[JOURNAL OF ADVANCES IN SCIENCE AND TECHNOLOGY
VOL.-II, ISSUE - I] August 1, 2011
ISSN-2230-9659

Performance of TCP/IP Model in Multihop Adhoc Networks

Inderpal Singh
Research Scholar, Singhania University
Rajasthan, INDIA

ABSTRACT

Incorporating the concept of TCP end-to-end congestion control for wireless networks is one of the primary concerns in designing ad hoc networks since TCP was primarily designed and optimized based on the assumptions for wired networks. In this study, our interest lies on tackling the TCP instability and in particular intra-flow instability problem since due to the nature of applications in multihop ad hoc networks, connection instability or starvation even for a short period of time can have a negative impact on the Quality of Service and may not be acceptable for the end user. Through a detailed analysis, it will be shown that the main causes of TCP intra-flow instability lies in overloading the network by sending more packets than the capacity of the channel. Based on this, the paper proposes a novel cross layer solution called "TCP Contention Control" that dynamically adjusts the amount of outstanding data in the network based on the level of contention experienced by packets as well as the throughput achieved by connections. The simulation results show TCP Contention Control can drastically improve TCP stability over 802.11 multihop ad hoc networks.

Index Terms – Contention, Intra-Flow Instability, Multiple Ad Hoc Networks, TCP.

I. INTRODUCTION

Multihop ad hoc networks are collection of wireless nodes dynamically forming a temporary network without the use of any preexisting network infrastructure or centralized admin- istration. Consequently, ad hoc networks are fundamentally different from conventional stationary wireless and wired computer networks. During recent years, ad hoc networks have attracted considerable commercial and research interest. In particular, the de facto adoption of the popular IEEE 802.11 standard [1] has further fuelled the deployment of wireless transceivers in a variety of computing devices such as PDAs by ensuring inter-operability among vendors thereby aiding the technology's market penetration. However, as initially the deployment of these wireless technological advances came in the form of an extension to the fixed LAN infrastructure model, the 802.11 standard was mostly evolved and optimized for infrastructure-based wireless LANs rather than ad hoc networks.

To enable seamless integration of ad hoc networks with the Internet, TCP seems to be the natural choice for users of ad hoc networks that want to communicate reliably with each other and with the Internet. Here also, despite the fact that in theory TCP should not care whether the network layer is running over wired or wireless connections, in practice, this does matter because TCP has been carefully optimized based on assumptions that are specific to wired networks.

For instance, since bit error rates are very low in wired networks, nearly all TCP versions assume that packet losses are due to congestion and therefore invoke their congestion control mechanism in response to such losses. On the other hand, because of wireless medium characteristic and multihop nature of ad hoc networks, such networks exhibit a richer set of packet losses, including medium access contention drops, random channel errors and route failure where in practice each are required to be addressed differently. Ignoring these properties of wireless ad hoc networks can obviously lead to poor TCP performance as shown in previous research studies (e.g. [2]–[10]).

Not surprisingly, multihop ad hoc networks exhibit serious performance issues when TCP runs over IEEE 802.11 as neither TCP nor IEEE 802.11 MAC have been designed based on the properties of such networks. Therefore, during the recent years, a number of research studies have highlighted some of the problems TCP encounters in ad hoc networks [4], [10]–[18]. However, one of the key areas that has not attracted enough attention and needs to be addressed further in the deployment of TCP in ad hoc networks is the problem of TCP instability where the receiver (data sink) does not receive any packets for a period of time and therefore the connection throughput drops to zero or fluctuates rapidly. In particular, due to the nature of scenarios in which ad hoc networks are used (e.g. emergency operation and battlefield communication), disconnectivity or starvation even for a short period of time can have a devastating impact on the Quality of Service and may not be acceptable for the end user. In other words, ad-hoc network users are likely more willing to receive a continuous and stable flow of data rather than sending/receiving large bulk of data instantly.

In general, TCP instability can be broken down into two broad categories named as TCP intra-flow and TCP inter- flow instability, where the former is caused by the interaction of nodes belonging to the same TCP connection, while the latter happens when nodes belonging to different connections interact. Due to complexity and different nature of TCP intra- flow and inter-flow instability, this paper only investigates the TCP intra-flow instability problem in fine details.

