

Integration of Application Code Simulations for Data exchange

Basankar Vikas¹, Dr. S. P. Metkar², Manoj Mane³, Bhuvaneshwar Kanade⁴

¹*Dept of Electronics and Telecommunication, College of Engineering,
Pune, INDIA*

²*Dept of Electronics and Telecommunication, College of Engineering,
Pune, INDIA*

³*Whirlpool of India Limited, Pune, INDIA*

⁴*Whirlpool of India Limited, Pune, INDIA*

¹*vikasbasankar@gmail.com, ²metkars.extc@coep.ac.in*

³*manoj_mane@whirlpool.com, ⁴bhuvaneshwar_k_kanade@whirlpool.com*

Abstract: *In every product development industry, automation plays a key role in increasing the throughput of the company and providing better planning in the product development and improved production quality. It is very necessary to find a solution to interdependencies during the product development process. During simulation-based analysis of a product, it is required that the need for actual hardware of the product is to be eliminated. Because of this, the functionality of the actual hardware can be analysed by using software using simulations. If simulations of different products are running, the data is to be exchanged between different simulations effectively. It can be considered as simulating data exchange, as it is implemented in the hardware form. A proper and suitable method is to be used to have this goal achieved. This paper will address the integration approach for application code simulations or programs that are built to perform specific task.*

Keywords: *Server, Client, Application Program, Simulation, Source, Destination*

1. Introduction

Integration of two simulations or application programs enables sharing and exchanging the processed data with each other. The integration approach might use different data exchange methods for exchanging the data. As this may be considered as communication between two processes that reside in two different individual programs, there is a need for such methods which can transfer the data between these processes. There are few methods that help to achieve the goal described earlier that are dependent upon different concepts like Inter-process communication, resource Sharing. As both the simulations are supposed to run on the same machine, a specific method is needed which can make the exchange of data while two simulations are running simultaneously. The method should also ensure that two simulations stay connected throughout the process and exchange the data.

2. Methodology

A. Proposed System:

In the integration approach, processed data need to be exchanged between two application programs which may act as two simulations of a particular system. The proposed system focuses on a communication Bridge, Communication bridge tries to implement the specific method between two application programs to transfer the data packets. The method should ensure no data loss or data corruption. The method should be also capable of establishing and retaining the connection between two application programs. The following figure shows a block diagram of the system which uses a particular communication method to exchange the data between two application programs. These two application programs may be controlled individually by the User control which may include a graphical user interface or any application program interface.

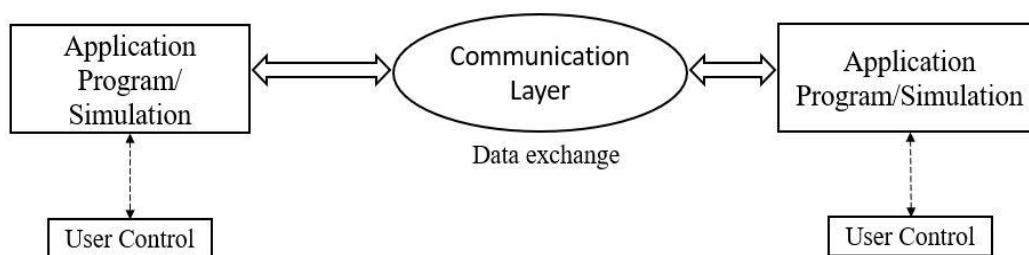


Figure 1. Block Diagram of Proposed System

B. Communication Methods:

1. File handling:

A File enables us to store a large amount of continuous data in different forms. File handling provides some operations such as File creation, opening, closing, etc. A file can be handled by using Read, Write and Append modes. These modes can be used according to requirements. File handling helps in transferring the data from one program to another program by becoming the intermediate medium. This intermediate file is created, and data is written to it. Another program reads that data and uses it for its processing. The limitations of this method are lack of synchronization, processing delay, and the problem of data corruption. Whenever data is to be transferred, the file is opened and when data is written to it, it needs to be closed. This process may create a lack of synchronization, also in this process data loss and corruption are possible. Figure shows Source and destination as two application programs. An intermediate file is created for storing the processed data. This file may be of any type text, binary or CSV.



