A Link Layer Adaptive Pacing Scheme for Improving Performance of Wireless Mesh Networks

A. Antony Franklin, B. Venkata Ramana and C. Siva Ram Murthy*
Department of Computer Science and Engineering
Indian Institute of Technology Madras, Chennai 600036, India
{antony,vramana}@cse.iitm.ernet.in, murthy@iitm.ac.in

***Abstract*— In this paper, we propose an adaptive pacing scheme at the Link layer for IEEE 802.11 based multihop wireless networks. Our objective is to improve the performance of higher layer protocols without any modifications to them. Our adaptive pacing scheme estimates the four-hop transmission delay in the network path without incurring any additional overheads, and accordingly paces the packets to reduce the contention in the network. We implemented the Link layer adaptive pacing scheme in ns-2.29 network simulator and extensively studied its performance for both User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) traffic in different network scenarios. In all the cases our scheme shows a significant improvement in the performance of both UDP and TCP.**

I. Introduction

Wireless Mesh Networking has emerged as a promising technology to meet the challenges in next generation wireless networks [1]. In a Wireless Mesh Network (WMN), backbone network by the mesh nodes is a multihop wireless network. In WMNs such as community networking, the clients are connected to the edge nodes of the backbone network. In the backbone network some nodes are called gateway nodes which provide Internet connectivity to the clients. This multihop wireless network has to be utilized efficiently to improve the overall performance of the network. Most of the Internet based applications use Transmission Control Protocol (TCP) as a transport protocol, since it provides end-to-end reliable transmission of data. Many streaming applications such as audio and video streaming use User Datagram Protocol (UDP) as a transport protocol, since they require faster delivery of data rather than reliable transmission. Hence, the traffic in WMNs constitute both UDP and TCP flows. The performance of TCP is greatly affected by the packet loss in the network. In wired networks, the packet loss is mainly due to the buffer overflow at the intermediate routers. But in the case of multihop wireless networks, the packet loss could also occur due to erroneous wireless channels, unstable network conditions, and mobility of nodes. The bursty traffic increases the contention and the packet loss in the network, thereby affecting the performance of the multihop wireless networks. It is shown in the literature that, in multihop wireless networks if the interference range is twice the transmission range, the contention in the network path can be reduced by evenly spacing the packets using the 4-hop transmission delay (FHD) at the source node [2]. The FHD is defined as the time for transmitting a packet from a node to the 4th hop node on the downstream path. The spacing between packet transmissions can be done at the transport layer by properly estimating the FHD in the network path. Although, this may give delay between successive transmission of packets for each flow at the higher layer, due to very high contention in the network or traffic from the multiple flows, in reality, this may not take place at the Medium Access Control (MAC) layer.

In multihop wireless networks like WMNs, a number of clients can generate TCP and UDP traffic which goes in the same multihop path from one edge node to another edge node or edge node to gateway node. Hence, all the packets going in that path have to be scheduled with an interval of FHD to reduce the contention between the packets, thereby achieving better spatial channel reuse. This spacing of packet transmissions is required in multihop wireless networks irrespective of higher layer protocols used when the flows are running for more than four hops.

In this work, we aim to reduce the MAC layer contention by using an adaptive pacing mechanism at the Link layer. A cross layer approach is used for scheduling packets and estimating FHD in a path. Our approach estimates the FHD in the path by measuring the queuing and *transmission delay*[1] incurred at the bottleneck node in a distributed manner. The main contributions of this paper are as follows:

- In order to reduce the contention in the network for achieving better spatial channel reuse, we propose a scheme for pacing of packets (based on their destination) at the Link layer.
- For the estimation of pacing delay, we do not seek any additional control packet exchange between the nodes. The nodes in the network path learn the congestion level at the bottleneck node in a distributed manner.

The rest of the paper is organized as follows. In Section II, we briefly discuss the existing work in the literature and provide the motivation for our work. In Section III, we describe the design and implementation of our Link layer adaptive pacing scheme. In Section IV, we demonstrate the responsiveness of our scheme to the congestion in the network. In Section V, we evaluate our Link layer pacing scheme for both UDP and TCP traffic in different network scenarios. Finally, we conclude the paper in Section VI.

*Author for correspondence.

