ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645
Indian Journal of Science and Technology, *Vol 8(S2), 284–290, January 2015* DOI:.10.17485/ijst/2015/v8iS2/71720

Reliable Connectionless Transport Protocol for Fast Message Delivery

Pranav Patel*, Rutul Shah, Chirag Patel and V. Vijayarajan

VIT University, India; pranav.uv@gmail.com, shahrutul_29790@yahoo.com, chirag.patel3214@gmail.com, virtual.viji@gmail.com

Abstract

TCP and UDP are the most used transport layer protocols in computer networks. These two protocols are used for almost all types of data transmission including files, messages, real-time streaming of media etc. In cases where loss of a single data unit then it makes the entire data transfer useless such as file transfer, TCP is used because it ensures reliability. UDP is unreliable and can be used where little loss is acceptable and sender does not need a conformation of successful delivery to the receiver. In past decade a large variety of network applications has emerged requiring different requirements for transport layer protocol. Currently in cases where a message needs to be sent to a destination and sender needs confirmation of successful delivery of the message TCP must be used. However TCP requires 3 way handshakes and has large over head of header bytes which may be not required in such cases. This paper proposes a transport layer protocol that is connection less like UDP and has very less header overhead bytes and ensures reliability of transmission. This protocol saves from connection establishment phase and overhead of additional header bytes. From calculation it can be observed that this protocol improves efficiency of transmission in certain cases.

Keywords: Network, Protocol, Reliable, RCTP, TCP, Transport Layer, UDP

1. Introduction

Latest network technologies have produced a keen interest in the solution which can provide both reliability and faster communication. TCP has mechanisms to ensure the delivery of each segment to the receiver. But with these there are some considerations that must be noted. If any link which uses TCP/IP and notifies that the received segments are out of order then according to TCP mechanism it stops the transmission. Moreover, it also discards the segments which are out of order and it sends the message to go back to the place where it can find what went erroneous. After that again it starts the transmission. If any error occurs under TCP then it consumes 3 seconds to go back to the missing point and restart from that point with discarding the successive segments and resend them again. There can be a scenario where only one segment loss can cause the retransmission of entire window. TCP is a connection oriented protocol and it needs 3 way handshaking which includes SYN, SYN-ACK and ACK. This is the overhead to the network. In some cases where the time does not matter but deliver of segments is important, TCP is the right choice to use for transmission. Therefore TCP is mostly used in infrastructure based wired network so that it is not appropriate for wireless ad hoc networks such as MANETs. But in some situations faster transmission protocol is necessary where time matters. User datagram protocol is used where little loss of segment is acceptable, no requirement of providing the acknowledgement and where time matters. Often time sensitive application and real time applications use user datagram protocol (UDP) where loss of segment or dropping of segments is more desirable instead of waiting for retransmitted delayed segments. UDP is appropriate where error detection and error correction is not necessary or it can be handled by the applications. Moreover in UDP if two messages are sent then in which order they will arrive to receiver cannot be anticipated. Hence there is no ordering or tracking of messages. This

*Author for correspondence

avoids the processing overhead on the network. Therefore we can conclude that UDP is lack of reliability. It does not provide reliability. Sender is not aware of delivery of segments as it does not get any acknowledgement. It also does not provide the congestion control mechanism. So we have to find the intermediate solution which can provide both reliable transmission and faster transmission. To make the reliable connectionless protocol we need to add some features. The features can be added to the reliable connectionless protocol are as followed:

- Acknowledgement.
- Windowing and flow control.
- Finding the appropriate MTU size.
- Retransmission of lost packets.
- Congestion Control.