The rest of this paper is organized as follows: section II analyzes the underlying cause of TCP intra-flow instability by giving a number of simple but yet important examples that will shed lights on the roots of the problem. Some of the most important related work are reviewed in section III. Section IV presents the details of the proposed cross layer solution that aims to alleviate the intra-flow instability issue in multihop ad hoc networks. The simulation results and analysis are given in section V. Finally, section VI summarizes the paper and outlines the future direction in this area.

Manuscript received Septmber 20, 2007. E. Hamadani is with the Centre for Communication Systems Research, University of Surrey, GU2 7XU, UK (phone: +44 (0) 148-368-3424, fax: +44 (0) 148-368-6011, e-mail: E.hamadani@surrey.ac.uk).
V. Rakocevic is with the Information Engineering Research Centre, City University, London, EC1V 0HB, UK, (e-mail: V.rakocevic@city.ac.uk).

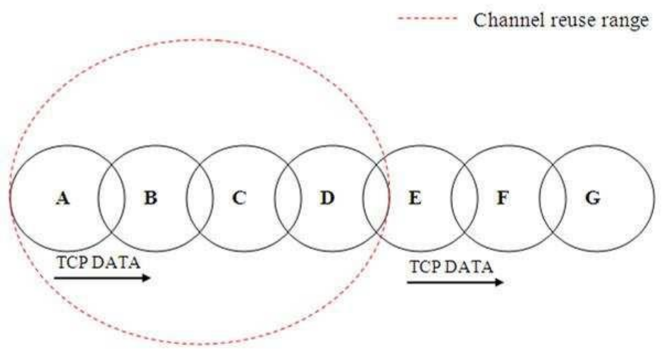

Fig. 1: TCP packet self interference.

II. INT RA-FL OW INSTABIL IT Y

A. Description

TCP intra-flow instability refers to the situation where the successive transmissions of packets in a single TCP flow, interfere with each other (link layer intra-flow interference) and result in large number of contention related packet drops and hence TCP instability in the network. Therefore, we begin our discussion of TCP intra-flow instability by reviewing different types of intra-flow interference and their impact on TCP instability.

B. Intra-flow Interference

As mentioned earlier, the intra-flow interference refers to situations where transmission interference in a single TCP flow causes packet drop in the network. In particular, when TCP runs over 802.11, the intra-flow interference can be broken down into the following categories:

1) Interference of TCP packets with each other
2) Interference between TCP packets and 802.11 control packets
3) Interference of 802.11 control packets with each other

Here, TCP packets refer to either TCP DATA or TCP ACK packets and 802.11 control packets include MACK (802.11 acknowledgements) and Request To Send/Clear To Send (RTS/CTS) if used.

To investigate and explain each category, in all subsequent analysis it is assumed one TCP flow is running on a 6 hop chain from node A (as data source) to node G (as data sink) and the transmission range of nodes is shown by a circle around them.

1) TCP self interference

In principle, the TCP self interference is caused by two effects. One is the interference caused between TCP DATA (TCP ACK) packets transmission with each other which prevents concurrent transmissions within a neighborhood area. For instance, as shown in figure 1, a transmission from node A interferes with node C, which cannot simultaneously communicate with node D. Similarly, a transmission by node D may cause a collision at node B. This type of interference can harm TCP mainly in two ways. Firstly, it greatly decreases the TCP throughput in ad hoc networks since in most occasions, very few simultaneous packet transmissions can occur in the network. For instance, in the above example, links A-B and E-F represent maximum possible concurrent channel usage while if link D-E is active, only one simultaneous transmission is possible. The other impact of such interference is on increasing the end-to-end delay. This is also because a successful transmission can occur only if nodes within the spatial channel reuse of that node are silent during the entire transmission. This means packets have to wait for a relatively long period of time in the node's buffer before the node can get a chance to access the channel. Therefore, the packets in multihop connections experience longer queuing delay and hence larger end-to-end delay.

The second part of the TCP self interference is caused by interference between TCP DATA and TCP ACK packets along the forward and return paths, respectively. In essence, this interference can specially result in TCP ACK drop as there are larger number of TCP DATA frames on the forward route compared to the smaller number of the TCP ACK packets in the return path. So, the medium will be on average mostly accessed by TCP DATA frames and as a result significant amount of ACKs will be lost because of collisions while accessing the channel.

