An Optimized Technique of Increasing the Performance of Network Adapter on EML Layer

Prashanth L
4th semester, M.Tech [SE],
ISE Dept
RVCE Bangalore

Shantharam Nayak
Associate Professor,
ISE Dept
RVCE Bangalore

ABSTRACT

Simple Network Adapter initially which acts as an interface between the Transaction server and Network Elements communicates over the channel through TCPPDU. Presently the disadvantage being involved in TCPPDU is to maintain the channel contention, reservation of channel bandwidth. The disadvantage being involved is certain features, version of network elements communicates by receiving the xml over the socket. So, it's not possible to change the entire framework, but by updating the framework an XML Over Socket(XOS) formation should be supported. The XOS implementation is being performed using Java language through mainly in JVM. Such that by this deployment machines would become easier and form a good communication gap between them. This simple network adapter being developed should support operations of the North bounded server and gives an established authorized, secured, reliable portal. The interface being developed should provide a good performance in meeting the network demands and operated conversions of respective objects.

Keywords

XML Over Socket(XOS), XML Req, XML Resp

1. INTRODUCTION

The Simple Network adapter acts as an interface between TL1 and NEs. Simple Network adapter receives Java objects through TL1. The SNA Framework is a set of loosely coupled java classes that provide a skeleton or supporting structure for building SNA adapter features. The SNA architecture is generic enough to handle the request related to TL1, CL1 and SNMP. The framework being designed in unmanaged memory languages are difficult to handle. Lot of memory related operations are being done manual[4].There are several solutions to the problem mentioned above; one is through the use of memory handling languages such as Java. Java being more advanced and powerful language performs handling of memory, run time exceptions automatically without the intervention of the user. Simple Network Adapter performs Operations such as Backup, Download on Network Elements depending on the feature which is being processed from the received request. Initiates Commands to the Network Elements understood language. Simple Network Adapter receives XML over the socket from TL1 based on the request object, finds out the feature name and processed. The Behavior of SNA is based on JVM Properties, which is used during initialization process; design of SNA is such that, we can have more than one instance of SNA in a machine based on different JVM properties file. There is to be very limited or no persistent data stored in the SNA. The required data is stored using Oracle with JDBC. The following categories of Data are to be stored by the SNA, NE connection information. The adapter which forms the major communication on the element management interface layer, operates over the various elements not only on the transaction server. Handles various other elements with respect to the request being involved. Simple network adapter should be configured before on operating over other networking devices. Simple network adapter should handle the information related to the elements. Information being considered should be present in the info queue. This performs the operations based on the queue priority[3].The Simple Network adapter handles the information depending on their availability. The information is placed in the database if the information is being huge. The database involves ne configuration details which are being secured over the other adapters through firewalls, gateways and so on. The security is being established over all the paths in the layers so that there is no unauthorized access to the operations of these parts. The authentication is being done before operating on these elements

1.1 ORGANISED MODULES

The SNA is logically organized in the following functional areas. Adapter Framework–A set of loosely coupled classes that provide a skeleton or supporting structure for building SNA adapter features. Platform supported– Utilities reused from the OMS core and platform portion of the project. Release Feature Set–The layout of classes that represent features supported by SNA. Feature need to be common for all NE Type Release, NE specific details are handled in feature specific, NE Type Release specific Mappers. Services-Data Storage–classes that are related to the support access to the data storage. Notification – Autonomous event processing and filtering. Connection Management–NE Management features includes watchdog, monitoring and recovery. During SNA Framework Initialization all the sub components of Framework and other communication layers like TL1, CLI, and SNMP get initialized and register with framework.

2. ARCHITECTURE

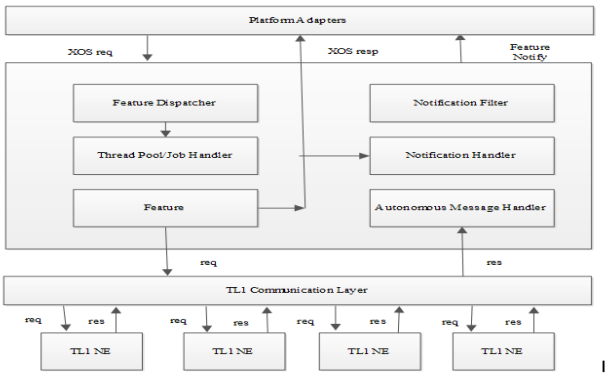

Figure 1: SNA Architecture

y

The SNA architecture is generic enough to handle the request related to TL1.

FRAMEWORK

The SNA Framework as shown in figure 1 is a set of loosely coupled java classes that provide a skeleton or supporting structure for building SNA adapter features that exchange information through transactions with a network element (NE). The goal of the framework is for any developer to use the framework to produce features in short period of time with high quality. The Framework is specifically written to address problems encountered in writing features. It is optimized to solve problems in this domain. The framework is easy to use, robust, and feature – rich. It allows feature designs to be implemented in a consistent manner. The major areas of functionality of the framework are:

- Initialization – start-up and control of the framework threads.
- Client Request Handler – processes client request.
- Feature Dispatcher – instantiates features based on requests and dispatches the features to the thread pool for execution.
- Transaction Deployer – classes that control the flow of transactions issued by a feature including routing all commands, responses and autonomous messages to the appropriate consumers.
- Data Storage – stores connection and performance data for SNA features to use without having to retrieve information from the NE or OMS.

