ADRRI JOURNAL OF ENGINEERING AND TECHNOLOGY

*ADRRI JOURNALS (***www.adrri.org***)*

pISSN: 2026-674X ***ISSN-L***: 2026-674X VOL. 2,No.1(1), September, 2014

Improving the Communication Functionality of User Datagram Protocol (UDP) For Real Time Messaging and Application Processing.

Anibrika Bright Selorm Kodzo

Computer Science Dept., Koforidua Polytechnic, box 981 Koforidua, E/R, Ghana, West Africa

Email: skbselorm@yahoo.com **Tel**: +233 0506618729

Received: 25th August, 2014 **Revised**: 24th September, 2014 **Published Online:** 30th September, 2014

URL: http://www.journals.adrri.org/

[**Cite as**: Anibrika, B. S. K. (2014). Improving the Communication Functionality of User Datagram Protocol (UDP) For Real Time Messaging and Application Processing. ADRRI Journal of Engineering and Technology, Ghana: Vol. 2, No. 1(1), Pp. 17-24, **ISSN**: 2026-674X, 30th September, 2014.]

Abstract

Communication is an essential part of human existence which enables us reach friends and relatives at distant or different geographical locations and boundaries. This communication is however based on certain rules and procedures without which communication would be unsuccessful. Protocol defines a set of rules that enables communication with 99.99 efficiency and effectiveness. Therefore one protocol that enables real time communication between devices and computers is the User Datagram protocol located in the transport section of the Open System Interconnection (OSI) model. On other hand, UDP has not performed well in the transport of short messages between devices especially when same messages are sent to different devices and ports through Application Programming Interfaces (APIs) and sockets that enable programmers to harness the full power of transport protocol. Therefore this paper seeks to identify the various limitations of the UDP and proposes a new UDP protocol known as ***UDP-RT***, and that is User Datagram Protocol for Real Time communication. This protocol is an extension of UDP that is robust and has the capacity to manage and control message communication in real time that ensures the best delivery and security at all times and also proposes reliable protocol UDP over UDP protocol layer that gains one millisecond average round trip time (RTT).

Keywords: real time, protocol, application, messages, socket, programming

INTRODUCTION

Real time networking and communication implementation is non-easy task, despite new emerging technologies that are meant to enhance data transfer from one source to the other especially between devices in close proximity, there still exist bottlenecks or delays that cause undesired results and even sometimes leads to attenuation and subsequently data loss in the system. Long messages may contain database tables, picture of different format like JPEG and bit mapped, sound, video and other parameters and data that requires movement from one end of the computer system to the other end without compromising on its performance .When analyzing the performance of protocols, **message sizes** and their **latency requirement** cannot be underestimated (Michael Pan, 2012). How these requirements mentioned above can be enough in selecting transport protocols and network topology. In selecting transport protocols focus would be on TCP/IP (Transmission Control Protocol) over Internet Protocol.

Furthermore two important protocols to be looked at are the TCP (Transmission Control Protocol) and UDP (User Data gram Protocols) protocols respectively (Cisco Press, 2001).Fast and reliable communication is a basic requirement for all modern application , however real time system take it to the extreme and also require real time responses to the extreme and also require responses from the network that occur in real time, there are some special cases that real time communication require special hardware to achieve desire results and performance. New technologies have emerged that sought to support the growing demand but their prices are high and their availability is quite low, for that matter most of the systems are still employing with well –known synchronous sockets Application Programming Interface (API).This paper therefore seeks to explore this dominating technologies in real time systems and shows how to design and develop secure communication within such systems.

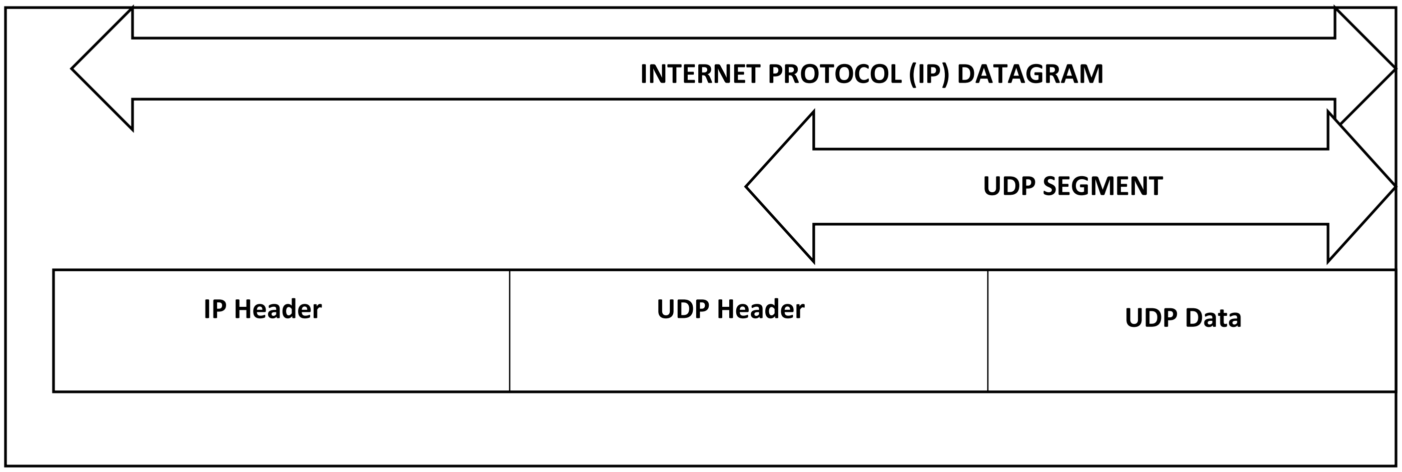
In this paper, there is a proposition of new reliable protocol over User Datagram Protocol (UDP) transport layer. This UDP transport layer is a versatile protocol that is capable of gaining less than one millisecond of average Round Trip Time (RTT) as well as processing of dozens of thousands of messages every second at each level and every port in the network. Before designing such systems, system requirements are analyzed to determine appropriate system specification as well as software and hardware requirements. Since the focus of this paper is on TCP/IP, TCP and UDP is compared and analyzed and choice is made between them based on message sizes and latency requirements and also the effect of these protocols on the bandwidth in transmitting messages.

