An Analysis of Database Replication Technologies With Regard to Deep Space Network Application Requirements
Andrea M Connell
Mentors: Paul Wolgast, Silvino Zendejas

Abstract
The Deep Space Network (DSN) has three communication facilities which handle telemetry, commands, and other data relating to spacecraft missions. The network requires these three sites to share data with each other and with the Jet Propulsion Laboratory for processing and distribution. Many database management systems have replication capabilities built in, which means that data updates made at one location will be automatically propagated to other locations. This project examines multiple replication solutions, looking for stability, automation, flexibility, performance, and cost. After comparing these features, Oracle Streams is chosen for closer analysis. Two Streams environments are configured – one with a Master/Slave architecture, in which a single server is the source for all data updates, and the second with a Multi-Master architecture, in which updates originating from any of the servers will be propagated to all of the others. These environments are tested for data type support, conflict resolution, performance, changes to the data structure, and behavior during and after network or server outages. Through this experimentation, it is determined which requirements of the DSN can be met by Oracle Streams and which cannot.

Introduction
The Deep Space Network (DSN) is an international system of antennas and associated telecommunications equipment used to communicate with spacecraft throughout our solar system. It is comprised of three complexes located at strategic points around the world. Each mission – including Opportunity, roving around Mars; Dawn, orbiting the Vesta asteroid; and Voyager 2, reaching the very edge of our solar system – receives commands and sends data back to Earth through this network. This is made possible through precise tracking information calculated by teams at the Jet Propulsion Laboratory and sent to the three complexes. Distributing this type of support data reliably is of vital importance to gathering knowledge about the solar system.

Database replication is the best way to ensure automatic, continuous, and reliable distribution of data. Replication means that any changes to the data at one location are automatically and almost immediately propagated to a second location. DSN applications are built upon the Oracle database management system, which provides multiple options for replication. These were considered as a solution to the DSN's distribution needs a number of years ago, but at that time they were not stable and mature enough to meet all of the application requirements. A workaround was devised to share data, but it is labor-intensive and not an ideal alternative. Now that replication technologies are more advanced, they may provide a more suitable solution. This paper reviews current replication technologies to determine whether they have improved enough to merit implementation.

Three replication technologies were researched: Oracle Streams, Oracle GoldenGate, and SharePlex for Oracle. While GoldenGate and SharePlex have some features that Streams doesn't, the features are not all needed by the DSN. Oracle Streams provides the best coverage of DSN application requirements with the lowest cost. The results of an in-depth analysis of Oracle Streams are promising. Two Oracle servers were configured as replication environments. It was decided to test for datatype support, conflict resolution, performance, and connection problems first. After ensuring that Oracle Streams could meet these basic requirements, we looked into filtering, transformations, recovering from being out of sync,
replicating through a firewall, and other more advanced features. Based on this initial testing, Oracle Streams appears to be a suitable solution to the DSN applications' replication needs.

The next section will explain these steps in detail. There is still some work to be done before Oracle Streams can be used in a production environment. This testing was done on a simple schema with artificial data. Before being implemented, in-depth testing will need to be done in a realistic database with representative data definition (DDL) and data modification (DML) patterns. If Oracle Streams 11.2 is used for DSN applications, it will save time and money by automating the process of data distribution.

**Methods**

*Replication Technologies*

There are three potential solutions for data replication in Oracle: Oracle Streams, Oracle GoldenGate, and SharePlex for Oracle. Streams is part of Oracle Enterprise Edition and has been evolving for many years. GoldenGate was recently acquired by Oracle and is being actively integrated into the current product base. While Streams is a very flexible and mature product, GoldenGate is more highly recommended by Oracle for new applications because of the ongoing upgrades to it. GoldenGate is real-time, provides support for heterogeneous replication environments, and is generally considered easier to configure. However, none of these features are driving requirements for the DSN. Since Oracle Streams is included in JPL’s current Oracle license and is available at no additional cost, it was chosen for testing. The main area that may be an issue with Streams is performance during high-throughput periods. Testing will need to focus some testing on the extreme limits of DSN applications, measuring how much latency is created when many records are updated within a short period of time. The final option, SharePlex, is a third-party application which did not appear to have any major advantages over Streams and GoldenGate which would affect the decision.