The acknowledgement provides the surety of delivering the segment that the segment has been reached to the recipient. Flow Control controls the volume of data a sender can send before getting an acknowledgement from the receiver. There are 2 cases can be occurred. In one case 1 byte can be sent through transport protocol and wait for acknowledgement and after getting ACK it will send next byte. This is tremendously slow process. In other case data can be sent through transport protocol without worrying about acknowledgements. This can speed up the process. But if some of the received segments are corrupted, out of order or missing then sender will not be aware of this until recipient checks all segments. Therefore one should think the intermediate solution. Sliding window protocol of TCP acts a major role in flow control mechanism. Transport layer protocol is responsible for connection oriented communication, reliability, flow control, congestion avoidance and multiplexing. But there are 2 significant functions for any transport layer protocol is focused in this paper. One is congestion control algorithm and second is reliable delivery. Discovering the appropriate MTU (Maximum Transfer Unit) is a major and important task. If the datagram is too large then datagram must be broken into pieces called as fragments. This process of breaking the datagram into small pieces is known as Fragmentation. This is expensive task because first router performs fragmentation and it is itself an expensive operation and second again intermediate routers perform fragmentation. So that the destination has to resemble all fragments and it carries additional headers. This is overhead to the network. If we set the MTU very small then the data we can transmit in one packet then the size is reduced. It causes more packets to send. Therefore it creates signaling overhead. Retransmission of lost packets is significant in this protocol. It decides to retransmit the packets by getting the idea from ACK or time out event.

2. Related Work

From the past few years various protocol has been developed. Among them some of connectionless protocol provides data transmission over the network where some provides connection oriented data transmission. Most of the proposed protocol main focus on providing data transmission. Thus the various proposal methods can be stated into two categories: connection oriented and connectionless protocol. The method proposed[9–11] are based on connectionless data transmission over the network. In which the data transmission rate at the sender side is based on the receiver as well as intimidates node which are in the path of transmission. Form this method of the data transmission the congestion can be accurate but the proposed algorithm in this paper cannot maintain the end-to-end communication over the transport protocol in data transmission and in addition to that at the sender side mathematical computation is required for windows controller which is very cereus task in data transmission.

In contrast with connectionless based protocol, UDP (User Datagram Protocol) is based protocol for major connectionless data transfer protocol. The proposed method in[12,13], in which at the source side based on some measured matric comes from reply messages congestion can be controlled, but the drawback of the method is that based on alone loss packet the congestion cannot be accurate indicates. Ei Rakabawy et al.[14–16], for reducing the ACK packets in the network the author from data and ACK packet used contention between them. But the major draw of this type of data transmission is that for congestion control it uses the AIMD base congestion control algorithm in which it is used very large data transmission window. Altman et al.[17] author proposed one method for multi hop network a small TCP sender window which size is limited by the maximum transmission in the BDP path. However, the window size id decrease means small window the AIMD cost is increases so that the source cannot detect the packet loss in the network while transmission, and also insufficient in throughput is occurred because of duplicate ACKs is not occurs sufficiently.

In connection oriented and connectionless transfer several reliable protocol has been developed which uses

SACK for transmission over the network. Based on some previous year's research the SACK base schemas can be partition in to two different type's first one is block-based and other one is bit-vector-based mechanism. In general the block-based SACK method have very large number of ACK so overhead is very high and in frequent data transmission it will provide limited information for loss of packets. Waldby et al.[7] author proposed that the bit-vector based SACK scheme can three different type of bit vector. The proposed method in Wilson So[8] is based on bit-vector-based for the transport protocol. In this method the protocol is depends on the time out of the packet loss, even if the packet contains either data packet or ACK packet unnecessary information with regarding to data transmission. In describe above both method uses noncumulative technique for ACK transmission but in this paper proposed algorithm is used reliable data transmission for connectionless services which provides batter enhancement over the existing technique.

Beside the describe above some connectionless as well as connection oriented protocol with reliability have some identified interaction in between the TCP layers as well as in UDP it is seen the some poor performance for the other type of networks. The proposed method in the Liu et al.[22–25] are provide solution for various type network interoperability in terms of the protocol efficiency. In this paper author has broad the issue related to the routing breakages, but this paper proposed algorithm main aim is to provide reliable connectionless fast data transmission over the network.

