Improving Fairness among TCP Flows crossing Wireless Ad Hoc and Wired Networks

Luqing Yang[1] Winston K.G. Seah[1,2] Qinghe Yin[2]
[1]Department of Electrical Engineering, National University of Singapore, Singapore 119260
[2]Institute for Infocomm Research, #02-34/37, Teletech Park, 20 Science Park Road, Singapore 117476
Email: [1]{engp1781@nus.edu.sg} [2]{winston, yinqh@i2r.a-star.edu.sg}

ABSTRACT
In scenarios where wireless ad hoc networks are deployed, sometimes it would be desirable that ad hoc nodes can communicate with servers in wired networks to upload or download data. In these cases TCP connections will span both wireless ad hoc and wired domains. However, TCP often faces severe unfairness in this type of connection scenario, which forces some TCP flows to completely stop transferring any data despite all links being in good states. In this paper, we propose a simple scheduling scheme, which helps competing TCP connections to achieve fairness without much throughput loss. Simulation results show that our scheme successfully eliminates the extreme unfairness existing in above-mentioned scenarios.

Categories and Subject Descriptors
C.2.0 [**Computer-Communication Networks**]: General – data communications.

General Terms
Algorithms, Performance.

Keywords
TCP, Fairness, Ad Hoc Network.

1. INTRODUCTION
Wireless ad hoc network have gained more and more attention in research community recently. It is a promising technology to provide communications between nodes, mobile or stationary, in situations where no backbone infrastructures or central control entities are available. Each node in wireless ad hoc networks can not only play the role of a communication endpoint, but also a router between nodes that are multiple hops apart. Thus it is very useful in battlefield, disaster rescue and many other scenarios. In these scenarios, there are times that the ad hoc nodes may need to establish TCP connections with a server in wired networks to update databases or download and upload data files. For example, a node that found a potential target may need to send the collected data back to command center for processing and further decisions; or the command center needs to update the node with latest target information.

However, previous research [1][2][3] has shown that in ad hoc networks TCP performance faces challenges both in terms of throughput and fairness. Moreover, [4] showed that in scenarios where TCP spans multihop wireless and wired networks, unfairness caused a new problem, i.e. the selection of the TCP maximum congestion window value. On the one hand, a large congestion window enables a TCP connection to effectively fill the pipe and ensures a desirable throughput, but at the cost of severe unfairness, the consequence of which is that some TCP connections are forced to stop transferring any data while other TCP connections capture the channel persistently, although all the nodes in the network are well connected. On the other hand, the small congestion window does help TCP connections to share the channel fairly, but the aggregate throughput is unacceptably low due to the stop-and-wait mode caused by the small congestion window.

To solve the above problem, we propose to use a simple non-work-conserving scheduling algorithm to work with the IEEE802.11 [5] Medium Access Control (MAC) protocol, replacing the normal FIFO work-conserving scheduling scheme in ad hoc networks. Simulation results show that our scheme achieves acceptable throughput and fairness at the same time, and alleviates the TCP maximum congestion window value selection problem.

The rest of the paper is organized as follows: we describe our scheduling algorithm in Section 2. Section 3 presents the simulation results. Brief analysis and discussions are given in Section 4. Section 5 reviews related work in the literature. Finally, Section 6 summarizes our contribution and proposes some future work.

2. DESIGN OF OUR SCHEME
In our subsequent discussions, we call the packets from application layer "data packets", and we use "routing packets" to denote packets generated by routing protocols. In general FIFO work-conserving scheduling for ad hoc networks, usually routing packets are treated as high priority packets over data packets; upon arrival they are enqueued before all data packets. When the queue knows from MAC that it can output another packet, it will send the packet at the head of the queue to MAC immediately; this is what has been implemented in the NS simulator [6]. Here,

we propose to set a timer every time after the queue sends a data packet to MAC. Only after this timer expires, can the queue output another data packet to MAC. Moreover, the duration of the timer is decided by the queue output rate. The detailed algorithm is as follows:

The queue will set a timer every time after it sends a data packet to MAC for transmission. The duration of the timer is a sum of three parts. We denote them as D_1, D_2, and D_3, respectively. D_1 represents the queue estimation on how long the channel needs to transmit this packet if no contention occurs. It can be calculated using packet length divided by the bandwidth of the channel. D_2 is a delay, the value of which is decided by the recent queue output rate. The queue calculates the output rate by counting the number of bytes, C, it outputs in every fixed interval T. Thus the value of D_2 is updated every T seconds. To decide the value of D_2, we set three thresholds X, Y and Z (X < Y < Z) for C, as shown in (1). D_3 is a random value uniformly distributed between 0 and D_2. We use D_3 to randomize the delay we add in the queue and avoid synchronization phenomenon. Although IEEE802.11 will have some randomization in choosing the backoff period, we can foresee that with D_3 the possibility of synchronization and collisions are further reduced.