Figure 2. File Handling Method

2. Inter-process communication using Named Pipe:

The pipe is a medium of communication between two or more than two related or interrelated processes which may or may not run on the same machine. The data exchange is achieved by one application program writing the data into the pipe and another application program reading from that pipe. Pipes are used for communication between related processes and in the case of unrelated processes advanced version of pipe called Named pipe is used.

One-way communication using Pipe:

In this case, a source can either read or write at a time. The same is the case for a destination.

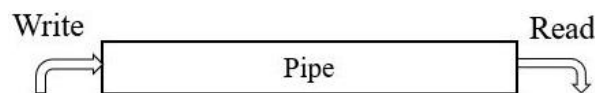


Figure 3. One Way Communication

Two-way communication using Pipe:

Two-way communication uses two pipes to achieve simultaneous bi-directional

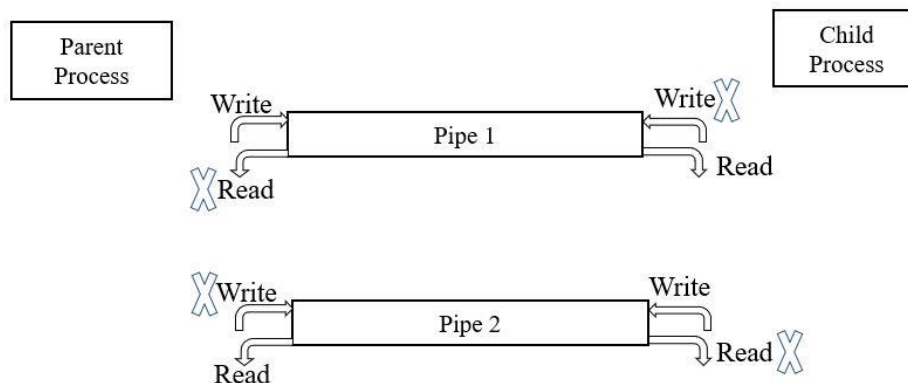


Figure 4. Two Way Communication

In the case of the unrelated process (the process in two different programs), to communicate between these processes, another type of process called Named Pipe is used. It is also called First-in-First-Out. It is an extension of traditional pipe.

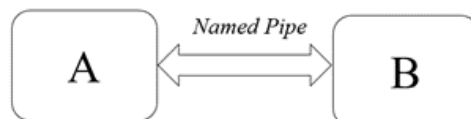


Figure 5. Application program A and B connected through Named Pipe

Algorithm:

Server:

1. Create named Pipe.
2. Connect Named Pipe
3. Write File or read File.
4. Flush file Buffers
5. Disconnect Named Pipe
6. Close handle

Client:

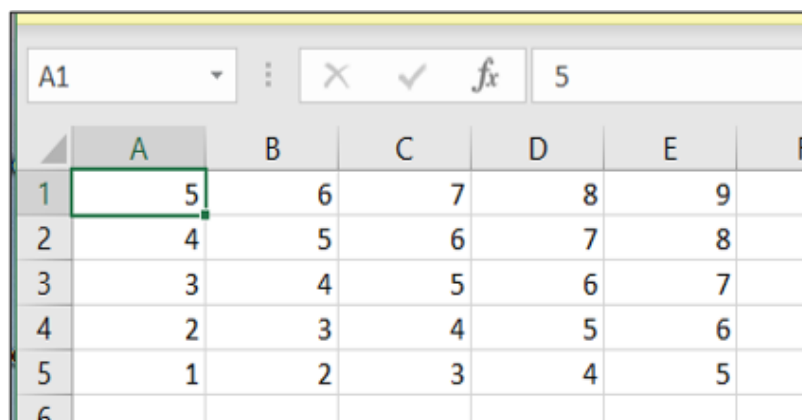
1. Create File or Call Name Pipe
2. Write/Read.
3. Close handle

The Named pipe method enables the integration of two independent programs by creating client-server pair. Hereafter HMI simulation is referred to as server-side and ACU simulation is taken as the client-side. One program is considered a server, and another is as a client. Communication is established between server and client with the help of the Named pipe. After successful connection, data can be exchanged by using Read and write functions.

