Improving the performance of TCP in the presence of interacting UDP flows in ad hoc networks

Vikram Gupta, Srikanth V. Krishnamurthy and Michalis Faloutsos

Department of Computer Science and Engineering,
University of California, Riverside,
Riverside, CA 92521, USA
{vgupta, krish, michalis} @ cs.ucr.edu

Abstract. In this paper, we study how fairness affects the performance of TCP over ad hoc networks with IEEE 802.11 at the MAC layer. The problem addressed is that the throughput of TCP flows degrades severely in the presence of heavily loaded UDP flows. Our contribution is twofold. First, we identify the factors affecting the TCP throughput by providing a micro-analysis of the performance at a level of detail that is not seen in previous studies. The intuition obtained from the first part leads us to our second contribution. We propose and study the use of per flow fairness through a mechanism we call **backpressure**. Backpressure improves the performance of TCP flows in the presence of heavy UDP flows. In fact, in some cases, this increased TCP throughput does not affect the throughput of UDP flows. We find that backpressure can increase the TCP throughput by as much as 95%. An advantage of backpressure is that it does not require any changes to the existing TCP or IEEE 802.11 protocols.

1 Introduction

In this paper, we study the effect of fairness on the throughput of TCP flows in the presence of UDP flows in ad hoc networks. The absence of a congestion control mechanism in UDP makes such flows monopolize the available bandwidth and drive TCP flows to starvation. In the Internet, this problem is partially addressed by intelligent buffer management policies such as the Random Early Drop (RED) and priority dropping of UDP packets [1, 2]. In ad hoc networks, the above schemes cannot be applied since congestion is manifested as channel access delays instead of queue build ups. In ad hoc networks, channel access delays at a node are dependent not only on its queue but also on that of its neighbors and possibly nodes further away. In this paper, we demonstrate the adverse effects of UDP flows on TCP performance and propose simple, yet effective mechanisms to overcome these effects.

It is well known that TCP performs poorly over IEEE 802.11 in multi-hop wireless networks [3]. The fundamental cause for this is the inability of the IEEE 802.11 MAC protocol to provide short-term fairness in medium access to certain nodes. In [4, 5], it has been shown that such short-term unfairness can lead to long-term unfairness with respect to goodput achieved by competing TCP flows. In [6], it has been shown that

in the presence of UDP based flows the goodput of TCP flows is significantly reduced.

Previous efforts to improve TCP performance in ad hoc networks follow different approaches from the one presented here. Research falls into two categories: a) modifying TCP, or b) replacing the IEEE 802.11 MAC protocol. Most of the TCP-based proposals attempt to make TCP distinguish between packet losses due to mobility and congestion. To this effect, some researchers propose explicit link failure notifications (ELFN) [8] [7]. Significant work has been done at developing novel MAC layer schemes. In [9], Yu et al propose a hybrid scheme where senders as well as receivers are allowed to initiate MAC transmissions. Their results show that in certain scenarios significant improvements in fairness can be achieved without sacrificing throughput. Protocols other than the IEEE 802.11 MAC scheme have also been proposed [10, 11]. In [5] it is shown that intelligently tuning the parameters used in TCP and the IEEE 802.11 protocols improves performance. While these proposals may provide efficient solutions for improving TCP performance over the IEEE 802.11 MAC protocol, it is difficult to change the existing standards for TCP or IEEE 802.11 Distributed Co-ordination Function. Moreover, to the best of our knowledge, none of these solutions is specifically targeted at improving TCP performance in the presence of UDP flows.

In this work, we study: a) how MAC-layer unfairness degrades TCP performance, and b) how fairness could improve TCP performance in the presence of UDP flows. In more detail, we provide an elaborate microscopic study by creating specific scenarios that elucidate the effects of UDP flows on TCP traffic and show that providing fairness alleviates these effects to a large extent. We then propose **backpressure**, a scheme that significantly improves TCP performance in the presence of UDP flows. In a nutshell, our scheme provides burst regulation at the flow level at each node. A forwarding node refuses to accumulate large number of packets from a flow. Once the buffering quota is reached, the node will not accept more packets from that flow before it can forward some of the buffered packets. We show that our scheme can provide a coarse control over bandwidth allocation to TCP and UDP streams. Finally, an advantage of our scheme is that it can be implemented on top of the IEEE 802.11 MAC and does not require any changes to TCP.

