An Efficient Error Control Scheme for TCP Segments

Toe Nai Win, Win Aye
Computer University, Mandalay
toenaiwin@gmail.com, mawinaye@gmail.com

Abstract

Transmission Control Protocol (TCP) use cumulative acknowledgement scheme to provide reliable stream delivery service. Cumulative acknowledgement has one major inefficiency, it does not provide information about all successful transmission. In this paper, we represent the problem of how lack of information about all successful transmission makes TCP inefficient and present a way to recover such inefficiency. When transmitting a segment, bits will be added to identify the segment's current position in window and receiving end will also send acknowledgement with extra bits for providing information about loss and arrived segments. This system will simulate cumulative acknowledgement of TCP and our modified scheme, and evaluate the performance in time, waste of network bandwidth of both schemes. Experiments show that cumulative acknowledgement is efficient when network is lossless because our scheme need a little more computation but our proposed scheme becomes more and more efficient as loss of segments in transmission increases.

1. Introduction

TCP/IP (Transmission Control Protocol/Internet Protocol) is a suite of protocols devised back in the 1970's to allow communication between hosts [6]. Although TCP\IP include many protocols, the entire protocol is referred to as TCP\IP because Transmission Control Protocol (TCP) and Internet Protocol (IP) are the two fundamental protocols. The Transmission Control Protocol (TCP) is intended for use as a highly reliable host-to-host protocol between hosts in packet-switched computer communication networks, and in interconnected systems of such networks [4]. TCP is oriented toward a more general environment, supporting the transfer of a stream of bytes between two communicating parties [5].

TCP provide reliable, full-duplex stream delivery service. TCP operate at transport layer of the OSI seven-layer model. TCP is a connection-oriented protocol. Participants must establish a connection before they can transmit data. TCP allow application program at one machine to send data to an application program on other machine. The unit of data transfer between two machines is called a segment. TCP segment is divided into two parts, a header followed by data. Figure1 show a TCP segment format [1]. The source and destination port number is used to identify the sending and receiving application. The sequence number identifies the byte in the stream of data from the sending TCP to the receiving TCP that the first byte of data in this segment represents. The acknowledgment number contains the next sequence number that the sender of the acknowledgment expects to receive. The header length gives the length of the header in 32-bit words. This is required because the length of the options field is variable. There are six flag bits in the TCP header. One or more of them can be turned on at the same time.

URG The urgent pointer is valid.
ACK The acknowledgment number is valid.
PSH The receiver should pass this data to the application as soon as possible.
RST Reset the connection.
SYN Synchronize sequence numbers to initiate a connection.
FIN The sender is finished sending data.

TCP's flow control is provided by each end advertising a window size. This is the number of bytes, starting with the one specified by the acknowledgment number field, that the receiver is willing to accept. This is a 16-bit field, limiting the window to 65535 bytes. The checksum covers the TCP segment: the TCP header and the TCP data. This is a mandatory field that must be calculated and stored by the sender, and then verified by the receiver. The urgent pointer is valid only if the URG flag is set. This pointer is a positive offset that must be added to the sequence number field of the segment to yield the sequence number of the last byte of urgent data [2].

Although TCP has many advantages, it has inefficiencies too. TCP uses a cumulative acknowledgment scheme in which received

segments that are not at the left edge of the receive window are not acknowledged. This forces the sender to either wait a roundtrip time to find out about each lost packet, or to unnecessarily retransmit segments which have been correctly received. Both of them are inefficient. But it also eliminates the need to retransmit for lost acknowledgements.

In this paper, we only focus on proving that adding some extra bits can improve the efficiency of cumulative acknowledgement. The system is designed as a simulator for simulating the behavior of cumulative acknowledgement and our proposed scheme.