*Environment Setup*

Based on the analysis above, an Oracle Streams environment was set up for testing. The simplest case is one-way replication using a Master/Slave architecture, in which one database is designated as the source and a second database is the destination. This involves one method to capture changes on the source and to insert them into an outgoing queue. Streams has two ways to capture data: the Capture Process and Synchronous Capture. The capture process scans redo logs to capture DML and DDL changes. Synchronous capture runs in the background and grabs DML changes on objects as they happen. Synchronous capture cannot replicate DDL changes and does not work on all datatypes; because of these limitations the following work uses the capture process. Before insertion into the queue, the captured changes are converted into Logical Change Records (LCRs) which hold information on the source database, the modified object, the old data values, and the new data values. The destination database has an apply process, which takes LCRs from its incoming queue and executes them on the object. The source database also needs a propagation process which tells the capture queue to send changes to the apply queue on the destination. After successfully implementing this architecture, a Multi-Master environment was created, in which each server has both a capture and an apply queue. Changes made on either database can be propagated to the other. By its nature, this architecture creates more opportunity for error. When two servers can modify data simultaneously, it is possible to have data conflicts. While it is imperative to test the impact of these conflicts, they are difficult to simulate reliably since manually injecting conflicts is more difficult. A single tester would need to insert rows almost simultaneously on both servers to create this effect. It is possible to simulate conflicts by enabling a firewall
between the two servers, making data modifications, and then removing the firewall, but using a Master/Slave architecture is simpler and has the same result. For this reason, the majority of the following work has been done on a Master/Slave architecture where a tester can simulate conflict errors by modifying data first in the Slave and then in the Master.

Data Type Support
The capture, propagate, and apply steps were examined to ensure that they were executed properly for each datatype of interest to DSN applications. Much of the testing time was focused on LOBs, which are large objects of either binary (BLOB) or character (CLOB) data, since they are heavily used in DSN applications and often work differently than other simple datatypes. BLOBs, CLOBs, and XMLTypes stored as CLOBs are supported natively (along with basic types like number, varchar2, date, etc). XMLTypes stored object relationally or as binary XML are supported with the addition of the Extended Datatype Support (EDS) package. However, behavior during conflicts varies among these types. Conflicts occur when values in the destination database do not match what the source reported having prior to the modification (including cases where the row has been added or removed). This may occur when the same record is inserted, deleted, or updated on both databases at roughly the same time, usually in a Multi-Master environment. Generally, the old values from the source are compared to the values in the destination before the new values are applied. If a conflict is found, an error is reported and the update does not occur on the destination. However, there are some exceptions to this rule. Conflict checking does not occur on columns with the datatype of BLOB, CLOB, or XMLType stored as CLOB. On tables that require EDS, conflict checking does not occur on any columns. In these situations, the previous values are simply overwritten and an error is not reported. These behaviors can be modified with built-in and custom handlers, which are discussed in the next section. Longs, rowIDs, BFiles, User-defined types, and some Oracle-supplied types (such as ANY types, URI types, spatial types, and media types) are not captured in Oracle Streams, but are not as important in DSN applications as the types mentioned earlier.

Conflict Handlers
Oracle Streams has four built-in methods to handle update conflicts: OVERWRITE, DISCARD, MAXIMUM, and MINIMUM. They can automatically compare the conflicting records and select which value to use depending on which method is selected. When a conflict is detected and the OVERWRITE handler is on, the new values will be used to replace the old ones. On the other hand, when the DISCARD handler is on, the new values are ignored. In a two-way Multi-Master architecture, these two handlers could be used to always give values from one database priority over the other. The MAXIMUM and MINIMUM handlers take a column name as a parameter and decide which record to keep based on the value of that column (keeping the LCR with the highest or lowest value, respectively). However, these built-in methods do not work for LOB types and may not be appropriate for all applications. Therefore, Oracle also allows a custom DML handler or error handler to be assigned to each object. A DML handler is executed on every row that is affected by an LCR and can run either SQL statements or a PL/SQL procedure. Error handlers are PL/SQL procedures that fire when a conflict is detected on a row. Experimentation has shown that it is possible to create an error handler that resolves unique constraint violations by changing the INSERT into an UPDATE, and that resolves "no data found" errors by turning the UPDATE into an INSERT. However, there are some limitations with custom handlers. LOBs are not allowed in user-constructed LCRs, so new change records cannot be created (but existing ones can be modified). In addition, if a row has a LOB column but the LOB was not altered, it is not included in the LCR and cannot be added to the LCR by the handler. In effect, this means that handlers cannot alter LOB columns unless the initial change updated or inserted it. Chapter 14 in the Oracle Streams Replication Administrator's Guide shows this as being possible, but this outcome could not be reproduced in testing. Another
limitation relates to Extended Datatype Support. To replicate datatypes that are not natively supported in Streams, EDS adds a DML handler to objects that require it. Only one handler can be applied to each object, so these tables cannot utilize custom procedure handlers.

**DDL**
The majority of Data Definition Language changes are captured in Oracle Streams. Creating and altering tables, views, packages, procedures and more were accomplished with no issues. However, there are some DDL statement that are not captured in Oracle Streams⁶.