The paper is organized as follows. In section 2, we build the background and summarize previous work relevant to the problem. In section 3, we explain the simulation set-up and present results demonstrating the adverse affect of UDP flows on TCP traffic. We also show that medium access control needs to be supported by fair queuing mechanisms, possibly at the network layer to alleviate the problem. In section 4, we present and evaluate backpressure. Finally, we conclude in section 5.

2 Background and Previous Work

In this section, we look into the poor performance of TCP over IEEE 802.11 multi-hop wireless networks. In particular, we first examine the limitations of the IEEE 802.11 MAC protocol when used in a multi-hop wireless environment. We then explain the implications on the performance of higher layer protocols. Due to space constraints, we have not covered certain well-known phenomenon in depth. We refer the reader to [12, 3] for more details on the section below.

One of the configurable parameters of the IEEE 802.11 DCF is the number of attempts made by a node to transmit a particular frame (retry limit). A node, wishing to transmit a frame to its neighbor, makes repeated attempts governed by the above parameter.[1] Upon failure to transmit, the link is assumed to be broken, i.e., the neighbor is assumed to have moved away. The duration between successive attempts to transmit a packet grows exponentially in accordance to the binary exponential back-off [13] algorithm. This algorithm is known to favor the last successful node [13]. Thus, once a node is successful in medium access contention, it is likely to send a large number of packets from its queue before losing the rights for medium access to another node. Nodes that are interfered by this transmission can falsely presume the links to be broken (termed as false link failures, see [3] for details). In general, with the default setting of parameters, frame sizes smaller than 1500 bytes can be transmitted without causing excessive false link failures.[2] However, in general, due to the capture effect [3, 5], a large number of failures occur. In particular, UDP flows are able to starve TCP flows in this scenario [5].

Solutions proposed for the above phenomenon propose changes to either TCP or the IEEE 802.11 DCF. In [5], Jiang et al show that increasing the retry limits leads to an increase in goodput of TCP connections. However, with this solution, it takes a long time to detect real link failures (typically due to mobility). In [14] the authors propose many changes to the IEEE 802.11 MAC such as a less aggressive back-off mechanism and an additional message to prevent false link failures. The false link failures described are known to lead to poor performance of TCP [3]. We have used the Ad hoc On demand Distance Vector (AODV) [15] as the routing protocol in our simulation studies. AODV provides an alternative to mechanism to determine link connectivity by periodically broadcasting Hello messages [16]. As the use of Hello messages has been shown to reduce false link failures, we use these messages in our simulations.[3]

We have utilized a simple fair-queuing scheme in developing our solution. To appreciate the novelty of our work in the presence of other literature on fair-queuing [22 23], consider that any fair-queuing scheme requires 3 policies [20] for choosing:

1) Which packet (queue) will be transmitted? For this, we use simple round-robin.

2) When is the packet transmitted?

3) Which packet(s) will be dropped in case of congestion?

The fundamental innovation of this work lies in combining the last 2 policies to control belligerent flows. Specifically, by restricting the queue size for a flow at each node in the network we are able to prevent a monopoly by belligerent flows. It should be noted that similar concepts have been proposed in the past, albeit for improving TCP congestion control [24] rather than for stemming UDP flows.

In summary, a UDP based flow can build up large queues on nodes on its route. This queue build-up, and the subsequent medium capture [5], creates congestion in the neighborhood of the path taken by the TCP flow. In such a scenario, the TCP flows that intersect[4] with the heavy UDP flow suffer delays and drops. TCP interprets this loss to be a mark of congestion and acts accordingly [17]. In this work, we first

---

[1] Depending on the frame-length, a transmitting node uses one of two different parameters [3].

[2] With default ns2 settings of 802.11, average time spent in repeated back-offs is more than the transmission time for a 1500 byte frame.

[3] This helps us in concentrating on the loss in goodput of TCP in the presence of UDP flows.

[4] Share medium or buffer

show the extent to which fair-queuing improves TCP performance in such a scenario. Later, we improve upon the gains achieved by fair-queuing through back-pressure.

