© 2016 IJSRSET | Volume 2 | Issue 1 | Print ISSN : 2395-1990 | Online ISSN : 2394-4099
Themed Section: Engineering and Technology

SCTP Reform for Receiver Organized Fractional Dependability

Ashish Parejiya[1*], Dr. V. K. Chaubey[2], Dr. Vinod Desai[3]

[1,2]Department of Computer Science and Engineering,. Mewar University, Rajasthan, India
[3]Computer Science Departments, Govt. Science College, Navsari, Gujarat, India

ABSTRACT

Stream Control Transmission Protocol (SCTP) was introducing to overcome several limitations of the Transmission Control Protocol (TCP). SCTP has been investigated for its real-time data streaming capabilities. Extensions and modifications to SCTP have been proposed, which take advantage of the reliability features. This paper proposes a partial-reliable modification to SCTP, which reduce the overhead when compared to other partial-reliable extensions (i.e. PR-SCTP). Description of the extension and simulations are discussed. Also a comparison is made between our proposed extension, PR-SCTP, and UDP.

Keywords: SCTP, Partial Reliability, Real-Time Streaming

I. INTRODUCTION

Stream Control Transmission Protocol [1] is a transport protocol proposed by IETF in October 2000. It provides reliable transport service such as acknowledged, error-free, non-duplicated, and sequenced transfer of user data on top of a connectionless packet network such as IP. Reference [2] explains the limitations of TCP that SCTP overcomes. Fault tolerance was added with multi-homing, which allows one association to contain multiple paths to the destination. With the introduction of multi-streaming, the dreaded head-of-line blocking can be reduced. Multi-streaming allows an association to contain reliable and unreliable streams. Important data such as control messages can be sent on reliable streams while unimportant data is sent on unreliable streams.

This paper proposes to give the receiver application the ability to tell the sender that it no longer requires packets even if they have not been received. This is important in real-time streamed data transfer in which receiving timely data is more important than receiving all of the data. By adding this real-time data transport can benefit from the reliable transmission of control messages and unreliable real-time transport of data without the necessity of multiple connections (associations). By allowing the receiver to ignore stale messages the sender can continue to send up-to-date, relevant messages that can be used by the receiver, rather than be discarded. This method can work harmoniously with TCP, since the congestion control schemes in SCTP are TCP-friendly.

Presently for real-time data streaming, UDP is the most prevalent transport layer protocol used. The problem with using UDP is that its delivery is only best effort and congestion or flow control must be done using proprietary methods at the application layer. This forces the application to create flow control on its layer. Not only does this add to the complexity of the application, it creates additional time and cost in development.

Another drawback to using UDP and these proprietary methods is that they are not necessarily TCP-friendly. TCP is based on fairness, which only works when all nodes on the network are using the same congestion control scheme. Without fairness, those more aggressive users become allocated more of the available bandwidth, thus bogging down the majority of users. While this may be beneficial to the unfair user in the short term, long term costs in terms of service costs, etc. can catch up to the user.

TCP is not generally used for real time data transport such as streaming audio and video. What makes TCP perfect for some applications, such as the transport of files, makes it a poor choice for real time streamed data. TCP's ability to reliably deliver in-order data cost at a cost. Problems such as head-of-line blocking, in which received data cannot be sent to the application layer because prior data is still outstanding are the reason TCP is not chosen for this application. TCP's methods for avoiding congestion can lead to a wide variation in the delay experienced by packets.

Another attractive characteristic of SCTP as opposed to TCP is the message oriented aspect of SCTP. TCP's byte stream forces the application to do the message framing.

SCTP can provide the best of these two transport protocols, but it still has its drawbacks. Per stream head-of-line blocking can still occur on ordered data. SCTP's reliability, like that of TCP hinders its usefulness as a real-time transport layer.

There is an RFC (3758) that proposes an extension to SCTP that allows for partial reliability [3], but it only allows the sending host to advance the transmission sequence number (TSN) of the receiver, thereby telling the receiver to ignore missing packets up to the transmitted TSN. [4] shows how partial reliability can work for real-time streaming communication.

