PERFORMANCE OF TCP OVER IEEE 802.11 BASED AD HOC NETWORKS

Matulya Bansal and Gautam Barua
Department of Computer Science and Engineering
Indian Institute of Technology, Guwahati - 781039
{gb@iitg.ernet.in}

Abstract. In this paper, we study the performance of TCP over the IEEE 802.11 MAC layer. We identify packet-size, hop-length and network load as the main factors that affect TCP performance. Based on our observations, we propose a sender-based heuristic that enables the TCP sender to identify route-failures, thus improving TCP performance. We also analyse the effectiveness of two of the most popular routing protocols, AODV and DSR, in supporting TCP traffic. In addition to helping understand the interactions between TCP and IEEE 802.11, our work makes clear the need for routing protocols to consider network load as an essential criterion in establishing routes.

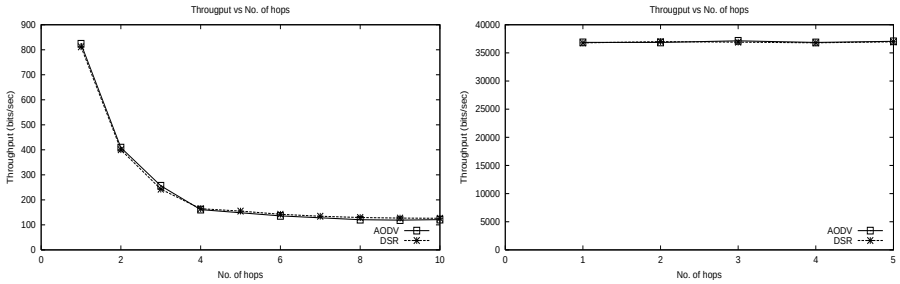
Keywords: Mobile Ad Hoc Networks, TCP, IEEE 802.11, Routing.

Introduction

The TCP protocol has been tuned extensively to give good performance for traditional wired networks. However, TCP in its present form, does not perform as well in dynamic environments such as mobile ad hoc networks where frequent and unpredictable route failures may lead to unnecessary invocation of congestion control by the TCP Sender causing TCP performance to degrade to unacceptable levels. Furthermore, most of the routing protocols that have been proposed for MANET, do not take into consideration the interactions between TCP and the MAC layer. Consequently, the performance of the TCP protocol further deteriorates. In this article, we study the interactions between TCP and the IEEE 802.11 MAC layer. Based on our observations, we identify the factors that affect the performance of TCP in ad hoc networks. We also evaluate the performance of TCP on two of the most popular routing protocols, AODV and DSR and suggest issues to be considered while designing routing protocols. In addition, we propose a sender-based heuristic that enables the TCP Sender to distinguish route-failures from congestion in the network, thereby improving TCP performance.

Related Work

Several recent studies have addressed the problems faced by TCP in ad hoc networks. Gerla et.al. [2] study the interactions between TCP and MAC layer in a multihop wireless environment using the CSMA and FAMA MAC layer protocols. Dyer and Bopanna [1] evaluate the performance of the three routing protocols, AODV, DSR and ADV with respect to TCP traffic. Their main focus was on observing the effect of routing protocol on TCP performance. Chandran et.al. propose a feedback based scheme [8] to improve TCP performance in ad hoc networks frequently plagued by route-failures. Liu and Singh [9] study the affect of transmission errors, congestion losses and route-failures on the performance of TCP. They make use of feedback from the network layer to identify route-failures and ECN to detect congestion. Holland and Vaidya [3] have studied the performance of TCP over IEEE 802.11 networks. Their work is similar to this, but their main focus was on the performance due to mobility aspects. This paper covers many other issues.



(a) (b)

FIGURE 1. Throughput vs Number of Hops (TCP, CBR)

TCP OVER IEEE 802.11

To understand the interactions between the Transport control Protocol and the IEEE 802.11 Mac Layer, we divided our experiments into two parts: fixed topology experiments and random topology experiments. The ns-2 network simulator [6] with extensions by CMU[7] was used for all the simulations.

Fixed topology experiment. The purpose of this experiment was to study the interactions between TCP and the IEEE 802.11 MAC layer. Simple static network topologies were set up and different parameters were varied to observe their effect on the performance of the TCP protocol. In particular, we were interested in knowing how TCP performance over IEEE 802.11 was affected by TCP segment and window size, the number of hops the TCP connection has to traverse and the number of connections sharing a channel.

Using segment sizes of 512 and 1460 bytes, the most commonly used segment sizes on the Internet [10] and maximum window sizes of 4, 32 and 64 segments, we found that TCP throughput increases with increasing segment size. However, throughput is hardly affected by window size. On the other hand, larger window sizes lead to larger queues at the intermediate nodes.