3 Tuning of the System Parameters and Simulation Scenarios

In this section, we describe the simulation scenarios and discuss the results. Although the conclusions are drawn from averages computed over many simulation runs, we perform a microscopic study to draw meaningful conclusions; towards this objective we set up specific scenarios and focus on individual traces.

Table 1. Parameters varied in simulations

Protocol	Parameter/Mode	Range Observed	Optimal Settings
TCP (FTP)	Max. segment size	200-1460 bytes	1460 bytes
UDP(CBR)	Data generation rate	50-800 KB/sec	800 KB/sec
	Packet size	200-2920 bytes	1460 bytes
	Num of application	1-2	2
	Time of start	+/-20 sec w.r.t. TCP start time	12 seconds after TCP
	Flow hop length	2-10	8
AODV	Local link maintenance mode	Link layer feedback or Hello Messages	Hello Messages
	Messages interval	0-	9 Seconds
	Route repair wait duration	0-	6 Seconds
802.11	Retry limits	(7,4) to (21,12)	21,12

3.1 Simulation Set-Up

We have used NS2 [18] for our simulations. A random topology or the incorporation of mobile nodes makes a microscopic performance analysis extremely difficult if not impossible. Thus, we test the various scenarios in a 13 x 13 static grid topology. Each node is separated from its neighbor by 200 meters. The transmission range of each node is fixed at 250m. A representative example of the topological structure of the network is shown in figure 1a. In this grid, the clients are placed at the corner nodes (0, 12, 156 and 168, refer figure 1a) and mid-way nodes (6, 78, 91 and 162, refer figure 1a) along the edges. The eight connections are labeled as Connection 1 to Connection 8, respectively. An FTP server is hosted at the node at the center of the grid (denoted by S). We use FTP application clients in NS2 to establish TCP based connections with the FTP server. Each client sends packets of fixed size to the server once a connection is established. The metric for performance is the goodput achieved by the 8 TCP clients. To simulate UDP flows, we place application agents that generate Constant Bit Rate (CBR) traffic at certain nodes (refer figure 1a). The traffic generated by these applications is transported through UDP. Further, the shortest hop

path between the CBR source and destination passes through the FTP server. We simulated various CBR rates, but only a few have been reported due to space constraints. All simulations were run for fixed durations of 150 seconds.

3.2 Model Used

Due to a wide range of values for parameters in various protocols we faced the difficult task of simulating many possible scenarios. First, we observed that the performance of TCP in the presence of UDP flows is poor with the default settings.[5] We then observed the TCP performance variation with different parameter values. Table 1 lists these parameters. Our strategy was to choose the parameters such that TCP goodput in the *absence* of UDP flows (initial goodput) is high.[6] This can be observed in our choice of values (see column 4 of table 1). It should be noted that using protocols like DSR [19] did not qualitatively affect the results reported in this paper.

3.3 TCP Performance in Presence of UDP Flows

Stabilized Routing. The performance of TCP connections in the presence of UDP flows was found to be poor. Specifically, we observed that in presence of UDP streams, the aggregate goodput of the TCP clients was reduced to 10% of the aggregate goodput achieved in the absence of the UDP flows (initial conditions). The actual throughput achieved is dependent on the routes taken by the UDP streams. To explain this, we analyze two different cases.
Case 1. UDP streams encompass the TCP server. The scenario is shown in figure 1a. Note that once the UDP flows are established, the TCP server is isolated from the clients. Even if the TCP data packets are delivered to the server, the TCP-ACK packets sent by the server still face MAC congestion and are likely to be dropped.

Case 2. UDP streams do not isolate the server. The scenario is shown in figure 1b. Figure 2 compares the performance of various clients for case 1 and 2. Clearly, connections 6 and 7 (node 156 and 162 in the grid) are able to achieve a reasonable goodput in case 2. From these studies (more results are available in [21]) we concluded that:

1) UDP flows load the nodes on the route, leading to medium capture along the paths.
2) It is difficult for TCP based flows to "cross" such a heavily loaded path.

[5] Here, default settings refer to the use of default settings of IEEE 802.11 MAC (DHSS), AODV (with link layer detection) and TCP (Reno) as used in ns2 [18].
[6] These settings provide the highest throughput amongst the simulations carried out.