$$\left.\begin{array}{l} C <= X, D_2 = D_{21} \\ X < C <= Y, D_2 = D_{22} \\ Y < C <= Z, D_2 = D_{23} \\ C > Z, D_2 = D_{24} \\ D_{24} > D_{23} > D_{22} > D_{21} >= 0 \end{array}\right\} \qquad (1)$$

D_{21}, D_{22}, D_{23}, and D_{24} denote the values of the delay that D_2 will take. The heuristics behind equation (1) is that we want to insert delay in scheduling according to the queue output rate in last interval T. If C is very low, it is quite possible that the node is at clear disadvantage in media contention, thus a very small delay value D_{21} (D_{21} can be set to zero) is inserted in scheduling. With the increasing of the value of C, the delay inserted in scheduling is also increased. Therefore, the more aggressive the node is, the more severely it is punished. We regard those nodes that output less than X bytes in last interval T as nodes experiencing unfairness, and we regard nodes that output more than threshold Z in last interval T as greedy nodes that probably use the channel more than their fair shares. There is a trade-off on deciding the values of X, Y and Z as well as the delay D_2 takes. For example, if the delays are too small, throughput is guaranteed but the fairness cannot be improved; on the other hand, if the thresholds are too low, it is possible that high delay is even imposed on the node with moderate output rate, thus the fairness is protected but the throughput degrades too much. We can use more than three thresholds to gain more efficiency but at the cost of higher implementation complexity.

For routing packets, we still treat them as high priority packets over data packets. Upon arrival, they will be enqueued before all other data packets. Once the queue knows from MAC that it can transmit a packet, the routing packet at the head of the queue is dequeued immediately regardless of whether there is a timer pending. Unlike data packets, the queue will not set any timer after sending a routing packet and will not count them in calculating C.

3. SIMULATION RESULTS

3.1 Simulation Environment

We use the NS simulator [6] with ad hoc extensions provided by the MONARCH project [7] to do simulations. DSDV [8] is selected as the routing protocol because in our targeted scenarios TCP connections span wired and wireless domains. At MAC layer, NS implemented the IEEE802.11 MAC protocol's Distributed Coordination Function. We implement our scheduling algorithm by modifying the priority queue implementation in NS. Table 1 and Table 2 list relevant parameter settings in NS and in our scheme, respectively.

Table 1. Parameter settings in ns

Wireless channel bandwidth	2 Mb/s
Wireless node interface queue limit	50 Packets
Version of TCP used	New Reno
Nominal radio transmission range	250 m
Buffer management for wired nodes	Drop Tail
Queue limit for wired nodes	50 Packets
Packet size	1024 Bytes

Table 2. Parameter settings in our scheme

X	10000 Bytes
Y	20000 Bytes
Z	50000 Bytes
D21	0 s
D22	0.002 s
D23	0.005 s
D24	0.01 s
Updating interval T	2 s

Xu Kaixin et al [4] identified the severe unfairness among TCP flows across wireless ad hoc and wired networks. Thus, we first adopt a scenario similar to situations used in [4] to show how our scheme solves the problem.

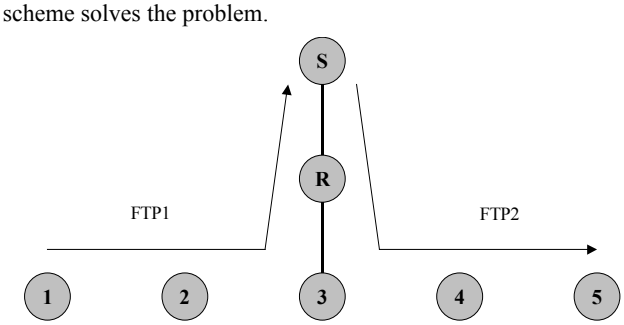

Figure 1. Scenario A for simulation

In Figure 1, node 3 is a gateway node linking wired and wireless domains. Wired links connect server S to router R and eventually to gateway node 3. Five wireless nodes, node 1 to node 5 are spaced evenly and each is 200m away from its neighbors. At this distance each node can only communicate with its immediate neighbors successfully. Two FTP sessions, FTP1 and FTP2 are used as TCP traffics. Both FTP sources have infinite backlogs to send. FTP1 is from node 1 to server S; FTP2 is from server S to

node 5. Each session traverses two hops in the wireless domain and two hops in the wired domain. To leave enough warm up period and also give DSDV time to exchange routing information, both FTP1 and FTP2 start at 150s and stop at 250s. The bandwidth of each wired link is 2 Mb/s.