This sender-side flow control has its drawbacks. For one, it requires a new chunk to specify the packet is to be forwarded. While the construction of this chunk is done in such a way that it is compatible with versions of SCTP that do not implement it, those implementations cannot benefit from the extension. Another drawback of this new chunk is the required overhead of sending this additional chunk.

Our method could work together with this method or separately. Theoretically, this paper's method would introduce less overhead than that proposed in RFC 3758 in that we could simply send acknowledgements for data that has timed out whether it has been received of not. This would generate network traffic that would be the equivalent to actually receiving the data. It appears to the sender a an ordinary SACK or delayed SACK. This allows compatibility with non-RPR-SCTP hosts without any loss of functionality of the extension. The receiver partial reliability extension is also fully compatible with the PR-SCTP extension and can be used in harmony with it to allow both end points to advance the cumulative TSN of the receiver.

One drawback of RPR-SCTP is that it requires the application to be aware of its presence to some degree. One possible way to get around this is when the application requests data that has not yet been received; this can act as a forward TSN. This could alleviate most knowledge that the application would need and could be setup when the association is created through the setting of a few parameters.

II. METHODS AND MATERIAL

A. Proposed Modification

RPR-SCTP (receiver partial-reliable SCTP) gives the receiver application the ability to acknowledge missing data in order to receive new data from the sender. The modification is implemented only on the receiver side. This is much like a SACK but the gaps in the TSN are artificially filled in to prevent retransmissions and keep fresh data flowing through the link.

An SCTP module for Network Simulator 2 (ns-2) is used for the modifications. The new RPR-SCTP module is actually a child of the SCTP module. The module relies on the application layer to determine if a forward acknowledgment is needed. The receiver application is designed to consume data similar to a real-time application. This application creates and monitors a circular buffer for data consumption. As the circular buffer is depleted, the application asks the transport layer for new data. The RPR-SCTP module will generate an acknowledgment that tells the sender to send the newest data.

B. Model Implementation

The RPR-SCTP class model in ns-2 is a derived class of the existing PR-SCTP model. It was decided that this was the best way to model RPR-SCTP since the SCTP class itself is very complicated. This was part of the reasoning behind choosing ns-2[6] [7] [8] over MLDesigner. The existing PR-SCTP allows optional partial reliability on a per-stream basis. The implementation of partial reliability allows the application or user to specify how many retransmissions are allowed when a SACK is received that has a gap for a given TSN. This number can be set to zero, in which case the sender, upon receiving a SACK with a gap in it, simply sends a forward TSN chunk to the receiver and continues transmitting as if all the TSNs up to and including the TSN in the forward chunk have been received.

This extension of the SCTP Agent class modifies the underlying class only slightly. The existing class did not have a method for passing received data to the application layer. This is not totally unique to our extension, but it was necessary in order for the application to consume data that was sent across the link, from the transport layer.

Second, a method was added to allow the application to request data when no more was available. This signals the transport layer to increment its cumulative selective acknowledge point is possible and send this acknowledgment to the sending node. This makes the sender think that the data has been received, whether it has or not, and it behaves as if the data had been successfully received.

This implementation differs from UDP in that it does not totally go out of control when it starts sending. The SCTP rules for flow control still apply and the sender uses acknowledgements to clock its transmissions.

III. RESULTS AND DISCUSSION

A. Simulator

Ns-2 [4] was used because it has an implemented SCTP [9][10][11] module from which the modifications were made. The PR-SCTP extension is also implemented in the SCTP module.

This is a powerful simulator that uses the OTcl event-driven scripting language. The modules and most applications are created in C++. Ns-2 has a steep learning curve; however, the benefits [6][7][8] of this simulator are evident with its fast simulation time and the support of the open source community. Being able to see the source code for the modules was invaluable in creating and debugging our own module.

The network setup that was used is simple and in no way does it attempt to simulate a working network. What it does do it create scenarios that could occur in a real network with multiple nodes and their associated congestion.

