WXCP: Explicit Congestion Control for Wireless Multi-Hop Networks

Yang Su and Thomas Gross

Department of Computer Science, ETH Zurich
Zurich Switzerland

Abstract. TCP experiences serious performance degradation in wireless multi-hop networks with its probe-based, loss-driven congestion control scheme. We describe the Wireless eXplicit Congestion control Protocol (WXCP), a new explicit flow control protocol for wireless multi-hop networks based on XCP. We highlight the approaches taken by WXCP to address the difficulties faced by the current TCP implementation in wireless multi-hop networks. Simulations with ns-2 show that WXCP outperforms current TCP implementations in terms of efficiency and fairness.

1 Introduction

With the progress in wireless technology, wireless networks become a potential candidate for constructing a broadband wireless backbone to provide ubiquitous low cost Internet connection [11, 2]. We need an efficient transport protocol for this new network architecture that should also deal with multi-hop networks and their variants (e.g., meshes).

Currently, the most widely-deployed TCP implementation is TCP Reno and its variations. However, recent studies have shown that TCP Reno suffers fairness and efficiency problems in wireless ad hoc network environment [9, 16, 8, 18]. There are several reasons for those problems.

1. TCP couples congestion control with reliability control. It detects congestion by packet loss events. Packet loss is strongly correlated to congestion in wireline networks but not a reliable congestion signal in wireless networks, where packet loss can also be introduced by medium related errors [6] and mobility related routing failures [9].
2. TCP relies on the AIMD (Additive-Increase Multiplicative-Decrease) adjustment of its congestion window to converge to a fair sharing of network bandwidth. It cannot acquire spare bandwidth efficiently after re-routing events [16, 5].
3. The throughput of multi-hop wireless networks is highly dependent on the traffic load. When traffic load increases over some threshold, the link error rate increases, and throughput drops down. TCP's flow control aims to fill the bottleneck interface queue and often puts too many packets into the network. [13, 8].

We attempt to improve the performance of the transport protocol in wireless multi-hop networks by designing a new congestion control scheme that tackles the second and third problems. Our solution adopts an explicit congestion control architecture where intermediate stations make congestion estimations and send rate feedback to senders. The senders adjust their transmit rate based on the received rate feedback. This protocol called WXCP (Wireless eXplicit Congestion Control Protocol) is inspired by XCP (eXplicit Congestion control Protocol) [12], a window-based explicit congestion control scheme designed for high bandwidth-delay networks. We integrate a number of mechanisms, both at intermediate stations and the sender, to adopt XCP to the wireless network environments. At intermediate stations, WXCP makes more precise estimation of congestion conditions than current approaches and computes the rate feedback based on multiple congestion metrics. By using explicit rate feedback instead of probing the available bandwidth, WXCP flows are able to converge to a transmission state where better throughput is achieved. At the same time, WXCP flows converge to the equilibrium more quickly than TCP. In addition, loss discovery and pacing mechanisms are introduced at the sender to deal with the tiny window and burst problem.

The rest of the paper is organized as follows. Section 2 discusses the related work. In Section 3, we describe the design of WXCP in details. Section 4 contains an evaluation of the protocol. We conclude the paper in Section 5.

2 Related Work

WXCP is an extension of XCP, which was developed for high bandwidth-delay product networks [12].

Many research contributions address improving TCP performance or design new schemes for reliable data transmission over multi-hop wireless networks.

1. A first group of approaches is based on end-to-end measurement. Wang and Zhang [17] explore the approach to improve TCP performance by detecting and responding to out-of-order packet delivery events. Fu et al. [7] present a TCP-friendly transport protocol that tries to distinguish events such as mobility-introduced disconnection, reconnection, high out-of-order delivery ratio, or channel error from network congestion by performing multi-metric joint identification for packet and connection behaviors based on end-to-end measurements.

2. Several researchers have attempted to improve the performance of reliable data transmission in wireless networks by using explicit feedback from the routing layer. Holland and Vaidy [9] investigates the effects of link breakage due to mobility on TCP performance and propose an explicit link failure notification technique (ELFN). Chandran et al. [4] present a similiar mechanism called TCP-Feedback. Liu and Singh [15] introduce a thin layer between transport and underlying routing layers which deals with explicit notifications from intermediate stations and shields TCP from the underlying behavior of an ad hoc network.

These approaches [15, 17, 7, 4, 9] improve the performance of TCP by distinguishing *congestion related* packet losses from packet losses caused by *medium errors* or *routing failures*. They solve the first problem listed in Section 1.