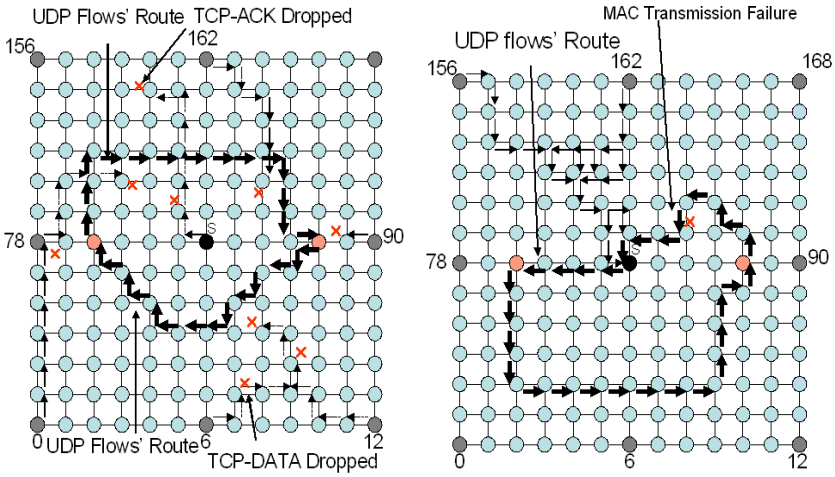

Fig. 1a. When the TCP Server is encompassed by the UDP streams, all TCP clients suffer
Fig. 1b. When TCP server is not isolated some TCP clients achieve normal throughput

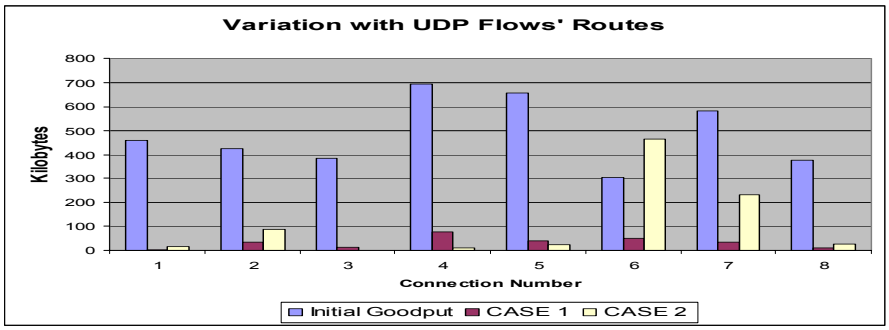

Fig. 2. *Column 1:* Initial Goodput, *Column 2:* Case 1 (all clients suffer), *Column 3:* Case2 (clients 6 and 7 are able to access the server)

Stabilized Routing and Increased Persistence for MAC Transmission. A large number of MAC transmission failures result in poor TCP performance. We increase the persistence of the 802.11 MAC protocol by increasing the retry limit parameter from the default 7 to 21. We observed that the aggregate goodput of TCP clients improves significantly. Specifically, in the presence of UDP streams aggregate goodput for clients is approximately 33% of the initial goodput, as opposed to the 10% observed previously. Increasing the retry limit is not an acceptable solution, especially in case of mobility. However, for a static topology, this increase seems to benefit TCP clients. Hence, we use this case as a base for comparison with other schemes.