2) **TCP and 802.11 control packets interference**

The other type of intra-flow interference in the link layer happens between the TCP packets (TCP DATA or TCP ACK) and one of the 802.11 control packets (RTS, CTS, or MACK). However, it is important to note that regarding 802.11 MAC timing specification, the DCF protocol ensures that CTS frame transmission will be successfully received at its destination (the one who sent the RTS), if the CTS frame has been issued in response to the RTS. This is because successful RTS frame trans- mission silences all the nodes in the neighborhood of the source either for a duration specified in the duration field of the RTS or for EIFS time (if collision occurs) which is large enough to transmit a CTS. Therefore, the CTS frame cannot collide with any frame at the source. Using a similar argument, it can be concluded that a successful TCP frame transmission ensures a successful MACK frame transmission. Thus, there cannot be CTS and MACK frames drop at the intended destination (the node who is waiting to receive the packet) because of medium contention.

Figure 2 reviews one the most common scenarios of TCP packet drop due to 802.11 control and TCP packets collision. Here, station D has TCP DATA to send to E and it sends its data after a RTS/CTS handshake with node E. Meanwhile B has a TCP DATA to send to C, thus starts its own

RTS handshake. However, due to ongoing TCP DATA transmission between D and E, the B's RTS is dropped at C. Although B resends the RTS after performing an exponential backoff, in most of the cases all its RTS retransmissions (7 by default) are collided at node C and therefore node B drops the TCP packet1.

3) 802.11 control packets self interference

The last type of intra-flow interference happens between 802.11 RTS, CTS control packets if the RTS/CTS hand- shake is used prior to data transmission. We should note that despite the small size of RTS and CTS packets

1 This is due to the relatively large amount of time the channel is occupied when sending TCP DATA.

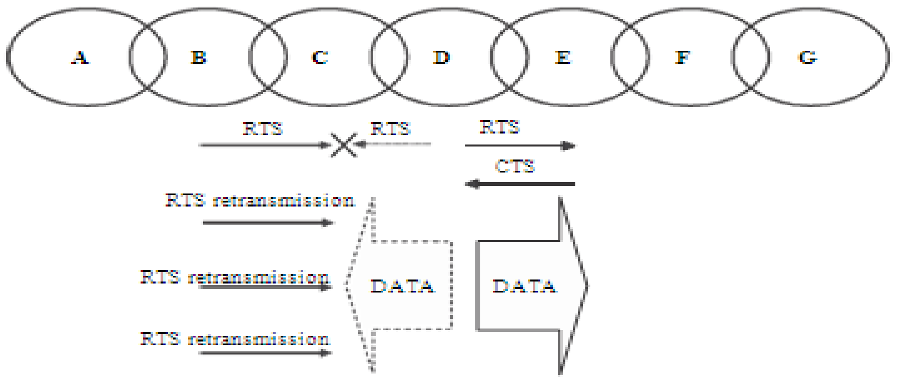

Fig. 2: RTS & TCP DATA collision.

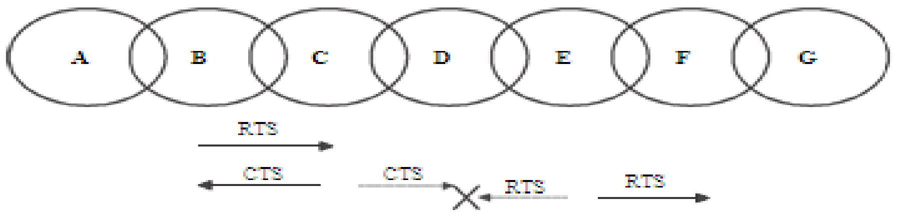

Fig. 3: 802.11 control packets collisions.

compared to data packets, the frequent losses of control packets can waste channel resources and have undesir- able impact on the performance of higher layers. On the other hand, the self interference between the RTS and CTS control packets can fail the channel reservation scheme and lead to a loss of data packets as well. Figure 3 depicts a typical scenario of control packets self interference and loss of TCP packet as a result. Here B starts an RTS-CTS handshake with C before transmitting a TCP packet. The CTS reply from C is received by B correctly, but it is not received by D, which is hidden from B, due to a collision with an RTS packet sent from E to F. This happens because E, being far away from both B and C, does not hear either the RTS or the CTS packet and is unaware of the communication between B and C. Node B assumes that the channel is successfully reserved and proceeds with transmission of the data packet to C. Therefore, the TCP transmission from B is vulnerable to interference from D, which has not been able to set its Network Allocation Vector (NAV) accordingly, and may initiate a transmission to any of its neighbors before the data transmission is over.