  The Framework code base consists of SNA platform utility classes to perform the following functions:

- Logging – mechanism for creating application activity logs.
- Trace – mechanism for presenting application debug information.
- Thread pool – mechanism for handling features in individual threads.
- Job Handler – classes for feature execution flow control.

FEATURE DISPATCHER

The Feature Dispatcher is implemented as thread which reads the client request objects of its queue based on the request type and scope in the client request, the feature dispatcher decides by looking it up in the features resource files which feature class to instantiate [1]. Then the Feature Dispatcher client requested "Feature" on to the thread pool for execution. The Feature Dispatcher also implements the functionality of iteration by recognizing the unique iteration requests. Once an iteration call is determined the feature dispatcher re- dispatches the awaiting iteration clients. The feature dispatcher helps in handling all the information to the related request.

JAVA THREAD POOLING

The thread pool policy is used within SNA for the feature executions. Thread pools are instantiated, as they are needed. One pool is initialized at start up for handling the features execution. The framework implements classes as static threads to perform functions needed by the features. These threads are static and exist over the lifetime of the SNA execution. The SNA thread pool is used for dispatching feature class for execution. The Framework implements classes as dynamic threads to perform functions needed.

USER JOB HANDLER

The Functionality of the Job Handler is provided by the SNA framework using the Abstract feature that extends Base job and Job classes. Abstract feature provides additional functionality to support feature related Handling [2].

PROCESSING FEATURE

The Feature controls the processing of the transactions with the NE. It also helps the developer to generate sequences of transactions into group. Each feature could contain one or more groups. The transactions that could be sent down at the same time are in the same group.

Each Transaction group controls the response of all the transactions in this group. Each feature controls the response of all the Transaction Groups. Based on the specific request, each feature- implemented by feature specific, NE specific Mappers needs to decide how many sub-tasks are needed. The feature is aware of each transaction and would know if the transaction from an originator and would know if the transaction is a TL1 transaction of a CLI transaction or a SNMP transaction. Also all autonomous messages are converted to Transactions and send to registered Listener. Based on the specific request, each feature-implemented by feature specific, NE specific Mappers needs to decide how many sub-tasks.

GROUP TRANSACTIONS

Transactions can either be Control Transactions, which configure the communications with the Network Element or can be Network Element Transactions which are sent to the Network Element. The Features creates Transactions and adds them to a Transaction Group sends each transaction and then waits for each response before reporting back to the feature that the group has completed.

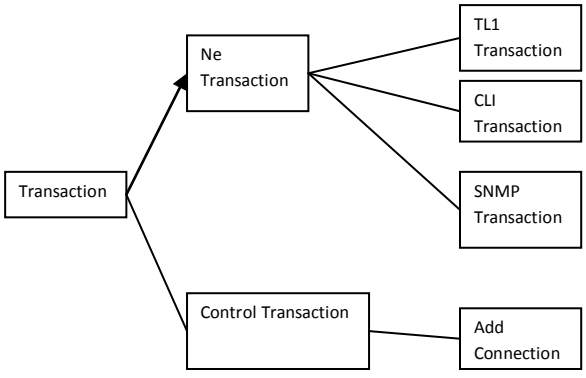

Figure 2: Transaction Flow

Communication

The SNA uses TL1, CLI and SNMP communication for communicating with NEs.

TL1 communication layer connects by TCP/IP or OSI with Network Elements. CLI communication layer connects by TCP/IP with Network Elements .SNMP communication layer connects by UDP with Network Elements.

y

3. WORKFLOW

- Communicate with the North Side and South Side and access management information.
- Process the Requests, obtains information in the specified format.
- Provide Secure Communications using TCP/IP.
- Create and support services.
- Backup, Download, Restore and Software Management Operations are being carried out.

3.1 METHODLOGY

- SNA (Simple Network Adapter) is used to communicate from North side to Network Elements.
- SNA Framework is used to transfer the requests from Thread pool to the Network Element Manager Understanding Format.
- Using Specification Communication layer provide Secured Communications.
- Handling of XML over Socket to SNA specific tags using JAVA XML templates to identify the feature and Network Element type element and check out whether the feature is being supported by specified Network Elements. Operations such as Backup, Download, Restore.
- Shell Scripting to invoke the SNA Server, with supportive Feature perform the Operations.

SNA parses the XML and analyses further to find the corresponding Feature. It then constructs the communication interface REQ object and handover the REQ object to communication layer by calling invoke Operation. Comms layer generates NE specific commands for corresponding "Service Name" and sends it to NE. When Response from NE arrives, it parses that response and forwards it back to SNA. SNA validates the structure of response and constructs the XML-RESP and sends it back to PA/COO.