MAC and Network Layer Fairness. One might expect that a fair medium access control scheme would alleviate the effects discussed thus far. One could also expect such improvements by enforcing fairness at the network layer; however for doing so different policies may be chosen. In order to understand the effects of enforcing fairness, we consider the following scenarios for study:
1) We replaced the IEEE 802.11 with a fair MAC protocol. A Time Division Multiple Access scheme was implemented for a 5x5 grid. Four clients were placed on the corners and the server was again at the center. The time slot was chosen such that a MAC frame carrying single TCP packet could be transmitted in a time-slot. Although this scheme is unrealistic, we performed this experiment in order to understand the effects of MAC layer fairness on TCP performance. The performance of TCP was still found to be poor. The primary cause for this degradation was the use of the FIFO queue at the interface which allows UDP based flows to fill queues on various nodes.
2) We implemented fair-queuing at the interface queue and used it in conjunction with the fair MAC protocol described above. We studied four packet classification schemes for enforcing fairness based on IP-source, IP-destination, next-hop and previous-hop respectively. To clarify the schemes, if we wish to enforce fairness based on the next-hop, and a node had K neighbors, it would maintain K queues, one for each neighbor and serve these queues in a round-robin fashion. Similarly, queues may be maintained on a previous hop, IP-source or IP-destination basis. Our results show that locally fair schemes (next-hop and previous-hop) are inadequate in preventing TCP performance degradation in the presence of the UDP streams. However, globally fair schemes based on IP-source and IP-destination address are able to prevent TCP performance degradation. This is because with the local schemes, if a TCP flow happened to share a link with a UDP flow, from then on, its packets would not be distinguished from the UDP flows' packets (for detailed results see [21]). Having understood the desirable effects of network fairness, we investigate its utility with the IEEE 802.11.

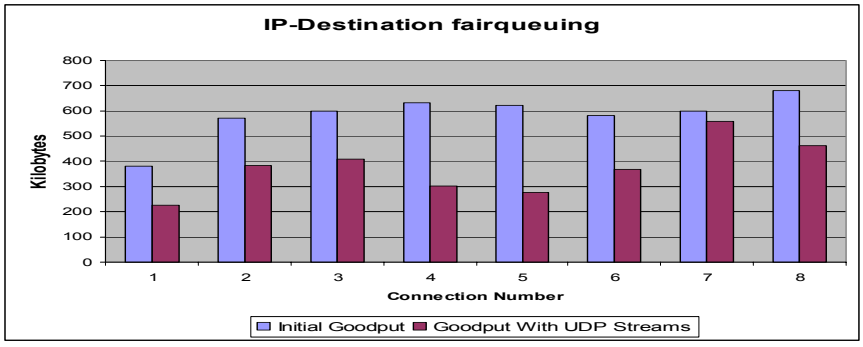

Fig. 3. IP-destination queuing improves aggregate TCP goodput significantly

Network Layer Fair-Queuing with the IEEE 802.11 MAC Scheme. In the scenario simulated, the TCP clients send data packets to the server. For this scenario, IP-destination based queuing is found to provide the best results. This is because TCP ACKs originating from the server are placed in separate queues by nodes on the

return path to various clients. Thus, the TCP ACKs can receive up to 80% of the share of the bandwidth. This reduces the round trip times for various TCP connections, thus, increasing their goodput. If instead the server is sending data packets, then IP-source based queuing would be better. Figure 3 shows the performance of the TCP connections with IP-destination based queuing. We observe that:
1) Under the initial conditions without UDP traffic, with fair queuing the average performance of the TCP clients improves marginally (5%).

2) In the presence of UDP based flows, the TCP clients can now achieve about 64% of the initial goodput. Previously, with the increased MAC persistence this value was about 33% of that achieved under initial conditions). For reasons explained above, for IP-source based fair-queuing this value was 42%.

3) The improvement comes at a price for UDP connections. The average reduction in UDP goodput is about 418 Kbytes. However, the TCP goodput increases by about 1501 Kbytes.

4) The UDP based flows were no longer able to create partitions in the network as observed earlier (Fig 2).
Fair-queuing schemes require each node to maintain state information with regards to the flows passing through the node. However, we argue that unlike in the Internet, this is less of a concern in ad hoc networks wherein most of the nodes would relay only a limited number of flows.

4 Back-pressure

4.1 Overview

Our objective is to prevent an aggressive UDP source from injecting packets at a rate higher than what the network can afford. We achieve this by restricting the allocated buffer space for each particular flow at every node that the flow traverses. Consider a node that suddenly starts serving incoming flows at a lower rate[7]. Gradually, the buffer occupancy of incoming flows at the node is bound to increase. With the use of back-pressure, gradually, the rate at which the flows send the data to the node will also reduce. The effect travels *backwards* all the way to the source[8]. The source is then forced to adapt its rate to conform to the available bandwidth for the flow. Furthermore, if the congested node is able to serve the flows at a higher rate, then sources are automatically able to send packets at a higher rate.

