Detecting Spatial Congestion in Multihop Wireless Networks

Changhee Joo
School of EE and INMC
Seoul National University
Seoul, Korea
cjoo@netlab.snu.ac.kr

Saewoong Bahk
School of EE and INMC
Seoul National University
Seoul, Korea
sbahk@netlab.snu.ac.kr

Hyogon Kim
Department of Computer Science and Engineering
Korea University
Seoul, Korea
hyogon@korea.ac.kr

ABSTRACT

While TCP is highly successful in the wire-line Internet, its performance fast degrades as the number of hops increases in multihop wireless networks. It is due to not only the half-duplex nature of the wireless medium, but also the congestion spreading phenomenon. Congestion in one wireless link spreads over space rather than localized to a link, causing interference to packet transmissions on neighboring links. Therefore, the space-shared feature of multihop wireless network makes congestion control different from that in wired networks. Since TCP often errs in estimating congestion level due to the wireless interference, it can overly inflate the transmission window and blast packets into the network, resulting in high level of congestion. We propose a novel algorithm to detect congestion in multihop wireless networks, which enables TCP to adjust the window size precisely. Performance evaluation through simulations confirms the advantage of our proposal in detecting spatial congestion in multihop wireless networks.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless Communication

General Terms

Algorithm

Keywords

Multihop wireless networks, congestion control, MAC

1. INTRODUCTION

As wireless networks become prevalent, there is an increasing demand of network connectivity in infrastructureless environments such as emergency situation. TCP is a natural choice as transport layer protocol because of its wide-spread use in the Internet. However, it has been shown that TCP performs poorly in multihop wireless environments [6].

The important nature of wireless link is that the transmission interferes with neighboring links. In particular, congestion spreads over space rather than limited to a link. Even with a single connection, packet transmissions (for the same connection) from neighboring nodes can interfere with one another, resulting in congestion and performance degradation. In such a case, limiting the number of outstanding packets is one solution. By forcing a small window limit, we can broaden the average inter-packet gap and reduce congestion.

There is considerable work on TCP over multihop wireless networks found that too large window size degrades its performance because packets interfere with each other, resulting in drops due to hidden/exposed terminal phenomenon [6, 10, 5, 1, 9]. While it is known that TCP can achieve better performance with limited window size, the optimal widow limit for maximal performance is not widely agreed upon.

TCP Vegas [4] can achieve better throughput in wireless environments because it adjusts window size based on measured round-trip time (RTT). Since congestion in wireless networks manifests itself as increased RTT delay, Vegas can make more precise decision on congestion than in wired networks [11]. However, it retains the fairness problem when used along with existing TCPs [7] and fails to increase window size in longer-haul networks [6].

Investigating factors affecting the optimal window limit in multihop wireless networks, we conclude that estimating the optimal value is very hard because of network dynamics. Instead, we make TCP adjust the window size based on a congestion measure using Explicit Congestion Notification (ECN) [8].

The ECN mechanism finds use in multihop wireless networks to tune TCP window size. A forwarding node can mark the ECN bit when it detects congestion, and the sender reduces transmission rate upon receiving an ACK with the ECN bit set. Fu *et al.* use ECN in multihop wireless network to inform network congestion that is detected from the number of MAC retries [6].

The distributed feature[1] of wireless Medium Access Control (MAC) protocol becomes an obstacle in determining congestion. In this paper, we propose a novel algorithm de-

IWCMC'06, July 3–6, 2006, Vancouver, British Columbia, Canada.

[1]We assume the standard IEEE 802.11 Distributed Coordination Function (DCF) as MAC layer protocol.

tecting incipient congestion from packet's staying time[2] in the MAC layer. We introduce the notion of congestion potential verifying that the staying time provides more precise measure of congestion than the retry count.

The rest of the paper is organized as follows. We investigate the factors affecting the optimal TCP window size in section 2. Taking the ECN mechanism as an alternative to optimize TCP performance, we propose a novel algorithm detecting congestion, which sets the ECN bit using staying time in MAC in section 3. After evaluating the performance of our proposals in section 4, we conclude our paper in section 5.