B. Simulation Scenarios

Our network setup is simply a sending node, a receiving node, and a link. The sending node has an application to send a steady stream of data to the receiver. The receiver has an application associated with it that consumes the arriving data at a constant rate from internal buffer that it maintains. When the buffer goes below a certain threshold it requests additional data from the transport layer. The link is configurable to set the bandwidth, delay, queuing method (drop tail, red, etc.), and loss model. The loss model can be tailored to drop certain packets from a list or use various statistical models to mimic behavior in a network.

All the scenarios consist of a source node and a destination node linked by a single line. Several different transport-layer modules are attached to these nodes for each scenario. Comparisons between RPR-SCTP, SCTP with the partial reliability extension and UDP are made. An example of this scenario is presented in Figure 1.

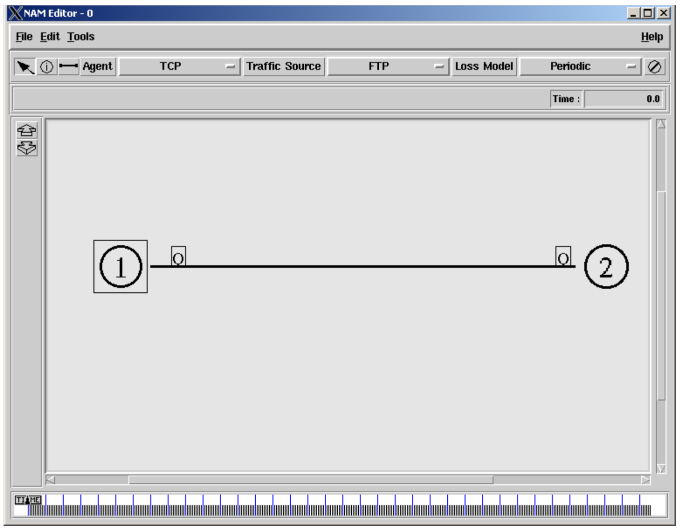

Figure 1 : Simulation Scenario

C. Results

In this section we will present a scenario where two packets are dropped and view its effects on the throughput of the link. This thus effects the average delay that is seen, a key factor for real-time data transmission.

In

Figure **2**, a network trace of PR-SCTP[9][10][11] is presented. In this scenario a packet was forced to be dropped twice in order to illustrate the one retransmission limit.

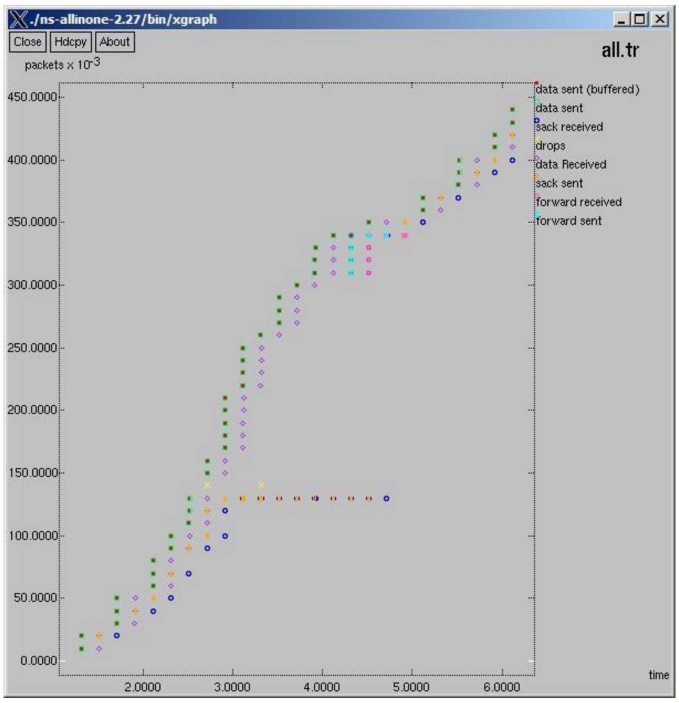

Figure 2 : PR-SCTP trace, 1 retransmission

In Figure 3, the case of a maximum of 2 retransmissions I exercised. It can be seen that the packet is successfully retransmitted the second time. This has a very negative effect, however, on the delay.

