Transport Layer Fairness and Congestion Control in Multihop Wireless Networks

Thomas Kunz and Hao Zhang
Carleton University, Ottawa, Canada
tkunz@sce.carleton.ca

Abstract

Transmission Control Protocol (TCP) is a reliable, end-to-end transport protocol, which is widely used for data services and is very efficient for wired networks. However, experiments and research showed that TCP's congestion control algorithm performs very poorly over Wireless Ad Hoc Networks with degraded throughputs and severe unfairness among flows. This paper studies TCP's fairness and throughput issues in Wireless Ad Hoc Access Networks, and designs an improved congestion control algorithm based on the characteristics of the Wireless Ad Hoc Access Networks. The protocol is designed as extension to DCCP (Datagram Congestion Control Protocol) with a new congestion control component. We also implemented this congestion control algorithm in NS2. Simulation results show improvements on fairness and throughput achieved by using our congestion control algorithm.

1. Introduction

Wireless Ad Hoc Networks are multi-hop wireless networks, consisting of mobile nodes which are free to move about arbitrarily [4]. Frequently, Ad Hoc network will have gateways as connection to the Internet, which we call a Wireless Ad Hoc Access Network. Extensive research has been conducted concerning media access, routing and transport protocols for such networks. Transport layer protocols, which are specifically modified or designed for Wireless Ad Hoc Access Networks, are the focus of this paper. As most Ad Hoc Networks are built based on IEEE 802.11 wireless links, the work here also assumes that the MAC layer is an IEEE 802.11-like random access protocol.

TCP/IP is the protocol suite that defines the Internet. Transmission Control Protocol (TCP) is a reliable end-to-end transport protocol widely used for data services, which is primarily designed for wired networks and became very efficient and robust with years of enhancements. However, experiments and research showed that TCP's congestion control algorithm performs very poorly over Wireless Ad Hoc networks with degraded throughputs and severe unfairness among flows [5]. Research therefore has focused on further improving TCP to address the special characteristics of Wireless Ad Hoc networks.

Currently, the vast majority of the traffic in the Internet relies upon the congestion control mechanism provided by TCP. However, applications such as streaming video and Internet Telephony prefer timeliness to reliability. The reliability and in-order delivery algorithm provided by TCP often results in arbitrary delay, and TCP's rate control AIMD (Additive Increase and Multiplicative Decrease) algorithm causes very sharp bandwidth change upon the detection of one packet loss. Consequently, such applications often choose UDP, with either their own congestion control mechanisms implemented on top of it or none at all. Long-lasting UDP flows without any congestion control mechanism present a potential threat to the network. Also, congestion control mechanisms are difficult to implement and may behave incorrectly. This argues for a common base transport protocol, which is able to provide different congestion control algorithms to suit the needs of different applications.

This paper reviews TCP and TCP's performance problems in more detail in Section 2. Section 3 surveys a range of possible solutions to improve TCP suggested in the literature. Section 4 discusses a new congestion control algorithm implemented within DCCP (the Datagram Congestion Control Protocol) [8] and presents some simulation results. The results show that flows with the new congestion control protocol have very good inter-flow fairness, even in scenarios where TCP flows experience severe unfairness. At the same time, the aggregate flow throughputs are increased as well.

2. TCP in Multihop Wireless Networks

TCP provides a connection-oriented, reliable data transmission. The basic idea of TCP congestion control

is that TCP senders probe the network for available resources, and increase the transmission rate until packet losses are detected. TCP takes packet loss as indication of network congestion and triggers appropriate congestion control schemes.

2.1. TCP Throughput

In [5], it is shown that, when mobile nodes are fixed, the measured TCP throughput over IEEE 802.11 links (nominal data rate of 2 Mbps) for a single TCP flow decreases rapidly when the number of hops increases. In addition, the measured throughput gets worse when nodes are moving. As a general pattern, the throughput decreases as the nodes' moving speeds increase.