2. OPTIMAL CONGESTION WINDOW

Since TCP performance is influenced by the window limit, we investigate factors determining the optimal window limit, which we define as the window limit when TCP achieves the *maximal* throughput. We found two such factors in the literature.

- Topology - Even in a chain topology, one of the simplest topology, the optimal window limit varies with the number of hops.
- ACK policy - ACK thinning affects the optimal window limit.

The number of hops in the topology affects the optimal window limit. TCP can take advantages of spatial reuse in the long-hop connection because enough space allows multiple packets to be transmitted without interfering. Fu *et al.* indicated that TCP achieves the best performance with the window size of $hop\_count/4$ in [6].

ACK thinning is a generalized term for intentional ACK drops such as the delayed ACK option [3]. By sending out an ACK for a number of data packets, it reduces the bandwidth share of ACKs allowing less collision with data packets [2]. Hence, ACK thinning can boost TCP performance by letting more data packets in the network, increasing the optimal window limit.

We also found other factors affecting the optimal window limit through extensive simulations. Using NS-2 [12], we build a simple chain topology, where neighboring nodes are apart by 200 m and a TCP connection is established between end nodes without background traffic. We use the standard IEEE 802.11 DCF MAC with 2 Mbps physical rate, 250 m transmission range, and 550 m interference range. As for routing protocol, AODV is used. The data packet size is fixed to 512 bytes unless otherwise specified. For ACK thinning, the receiver generates two ACKs for a window worth of packets. Two ACKs for an RTT make the network fully-utilized and also provide minimum reliability for an ACK drop.

Table 1 reveals that the packet size has an influence upon the optimal window limit. The simulation with different packet sizes was run in the 8-hop chain topology. Since the network capacity is limited, the optimal window limit decreases with larger packet size but the maximal throughput increases due to smaller packet overhead. TCP with ACK thinning has larger window limit than normal TCP, making more efficient use of the network, especially when the packet size is small. It achieves 24.5% throughput increase with 1460-byte packet, and 39.2% increase with 512-byte packet.

[2]By 'staying time', we mean the time taken for a packet to be successfully transmitted or dropped after it starts competing for the medium.

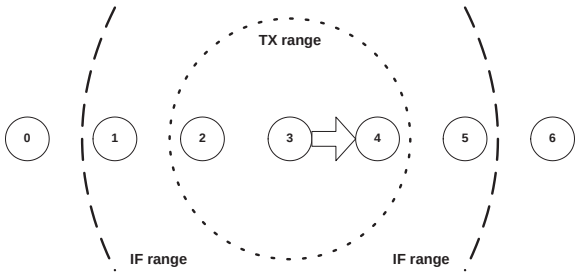


Figure 1: Transmission and interference range of a node in chain topology.

Table 1: Optimal window limit and maximal throughput with different packet sizes in the 8-hop chain topology.

Packet size	512 bytes		1000 bytes		1460 bytes	
	W*	T	W*	T	W*	T
TCP w/o ACK Thinning	3	139.0	3	200.5	3	228.9
TCP with ACK thinning	12	193.6	8	254.6	6	285.0

W* denotes the optimal window limit in packets,
T denotes the maximal throughput in Kbps.

Congestion level also causes differences in the optimal window limit. To show this, we simulate TCP and UDP in the 8-hop chain topology. First we establish a TCP connection from node 0 (the left-most) to node 8 (the right-most), and then a UDP flow with 250 Kbps in the reverse direction from node 7 to node 6. The results are shown in Table 2. The optimal window limit decreases when the network is congested with background traffic. Since the medium is shared with the UDP flow, TCP takes less share of the network capacity.

Table 2: Optimal window limit and maximal throughput with different congestion levels in the 8-hop chain topology.

Background traffic	UDP 0 Kbps		UDP 250 Kbps	
	W*	T	W*	T
TCP w/o ACK thinning	3	139.0	2	113.1
TCP with ACK thinning	12	193.6	6	127.6

W* denotes the optimal window limit in packets,
T denotes the maximal throughput in Kbps.

In summary, we illustrated that there are at least four factors affecting the optimal window limit; ACK policy, topology, packet size, and network congestion level. Hence, it is very hard if not impossible to estimate the optimal window limit through analysis. For instance, it will be infeasible in congested network to apply the upper bound of window limit as in [5], especially when there exist various sizes of packets.

