ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 3, March 2014

Real Time Communication Simulator

Prof. Geeta Patil[1], Amit Soni[2], Vinay Singh[3], Sumit Dasila[4], Shashi Mohan Kumar[5]

M.E (E & TC), Information Technology, Army institute of Technology, Pune, India[1]

Information Technology, Army Institute of Technology, Pune, India[2,3,4,5]

Abstract: To develop communication simulator over wired network by modifying UDP protocol. The scope of work is to improvise the reliability, latency of system during data (like text, audio, video) transfer to make it work for real time applications. A communication system for robotics application should allow fast exchange of real-time data for low level robot control and large amount of data for high level algorithms like intelligent vision or map building. Ethernet seems to be the perfect candidate for the low level communication interface because of its large diffusion; high bandwidth and low cost .Real time system are being use increasingly in control applications such as in automobiles, aircraft and process control. Real-time communication protocols are designed in order to satisfy basic requirements of these systems such as reliability, safety and in time message delivery. Time update of real-time data is crucial requirement of such critical systems which has to be facilitated by these protocols. RDM (Round Data Mailer) and RDM+ are two new protocols that have been designed to fulfill the requirements of distributed real-time systems. The project is useful to achieve real time application in many things like audio, video communication.

Keywords: Communication, Simulator, ROV, HMI

I. INTRODUCTION

Fast and reliable communications is a basic requirement in almost all modern applications, but the real time systems take it to extreme and require real time responses from the network. In some cases, the requirements are so tough that may require special hardware to achieve desired performance. During the last decade, new technologies have emerged to support the growing demand, but their prices are high and availability is low. As a result, most of the systems are still using TCP/IP protocols stack over Ethernet backbone. Moreover, most of the programmers are still developing with well-known synchronous sockets API. This article explores usage of these dominating technologies in the real time systems and shows how to design and develop communications within such systems. In particular, it proposes reliable protocol over UDP transport layer that gains less than one millisecond average round trip time (RTT) and processes dozens of thousands of messages every second at each and every port in network.

Most earlier work in the area of supporting real-time communication over Ethernet focused on modifying the Ethernet MAC sub-layer so that a bounded channel-access time may be achieved, thus making hard real time communication possible. These approaches are very costly compared to the widely-used current Ethernet standard. In the quest for real-time communication over Ethernet, several techniques were developed and used by both industry and academia, some of which are briefly referred to in this section, grouped according to their main features. CSMA/CD based protocols achieve real-time behavior over shared Ethernet relying solely on the original CSMA/CD contention resolution mechanism [1], taking advantage of the fact that the probability of collision between concurrent nodes is closely related to the traffic properties such as the bus utilization factor, message lengths and burstiness. In CSMA/CD mechanism, when an application requests for a frame and if at that time bus is free then only the network interface cards (NICs) carry out frame transmission. As soon as transmission is started, NICs continue sensing the bus for a time slot to detect a collision. If it occurs, all the stations abort the ongoing transmission, issue a jamming signal, and wait for a random time interval before repeating the process. An exponentially increasing wait time is used to reduce the probability of chained collisions on heavy loaded networks.

Real time networking is a non-easy task, so before move any further, system connectivity requirements must be analyzed. The right way to do it is to map and classify traffic in the system. First classification to be made is messages sizes and their latency requirements. In many cases, these two parameters are sufficient to choose transport protocols and network topology.

We would be dealing with two modules for the real time communication as mentioned below:

Human machine interface (HMI) is the unit that is responsible for displaying the interface to interact with the ROV via System Module.

Remotely Operated Vehicle (ROV) receives the messages from Communication Module and perform required task.

As packet transmission using TCP takes much time we are going to use UDP transmission, but UDP transmission is not reliable. So, we need to exploit UDP and make it reliable to fulfill the following features: Low Latency, Receive and deliver message to application as soon as possible, Send application message as soon as possible, Reliability, Avoid losing Message, In case of lost message, detect and recover it (retransmission of the message).