3. Recently, researchers also proposed to use explicit rate feedback from intermediate stations to deal with the second and third problems in Section 1. Sundaresan et al. [16] present a rate-based transport layer protocol called ATP. In ATP, the sender adjusts its transmission rate based on explicit rate feedback from the bottleneck station. Chen et al. introduce EXACT (EXplicit rAte-based flow ConTrol), which adapts the rate-based feedback framework of ATM's ABR (Available Bit Rate) congestion control to ad hoc networks [5]. Our scheme, which shares the same target and also makes use of explicit rate feedback, differs from these approaches in that we deploy window-based congestion control instead of pure rate-based congestion control at the sender. In contrast to [5], intermediate stations in WXCP do not maintain per flow information. Therefore implementation and deployment of WXCP are much simplified. [16] also does not maintain per-flow information, but it does not take into account the spatial characteristics of the wireless medium, whereas WXCP maintains time fairness instead of throughput fairness among flows to different next hop stations. Consequently WXCP achieves an higher aggregate throughput.

3 Explicit Congestion Control Protocol for Wireless

3.1 Motivation

In this paper, we make two assumptions about the MAC protocol: 1. It is based on CSMA/CA. 2. Unicast packets are acknowledged. As shown in [19], it is challenging to apply XCP to shared media wireless networks. To accurately calculate feedback, the XCP router must know the precise link capacity in advance. However, in shared media wireless networks, all the stations contend for the media. The true output capacity is changing depending on the contention traffic load. XCP takes the link capacity at interface to compute the rate feedback. That introduces capacity overestimation with which XCP will generate inflated feedback, the senders will send more than the link can transfer, and the queue will build up. Instead of using a fixed interface capacity, WXCP estimates how much capacity that it has fair access to by locally monitoring the channel conditions at intermediate stations. In addition, at the sender, loss discovery and pacing mechanisms deal with the tiny window and burst problem.

In the rest of this section, we describe congestion metrics used to estimate the available capacity and calculate the rate feedback. Then we present the design of WXCP for intermediate stations and for the sender.

3.2 Congestion Metrics

WXCP uses three metrics to measure the state of resource usage and the level of congestion at a station:*available bandwidth, interface queue (IFQ) length and average link layer retransmission (ALR).*

We use available bandwidth to represent how much network capacity is still available. The less bandwidth is available, the more probable it is that congestion will happen. Available bandwidth can be estimated based on local observation without exchange of additional control packets. If the estimation is made periodically, channel free time represents network capacity still available during the estimation period. To convert channel free time to a rate, we need the link layer throughput. Since the wireless medium condition at different locations might be different, link layer throughput to different neighboring stations might also be different. Hence, although the same channel is used, the available bandwidth estimation to different destination stations might be different. The available bandwidth we use in WXCP is the average available bandwidth of all the paths.

If the estimation period is T, average available bandwidth B is:

$$B = \frac{T_{free} \cdot bw}{T} \tag{1}$$

where T_{free} is channel free time during period T; bw is the average link layer throughput to all the different destinations. The model can be implemented with the IEEE 802.11 DCF MAC protocol, where CSMA/CA mechanism is used to control multiple stations visiting the same channel. By monitoring the radio state, we can get T_{busy}, which is the sum of "time used by station itself", "physical carrier sense time", and "virtual carrier sense time" during the observation interval T. Then, T_{free} can be computed as $T - T_{busy}$. In IEEE 802.11 DCF, any non-broadcasting data packet is always companied with an acknowledgment packet. bw is an average of each link layer throughput measurement sample, which is computed as:

$$\frac{s_j}{t_t - t_r} \tag{2}$$

where s_j is the size of packet j, t_t is the time when the packet is delivered to the MAC layer, t_r is the time when the corresponding ACK packet is received.

The second metric is the state of the output interface queue (IFQ). When the input traffic rate is greater than the output rate, packets start to be buffered in IFQ and the length of the queue increases. When the queue is full, further packets coming to the queue are dropped. TCP uses this event to infer the existence of congestion in the network.

Because of the hidden terminal problem [1], without any coordination, sender contends for a channel around the receiver against stations out of its sensing range, but still in the receiver's sensing range. If the hidden traffic comes from the flow itself, it is the well-known self-interference. When a flow puts too many packets in the network, self-interference happens, transmission delay increases and throughput drops down. By adjusting the transmission rate, a flow can change the degree of self-interference. However the length of a sender's IFQ is

not sensitive enough to detect this condition [20]. Hence, we use the average link layer retransmission (ALR) from successfully transmitted packets as the third congestion metric to detect the degree of self-interference.