As to wired link delay, we consider two different situations. In the first situation, the delay of each wired link is 5ms whereas in the second situation the delay of each wired link is 45ms. We use the former to simulate situations where ad hoc nodes communicate with a local wired server and the latter to simulate the case where ad hoc nodes access remote servers on the Internet. We think that these two situations do have their practical applications in reality, such as, at battlefield or disaster rescue site, sometimes the ad hoc nodes at the front may have the requirement of reliable data communication with a local wired command center whereas sometimes ad hoc nodes need to access a remote server through the Internet to download or upload data files.

Because one parameter of TCP, the maximum congestion window, has great impact on fairness among TCP flows in wireless ad hoc networks, we run simulations with different maximum congestion window values. In the rest of the paper, we denote maximum congestion window as maxcwnd. The values of maxcwnd we select are 1, 4, 8, 16 and 32 packets.

We will give a brief explanation on the reasons of the unfairness among TCP flows in Section 4 of this paper. But we would refer interested readers to [3][4] for detail analysis and explanations.

For convenience, in the rest of the paper we refer to the work-conserving FIFO scheduling scheme that is commonly used as the normal FIFO scheme. We call the scheme proposed in this paper our scheme. TCP1 and TCP2 are used to denote the TCP connections corresponding to FTP1 and FTP2. We use goodput of each connection, aggregate goodput, and fairness as the metrics for evaluation. Here, we define the goodput of a connection as total number of bits that are successfully received by the receiver in a unit time.

3.2 Simulation Results

Figure 2 and Figure 3 present the respective results obtained by using the normal FIFO scheme and our scheme. The wired link delay is 5ms. It can be seen that in both figures the aggregate goodput increases with the increasing of maxcwnd value. But the individual goodput of each connection undergoes quite different changes in the two figures.

In Figure 2, the two TCP flows share the channel fairly only when maxcwnd is one packet. Once the maxcwnd is greater than one packet, the goodput of FTP1 drops drastically and the aggregate goodput almost completely belongs to FTP2. The higher the maxcwnd value is, the worse the fairness.

As a sharp contrast, in Figure 3 where we use our scheme, the two TCP flows always share the channel fairly regardless of the value of maxcwnd. Meanwhile, the aggregate goodput of our scheme is also quite acceptable. For example, comparing the aggregate goodput in Figure 2 and Figure 3 for maxcwnd value of 16 packets, the degradation caused by our scheme is only 11.2 percent. Actually, as a common phenomenon in any resource sharing system, it is not surprising that only serving one or few connections would achieve higher throughput than serving every connections fairly.

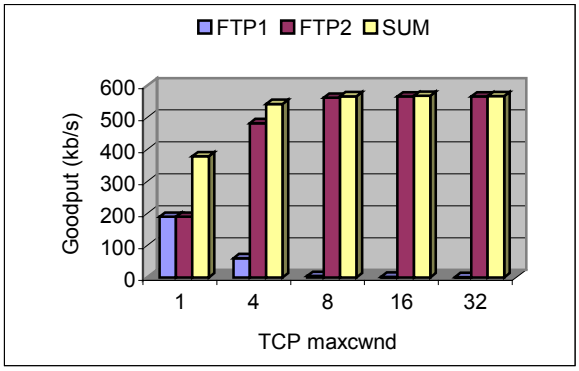

Figure 2. Goodput of TCP using normal FIFO scheme for Scenario A; Wired link delay = 5 ms

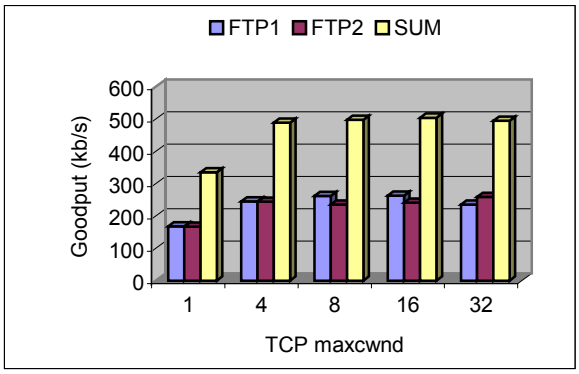

Figure 3. Goodput of TCP using our scheme for Scenario A; Wired link delay = 5 ms

To give better insights into the results of Figure 2 and Figure 3, we show examples of TCP1 and TCP2 acknowledgement number progress in Figure 4 to Figure 7.

