Connection Establishment Algorithm for Multi-destination Protocol

Sergii Maksymov, Dmitry Kachan, Eduard Siemens
Department of Electrical, Mechanical and Industrial Engineering
Anhalt University of Applied Sciences
Bernburger Str. 55, 06366 Köthen, Germany
E-mail: {s.maksymov, d.kachan, e.siemens}@emw.hs-anhalt.de

***Abstract*—connection establishment is a fundamental function for any connection-oriented network protocol and the efficiency of this function defines the flexibility and responsiveness of the protocol. This process initializes data transmission and performs transmission parameters negotiation, what makes it mandatory process and integral part of entire transmission. Thus, the duration of the connection establishment will affect the transmission process duration. This paper describes an implementation of a handshake algorithm, designed for connection with multiple peers, that is used in Reliable Multi-Destination Transport (RMDT) protocol, its optimization and testing.**

***Keywords:* multi-destination; handshake; connection establishment; network protocol.**

I. Introduction

[Sergii Ma1]Reliable Multi-Destination Transport (RMDT) is a protocol designed to deliver data from one source (sender) to multiple destinations simultaneously, accurately regardless network impairments.

It is implemented as a C++ library and aimed to effectively transmit the same data to multiple recipients. During transmission the only one sending instance is created which initializes a common buffer for all recipients, thus it utilizes less system resources and performs minimum copy operations during the transmission process. Big amount of data can be transmitted to many recipients with less load on a system. The protocol is aimed to send data within whole available bandwidth, what is especially important for Long Fat Pipes – links with high bandwidth and latency. Such links are unsuitable for legacy protocols, primarily Transmission Control Protocol (TCP), which is not able to utilize the full bandwidth within high latency links [1].

To provide its benefits RMDT requires a performance hardware that supports multi-threading and has enough memory to allocate big buffers (up to 1000 MB). Sending and receiving operations are split into 2 threads. One thread is responsible only for reception and another one – only for sending. Sender side application has additionally Event thread, which is responsible for data preparation within buffers and processing of acknowledgments from recipients.

II. Background and Related Work

Multi-destination data delivery is closely related to multicast, but implies a uni-direction transmission. The RMDT protocol is based upon UDP and can be considered, from the network point of view, as the set of multiple unicast streams which transport data in one direction from sender to recipients. Due to this fact, the experience of the legacy transport protocols can be used to design the connection establishment function of RMDT.

The well-know TCP utilizes three-way handshake algorithm [2] to establish a connection, see Figure 1. The server should be switched into LISTEN state, this action is called Passive Open. In the LISTEN state server is able to accept a connection request, Synchronization (SYN) packet.

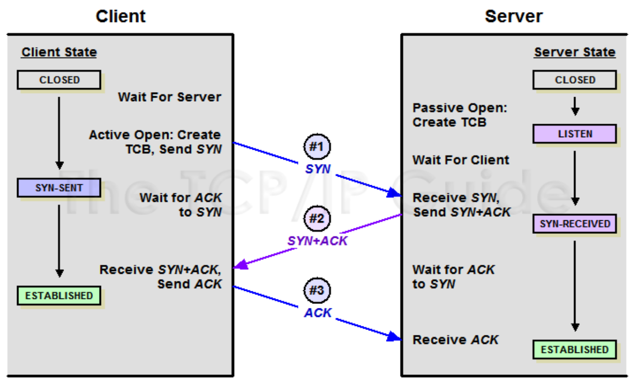

Fig. 1. Three-way handshake process for TCP connection establishment [2]

The connection request must be acknowledged by the server along with negotiation of transport parameters (SYN+ACK packet), primary Initial Sequence Number (ISN) negotiation. The Sequence Number is a number representing a sequence number of the first byte of data in a segment. At the moment of connection establishment, this number is chosen from a special counter [2], that ticks every 4 µs, to eliminate conflicts between different TCP connections. The most basic reason for this is to detect duplicate SYNs and to distinguish the SYN packet belongs to the same connection or it is a part of a new connection, in the case of lost message during 3-way handshake, for example. The ISN is also used against IP-spoofing technique, but with enhanced degree of randomization [4]:

$$ISN = C(t) + hash(L_{addr}, L_{port}, R_{addr}, R_{port}, key) \quad (1)$$

Where, $C(t)$ – value of the counter, L_{addr}, L_{port} – local address and port, R_{addr}, R_{port} – remote address and port, key – a random value chosen by the host on startup. Any hash function can be used, but as written in [4], MD5 hash function is recommended, as it is well supported by different hardware and has a lot of implementations. Such ISN generation complicates IP-spoofing attacks.