There are many noise sources for using ALR to sense self-interference, because packet losses in wireless ad hoc networks are caused not only by self-interference but also by other kinds of wireless medium errors (such as interference due to multi-path reflection and signals from other kind of sources, attenuation and path dispersion[21, 6]) and route failures. It is difficult to distinguish self-interference from other kinds of wireless medium errors. It is possible that when network experiences severe wireless medium errors, a station gets a number of high re-transmissions and hence infers the existence of a high degree of self-interference, but in fact, there is no self-interference at all. In this case, a transmission protocol that uses this metric becomes too conservative. WXCP deals with this problem by following the convention of TCP implementations that keep the minimum CWND greater or equal to one packet. From Section 3.4 and Section 4, we see that a CWND of one packet is close to the optimal value for most network settings. In the worst case, when severe wireless medium errors happen and WXCP senses the self-interference incorrectly, it can at least work with a CWND of one and achieve reasonable performance.

3.3 Explicit Congestion Feedback Computation in WXCP

In WXCP, intermediate stations make congestion control and fairness control decisions separately, based on flow information carried in data packet headers and the estimation of congestion metrics described in the last section. Because in wireless networks, link layer throughputs over different paths are different, to achieve higher overall throughput, the WXCP fairness controller maintains time fairness instead of throughput fairness among flows. Every control interval T, which is the average of RTT of all the flows over this station, WXCP calculates the aggregate feedback:

$$\Phi = \alpha \cdot \frac{T \cdot B}{n+1} - \beta \cdot Q_{ifq} - \zeta \cdot Retry_{avg} \qquad (3)$$

where B is the estimation of available bandwidth that is shared by n neighboring stations and the station itself. n can be obtained by counting the number of different sources from packets overheard during the last control interval. Q_{ifq} represents the minimal length of interface queue observed during the control interval. $Retry_{avg}$ represents the average ALR from successfully transmitted packets over all the destination stations during last control interval. α, β, ζ are constants.

When aggregate feedback is positive, we want to increase the active time of all flows by the same amount. Thus Δt_i, the change of active time of any flow i, will be proportional to the same constant. Assume that flow i flows over path k with link layer throughput bw_k. Since Δt_i is proportional to the change in throughput of the flow $\Delta throughput_i$, and inversely proportional to link throughput bw_k

(i.e.,$\Delta t_i = \frac{\Delta throughput_i \times T}{bw_k}$), the change of throughput of flow i is proportional to the link layer throughput of the path. In addition, the change in congestion window of flow i is the change in its throughput multiplied by its RTT. Hence, the change in the congestion window of flow i should be proportional to the flow's RTT and link layer throuthput,(i.e., $\Delta cwnd_i \propto rtt_i \times bw_k$).

The total change in congestion window of a flow is the sum of the per-packet feedback it receives. The expected number of packets from flow i seen by the router in a control interval T is proportional to the congestion window of the flow $cwnd_i$ and inversely proportionally to its round trip time rtt_i and packet size s_i (i.e.,$\frac{T}{rtt_i} \times \frac{cwnd_i}{s_i}$). Thus, per-packet positive feedback p_i is given by:

$$p_i = \xi_p \cdot bw_k \cdot \frac{rtt_i^2 \cdot s_i}{cwnd_i} \tag{4}$$

where ξ_p is a constant. Since the total increase of the aggregate traffic rate is equal to the sum of the increase in the rate of all flows in the aggregate, ξ_p can be derived as:

$$\xi_p = \frac{\Phi}{T \cdot \sum^L bw_k \cdot \frac{rtt_i \cdot s_i}{cwnd_i}} \tag{5}$$

where L presents all the flows over the station. Similarly, we compute the per-packet negative feedback when the aggregate feedback is negative. In this case, we want the decrease in the active time of flow i to be proportional to its current active time (i.e.,$\Delta t_i \propto t_i$). At the same time, t_i is proportional to $throughput_i$ and inversely proportional to bw_k. Δt_i is proportional to $\Delta throughput_i$ and inversely proportional to bw_k. It can be derived that the decrease in the throughput of flow i is proportional to its current throughput (i.e., $\Delta throughput_i \propto throughput_i$), which is the same as XCP. From XCP [12], per-packet negative feedback n_i is given by:

$$n_i = \xi_n \cdot rtt_i \cdot s_i \tag{6}$$

Constant ξ_n can be derived as:

$$\xi_n = \frac{\Phi}{T \cdot \sum^L s_i} \tag{7}$$

3.4 Loss Discovery

As with XCP, a WXCP sender maintains a congestion window of the outstanding packets, $cwnd$, and an estimate of the round trip time, rtt. On packet departure, the sender fills its current $cwnd$ and rtt into the packet header. Whenever a new acknowledgment arrives, the sender adjusts its $cwnd$ according to $feedback$ contained in the acknowledgment:
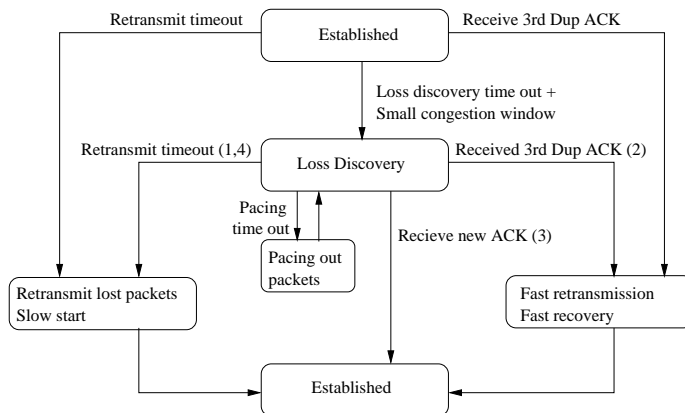
$$cwnd = max(cwnd + feedback, s) \tag{8}$$

Fig. 1. State machine for WXCP sender.

where s is the packet size. In addition to feedback, WXCP also responds to losses in the similar manner to TCP.

[14, 3] show that when TCP works with a small congestion window the window size limits the number of returning ACKs the sender may receive. Because TCP requires three duplicate ACKs to trigger fast retransmission, small windows may prevent these algorithm from being effective.

WXCP keeps the congestion window with a size close to the optimal value where better throughput is achieved. As shown in [8], the optimal window size for window-based flow control over multi-hop wireless ad hoc networks remains a small value (e.g., smaller than 5) in most of network settings. To avoid unnecessary timeouts, a WXCP sender needs to put enough packets into the network to make a decision about current packet loss pattern. Hence we introduce loss discovery state into sender side WXCP state machine. The basic idea is that when the congestion window is small, if there is no new ACK received and there are not enough duplicated ACKs returned, instead of waiting for a retransmission timeout, the sender switches from window-based control to rate-based control with reduced transmission rate. As shown in Figure 1, two timers, loss discovery timer and pacing timer, are added to sender side implementation. Every time when the retransmission timer is set, the loss discovery timer is also set with timeout period equal to current smoothed RTT estimation. If there is no packet lost, and RTT over the path does not increase dramatically, the loss discovery timer will not time out before it is reset. Otherwise, when a loss recovery timeout happens, the current congestion window size is checked. If the congestion window is smaller than the threshold W_{min}, the sender enters the loss discovery state. In loss discovery state, packets are paced out with a rate that is half to the current transmission rate (e.g., $\frac{cwnd}{2 \cdot RTT}$). The sender keeps reducing the transmission rate to half when it paces out a CWND of packets. During loss discovery period, CWND remains unchanged. RTT estimation is updated as normal. As

shown in Figure 1, the sender exits the loss discovery state under the following conditions:

1. There have already been W_{min} packets inserted in the network but still no new ACKs returned, and the number of duplicated ACKs is less than 3. Now it is reasonable to infer that most of the packets are lost and the network is in congestion. The sender will stop injecting new packets into the network and wait for a retransmission timeout.
2. Sender receives 3 duplicated ACKs. This situation indicates that there are packets lost. Sender quits loss discovery state and starts fast retransmission and fast recovery.
3. Sender receives new ACKs. There are no packets lost and there is no congestion present in the network.
4. Retransmission timer expires. If RTT of the path is large, it is possible that the retransmission timer times out before packets in the network are increased to W_{min}. In this case, the sender enters retransmission timeout state and retransmits lost packets in the same manner to TCP.

3.5   Pacing

Window-based transmit schemes have the advantage that they make use of the ACK self-clock instead of relying on a fine grained timer. They also have the drawback that they introduce bursts when many ACKs are received and the sender sends out back-to-back packets. Those packets would interfere with each other over the wireless medium. To smooth this burst, a pacing mechanism is introduced into the window-based WXCP sender. We define B_t as maximum tolerable burst. The WXCP sender shapes its outgoing traffic by B_t. Normally, when it has available congestion window space W_a, the sender tries to send as much packets as W_a allows. In WXCP, the sender checks B_t. If $W_a < B_t$, packets are sent out as usual, otherwise, packets belonging to one congestion window are paced out with rate $\frac{cwnd}{rtt}$, where $cwnd$ is current congestion window, rtt represents the last RTT sample. As soon as one of the acknowledgments for paced out packets is received, the pacing phase ends. Further packets are sent out with ACK self clocking.