[7] This can be due to medium access delays or from additional flows passing through the server.
[8] Hence the name back-pressure is adopted for the scheme.

4.2 Implementation

To implement back-pressure, we set a threshold, referred to as the back-pressure-threshold that restricts the buffer allocation to a particular IP-source (or IP-destination) at any node. Then, using the promiscuous mode of operation[9], a node keeps track of the number of packets in a downstream neighbor's queue. Upon receiving a MAC frame, the nodes operating in a promiscuous mode can determine if a neighbor has transmitted a packet belonging to a particular IP-source (or destination). Thus, for a flow, each upstream node is aware of the queue size at a downstream neighbor to which it forwards packets from the flow. Once the node recognizes that the back-pressure-threshold is reached at a downstream neighbor, it stops transmitting packets from the flow to that neighbor. Subsequently, the node's back-pressure limit is reached which would prevent its previous upstream relay from sending further packets from that flow. This effect is propagated all the way to the source of the flow.

It should be noted that the interface queue is a passive queue, i.e., it needs stimulus with respect to de-queuing of the packet. In our scheme the de-queuing of a packet is dependent on buffer of the neighbor. At present, our objective is to measure the gains possible by use of the backpressure. Thus, we have simplified the implementation by utilizing a virtual, globally accessible array that dynamically records the queue lengths for each flow at each node in the network.

4.3 Simulation Results

We observe that back-pressure prevents continuous medium occupation by nodes on the path of UDP flows. TCP goodput increases with this availability of the medium and reduction in transmission failures and buffer overflow. The improvements with IP-Destination based queuing combined with back-pressure are similar and are shown in figure 4. We observe that:

1) With a back-pressure threshold value of 1, the average gain in goodput over IP-Source Address based fair-queuing scheme for the TCP clients is 95%. (82% over the IEEE 802.11 MAC scheme as compared to the 42% achieved with fair-queuing).

2) The gain in TCP does not bring down the goodput of the UDP flows. In fact, we witness a gain of 4.5% in throughput with UDP back-pressure.

4.4 Cause of Goodput Improvement

In figure 5, we plot the packets sent and received by UDP and TCP agents for IP-Source fair-queuing and UDP back-pressure. Notice that with back-pressure, the number of packets injected into the network by the UDP source is almost one-fifth (22%) of that with the simple IP-source fair-queuing. However, the number of packets actually delivered by the UDP flows is almost equal in both cases (back-pressure in fact results in the delivery of 4.5% more packets). With the increased medium availability TCP clients achieve a better throughput as seen in figure 5. This rate

[9] In promiscuous mode, nodes can overhear information broadcast on the wireless channel.

adaptation of the TCP source also leads to a reduction in MAC and Interface queue (IFQ) related drops for the TCP connections as well as for UDP flows.
Another significant benefit of backpressure is that it reduces the jitter in the end-to-end delay of the UDP flows' packets. Further, we also witness an increase in jitter with increase in backpressure threshold value (see [21] for details).

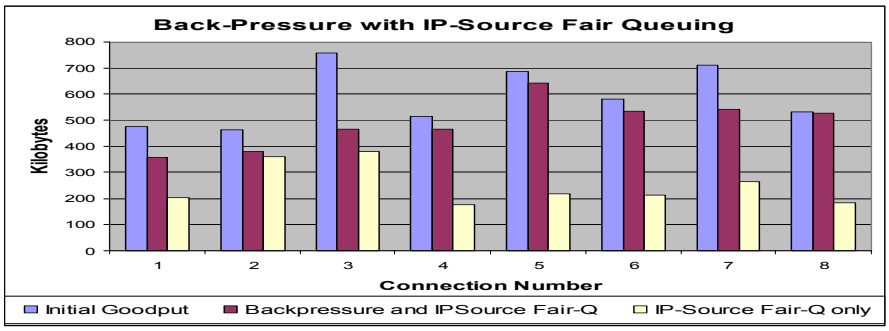

Fig. 4. Back-pressure Improves Performance Significantly in comparison to fair-queuing

