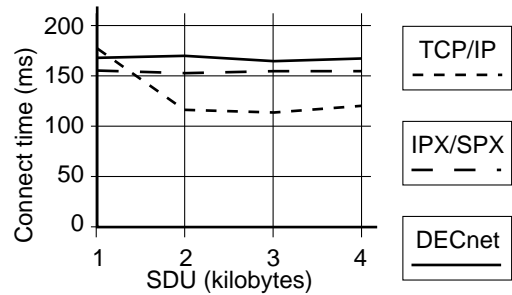
**TABLE 3** Table Stats2 Row size = 100, Batch size = 1500 bytes, TDU = 1024, Array size = 15 (default), and client SDU

SDU (kilobytes)	Connect time		
	TCP/IP	IPX/SPX	DECnet
1	162.6090	153.003	163.792
2	159.7580	151.897	164.036
4	161.8337	152.349	164.492
8	160.5553	152.102	164.123



**TABLE 4** Table Stats2 Row Size = 100, Batch size = 1500 bytes, TDU = 2048, Array size = 15 (default), and client SDU

SDU (kilobytes)	Connect time		
	TCP/IP	IPX/SPX	DECnet
1	172.2118	152.001	164.514
2	119.0573	151.923	165.121
4	118.9678	152.147	163.447
8	120.9134	152.391	164.823



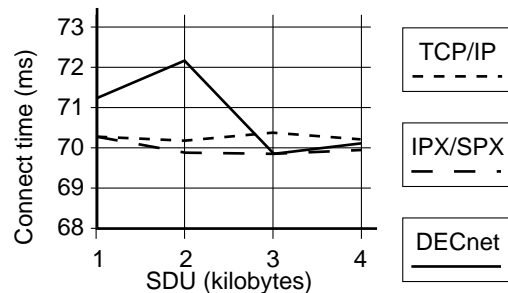
Based on the preceding three performance curves, the best performance is at a TDU size of 2048 bytes. Also, further increments of SDU after 2 kilobytes have negligible effect on the connect time.

The preceding three graphs provide a performance comparison between the three network protocols in a client/server environment. By calibrating all the variables discussed in this paper, it was observed that the protocol that contains the largest buffer size provides the best performance. Protocols SPX/IPX and DECNet maximum buffer sizes are 576 bytes and TCP/IP is 1500 bytes. The SDU parameter is always greater than SPX/IPX and DECNet buffer sizes, so changing SDU parameter did not improve connect time.

It can also be observed from the curve in TABLE 3 that when SDU is much greater than TDU, no performance gains are realized as the TDU becomes a bottleneck, causing numerous smaller packets to be generated. Because these connection-oriented protocols generate a corresponding number of ACK packets, the network traffic increases proportionally.

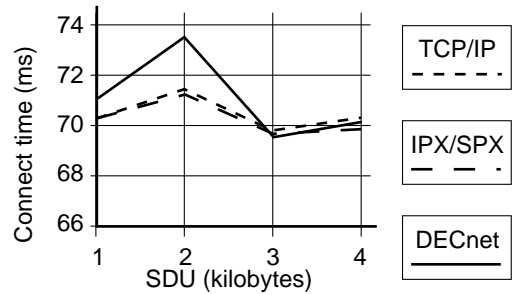
**TABLE 5** Table Stats1 Row size = 20, Batch size = 300 bytes, TDU = default size, Array size = 15 (default), and client

SDU (kilobytes)	Connect time		
	TCP/IP	IPX/SPX	DECnet
1	70.2852	70.281	71.237
2	70.1914	69.953	72.112
4	70.3982	69.963	69.923
8	70.2184	69.852	70.167



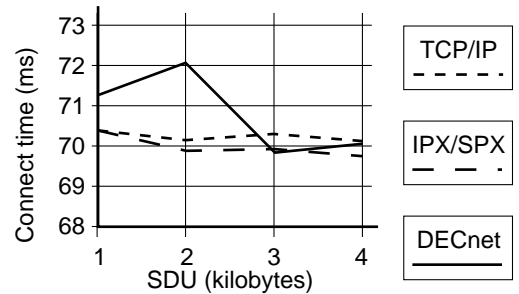
**TABLE 6** Table Stats1 Row size = 20, Batch size = 300 bytes, TDU = 1024, Array size = 15 (default), and client SDU

SDU (kilobytes)	Connect time		
	TCP/IP	IPX/SPX	DECnet
1	70.2717	70.212	71.167
2	71.6192	71.375	73.578
4	69.7773	69.351	69.306
8	70.2174	69.851	70.166



**TABLE 7** Table Stats1 Row size = 20, Batch size = 300 bytes, TDU = 2048, Array size = 15 (default), and client SDU

SDU (kilobytes)	Connect time		
	TCP/IP	IPX/SPX	DECnet
1	70.3445	70.340	71.297
2	70.1581	69.919	72.077
4	70.3625	69.933	69.887
8	70.1279	69.761	70.076

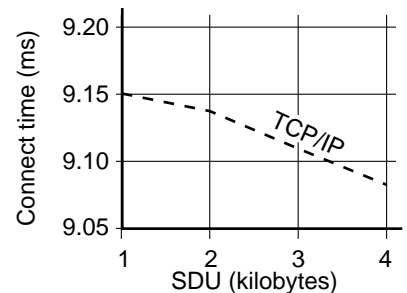


According to the preceding test results, the batch size was 300 bytes ( $300 \ll \text{SDU}$ ) and not enough data filled the Net8 and/or the UNP buffers. Therefore, the connect time was consistent for each protocol and the performance of each protocol matched closely.