3.6   Discussion of Parameter Settings

1. **Loss discovery threshold W_{min}:** When its congestion window reaches this threshold, the WXCP connection is considered to be resilient to timeout and always works with window-based rate control. When its congestion window is below the threshold, the WXCP connection operates in the loss discovery state to reduce unnecessary timeout. In this paper, we set the threshold to be 7 packets.
2. **Maximum tolerable burst B_t for pacing:** B_t represents the burst that a WXCP connection wants to tolerate in the network. To minimize the self-contention, we set its value to 2.

3. **Parameters for intermediate stations:** There are three parameters concerning the algorithms at intermediate stations. α controls the allocation of available bandwidth to WXCP flows. Its value is between 0 and 1. With large α, WXCP increases its rate quickly to fill the available bandwidth. To reduce the transmission delay, WXCP tries to keep both short IFQ and low self-interference at intermediate stations, which are controlled by β and ζ. Hence, these three parameters must be set to balance the increased protocol responsiveness to available bandwidth with the increased delay from queuing at IFQ and self-interference. Currently we set α, β, ζ to 0.20, 0.11 and 67. A detailed discussion of parameters setting can be found in [20].

4 Evaluation

4.1 Experimental Setup

All the experiments in this paper rely on simulation. Unless explicitly mentioned, all simulations use the configuration described here. The simulation platform that we use is the ns-2 simulator (version 2.27). We use TCP Newreno as the basis for comparison. In all the experiments, we use FTP as application traffic, AODV as routing protocol and at the MAC layer, the IEEE 802.11 DCF MAC protocol with RTS/CTS enabled. Channel bandwidth is 2Mbps. The effective transmission range is 250 meters, and the interference range is about 550 meters. The active simulation time is 120s.

4.2 Chain topology

To provide insight into the protocol's baseline behavior, we investigate the performance of WXCP with stationary chain topologies where stations are arranged in a chain and two adjacent stations are 200 meters apart. We repeat the experiments with the length of chain changing from 1 hop to 16 hops. Each simulation runs for 140 seconds. From 5s to 125s, an FTP traffic flows over the chain from the first station to the last station.

1. **Throughput.** We present overall throughput data over 120s simulation for TCP and WXCP with packet size of 1000 bytes in Figure 2. Each plot presents the average result of 10 independent simulations running with different random seeds. We observe that WXCP achieves the same throughput as TCP in short chain scenarios (less than 3 hops) and realizes a 10% to 25% throughput improvement over chains with medium length (3 to 7 hops). When it operates over long chains (more than 8 hops), WXCP can achieve 34% to 110% more throughput than TCP. Due to different per-packet overhead, the transport protocols achieve different throughput when using different packet sizes. However, as shown in [20], both protocols, TCP and WXCP, change their behavior in a similar way. Hence, in the following experiments, we only show the results when the packet size is 1000 bytes.
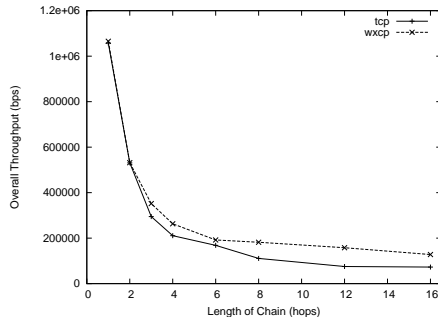
Fig. 2. Comparison of overall throughput of TCP and WXCP over chain topology.

Hops	TCP	WXCP	Ideal
1	120.6	2.0	1.0
2	86.7	1.9	1.0
3	31.0	1.8	1.0
4	7.3	1.8	1.0
6	7.9	2.0	1.5
8	8.3	3.0	2.0
12	8.8	5.0	3.0
16	9.7	6.7	4.0

Table 1. Average congestion window over chain topology.