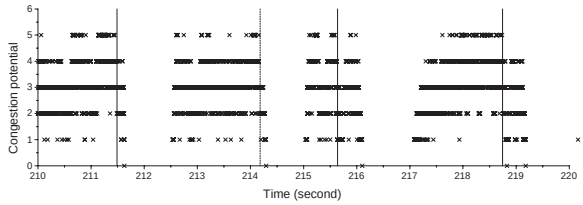
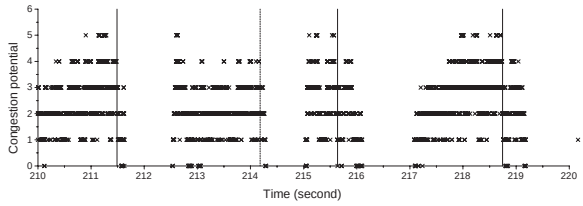
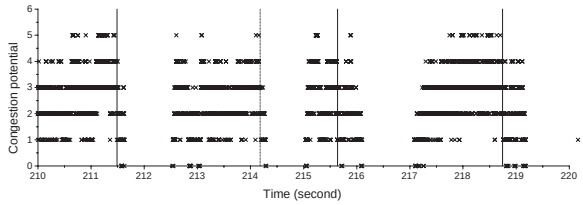


Figure 2: Trace of maximum CP among nodes and packet drops in the 8-hop chain network.



(a) CP of node 2



(b) CP of node 3

Figure 3: Trace of CP of individual nodes.

3. LINK-LAYER EXPLICIT CONGESTION NOTIFICATION

In the previous section, we showed that it is extremely difficult to analytically compute the optimal window limit. Alternatively, however, we can adjust the window size based on measured network congestion. For instance, TCP Vegas [4] adjusts its congestion window using end-to-end RTT measurement. Xu *et al.* showed, for this reason, that Vegas performs well in multihop wireless network [10]. Unfortunately, it has fairness problem when used with other TCP variants [7].

Another approach is to use ECN. Link RED proposed in [6] marks the ECN bit with a probability based on average number of retries in MAC. The sender reduces its transmission rate on receiving an ECN-set ACK. The main issue in this ECN mechanism, however, should be how to detect network congestion. Thus below we propose a congestion level detection algorithm based on packet's staying time in the MAC queue, instead of retries.

3.1 Congestion potential

In multihop wireless networks, congestion is not limited to a link between two nodes. Since nodes share air medium, transmissions interfere with one another causing congestion to spread over space. To understand congestion in space, we introduce a novel concept of *Congestion Potential* (CP). We define CP of a node as the number of neighboring nodes (including itself) that want to access the shared wireless medium for packet transmission. For instance, if node 3 in Fig. 1 has a packet for node 4 and starts contending to access the medium, the CP of the nodes in node 3's interference range, *i.e.* nodes 1, 2, 3, 4, and 5, increases by one. The CP decreases by one when node 3 finishes the transmission.

In the 8-hop chain topology, we measure the CP of all nodes after each packet transmission. Fig. 2 illustrates maximum CP among nodes. Three solid vertical lines represent data packet drops and dotted ones represent ACK drops. Each Cross hairs indicates the maximum CP just after a packet transmission or a drop. The correlation between CP and congestion level is evident in Fig. 2. Before a drop, the CP is driven high, indicating network congestion.

The CP of node 2 and 3 are very similar to maximum CP as shown in Fig 3. It implies that each node has a similar view of congestion in the network. Having the similar view of congestion is important in congestion control in multihop wireless network. If nodes have different view of congestion, some nodes would not be able to detect network congestion while others do. It would result in unfairness between transmissions on link layer.

While CP is a good measure of congestion in multihop wireless network, it is hard for a node to share its transmission information with neighboring nodes under the standard IEEE 802.11 DCF MAC protocol. The distributed nature of the protocol precludes a centralized coordinator that is responsible for calculating the CP of nodes. Hence, each node should infer its own CP from indirect estimation.