C. Intra-flow Interference and Intra-flow Instability

Having reviewed the three types of intra-flow interferences, let us explain in more detail how such link layer interfer- ences and packet drops can create TCP intra-flow instability. However, before that and to show the impact of intra-flow interference on TCP stability, let us review figure 4 that shows the change of congestion window (cwnd) and the instances of TCP retransmission in a static 6 hop chain topology (similar to figure 1) using 802.11 MAC.

Here, to confine the packet losses to contention drop, it is assumed the channel is error-free, no routing messages are exchanged between the nodes and all nodes have infinite buffers. Therefore, the packet losses and retransmissions are restricted to intra-flow interference related drops. It is clear from the result in figure 4 that intra-flow interference can trigger a large number of TCP retransmissions/TCP congestion window fluctuation and therefore TCP instability.

To explain further how intra-flow interference can cause TCP instability, we should note that according to 802.11 MAC standard, if a node cannot reach its adjacent node within the limited number of allowed retries (MAC-Retry-Limit), it will drop the packet. These packet drops are wrongly perceived as congestion by the TCP and result into false trigger of TCP congestion control algorithm, frequent TCP retransmissions and therefore TCP instability.

Having shown the impact of intra-flow interference on TCP instability, the next question is how it is possible to minimize the intra-flow interference in multihop ad hoc networks? The answer to this question is not straightforward as each type of intra-flow interferences discussed above are different in nature and therefore needs to be addressed separately. For instance, TCP packets or 802.11 control packets self interference can be best eliminated by designing a smart decentralized link layer schedule that coordinates the concurrent transmission between different pairs to maximize the channel utilization and minimize the number of collisions. On the other hand, TCP with 802.11 control packets interference is best to be addressed by reconsidering the link layer timing specifications, packet transmission coordination and prioritization. However, such schemes can be quite topology dependent and confined to specific scenarios. This is obviously hard to achieve in dynamic multihop ad hoc network environments where the topology of the network is changing rapidly and it is not feasible to propagate global topology information to individual nodes. More importantly, due to scarce channel resources, it is simply unrealistic to broadcast information regarding the topology and the current activity of nodes across the network. The next alternative solution is to alleviate all types of intra-flow interferences discussed above by controlling the amount of outstanding data in the network. It should be noted that, though this approach does not fully solve the individual

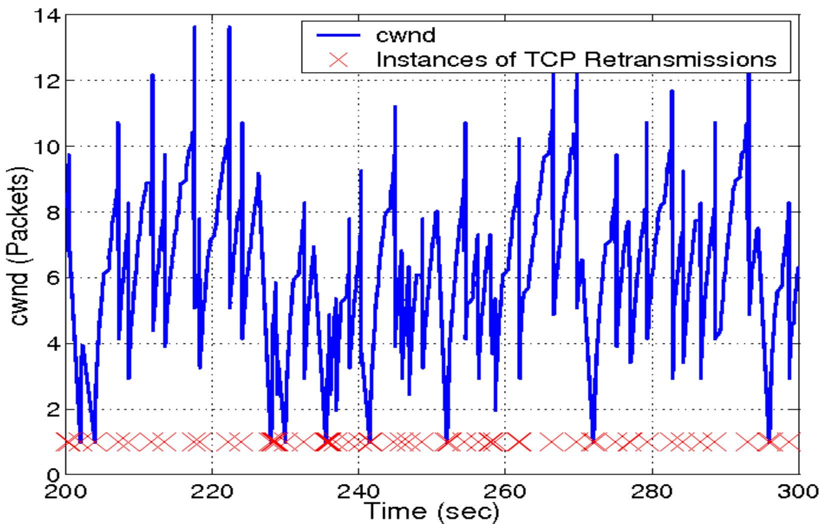

Fig. 4: Illustration of TCP congestion window change in a 6 hop chain topology.