LITERATURE REVIEW

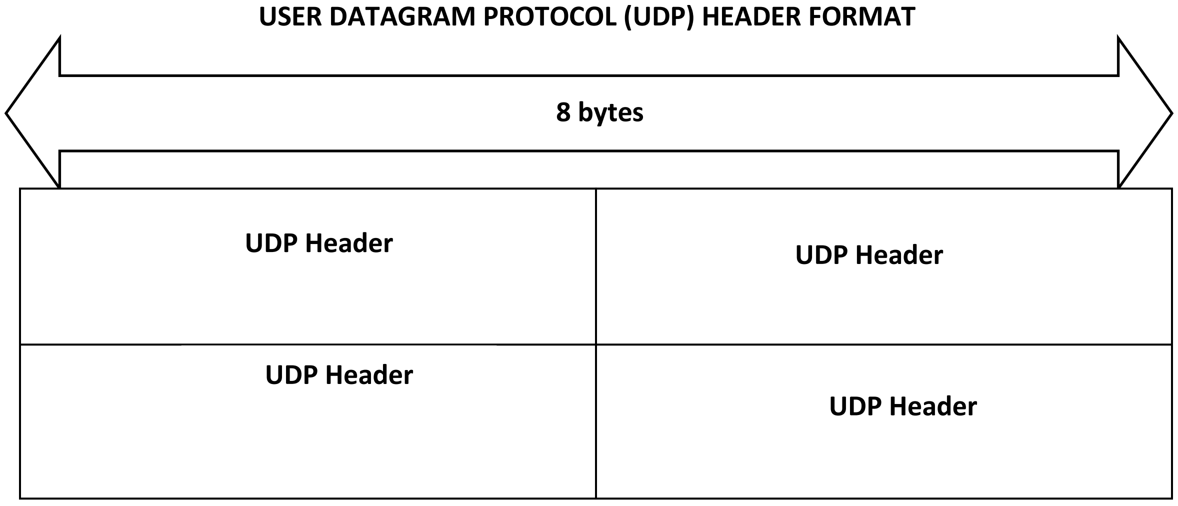
This section gives an over view of the research and the prior research studies that have explored in regards to this research stream, it considers the demands such as what analysis has been before and what results have been produced (Burg and Kenny, 2000).User Datagram Protocol (UDP) provides a mechanism for applications to send encapsulated raw Internet Protocol (IP) datagrams and send them without having to established a connection. Thus, it adds low overhead but requires the application to take responsibility for error recovery (Pornpan Tadthong, 1999).In addition, UDP is functionally a transport layer protocol and it is also connectionless and does not and does not provide a reliable transport. On the other hand, UDP gives an application a direct access to the datagram service of the IP layer. The multicast and broadcast services are available by using UDP

(Cisco Networking Academy, Program, 2001). The UDP is called user datagram and has no flow control mechanism. UDP only attempt at error control is the checksum. The UDP header contains the source port number, destination port number and total length and checksum (Palito Anthony, 2002).

USER DATAGRAM PROTOCOL (UDP) ENCAPSULATION FORMAT



(Cisco Networking Academy Press, 2002)