- CREATE DATABASE
- ALTER DATABASE
- ALTER SYSTEM
- ALTER SESSION
- CREATE CONTROLFILE
- CREATE PFILE
- CREATE SPFILE
- SET ROLE
- CALL
- EXPLAIN PLAN
- LOCK TABLE

In addition, GRANT statements are not captured on views. The values of Sequences cannot be synchronized. CREATE SEQUENCE and ALTER SEQUENCE statements are captured, but NEXTVAL is not. Documentation recommends initializing servers to use different values (for instance, even numbers on one server and odd on another) in order to prevent duplicate sequence values. After EDS has been configured, tables that require it should not be altered. This means that DDL changes are not captured for tables with XMLType data stored either object relationally or as binary XML. DDL for other tables and objects will still be captured normally. Queue names should be no more than 24 characters because extra information is automatically appended to the name. Tables requiring EDS should be named with no more than 28 characters because procedures and other objects will be created to handle replication of types that are not natively supported⁸.

**Performance**
Performance was tested next. Since some documentation implies that Oracle Streams does not scale well when many changes occur within a small time frame⁴, the tests attempted to stress the system. First, 50,000 small records were inserted at one time, and took less than ten seconds to propagate. Clearly Oracle Streams was not overloaded by a large number of small records, so the next logical thing to test was a smaller number of larger records. The configuration was tested with four transactions, each containing 500 inserts. The records in the first transaction were all 1KB each, the second were 10KB each, the third were 100KB each, and the fourth were 1,000KB each. These results are shown in Graph 1, with the size of each record along the x-axis and the number of seconds for the task to finish along the y-axis. The graph shows both the time for the initial insert on the source database and the time for the destination database to reflect the changes after the source insert finished. This allows viewers to consider the time that the destination server needed to apply these large chunks of data in addition to the propagation time.

Graph 2 displays the calculated rate of transfer for each transaction as compared to the network speed of about 8 Megabyte/second. This graph shows the transfer rate leveling off rather than decreasing like research implied would happen. It is possible that this testing did not include large enough data sizes to see the negative impact on performance. Another piece of information obvious from Graph 2 is that the propagation speed is much slower than the overall network speed. There are two reasons for this. One is that LCRs contain more information than just the new values, so they incur overhead in propagation time. The second explanation is that the time to be applied to the destination server is also
included here. Nonetheless, it is beneficial to see the speed of propagation relative to the network speed.

**Graph 1: Propagation Time**

<table>
<thead>
<tr>
<th>KB Per Row</th>
<th>Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>10</td>
<td>0.4</td>
</tr>
<tr>
<td>100</td>
<td>3.8</td>
</tr>
<tr>
<td>1000</td>
<td>14.2</td>
</tr>
<tr>
<td>10000</td>
<td>107.8</td>
</tr>
<tr>
<td>100000</td>
<td>96.9</td>
</tr>
<tr>
<td>1000000</td>
<td>1011</td>
</tr>
</tbody>
</table>

**Graph 2: Propagation Speed**

<table>
<thead>
<tr>
<th>KB Per Row</th>
<th>MB/second</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>100</td>
<td>8</td>
</tr>
<tr>
<td>1000</td>
<td>8</td>
</tr>
</tbody>
</table>

These test results may not accurately predict how long propagation will take because of differences in data types and network configuration, but the measurements of scalability should prove useful. During testing, it quickly became obvious that large updates filled the archive logs and froze the transaction until old archive logs were deleted or the DB_RECOVERY_FILE_DEST_SIZE was increased. The size of archive logs will be determined by the average amount of data modifications (DML and DDL) made in a day and the retention needs of the application. Streams also requires supplemental logging, but the level
required will depend on the database – the use of primary keys, DML handlers, and transformations are some of the factors that affect logging needs. This testing has demonstrated the negative effects of the extra logging required by Streams.

Connection Problems
Another important feature to test is behavior during and after network or server outages. Outages were simulated with a firewall between the two databases to ensure that updates remain in the queue until they are appropriately applied to the destination server, even if it takes a couple of days for the connection to be restored. Documentation reveals that if a large number of LCRs are created while the destination is unavailable, the queue contents will spill to disk. If more than one destination is using the queue, this will impact the performance on those destinations that are still applying changes. One way to mitigate this problem is to use DBMS_STREAMS_ADM.SPLIT_STREAMS on the source to disable propagation to the unavailable server, and then MERGE_STREAMS_JOB when it becomes available again. Administrators can be alerted when a server is unavailable through Enterprise Manager Database Console or by watching the DBA_OUTSTANDING_ALERTS table. An alert will be generated if the capture or apply processes abort, or after a propagation has failed 16 times in a row.