This throughput loss is caused by the unique characteristics of the Wireless Ad Hoc Networks. Wired networks have relatively low bit error rate, and TCP treats packet errors as indication of network congestion. In wireless transmission, bit error rates (BER) are higher due to fading and interference within wireless channels. Assuming that each error indicates network congestion and triggering the congestion control mechanism affects the throughput and link utilization. In addition, when nodes are moving, existing links may break, so the route between two nodes becomes obsolete and a new route has to be selected by the routing protocol. If the time of establishing a new route is longer than RTO, the TCP sender invokes congestion control and reenters the slow start phase. Mobility may also cause network partitions. If the TCP sender and receiver are in two different partitions, the TCP sender invokes congestion control and exponentially backs off the retransmission. If the partitions last longer than several RTOs, network inactivity could happen: even though the route has been reestablished, the sender still needs to wait until the RTO timer expires. Finally, TCP is based on ACKs for correct congestion control, so timely ACK reception is necessary for packet transmission and correct calculation of congestion window size and RTO. Common MANET MAC protocols such as IEEE 802.11 can result in a bunching of ACKs. Bunched ACKs cause bursty traffic and highly variable round trip times (RTTs). It may even cause the TCP sender triggering the congestion control due to starvation of ACKs.

2.2. TCP Fairness

In addition to throughput, it is also important to ensure that access to the network by each user remains fair. Fairness can be intuitively defined as the obtained throughput to its fair share of the bandwidth and can be quantified with metrics such as Jain's Fairness Index [6]. TCP flows experience severe unfairness in Ad Hoc Networks. TCP's window-based congestion control adjusts the congestion window size every RTT. Flows with longer RTT increase the congestion window slower than flows with shorter RTT. At the network routers, an unfair packet-dropping scheme, such as a simple FIFO drop tail scheme, may cause some flows to experience more losses than others. Medium access at a gateway is inherently unfair when using a MAC protocol such as IEEE 802.11. Upstream flows (from senders to the gateway) tend to occupy the whole media and the downstream flows (from the gateway to receivers) almost stop transmission when multiple upstream and downstream flows co-exist. Unfairness between the upstream and downstream flow throughputs is extremely high, with a ratio of up to 800 between them [10]. In a Wireless Ad Access Hoc network, IN TCP flows (from the wired part to the wireless part) get more bandwidth than the coexisting OUT TCP flows (from the wireless part to the wired part) [12]. IN flows obtain a much higher share of the bandwidth when mixed flows exist due to exposed and hidden node effects. TCP's own timeout and backoff schemes further worsen the unfairness.

In general, TCP works poorly in Wireless Ad Hoc Networks. This is caused by the high bit error rate over wireless links, arbitrary node mobility, as well as TCP's built in congestion control algorithm working with the contention based media access of IEEE 802.11. A large amount of research has focused on improving the throughput and fairness issues discussed above.

3. Related Work

In this section, existing proposals to improve the throughput and fairness problems of TCP over Mobile Ad Hoc Network are briefly reviewed. Most proposals focus on one of the two performance problems so the review is organized along these two categories.

3.1 TCP Performance Improvements

[5] analyzes of the use of explicit link failure notification (ELFN) on the performance of TCP over Mobile Ad Hoc networks. The objective of ELFN is to provide the TCP sender with information about link and route failures so it can respond properly. Upon receiving a route failure notice, the TCP sender enters a "stand-by" state and freezes all timers. A probe packet is sent periodically to probe the network to see if the

route has been reestablished. If an ACK is received, the TCP sender leaves the "stand-by" state, restarts the data transmission and resumes timers. The study shows significant throughput increase with the use of ELFN. But the simulation is only conducted with DSR as routing protocol choice and a single TCP flow.

When multiple flows exist, [2] shows that this approach cannot achieve throughput improvements, and it even degrades the performance as the mobility rate increases. It shows that, when the probing is conducted by several connections, the flooding of probe packets increases the congestion of the network. Also when a new route is determined, the TCP sender restarts to send at the old rate (i.e., using the old congestion window). If this congestion window is too big for the new connection, network congestion is likely to happen.