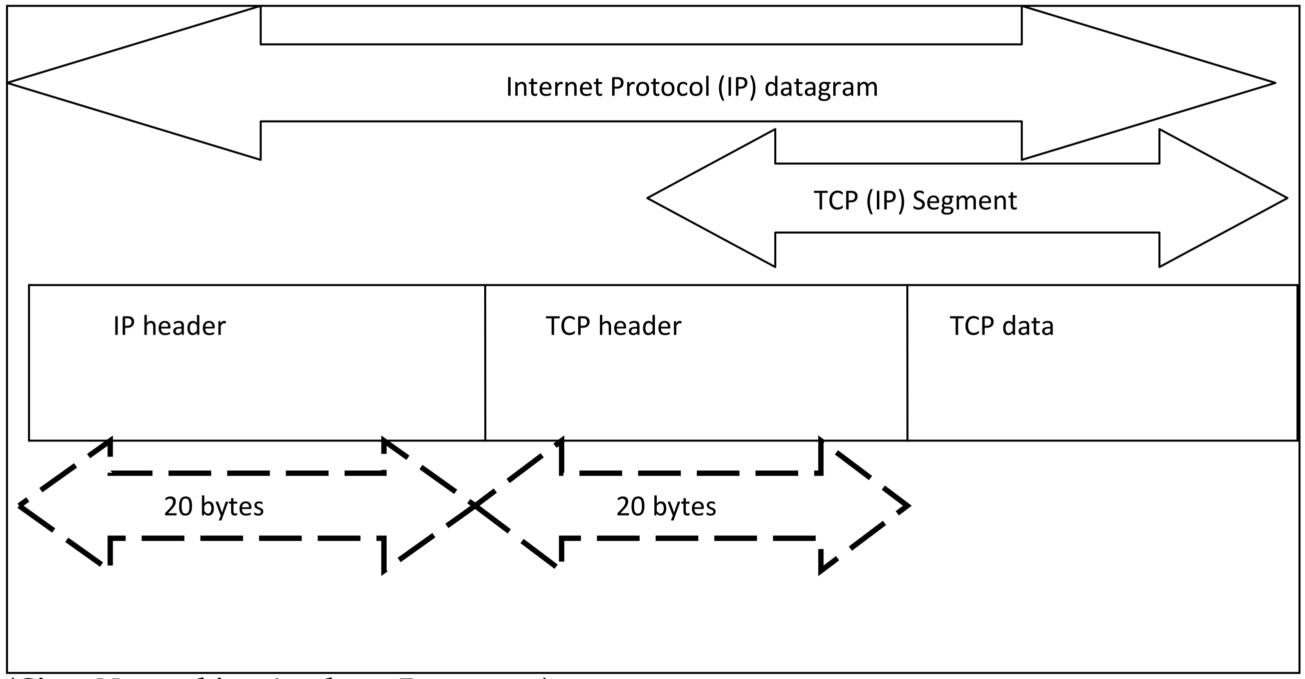


(Cisco Networking Academy Press, 2002)

TRANSMISSION CONTROL PROTOCOL (TCP)

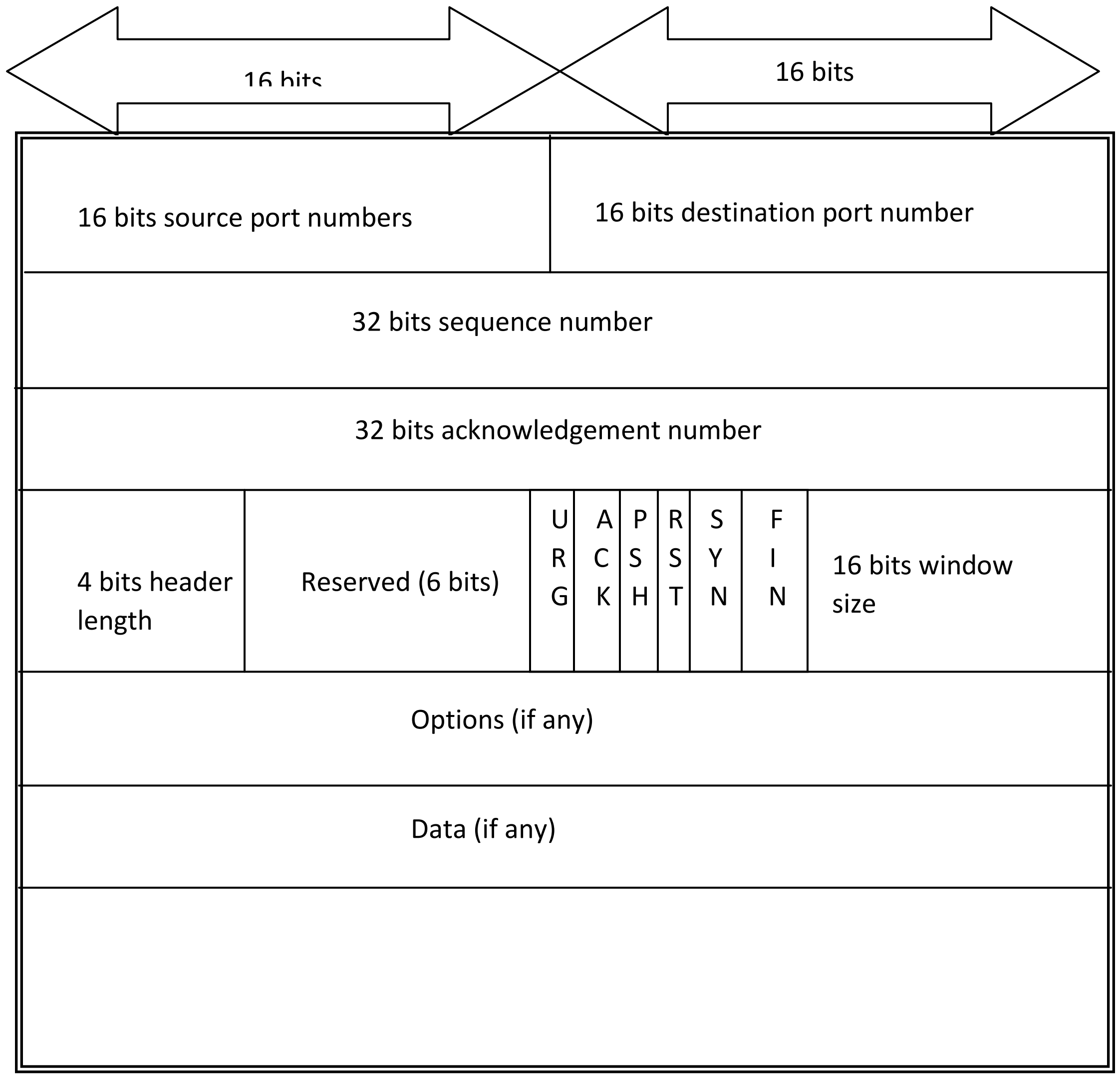
TCP is the most important connection-oriented protocol .It provides reliable information transfer service for higher layer applications (Cisco Networking Academy, 2001). Furthermore, TCP is connection establishment, error-recovery and have a flow control avoiding errors. Error recovery is implemented through retransmissions and packet reordering. TCP normally provides two main functions for the dynamic flow control. First a retransmission timer is used for determining a lost packet at the sending TCP host. The second approach is to control the window size for the sent but acknowledged packets. In addition to the stuff above, TCP have a header that contains various fields including the source and destination ports, sequence and acknowledgement numbers, window size, TCP flags, urgent pointer and reserved bits (Karen Kent Frederick, 2001).