TCP (Transmission Control Protocol) is connection-oriented protocol based on that proposed one method in[23] which is establishes connection in form of virtually between source and destination. In this method data is transmitted over this virtual path for establishing the connection between the sender and receiver in the three steps and they are as followed connection establishment, data transfer and connection terminates. Generally this type of protocols is process to process communication. Flow controlling mechanism also provide for this type of protocol in order to achieve less data packet loss. In this purposed method have major drawback is that sender has to wait each ACK come from receiver side after sender can data means that the protocol is based on cumulative ACK. Although this type of communication provide reliable data transmission but it has no efficient transmission over the network because it increases the waiting time of the sender so indirectly the transmission time is also increases and the efficiency of the protocol is decreases in this method.

In paper[24,25] the proposed method is based on data transmission over the network for connectionless protocol so it can be said that it is UDP based method. The proposed method in this paper is not establish connection between the sender and receiver but virtual part is first decided at data link layer in this protocol in order make fast data transmission. For each sent data there will be ACK come from the receiver side to make protocol reliable for data transmission. In this method the data packets are divided into numbers of segments and then with join them sequence number and send to the receiver side and at receiver side there will maintain one queue for segments if any packet receives the ACK will send for this packet to the sender. Due to sending ACK for each segment which may result the network traffic in data communication. Although it achieve the connection less and reliability but increases the unnecessary network overhead.

3. Proposed Work

This section gives the idea of proposed methods in parts. Key features are explained in this section that is used in formation of this transport protocol.

3.1 Datagram Oriented Message Transfer

Instead of using a byte stream data transfer like in TCP this protocol uses stream of datagrams. I.e. the unit of transfer is datagram. Instead of each byte each datagram is provided a sequence number. The header sequence number does not give the first byte of that data like the case of TCP. In a datagram protocol fragmentation in the IP layer is not advised. So research like[18] has proposed to avoid IP layer fragmentation as much as possible. Because fragmentation requires use of extra processing added at the intermediate routers in the path that might be wastage of resources and result in insufficient communication mechanism. Also in case of fragmentation if any one fragment is lost or corrupt during transmission the receiver need to discard entire data gram and hence successfully

Source port (16)				Destination port (16)
Total datagram length (16)				Checksum (16)
S_1	S_0	N	Sequence Number (5)	Data(~)

Figure 1. RCTP datagram header format with 9 bytes.

transmitted fragments. This may lead of retransmission overhead and decrees overall efficiency of the protocol. Therefore dissection in this paper is based on assumption that the datagrams are small enough that does not require fragmentation at the IP layer, however this protocol is not restricted by that. Fragmentation can be used as and when required at IP layer but, that might reduce efficiency of the protocol.

3.2 ACK with Numbers

Figure shows the header format for ACK. New ACK technique is used for this protocol that is based on selective negative ACK. There is a flag in the header that indicates that the current window is completely received the sender may now start transmitting new window of datagrams; this flag is denoted by N. The ACK contains the list of sequence numbers of the segment that has not been received in the current window. ACK is send after a certain amount of waiting time after the first segment in the window is received. This ACK contains a vector that gives idea to the sender that these segments are not arrived. Sender after receiving ACK retransmits only those selected segment and waits for the ACK. The N flag is used in the ACK header because the sequence numbers are limited to current window. Hence after every complete window is received the sequence numbers are repeated.

3.3 Flow Control with Small Window Size

This protocol uses sliding window techniques for synchronization and flow control between sender and receiver. This is combined with ACK mechanism to provide error control. A fixed size of window is created depending on the need this size of window can be 8 or 16 or 24 or 32. This is informed to the receiver by 2 bit in the header field. These fields are denoted by S_1 and S_0 bits of header. Larger window size has less communication overhead but requires more waiting time. In case of large window size if network is slow than more retransmission accrues and some of them might be unnecessary as segment may arrive late. So in case of slow network small window size is desirable. In case of large window size large window size can improve performance by reducing iteration of send ACK.