3. Implementation and Results

A. File Handling

For the communication layer, the file handling method creates a file that acts as a communication medium between two programs. This intermediate file may be of different types such as binary, hex, CSV, text, etc. The sample data is written two the file the CSV file is as shown below.



	A	B	C	D	E	F
1	5	6	7	8	9	
2	4	5	6	7	8	
3	3	4	5	6	7	
4	2	3	4	5	6	
5	1	2	3	4	5	
6						

Figure 6. CSV File data written by application code

The console output of the source and destination side application program is as shown below, which indicated data being sent to the destination through an intermediate file.

```
feedback from Destination:No data
Enter the choice
5
Data Sent To Destination Sent!
Enter the choice
4
Data Sent To Destination Sent!
Enter the choice
3
Data Sent To Destination Sent!
Enter the choice
2
Data Sent To Destination Sent!
Enter the choice
1
Data Sent To Destination Sent!
```

Figure 7. Console Window of Server Side

```
5,6,7,8,9,
4,5,6,7,8,
3,4,5,6,7,
2,3,4,5,6,
1,2,3,4,5,

Process returned 0 (0x0)   execution
Press any key to continue.
```

Figure 8. Console Window of Client Side

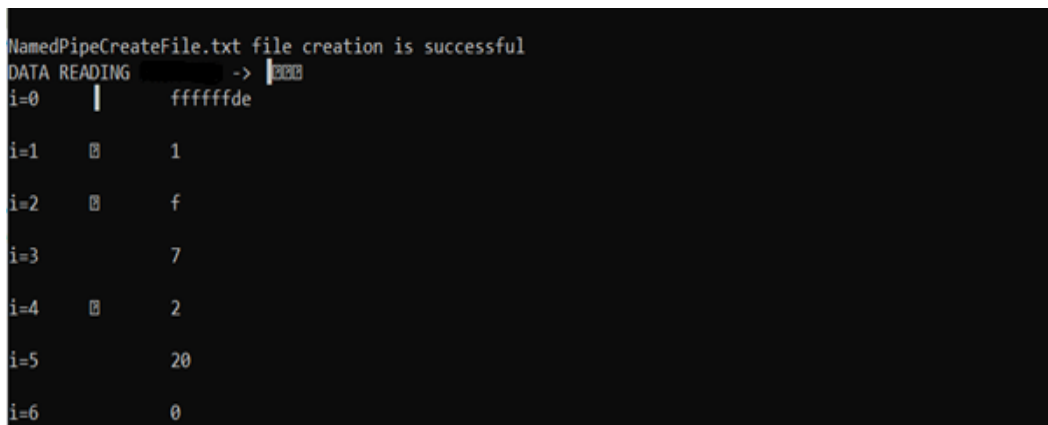
The limitation of this method is that it is not suitable for synchronous communication because for every data transfer cycle file is to be opened and closed. A lot of processing time is consumed for opening or

creating the file in the desired mode such as read, write, or append. It may also lead to data loss and corruption because of random access of the file by the different application programs.

B. Inter-process communication using Named Pipe Method:

The inter-process communication method using the Named pipe method will connect both source and destination with the help of named pipe or FIFO. On the successful connection of both source and destination the data read and write starts by calling the respective functions. It is important to note that the application codes will run on the same machine.

The console window showing the data packet received is as shown below.



```
NamedPipeCreateFile.txt file creation is successful
DATA READING -> [input]
i=0 | fffffffde
i=1 | 1
i=2 | f
i=3 | 7
i=4 | 2
i=5 | 20
i=6 | 0
```

Figure 9. Console Window at the Client

The console window showing the connection status of the Named pipe and received data from the client is shown below.



```
NamedPipe creation is successful
ConnectNamedPipe connection is successful

Write is successful
FlushFileBuffer successful
ReadFile is successful
DATA READING FROM CLIENT -> ABCDEFGGIJKLMNOPQRSTUVWXYZ[\\]^_`ab
Write is successful
FlushFileBuffer successful
```

Figure 10. Console Window at the Server

As this Named pipe communication program is connected with two different programs at the same time, it may lead to loss of synchronization. If both the application programs make use of reading and writing operations synchronously then data gets transferred effectively.