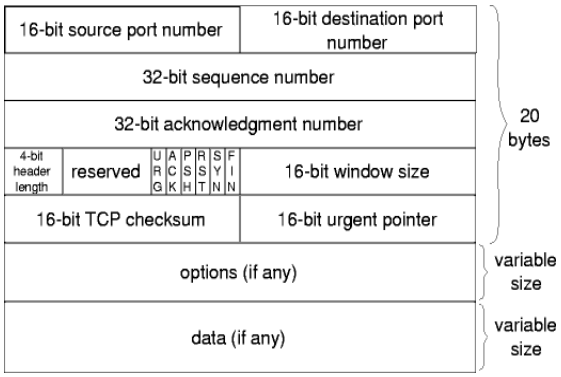


Figure 1.The format of a TCP segment with a header followed by data.

2. Related works

There hasn't been much of the work with TCP's acknowledgement. Kevin Fall and Sally Floyd have discussed the simulation-based comparison of Tahoe, Reno and SACK TCP. They use simulations to explore the benefits of adding selective acknowledgments (SACK) and selective repeat to TCP [3]. Matt Mathis and Jamshid Mahdavi from Pittsburgh Supercomputing Center, Sally Floyd from Lawrence Berkeley National Laboratory and Allyn Romanow from Sun Microsystems, Inc have proposes an implementation of SACK and discusses its performance and related issues [7]. Richard Fox also discusses about TCP big window and Negative acknowledgement options to provide a more efficient operation over a network with a high bandwidth*delay product [8].

3. Proposed scheme

In our proposed scheme, the inefficiency of cumulative acknowledgement is intended to solve by adding additional octets as shown in Figure 2. The 5-bit segment identifier bits identify the current segment position in window. The 5-bit identifier field is used to determine what bits are legal in information field. The 27-bit information field describes which segments in the current window are arrived and which does not arrived. So the sender may only need to retransmit the lost segments.

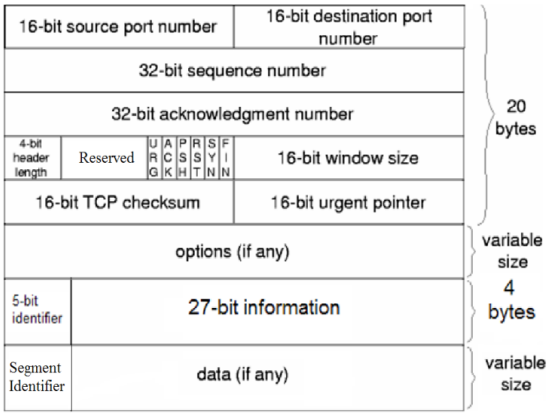
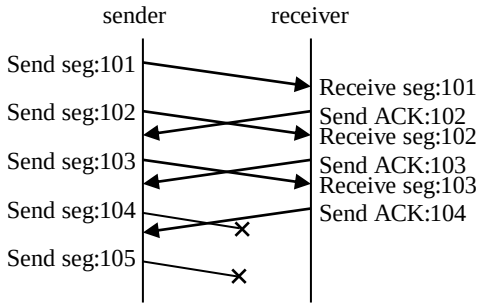


Figure 2 .The Modified Segment Format

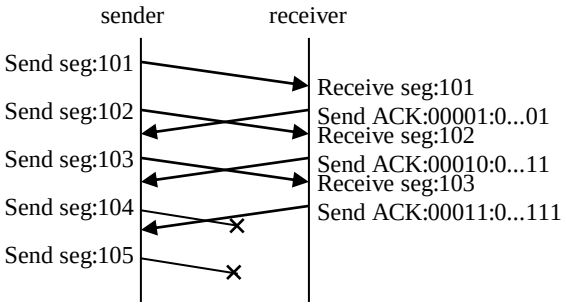
Assume all data in window are sent by transmitting five segments and starting at sequence number 101, without including any data.

Case 1:

The first 3 segments are received but the last 2 are lost. In this case, both schemes need to retransmit segment 4 and segment 5. Figure 3 illustrate the case.