[1]Transmission delay is the sum of channel access time and transmission time

II. Related Work and Motivation

The performance of TCP over multihop wireless network is greatly affected by packet loss due to Link layer contentions rather than loss due to congestion (buffer overflow at the intermediate nodes). To improve the performance of TCP over multihop wireless networks, several variants of TCP such TCP ELFN [3], TCP-Feedback [4], and TCP-AP [5], were proposed. These protocols try to distinguish between congestion and non-congestion losses in the network and appropriately take actions to achieve better performance. The rate based protocols such as ATP [6], estimate the available bandwidth between the source and destination, and transmit the packets at the estimated rate. There are other solutions such as Distributed Link RED (LRED) [7] and Neighborhood RED [8] that incorporate Randomly Early Detection (RED) mechanism in queues to improve the TCP performance. LRED improves the performance of TCP flows by implementing RED at the queues and reacting to continuous packet collisions by increasing the MAC backoff time by one packet transmission time.

Several researchers identified that the poor performance of TCP in IEEE 802.11 based multihop wireless networks is due to the underlying routing and MAC layer protocols [9]. Due to the broadcast nature of wireless channel, the neighboring nodes in the network can not transmit simultaneously. So the packets of the multihop flow contend with each other for the channel at successive hops, if the data arrived at the source is bursty in nature. It is well known that, TCP generates bursty traffic based on the current congestion window size. This leads to self contention, thus increasing chances for dropping the packets. To solve the above said problem, TCP-AP spreads the transmission of successive packets according to the computed transmission rate. The transmission rate is computed by the round trip time (RTT) at the TCP sender. As the spreading of packets is done at the transport layer, it fails to spread the transmission of packets at the MAC layer. The rate estimate at the TCP sender assumes the bandwidth of all links in the path between source and destination is same. This is always not true as MAC selects the transmission rate based on the quality of the channel. Further, if there are two or more TCP flows between same source and destination, the Link layer contention can not be reduced by TCP-AP.

The spacing of packet transmissions at the MAC is essential for all higher layer protocols, if the traffic is bursty in nature or the network is congested. Hence, we propose the adaptive pacing at the Link layer in IEEE 802.11 multihop wireless networks. In IEEE 802.11 multihop wireless network, by perfectly scheduling the transmission of packets (i.e., pacing packets with FHD), the self contention of packets belonging to a flow can be reduced which reduces the contention and achieves a maximum transmission rate of $R_{max} = \frac{1}{FHD}$ [2].

III. The Adaptive Pacing Scheme

The implementation of Link layer adaptive pacing scheme is two fold. One is the pacing of packets at the source node with FHD and other one is the estimation of FHD at the source node by propagating the bottleneck node's congestion information in a distributed manner. In the absence of congestion in the network, the FHD is four times the one-hop transmission delay in the path. But if the network is congested, it should be calculated as four times the queuing delay, channel access time, and transmission time at the bottleneck node.

A. Node Architecture and Distributed Scheduler

Here, we provide the implementation details of our Link layer pacing scheme by providing a node architecture and a distributed scheduler at the Link layer. It is assumed that each node, say *N* maintains a separate queue called *Input Queue* for the packets destined to the same destination node and one *Transmission Queue* which contains the packets that are ready for transmission. The scheduler moves the packets from *Input Queue* to *Transmission Queue*. Both these queues serve the packets in First-In-First-Out (FIFO) fashion. The key idea here is to introduce delay between arrival and departure of packets belonging to a particular destination whenever required, to identify the congestion of the downstream node at the source. All the incoming packets are placed into the corresponding *Input Queue* based on the destination of the packets. Each node maintains the average time that the packets belonging to particular destination spent in this node (HT^d) and in the downstream node (NHT^d). The scheduler moves the packet from *Input Queue* to the *Transmission Queue* based on the values of HT^d and NHT^d. If NHT^d is greater than HT^d for destination *d* at a node, it adds additional delay to move the packet from its *Input Queue* to the *Transmission Queue*. At the source node the delay incurred in moving the packet from *Input Queue* to *Transmission Queue* is the estimated value of FHD^d for that destination *d*. An illustration of node architecture that consists of queues and the scheduler is shown in Fig. 1.

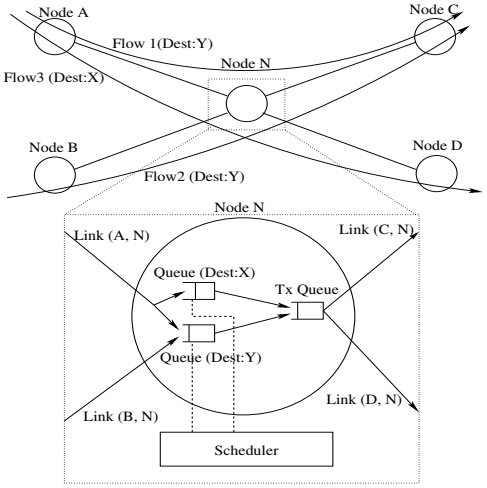

Fig. 1: An Illustration of Node Architecture.

