# Improving TCP Performance over Optimal CSMA in Wireless Multi-Hop Networks

Jinsung Lee, Hyang-Won Lee, Yung Yi, and Song Chong

***Abstract*—Optimal CSMA (oCSMA) has been receiving extensive attentions for its provable optimality in throughput and fairness without message passing over wireless multi-hop networks. However, recent studies suggest that TCP over oCSMA performs poorly, hindering the deployment of oCSMA in real networks. In this letter, we show that just a simple, additional virtual queue at the MAC layer can significantly improve TCP performance when oCSMA is used as the underlying MAC. Through testbed-based experiments, we demonstrate that with our virtual queueing scheme, TCP flows achieve near-optimal throughput performance in various scenarios.**

***Index Terms*—CSMA, TCP, fairness, 802.11 implementation.**

## I. Introduction

OPTIMAL Carrier Sense Multiple Access (oCSMA) has emerged as a promising practical solution for achieving optimality in fairness and throughput in wireless multi-hop networks [1]–[4]. The key idea of oCSMA is to adapt the CSMA's parameters dynamically to the network loads. More precisely, each node chooses its backoff time and transmission duration such that the links temporarily experiencing less transmission opportunities (e.g., due to high contention) access the medium more aggressively. This behavior is in sharp contrast to other MAC protocols in practice, e.g., 802.11 DCF, which reduces transmission aggressiveness under high contentions. Such contention-aware control of "transmission intensity" in oCSMA is known to achieve optimal throughput and fairness without any message passing (see e.g., [1]–[3] for theoretical results). There has also been research that seeks to implement oCSMA over real wireless networks [5]–[7].

In this letter, we study the practicality of oCSMA with a focus on its compatibility with TCP that is the most prevalent transport-layer protocol in the Internet. The interaction between TCP and oCSMA has been studied in [7], [8], both of which have reported that TCP over oCSMA performs poorly. The main reason is that, as opposed to how the transmission intensity should be controlled for optimality, the TCP flows over less served links (due to interference) operate toward *decreasing* the transmission intensities of those links under oCSMA. This further reduces transmission opportunities for those under-served links. Such a mismatch between TCP and oCSMA (is due to window-based rate update and ack clocking in TCP) will be further discussed in Section III-A. The authors in [8] study asymmetric contention scenarios where some TCP flows can be starved. They propose a *multi-session* approach that allows starved flows to open multiple TCP sessions, which eventually increases their transmission intensities to overcome starvation. However, such a multi-session approach incurs substantial overhead due to frequent open and close of TCP sessions. More importantly, it is specialized to work with oCSMA, and thus, when a node (e.g., a laptop) using the multi-session method gets connected to wired networks, the modified TCP may operate abnormally.

We take an alternative approach that does not require to modify TCP. We develop a simple *virtual queueing scheme* in the MAC layer that enables a less served link to increase the transmission intensity and thus get more scheduling chances; so that starved TCP flows along the link can have much improved throughput and fairness. Indeed, we show through extensive experiments that our virtual queue based oCSMA significantly improves the performance of TCP compared to the original oCSMA.

Manuscript received January 19, 2012. The associate editor coordinating the review of this letter and approving it for publication was C. Assi.

J. Lee, Y. Yi, and S. Chong are with the Department of Electrical Engineering, KAIST, Daejeon, Republic of Korea (e-mail: ljs@netsys.kaist.ac.kr, {yiyung, songchong}@kaist.edu).

H.-W. Lee (corresponding author) is with the Department of Internet and Multimedia Engineering, Konkuk University, Seoul, Republic of Korea (e-mail: leehw@konkuk.ac.kr).