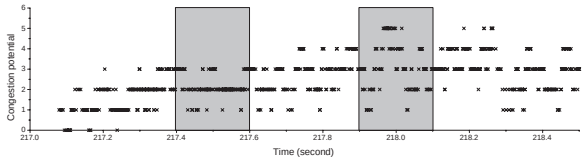
3.2 Link-layer ECN

Staying time in the MAC queue can be made to provide an indication of network congestion in the following way. A node starts a timer when a packet enters its MAC, and sets the ECN bit of the packet if the packet still waits for successful transmission after a threshold T. We call this ECN mechanism based on staying time in link layer, the Link-layer ECN (LECN).

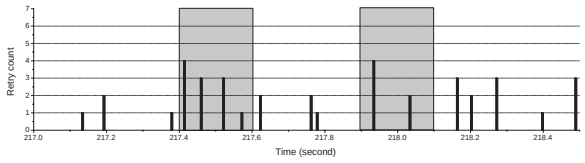
Retry count of MAC, which is used in [6], is another candidate. A node sets the ECN bit of the packet if the packet collides several times or the history of retry count exceeds a certain level.

While the retry count reflects the packet collisions, the staying time accommodates both backoff time and collisions. Fig. 4 illustrates the relevance of CP, retry count, and staying time of a node. We compare two shaded areas near 217.5 and 218.0 sec. It is evident that CPs near 218.0 sec are higher than those near 217.5 sec implying higher level of congestion. However, retry count does not correspond with CP. Retry counts in two shared areas are similar and even higher near 217.5 sec. In contrast, the staying time coincide with congestion levels, for instance showing outstanding increase of staying time near 218.0 sec.

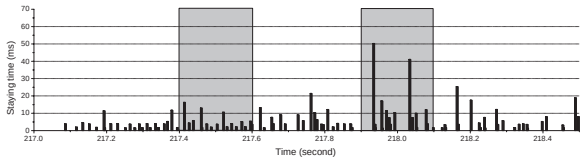
We simulate TCP with and without ACK thinning and LECN in the 8-hop chain topology. The threshold T for LECN is set to 60 ms. We measure the maximum CP for each case and get percentage of time for each congestion potential. The resulting distribution is shown in Fig. 5.



(a) Congestion potential

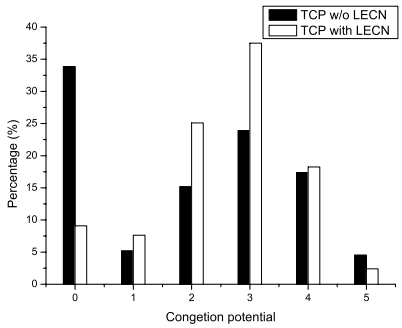


(b) Retry count

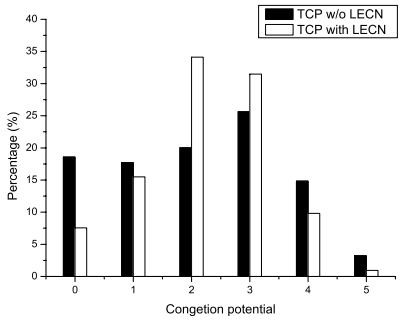


(c) Staying time

Figure 4: Comparison between CP, retry count, and staying time at node 2.



(a) TCP without ACK thinning



(b) TCP with ACK thinning

Figure 5: Distribution of the maximum CP among nodes. The threshold T of LECN is set to 60 ms.

For TCP without ACK thinning, LECN makes dramatic changes. CP of level 0 reduces from 33.85% to 9.90%, and CP of level 5 also decreases from 4.51% to 2.41%. CP of level 0 means that there is no packet in the network and CP of level 5 signifies that the network is so heavily congested that most nodes have packet to send. Hence, decreases of the high CPs mean increase of network utilization and less congestion. On the other hand, CP of levels 2 and 3, which state that 2 or 3 packets are spaced in the 4-hop range, increase from 15.16% to 25.10% and from 23.89% to 37.50% respectively.

For TCP with ACK thinning, the improvement from LECN is less dramatic but the tendency is similar and evident in both decrease of CP levels 0 and 5, and increase of CP levels 2 and 3. They also imply that the network is more highly utilized and heavy congestion situation is avoided.