III. 2-WAY HANDSHAKE WITH MULTIPLE PEERS

In the case of point-to-multipoint data delivery protocol all the destination points should be notified and connected. This fact introduces some difficulties, because an error triggered by one peer might fail or, at least hinder the common connection establishment process. The increase of amount of receivers leads to the probability of such a failure increase. That is the major distinction from point-to-point connection establishment. In RMDT the 2-way handshake is performed with each peer. Sender initiate a connection by sending Handshake Request (HS Request) packets to each of the receivers and then waits for their responses. Receiver, in turn, waits for this HS Request from the beginning. Request contains transmission session parameters, such as Initial Sequence Number, Receiver ID, Maximal Segment Size and protocol Version Number, see Figure 2. This

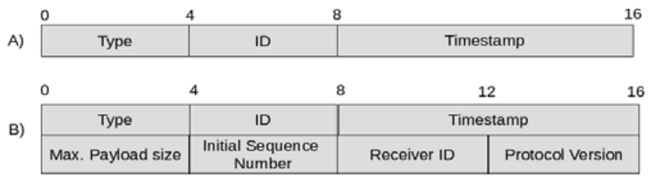

Fig. 2. Packet headers - A) RMDT control packet header; B) Handshake Request packet header.

parameters are checked and set at the receiving application. Then it must send Handshake Response (HS Response) which is expected by the sender for some amount of time. The sender gathers HS Responses during specified time interval from the recipients and then repeats the send operation to those of them which did not responded. Connection is established when all the recipients have responded. It might happen, so the handshakes cannot be exchanged with one or more recipients and the connection would never be established, so the process of handshake will freeze. For such a situation there is a timeout for connection operation, which can be set by the application. In the case of timeout, connection will be also established, if at least one recipient have responded, but an application will be warned about the fact of timeout. This mechanism relieves an application from possible freezing of the connection establishment process.

Because RMDT library runs in 3 threads, there is an Inter-Thread Communication mechanism via queues and notification method of *conition_variable* object provided by the Standard C++ Library and which is used by the handshake process implementation. The implementation is based on two timeouts: timeout for overall process duration and timeout for responses expectation. The first one guarantees that the connection will last no longer then the specified time interval regardless the result of the handshake process, thus protecting against unnecessary hanging inside this process. The second one lets the process to use system resources more effectively and not to flood the network with frequent HS Requests. This timeout has one feature: the process can be waked up before this timeout, when the HS Response comes. This feature allows to save the time on connection establishment in the case, when all the expected responses are come before the timeout. Presence of this feature distinguish two versions of the handshake process implementation – asynchronous and synchronous.

The synchronous handshake is performed in the same way, but its duration is aligned to the specified time period for HS Responses expectation. Owing to the fact that operation status check is performed after timeout at each iteration of this algorithm. The flowchart of the algorithm is shown in the Figure 3.

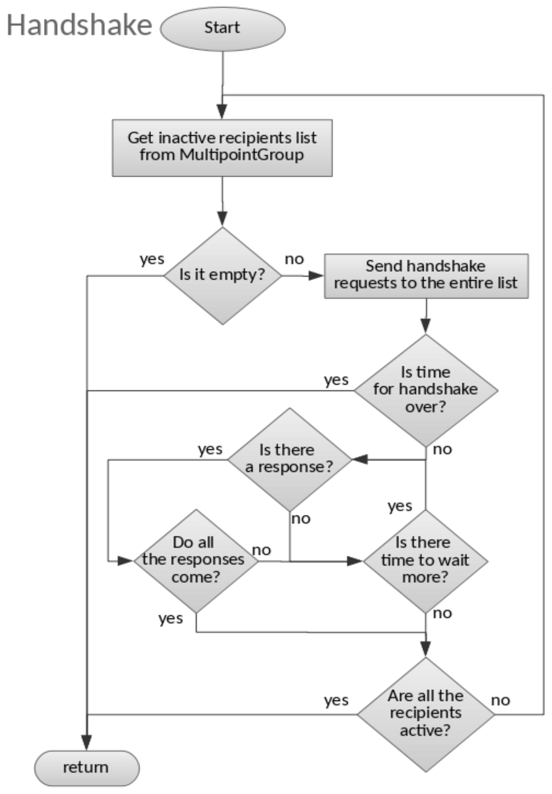
Handshake
Start
Get inactive recipients list from MultipointGroup
Is it empty?
yes
no
Send handshake requests to the entire list
Is time for handshake over?
yes
no
Is there a response?
yes
no
Do all the responses come?
no
yes
Is there time to wait more?
yes
no
Are all the recipients active?
yes
no
return

Fig. 3. Flowchart of the asynchronous handshake process.