Digital Object Identifier 10.1109/LCOMM.2012.071612.120157

## II. Background: Optimal CSMA

In CSMA, a node with backlogged data initiates a clock that expires after a random period, referred to as the *backoff time*. When the channel is sensed busy, the clock stops until the channel becomes idle, after which the clock resumes. Once the clock expires, the node grabs the channel and starts transmitting its packet for a duration, called the *holding time* or *transmission duration.*

**Algorithm 1** Optimal CSMA

1: During frame $t$, the transmitter of link $l$ runs CSMA($p_l[t], \mu_l[t]$), and records the amount $S_l[t]$ of service received during this frame;

2: At the end of frame $t$, the queue length of link $l$ is updated as

$$q_l[t+1] = \Big[q_l[t] + A_l[t] - S_l[t]\Big]^+, \tag{1}$$

3: Channel access probability $p_l$ and holding time $\mu_l$ are updated such that $p_l[t+1] \cdot \mu_l[t+1] = \exp(b \cdot q_l[t+1])$.

oCSMA is a version of CSMA that has a specific rule of setting the backoff and holding times. Algorithm 1 summarizes how oCSMA works. To achieve optimality in throughput and fairness, a link running oCSMA determines its access probability $p_l$, and holding time $\mu_l$, adaptively to the link queue length $q_l$. Given that time is divided into frames, let CSMA$(p, \mu)$ be the CSMA having a random backoff time with mean $1/p$ and a random transmission duration with mean $\mu$. The queue lengths are updated as in (1), where $A_l[t]$ is the amount of incoming packets (to the MAC layer) at frame $t$,

and $S_l[t]$ is the amount of served packets at frame $t$ over link $l$, and $[\cdot]^+ \equiv \max(\cdot, 0)$. We define the transmission intensity of link $l$ as the product of $p_l$ and $\mu_l$, and hence, it represents the aggressiveness of link $l$'s transmission attempts. As in Step 3 of Algorithm 1, the transmission intensity is equal to $\exp(b \cdot q_l)$, where $b$ is a positive scaling parameter that prevents the exponential function from growing too large. Therefore, for a given $\mu$, the access probability $p_l$ is an exponential function of the queue length of link $l$, and this makes a link with larger backlog attempt to transmit more aggressively. This property has an important implication to the upper-layer congestion control as discussed in the following.

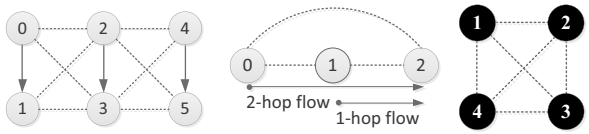
The amount of incoming packets over each link $l$, $A_l[t]$ is determined by the source rate (or congestion) control algorithm. It was shown in [1], [2] that when a utility based congestion control (UBC) is used, the optimal throughput is guaranteed in the sense that the long-term rate $\boldsymbol{x}^\star = (x_l^\star : l \in L)$ ($L$ is the set of all links) is the solution of the following optimization problem: $\boldsymbol{x}^\star = \arg\max_{\boldsymbol{x}} \sum_l U(x_l)$. Of particular interest is log utility function, i.e., $U(\cdot) = \log(\cdot)$, in which case the UBC source rate control is given as $A_l[t] = V/q_l[t]$ for some positive constant $V$. This source rate control together with oCSMA achieves proportional fairness [5], [7]. Intuitively, the MAC-layer queues of less served links will build up due to the arrival, and thus, their transmission intensities will increase to obtain more transmission opportunities. On the other hand, the source injection rate decreases as the queue length increases, so that less served links do not attempt transmission too aggressively. This prevents heavy contention and has the effect of balancing throughput among competing flows.

## III. Virtual Queue based oCSMA

### A. Why TCP over oCSMA performs poorly?

The main reason for TCP's poor performance is that *with TCP, even a link experiencing few scheduling chances would not be able to increase its queue length,* which keeps the starved links to stay starved. This is mainly due to the window-based congestion control in TCP, which generates packets based on ack clocking. We briefly summarize the ack-clocking mechanism of TCP. A TCP sender maintains a congestion window (CWND), which is the maximum amount of packets that can be transmitted without acknowledgement (ACK) packets from the corresponding receiver. The CWND value is increased when the transmitted packets have been acknowledged by the receiver within a certain time, and decreased otherwise. This window-based rate control can result in the starvation of TCP flows under oCSMA that assigns scheduling priority to heavily backlogged links.