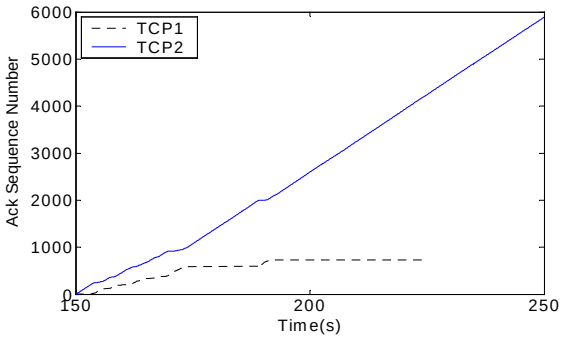

Figure 4. maxcwnd = 4, wired link delay = 5ms, FIFO scheme

We can see that the problem here is more than unfairness. In fact, during the simulations, one connection is stifled by the other connection. This is different from the situations in wireless LAN or cellular network, where unfairness usually only causes some connections to use resources less than their fair shares. However, here in ad hoc networks, the unfairness results in TCP1 being active only at the beginning of simulation, then it timeouts consecutively and is forced to stop transferring any packets,

despite all links being in good state. It is clear that the normal FIFO scheduling scheme cannot ensure every connection a sustainable throughput, let alone a fair share. On the contrary, both Figure 5 and Figure 7 show that our scheme maintains a good fairness property.

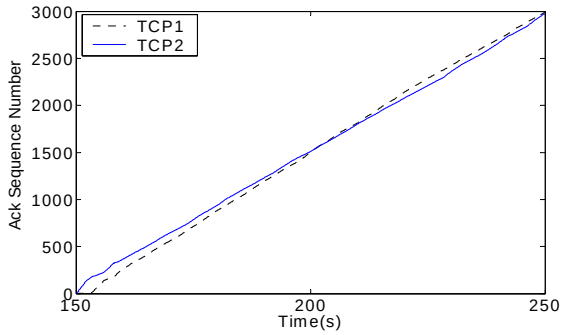

Figure 5. maxcwnd = 4, wired link delay = 5ms, our scheme

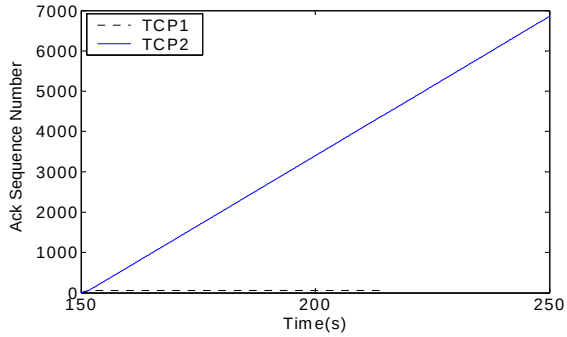

Figure 6. maxcwnd = 8, wired link delay =5ms, FIFO scheme

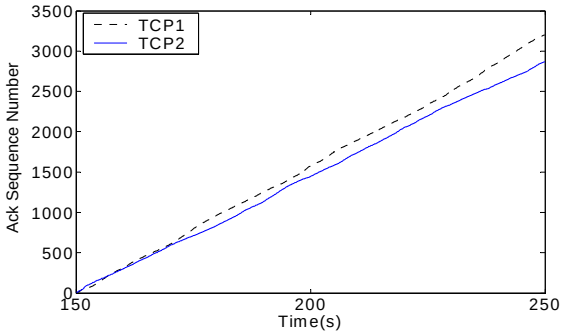

Figure 7. maxcwnd = 8, wired link delay = 5ms, our scheme

From another perspective, we argue that our scheme does improve the aggregate goodput compared with normal FIFO scheduling if fairness is taken into considerations. For example, when situations require sustainable throughput for every TCP connection, if using normal FIFO scheduling for the scenario in Figure 1, the only choice is to set the maxcwnd to one packet, and the aggregate goodput is 378.4 kb/s, as shown in Figure 2; if using our scheme, we can set maxcwnd to any large value to achieve both satisfactory goodput as well as fairness. In Figure 3, when maxcwnd is eight packets, the aggregate goodput is 497.5 kb/s, 31.5 percent higher than the 378.4 kb/s of the normal FIFO scheme.

To have a thorough investigation of our scheme, we change the wired link delay from 5ms to 45ms. Figure 8 and Figure 9 give the respective results of normal FIFO scheduling and our scheme.