II. PROCESS DEVELOPMENT

Sending Module:

When the sending operation is performed by application, the protocol engine does the following. First it assigns ID to the message and passes it to the sending module which, in turn, transmits the message to its destination. Then it adjusts retransmission time out to the message and pushes it into queue of acknowledge pending messages (there is queue per

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 3, March 2014

channel).Whenever retransmission time out of the message expires, the protocol engine retransmits the message (hands it over to the sender module) and sets a new time out for the message. If acknowledge is received, the protocol engine is notified about it by receiver module. It examines the acknowledge pending messages and if acknowledge ID matches one of them, it removes the message from the queue and notifies application about successful delivery of the message (if application requested such notification). Figure 3.1 shows flow chart of outgoing messages processing in protocol engine. The idea is quite simple. In order to know if message has arrived to its destination, it must be acknowledged by the receiver. The sender is storing the message in its memory as long as it is waiting for acknowledge. Once it gets acknowledge, the message can be discarded. In case the message is not acknowledged during some period of time, it must be retransmitted.

The UDP-RT protocol defines maximum message delay as a time that application permits for message delivering. Message latency is the actual time that it takes to transmit message; its value depends on network hardware and peers operating systems behavior. We assume that the maximum message delay is significantly bigger than average message latency and there is enough time for retransmissions. As a result, the timeout value of the retransmission timer becomes the most important parameter in the protocol configuration. To set a reasonable retransmission timeout, sender calculates average message RTT.

It writes timestamp into message header and receiver copies the timestamp into corresponding ACK message, so the sender can calculate the RTT upon acknowledge receiving. Given the average RTT, the retransmission timeout could be set to average (RTT) +, where a compensation for RTT jitter is. Since the distribution of messages RTT-s could be considered as normal, the could be set to 2*, which covers over 97the average RTT retransmission model works well as long as number of channels is relatively small and losses of messages are sporadic over time and channels.

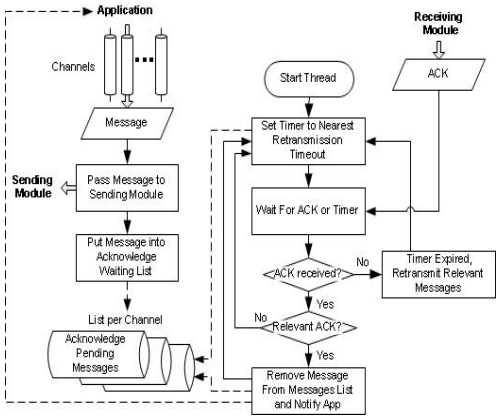

Fig 1: Message Retransmission Timer

Since the distribution of messages RTT-s could be considered as normal, the could be set to 2*, which covers over 97the average RTT retransmission model works well as long as number of channels is relatively small and losses of messages are sporadic over time and channels. Indeed, if some random message is lost, the protocol will retransmit it after average (RTT) + timeout. Now consider a situation where message is lost over and over again. Once it exceeds its maximum message delay value, the protocol should not attempt to retransmit the message and, instead, proceed according to the message dropping policy setting. The setting allows application to configure the UDP-RT behavior for expired messages; the protocol can either silently discard these messages or declare system failure. By default, sender is starting DATA messages ID-s from zero and assigns IDs to the following messages in increasing order. The receiving side propagates the ID along with the message to the application.