B. Estimation of 4-hop Transmission Delay

In our adaptive pacing scheme the bottleneck node queuing and transmission delay is propagated to the source by incurring additional delay to the transmission of packets at

the intermediate nodes so that all upstream nodes adjust the queuing and transmission delay equal to that of the bottleneck node. Each node can measure NHT^d by overhearing the transmission of the downstream node except the node preceding the destination node. If the downstream node's queuing delay and transmission delay is more than the node's queuing and transmission delay, the node incurs additional delay which makes the queuing and transmission delay of this node equal to the downstream node. By doing this, it is indirectly notifying its immediate upstream node about the congestion information at the bottleneck node. This process is repeated at each upstream node and finally the source node finds the bottleneck node's queuing delay and transmission delay by the estimate of NHT^d. Once the source node gets the congestion level at the bottleneck node, it estimates the FHD for destination *d* (FHD^d).

C. Propagation of Congestion Information

The distributed scheduler discussed above propagates the congestion information in the network to the source node. The HT^d and NHT^d are estimated as follows. Whenever packets are transmitted from the node, the amount of time that the packet spent in this node is calculated by subtracting the time of arrival of the packet to this node from the time of completion of packet transmission in the MAC layer. Let us denote this by $HT^d_{current}$. HT^d is calculated using Weighted Moving Average (WMA) with weight α as shown below.

$$HT^d = HT^d_{old} \times \alpha + HT^d_{current} \times (1 - \alpha)$$

The amount of time the packet spent in the downstream node is the difference between the time at which the packet is transmitted from this node to the time at which the node overheard the transmission of the same packet from the downstream node. Let us denote this by $NHT^d_{current}$. NHT^d is calculated using WMA with weight α as shown below.

$$NHT^d = NHT^d_{old} \times \alpha + NHT^d_{current} \times (1 - \alpha)$$

The pacing delay to be incurred for each packet belonging to destination *d* is PD^d and is calculated as given below with initial value of PD^d as 0.

$$PD^d = PD^d + (NHT^d - HT^d)$$

D. Pacing at Source Node

As discussed earlier, each upstream node in the path from the bottleneck node ensures that, the amount of time the packets belonging to a particular destination spent in it is equal to the amount of time the packets spent at the bottleneck node. Now, the source node can estimate the FHD^d by multiplying the estimate of NHT^d at this node by a constant *k* depending upon the hop length of the path. If the hop length is less than four then *k* is equal to hop length of the path, and four otherwise. Hence, the pacing delay at the source is calculated as follows.

$$PD^d = k \times NHT^d$$

The scheduler moves the packets from the *Input Queue* to the *Transmission Queue* with a pacing delay (PD^d) which makes the delay between successive transmission of packets for that particular destination to be FHD^d.

E. Identification of Congestion

To validate the effectiveness of the proposed scheme, we analyze the FHD estimation at the source node in the following scenario. Two CBR flows (*flow1* and *flow2*) are generated on a chain topology of length 10 hops, *flow1* from node 0 to node 9 and *flow2* from node 6 to node 7. We assumed a channel capacity of 2 Mbps. The data rate of *flow1* is set to 200 Kbps and varied the data rate of *flow2*. We ran the simulation for 90 sec by running *flow1* throughout the simulation time and *flow2* from T1=30 sec to T2=60 sec. In the absence of *flow2* the estimated FHD at node 0 is around 0.03 secs, but when *flow2* started at T1 the FHD estimation increases significantly and reduces the transmission rate of *flow1*. When *flow2* terminates at T2, the FHD estimate again comes down to 0.03 secs and there is an increase in the transmission rate. The FHD estimation at the source for data rates of 20 Kbps and 50 Kbps for *flow2* are shown in Fig. 2 and 3, respectively. This experiment shows that the congestion in the path can be identified at the source by implementing the distributed scheduler discussed above.

