Spacecraft Checkout Data Distribution System: A Survey

Mithalee. V. Kadkol
Student, M.Tech 4th sem, CSE,
Sai Vidya Institute of Technology, Bangalore, India

Ms. Sreedevi. S
Sci/Eng 'SD', CSAD/SCG
ISAC, old Airport road, Bangalore, India

Mr. Harisha D. S
Asst. Prof, Dept of ISE
Sai Vidya Institute of Technology, Bangalore, India

Abstract: **Automated Checkout System Software (ACSS) is a set of software products, developed for the automation of Spacecraft Checkout operations. ACSS will acquire spacecraft telemetry continuously from the Data Acquisition Systems in real time, processes it and displays the Raw and Processed data to the user in the form of Page Displays, Plot etc.**

During the Checkout of a Spacecraft, telemetry data in Raw and Processed forms has to be distributed to various external agencies for their independent analysis of the Spacecraft. Each agency shall request for either raw telemetry data or specific parameters of Processed Telemetry Data. Data also needs to be distributed to various other analysis software packages which are Graphics intensive applications. Each client of the agency shall establish a TCP/IP communication with the TCP Server and shall have a defined protocol for communication.

Keywords: TCP/IP, Checkout, Telemetry

INTRODUCTION

Spacecraft checkout activities are automated to a great extent through a set of software products under the name Automatic Checkout Software System (ACSS). ACSS is housed in the Spacecraft Checkout Computer (SCC). The data acquired from satellite by the SCC from Telemetry as well as other sources has to be distributed to different agencies and support packages based on their requirements in raw or processed format. This paper discusses a new scheme for the centralized distribution of data acquired by the ACSS. It details on the design of the Spacecraft Checkout Server for Data Distribution System (SCS) as a concurrent TCP server with multi threaded architecture. It also discusses the communication protocols involved, the synchronization efforts involved among multiple threads in accessing the data, the dynamic handling of requests from clients etc used by the SCS.

The highlight of SCS is that it supports real-time without placing any restriction on the ACSS execution status. Also, it does not restrict the support packages of ACSS to run in the same computer without compromising its real time performance capabilities for the first time in the legacy of ACSS.

The Spacecraft data acquired by the SCC is required by several external and analysis packages of ACSS for display / plotting or automated performance verification. The objective of this paper is to describe the salient features, design scheme and architecture of the Spacecraft Checkout Server for Data Distribution System (SCS) which is the central agency in distributing the acquired data from Spacecraft to the support packages as well as other external agencies; which use this data for further analysis.

The requirement for a single Spacecraft Checkout Server for Data Distribution System as part of ACSS came up to maintain the Spacecraft data integrity, by preventing multiple agencies accessing the data in parallel; as well as to prevent the software management from becoming cumbersome. The SCS acts as a nodal point for data requirements for all packages which uses a modularized approach for data transmission to different applications. This approach in turn provides for a better platform to add any new analysis package. To realize the SCS, various design options were worked out and evaluated. The finalized design strategy is discussed in detail in this paper.

ACSS acquires two streams of telemetry in real-time. The data acquired is available in shared memory for all the tasks for processing, analysis and archival in real-time. The shared memory has two alternating buffers where one buffer is used for writing the data acquired and the other buffer for reading the data. Binary Semaphores are used to synchronize the operation of writing and reading into the buffers. SCS gets data from this shared memory region.

Concurrent TCP servers can accept multiple client connection requests on the same logical port simultaneously. There are different ways of designing a Concurrent TCP server.

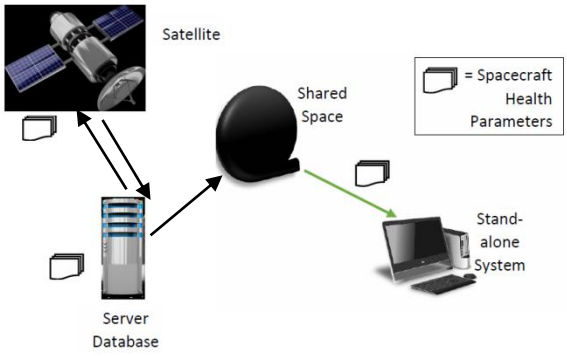
1. Iterative Server
2. Concurrent Server, one fork per client request
3. Pre-fork, each child calling accepts.
4. Pre-forking, file locking around accept
5. Pre-forking, thread mutex locking around accept
6. Pre-fork, parent passing descriptor to child
7. One thread per client request

8. Pre-threaded, mutex locking to protect accept
9. Pre-threaded, main thread calling accept.

The next section gives an insight into the existing system and its drawbacks/ disadvantages along with the proposed system and its advantages

II. RELATED WORK

A) EXISTING SYSTEM:

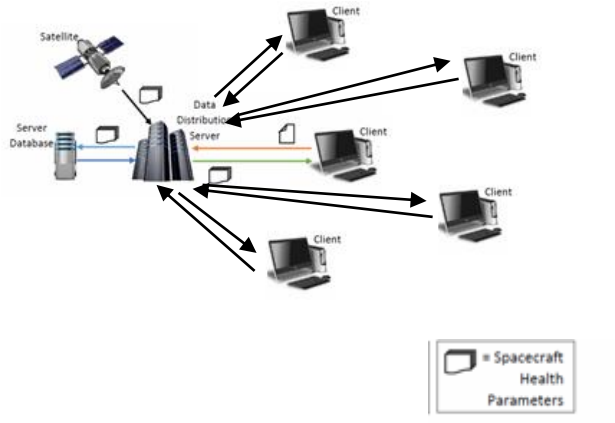


Satellites relay their overall health parameters to Spacecraft Checkout Server for Data Distribution Systems. These servers collect process and store the parameters in a shared space for access to all analysis packages. Any system accesses the shared space for the retrieval of parameters. [1]

Disadvantages:

1. Maintaining security of data.
2. Overhead of data processing due to absence of a processing system.

B) PROPOSED SYSTEM



Requirements:

1. Maintaining data integrity.
2. Preventing data contention when multiple agencies try to access data in parallel.

The proposed system is designed based on the TCP/IP client-server model[2]. The system is expected to relay real-time data to the user, as received by the client from the server. The server is capable of establishing multiple client connections at a time through multithreading. The client must send a request to the server to retrieve the relevant data

Any data analysis package shall request and retrieve the data from the SCS and tailor it according to the user requirement. [3].

There is a defined protocol for communication/transmission between the client and the server. The data is embedded in a well–defined frame format. The client system must be able to extract the relevant data (health parameters) from the received frames and display them appropriately. The user must be able to select the server he wishes to receive telemetry data from, following which the client will relay all requests from the user to the selected server.

Advantages:

1. A secure and reliable TCP/IP connection is established between the client and the server to receive real–time data without the intervention of any intermediary storage space
2. At any time, the client is aware of the 'freshness of data'. If a server is down, the client will be able to take the necessary steps to notify the user of the situation
3. Any data received by the client from the server is ensured to be the latest data available at the server. The user is never presented with old data, which makes the system more reliable for any critical processing

Some other key points that are present or utilized in this proposed system are follows:

1. When a client connection request arrives, the main thread calls accept function and passes the socket descriptor to one of the threads. This reduces the **Process control CPU time.**
2. When the socket is terminated due to an invalid request or due to any other reason, the child thread is removed.
3. The parent thread creates the socket and binds it to a defined port number. It listens for an incoming connection, accepts the connection. Once the connection is accepted, the pre created child thread is awakened and the socket descriptor is passed to the child thread. Further communication is handled by the child.
4. On invalid request a negative acknowledgement packet is formed and sent to the client. This packet embeds the reason for failure and the client connection is terminated.
5. If the request is valid, child thread accesses the shared memory, where the ACSS acquisition tasks hold the Telemetry frame data before being archived. The various semaphores attached to the

shared memory, gives the status of the telemetry data buffer status

6. In data available condition, data packets are formed and sent to the client at the frame rate. In the Data Break and Data Buffer in Wait mode, since the duration is more than 2 seconds, Dummy packets are formed and transmitted to the client at the rate of 2 seconds per packet.
7. Synchronization issues and contentions arose among threads during the transition state of Data Availability to Data Buffer in Wait Mode. Data Acquisition Process of ACSS goes to a blocked mode state when data is not received from the Data Acquisition System. A Timer is set and a SIGNAL is generated to change the mode of process from blocked to running state.
8. When multiple threads were trying to access the status of semaphore in the 'Data Buffer in Wait Mode' State, it led to a race condition. Multiple threads behaved differently and it led to unpredictable results. This was resolved by using a 'mutex' operation to lock a few variables while it is being updated.
9. The number of connections from the SCC for the different types of external agencies is fixed. SCS has to reject any further request for connection.
10. There is no restriction on the number of instances of each support/analysis packages which can connect to SCS for Spacecraft data. An upper bound is defined for the number of instances of each type of package.
11. The communication between support packages and SCS utilizes a request driven approach wherein the data requirement of the package is send to SCS via a request package and SCS packetizes the data accordingly
12. SCS accesses the shared memory to read the TM, TC and event information to packetize and transmit the data as per client requirement.
13. The SCS present in SCC has to run in parallel with the ACSS in real time scenario. Hence it was required to ascertain that SCS does not slow down the ACSS functioning as a whole, thereby affecting the real time performance.
14. Also the design of SCS had to be such that any failure in the client package or the server thread that services that client; does not affect the other threads or ACSS system performance.