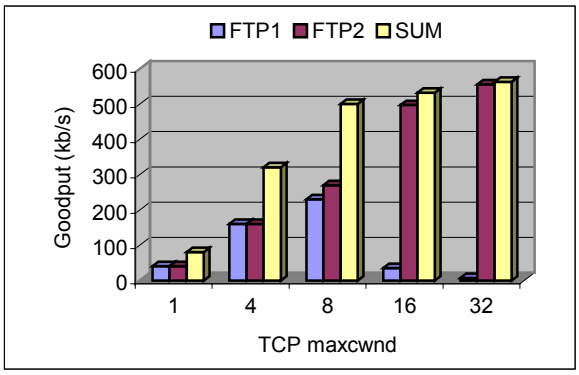

Figure 8. Goodput of normal FIFO scheme for Scenario A, Wired link delay = 45 ms

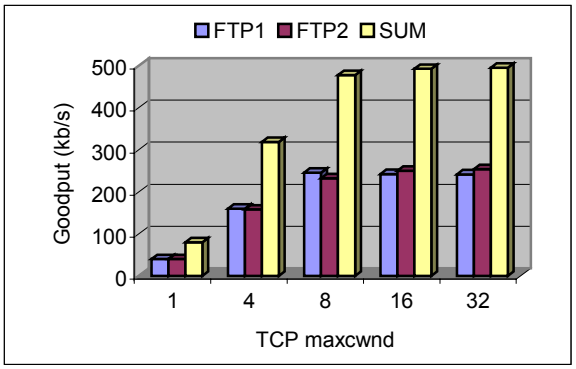

Figure 9. Goodput of our scheme for Scenario A, Wired link delay = 45 ms

Figure 8 indicates that as long as maxcwnd is no more than eight packets, the two TCP flows share the media fairly when using the normal FIFO scheme. When maxcwnd is eight packets, the aggregate goodput is 500.5 kb/s and quite acceptable. Generally, we could say that eight packets is an optimal point for maxcwnd value in this case if we take both fairness and goodput into considerations.

But the problem of using normal FIFO scheme is that there is no method with which we can determine this optimal point before we run simulations. Furthermore, the delay and available bandwidth on wired network may be dynamic due to cross traffic, thus the corresponding optimal point is also dynamic. Therefore, even if we have a method to determine the optimal value for maxcwnd, it is quite difficult to set it dynamically, not to mention sometimes the optimal point does not exist at all. An example is shown Figure 2, where you cannot obtain high aggregate goodput and acceptable fairness at the same time.

However, by using our scheme, there is no need to find the optimal point for maxcwnd. This is because in our scheme the aggregate goodput increases with the increasing maxcwnd value while maintaining fairness; we only need to set maxcwnd to a value high enough to achieve high goodput. In Figure 9 aggregate goodput corresponding to maxcwnd of 8, 16 and 32 packets are

477.8 kb/s, 492.5 kb/s, and 495.0 kb/s. Compared with 500.5 kb/s goodput obtained at the optimal point in Figure 8, our scheme works very well.

In fact, besides the scenario in Figure 1, we found that severe unfairness exists in many situations where there are TCP flows crossing ad hoc and wired domains. Here, we present another scenario to demonstrate the effectiveness of our scheme. As shown in Figure 10, this time FTP2 is still from node S to node 5 whereas FTP1 is from node 1 to node 2. Both sessions start at 150s and stop at 250s. TCP1 and TCP2 are the TCP connections corresponding to FTP1 and FTP2. The bandwidth and delay of each wired link is 2Mb/s and 5ms.