ENCAPSULATION OF TCP IN AN INTERNET PROTOCOL (IP) HEADER



(Cisco Networking Academy Press, 2002)

TRANSMISSION CONTROL PROTOCOL (TCP) FORMAT



(Cisco Networking Academy Press, 2003)

TCP also provides reliable connection service to pairs of processes. It does not assume reliability from the low-level protocol such as IP. TCP assigns a sequence number to each byte transmitted and expects a positive acknowledgement (ACK) from the receiving TCP. If the ACK is not received within the timeout interval the data are transmitted. As the data are transmitted in blocks, namely, TCP segment, the sequence number of the first data byte in the segment is sent to the destination host. The receiving TCP uses the sequence numbers rearrange the segments when they arrive out order and eliminate duplicate segments. The TCP is able to transfer a continuous stream of octets in each direction between its users. In general, the TCP, decide when to block and forward data at their own convenience.

METHODOLOGY

In designing real time networking is quite a difficult task therefore before continuing , there is the need for the analysis of system connectivity requirements. Normally, the best approach is to map and classify traffic in the system. Therefore the first approach is messages sizes and their latency requirements. In most cases, these two parameters are sufficient to choose transport protocols and network topology. First of all, I begin with the transport protocol choice. The main focus of this paper would be on TCP and UDP derived from the concept of TCP/IP. The table below proposes protocol selection rules:

Latency/Messages Length	Few KB or Less	Dozens of KB or more
1 ms (could be less) to dozens of ms	UDP	????......
Hundreds ms or more	TCDP	TCP

Now there is a question of low latency and large messages sizes in the table above.

TECHNICAL BACKGROUND ON TCP

The TCP protocol could deal quite well with big amounts of data. Long messages enable TCP to warm up its engine and find best fit sliding window retransmission and achieve good utilization of bandwidth. TCP latency is not deterministic. First of all, consider a server with multiple clients. Because of the fact that, TCP protocol is lacking fairness and quality of service features, the latency variations may exceed hundreds of milliseconds. But in the case of multiple clients, the bandwidth utilization may significantly drop down (in legacy systems, it may collapse to 50% down time) which is not conducive for designing real time networking. Furthermore, another problem that TCP suffers from is slow recovery process. The regular recovery time, period for a lost TCP fragment may take hundreds of milliseconds but there are also some tricks which allow the process reducing recovery time.

TECHNICAL BACKGROUND ON UDP

Short-end messages in real time systems are normally used for managing hardware devices , data auditing and closed-loop communication (that is controlling the behavior of the device based on the data that device produces in real time). Due to real time nature of data, short messages must be delivered and reach their destination within milliseconds of time. The transport that may withstand this latency is UDP. The Round Trip Time (RTT) of a message over UDP protocol largely depends on hardware and on operating systems of communicating peers. Even so, 100MB network non-real time OS (Operating System) such as **Windows XP™** provide average RTT of one millisecond. But use of real time OS and more advanced network cards may reduce it to microseconds .As a matter of fact, UDP is a non-reliable so to use it reliability features must be added. This is complicated task but if have UDP fairly big, there should be an alternative implemented. A lot of UDP-based protocols are available which provide reliability features but none is suitable real time communications in heterogeneous systems. This article therefore presents new reliable protocol over UDP especially designed such systems. Normally, if you are asked about networking requirement an answer is given in terms of bandwidth. This answer insufficient for real time systems. As indicated earlier, traffic between peers could be classified by its latency requirements and message sizes. For short messages, there are two main important parameters that describe the traffic: number of messages and their distribution. Consider a 1 GB (Gigabyte) link that is used for sending and receiving of short messages. In theory 1 GB card should be able to process over 2,000,000 packets per second (minimum Ethernet frame size is 64 bytes, but in practice it is far from being feasible.