This paper involves the following technologies or implementation methods.

1) SOFTWARE USED:

Qt [4,5] is a cross-platform application framework that is widely used for developing application software that can be run on various software and hardware platforms with little or no change in the codebase, while having the power and speed of native applications. Qt is currently being developed both by the Qt Company, a subsidiary of Digia, and the Qt Project under open-source governance, involving individual developers and firms working to advance Qt.[6,7,8] .

Programs created with Qt can have a native-looking graphical user interface (GUI), in which cases Qt is classified as a *widget toolkit*, but they are not limited to GUI. Qt is also used for developing non-GUI programs such as command and consoles for servers.

Qt uses standard C++ with extensions including signals and slots that simplifies handling of events, and this helps in development of both GUI and server applications which receive their own set of event information and should process them accordingly. Qt supports many compilers, including the GCC C++ compiler and the Visual Studio suite. Qt also provides Qt Quick, that includes a declarative scripting language called QML that allows using JavaScript to provide the logic. With Qt Quick, rapid application development for mobile devices became possible, although logic can be written with native code as well to achieve the best possible performance. Qt can be used in several other programming languages via language bindings. It runs on the major desktop platforms and some of the mobile platforms. It has extensive internationalization support. .Non-GUI features include SQL database
access, XML parsing, JSON parsing, thread management and network support.

2) PROTOCOL USED:

The Internet protocol suite is the computer networking model and set of communications protocols used on the Internet and similar computer networks. It is commonly known as TCP/IP, because it uses the most important protocols, the Transmission Control Protocol (TCP) and the Internet Protocol (IP),which were the first networking protocols defined in this standard. Often also called the Internet model, it was originally also known as the **DoD model**, because the development of the networking model was funded by DARPA, an agency of the United States Department of Defense.

TCP/IP provides end-to-end connectivity specifying how data should be packetized, addressed, transmitted, routed and received at the destination. This functionality is organized into four abstraction layers which are used to sort all related protocols according to the scope of networking involved.[1][2] From lowest to highest, the layers are the link layer, containing communication technologies for a single network segment (link); the internet layer, connecting hosts across independent networks, thus establishing internet working; the transport layer handling host-to-host communication; and the application layer, which provides process-to-process application data exchange.

The TCP/IP model and related protocol models are maintained by the Internet Engineering Task Force (IETF).

Since the above requirements are satisfied by the use of **TCP/IP** connections, and **Qt** is cross- platform software

it can be used as it can integrate with the other software modules that are being developed by different checkout operations.

CONCLUSION

The server must be able to:

1. Always service a request to client whenever a request comes.
2. Establish a TCP connection with the client.
3. Access real time/ archived data
4. Send data packets.
5. Packetize the data in a readable format.
6. Process the data that is acquired.
7. Identify each and every client with a unique ID.

The future enhancements that can be done are:

1. Identifying the satellite status like satellite shutdown, interpreting the general health parameters and signaling to the test engineer if any unexpected errors occurs.
2. The SCS only alerts by sending alert message to the test engineer. The future enhancement can be to take actions against those unexpected behavior.

REFERENCES

[1] 2008 IEEE Asia-Pacific Services Computing Conference The Spacecraft Automatic Testing System based on Workflow Dan Yu 1 , Gang Ye 1, Shilong Ma 1, Naixue Xiong 2, Laurence T.Yang 3
1 State Key Laboratory of Software Development Environment, Beihang University,Beijing 100191, China
2 Department of Computer Science, Georgia State University, Atlanta 30303, USA
3 Department of Computer Science, St. Francis Xavier University, Antigonish, NS, Canada{yudan,yegang,slma}@nlsde.buaa.edu.cn , nxiong@cs.gsu.edu , lyang@stfx.ca

[2] Autonomous Operations Through Procedure Automation Category – Autonomous Operations – II Authors: Terma:: R. Patricki, Vitrociset SpA (Italy): F. Croceii

[3] The Design and Implementation of Workflow Engine for Spacecraft Automatic Testing,JOURNAL OF COMPUTERS, VOL. 6, NO. 6, JUNE 2011

[4] "That Smartphone Is So Qt". *Ashlee Vance*. 16 February 2010. Retrieved 19 February 2010.

[5] **Jump up** "The Qt 4 Dance" (video). Retrieved 19 February 2010.

[6] **Jump up** Lydia Pintscher (21 October 2011). "KDE Applauds Qt's Move to Open Governance". KDE.News. Retrieved 8 May2013.

[7] **Jump up** Meyer, David (24 October 2011). "Nokia gives Qt open-source governance". ZDNet. Retrieved 8 May 2013.

[8] **Jump up** Knoll, Lars (6 August 2014). "Defragmenting Qt and Uniting Our Ecosystem".