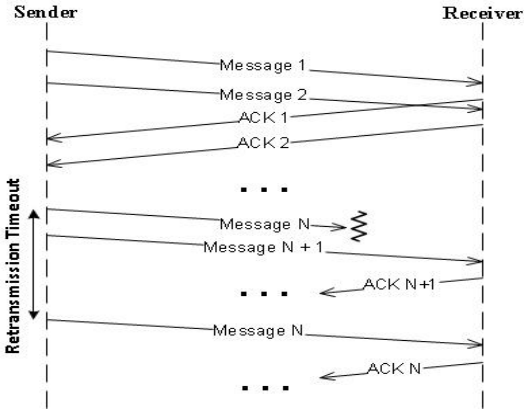

Fig 2: Resetting Message Numbering

The application may reset the counter to any value any time. The reset feature is usually used by application to start new processing sequence or prevent messages ID overflow. When application asks to reset ID counter, the protocol engine discards all acknowledge pending messages and waits enough time to allow the system to clean itself up from already sent messages. The waiting time is defined by reset waiting time channel setting. Then the sender issues RESET message. The receiver cleans up the heap of received messages and resets receiving ID counter to the requested value. The RESET message is acknowledged in the same way as DATA messages. The RESET command is also propagated to receiving application, so the application could properly handle the RESET notification. For example, it may request RESET from its protocol engine and, thus, reset traffic in both directions on the given channel.

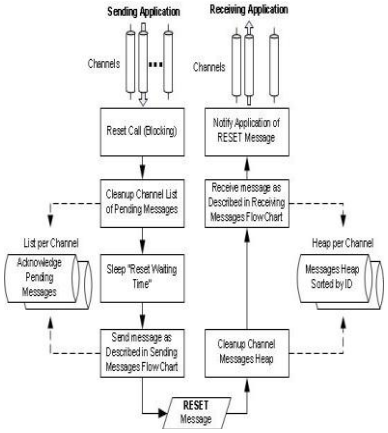

Fig 3: Receiving Module

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 3, March 2014

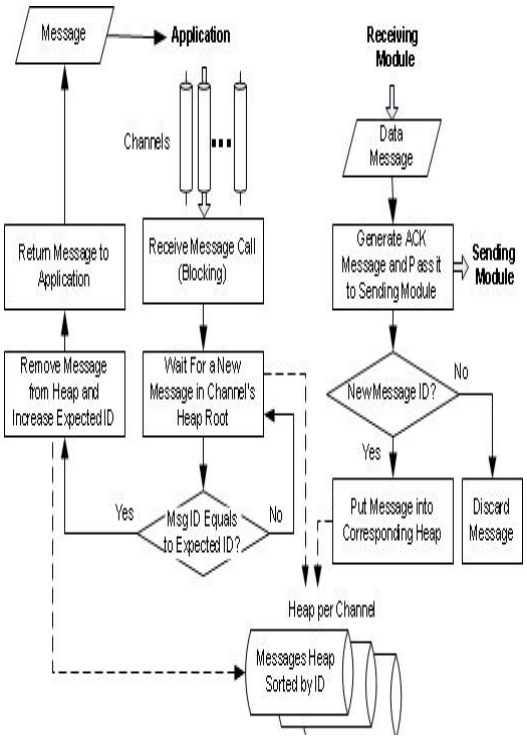

Fig 4: Sending module

Receiving Module:

The receiver module passes received messages to the protocol engine. If this is an acknowledge message then it is processed as described in the previous section. If this is a data message then the protocol engine must acknowledge it. So it generates acknowledge message and passes it to the sender module. Then it stores the received message till it is requested by application.

II. EXPERIMENTAL RESULTS

Develop a real time communication protocol simulator: Simulating a communication module with features like Simplifies debugging and testing makes it economically feasible. Improving and enhancing communication features like: Latency, Reliability. Main objective to be achieved are: Simulate a protocol engine, Decrease latency to avoid delays, Increase reliability.

UDP modify reason?

To add features like: Packet loss detection, retransmission of lost packets, acknowledgement, ordering of messages, maintaining Message priority.

By modifying UDP, we will be giving the UDP features of TCP thereby keeping the features of UDP. So, we can provide real time constraint for the message being transferred from the sending module to receiving module. We proposed the idea of making the UDP reliable along with its speed which it already delivers in communication between two modules.

UDP modify?

Assign priorities to messages, Implement sequencing of packets, Implement acknowledgement queue, Implement sending queue, Implement receiving queue.

Maintain a priority queue at ROV: If two commands receive at the same time, execute one with lower priority. If lost packet arrive at same time.