The asynchronous handshake operation is more responsive, but has a small overhead in form of inter-thread communication and thread synchronization comparing to synchronous one. But it is insignificant for performance systems that the library is designed to.

IV. TEST-BED AND DESCRIPTION OF EXPERIMENTS

To test the protocol a 10 Gbps network, shown in Figure 4, is used. This network is located in the laboratory Future Internet Lab Anhalt (FILA) [5] and is used for experiments and protocol testing. With the help of this network the two implementations of handshake process were tested, namely measuring the duration of the connection establishment in different conditions of both synchronous and asynchronous implementations. The network interconnects 4 multicore servers with 10 Gbps interfaces, 2 Extreme Networks Summit x650 10 Gbps capable switches and 2 network emulators Apposite 10G. Detailed configuration of the servers is shown in the Table I.

TABLE I
CONFIGURATION OF THE SERVERS

Server name	Linux kernel	CPU	RAM
Germany	4.2.0-23-generic x86_64	2x Intel Xeon X5690 (6-core) 3.5 GHz	40 GB DDR3 1066 MHz
Brazil	3.13.0-37-lowlatency x86_64	2x AMD Opteron 4238s (6-core) 3.3 GHz	32 GB DDR3 1333 MHz
Argentina	3.13.0-35-generic x86_64	2x AMD Opteron 4238s (6-core) 3.3 GHz	32 GB DDR3 1333 MHz
Kazakhstan	3.13.0-45-generic x86_64	2x Intel Xeon E5-2630 (6-core) 2.3 GHz	64 GB DDR3 1333 MHz

The Apposite 10G network emulator allows to introduce in the network different impairments, such as packet delay and packet loss with high accuracy up to nanoseconds [6]. Moreover, it displays and controls link capacity passing through the emulator.

Extreme Network Summit X-650 10Gbps switches perform layer 3 packet switching. All the connections are implemented with optical fiber links and have capacity of 10Gbps.

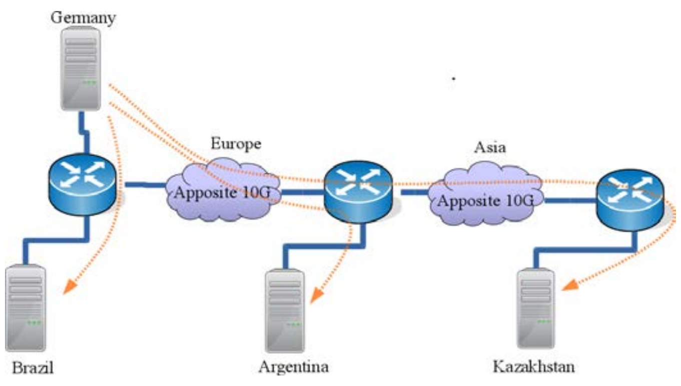

Fig. 4. Network topology

Experiment scenario is to initialize data transmission from source (Germany) to 3 destinations (Brazil, Argentina, Kazakhstan) using the RMDT test application and measure the duration of the handshake process in the network with low impairments: 50 ms of Round-Trip-Time (RTT) and no packet losses. The second scenario introduces more impairments into the links: up to 250 ms of RTT and up to 0.7% of packet losses. These parameters are even more worse than the real links between Germany and USA, for example. In both scenarios the synchronous and asynchronous implementations are compared.

V. Experimental Results

The experiments are divided into 4 groups by allocated send buffer size – 10 MB, 100 MB, 500 MB and 1000 MB, because the buffer allocation time depends on its size. For each of the buffer sizes, 10 measurement iterations were done for both implementations, so 20 in total. Multiple measurements are done to get more precise results, because the system activity affects experiment results.

The results have very slight deviation, so it can be hardly seen in the Figure 5.

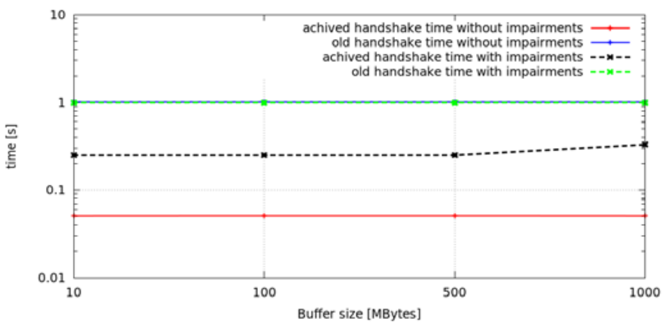

Fig. 5. Experiment results (with std. deviation).