Source port (16)				Destination port (16)
Total datagram length (16)				Checksum (16)
U_n	U_n	N	Sequence Number (5)	Sequence number vector (0–160)

Figure 2. RCTP ACK header format with 9 bytes.

3.4 Retransmission on Receiving ACK

When receiver receives datagrams transmitted by sender it waits for certain amount of time, this is to wait for other datagrams to arrive. After this time expires it send an ACK with list of those segments that are not arrived to the receiver. This ACK is received by the sender and sender only retransmits those datagrams whom sequence number is contained in the received ACK. This is way the amount of retransmission is controlled.

Sender first creates a window in the buffer as required which is based on the number of datagrams to be transmitted, type of network, available buffer space or any other para meter that sender would like to consider. Then a sender sends all this datagram as per their order labeled by sequence number in same order. Each datagram header contains size of window bits. They are S_1 and S_0. Window size is stored in these 2 bits in divide by 8 form i.e. binary value 00 indicates window size is 8 datagrams, binary value 01 indicates size is 16, binary value 10 indicates size is 24 and binary value 11 indicates size is 32. N flag is set to 1 if this is new window for which the transmission is happening for the first time otherwise the N bit is set to 0. Other than this 5 bit sequence number is stored in each datagram header. This datagrams are then transferred to next layer for transmission. The datagrams may arrive out of order at receiver. However each datagram contains size of window in the header field so when the receiver receives first segment it creates the window in buffer. Then places this segment depending on its sequence number for arrangement. Receiver waits a fixed amount of time for the other datagrams to arrive. This time is based on the size of window. In this time as other datagrams arrive they are place in the receiver's window as per their sequence number. After this waiting time expires receiver sends ACK with N bit set if the entire segment in the widow are received. If all the datagrams in this window are not received then receiver sends ACK with N bit as 0 and the ACK contains a list of sequence numbers of datagrams that are not received in the current window. After receiving this ACK sender either moves it window to next datagrams in the queue or retransmits only those datagrams for which sequence numbers are in ACK.

3.5 Sender's Procedure

1. Get data from above layer.
2. Create sender's window W_s.
3. Divide data into datagrams of equal size PL.
4. Prepare and add header to each datagram.
5. Send current window W_s.
6. Wait for ACK,

- If ACK N bit is set then go to step 1.
- Else read ACK for sequence number and re-transmit those datagrams. Go to step 6.

Initially transport layer acquires payload data from the above layer. This data will be then divided in number of data gram payloads. Each payload is appended with a RCTP header. And send to next layer (Network layer). The important part in this phase is to generate header for each datagram. Figure above shows various fields of header. Sender determines size of the window. This protocol provides sizes 8, 16, 24, 32. Various network conditions require use of different size of window. This size is indicated in header by S_0 and S_1 bits. The actual size is multiple of 8 of this value. N bit is set to 1 if this is completely new transmission and 0 if these are retransmitted datagrams of previous transmission. All segments in the window are then sent for transmission. After sending sender waits for ACK timeout period. If ACK is not received after timeout then sender re-transmits all segments in the current window. If ACK is received then it checks for N bit. If N bit is 1 then sender moves it window to next datagrams that are to be send. If N bit is 0 then sender extracts all the sequence numbers from ACK. Then selectively re-transmits those segments and again waits for ACK.

3.6 Receiver's Procedure

1. Get data from above layer.
2. Create receiver's window W_r it is not already created.
3. Read sequence number of current datagram. Order current datagram as per its sequence number in W_r.
4. Wait for timer to time out.
5. If all datagrams in W_r are received,
    - Then send an ACK with N bit set
    - Else send ACK with N bit as 0 and sequence numbers of those datagram that are not in W_r.