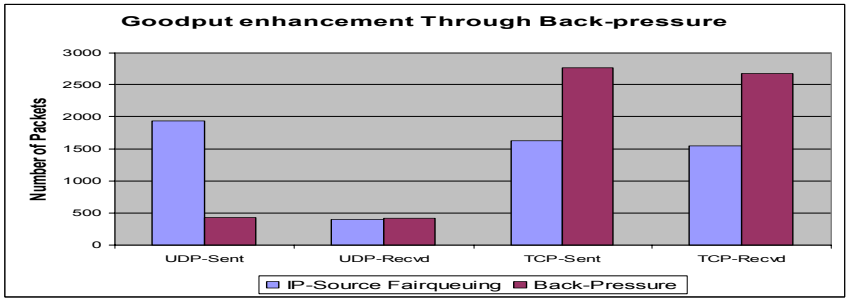

Fig. 5. With back-pressure number of packets injected by the UDP sources in the network is reduced by 80%, while the number of packets received by UDP destination remains the same as with fair-queuing, i.e. source adapts to delivery rate at a destination. Number of TCP packets sent by the clients increases by 95% and so does the packets reaching the TCP server

4.5 Tunable Operation through UDP Back-Pressure

Figure 6 shows the performance of backpressure with different threshold values. By tuning the backpressure threshold, we are able to offer variable TCP and UDP goodputs. For example, if the threshold value is tuned to 1, the TCP clients achieve 82% of the goodput under initial conditions. Tuning the threshold to a value of 6 provides greater UDP throughput at the cost of TCP goodput. In general, a large threshold value will provide an advantage to UDP flows and a smaller threshold provides an advantage to TCP flows. Detailed results regarding the trade off between

the choice of various back-pressure thresholds, retry limits and frame sizes can be found in [21].

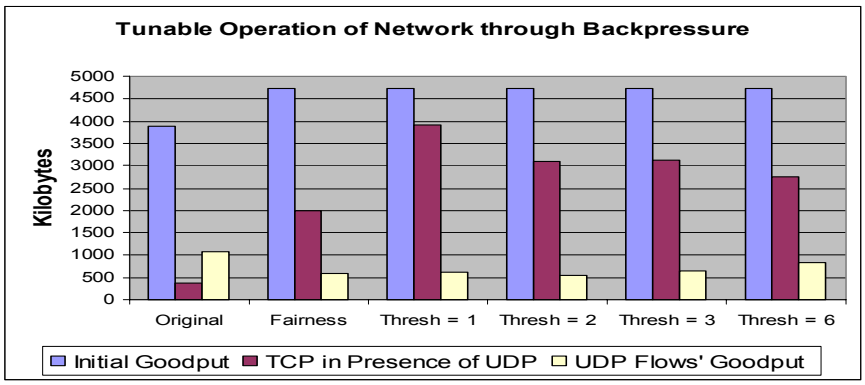

Fig. 6. Tunable operation of Network can be achieved by varying back-pressure threshold value. With Threshold =1, TCP achieves high goodput. Increasing the threshold increases the UDP goodput and reduces TCP goodput. For large value of Threshold, the results are similar to simple fairness

5 Conclusions and Future Work

In this work we highlighted the poor performance of TCP connections in the presence of UDP flows. Our main conclusions are:
1) Simple measures like stabilized routing or increasing MAC persistence help achieve a 10% to 33% in the TCP goodput in the presence of interacting UDP flows.
2) A fair MAC protocol alone is insufficient in improving the performance of TCP in the presence of UDP flows.
3) Fair-queuing improves the TCP performance significantly. (33% to 42%, and 33% to 64% for IP-source and IP-destination based queuing respectively).
We propose a new scheme called back-pressure that provides hop-by-hop flow control. Back-pressure forces UDP sources to adapt to the dynamics of congestion in the network. We perform simulations that show that back-pressure
a) Can provide gains in goodput of as high as 95 % when compared to fair queuing.
b) Improves TCP performance significantly without sacrificing UCP flows' goodput.
c) Can be tuned for coarse allocation of bandwidth amongst TCP and UDP flows.