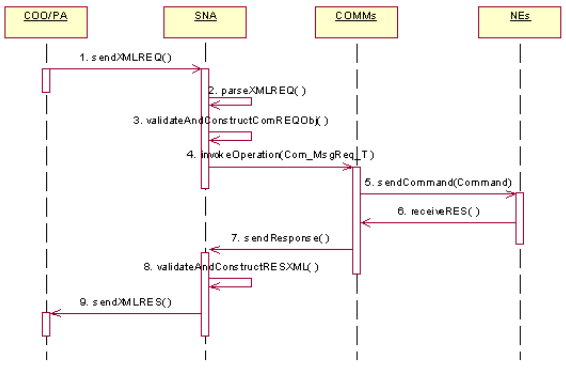

Figure 3: Control Flow

Simple Network Adapter which handles large Number of requests and responses, one of the major constraints were run time exception, handling of huge Network Elements were complicated. Present Simple network adapter architecture lead to handling of memory management explicitly using garbage collection.

Memory Utilization is one of the complicated issues; the past design of Simple Network adapter which was done in C++ was having a major conflict in memory leakages, garbage collections. Cautions' handling of dangling pointer, Memory out of bound exception, segmentation fault were the major issues, which lead to improper working[5]. Adaptation, consistency to work with neighboring instances of simple network adapters is the major criteria being involved in the design.

4. PERFORMANCE

In Telecommunication domains performance is one of the key factor in meeting the market crisis and competing with the competitors. The new designed architecture is reliable enough in handling communications between the integrated components. The key factor of adapter is it accommodates to newer updated network elements and its software versions[6]. The adapter is capable enough in handling real time scenario exceptions. The adapter designed is performed using multi threading capabilities. The legacy design has a constraint of memory operations which slows downs the operations of neighboring components. The java designed architecture is executed over platform jvm which adds core to the performance issue. The adapter is integrated enough in doing tasks related to reducing network bandwidth and capabilities of failure. The adapter is motivated in increasing the success of progress rates and optimized way of performing operations. The C++ and java versus graph are drawn in fig 4, where various tests are carried over each other on several different tasks and graph is drawn.

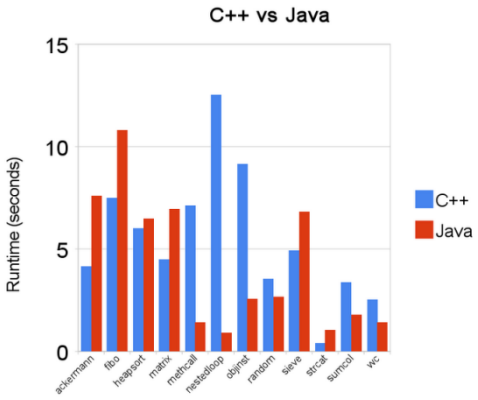

Figure 4 C++ v/s JAVA

5. CONCLUSION

Simple Network Adapter which is being newly developed with new Architecture involves lot of challenges such as deployment of this architecture on any of Network Elements Platform, Platform Independent. Handling of Multiple Threads is done by synchronization and best utilization of Memory. The Simple Network Adapter could be deployed either co-resident within the same Java Virtual Machine as the Optic Management System Platform adapter application or remotely in a separate Java Virtual Machine that would run in the same or separate machine. The behavior of Simple Network Adapter is based on the Java Virtual Machine Properties, which is used during Initialization Process; design of Simple Network Adapter is such that, we can have more than one instance of Simple Network Adapter in a machine based on different Java Virtual Machine Properties File. Simple Interface over across all the features being supported and handling of all types of network elements and at all layers.

y

6. REFERENCES

[1] Priority inversion at the network adapter when scheduling messages with earliest deadline techniques. Meschi, A., Di Natale,; Spuri , M, ;

Digital Object Identifier: 10.1109/EMWRTS.1996.557931, Page(s) : 243 – 248.

[2] Cost Effective Network adapter by Schaff, F. ; Willebeek LeMair, M.: Patel, B.; Digital Object Identifier: 10.1109/HPCS.1992.759255. Page (s) 292- 300.

[3] A high performance software solution for packet capture and transmission by Dashtbozorgi, M. : Azgomi , M.A;. Digital Object Identifier: 10.1109/ICCSIT. 2009.5234920. Page (s) 407-411.

[4] Introduction to Simple Network Adapters fromquinxhttp://www.quinx.com/en/products/smart-network-adaptors/index.html.

[5] Schonwalder J, Marinov V "On the Impact of Security protocols on the performance of SNMP" this paper appears in Network and Services Management, March 2011, ISSN 1932-4537, page(s) 52-64, vol 8, issue 1.

[6] Aleksic S, Fehratovic N "Requirements and limitations of optical interconnect for high capacity network elements" this paper appears in Transparent Optical Networks (ICTON) 2010, ISBN 978-1-4244-7799-9, Munich, 27 June 2010- 1 July 2010, page(s) 1-4.

y