TCP-Feedback [11] is a similar feedback scheme in which the TCP sender utilizes the network layer feedback (Route Failure Notification or RFN) from intermediate nodes to distinguish route failure and network congestion. After receiving an RFN, TCP enters into the "snooze state". In this state, TCP stops sending packets and freezes all its variables such as timers and *cwnd* size. Upon receiving a Route Re-establishment Notification (RRN), via the routing protocol, TCP knows the route is reestablished and leaves the frozen state and resumes transmission using the same variable states before the "snooze state". In addition, a route failure timer is used to prevent infinite wait for RRN messages. When a route failure timer expires, the TCP normal congestion control is invoked. The results in [3] show that TCP-Feedback performs significantly better than standard TCP when route reestablishment delay grows. This is mainly due to the reduction of the number of unnecessary packet retransmission/timer backoffs during the route failure interval. However, similar to the first approach, upon route re-establishment the TCP state reflects the conditions on the old route and not necessarily on the new route.

The Fixed Retransmission Timeout scheme [2] is based on the idea that a regular exponential backoff mechanism is unnecessary, because route disconnection should be treated as a transitory period. Fixed RTO disables the exponential backoff after two successive retransmissions due to expired RTO, assuming it is caused by route failures. TCP retransmits a data packet more frequently because the retransmit timer is fixed; this reduces the inactive period after a route is reestablished. In [2], significant improvement of throughput was achieved by the use of Fixed RTO. The article also studied the TCP selective and delayed acknowledgments options, which could only achieve marginal gains. As pointed out by the authors themselves, their approach is limited to pure wireless networks only.

In ATCP [9], to maintain compatibility with the standard TCP/IP protocol suite, a thin layer called Ad Hoc TCP is inserted between TCP and IP. This scheme is different from the above three approaches where standard TCP is modified. ATCP utilizes the ICMP protocol and the ECN (Explicit Congestion Notification) scheme to detect network partition and congestion respectively. The intermediate layer ATCP keeps track of the packets to and from the transport layer. The feedback from intermediate nodes are used to put the TCP sender into either a persist state, congestion control state, or retransmit state. When a "Destination Unreachable" ICMP message is received, indicating route failure happened, the TCP sender enters a "persist state" which ends when the connection is reestablished. When three duplicate acknowledgements are received, indicating random errors, ATCP puts the TCP sender into "retransmit state" and quickly retransmits the lost packets from the TCP buffer. When an ECN message is received, which indicates real network congestion, ATCP puts the TCP sender into "congestion control state" and the TCP sender invokes the normal congestion control procedure. ATCP maintains end-to-end TCP semantics and is transparent to all nodes. Results in [9] show improvement of throughput under congestion, packet loss, and network partitions.

3.2. TCP Fairness Improvements

The above schemes address TCP's throughput problem. Several researchers have also studied TCP fairness. In [10], TCP unfairness among upstream and downstream flows is demonstrated and investigated. A gateway is used to forward traffic, and the buffer size in the gateway plays a key role in obtaining fair sharing of the medium among upstream and downstream flows. [10] shows via simulation that, when equal number of downstream and upstream flows exist, the average throughput ratio between the upstream and downstream flows can go up to 800. The reason is that upstream flows' ACKs clutter the gateway buffer and cause the buffer to overflow. Downstream flows experience timeouts and transmit only with a window of 0-2 packets because of the packet drops at the gateway buffer. Upstream flows normally can reach their

maximum window size. Because of the cumulative nature of TCP ACKs, small losses of ACKs do not affect the window size.

The proposed solution is to advertise the available buffer size to the sender. The gateway keeps the number of current TCP flows in the system. If the buffer size at the gateway is B and the number of flows is N, then the receiver window of all the TCP flows are set to the minimum of advertised receiver window or [B/N] by modifying the receiver window field of ACKs traversing the gateway. Through simulation and test bed implementation, this proposal shows a very good fairness, with the throughput ratio of upstream and downstream flows being 1 in the simulation and 1.007 in the test bed. The study is based on the assumption that all the losses happen in the gateway due to buffer overflow and all RTTs are the same among flows.