4. PERFORMANCE EVALUATION

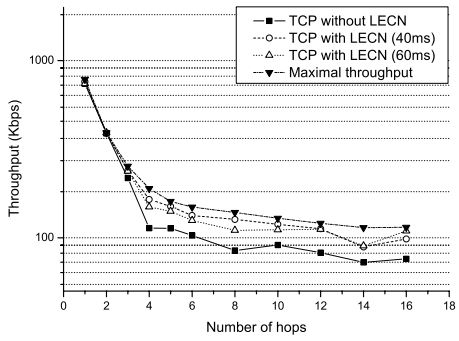
We compare the performance of TCPs with and without LECN. We simulate them in chain and lattice topology. Distance between nodes and other network settings are identical. TCPs have fixed window limit of 64. Although we simulate the threshold T of LECN from 0 to 100 ms, we present only the results for thresholds 40 and 60 ms because they perform best.

We first simulate TCPs with various number of hops in the chain. Fig. 6 presents throughput and average window limit of TCP without ACK thinning. The maximal throughput is obtained from exhaustive simulations changing the window limit. Notice that the y-axis has logarithmic scale. When the number of hops is less than 4, difference between TCP with and without LECN is not significant. However, as the number of hops increases beyond 4, TCP with LECN outperforms TCP without LECN. Fig. 6 shows that TCP needs to keep the window size small in order to achieve better performance. LECN suppresses TCP's window size growth and boots the throughput close to the maximum.

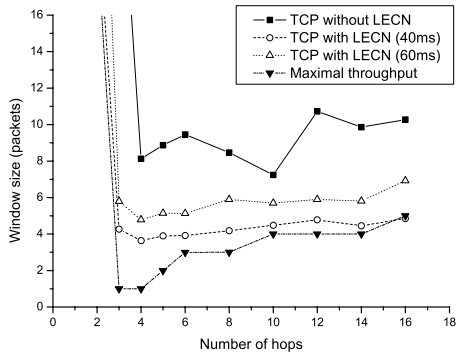
As to TCP with ACK thinning, the results shown in Fig. 7 are similar to those of TCP without ACK thinning. LECN successfully improves performance as well, and TCP with LECN outperforms TCP without LECN, achieving a bit less throughput than the maximal case.

In the next experiment, we compare some variant TCPs including TCP Vegas, which monitors end-to-end delay to adjust its window size. TCP Vegas outperforms TCP without ACK thinning (with and without LECN) but achieves less throughput than TCP with ACK thinning and LECN in Fig. 8. Compared with TCP with ACK thinning but without LECN, Vegas shows better performance only in connections with moderate number of hops between 5 and 10. TCP with both ACK thinning and LECN exhibits the most outstanding result. It outperforms all other TCP variants in almost all cases and achieves twice the throughput of TCP without ACK thinning and LECN.

Finally, we test TCP variants in the 9x9 lattice topology, with 8 horizontal and vertical connections, respectively.

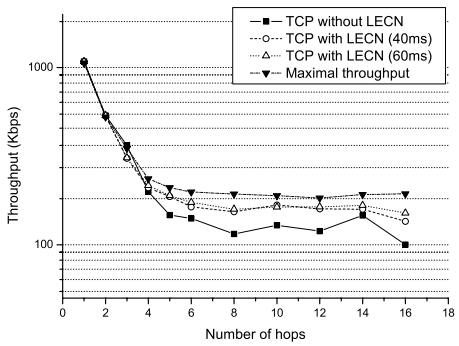


(a) Throughput

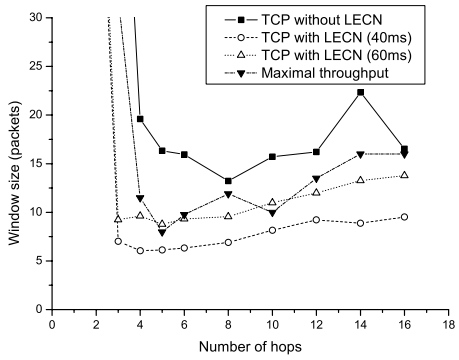


(b) Average window size

Figure 6: Performance of TCP without ACK thinning.