NETWORK PLANNNING APPROACH

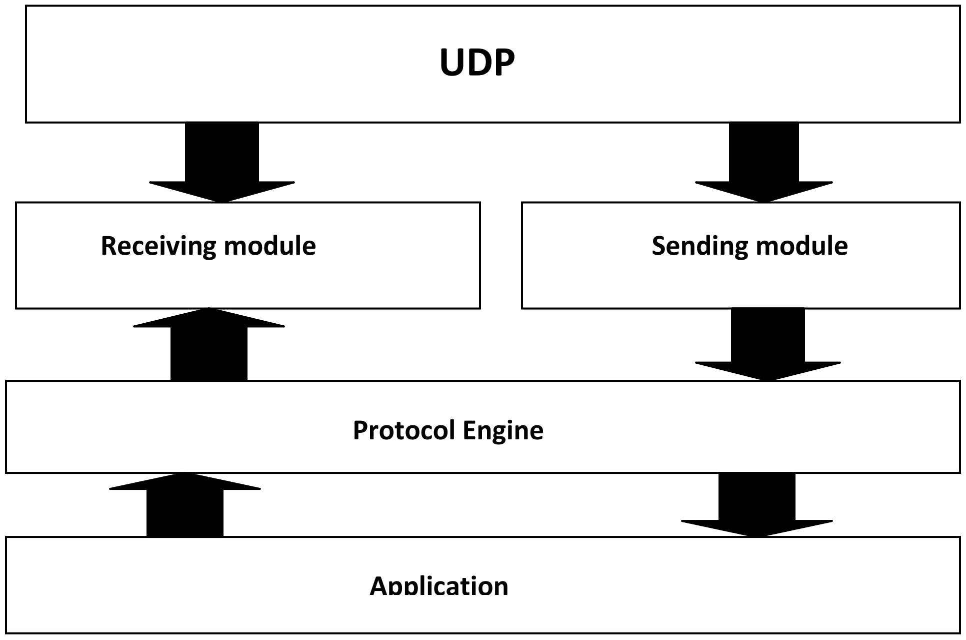
Once traffic has been classified in the system and transport protocol has been selected , the next step is network planning .First of all, the existing network topology must be examined and verified that it can handle traffic requirements, since real time systems have dedicated network for real time communications. Also, this network could be shared between different modules within the system and these modules may have different performance requirements. Let us also assume that, a proper network backbone exists in the system and **the objective of this paper is resolving its utilization.** Now the next step in network planning is separating different types of traffic. For instance, UDP and TCP should be messed up at all. Separating the traffic, density is decreased and thus clashes of packets are simplified for network analysis as well. But the best separation is achieved by creating dedicated physical connections for every traffic (separated network cards and separated wires between peers. Nevertheless, in many cases such a separation is not possible due to budget and hardware limitations. But if this not possible there should be benchmarking of communication and depend of findings. Also implementing a proprietary protocol that will satisfy network requirement. Therefore the protocol proposed in this paper for real time communication is **UDP-RT protocol**.

UDP-RT (USER DATAGRAM PROTOCOL for REAL TIME COMMUNICATION) PROTOCOL

The UDP-RT protocol is the UDP based protocol for real time communications that allows the sending of short messages with low latency and provides protocol based reliability features .Apparently, the UDP-RT protocol fulfill the following requirements :

1. **Low latency:** Send application messages to its destination as soon as possible. Also receive and deliver messages to application promptly (as soon as possible).
2. **Reliability:** Avoid losing messages .Incase of lost message, detect and recover it (time that it takes to recover a message is related to both reliability and latency, so the protocol should pay a special attention to this issue. Handles messages reordering. There is also the need to keep protocol simple aside the above functional requirements. Modules in the real time systems could also run embedded software which is hard to debug and the focus of this paper is to make design and implementation simple.