(a)Cumulative acknowledgement

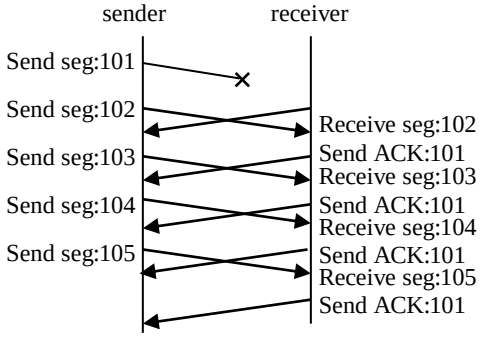


(b)Proposed scheme

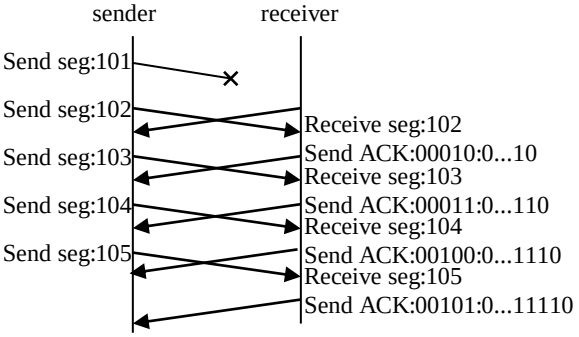
Figure3. (a)Cumulative acknowledgement and (b) Proposed scheme; on both schemes, segment 4 and segment 5 will be retransmitted when retransmission timer expired.

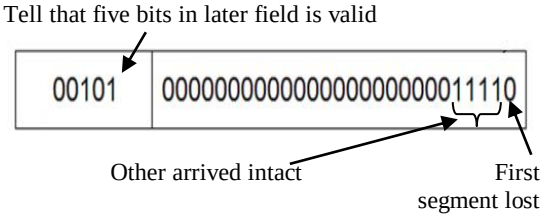
Case 2:

The first segment is lost but the remaining 4 are received. In this case, our proposed scheme only needs to retransmit first segment. In cumulative acknowledgement scheme, the sender needs to retransmit all segments or learn about loss segments one per round trip time. How our scheme handles this case is illustrated in Figure 4.



(a)Cumulative acknowledgement



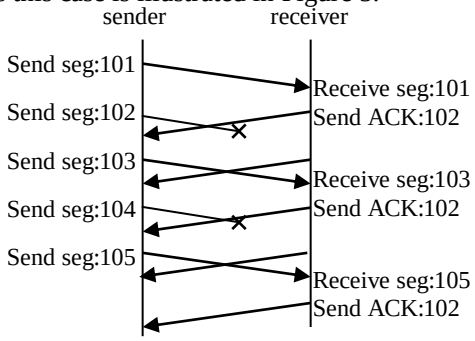


(b)Proposed scheme

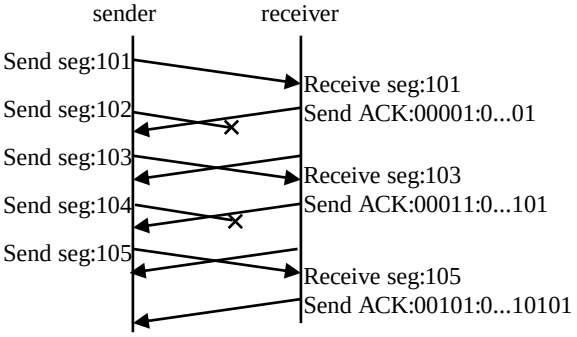
Figure4. (a)Cumulative acknowledgement, sender can't know which segments are lost and which arrived and (b) Proposed scheme, the rightmost 0 represent first segment lost and other four 1s mean arrived segments.

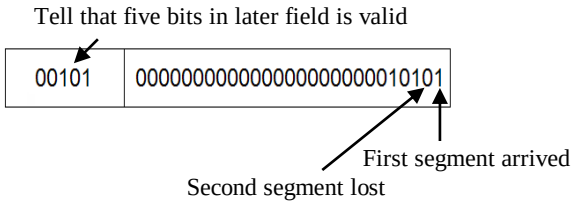
Case 3:

The 2nd and 4th segments are dropped. In this case, our proposed scheme only needs to retransmit second segment and fourth segment. In cumulative acknowledgement scheme, the sender needs to retransmit last four segments or learn about loss segments one per round trip time. How our scheme handles this case is illustrated in Figure 5.