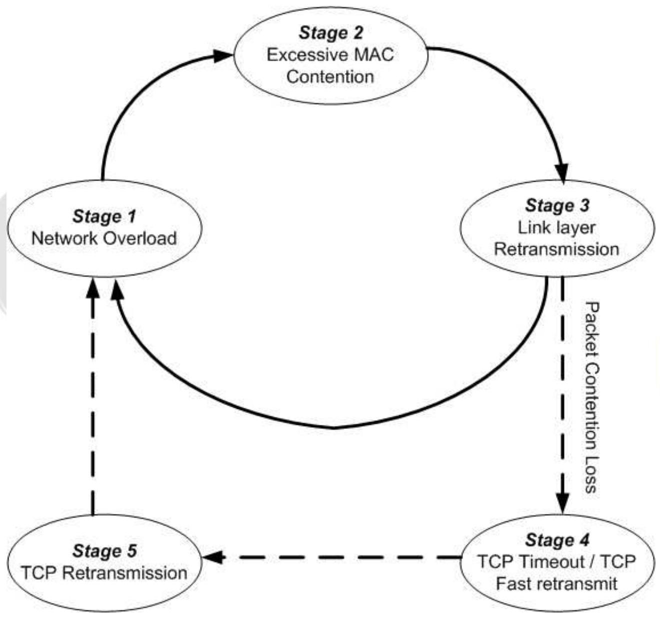

Fig. 5: Network overload and intra-flow instability cycle.

intra-interference scenarios explained before, it addresses the problem by minimizing the unnecessary intra-flow interference and its adverse effects on TCP instability. To better understand this, we should note that the performance of TCP directly depends on its flight size (swnd) which its optimal value should be proportional to bandwidth-delay product (BDP) of the entire path of the data flow [6], [19]. The excess of this threshold does not bring any additional performance enhance- ment, but only leads to increased queue size in intermediate nodes along the connection. On the other hand, as shown in [4], [6], [14], [20], the BDP of a TCP connection over multihop 802.11 networks tends to be very small. This is mainly because in 802.11, the number of packets in flight is limited by the per-hop acknowledgements at the MAC layer. Such property is clearly quite different from wire-line networks, where multiple packets can be pushed into a pipe back-to-back without waiting for the first packet to reach the other end of the link [21]. Therefore, as compared with that of wired networks, ad hoc networks running on top of 802.11

MAC, have much smaller BDP. However, as shown in [4], [14], TCP grows its congestion window far beyond its optimal value and overestimates the available BDP. Figure 5 explains the chain of events that occur following a network overload and lead to TCP intra-flow instability.

The intra-flow instability cycle initially starts when increas- ing the network overload (stage 1) causes more contention among nodes as all of them try to access the channel (stage 2). On the other hand, when the level of contention goes up, more packets need to be retransmitted as the probability of collision increases with the increasing level of contention (stage 3). This in turn introduces extra network overload and therefore closing the inner part of the cycle (stage 1 → stage 2 → sage 3 → stage 1). This cycle is continued until one or more nodes cannot reach its adjacent node within a limited number of tries (specified by the MAC Retry Limit in 802.11 MAC standard) and drop the packet (packet contention loss). This packet loss is then recovered by the TCP sender either through TCP fast retransmit or through TCP timeout (stage 4). In both cases, TCP drops its congestion window resulting in a sharp drop in number of newly injected packets to the network (stage 5) and therefore giving the network the opportunity to recover. However, soon after TCP restarts, it creates network overload again by overestimating the available BDP of the path, and the cycle repeats.

III. RELATEDWORK

During recent years, many studies have shown that limiting the amount of outstanding data in the network can greatly improve TCP stability. In an early paper by Gerla et.al. [3], the authors showed by simulations that TCP performance degrades for congestion window greater than 1 packet when the MAC layer offers no ACK protection; They further showed that, with the link-layer ACK (MACK) protection, certain performance gain can be realized by allowing a slightly larger congestion window (2-3 packets).

To limit the amount of outstanding data in the network, [22] proposed to adjust the maximum window size parameter to 4 packets as this is the smallest value of window size for facilitating the fast retransmission scheme for TCP connections running over IEEE 802.11 based ad-hoc networks.