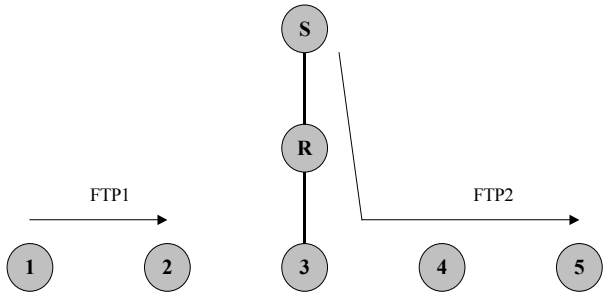

Figure 10. Scenario B for simulation

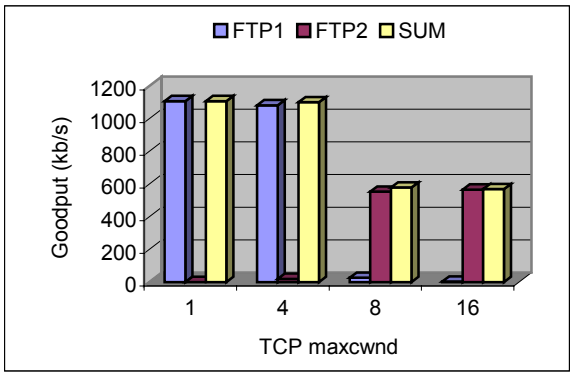

Figure 11. Goodput of TCP using FIFO scheme for Scenario B

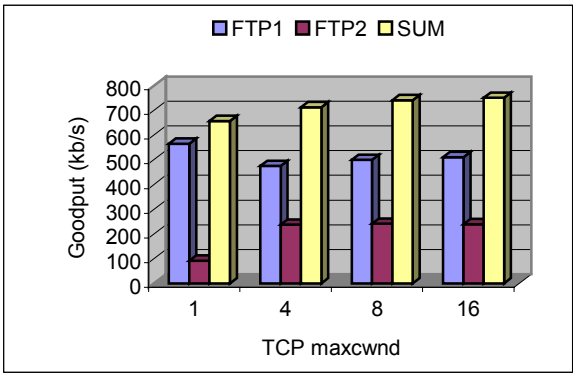

Figure 12. Goodput of TCP using our scheme for Scenario B

Figure 11 gives the results of using normal FIFO scheduling. The severe unfairness is shown clearly. When maxcwnd is one or four packets, FTP1 will stifle FTP2. On the contrary, when maxcwnd is eight or sixteen packets, FTP2 will stifle FTP1. It is evident that these two connections cannot coexist whatever maxcwnd value is used.

Figure 12 shows the results obtained by using our scheme. Now, the two FTP sessions coordinate well in sharing the channel and they both achieve sustainable throughput during simulation. It is reasonable that the goodput of FTP1 is more than two times the goodput of FTP2, because FTP1 is a one-hop connection whereas FTP2 crosses two hops in wireless and two hops in wired domain.

To gain more insights, Figure 13 and Figure 14 present examples of TCP acknowledgement number progress by using the normal FIFO scheme and our scheme, respectively.

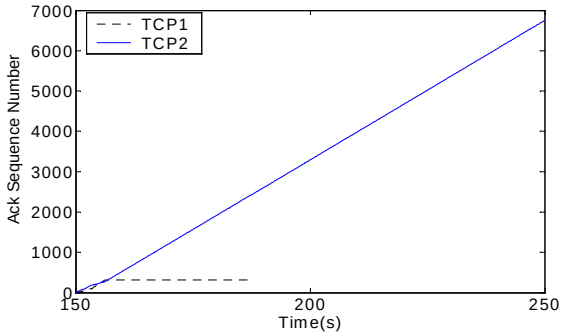

Figure 13. FIFO scheme, maxcwnd = 8, Scenario B

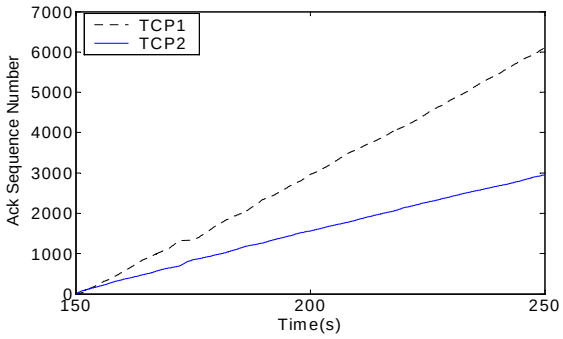

Figure 14. Our scheme, maxcwnd = 8, Scenario B

The above-mentioned extreme unfairness not only exists in wired-cum-wireless ad hoc environments, but also exists among TCP flows in pure ad hoc environments. Figure 15 shows a pure ad hoc scenario. Five nodes evenly spaced with 200 meters away from each other. One TCP flow (TCP1) is a two-hop connection from node 5 to node 3, whereas another TCP flow (TCP2) is from node 1 to node 2. Xu and Saadawi [3] have showed that in this scenario once the one-hop connection starts, the two-hop connection is completely forced down and even cannot get a chance to restart. In our simulation, DSR [9] is used as the routing protocol. The two-hop connection starts at 0 second, whereas the one-hop connection starts at 10.0 second.

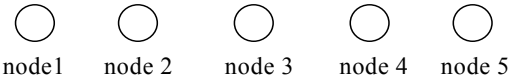

Figure 15. A pure ad hoc scenario with extreme unfairness

From Figure 16 we can see that just as [3] has stated, the one-hop TCP flow completely force down the two-hop TCP connection. The TCP1 flow does not receive any new acknowledgement after 31.5 second. Figure 17 shows the results obtained by using our scheme, now both flows progress roughly smooth and co-exist well. The extreme unfairness once again was eliminated by our scheme.