THE UDP-RT PROTOCOL ARCHITECTURE



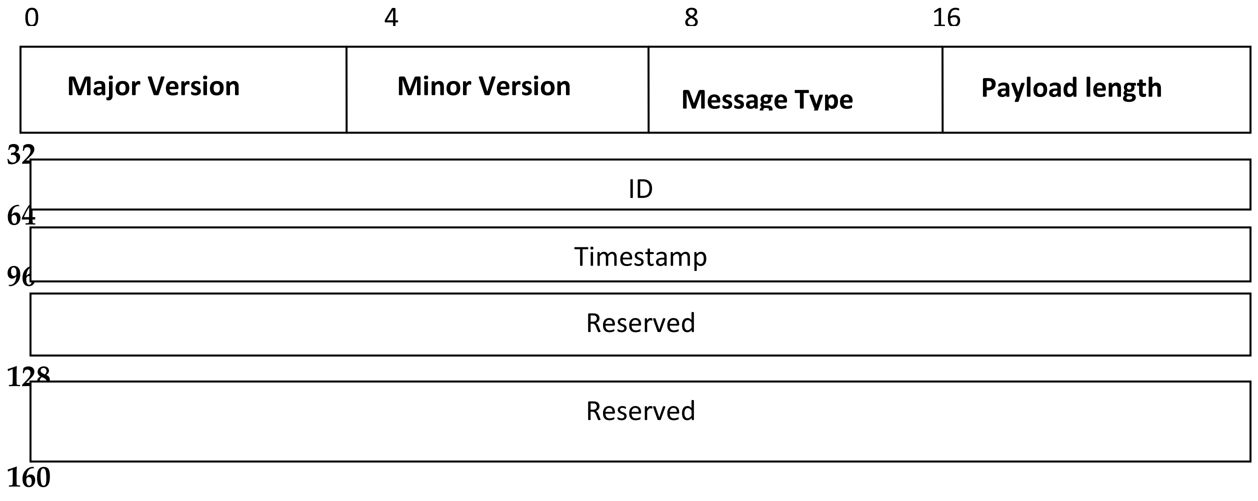
(Michael Pan and Anibrika Bright S.K., 2014)

The UDP-RT suite is divided into three main modules: Sending Module, Receiving Module and Protocol Engine. The first two are responsible for efficient sending and receiving of messages. The modules provide customization abilities to optimize protocol performance at a hosting operating system. The protocol engine covers reliability requirements from the diagram above.

MESSAGE STRUCTURE

Every UDP datagram may carry several messages of UDP-RT protocol. Messages cannot be split over UDP datagrams, so the maximum message size is limited by maximum datagram size (64KB) . A UDP-RT message consists of header and payload.

UDP-RT MESSAGE HEADER PAYLOAD



(Michael Pan and Anibrika Bright S.K., 2014)

1. **Version field** allow the protocol to grow up. New functionalities could be added and the version fields will help peers to determine their compatibilities. Recent version is 0.0
2. **Type**: Messages could be of type DATA(0X0), ACK(0x1), RESET(0x2).The DATA messages are carrying payload .The RESET messages reset messages' counter. The ACK messages are used to acknowledge the arrival of the DATA and RESET messages. The RESET and the ACK messages carry no payload.
3. **Payload length** defines the number of bytes in the message payload.
4. **ID** is a serial number that sender assigns to DATA and RESET messages. The ACK messages must copy the ID from the corresponding DATA and RESET messages.
5. **Timestamp** contains sending time of DATA and RESET messages and filled in by message sender. The ACK messages must copy the timestamp from the corresponding DATA and RESET messages.

CHANNEL SETTINGS

A pair of peers that uses UDP-RT protocol is considered as UDP-RT connection and is referred to as channel. Every channel has its own set of settings which defines the UDP-RT behavior on the channel. Below are the channel configuration settings.

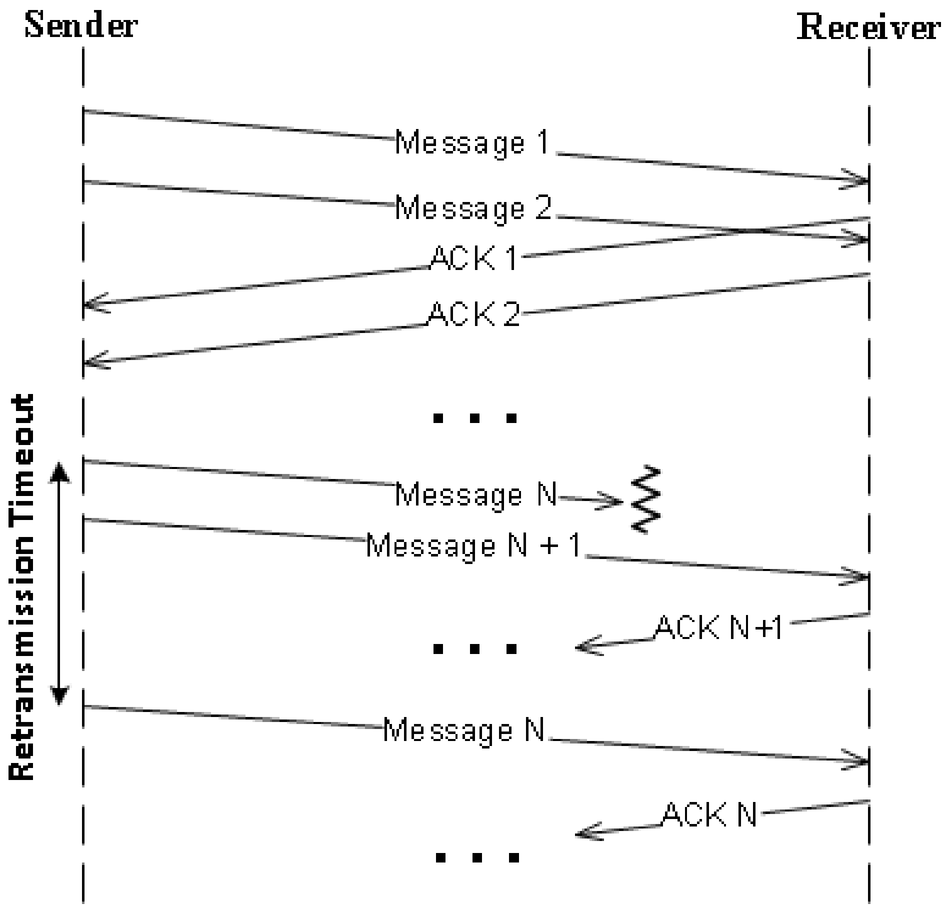
- Initial retransmission timeout – Default value of retransmission
- Retransmission timeout model – Algorithm to be used by protocol engine for calculating retransmission time out value.
- Maximum message delay - Maximum time that application permits for message passage.
- Message dropping policy - Defines how to handle for which "maximum message delay" expires.
- Out-of-order messages policy - Defines whether to allow out-of-order messages at the channel.
- Reset waiting time – Time to wait during reset sequence.