To understand the effect of hop-count on TCP performance, we set up a linear chain of nodes and performed simulations for different path lengths. In these experiments, we observed that TCP throughput initially decreases rapidly as the hop count of the connection increases but then stabilizes after three-four hops. After this the decrease in throughput with hop-count is gradual.

By setting up several simultaneous connections between end-nodes sharing a common node, we sought to understand the effect of channel sharing on TCP performance. We found that the total throughput of all the connections combined together was almost constant while the throughput of the individual connections decreased as the number of connections was increased.

In order to measure the fairness of TCP over IEEE 802.11 networks, we used the 'fair share per link metric' i.e. if there are n flows through a link, then each share has $1/\mathrm{n}^{th}$ of the capacity of the bottleneck link. For n flows, with flow_i receiving a fraction b_i of the bottleneck link's bandwidth, the fairness of the allocation is defined as $\frac{(\sum_{i=1}^{n} b_i)^2}{n*(\sum_{i=1}^{n} b_i^2)}$. The metric ranges from 1/n to 1, with 1 corresponding to equal allocation to all users. Topologies, similar to those in [11] were used to measure TCP fairness. In these experiments, we observe that TCP over IEEE 802.11 is heavily biased in favour of connections with a short hop-count. While connections with equal path lengths are pretty fair, fairness decreases as the hop-count of the longer connection increases until finally the connection with longer path is unable to transmit any data.

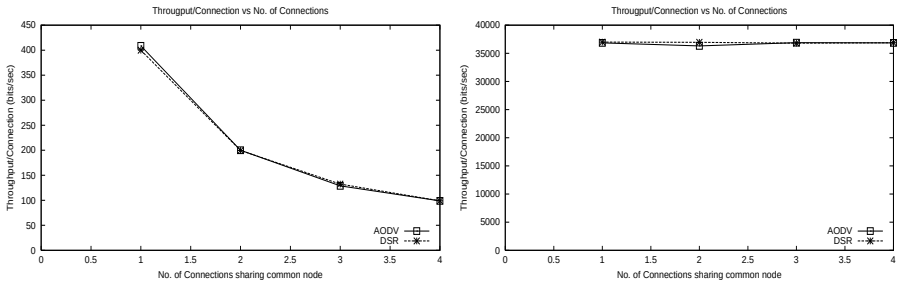


FIGURE 2. Throughput vs Number of Connections sharing a Node (TCP, CBR)

Table 1 : TCP Fairness over IEEE 802.11

Experiment	No of connections	Hop Counts	No of Pkts sent by each connection	Fairness
1	2	1, 2	50681, 48974	.99971
2	2	1, 3	84812, 9351	.60893
3	2	1, 4	100029, 5	.50004
4	2	1, 5	100057, 2	.50001
5	4	2, 3, 4, 5	99717, 84, 2, 0	.25043

Analysis. In the above experiments, we observed that the performance of TCP over IEEE 802.11 depends on several factors. Packet size, number of hops the connection has to traverse and the number of connections sharing the channel are the most important factors.

It is interesting to compare these results with CBR performance under similar conditions. When a CBR data transfer over UDP is performed over a chain with increasing hop count, the throughput of the connection does not decrease, rather it remains the same. When the number of connections sharing the channel (path in our case) is increased, the total throughput increases proportionally to the number of connections, throughput of a connection remaining the same. Moreover, even when two CBR connections are started between the same pair of nodes in opposite directions, the throughput per connection remains the same.

This is because of the fact that IEEE 802.11 MAC Layer uses Carrier Sense for Collision avoidance along with a CTS-RTS exchange scheme. This implies that when a node is transferring data, all its neighbours have to wait for their turn to transmit data. Though this is true for both CBR as well as TCP traffic but it is only TCP that shows a degradation in performance. TCP is a feedback-based protocol and since both the reverse and the forward paths are the same, a TCP acknowledgment has to travel through the same path back. This leads to channel contention between TCP packets and acknowledgements. All this while, the TCP sender has to wait for the acknowledgment to arrive. These delays get larger with increasing hop counts and this affects its performance. However, when hop counts becomes large i.e. four or five, a degree of parallelism is introduced into the network, and TCP throughput stabilizes. UDP Traffic on the other hand requires no such acknowledgments. So, even when two connections are started in the opposite directions, contention for the channel still occurs, but the application/transport layer entity is not affected. It continues to send packets at a uniform rate which may get queued up but are sent as soon as the channel becomes available.