2. **Congestion window**. From [8], we know that in multi-hop wireless networks, for window-based flow control, there exists a value for window size such that spatial channel reuse is maximized. This window size results in best channel utilization and consequently highest throughput. In the chain topology, optimal throughput is achieved when the window size is $h/4$ packets, h is number of hops, if the impact of ACK packets is omitted. The reason is that stations can only transmit concurrently with stations 4 hops away without interference. Table 1 shows the average congestion window of WXCP, TCP and $h/4$ for chains of different lengths. We observe that by estimating the channel condition more accurately, WXCP keeps the congestion window to a reasonable size, whereas TCP makes the estimation by using the AIMD adjustment of its congestion window. This behavior of TCP leads to too many packets in flight. These packets interfere with each other and finally hurt performance. For TCP over shorter paths (1 or 2 hops), chances for packets from one flow to interfere with each other is small. Hence, although TCP's CWND is far from the optimal value, it still can achieve a throughput close to WXCP, as shown in Figure 2. In Figure 3 we present snapshots of the instantaneous congestion window for TCP and WXCP in a 6-hop chain. The vibration of TCP's congestion window is due to its frequent packet loss. In contrast, WXCP shows a relatively stable behavior.

3. **Packet loss**. Figure 4(a) shows the number of packets dropped at the MAC layer; this number presents the degree of contention for the wireless medium. It includes all kinds of packets dropped at the MAC layer (MAC layer control packets, routing layer control packets, and data packets). Packet loss at the MAC layer does not always introduce packet loss observed by the transport protocol, because of the local error control mechanism of IEEE 802.11. We observe that in short chains there are fewer packets dropped at the MAC layer using WXCP than using TCP. This result shows that WXCP can keep the channel in a lower contention state while achieving a little better throughput than TCP. When the length of the chain increases, the number of packet losses at the MAC layer for WXCP approaches the number for
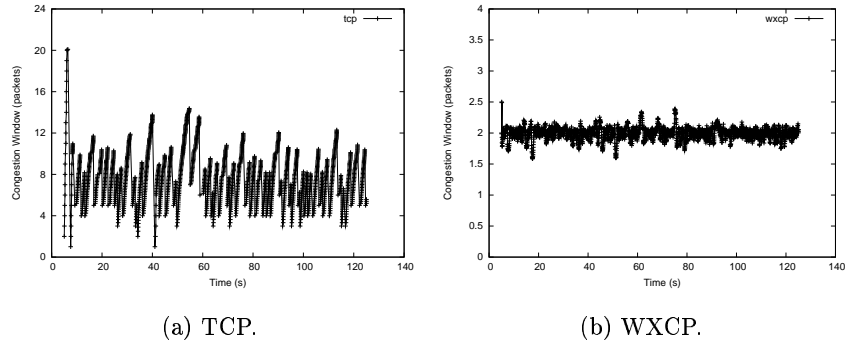
(a) TCP.   (b) WXCP.

Fig. 3. Instantaneous congestion window vs. time.

TCP. However, WXCP achieves much higher throughput than TCP. From Figure 4(b), which shows the number of packets lost at MAC layer per successfully transmitted packet, we conclude that WXCP maintains lower contention for the wireless medium than TCP.
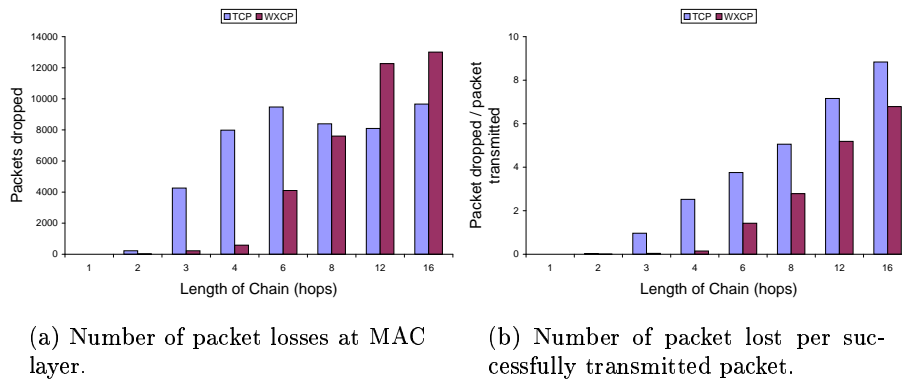

(a) Number of packet losses at MAC layer.   (b) Number of packet lost per successfully transmitted packet.

Fig. 4. Packet loss at MAC layer.