In [12], the TCP fairness problem in a combined wireless and wired network is investigated. The study shows that IN flows get significant more bandwidth than OUT flows. This unfairness is the joint result of the MAC layer's exposed nodes and hidden nodes problem and TCP's timeout and backoff schemes (see Section 2). In a study performed on the test bed, it is found that when the maximum congestion window size is smaller than a certain value (8 in the test), the two flows share the bandwidth fairly and the aggregate throughput reaches the upper limit. The problem is that this window size could not be preconfigured. A similar study is conducted in a pure Ad Hoc network, and the optimal congestion window size is found to be 1-2 packets. For connections with a long propagation delay, such a small window size will affect the efficiency.

To improve fairness over a combined wired and Ad Hoc network, a non-work-conserving scheduling algorithm working with IEEE 802.11 MAC is proposed in [13]. In the proposal, the normal FIFO work-conserving scheduling scheme is replaced, which treats routing packets (generated by routing protocols) as high priority packets over data packets (generated by applications), and puts the high priority packets in the queue before all data packets upon arrival. The head of the queue is send to the MAC after knowing that the MAC is ready to send another packet. A timer is set after a data packet is sent to the MAC. Only after the timer expires can the queue send another data packet. The routing packets have high priority and dequeue immediately after knowing that the MAC is ready. No timer is set after a routing packet is sent. The duration of the timer is based on the queue output rate and is the sum of three parts: transmission delay without contention; transmission delay based on recent queue output (choosing from four predefined values based on the queue output rate); and a random value uniformly distributed from zero to the value of the second part. The timer adds extra adaptive delay in the scheduling, so the more aggressively a node is sending packet, the more severely it is penalized, thereby nodes failing to grab the medium can compete with the fast sending nodes now.

Through simulations, [13] shows that the severe unfairness among flows can be eliminated while the aggregate throughput experiences a small degradation. Also, the maximum congestion window size does not adversely impact fairness in this scheme, so unlike the previous schemes there is no need to pre-configure the maximum congestion window size or to modify the advertised receiver window.

In summary, a range of proposals have addressed how to improve TCP throughput and to increase TCP fairness, with varying degrees of success. However, these approaches are all limited by their intent to keep at least the TCP semantics unchanged, if not the TCP implementations at each node, often resulting in improvement in one aspect (such as throughput) while trading off another aspect (such as fairness). Also, some of the proposals are only applicable in pure Wireless Ad Hoc Networks. However, we believe that the more relevant network architecture are Wireless Ad Hoc Access Networks. Finally, none of these proposals will address the congestion control problem for streaming UDP flows in such networks. In the next section, we discuss a new transport layer protocol that improves on both fairness and throughput (compared to TCP), and can be suitable for both reliable data transfer and streaming media flows.

4. Proposed Congestion Control Scheme

Datagram Congestion Control Protocol (DCCP) [8] is a new protocol designed for applications that require the flow-based semantics of TCP, but prefer timely delivery to in-order delivery, or a congestion control mechanism different from what TCP provides. DCCP aims to be a minimal overhead and general-purpose transport-layer protocol providing only two core functions: The establishment, maintenance and teardown of an unreliable packet flow, and Congestion control of that packet flow.

The purpose of DCCP is to provide a standard way to implement congestion control and congestion control negotiations for special applications. Our proposed

protocol utilizes DCCP with the congestion control mechanism specified in a new Congestion Control Identifier (CCID). We also added an optional ACK-based reliability layer on top of the DCCP connection, similar to TCP's reliability scheme. The new CCID profile defines when acknowledgments are sent and how to identify the true reasons of packet loss. Additional ECN support and ELFN support is used to provide network-detected information to the sender.