Larger packet sizes increase TCP throughput because of the lower routing and MAC layer overhead experienced in sending the same amount of data through the network. Window size has little affect on TCP throughput. On the contrary, using larger windows leads to queueing up of packets at the intermediate nodes. This is because the throughput of the channel being constant, extra packets transmitted by the TCP sender are unable to be delivered to the receiver and are instead queued up in the network waiting for their turn.

IEEE 802.11 biases TCP in favour of connections with shorter hop-counts. TCP itself favours connections with short round-trip times and the channel contention and capture effect of IEEE 802.11 further exacerbate this problem. Fairness decreases with increasing hop count difference between the competing connections, until one connection is finally unable to transmit any data.

Random topology experiment. The purpose of this experiment was to study the performance of TCP under dynamic network conditions. In particular, we were interested in knowing as to how TCP performance is affected by route failures. We also wanted to evaluate the effectiveness of AODV and DSR, the two prominent routing protocols for ad hoc networks in providing network layer service TCP. The simulation set up is similar to that used in [4].

The route-failure detection heuristic

This sender-side heuristic enables the TCP sender to identify route-failures. When retransmission time-outs occur successively and no acknowledgment is received in the meanwhile, the TCP sender assumes that a route failure has occurred. It then goes into a probe state wherein it freezes the value of all its state variables and periodically sends out probe packets for detecting connectivity. When a route is re-established (an acknowledgement is received), it unfreezes its state and continues TCP's normal operation.

This mechanism of detecting route-failures is similar to the one used in [1]. However, there are certain differences. Instead of using the retransmission timeout value for the second timeout directly, we use a *moderated* retransmission timeout value. Basically, we check to see if the timeout value lies between a lower threshold α and an upper threshold β. If it does, the retransmission timeout value is chosen for the second timeout. However, if it is less than α, we choose α as the timeout value. Similarly, if it is greater than β, we choose β to be the timeout value. This is important because at low bandwidths, the timeout values are large, and using them undermines TCP's ability to identify route re-establishments. Similarly, at high bandwidths, the timeout values are low, and aggressively transmitting packets when the route is down degrades overall network performance. Secondly, instead of assuming route failure when the missing acknowledgment does not arrive even after the second timeout, the arrival of no acknowledgment is taken to be an indication of a route failure. Again, under unfavourable conditions as exist in such environments, it is quite possible that the same packet may be lost due to transmission errors, congestion etc. Under such conditions, though that packet may have been lost, the arrival of duplicate acknowledgments clearly indicates the existence of a path.

Traffic and Mobility Models

The traffic and mobility models used are similar to the ones as previously used in [4, 5]. Two sets of experiments were performed : the first set had only TCP sources (1, 2, 10) while the second set had both CBR (10, 20, 40) as well as TCP(1, 2, 10) sources. TCP and CBR packet size was 512 bytes. TCP congestion window was set to 16KB.

Performance Metrics

In addition to the standard metrics like *Packet Delivery Fraction, Throughput, Goodput, End-to-end delay of data packets* and *Normalized Routing Load* we also measured *Average Queue Length* (calculated by monitoring the queue length at each node whenever a packet was enqueued), *Average Hop Length* (the average weighted length of paths from the source[s] to the destination[s] where weights were the lifetime of a path with a particular hop-count) and *Average maximum no. of connections sharing a node in the path* (the average weighted maximum number of connections that shared the node in the path from the source[s] to the destination[s] with this connection).

Packet Delivery Fraction, *Goodput*, *Throughput* and *End-to-end Delay* are important for best-effort traffic. A Low *Routing Overhead* is important for scalability. *Average Queue Length* is important where buffer space is a limitation. *Average Path Length,* as observed earlier, is important for TCP traffic. *Average Maximum Node* and *Link Overlap* reflect load awareness and as seen, are important for TCP Traffic.

Table 2 : Simulation Results for Experiment with 100 nodes, a single TCP connections and 10 CBR Connections

Metrics/Pausetimes	0	125	250	375	500
TCP Throughput (AODV)	60.88	209.54	128.79	227.85	91.74
(AODV with heuristic)	81.73	217.71	129.21	233.82	93.33
(DSR)	20.02	98.392	81.24	213.00	86.77
(DSR with heuristic)	25.02	107.98	87.04	223.63	88.50
Path Length (AODV)	7.89	3.70	6.65	4.43	7.16
(DSR)	4.09	3.19	3.98	3.64	6.66
Node Overlap (AODV)	3.72	3.20	3.42	2.97	3.73
(DSR)	2.60	2.29	2.29	2.48	3.92
Routing Overhead (AODV)	.068	.085	.065	.076	.054
(DSR)	.019	.043	.048	.063	.051
Queue Length (AODV)	1.61	5.80	3.07	5.30	2.18
(DSR)	10.09	9.97	6.92	6.44	3.08
TCP Delay (AODV)	.241	.209	.257	.190	.415
(DSR)	.259	.245	.259	.182	.462