## Relationship Between Long Data Type and SDU

Packet size = 131 bytes Datatype LONG

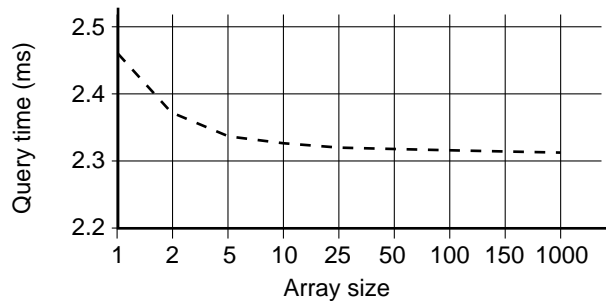
SDU (kilobytes)	Connect time
	TCP/IP
1	9.1504
2	9.1395
4	9.1149
8	9.0861



According to the preceding test results, if `datatype` is `LONG`, synchronizing buffer sizes has no effect on performance. Also, changing `SDU`, `TDU`, or array size has no effect on packet sizes. The packet size was approximately 131 bytes. This may be the reason why the tuning parameters have no effect. Further studies are necessary to investigate this. Repeating the same test for `IPX/SPX` and `DECNet` protocols produced similar results.

TABLE 8 Query Time

Array size	Time (ms)
1	2.4602
2	2.3780
5	2.3346
10	2.3202
25	2.3126
50	2.3122
100	2.3120
150	2.3117
1000	2.3105



When retrieving an arbitrary number of rows, a smaller `array_size` parameter reduces fetch time but increases overhead. A larger `array_size` parameter increases fetch time but reduces overhead. Overall, a larger array size produces better results.

---

## Recommendations/Standards

1. Synchronize buffer sizes.
2. The application must have the capability to determine the array size.
3. Implement an indexing method to read the `Tnsnames.ora` file.
4. Employing the Thin JDBC driver eliminates the need for the Net8 layer in the client so the `connect_time` could be faster.

---

## Conclusions

Based on the preceding results, changing the SDU parameter makes a significant difference in transmitting time independent of ULP. A LAN analyzer reports that the size of the data packets is 350 bytes. The default array size is 15 and each row in the stats table is 20 bytes.

Example:

$$[15 (\text{array\_size}) * 20 (\text{bytes row size})] + 10 (\text{bytes Net8 header}) + 40 (\text{bytes TCP/IP header}) = 350 \text{ bytes} \quad (2)$$

Since the minimum value of SDU (1 kilobyte) is greater than 350 bytes, changing the SDU value did not affect the lower levels, which were able to absorb all the data without fragmentation. A product of row size and array size plus Net8 header greater than 1 kilobyte would cause fragmentation. Calibrating the SDU parameter does not improve performance (connect time).

Example:

If row size is 100 bytes, according to (2) a frame of 1550 bytes would be produced. If the SDU is 1 kilobyte, the results would be  $((950 \text{ data} + 10 \text{ Net8 header} + 40 \text{ TCP/IP header} = 1000) + (550 \text{ data} + 10 \text{ Net8 header} + 40 \text{ TCP/IP header} = 600)) = 1600 \text{ bytes}$ . This is where the fragmentation occurs.

Based on the preceding results, implementing a network application (client/server) is a complex and tedious process because the efficiency of the application depends on many other external factors.

A network application developer must understand the stack paradigm in order to design an efficient client/server application. Also, in order to have an efficient environment, it is very important to select the proper ULP, LLP, and network hardware implementation.

Thus, performance is based on raw length of a table as well as datatype. Connect time and query time are independent of each other, but total performance is based on these two components.

Synchronizing Net8, ULP, and LLP buffer sizes produces the optimum client/server performance. Net8 should be intelligent enough to adjust its buffer size automatically according to UNP buffer size. UNP should have the capability to synchronize its buffer with LLP. Having two different parameters (SDU/TDU) complicates the situation because the smaller of the two determines the buffer size, so SDU or TDU becomes the bottleneck.

These tests results do not reflect the testing on wide area networks (WAN), which is left for future discussions. Future studies are necessary to investigate the effectiveness of `LONG` datatype on client/server applications. Also, row size does not have a one-to-one relationship with the connect time. Note that table `STATS1` contains 100bytes@row and `STATS2` 20bytes@row. The row length proportion is 5:1, but the connect time based on these results (table numbers) is approximately 2:1.

---

## References

1. Postel, J.B., "Transmission Control Protocol," 1981 September; 85 p. (RFC 1123)
2. Branden, R.T., ed. "Requirements for Internet hosts-application and Support," 1989 October; 98 p. (RFC 793)
3. Naugle, M., "Network Protocol Handbook," 1994, 186 p.
4. Nassar, D.J., "Ethernet and Token Ring Optimization," 1996, 513 p. 5.

---

## About the Author

Gamini Bulumulle (Gamini.Bulumulle@sun.com) is a Senior IT Architect, with Sun Microsystems Professional Services. He has been with Sun for six years. Previously he worked with AT&T Bell Labs as a Senior Network Engineer and Oracle Corporation as a Senior Consulting Technical Specialist. He has a Bachelor of Engineering with a Minor in Mathematics and Major in Computer Engineering, a Master of Science in Computer Engineering, and is presently a Ph.D. candidate and working on his dissertation in Computer Engineering.

---

## Ordering Sun Documents

The SunDocs<sup>SM</sup> program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals through this program.

---

## Accessing Sun Documentation Online

The docs.sun.com web site enables you to access Sun technical documentation online. You can browse the docs.sun.com archive or search for a specific book title or subject. The URL is <http://docs.sun.com/>

To reference Sun BluePrints OnLine articles, visit the Sun BluePrints™ OnLine Web site at: <http://www.sun.com/blueprints/online.html>