IV. Parameter Tuning and Responsiveness

A. Simulation Environment

We conducted all simulations using the network simulator ns-2.29. We implemented the adaptive pacing scheme in the interface queue of ns-2. For the simulation setup, the transmission range and carrier sense range are set to 250 meters and 550 meters, respectively. We enabled RTS/CTS handshake for MAC transmission. The channel capacity is set to 2 Mbps unless otherwise stated. The packet size for TCP and UDP traffic are set to 1000 bytes. We assume the *Input Queue* and *Transmission Queue* size to be 25 for all nodes. Unless otherwise stated, in all considered topologies, each node is 200 meters apart of each of its adjacent nodes. AODV [10] is used as a routing protocol.

In all experiments we ran the simulation for 500 secs and ignore the behavior for first 50 secs to neglect the initial transient. All the results presented in this paper are averaged over 30 simulation runs with different seed.

B. Parameter Tuning

The value of α used in WMA smoothens the estimates of NHT^d and HT^d with changes in the network conditions. To find the best value of α, we conduct an experiment with two competing TCP NewReno flows in the cross topology shown in Fig. 4. The measured aggregate goodput and Jain's fairness index between the two flows are shown in Fig. 5. One can observe from the figure that, for the value of α = 0.8, it gives better aggregate goodput and fairness. We choose the value of α as 0.8 for the remaining simulation experiments.

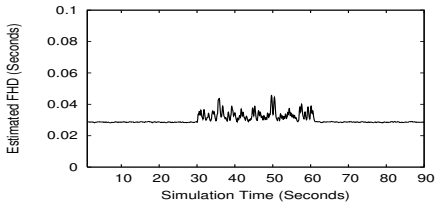

Fig. 2: FHD Estimation in Chain Topology while *flow2* = 20 Kbps.

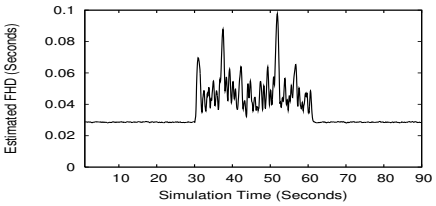

Fig. 3: FHD Estimation in Chain Topology while *flow2* = 50 Kbps.

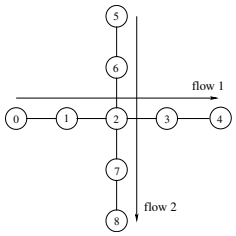

Fig. 4: Cross Topology with Two TCP NewReno Flows.

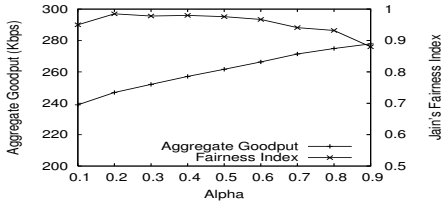

Fig. 5: Aggregate Goodput and Fairness Index for Varying α.

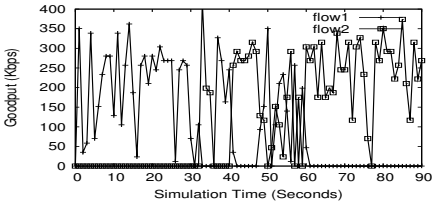

Fig. 6: Responsiveness of NewReno without Link Layer Pacing.

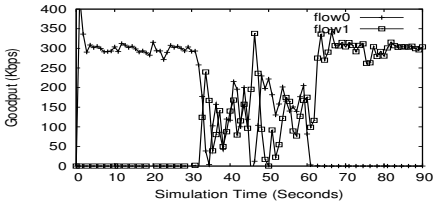

Fig. 7: Responsiveness of NewReno with Link Layer Pacing.

C. Responsiveness

The responsiveness denotes how quickly our scheme adapts to the changes in network congestion. To illustrate the transient behavior of our proposed scheme, we measure the goodput of two TCP flows in the same cross topology shown in Fig. 4. Here the two TCP NewReno flows compete for the channel access. For this experiment, *flow1* runs from beginning of the simulation to T2=60 secs and *flow2* runs from T1=30 secs to end of simulation T3=90 secs. Fig. 6 and Fig. 7 plot the goodput vs simulation time for both flows with and without pacing at Link layer. With pacing at Link layer both TCP NewReno flows utilize the available bandwidth when there is no competing flow, and share the bandwidth fairly between them when they compete for the bandwidth of the channel. But without pacing at Link layer, *flow1* captures the channel and utilizes almost entire bandwidth even after *flow2* starts. The other observation here is that, when congestion occurs due to the competing flows, the propagation of congestion information to the source makes the flows to respond quickly.