(a)Cumulative acknowledgement





(b)Proposed scheme

Figure5. (a)Cumulative acknowledgement and (b) Proposed scheme, 0s in valid bits represent lost segments and 1s represent arrived segments.

4. Proposed system

In this section, we represent the detail of our proposed system. Because we can't code the entire TCP\IP protocol, we will only simulate the work of TCP's cumulative acknowledgement and will show how our modification has improved the efficiency. First, the program will transmit a segment, records record the time at which each segment is sent and the time at which an acknowledgement arrives for the segment. From the two times, round trip time will be computed, which will be used to find the retransmission time. In an Internet, a segment traveling between a pair of machine may traverse a single, low-delay network (example; a high-speed LAN) or it may travel across multiple intermediate networks through multiple routers. Thus, every time a segment is transmitted, the delay may vary over

time. But our simulation is made on local area network, so it is reasonable to compute the retransmission time once and use it. Because the intention of our system is to show how our proposed scheme work on various case of segment loss, the system let the user choose which segments will be lost in sending.

Then, we compare the simulation of cumulative acknowledgement scheme with our modification using five segments and shown the result of each. Our system is not simulating the entire TCP protocol but merely simulating the TCP's cumulative acknowledgement without including any other TCP's complex scheme and error handling. The system is design just to show how our modified system work and how much it can improved TCP's cumulative acknowledgement scheme. Firstly, the system let user to initiate the testing by allowing establishing connection between sender and receiver application. The real world TCP uses a three-way handshake to establish a connection. In our system, sender send a segment to receiver which identify that the current testing is using the cumulative acknowledgement or our modified one and the sender's IP address. The receiver accepts the connection by returning an acknowledgement. A round trip sample is also measured in connection establishment by subtracting the time at which the first segment is sent from the time at which the acknowledgement arrives. Then the round trip sample is used to calculate the round trip time which is used to compute the retransmission time. Then the user must configure the segment lost. Once the connection has been established and loss segments have been configured, the user can start testing. The sender application simply encapsulates the segment ID into TCP segments and sends it to the receiver. The receiver wills acknowledged by using the scheme they agreed on during connection establishment stage.

5. Experimental results

Assume that the NICs on our 10Mbps LAN are set to transmit data using Ethernet. Unfortunately some of the data gets consumed by the packaging necessary to get an Ethernet packet between two points. The transport layer adds source and destination port numbers as well as status bits and sequence numbers to our data. The IP layer adds a source and destination IP address. All totaled 40 Bytes are used to get our packet of data to the destination and to identify where the data come from. So every time a segment is sent, 40 Bytes of data is waste. It's much worse, because the source and destination Medium Access Control (MAC) addresses have take another 12 bytes per packet, plus 4 bytes for the Frequency Check Sequence (error detection) that is attached to every packet. It would take 0.04375ms to transmit a segment (56*8/10240000=0.00004375s).Therefore, the retransmission time is 0.0875ms.

In case 2, our proposed scheme only needs to retransmit first segment. In cumulative acknowledgement, if the sender chooses to retransmit all five segments, the scheme would save 56*4=224 bytes of network bandwidth. Notice that the data bytes haven't included in the calculation. If the sender choose to learn about loss packet one at a time, in case 2, it wouldn't take more time.

Using cumulative acknowledgement in case 3, if the sender choose to retransmit last four segments, it would waste 56*2=112bytes of network bandwidth. If the sender chooses to retransmit one packet per time, 2*0.875ms will have to be waited. Figure 6 shows the result of our simulator for case I. In case I, both schemes have no differences because both need to retransmit segment four and five. Figure 7 shows the simulator's result for case II.