(a) Throughput



(b) Average window size

Figure 7: Performance of TCP with ACK thinning.

LECN uses the threshold of 60 ms. The results shown in Table 3 are aggregate throughput of all connections.

TCP with ACK thinning and LECN makes efficient use of the network more than others. The network capacity have increased upto 30% compared to TCP without ACK thinning with 512 bytes packet size. ACK thinning achieves this effect by reducing the number of ACKs. TCP without ACK thnning and with LECN shows similar performance with TCP Vegas. We claim that the choice between end-to-end type congestion detection like Vegas or hop-by-hop type one like ECN does not make significant difference in terms of network capacity.

5. CONCLUSION

We demonstrate that the optimal window size of TCP is dependent on ACK policy, topology, packet size, and network congestion level through extensive simulations. Since it is very hard to take account of all factors analytically, we propose LECN to improve TCP performance based on the ECN mechanism. LECN focuses on staying time in MAC in order to detect incipient congestion. The introduction of congestion potential verifies that staying time enables more precise decision than retry count. We evaluate performance of TCP variants including TCP Vegas. The comparison demonstrates that TCP achieves the best performance when equipped with both LECN and ACK thinning.

6. ACKNOWLEDGMENTS

This research was supported partially by the University IT Research Center Project and the ubiquitous Autonomic Computing and Network Project, Ministry of Information and Communication, in Korea.

7. REFERENCES

[1] E. Altman and T. Jimenez. Novel delayed ack techniques for improving tcp perfomance in multihop wireless networks. In *PWC*, 2003.

[2] H. Balakrishnan, V. Padmanabhan, and R. Katz. The effects of asymmetry on tcp performance. In *ACM Mobicom*, 1997.

[3] R. Braden. Requirements for internet hosts - communication layers. In *RFC 1122*, October 1989.

[4] L. Brakmo and L. Peterson. Tcp vegas: End to end congestion avoidance on a global internet. *IEEE Journal on Selected Areas in Communication*, 13(8), October 1995.

[5] K. Chen, Y. Xue, S. Shah, and K. Nahrstedt. Understanding bandwidth-delay product in mobile ad hoc networks. In *Computer Communication*, 2004.

[6] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla. The impact of multihop wireless channel on tcp throughput and loss. In *INFOCOM*, 2003.

[7] J. Mo, R. La, V. Anantharam, and J. Walrand.

Table 3: Throughput in lattice topology.

Packet size	512 bytes	1000 bytes	1460 bytes
TCP w/o ACK thinning	189.2	264.5	290.8
TCP w/o ACK thinning and with LECN	223.4	295.4	301.2
TCP with ACK thinning and LECN	**246.0**	**326.6**	**333.0**
TCP Vegas	226.4	287.2	296.2

Throughput is in Kbps, and threshold T of LECN is set to 60 ms.

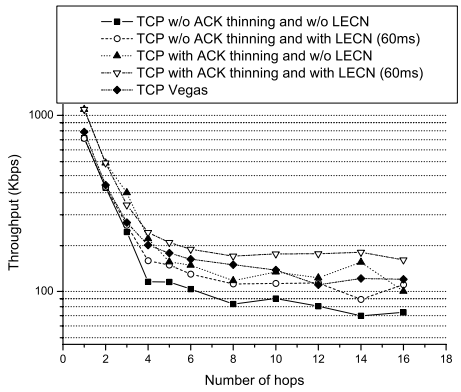


Figure 8: Throughput comparison of TCP variants.

Analysis and comparison of tcp reno and vegas. In *INFOCOM*, 1999.

[8] K. Ramakrishnan, S. Floyd, and D. Black. The addition of explicit congestion notification (ecn) to ip. In *RFC 3168*, September 2001.

[9] K. Xu, S. Bae, S. Lee, and M. Gerla. Tcp behavior across multihop wireless networks and the wired internet. In *WoWMoM*, 2002.

[10] S. Xu and T. Saadawi. Performance evaluation of tcp algorithms in multi-hop wireless packet networks. In *Wireless communications and mobile computing*, 2002.

[11] S. Xu, T. Saadawi, and M. Lee. Comparison of tcp reno and vegas in wireless mobile ad hoc networks. In *LCN*, 2000.