V. Simulation Results

In this section, we study the performance of Link layer pacing in different network scenarios for both UDP and TCP traffic.

A. Chain Topology

We measured the performance of both UDP traffic and TCP traffic over the chain topology of hop length 10. For UDP traffic we considered a CBR flow of different data rate and measured the throughput at the receiver. We measured the achieved throughput with and without Link layer pacing by varying the data rate of CBR traffic from 20 Kbps to 400 Kbps. The increase in data rate at the source increases the throughput at the receiver linearly as long as the time between packet arrival at the source node is less than FHD in the path for both the cases. Without Link layer pacing, further increase in data rate of the source decreases the throughput at the receiver as it increases the contention in the path. But in the case of Link layer pacing, though the packet inter arrival time decreases below FHD, the source node pushes the packets into the network with interval of FHD. So the throughput at the receiver remains constant and does not reduce with increase in data rate of the source. This is shown in Fig. 8.

For TCP traffic, we considered two different experiments, (1) Measurement of goodput of one TCP NewReno flow by varying the hop length in a chain topology and (2) Measurement of aggregate goodput and fairness index of TCP NewReno flows by varying the number of flows between the end nodes of the chain.

In the first experiment by changing the length of the chain topology we ran an FTP flow using TCP NewReno connection between end nodes of the chain and measured the goodput, with and without Link layer pacing. For comparative purpose we measured the performance of TCP-AP without Link layer pacing also. The results are shown in Fig. 9. The goodput of TCP NewReno over IEEE 802.11 MAC without Link layer pacing reduces drastically when the number of hops increases from 4 to 7. This is due to the fact that, the number of hidden terminal collisions are more near the source node. But the Link layer pacing reduces the contention in the path and improves the goodput of TCP NewReno. We noted that TCP NewReno with Link layer pacing achieves goodput as good as that of TCP-AP.

In the second experiment, over a chain topology of fixed length (we took hop length of 10), we measured the aggregate goodput of all flows by increasing the number of TCP NewReno flows between the end nodes in the chain. As discussed in the introduction section, spreading the transmission of packets at the transport layer may not spread the packet transmissions in the MAC layer. Hence, TCP-AP does not perform well with multiple flows. But TCP NewReno over Link layer pacing achieves maximum aggregate goodput. It is shown in Fig. 10. We analyzed the fairness between these

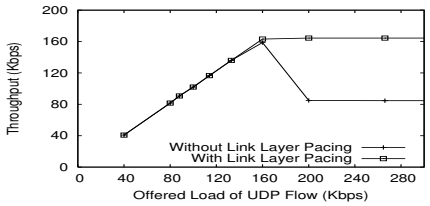

Fig. 8: Throughput vs Offered Load with and without Link Layer Pacing.

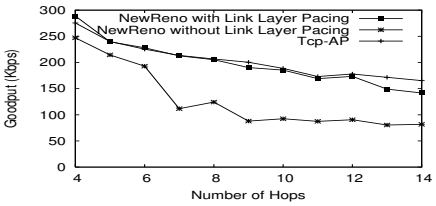

Fig. 9: TCP Goodput vs Wireless Hop Length.

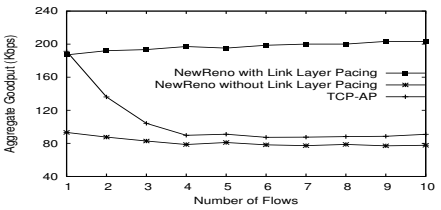

Fig. 10: Aggregate Goodput of TCP NewReno Flows.

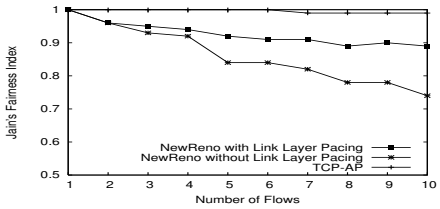

Fig. 11: Jain's Fairness Index of TCP NewReno Flows.

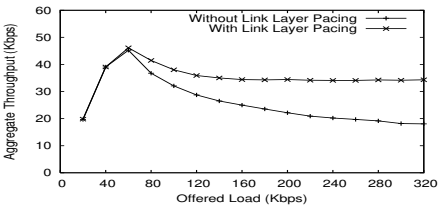

Fig. 12: Aggregate Throughput vs Offered Load.