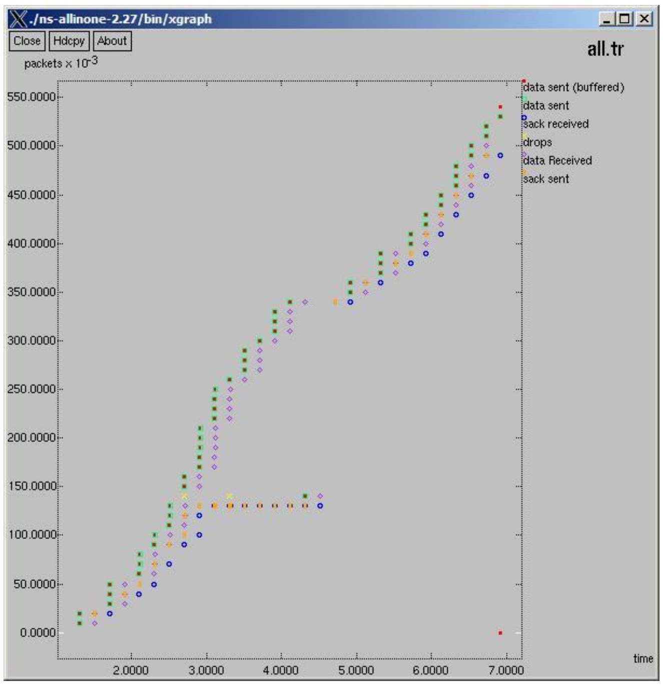

Figure 3 : PR-SCTP, 2 retransmissions

In Figure 4, the zero retransmission case is tested. It can be seen that the delay is improved from the other cases.

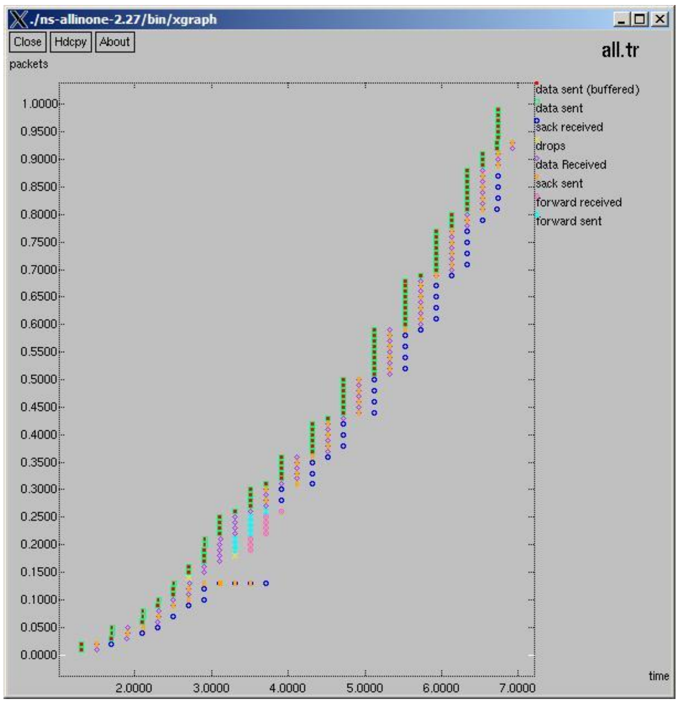

Figure 4 : PR-SCTP, 0 retransmissions

In Figure 5, the RPR-SCTP module is used. In this case the cumulative acknowledgment is simply incremented and the packet is never requested a second time. This has the best delay.