Following table illustrates the comparison between the methods discussed above. It discusses integration method used to joint two application codes, advantages, disadvantages of each method.

Table 1. Comparison of Two Data Exchange Methods

Sr. No	Method	Integration medium	Advantages	Disadvantages
1	File Handling	File (CSV, text, binary, etc)	The process is transparent in terms of execution. Easy for bug finding and elimination.	Only half duplex data transfer is possible, efficiency is less.
2	Inter-process communication Using Pipe.	Named pipe	Programming Complexity for Bi-Directional data transfer is less.	As the process takes place through the kernel, the processing time is more.

4. Conclusion

In this work, for the integration part first method is file handling. In file handling, data exchange is easy for low-level asynchronous simplex communication systems. for event-based application codes, this method may not be suitable because every time of data transfer file is to be opened and closed. Also, data loss and corruption are possible because random nature of file access by different application codes. This leads to a lack of synchronization and efficiency.

Another inter-process communication using a Named pipe creates a dedicated connection between two different application programs. It ensures if the connection is established between two application codes or not. Because of this, the working efficiency is more in this method. Some synchronization issues may arise which may also depend on data transfer frequency and pattern of the application program simulation to which this method is used.

5. References

- [1] Raafat Feki and Edgar Gabriel. "On Overlapping Communication and File I/O in Collective Write Operation" In the IEEE Conference on International Parallel and Distributed Processing Symposium Workshops, IEEE 2020 May 18-22.
- [2] Xiao-hui Cheng and Liang Zhang. "A Research of Inter-process Communication Based on Shared Memory and Address-mapping" In the International Conference on Computer Science and Network Technology, IEEE 2011 Dec 24-26
- [3] Ming Liu, Zhonghai Lu, Wolfgang Kuehn, Axel Jantsch. "Inter-Process Communication using Pipes in FPGA-based Adaptive Computing" In the Annual Symposium on VLSI, IEEE 2010 July 5-7

- [4] J.A.Williams, N.W. Bergmann, X. Xie. "FIFO Communication Models in Operating Systems for Reconfigurable Computing" Proceedings of the 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, IEEE 2005 April 18-20
- [5] Guofu Huang, JiapingWu, Jianhua Shen. "A Design and Implement of Virtual Device System Architecture" In Eighth IEEE International Conference on Embedded Computing; IEEE International Conference on Scalable Computing and Communication, IEEE 2009 September 25-27
- [6] YUAN Hong, GU Ping-ping. "Application of Windows Inter-process Communication in Software System Integration" In the International Conference on Intelligent System Design and Engineering Application, IEEE 2010 October 13-14
- [7] Rolou Lyn R. Maata, Ronald CordovaBalaji Sudramurthy, Alrence Halibas. "Design and Implementation of Client-Server Based Application using Socket Programming in a Distributed Computing Environment" In International Conference on Computational Intelligence and Computing Research, IEEE 2017 December 14-16
- [8] Ming Xue, Changjun Zhu. "The Socket Programming and Software Design for Communication Based on Client/Server" In the Pacific-Asia Conference on Circuits, Communications and System, IEEE 2009 May 16-17
- [9] Hosein Marzi, Larry Hughes, Yanting Lin. "Optimizing Inter-process Communication for Best Performance in Real-Time Systems" In the Canadian Conference on Electrical and Computer Engineering, IEEE 2011 May 8-11
- [10] Qiao Kang, Sunwoo Lee, Kaiyuan Hou, Robert Ross, Ankit Agrawal, Alok Choudhary, Wei-keng Liao. "Improving MPI Collective I/O for High Volume Non-Contiguous Requests with Intra-Node Aggregation" IEEE Transactions on Parallel And Distributed Systems, Vol. 31, 2020 November 11, pp. 2682 - 2695
- [11] Windows Development Kit: <https://docs.microsoft.com/en-us/windows/win32>