References

1. Floyd, S., and Jacobson, V.: Random Early Detection gateways for Congestion Avoidance, in IEEE/ACM Transactions on Networking, V.1 N.4, August 1993, p. 397-413
2. http://www.cisco.com/univercd/cc/td/doc/product/software/ios112/ios112p/gsr/wred_gs.pdf

3. Xu, S., Saadawi, T.: Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks?, in IEEE Communications Magazine, 39(6), Jun. 2001
4. Gerla, M., Tang, K., Bagrodia, R.: TCP performance in wireless multi-hop networks, in 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'99), Feb. 1999
5. Jiang, R., Gupta, V., Ravishankar, C.V.: Interactions Between TCP and the IEEE 802.11 MAC Protocol, in DARPA Information Survivability Conference and Exposition (DISCEX), Volume I, 2003
6. Gupta, V., Krishnamurthy, S., Faloutsos, M.: Denial of Service Attacks at the MAC Layer in Wireless Ad Hoc Networks, in Proceedings of IEEE Milcom, 2002
7. Monks, J.P., Sinha, P., Bharghavan, V.: Limitations of TCP-ELFN for ad hoc networks, in Proc. of The 7th Int'l Workshop on Mobile Multimedia Communications MoMuC 2000
8. Holland, G., Vaidya, N.: Analysis of TCP performance over mobile ad hoc networks, in 5th annual ACM/IEEE International Conference on Mobile Computing and Networking, Aug. 1999, pp. 219--230
9. Wang, Y., Garcia-Luna-Aceves, J.J.: Throughput and Fairness in a Hybrid Channel Access Scheme for Ad Hoc Networks, in Proc. of the IEEE Wireless Communication and Networking Conf. (WCNC 2003)
10. Luo, H., Medvedev, P., Cheng, J., Lu,S.: A Self-Coordinating Approach to Distributed Fair Queueing in Ad Hoc Wireless Networks, in IEEE INFOCOM 2001
11. Kanodia, V., Li, C., Sabharwal, A., Sadeghi, B.,Knightly, E.: Distributed Multi-Hop Scheduling with Delay and Throughput Constraints, in proceedings of ACM MOBICOM 2001
12. http://standards.ieee.org/getieee802/802.11.html
13. Almes, G.T., Lazowska, E.D.: The Behavior of Ethernet-like Computer Communication Networks, Technical Report 79-05-01, University of Washington, 1979
14. Bharghavan, V., Demers, A., Shenker, S., Zhang, L.: MACAW: A Media Access Protocol for Wireless LAN's, in Proc. ACM SIGCOMM 9, pp. 212-25, London, UK, 1994
15. Perkins, C.E., Royer, E.M.: Ad-hoc On Demand Distance Vector Routing, in 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'99)
16. Chakeres, I.D., Belding-Royer, E.M.: The Utility of Hello Messages for Determining Link Connectivity, in Proceedings of the 5th International Symposium on Wireless Personal Multimedia Communications (WPMC) 2002, Honolulu, Hawaii, October 2002
17. Mathis, M., Mahdavi, J., Floyd, S., Romanow, A.: TCP Selective Acknowledgement Options. RFC 2018, 1996
18. http://www.isi.edu/nsnam/ns/
19. Johnson, D.B., Maltz, D.A., Broch, J.: DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks. in Ad Hoc Networking, Addison-Wesley, 2001
20. Demers, A., Keshav, S., Shenker, S.: Analysis and simulation of a fair queuing algorithm, in Journal of Internetworking Research and Experience, vol. 1, no. 1, pp. 3--26, Sept. 1990
21. Gupta, V., "Denial of Service attacks in Wireless Ad Hoc Networks", Masters Thesis. University of California, Riverside, Department of Electrical Engineering
22. Lu, S., Bharghavan, V., Srikant, R.: Fair Scheduling in Wireless Packet Networks, in IEEE/ACM Trans. Networking, vol. 7, no. 4, pp. 473--489, Aug. 1999
23. Vaidya, N.H, Bahl, P, Gupta, S.: Distributed fair scheduling in a wireless LAN, in Sixth Annual International Conference on Mobile Computing and Networking, Boston (2000)
24. Pazos, C.M., Sanchez Agrelo, J.C, Gerla, M.: Using BackPressure to Improve TCP Performance with Many Flows, in IEEE INFOCOM'99, New York, NY, USA, March 1999