The asynchronous handshake takes almost 20 times less time then the synchronous one within good link (50 ms RTT) and about 4 time faster within link with more impairments (250 ms RTT and 0.7% of packet losses). Impact of packet delay can be easily seen on the asynchronous handshake plot (the black line), 200 ms increase is clear, but it is invisible in synchronous handshake plot. It is pretty clear, that the timeout in synchronous handshake implementation is major factor that defines the duration of the handshake.

TABLE II
EXPERIMENT RESULTS WITHIN GOOD LINK (AVERAGE)

Buffer size, MB	Asynchronous implementation, s	Std. deviation	Synchronous implementation, s	Std. deviation
10	0,050509	0,000031	1,000868	0,000181
100	0,050504	0,000025	1,000756	0,000087
500	0,050497	0,000031	1,000781	0,000059
1000	0,050512	0,000028	1,000806	0,000019

The numbers in the Table II and Table III contains the mean handshake duration obtained from experiments. As it was mentioned above, the duration of the synchronous handshake is aligned to the timeout for HS Responses expectation, which is 1 s in our experiments. There can be smaller value specified for this expectation, in other words sleep of the thread, but in this case the network link will be overloaded by frequent HS Requests send, what is redundant and dangerous especially for links with high latency.

TABLE III
EXPERIMENT RESULTS WITHIN BAD LINK (AVERAGE)

Buffer size, MB	**Asynchronous implementation, s**	**Std. deviation**	**Synchronous implementation, s**	**Std. deviation**
10	0,250000	0,000033	1,000000	0,000011
100	0,250406	0,000011	1,000512	0,000008
500	0,250415	0,000011	1,000513	0,000006
1000	0,330370	0,252299	1,000515	0,000009

VI. CONCLUSION

There is a need of reliable and fast point-to-multipoint data delivery, especially on the side of huge content distributors. And there is also a lack of new ideas regarding such kind of data transmission. Thus, design of multi-destination protocol, which is aimed to correspond contemporary data delivery requirements and to be able to effectively utilize the available hardware resources is in demand. Connection establishment process of such protocol requires attention as it is the fundamental function of the data transmission.

Designing the handshake operation, two approaches were developed and compared, synchronous and asynchronous one. The asynchronous handshake implementation demonstrated much effective work, especially within the links with good parameters, which are widely used over the world, for example link from [Sergii Ma2]Berlin to Moscow or from Madrid to Tokyo . This approach has the only drawback that it has some overhead on resources utilization due to inter-thread communication, but it is insignificant for contemporary systems which are required for RMDT.

VII. FURTHER WORK

The further work on the protocol will be focused on implementation of the rest necessary features, namely congestion control based on the Available Bandwidth Control (ABC) [7] and finalize the session management to isolate the recipients with different throughput. And much further work will be aimed on inspection of the security issues, because for now the security of the protocol completely relies on the security aspects of underlying UDP. Besides, there is a research can be performed on the security issues of the connection establishment, for which the experience of the TCP with IP-spoofing can be used to prevent similar attacks.

REFERENCES

[1] D. Kachan, E. Siemens, and V. Shuvalov, “Comparison of Contemporary Solutions for High Speed Data Transport on WAN 10 Gbit/s Connections,” *J. Commun. Comput.*, vol. 10, no. 6, pp. 783–795, 2013.

[2] “The TCP/IP Guide - TCP Connection Establishment Process: The ‘Three-Way Handshake.’” [Online]. Available: http://www.tcpipguide.com/free/t_TCPConnectionEstablishmentProcessTheThreeWayHandsh-3.htm. [Accessed: 01-Mar-2016].

[3] R. Braden, “Requirements for Internet Hosts - Communication Layers.” [Online]. Available: http://tools.ietf.org/html/rfc1122.html#page-87. [Accessed: 07-Mar-2016].

[4] S. Bellovin, “Defending Against Sequence Number Attacks.” [Online]. Available: http://tools.ietf.org/html/rfc1948.html. [Accessed: 07-Mar-2016].

[5] “F I L A.”, [Online]. Available: http://fila-lab.de [Accessed: 07-Mar-2016].

[6] “Apposite Technologies :: Netropy 10G2 Network Emulator.” [Online]. Available: http://www.apposite-tech.com/products/netropy-10G2.html. [Accessed: 07-Mar-2016].

[7] D. Kachan, E. Siemens, and V. Shuvalov, “Available bandwidth measurement for 10 Gbps networks,” in *2015 International Siberian Conference on Control and Communications (SIBCON)*, 2015, pp. 1–10.