If the Master in a Master/Slave environment goes down for a substantial period of time, updates may need to be made to the Slave. In this case, when the Master comes back online it will not automatically sync with the Slave and the system will be inconsistent. Oracle's DBMS_COMPARISON package can be used to detect discrepancies and converge the two sets of data.

Transformations and Filtering
In some applications the destination database may not be an exact copy of the source. Tables may be set up differently or perhaps not all of the data will be necessary in the destination. Streams has built-in methods to add, remove, and rename columns as well as methods for renaming the schema or table. Other transformations can be done with custom PL/SQL, but this is a bit slower and more complicated. Filtering rules can be set to global (selecting all data), schema (selecting all data in the given schema), table (selecting all data in the given object), or subset (selecting all data in an object which meets a specified condition). Subset rules cannot be applied to tables with LOB or XMLType columns, but other filtering methods can. Each step (capture, propagate, and apply) can have up to two rule sets associated with it that determine which records are affected. One rule set is positive, indicating everything that should be included in the replication. The other set is negative, determining what data to exclude. The combination of transformations and filtering allows a great deal of flexibility in which parts of the database are replicated, which can improve performance by limiting how much data needs to be sent over the network and allows the replicas to each serve unique functions in the application.

Oracle Database File System
Oracle 11.2 has introduced Database File System (DBFS). This allows a file system interface to be used within an Oracle database. There are two ways to implement storage in DBFS. This testing uses SecureFile Store, in which the file system data is stored in a table in the database. Files are stored as SecureFile LOBs, which are an improved storage method for large objects that allow encryption, compression, deduplication, and faster access times. The second storage option is Hierarchical Store, which allows files to be written to tape or cloud storage. In Oracle 11.2, Amazon S3 is the only cloud system supported. The files and directories within DBFS can be managed through a file system mount, a command line client, or PL/SQL procedures. The file system mount is currently only available on Linux and Linux.X64 platforms with the FUSE package installed. FUSE (Filesystem in Userspace) is supported on Linux kernels 2.4.X and 2.6.X. The possibility of replicating DBFS is still
unclear. The setup of this feature does not seem to be copied to the replica entirely. Although the results from the PL/SQL procedures to create files and directories are captured and propagated correctly, the functionality to view the files does not work on the destination server.

**HTTP/HTTPS Propagation**
Some of the DSN servers may be located behind firewalls that do not accept all network traffic. Under a typical setup using Oracle Net Services, replication activities could be blocked by the firewall and never arrive at the destination. Changing the protocol to use HTTP or HTTPS makes it easier to allow this traffic through the firewall. This solution is simple to implement, requiring only a change in the tnsnames file, and does not affect any functionality of Oracle Streams. HTTP/HTTPS propagation will need to be analyzed further to ensure that it meets the security needs of DSN applications.

**Conclusion**
This research discovered many improvements to the earlier version of Oracle Streams that DSN previously considered. Although the inability to synchronize sequences and limitations to the functionality of LOBs and XMLTypes will need special consideration, overall this seems like a good solution to the data replication needs of DSN applications. Further testing with realistic data will reveal any restrictions that were not discovered in this basic test environment and determine the success of potential work-arounds. The automation of data replication will decrease the man-hours needed to distribute data among the various sites. Implementing Oracle Streams 11g will save time and money for the Deep Space Network.

**Acknowledgments**
This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by the Jet Propulsion Laboratory Student Internship Program and the National Aeronautics and Space Administration. I would like to acknowledge my mentor, Paul Wolgast, for providing context and support to my project. My co-mentor, Silvino Zendejas, was an instrumental source of information regarding the data replication requirements of the Deep Space Network applications. I would also like to thank Syed Sadaqathullah for his time setting up and supporting my test servers.
References

   http://www.quest.com/shareplex-for-oracle/
5. Chapter 245: Logical Change Record Types. Oracle Database PL/SQL Packages and Types Reference 11g Release 2 (11.2). Oracle.  
   http://download.oracle.com/docs/cd/E14072_01/appdev.112/e10577/t_lcr.htm
   http://download.oracle.com/docs/cd/E14072_01/server.112/e10704/toc.htm
   http://www.packtpub.com/article/n-way-replication-in-oracle-11g-streams-1
   https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&doctype=REFERENCE&id=556742.1
   http://download.oracle.com/docs/cd/E14072_01/server.112/e10705/toc.htm
    http://www.springerlink.com/content/j0kv20456721t311/fulltext.pdf
    http://download.oracle.com/docs/cd/E14072_01/appdev.112/e10645/adlob_fs.htm
    http://www.oracle-base.com/articles/11g/DBFS_APIs_11gR2.php