Consider a TCP flow traversing a link, say $l$. Assume that link $l$ is less scheduled than other links due to such as heavier interference. The TCP flows along the well-served links easily increase their CWNDs, whereas the TCP flow along link $l$ has difficulty in doing so since its packets rarely reach to the destination. Note that the number of packets that can be enqueued into the MAC buffer is limited by the CWND value. Thus, under oCSMA, link $l$ is likely to have *lower transmission intensity* than other links, and consequently, link $l$ will receive fewer transmission opportunities. This can repeatedly occur, severely throttling the TCP flow over link $l$. This mismatch between TCP and oCSMA can be more visibly explained in the example network topologies appearing in many practical scenarios (see Fig. 1).




(a) FIM (b) Multihop (c) TCP+UBC

Fig. 1. Problematic scenarios; gray (resp., black) vertices represent nodes (resp., links), dotted lines represent connectivity (resp., interference), and arrows represent flows; a) FIM: only the middle link $(2, 3)$ interferes with two outer links $(0, 1)$ and $(4, 5)$; b) Multi-hop: there exist a 2-hop and a 1-hop flows in the fully connected network; c) TCP+UBC (contention graph): all links interfere with each other; links 1, 2, and 3 have UBC flows and link 4 has a TCP flow.

***Flow-in-the-middle (FIM).*** The topology in Fig. 1(a) is a well-known asymmetric contention scenario, where the middle link may often defer its transmission since it can transmit only when both of outer links are idle. In this case, the inner link is scheduled much less than the others, and as discussed above, the TCP flow along the inner link will struggle to increase its CWND. This leads to the starvation of TCP flow over the middle link when the underlying MAC adopts oCSMA.

***Heterogeneous hops.*** In oCSMA, the links along the path of a multi-hop flow may have lower transmission intensities, since the number of packets on the path is constrained by the TCP's CWND and the packets are spread over the links (see Fig. 1(b)). Consequently, with oCSMA, the links along the path of a multi-hop TCP flow are scheduled sporadically compared to that of a single-hop TCP flow, and thus, multi-hop TCP flows yield extremely low throughput.

***Heterogenous protocol.*** Consider a scenario where TCP flows coexist with non-TCP flows that work well with oCSMA such as UBC flows (see Fig. 1(c)). As discussed above, when there is heavy contention, the TCP flow rarely increases its CWND, leading to small backlog in the MAC buffer. On the other hand, the UBC sources keep injecting their packets (although the injection rate decreases as the queue builds up), leading to large backlog. Therefore, the UBC flows will be served better than the TCP flow and eventually, UBC flows will dominate TCP flows.

### B. Virtual Queue based oCSMA

***Mechanism.*** To improve TCP performance over oCSMA, we develop a mechanism that ensures that whenever a link traversed by a TCP flow is scheduled less than the others, its transmission intensity should be appropriately increased; so that the TCP flow escapes from the starvation loop discussed in Section III-A. Note that the poor TCP performance results from the coupling between the MAC-layer queue and TCP. Hence, our idea is *to decouple the actual MAC queue and TCP dynamics.* In particular, we introduce an additional counter, called *virtual queue* and use the virtual queue length to determine the parameters of oCSMA.

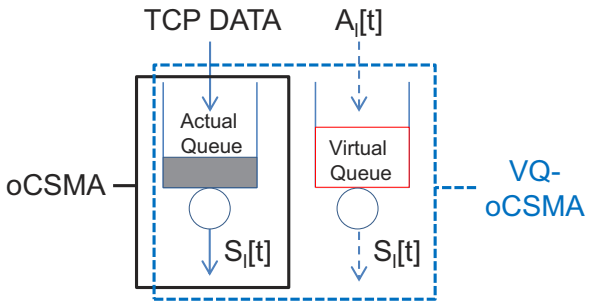



Fig. 2. Implementation of VQ-oCSMA algorithm; TCP packet triggers the update of virtual queue ($vq_l$) whose arrival process is $A_l[t]$ ($= V/vq_l[t]$) and departure process is the amount of scheduled packets, $S_l[t]$, at frame $t$.