To study the interaction between WXCP flows, we create two WXCP flows from station 0 to the last station in the chain. The flows share the same path. These simulations are repeated with TCP flows in the same setting. Figure 5 shows the aggregate overall throughput. Depending on network configuration, WXCP achieves 1% to 97% improvement over TCP. At the same time, different flows sharing the same path are expected to obtain nearly the same throughput. In this paper, we use the coefficient of variation as fairness index; this value will

be 0 under fair allocation of resources [10]. Table 2 shows the comparison of fairness between TCP and WXCP. Each number represents the average coefficient of the variation of throughput. We observe that WXCP guarantees better fairness among two flows over networks with a chain topology than TCP.
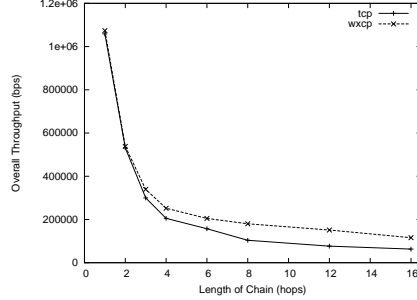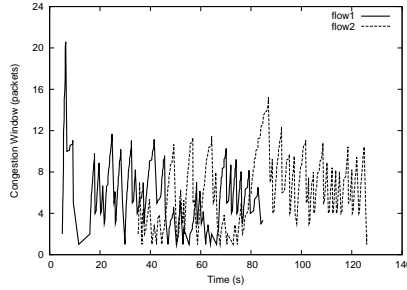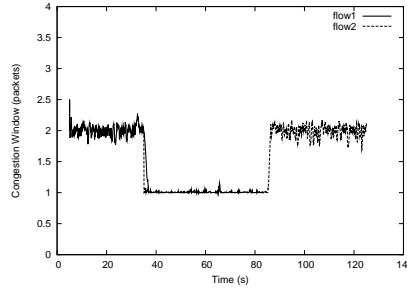
Fig. 5. Aggregate overall throughput of two flows over chain topology.

Hops	TCP	WXCP
1	0.0007	0.0005
2	0.0018	0.0004
3	0.8985	0.0003
4	0.1336	0.0009
6	0.1097	0.0004
8	0.1132	0.0157
12	0.1620	0.0368
16	0.1122	0.0281

Table 2. Average coefficient of variation of throughput over chain topology.

(a) TCP.   (b) WXCP.

Fig. 6. Instantaneous congestion window dynamics.

While the scenarios discussed thus far comprise static traffic loads, to investigate the performance of WXCP's rate adaptation mechanism in the event of network traffic dynamics, we conduct an experiment over a 6-hop chain with two flows $f1$ and $f2$. $f1$ exists in the interval between 5s and 85s, while $f2$ exits in the interval between 35s and 125s. The instantaneous congestion windows of the two WXCP flows are presented in Figure 6(b). We observe that after the arrival of $f2$, the two flows converge to the fair sharing of available channel capacity in a short time. After $f1$ leaves, $f2$ is able to catch up to the total available capacity again. The corresponding instantaneous congestion window dynamics for

two TCP flows is shown in Figure 6(a). It can be seen that when flow $f2$ joins, the two flows cannot converge to a stable fair sharing of bandwidth. Instead the congestion windows oscillate. When $f1$ leaves, $f2$ is unable to properly catch up to the available capacity.

4.3   Grid topology

To evaluate WXCP in more complex topologies, we create a stationary wireless ad hoc network whose nodes are arranged in a 13×13 grid topology as shown in Figure 7. For TCP and WXCP, we run 2, 4, 6, 8,10 and 12 flows respectively. In each of these cases, flows are spaced evenly in two directions (top-to-down and left-to-right). The aggregate throughput is summarized in Figure 8. In all cases, WXCP increases throughput by about 23% to 34% relative to TCP.
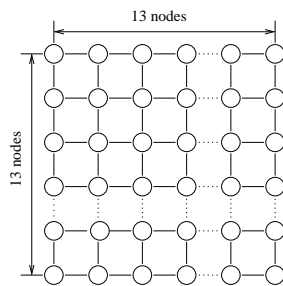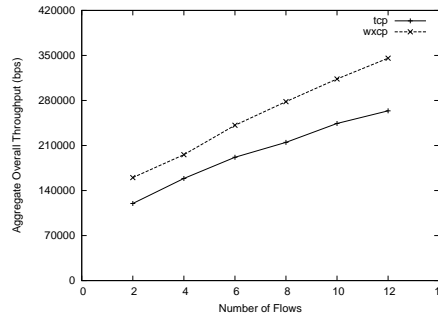

Fig. 7. Grid topology.

Fig. 8. Aggregate overall throughput vs. Number of flows over grid topology.