PROTOCOL ENGINE

LOST MESSAGES DETECTION AND RECOVERY

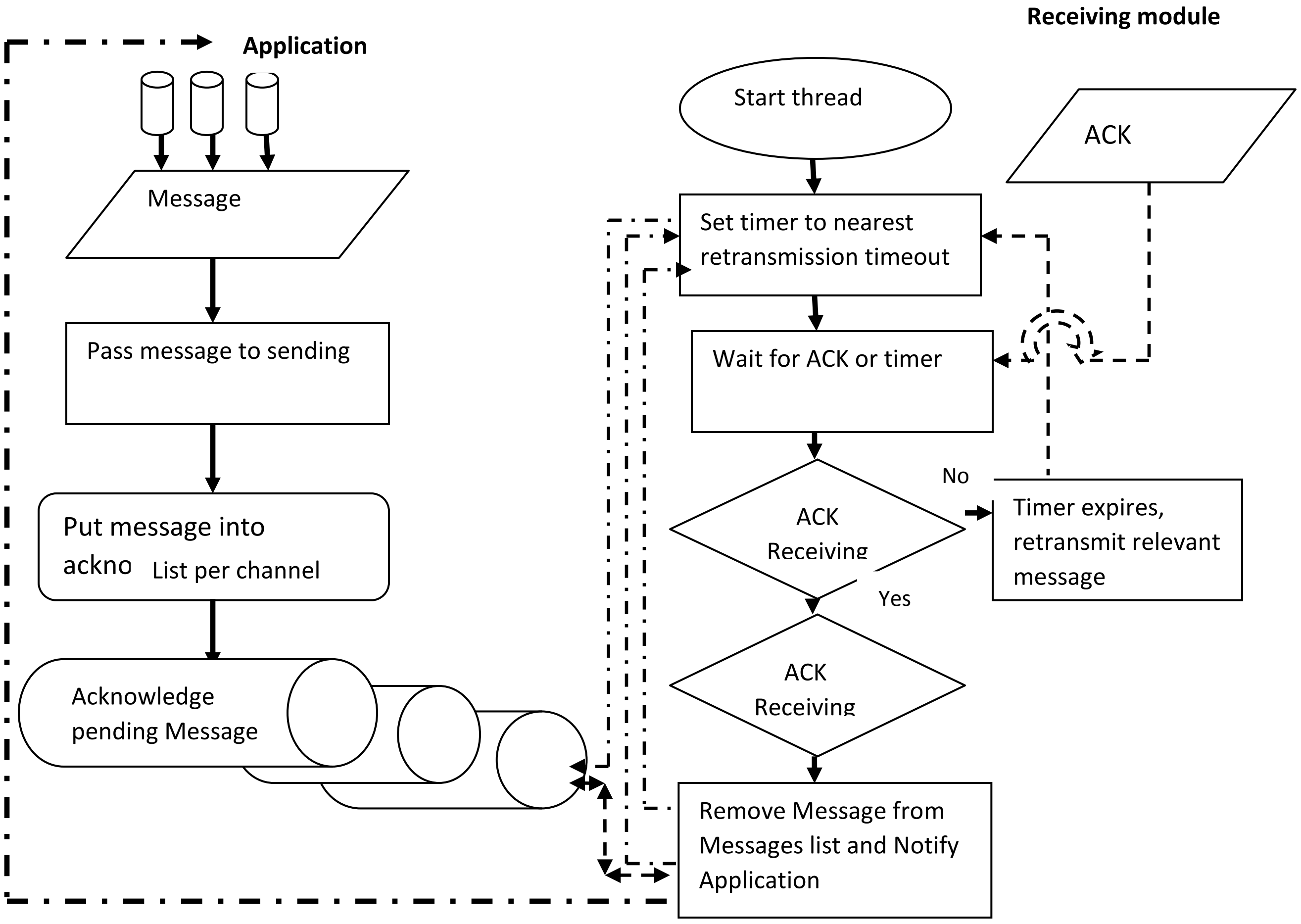
The concept is quite simple. In order to know if messages have arrived at its destination, it must be acknowledged by the receiver. The sender is storing the message in its memory as long as it is waiting for acknowledgement. Once it gets acknowledged, the message can be discarded. In case the message is not acknowledged during some period of time, it must be retransmitted. The figure above explains in details:

(Figure below: Message Retransmission)



When the sending operations is specified by the application, the protocol engine does the following .First it assigns ID to the message and passes it to the sending module which in turn repackages it and transmit the message to its destination .Then it adjusts retransmission time out to the message and pushes it into queue of acknowledging pending messages (there is queue per channel).