In the MAC layer, each link $l$ maintains a virtual queue $vq_l$ updated as follows:

$$vq_l[t+1] = \left[ vq_l[t] + \frac{V}{vq_l[t]} - S_l[t] \right]_{\underline{vq}}^{\infty} \tag{2}$$

where $[\cdot]_d^c \equiv \max(d, \min(c, \cdot))$. The update of $vq_l$ is similar to that of $q_l$ in (1) except that the arrivals to $vq_l$ are virtually generated according to $V/vq_l[t]$ (where $V$ is a positive control parameter). Thus, the arrival rate decreases when the virtual queue builds up, and increases when the virtual queue decreases. Each link chooses the transmission intensity using the virtual queue length instead of the actual backlog, i.e., $p_l[t+1] \cdot \mu_l[t+1] = \exp(b \cdot vq_l[t+1])$. As a result, even when a link that a starved TCP flow traverses has small backlog, its virtual queue builds up and thus the transmission intensity increases. This will provide more scheduling chances to the link, improving the starved TCP flow's throughput. The parameter $\underline{vq}$ lower-bounds the virtual queue length to prevent a pathological case that when $vq_l$ is (close to) zero, the arrival $V/vq_l$ grows to infinity. For convenience, in the experiment, we use $\underline{vq} = 1$, which indicates that there exists at least a single virtual packet in the virtual queue.

***Implementation.*** Our mechanism requires only a slight modification of adding a counter operation for the virtual queue update at MAC layer, thus it is easy to implement on real hardware at a negligible cost. Fig. 2 describes the implementation of VQ-oCSMA. The update of the virtual queue is triggered when the first TCP DATA packet arrives at the empty MAC buffer in the source node. TCP DATA packets are enqueued into the actual MAC buffer and the virtual queue is updated according to (2). Since our VQ-oCSMA only modifies the MAC layer, it is transparent to higher layer protocols such as TCP and UDP.

## IV. Performance Evaluation

To validate the proposed scheme, we implement the VQ-oCSMA over off-the-shelf 802.11 hardware using the CommonCode platform [9]. Table I summarizes the experimental setup. We use TCP Reno which is the most popular TCP in the current Internet. For all experiments, maximum CWND value ($\text{CWND}_{\max}$) and maximum segment size (MSS) are fixed with 256 kB and 1024 bytes, respectively. All measurements are averaged over 20 runs, each running for 300 seconds.

TABLE I
Setup for TCP Experiments

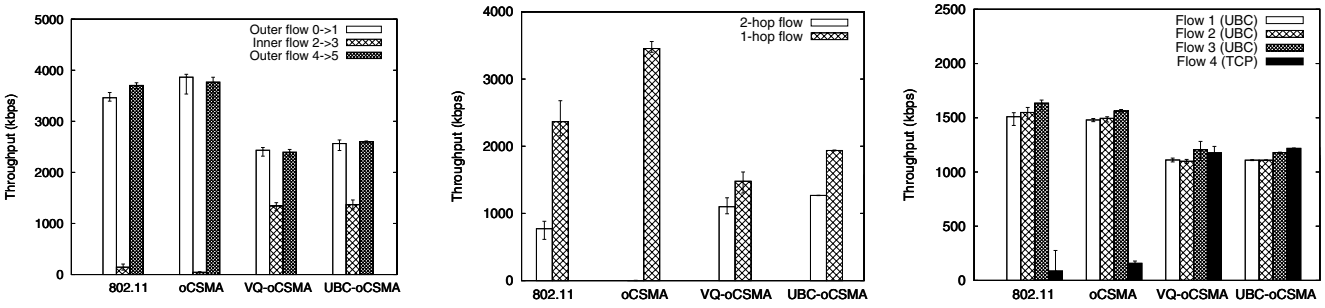
| | |
|---|---|
| *PHY* | 802.11a, 5.805 GHz band, 6 Mb/s rate |
| *TCP MSS and* $\text{CWND}_{\max}$ | 1024 bytes, 256 kB |
| *Parameters* | $b = 0.01$, $V = 500$, $\underline{vq} = 1$ |