In our protocol, the sender has four states: *Normal State*, *Congestion State*, *Failure State* (route change or link failure) and *Error State* (transmission error). Rate-based congestion control is used to avoid the frequent slow starts. The most important task is to design the rate equation for each state, which is the key for throughput and fairness.

In the research of ATP [14], the packet queuing and sending delay at each node is calculated and the maximum delay is recorded in each packet. The receiver then calculates the rate based on the delay information and feeds it back to the sender. This approach was studied for a standalone Wireless Ad Hoc network without an access point or gateway connecting to the wired networks. When cooperating with a wired network, the relationships between the delay and rate are different in the wired and wireless parts, so the receiver cannot make decisions without knowing where the maximum delay happened. Also, intermediate nodes are working as routers, which process packets up to the network layer. To record the delay information at each node through the path and later to be used at the receiver for transport layer, additional effort is needed to make changes at the intermediate nodes. So, the ATP approach is excluded from our solution.

To determine the available end-to-end bandwidth, we adopted the delay based rate estimation mechanism in FAST TCP [7]. The sender maintains two RTT values, one is base RTT (*baseRTT*), which is the minimum recorded RTT, and the other is exponentially averaged RTT (*avgRTT*). Each time the sender goes into the failure state, the *baseRTT* will be reset by the round trip time of a probe packet and its corresponding acknowledgment, after being temporarily saved as old *baseRTT*. The sending rate after the route establishment is proportional to *baseRTT/oldbaseRTT*.

In the *Normal State*, the sender adjusts the rate proportional to *baseRTT/avgRTT*. In the C*ongestion State*, when ECN mark without packet loss happened, the rate adjustment is the same as in *Normal State*. But when packet loss happened, the sending rate will halve. This idea is based on FAST TCP for High-Speed Long-Distance Networks, which showed proportional fairness under no congestion or mild congested situations when packet loss occurs infrequently.

In *the Error State*, the rate is set to β*rate, calculated using the above equation, where β ranges from ½ to 1, according to the error rate.

In the *Failure State*, probe packets are send out to monitor the network situation. The rate of sending probe packets can be set to one packet per RTO like in Fixed RTO, but it should be studied further by experiments.

A simplified DCCP with rate-based congestion control is implemented based on the TCP implementation in NS2. Because wireless nodes do not support ECN and the limitation of getting network-detected link failure in NS2, the implementation has only two states: Congestion State and Normal State.

In the implementation, ACKs are sent back to the sender whenever the receiver receives a packet. ACKs have the ACK Vector option as specified in the DCCP specification. ACK vectors contain packet reception information (whether they are received, not received or ECN marked). Also, the ACK Vector can be used to return information about several packets to make sure the sender receives information though some ACKs may be lost. A weighted average RTT (3/4* RTT +1/4*current RTT) is calculated using the timestamp echo contained in the ACKs. The congestion window size (*cwnd*) is adjusted accordingly using the control equation. The function used in the simulation is:

$$cwnd = \{cwnd + (1 + (\text{int})(1 - cwnd * \frac{qdelay}{RTT}))\}$$

In the equation, *qdelay* is the difference between newly calculated weighted average RTT and *baseRTT*. When *cwnd* is 1, the equation will increase *cwnd* by 1 each RTT; when *qdelay* is zero, the *cwnd* is higher than 1, the equation will increase by 2 packets per RTT.

A timeout timer (RTO) is set for the transmitted packets. Since the test scenario is static, and no movement-caused packet drops are involved, the sender enters *Congestion State* whenever the timeout timer expires. In this *Congestion State*, probe packets with only headers are sent by the sender every RTO until an ACK is received. Upon successfully receiving an ACK, the sender resets the RTT and *baseRTT*, sets the congestion window size to *cwnd*oldBaseRTT/baseRTT*, and enters the *Normal State* again.