Receiver's process is simple than sender's process. Receiver when receives first segment creates window in its buffer. This is based on S_0 and S_1 bits. Then it places this datagram to its appropriate place based on its sequence number. Receiver waits until a predefined time out so that the entire segment can be received. After this time out receiver generates ACK header format of which is denoted in the figure. N bit is set to 1 if all the datagrams are received and receiver is ready to receive next set of segments. Otherwise N bit is set to 0. In this case the data part of ACK contains list of segments that are not received at the receiver. This ACK datagram is then send to sender.

4. Experiments and Results

We conduct a series of experiments on different types of machine for results evaluations and analysis.

4.1 Comparison of Throughputs between TCP and RCTP

Figure 3 represents the comparison of throughputs of TCP vs. RCTP. X-axis denotes the number of packets loss and Y-axis denotes the throughput in Kilo Byte (KB). Dashed line shows TCP throughput and continuous line shows RCTP throughput. The ideal case is considered when there is no loss of packets. In such case throughput of TCP and RCTP is same. It is seen that as the number of packet loss increases the TCP throughput decreases. At the same time as the number of packets loss increases RCTP throughput increases. Let's consider 2 packets are lost. It is assumed that one packet is lost from first 16 packets and second packet is lost from remaining 16 packets. Throughput has been calculated based on the probability of loss of packets. Therefore the final calculation is done such as

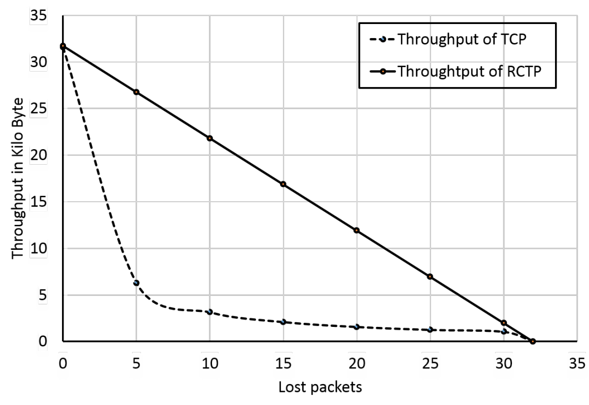

Figure 3. Comparison of throughput between TCP and RCTP.

Throughput of TCP = (W*p) * (PL – TCP header length)

Throughput of RCTP = (W – number of packet loss) * (PL – RCTP header length)
Where,

p = Window size/number of packet lost (Probability).
W = Window size.
PL = Payload length.

According to equation throughput of TCP in case of packet loss 2 is 32*(1/2)*1004 equals to 16064kb and throughput of RCTP is (32-2)*1015 equals to 30450.

In the Figure (4) we define graph shows the comparison between the number of packet loss in TCS and RCTP algorithm against the throughput of the algorithm in kilobytes. We take some scenario based on the mathematical analysis we proposed the graph. In this graph at time 0.5 millisecond we take one scenario that the total number of packet loss is 2 and their sequence numbers are 20 and 28. In this case the TCP will retransmit the packets from 21 to 31 where as new proposed algorithm will transmit only loss packets so the graph point as shown in graph will be plot for calculation at each point:

Throughput of TCP = (fist packet lost sequence number) * PL

Throughput of RCTP = (W – number of packet loss) * PL

For the second case we define the scenario that the total 7 packets are loss which have starting sequence number is 12 and so on, so the throughput of the both method are TCP throughput is (32-12)*1024 and for easy understand we convert the result into kilobyte so the result is 12 only. Where as in the RTCP method the throughput of is (32-7)*1024 so the result is 25. Let's take another scenario in which we take total packet loss are 16