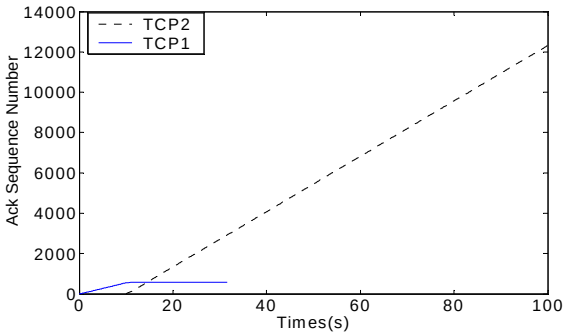

Figure 16. Ack Sequence Progress, normal FIFO scheme

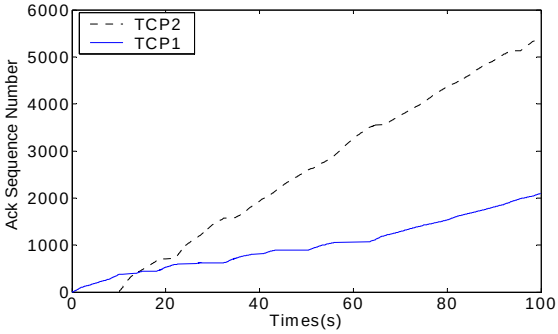

Figure 17. Ack Sequence Progress, our scheme

One thing to note is that in the simulation results the aggregate goodput experiences moderate degradation by using our scheme compared with using normal FIFO scheme. However, the degradation is acceptable considering the fact that it completely eliminates the extreme unfairness and ensures a sustainable throughput for every TCP connection. Moreover, in situations with extreme unfairness, the term of aggregate goodput is misleading because it is almost completely contributed by one or few connections with all other TCP connections starved out. Considering the situations where ad hoc networks are deployed, such as battlefield and disaster rescue, everyone could foresee the implications of severe unfairness. No one will have the desire to implement an ad hoc network if it even cannot ensure every connection a sustainable throughput, let alone fair share, when all links are in good states.

4. ANALYSIS AND DISCUSSIONS

The severe unfairness among TCP flows in IEEE802.11-based wireless ad hoc networks is the result of joint interactions of TCP, the FIFO work-conserving scheduling scheme, and the IEEE802.11 MAC protocol. First, TCP would send packets back-to-back when congestion window allows. Moreover, TCP traffic is self-clocking, i.e. the faster the acknowledgement comes back, the faster the data packet is sent out. Second, IEEE802.11 protocol always favors the node that is successful in the latest contention and leaves the failed nodes in exponential backoff. Lastly, work-conserving scheduling helps the last successful node to occupy the channel persistently by always providing packets in time.

In fact, as long as normal work-conserving scheduling is used with IEEE802.11 MAC protocol, even if other traffic types instead of TCP run on ad hoc networks, the unfairness problem will exist. From a different perspective, the packet generation interval of CBR traffic plays a similar role to some extent as the delay we added in scheduling. That is the reason why using work-conserving scheduling with CBR traffic usually does not show clear unfairness. However, once the CBR load is high, i.e. the packet generation interval is small enough, the unfairness will be shown clearly as well. Therefore, just as [3] has pointed out, TCP just exacerbates the unfairness compared with other traffic types.

By adding extra adaptive delay in scheduling, our scheme successfully eliminates the severe unfairness among TCP flows spanning multihop wireless and wired networks. The heuristics behind this design is that by introducing extra adaptive delay we want to only penalize those aggressive nodes to some extent, which grab the channel persistently, and help nodes that fail medium contention consecutively to enjoy the resource. The faster the node sends out packets, the longer the delay is inserted; the slower the node sends out packets, the shorter the delay is inserted, or even no delay is inserted.

As reported in [4], there is fundamental difference between TCP parameter tuning in wireless ad hoc networks with wired connection and in pure ad hoc scenarios. In pure ad hoc networks, large congestion window usually does not increase the throughput but worsen fairness. However, for TCP connections spanning both wired and wireless ad hoc domains, a small congestion window usually causes unacceptably low throughput. Using our scheme, large congestion window is no more a threat to fairness; on the contrary, it contributes to the satisfactory throughput.

We could foresee that in scenarios where there is only one TCP connection in the network, or each TCP connection always run in disjoint areas without contentions, our scheme would no doubt lead to unnecessary throughput loss. However, we argue that those cases are rare and unrealistic in a foreseeable practical wireless ad hoc network.