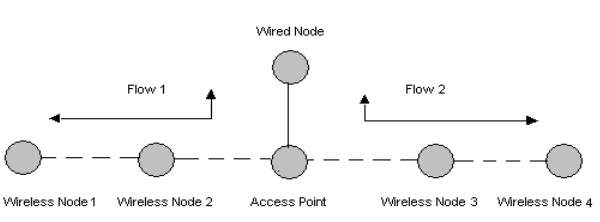

Figure 1: Wireless Ad Hoc Access Network Simulated Scenario

An alternative design is to reset the *cwnd* every RTT based on the average RTTs collected and keep the same *cwnd* for this RTT as in FAST TCP. This management of *cwnd* is similar to the approach in TCP, so it may provide a fairer sharing between DCCP and TCP flows in some cases. The advantages of the first design are that the unfairness caused by different RTTs between flows is removed, and it may be more suitable for the wireless situations where mobility is involved.

The test scenario used in the simulation is shown in Figure 1 (inspired by [12]). In the simulation, data is sent from the wired node to the wireless nodes (1 and 4) that are two hops away from the access point; or from those wireless nodes to the wired node via the access point. All wireless nodes are stationary in the simulation. All data flows are 10MB FTP flows. First, a single flow traversing a chain of nodes (4 hops) was tested using TCP Reno, our base protocol (called DCCP from now on), and the base protocol with an ACK-based retransmission scheme (called Reliable DCCP). Table 1 shows both throughput and goodput for each protocol.

In a second set of experiments, we simulated two flows, similar to the scenario in Figure 1. Each flow could be either an IN flow or an OUT flow, and we also varied the RTT for mixed flow scenarios. Table 2 shows the results for TCP as transport protocol and Table 3 shows results for the same scenarios using the DCCP-based protocol without reliable packet delivery. To evaluate various alternatives, we measured the throughput for each flow and the aggregate throughput, and we also measured inter-flow fairness using Jain's fairness index. An index value of 1 indicates a perfectly fair bandwidth allocation (i.e., each flow obtained the same normalized bandwidth share), as we typically assume that each flow has equal rights to the network resources.

Compared with TCP's congestion control, our proposed congestion control algorithm shows improved inter-flow fairness for all combinations of IN and OUT flows. The aggregate throughputs are also higher, but then again this version of DCCP does not provide reliable data delivery. In the current Internet, TCP is widely used, so it is also important to study the inter-protocol fairness between DCCP flows and TCP flows. These comparisons are summarized in Table 4.

DCCP and TCP flows share the network in a fair manner when they are both IN flows. There is some unfairness when there are OUT flows and the most unfair bandwidth sharing happens when the TCP and DCCP flows are both OUT flows. In the simulations for mixed flows, when the TCP flow is the OUT flow, the unfairness is also rather severe. The reason is that a DCCP flow is more aggressive in obtaining bandwidth compared with a TCP flow. This behavior also shows up when TCP or DCCP flows coexist with UDP flows, though for brevity, these results are not shown.

	Total Pkts	**Received Pkts**	**Throughput**	**Goodput**
TCP	6667	6912	0.447Mbps	0.444Mbps
DCCP	6667	5871	0.829Mbps	0.829Mbps
Reliable DCCP	6667	8801	0.765Mbps	0.629Mbps

Table 1: Single Flow Performances

Throughput (Mbps)	**Both IN (110ms)**	**Both OUT (110ms)**	**IN/OUT (110ms)**	**IN/OUT (60ms)**
Flow 1	0.65	0.84	1.12	1.37
Flow 2	0.76	0.33	0.10	0.02
Sum	1.41	1.17	1.21	1.38
Jain's Fairness Index	0.994	0.840	0.585	0.514

Table 2: TCP Performance for 2 Flows: Throughputs and Fairness

Throughput (Mbps)	Both IN (110ms)	Both OUT (110ms)	IN/OUT (110ms)	IN/OUT (60ms)
Flow 1	0.85	0.87	0.97	0.93
Flow 2	0.80	0.87	0.77	0.80
Sum	1.66	1.74	1.73	1.73
Jain's Fairness Index	0.999	1.000	0.987	0.994