5   Concluding remarks

WXCP is an explicit congestion control protocol for wireless multi-hop or ad hoc networks. In WXCP, based on the estimation of multiple congestion metrics, intermediate stations maintain an estimate of congestion conditions and explicitly notify flows about their available bandwidth. As a result, WXCP flows are able to catch the available bandwidth quickly and precisely. WXCP also integrates a new loss recovery algorithm at the sender to deal with the potential small-window problem companied with window-based transfer. Simulation results show that WXCP outperforms TCP in terms of both efficiency and fairness.

References

1. Allen, D.: Hidden Terminal Problems in Wireless LAN. IEEE 802.11 working Group paper 802.11/93-xx

2. Bahl, V.: Self-Organizing Neighborhood Wireless Mesh Networks. Microsoft mesh network homepage

3. Balakrishnan, H. and Padmanabhan, V. and Seshan, S. and Stemm, M. and Katz, R.: TCP behavior of a busy internet server: Analysis and improvements. Proceedings of IEEE INFOCOM, 1998

4. Chandran, K. and Raghunathan, S. and Venkatesan, S. and Prakash, R.: A Feedback-based Scheme for Improving TCP Performance in Ad Hoc Wireless Networks. Proceedings of ICDCS, 1998

5. Chen, K. and Nahrstedt, K. and Vaidya, N.: The Utility of Explicit Rate-Based Flow Control in Mobile Ad Hoc Networks. Proceedings of IEEE WCNC, 2004

6. Eckhardt, D. and Steenkiste, P.: Measurement and Analysis of the Error Characteristics of an In-Building Wireless Network. Proceedings of ACM SIGCOMM, 1996

7. Fu, Z and Greenstein, B. and Meng, X. and Lu, S.: Design and Implementation of a TCP-Friendly Transport Protocol for Ad Hoc Wireless Networks. Proceedings of IEEE ICNP, 2002

8. Fu, Z. and Zerfos, P. and Luo, H. and Lu, S. and Zhang, L. and Gerla, M.: The Impact of Multi-hop Wireless Channel on TCP Throughput and Loss. Proceedings of IEEE INFOCOM, 2003

9. Holland, G. and Vaidya, N.: Analysis of TCP Performance over Mobile Ad Hoc Networks. Proceedings of ACM Mobicom, 1999

10. Jain, R. and Chiu, D. and Hawe, W.: A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer System. Technical Report TR-301, Digital Equipment Corporation, 1984

11. Karrer, R. and Sabharwal, A. and Knightly, E.: Enabling Large-scale Wireless Broadband: The Case for TAPs. Proceedings of the 2nd Workshop on Hot Topics in Networks (Hot-Nets II), 2003

12. Katabi, D. and Handley, M. and Rohrs, C.: Congestion Control for High Bandwidth-Delay Product Networks. Proceedings of ACM SIGCOMM, 2002

13. Li, J. and Blake, C. and Couto, D. and Morris, R.: Capacity of Ad Hoc Wireless Networks. Proceedings of ACM MOBICOM, 2001

14. Lin, D. and Kung, H.T.: TCP Fast Recovery Strategies: Analysis and Improvements, Proceedings of IEEE INFOCOM, 1998

15. Liu, J. and Singh, S.: ATCP: TCP for Mobile Ad Hoc Networks, IEEE Journal on Selected Areas in Communications, 2001

16. Sundaresan, K. and Anantharaman, V. and Hsieh, H-Y. and Sivakumar, R.: ATP: A Reliable Transport Protocol for Ad-hoc Networks, Proceedings of ACM MOBI-HOC, 2003

17. Wang, F. and Zhang, Y.: Improving TCP Performance over Mobile Ad-Hoc Networks with Out-of-Order Detection and Response, Proceedings of ACM MOBIHOC, 2002

18. Xu, K. and Gerla, M. and Qi, L. and Shu, Y.: Enhancing TCP Fairness in Ad Hoc Wireless Networks Using Neighborhood RED, Proceedings of ACM MOBICOM, 2003

19. Y.G. Zhang and Henderson,T: An Implementation and Experimental Study of the eXplicit Control Protocol (XCP), Proceedings of IEEE INFOCOM, 2005

20. Su, Y. and Gross, T.: WXCP: Explicit Congestion Control for Wireless Multi-Hop Networks, Technical Report, ETH Zurich, Feb.2005

21. Eckhardt, D. and Steenkiste, P.: Improving Wireless LAN Performance via Adaptive Local Error Control, Proceedings of IEEE ICNP, 1998