Whenever retransmission timeout 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 acknowledgement is received, the protocol engine is notified about it by the receiver modules. It examines the acknowledged pending messages and if acknowledge ID matches with one of them, it removes the message from the queue and notifies application about successful delivery of the message (if application requested such notification).The Sending module is illustrated below:


Receiving module
Application
Message
Pass message to sending
Put message into ackno List per channel
Acknowledge pending Message
Start thread
ACK
Set timer to nearest retransmission timeout
Wait for ACK or timer
No
ACK Receiving
Timer expires, retransmit relevant message
Yes
ACK Receiving
Remove Message from Messages list and Notify Application

SENDING MESSAGES FLOWCHART ABOVE (Michael Pan and Anibrika Brigh S.K)

RESULTS AND FINDINGS

The UDP-RT performance could be measured by average RTT of messages in both sending and receiving modules. If the RTT is significantly smaller than maximum message delay, the protocol definitely serves system well without any system interruptions. Despite, there are instances when the RTT parameter is not conclusive during real time communication. The RTT could be high and have a significant jitter (noise interference variations) in its values. Another parameter should be examined to measure UDP-RT performance: average length of queue of acknowledge pending messages. This parameter is particular useful in the systems with high latency and gives a good measurement for network performance stability.

The server receives messages from clients, does some calculations for every message and sends the results back, so the same amount of messages flows in both directions. This does not count the ACK messages that the UDP-RT is sending for every DATA message. During system modeling of UDP-RT, I recognized that there are no messages lost in server to clients direction, so the UDP-RT has been changed not to send ACK messages from clients. This way the traffic density has been decreased in clients to server direction, where messages delays and losses frequently happened.

CONCLUSIONS/RECOMMENDATIONS

If all clients send and receive messages every 3ms, then, in total, counting the ACK messages, the server receivers 130000 messages per second and sends 260000 messages back to clients. It is equivalent to receiving a message every 7.7 microseconds and sending one every 3.85 microseconds. Messages from clients are of size of Ethernet MTU (1.5KB). Messages from server are significantly smaller and do not exceed 100 bytes in size. The ACK messages are of 20 bytes length, so the total bandwidth required for communications is roughly estimated by 2.2 Gbit (Gigabits of data size), which is less than a half of available bandwidth on the server . The biggest challenge in UDP-RT is decreasing number of packets running in the network. During the design stage, the researcher determined reliable path in the network and exempt corresponding ACK messages. Another, more generic, way to reduce number of packets is to piggyback ACK messages on packets carrying DATA messages. To do so, the receiver won't send the ACK immediately upon DATA message receiving, but delay its sending. If there will be DATA message to transfer to the same peer, it will unite the messages and send them in one data stream.

REFERENCES

Armbruster B., Smith J. Cole and Park K., "A Packet Filter Placement Problem with Application to Defense against Spoofed Denialof-Service Attacks", European Journal of Operational research, Vol. 176, Issue 2, pp. 1283-1292, 16 January 2007.

Bremler-Barr A. and Levy H., "Spoofing Prevention Method", In Proceedings of IEEE INFOCOM, Miami, FL, March 2005.

Cabrera J. B. D., Lewis L., Qin X., Lee W., Prasanth R.K., Ravichandran B. and Mehra R. K., "Proactive Detection of Distributed Denial of Service Attacks using MIB Traffic Variables - A Feasibility Study", Proceedings of the 7th IFIP/IEEE International Symposium on Integrated Network Management, Seattle, WA - May 14-18, 2001.

Chang R., "Defending Against Flooding-Based Distributed Denial-of-Service Attacks: A Tutorial", In Telecommunications Network Security, IEEE Communications Magazine, pp. 42-51, October 2002.

Douligeris C. and Mitrokotsa A., "DDoS Attacks and Defense Mechanisms: Classification and State-of-the-Art", Computer Networks, Vol. 44, pp. 643–666, 2004.

Freiling C., Holz T., and Wicherski G., "Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks", In ESORICS 2005, LNCS 3679, pp. 319–335, Springer-Verlag Berlin Heidelberg, 2005.

Hole K, "Denial-of-Service Attacks", Nowires research Group, Department of Informatics, University of Bergen, September 1, 2008.

Hussain A., Heidemann J., and Papadopoulos C., "A Framework for Classifying Denial-of-Service Attacks", Karlsruhe, Germany, pp.99–110, 2003.

Kim Y., Lau W., Chuah M. and Chao H., " PacketScore : Statistics-based Overload Control against Distributed Denial-of-Service Attacks", IEEE Transactions on Dependable and Secure Computing, Vol. 3, No. 2, pp. 141-155, April-June 2006.