The authors in [8] showed that due to the spatial reuse and transmission interference property of the IEEE 802.11 MAC layer protocol in a chain topology, a sensible choice is to set TCP congestion window to 1 h, where h is the length of the chain. They ^{4}further showed that TCP tends to overshoot this optimal value and operates in larger window size as we explained earlier. In the same paper, they proposed Link-layer Random Early Dropping (LRED), which aims to control the TCP window size by tuning the link-layer dropping probability according to the perceived channel contentions. In essence, similar to the RED algorithm [23] with a linearly increasing drop curve as the queue size exceeds a minimum value, LRED increases its packet dropping probability when the link- layer contention level, measured by the retransmission counts, exceeds a minimum threshold. To this aim, in LRED the link layer maintains a moving average of the number of packet retransmissions and the head-of-line packet is dropped/marked with a probability based on this average retransmission count. In particular, at each node, if the average retransmission count is smaller than a minimum threshold, the head-of-line packets are transmitted as usual. When the average retransmission count becomes larger, the dropping/marking probability is set as the minimum of the computed dropping probability and an upper bound. It was shown there that LRED can provide an early sign of network overload to the transport layer protocol and therefore force the TCP sender to reduce its transmission rate.

Finally, to decrease the amount of channel contention in the network, the authors in [24] present a cross layer approach named adaptive TCP that adaptively adjust the TCP maximum window size according to the number of RTS retransmissions at the MAC layer at the TCP sender to control the number of data packets in the network and thus decrease the channel contention.

III. TCP CONTENTION CONTROL

A. Description

As discussed earlier, a high percentage of intra-flow interference and therefore contention drops can be eliminated by decreasing the amount of traffic load in the network. However, this can be a very challenging task. On one hand, if the amount of data in the network is reduced beyond the bandwidth delay product (BDP) of that flow, the channel resources are under- utilized. On the other hand, any increase above the BDP does not bring additional performance enhancement, but only leads to increased queue size in intermediate nodes along the connection and other consequences as discussed earlier. Therefore, the main question in limiting the traffic load is how to set properly the amount of outstanding data in the network to achieve maximum throughput while minimizing the queueing delay experienced by individual packets.

In this section, the above issue is addressed by introducing a cross layer solution called TCP ConTention Control (TCTC). In simple words, TCTC adjusts the TCP transmission rate to minimize the level of unnecessary contention in the in- termediate nodes. To this aim, during fixed probe intervals, TCP receiver monitors both the achieved throughput and the level of contention experienced by packets during that interval. Then, based on these observations, the receiver estimates the optimum amount of traffic to get the maximum throughput and the minimum contention delay for each connection. Finally, the TCTC propagate the information back to the sender to adjust its transmission rate.

Using this information, the TCP sender now sets its trans-mission rate not merely based on the level of congestion in the network and the available buffer size at the receiver but also on the level of medium contention experienced by intermediate nodes. More precisely, while TCP congestion control adjusts the TCP transmission rate to avoid creating congestion in the intermediate network buffers, TCP contention control adjusts the TCP transmission rate to avoid creating queue build up in the intermediate network buffers. Therefore, the main advantage of TCTC over previous algorithms (e.g. [3], [8], [22]) is its ability to adjust the TCP transmission rate dynamically based on the current level of network load.

Since the key element in TCTC is optimum TCP flight size, in the following we explains how TCTC estimates the optimum TCP flight size.

REFERENCES

[1] "IEEE Standards for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY), Part 11:Technical Specifications", 1999.

[2] R. Caceres and L. Iftode, "Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments", *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 5, 1995, pp. 850–857.

[3] M. Gerla, R. Bagrodia, L. Zhang, K. Tang, and L. Wang, "TCP over Wireless Multi-Hop Protocols: Simulation and Experiments", in *Proc. IEEE ICC*, Vancouver, Canada, June 1999.

[4] Z. Fu, X. Meng, and S. Lu, "How Bad TCP Can Perform in Mobile Ad Hoc Networks", in *Proc. IEEE International Symposium on Computers and Communications*, pp. 298–303, Taormina, Italy, July 2002.

[5] X. Li, P.-Y. Kong, and K.-C. Chua, "Analysis of TCP Throughput in IEEE 802.11 Based

Multi-hop Ad hoc Networks", in *Proc. International Conference on Computer Communications and Networks (ICCCN)*, San Diego, USA, October 2005.

[6] X. Chen, H. Zhai, J. Wang, and Y. Fang, "TCP Performance over Mobile Ad Hoc Networks", *Canadian Journal of Electrical and Computer Engineering*, vol. 29, no. 1, 2004, pp. 129–134.