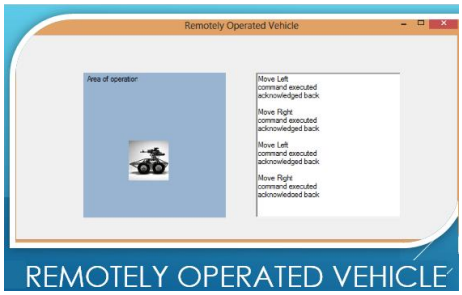

Fig 5: Remotely Operated Vehicle

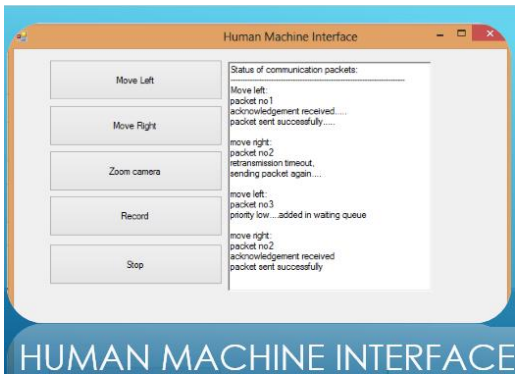

Fig 6: Human Machine Interface

III. CONCLUSION

To develop communication simulator over wired network by modifying UDP protocol. The scope of work is to improvise the reliability, latency of system during data (like text, audio, video) transfer to make it work for real time applications.

ACKNOWLEDGEMENT

First of all I would like to express our profound sense of gratitude towards our guide Miss Nirja Thakur, Defence Research and Development Organization, for his valuable guidance, support and encouragement throughout the period this work was carried out. His readiness for consultation at all times, his educative comments, his concern and assistance even with practical things have been invaluable. I would also like to convey ours sincere gratitude and indebtedness to our H.O.D. Mrs. Sangeeta Jadhav, Department of Information Technology, Alandi Road, Dighi Hills, who bestowed their efforts and guidance at appropriate times without which it would have been very difficult on our part to complete the seminar work. I also thank to the Dr. V.P. Gosavi, Principal of Army Institute of Technology for providing me all the necessary facilities to carry out the seminar work.

REFERENCES

[1] Paulo Pedreiras, Paolo Gai, Lus Almeida, and Giorgio C. Buttazzo, \FTT-Ethernet: A Flexible Real-Time Communication Protocol That Supports Dynamic QoS Management on Ethernet-Based Systems", IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, VOL. 1, NO. 3, AUGUST 2005.

[2] Abdelhamid Helali, Adel Soudani, Salem Nasri, Thierry Divoux, \An approach for end-to-end QoS and network resources management", Computer Standards Interfaces 28 (2005) 93 108.

[3] Seok-Kyu Kweon and Kang G. Shin, "Achieving Real-Time Communication over Ethernet with Adaptive Traffic Smoothing", Real-Time Technology and Applications Symposium, 2000.

[4] Paolo Ferrari, Alessandra Flammini, Daniele Marioli, and Andrea Taroni,\A Distributed Instrument for Performance Analysis of Real-

Time Ethernet Networks", IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, VOL. 4, NO. 1, FEBRUARY 2008.

[5] J. Loeser, H. Haertig, \Low-latency hard real-time communication over switched Ethernet", in: Proceedings of the 16th Euromicro International Conference on Real-time Systems (ECRTS2004), Catania, Italy, June 30 July 2012.

[6] D. Ferrari, D.C. Verma, \A scheme for real-time channel establishment in wide area network", IEEE Journal on Selected Areas in Communications 8 (3) (1990) 368379.

[7] Heejo Lee. ,Toda, K. , Jong Kim ,Nishida, K. , Takahashi, E. , Yamaguchi, Y., \Performance comparison of real-time architectures using simulation", Real-Time Computing Systems and Applications, 1995. Proceedings. Second International Workshop, pp-150-157.