Table 3: DCCP Performance for 2 flows: Throughput and Fairness

Throughput (Mbps)	Both IN (110ms)	Both OUT (110ms)	IN/OUT (DCCP IN)	IN/OUT (TCP IN)
DCCP Flow	0.764	1.675	1.443	0.699
TCP Flow	0.747	0.071	0.118	0.843
Sum	1.511	1.747	1.561	1.541
Jain's Fairness Index	1.000	0.542	0.581	0.991

Table 4: DCCP/TCP Inter-flow Throughputs and Fairness

Overall, the simulation results show that DCCP flows have good inter-flow fairness due to the modified congestion control algorithm, which uses a rate based window control algorithm based on the feedback from the acknowledgments. When DCCP flows coexist with TCP flows, DCCP flows starve the TCP flows only when there is an OUT TCP flow, and DCCP flows have better throughput when co-existing with UDP flows. The results show that the proposed congestion control algorithm is promising: flows using the proposed congestion control algorithm share the bandwidth almost fairly regardless of where the senders are (i.e., whether they are IN or OUT flows). When co-existing with TCP flows, the bandwidth sharing shows similar fairness issues as pure TCP flows, and the unfairness is somewhat more severe in these cases. This implies that the TCP-friendliness of the proposed congestion control algorithm should be further studied and improved.

Although Datagram Congestion Control Protocol (DCCP) is designed for applications which do not need reliability, it has features which can be used to implement reliable transmission based on DCCP, such as a sequence number for each DCCP Request or DCCP response packet, a checksum field, which uses the same algorithm as TCP's checksum algorithm, and an ACK option that provides packet loss and corruption information to DCCP senders.

Throughput (Mbps)	Both IN (110ms)	Both OUT (110ms)	IN/OUT (110ms)	IN/OUT (60ms)
Flow 1	0.71	0.75	0.72	0.84
Flow 2	0.71	0.79	0.77	0.69
Sum	1.42	1.53	1.49	1.53
Jain's Fairness Index	1.000	0.999	0.999	0.990

Table 5: Reliable DCCP Performance for 2 Flows: Throughputs and Fairness

Both DCCP and TCP are end-to-end sliding window protocols. Data packets are transmitted in both directions: packets are sent from the senders to the receivers and acknowledgements are sent from the receivers to the senders. Senders are allowed to send a window of packets before receiving the acknowledgment. This window starts at a constant size and is later controlled by the congestion control algorithms implemented in the protocols. Acknowledgments are valid when sequence numbers of acknowledged packets are within the range of the current window. To implement reliable transmission based on DCCP and provide a comparable level of reliability as TCP does, we added the following functions to DCCP: Buffering of received packets at the receivers, retransmission of lost or corrupted packets by the senders, detection and deletion of duplicated packets at the receivers, and in-order delivery of received packets to the application program at the receivers.

The results for this implementation of Reliable DCCP flows shows that they achieve better throughputs

and fairness, compared to Table 2. While TCP shows severe unfairness when the two flows are mixed (see Table 2), Jain's fairness indexes in Table 5 are all close to 1. At the same time, the aggregate throughputs of the two flows are higher as well. Mixing TCP and Reliable DCCP flows shows similar fairness results as the combination of TCP and DCCP flows presented in Table 4.

5. Conclusions and Future Work

TCP was designed for wired networks, and has benefited from substantial research efforts over the years. Yet it shows poor performance over multihop wireless networks and severe inter-flow fairness challenges, as shown in Section 2. Section 3 reviews a number of proposals to enhance TCP, with some of the proposed protocols showing quite promising results. However, none of these improvements will benefit UDP streams that are often used in streaming media-content. Section 4 gives a high-level overview of a congestion-control approach based on DCCP that could be beneficial to both unreliable data streams and reliable data transfers. Simulation results in NS2 confirm that the approach improves both fairness and aggregate throughput, providing users with fair and high-throughput access to the shared multihop wireless access network.