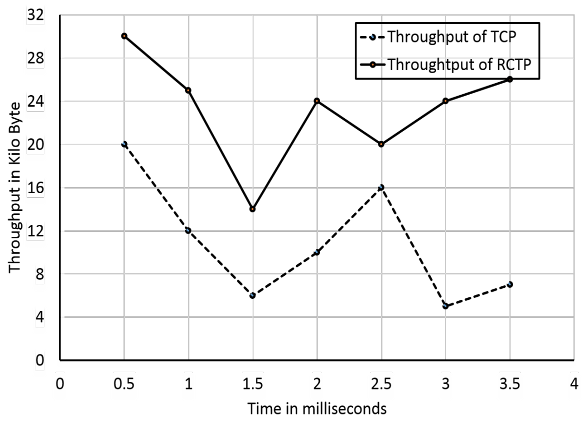

Figure 4. Comparison of throughput between TCP and RCTP for a particular scenario of network flow.

so that for the calculation of both methods are 20 and 16 receptivity.

5. Discussion and Conclusion

TCP is provides powerful mechanism for transport layer datagram transfer for in order, reliable and connectionless datagram transmission. However all this three properties might not needed in an application. This paper derives a transport mechanism called RCTP that is connection less but provides in order and reliable transmission of datagrams. Need of this type of mechanism is identified in the beginning. The RCTP is based on UDP protocol but adds reliability and oredered delivery of datagram. Later part of this paper describes properties of the RCTP and sender's and receiver's procedure. Then throughput of this protocol is measured with TCP. This protocol gives batter throughput in case of packet loss which is indicated in the next section. A scenario has been created to identify throughput in real time working of network. It is clear from the discussion that RCTP provides several advantage when compared to both TCP and UDP protocols. RCTP gives batter throughput and reduces retransmission in network.

6. References

1. Li X, Kong P-Y, Chua K-C, IEEE DTPA: A Reliable Datagram Transport Protocol over Ad Hoc Networks. IEEE Transactions on Mobile Computing. 2008 Oct; 7(10):1285–94.
2. Li J, Blake C, De Couto D, Lee HI, Morris R. Capacity of Ad Hoc Wireless Networks. Proceedings of ACM MobiCom'01. In: Li et al., editors. DTPA: A Reliable Datagram Transport Protocol Over Ad Hoc Networks 1293; 2001. p. 61–9.
3. Chen K, Xue Y, Shah S, Nahrstedt K. Understanding bandwidth-delay product in mobile Ad Hoc Networks. Computer Communications on Protocol Engineering for Wired and Wireless Networks. 2004; 27:923–34.
4. Fu Z, Zerfos P, Luo H, Lu S, Zhang L, Gerla M. The impact of Multihop wireless channel on TCP throughput and loss. Proceedings of IEEE Infocom'03; 2003. p. 1733–53.
5. Mathis M, Mahdavi J, Floyd S, Romanow A. TCP selective acknowledgement options. IETF RFC 2018; 1996.
6. Kettimuthu R, Allcock W. Improved selective acknowledgment scheme for TCP. Proceedings of International Conference on Internet Computing (IC'04); 2004. p. 913–9.
7. Waldby J, Madhow U, Lakshman TV. Total acknowledgements: A robust feedback mechanism for

end-to-end congestion control. Proceedings of ACM Sigmetrics '98; 1998. p. 274–5.