5. RELATED WORK

Several researchers have studied TCP fairness in multihop wireless networks. Tang and Gerla [10] investigated the issue of fair sharing of MAC among TCP flows in wireless ad hoc networks and a yield time scheme is proposed to improve fairness by introducing a larger yield time. However, larger yield time for every node will even penalize the node that used the channel less than its fair share, only because it is the node that used the channel last. Xu Kaixin et al [4] identified the unfairness problem when TCP spans both wired and wireless ad hoc networks, and they give valuable insights into TCP behavior over this type of environment. Xu and Saadawi [3] showed that hidden and exposed terminal problems, large sensing and interfering ranges are the main reasons of unfairness among TCP flows over IEEE802.11 MAC protocol.

To address fairness at MAC layer, several fair scheduling schemes have been proposed for general shared wireless channel environment. For example, Vaidya et al [11] presented a

distributed fair scheduling algorithm for wireless LAN that emulates Self-Clocked Fair Queuing in a distributed manner and chooses a backoff interval that is proportional to the finish tag of the packet to be transmitted, while Nandagopal et al [12] proposed a general analytical framework that can translate any given fairness requirement into a matching backoff scheme. These schemes address the fairness of MAC in general whereas here we try to eliminate the extreme unfairness among TCP flows in a broad class of environments. On the other hand, compared with the scheme proposed in this paper, these schemes are backoff-based solutions, i.e. they try to achieve fairness by modifying the backoff policy of MAC protocol.

6. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a scheme, which successfully eliminates the severe unfairness that TCP will likely face when the connections cross wired and IEEE802.11-based wireless ad hoc networks. Our work reveals that the work-conserving scheduling may amplify the adverse effects caused by interactions between TCP and the IEEE802.11 MAC protocol. The simulation results show that our scheme improves the fairness among TCP connections greatly and ensures every connection a sustainable throughput at the cost of moderate throughput degradation. Our future work includes testing of this scheme in scenarios with more sources contending for service from the base station. For proper parameterization, we will also use mathematics methods to analyze this scheme.

7. ACKNOWLEDGMENTS

We would like to thank the three anonymous reviewers of this paper who gave valuable comments. We also thank Mani Srivastava for his help and feedback on the revision of this paper.

8. RERERENCES

[1] M. Gerla, K. Tang, and R. Bagrodia, "TCP Performance in Wireless Multi-hop Networks," In proceedings of IEEE WMCSA '99, New Orleans, LA, Feb 1999.

[2] G. Holland, and N. Vaidya, "Analysis of TCP Performance over Mobile Ad Hoc Networks," In proceedings of MobiCom '99, Seattle, WA, Aug 1999.

[3] S. Xu, and T. Saadawi, "Does the IEEE 802.11 MAC Protocol Work Well in Multihop Wireless Ad Hoc Networks?" IEEE Communications Magazine, Volume 39, Issue 6, Jun 2001.

[4] K. Xu, S. Bae, S. Lee, and M. Gerla, "TCP Behavior across Multihop Wireless Networks and the Wired Internet," In proceedings of WoWMoM '02, Atlanta, GA, Sep 2002.

[5] IEEE, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," IEEE Standard 802.11, Jun 1999.

[6] Network Simulator. http://www.isi.edu/nsnam/ns/

[7] J. Broch, D. A. Maltz, D. B. Johnson, Y. Hu, and J. Jetcheva, "A Performance Comparison of Multi-hop Wireless Ad Hoc Network Routing Protocols," In ACM/IEEE Int. Conf. on Mobile Computing and Networking, Oct 1998.

[8] C. Perkins, and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," In proceedings of the SIGCOMM '94 Conference on Communications Architectures, Protocols and Applications, Aug 1994.

[9] J. Broch, D. B. Johnson, and D. A. Maltz, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks," Internet-Draft, draft-ietf-manet-dsr-03.txt, October 1999.

[10] K. Tang, and M. Gerla, "Fair sharing of MAC under TCP in wireless ad hoc networks," In proceedings of IEEE MMT '99, Venice, Italy, Oct 1999.

[11] N. Vaidya, P. Bahl, and S. Gupta, "Distributed Fair Scheduling in a Wireless LAN," In proceedings of ACM MobiCom 2000. Boston, Aug 2000.

[12] T. Nandagopal, T. E. Kim, X. Gao, and V. Bharghavan, "Achieving MAC Layer Fairness in Wireless Packet Networks," In proceedings of ACM MobiCom 2000, Boston, Aug 2000.