The work presented here will be further extended in the following areas to verify and improve the design. We will conduct more performance runs to verify the test results under multiple-flow scenarios. We will also add node mobility to the simulations and study the impact of additional loss scenarios caused by broken links during an active flow on throughput and fairness. We will also study and improve throughput and fairness when mixed Reliable DCCP and TCP flows co-exist. The core congestion control protocol can be further optimized by tuning the rate control formula and retransmission timer to optimize the packet sending rate and adding new features to the implementation in the simulation such as support of ECN, to provide additional information for the sender to identify network condition and to adjust the sending rate accordingly.

Finally, we are very interested in implementing the proposed congestion control protocol in our wireless mesh test bed to verify the simulation results.

Acknowledgements

The authors would like to thank NSERC for its financial support of this work.

References

[1] M. Allman, V. Paxson and W. Stevens, "TCP Congestion Control", RFC 2581, IETF, April 1999.

[2] V. Anantharaman and R. Sivakumar, "TCP Performance over Mobile Ad Hoc Networks – a Quantitative Study", Wireless Communication and Wireless Networks, pp. 203 – 222, 2003.

[3] K. Chandran et al. "A Feedback Based Scheme For Improving TCP Performance In Ad-Hoc Wireless Networks", Proc. of Int. Conf. on Distr. Comp. Systems, pp. 472-479, Amsterdam, Netherlands, 1998.

[4] S. Corson and J. Macker, "Mobile Ad Hoc Networking [MANET]: Routing Protocol Performance Issues and Evaluation Considerations", RFC 2501, IETF, Jan. 1999.

[5] G. Holland and N. Vaidya, "Analysis of TCP Performance over Mobile Ad Hoc Networks", Proc. 5th ACM/IEEE Int. Conf. on Mobile Comp. and Networking, pp. 219 – 230, Seattle, USA, 1999.

[6] R. Jain, D. Chiu, and W. Hawe, "A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems", Technical Report, DEC Research Report TR-301, Digital Equipment Corporation, Hudson MA, USA, Sept. 1984.

[7] C. Jin, D. Wei, S. H. Low, "FAST TCP: Motivation, Architecture, Algorithms, Performance", Proc. of the 23rd Conf. of the IEEE Communication Society, pp. 81-94, Hong Kong, China, March 2004.

[8] E. Kohler, M. Handley and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, IETF, March 2006.

[9] J. Liu, S, Singh, "ATCP: TCP for Mobile Ad Hoc Networks", IEEE Journal on Selected Areas in Communication, 19(7):1300 – 1315, July 2001.

[10] S. Pilisof, R. Ramjee, D. Raz, "Understanding TCP Fairness Over Wireless LAN", Proc. of the 22nd Annual Joint Conf. of IEEE Computer and Communications Societies, pp. 863 – 872, April 2003.

[11] F. Wang and Y. Zhang, "Improving TCP Performance over Mobile Ad-Hoc Networks with Out-of-Order Detection and Response", Proc. of 3rd ACM Int. Symposium on Mobile Ad Hoc Networking & Computing, pp. 217 -225, Lausanne, Switzerland, June 2002.

[12] K. Xu et al., "TCP Behavior across Multihop Wireless Networks and the Wired Internet", Proc. of the 5th Int. Workshop on Wireless Mobile Multimedia, pp. 207 – 218, Seattle, USA, Sept. 2002.

[13] L. Yang et al., "Improving Fairness among TCP Flows crossing Wireless Ad Hoc and Wired Networks", Proc. of the 4th ACM Int. Symposium on Mobile Ad Hoc Networking & Computing, pp. 57 – 63, Annapolis, USA, 2003.

[14] K. Sundaresan and V. Anantharaman, "ATP: A reliable Transport Protocol for Ad-Hoc Networks", IEEE Transactions on Mobile Computing, volume 4, issue 6, pages 588 – 603, Nov/Dec, 2005.