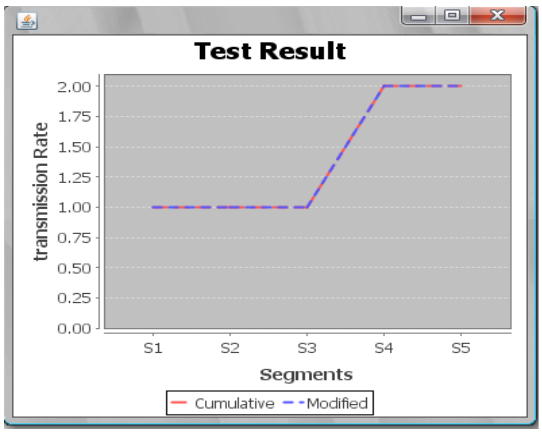


Figure 6.Simulator result for case I.

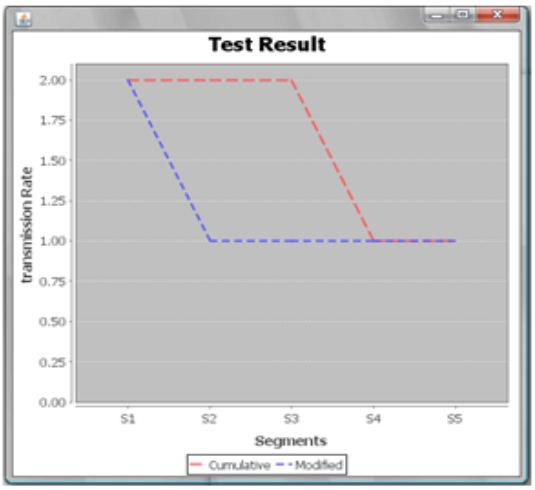


Figure 7.Simulator result for case II.

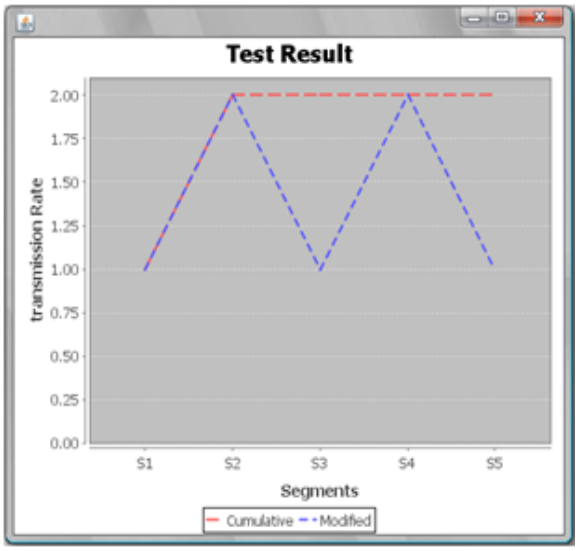


Figure 8.Simulator result for case III

Figure 8 shows the simulator result for case III. In case II and case III, our proposed system is more efficient because it only need to retransmit loss segment, segment one in case II and segment two and four in case III.

6. Conclusion and ongoing work

Although TCP has many advantages, it has many disadvantages. In this paper, we proposed a system that can correct one of the inefficiency of TCP caused by cumulative acknowledgement scheme. Our system is rather an application simulating the work of cumulative acknowledgement without including any other complex scheme used by TCP to show that the inefficiency can be improved and the system can only be tested for five segments. As an ongoing work a simulator which included others complex scheme and is capable of simulating and visualizing the operations of any TCP flow control algorithm working under any type of Internet data traffic conditions should be developed.

7. References

[1] Douglas E. Comer, “Internetworking with TCP/IP”, Fourth Edition, ISBN 0-13-019353-4.

[2] Stevens, W., TCP/IP Illustrated, Volume 1: The Protocols, Addison-Wesley, 1994.

[3] Fall, K. and Floyd, S., "Simulation-based Comparisons of Tahoe, Reno, and Sack TCP", July, 1996.

[4] Postel, J. , "Transmission Control Protocol–DARPA Internet Program Protocol Specification" , RFC 793 , DARPA , September 1981.

[5] Velten, D., Hinden, R., and J. Sax, "Reliable Data Protocol", RFC 908, BBN, July 1984.

[6] John Barnett, “Introduction to TCP/IP”, 2001.

[7] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, “TCP Selective Acknowledgement Options”, October, 1996.

[8] R. Fox, “TCP Big Window and Nak Options”, June, 1989.