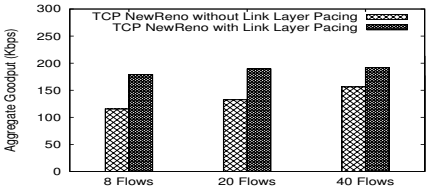

Fig. 13: Aggregate Goodput for Varying Number of Flows.

flows also. We found that, fairness between the flows increases while running over Link layer pacing. But when compared to TCP-AP there is a reduction in fairness between the flows. But this reduction in fairness compensates the huge increase in aggregate goodput when compared with NewReno over Link layer pacing. The measured fairness using Jain's fairness index is shown in Fig. 11.

B. Grid Topology

We measured the performance of both UDP and TCP traffic over the grid topology of size 10×10 with inter node distance of 200 meters. For UDP traffic we considered 20 CBR flows and measured the aggregate throughput at the receiver for different data rates. We took packet size of 50 bytes as typical voice applications generate small size packets. We setup flows from one edge node to other edge node in the opposite side. This is to make sure that hop length of all flows are more than four. We randomly picked five sources in each side of the grid and randomly picked destination in the opposite side of the grid. We ran the simulation for 30 such different flow patterns and took the average of aggregate throughput. We ran experiments with and without Link layer pacing by varying the data rate of CBR flows from 1 Kbps to 16 Kbps. Fig. 12 shows the aggregate throughput for different offered loads. Without Link layer pacing, as the data rate increases above 3 Kbps, the overall throughput at the receiver decreases drastically. But with Link layer pacing the decrease in throughput is less. Thus, the overall network throughput increases with Link layer pacing.

For TCP traffic, we measured the aggregate goodput of the flows by varying the number flows as 8, 20, and 40. The flows are randomly generated as discussed in UDP experiment and aggregate goodput is measured for 30 different flow patterns. TCP NewReno gives better aggregate goodput when running over Link layer pacing compared to running without Link layer pacing. It is shown in Fig. 13.

VI. Conclusion

In this paper, we proposed a new Link layer pacing scheme which adaptively estimates the FHD on the path and transmits the packets with estimated FHD interval. This scheme also performs the congestion control by propagating the congestion information to the source without any additional control overhead. As our proposal does the pacing at the Link layer, it improves the performance of the multihop mesh networks irrespective of higher layer protocols. Through ns-2 simulation we showed an improvement in the performance of both UDP and TCP traffic with Link layer pacing implemented with IEEE 802.11 MAC in different network scenarios.

References

[1] I. F. Akyildiz, X. Wang, and W. Wang, "Wireless Mesh Networks: A Survey," *Computer Networks*, vol. 47, no. 4, pp. 445–487, March 2005.

[2] K. Chen, Y. Xue, S. H. Shah, and K. Nahrstedt, "Understanding Bandwidth-Delay Product in Mobile Ad Hoc Networks," *Computer Communications Journal*, vol. 27, no. 10, pp. 923–934, June 2004.

[3] G. Hallond and N. Vaidya, "Analysis of TCP Performance over Mobile Ad Hoc Networks," *Wireless Networks*, vol. 8, no. 2/3, pp. 275–288, March 2002.

[4] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash, "A Feedback Based Scheme for Improving TCP Performance in Ad Hoc Wireless Networks," *IEEE Personal Communications Magazine*, vol. 8, no. 1, pp. 34–39, February 2001.

[5] S. M. ElRakabawy, A. Klemm, and C. Lindemann, "TCP with Adaptive Pacing for Multihop Wireless Networks," in *Proceedings of ACM MOBIHOC*, May 2005, pp. 288–299.

[6] K. Sundaresan, V. Anantharaman, H.-Y. Hsieh, and R. Sivakumar, "ATP: A Reliable Transport Protocol for Ad-hoc Networks," in *Proceedings of ACM MOBIHOC*, June 2003, pp. 64–75.

[7] S. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla, "The Impact of Multihop Wireless Channel on TCP Throughput and Loss," in *Proceedings of IEEE INFOCOM*, March 2003, pp. 1744–1753.

[8] K. Xu, M. Gerla, L. Qi, and Y. Shu, "Enhancing TCP Fairness in Ad Hoc Wireless Networks Using Neighborhood RED," in *Proceedings of ACM MOBICOM*, September 2003, pp. 16–28.

[9] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A Comparison of Mechanisms for Improving TCP Performance Over Wireless Links," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 756–769, 1997.

[10] C. Perkins, E. Royar, and S. Das, "Ad Hoc On-Demand Distance Vector (AODV) Routing," *IETF RFC 3561*, 2003.