Results and Discussion. From these experiments (Table 1 shows the case for 100 nodes, 1 TCP and 10 CBR sources; results obtained from other experiments are similar and are not presented here due to space constraints), we observe that under conditions of high mobility, AODV performs better than DSR. However, DSR's performance increases with increased pause-times and it outperforms AODV under conditions of low mobility. Moreover, the performace of DSR increases wih the connection density of the network. The route-failure detection heuristic at the TCP sender results in improved TCP throughput, both for AODV and DSR.

At small pause times, the paths chosen by AODV are invariably longer and have a larger degree of node and link overlap. This seems logical as DSR always uses the shortest discovered path while AODV uses the first path which is discovered. However, as pause times increase, the cache hit ratio increases and DSR also chooses longer paths. An important consequence of DSR having larger paths at low mobility is its inablity to outperform AODV because, as seen, TCP throughput varies inversely as the pathlength. That AODV chooses paths which are longer, however, is a bit surprising. AODV chooses the first route reply and it should have also implied that AODV chooses the less loaded paths. However, AODV chooses the paths that are more loaded. This happens because firstly, routing packets are given a priority over data packets, largely nullifying the effect of network load upon choice of paths and secondly because replies from intermediate nodes already using the route may reach the source first resulting in increased route sharing.

Conclusion

In this article, we studied the performance of TCP over 802.11 based ad hoc networks. We identified path length, number of connections sharing the channel and segment size as the main factors affecting TCP. We evaluated TCP performance over AODV and DSR, the two prominent routing protocols for ad hoc networks and found that neither, like most routing protocols for ad hoc networks, considers network load as a criterion in selecting routes. However, as TCP performance is significantly affected by the offered network load, we feel that in order to deisgn practically deployable routing protocols, network load must also be taken into account while choosing routes.

We also proposed a sender based heuristic that enables the TCP sender to differentiate route-failures from congestion and therfore improve TCP performance, particularly when the connection density s low and mobility high. We believe that the use of heuristics or feedback from the lower layers to detect route-failures is important and justified for improving TCP performance in these dynamic environments plagued by frequent route-failures.

References

[1] T. D. Dyer and R. V. Bopanna, "A Comparison of TCP Performance over Three Routing Protocols for Mobile Ad hoc Networks", *Proc. ACM MOBIHOC '01*, Oct. 2001.

[2] M. Gerla et. al, "TCP over Wireless Multi-hop Protocols: Simulation and Experiments", *Proc. IEEE ICC '99*, Jun. 1999.

[3] G.Holland and N.Vaidya, "*Analysis of TCP Performance over Mobile Ad hoc Networks*" Proceedings of the 5th International Conference on Mobile Computing and Networking (MobiCom 99), August 1999, pp. 219-230.

[4] S. R. Das et. al, "Performance Comparison of Two On-demand Routing Protocols for Ad hoc Networks", *IEEE Personal Communications*, Feb. 2001, pp 16-28.

[5] J. Broch et.al., " A Performance Comparison of Multi-hop Wireless Ad hoc Network Routing Protocols", *Proc. IEEE/ACM Mobicom '98*, Oct. 1998, pp. 85-97.

[6] K. Fall and K. Varadhan, "NS notes and documentation", *The VINT Project, UC Berkeley, LBL, USC/ISI and Xerox PARC*, Available from http://www.isi.edu/nsnam/ns/ns-documentation.html, Apr. 2002.

[7] CMU Monarch Group, "CMU Monarch extensions to NS-2 simulator", Available from http://www.monarch.cs.cmu.edu/cmu-ns.html, Apr. 2002.

[8] K. Chandran et.al., "A feedback based scheme for improving TCP performance in ad-hoc wireless networks", *Proc. International Conference on Distributed Computing Systems*, 1998.

[9] J. Liu and S. Singh, "ATCP : TCP for Mobile Ad hoc Networks", In Proceedings of Journal on Selected Areas in Communications, July 2001.

[10] H. Abrahamson, "Traffic Measurement and Analysis", *SICS Technical Report T99/05*, Sep. 1999.

[11] R. H. Katz et. al, "On Improving the fairness of TCP Congestion Avoidance", I*n Proceedings of IEEE GLOBECOM 1998,* Nov 1998.