[7] M. D. Baba, N. Ahmad, M. Ibrahim, R. A. Rahman, and N. H. Eshak, "Performance Evaluation of TCP/IP Protocol for Mobile Ad hoc Network", *WSEAS Transactions on Computers*, vol. 5, no. 7, 2006, pp. 1481–1486.

[8] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla, "The Impact of Multihop Wireless Channel on TCP Throughput and Loss", in *Proc. IEEE INFOCOM*, San Franciso, CA, USA, 2003.

[9] Y. Wang, K. Yu, Y. Liu, and H. Zhang, "Analysis and Improvement of TCP Performance in Mobile Ad Hoc Networks", in *Proc. International Conference on Communication Technology*, Beijing, China, April 2003.

[10] W. Xu and T. Wu, "TCP issues in mobile ad hoc networks: Challenges and solutions", *Journal of Computer Science and Technology*, vol. 21, no. 1, 2006, pp. 72–81.

[11] H. Lim, K. Xu, and M. Gerla, "TCP Performance over Multipath Routing in Mobile Ad Hoc Networks", in *Proc. IEEE International Conference on Communications*, Anchorage, Alaska, USA, May 2003.

[12] H. Elaarag, "Improving TCP Performance over Mobile Networks", *ACM Computing Surveys*, vol. 34, no. 3, 2002, pp. 357–374.

[13] A. Ahuja, S. Agarwal, J. P. Singh, and R. Shorey, "Performance of TCP over different routing protocols in mobile ad-hoc networks", *IEEE Vehicular Technology Conference*, Tokyo, Japan, 2000.

[14] Z. Fu, H. Luo, P. Zerfos, S. Lu, L. Zhang, and M. Gerla, "The Impact of Multihop Wireless Channel on TCP Performance", *IEEE Transactions on Mobile Computing*, vol. 4, no. 2, 2005, pp. 209–221.

[15] L. Zhang, X. Wang, and W. Dou, "Analyzing and Improving the TCP Flow Fairness in

Wireless Ad Hoc Networks", *Ruan Jian Xue Bao/Journal of Software*, vol. 17, no. 5, 2006, pp. 1078–1088.

[16] H. Zhai, X. Chen, and Y. Fang, "Improving Transport Layer Performance In Multihop Ad Hoc Networks by Exploiting MAC Layer Information", *IEEE Transactions on Wireless Communications*, vol. 6, no. 5, 2007, pp. 1692–1701.

[17] S. Xu and T. Saadawi, "Revealing the Problems with 802.11 Medium Access Control Protocol in Multi-Hop Wireless Ad Hoc Networks", *Computer Networks*, vol. 38, no. 4, 2002, pp. 531–548.

[18] H. Balakrishnan and V. Padmanabhan, "How Network Asymmetry Affects TCP", *IEEE Communications Magazine*, vol. 39, no. 4, 2001, pp. 60–67.

[19] K. Chen, Y. Xue, and K. Nahrstedt, "On Setting TCP's Congestion Window Limit in Mobile Ad Hoc Networks", *IEEE International Conference on Communications*, Anchorage, Alaska, USA, May 2003. [20] J. Song, K. Ahn, D. Cho, and K. Han, "A Congestion Window Adjustment Scheme for Improving TCP Performance Over Mobile Ad- Hoc Networks", *Lecture Notes in Computer Science*, vol. 4104 NCS, 2006, pp. 404–413.

[21] D. Katabi, M. Handley, and C. Rohrs, "Congestion Control for High Bandwidth-Delay Product Networks", *Computer Communication Re- view*, vol. 32, no. 4, 2002, pp. 89–102.

[22] S. Xu and T. Saadawi, "Revealing and Solving The TCP Instability Problem in 802.11 Based Multi-Hop Mobile Ad Hoc Networks", *IEEE Vehicular Technology Conference*, Atlantic City, USA, September 2001.

[23] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance", *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, 1993, pp. 397–413.

[24] Y. R. Y. Xiao, X. Shan, "Cross-Layer Design Improves TCP Per- formance in Multihop Ad Hoc Networks", *IEICE Transactions on Communications*, vol. E88-B, no. 8, 2005, pp. 3375–3381.

[25] R. Braden, "RFC 1122 - Requirements for Internet Hosts - Communi- cation Layers", Oct 1989. http://www.ietf.org/rfc/rfc1122.txt .