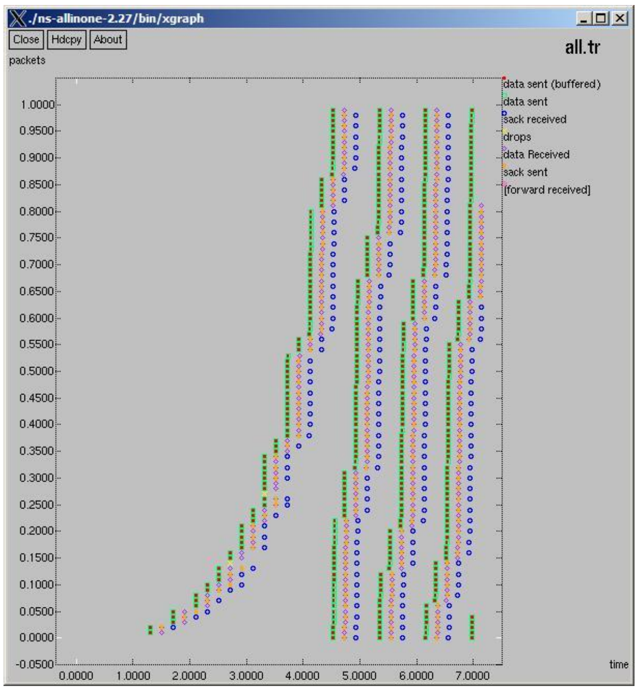

Figure 5 : RPR-SCTP

IV. CONCLUSION

A conclusion section is not required. Although a conclusion may review the main points of the paper, do not replicate the abstract as the conclusion. A conclusion might elaborate on the importance of the work or suggest applications and extensions.

V. REFERENCES

[1] Stewart and Q. Xie et. al., “Stream Control Transmission Protocol.” IETF RFC 2960. October 2000.

[2] S. Fu and M. Atiquzzaman. “SCTP: state of the art in research, products, and technical challenges.” Computer Communications. pp. 85-91. October 2003.

[3] R. Stewart, M. Ramalho, Q. Xie, and P. Conral, “SCTP Partial Reliability Extension.” draft-stewart-tsvwg-prsctp-01.txt. July 2002.

[4] H. Wang, Y. Jin, W. Wang, J. Ma, and D. Zhang, “The performance of PRSCTP, TCP, and UDP for MPEG-4 multimedia traffic in mobile network.” International Conference on Communication Technology. pp. 414-419. April 2003.

[5] K. Fall and K. Varadnam, “The ns Manual.” December 2003. [Online] Available: http://www.isi.edu/nsnam/ns/ns-documentation

[6] Ashish Parejiya, Dr. Vinod Desai, "Congestion Control for Streaming Data Broadcasting over Internet", IJMTER Vol.1 Issue. 5, pg. 352 to 362, Novmber 2014, E-ISSN. 2349-9745, http://www.ijmter.com

[7] Ashish Parejiya, Dr. Vinod Desai, "A Comparative Study and Survey on Broadcasting Multimedia Streaming Data Congestion Control Mechanisms", IJCSMC, Vol. 3, Issue. 12, December 2014, pg.567 – 573, E-ISSN. 2320–088X, http://www.ijcsmc.com

[8] Ashish Parejiya, Dr. Vinod Desai, "Multicasting Of Adaptively-Encoded MPEG4 Over Qos-Cognizant IP Networks",IJMTER Vol.2 Issue. 1, pg. 563 to 570, January 2015, E-ISSN. 2349-9745, http://www.ijmter.com/

[9] Ashish Parejiya, Dr. Vinod Desai "Transforming Network Coding with TCP for Broadcasting Streaming Data in Multi-Hop Wireless LAN", IJESRT, Vol. 4, Issue. 2, February 2015, E-ISSN. 2277-9655, http://www.ijesrt.com

[10] Ashish Parejiya, Dr. Vinod Desai, "MULTI-PATH MECHANISM FOR BROADCASTING AUDIO / VIDEO STREAMING BASED ON BANDWIDTH ESTIMATION", IJESRT, Vol. 4, Issue. 5, May 2015, E-ISSN. 2277-9655, http://www.ijesrt.com

[11] Ashish Parejiya, Dr. Vinod L Desai, & Dr. V. K. Chaubey. (2016). STREAMING APPLICATION QOS: CONTROLLING CONGESTION VIDEO STREAMING MEDIA DATA QUALITY THROUGH SENDING THE FINEST DATA PACKAGE SUBSEQUENT. International Journal of Advance Research and Innovative Ideas in Education, Volume 2 Issue 1 2016 Page 174-183 E-ISSN. 2395-4395,http://www.ijariie.com