8. Wilson So H-S, Xia Y, Walrand J. A robust acknowledgement scheme for unreliable flows. Proceedings of IEEE Infocom'02; 2002. p. 1500–9.
9. Anastasi G, Ancillotti E, Conti M, Passarella A. TPA: A Transport Protocol for Ad Hoc Networks. Proceedings of 10th IEEE Symposium on Computers and Communications (ISCC '05); 2005. p. 51–6.
10. Chen K, Nahrstedt K, Vaidya N. The utility of explicit rate-based flow control in mobile Ad Hoc Networks. Proceedings of IEEE Wireless Communications and Networking Conference (WCNC '04); 2004. p. 1904–9.
11. Zhai H, Chen X, Fang Y. Rate-based transport control for mobile Ad Hoc Networks. Proceedings of IEEE Wireless Communications and Networking Conference (WCNC '05); 2005. p. 2264–9.
12. Sundaresan K, Anantharaman V, Hsieh H-Y, Sivakumar R. ATP: A reliable transport protocol for Ad Hoc Networks. IEEE Transcations on Mobile Computing. 2005 Nov; 4(6):588–603.
13. Fu Z, Greenstein B, Meng X, Lu S. Design and implementation of a TCP-Friendly Transport Protocol for Ad Hoc Networks. Proceedings of 10th IEEE International Conference on Network Protocols (ICNP '02); 2002. p. 216–25.
14. Ei Rakabawy S, Klemm A, Lindemann C. TCP with adaptive pacing for Multihop Wireless Networks. Proceedings of ACM MobiHoc '05; 2005. p. 288–99.
15. Singh A, Kankipati K. TCP-ADA: TCP with adaptive delayed acknowledgement for mobile Ad Hoc Networks. Proceedings of IEEE Wireless Communications and Networking Conference (WCNC '04); 2004. p. 1679–84.
16. Oliveira R, Braun T. A dynamic adaptive acknowledgment strategy for TCP over Multihop Wireless Networks. Proceedings of IEEE Infocom '05; 2005. p. 1863–74.
17. Altman E, Jimenez T. Novel delayed ACK Techniques for improving TCP performance in Multihop Wireless Networks. Proceedings of 8th International Conference Personal Wireless Communications (PWC '03); 2003. p. 237–53.
18. Chen K, Xue Y, Nahrstedt K. On setting TCP's Congestion window limit in mobile Ad Hoc Networks. Journal of Wireless Communications and Mobile Computing. 2002; 2(1):85–100.
19. Kent C, Mogul J. Fragmentation considered harmful. Proceedings of ACM SIGCOMM '87. 1987; 17(5):390–401.
20. Li X, Kong PY, Chua KC. Analysis of TCP Throughput in IEEE-802.11-Based Multi-Hop Ad Hoc Networks. Proceedings of 15th IEEE International Conference on Computer Communications and Networks (ICCCN '06); 2006. p. 297–302.
21. Chandran K, Raghunathan S, Venkatesan S, Prakash R. A feedback-based scheme for improving TCP performance in Ad Hoc Wireless Networks. Proceedings of 18th IEEE International Conference Distributed Computing Systems (ICDCS '98). 1998; 8(2):34–9.
22. Liu J, Singh S. ATCP: TCP for Mobile Ad Hoc Networks. IEEE Journal of Selected Areas in Communications. 2001; 19(7):1300–15.
23. Wang F, Zhang Y. Improving TCP Performance over Mobile Ad Hoc Networks with out-of-order detection and response. Proceedings of ACM MobiHoc '02; 2002. p. 217–25.
24. Kim D, Toh C, Choi Y. TCP-Bus: Improving TCP Performance in Wireless Ad Hoc Networks. Journal of Communications and Networks. 2001; 3(2):175–86.
25. Kopparty S, Krishnamurthy S, Faloutous M, Tripathi S. Split TCP for Mobile Ad Hoc Networks. Proceedings of IEEE Global Telecommunications on Conference (GLOBECOM '02); 2002. p. 138–42.
26. Holland G, Vaidya N. Analysis of TCP Performance over Mobile Ad Hoc Networks. Wireless Networks. 2002; 8(2):275–88.
27. Dyer T, Boppana R. A comparison of TCP performance over three routing protocols for mobile Ad Hoc Networks. Proceedings of ACM MobiHoc '01; 2001. p. 56–66.
28. Xu K, Gerla M, Qi L, Shu Y. TCP unfairness in Ad Hoc Wireless Networks and a neighborhood RED solution. Wireless Networks. 2005; 11(4):383–99.
29. Xu Y, Wang Y, Lui J, Chiu D. Balancing throughput and fairness for TCP flows in Multihop Ad Hoc Networks. Proceedings of 5th IEEE International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt); 2007.