We compare the virtual queue-based oCSMA (VQ-oCSMA) with 802.11 and oCSMA in the three scenarios in Fig. 1. To compare with the optimal control, we also present the throughput of UBC flows over oCSMA with log utility function (UBC-oCSMA), which achieves the proportional fairness, as discussed in Section II. In addition, each TCP ACK packet is delivered through the same MAC scheme used for its corresponding TCP DATA packet.

***FIM scenario (Fig. 1(a)).*** Fig. 3(a) shows that our scheme significantly improves throughput fairness among TCP flows in the FIM scenario. Since the TCP flow along the middle link cannot increase CWND due to higher contention than outer ones, both 802.11 and oCSMA do not grant the middle link sufficient transmission opportunities to overcome such low throughput. This results in the starvation of the TCP flow traversing the middle link. On the other hand, under our VQ-oCSMA, the virtual queue enables the middle link to attempt transmissions more aggressively and thus the TCP flow to increase its CWND because the virtual queue builds up quickly whenever the TCP flow along the middle link starves. Note that the throughput distribution of VQ-oCSMA is very similar to that of UBC-oCSMA.

***Multi-hop scenario (Fig. 1(b)).*** We conduct the multi-hop experiment where a 2-hop flow and a 1-hop flow contend in the network. Fig. 3(b) compares the throughput achieved by each flow under the four schemes. The 802.11 schedules links $(0,1)$ and $(1,2)$ in a round-robin manner (on average), thus fairly allocating resources among flows, while oCSMA grants much more channel access to link $(1,2)$, resulting in frequent timeouts and starvation of the 2-hop flow. However, our virtual queueing scheme helps mitigate the channel access imbalance by raising the virtual queue length at link $(0,1)$ and lowering the virtual queue length at link $(1,2)$. As a result, the transmission intensity of link $(0,1)$ gets closer to that of link $(1,2)$. Unlike in the FIM scenario where all the actual queues are almost always nonempty, in the multihop scenario, link $(0,1)$ traversed by only 2-hop TCP flow sometimes experiences the empty queue. In this case, the VQ-oCSMA keeps the virtual queue high enough so that the TCP packets arriving at the empty queue can be served with high transmission intensity. As a result, our VQ-oCSMA guarantees the improved fairness of 2-hop flow and approximates the throughput distribution of UBC-oCSMA.

***TCP+UBC scenario (Fig. 1(c)).*** The last experiment examines the coexistence of TCP and UBC flows. Fig. 3(c) shows that VQ-oCSMA achieves much fairer throughput than 802.11 and oCSMA. Under 802.11, all the links experience severe collisions due to heavy contention, leading to frequent timeouts to the TCP flow. Thus, the TCP flow yields extremely low throughput. In contrast, under oCSMA, the (starved) TCP flow




(a) FIM: 3 TCP flows (b) Multihop: 2+1 hop TCP flows (c) TCP+UBC: 3 UBC and 1 TCP flows

Fig. 3. Throughput comparison of 802.11, oCSMA, VQ-oCSMA, and UBC-oCSMA: 95% confidence intervals are also shown. The average throughput distribution of VQ-oCSMA is similar to that of UBC-oCSMA, implying that VQ-oCSMA nearly achieves the proportional fairness.

is likely to have more chances to increase its CWND because the well-served links tend to be less aggressive. So, the TCP flow achieves higher throughput, but the gain is marginal, since as discussed in Section III-A, the TCP flow has much lower transmission intensity than other UBC flows. Our VQ-oCSMA gives higher transmission intensity to the TCP flow by increasing the virtual queue of the link associated with TCP flow and this significantly improves throughput fairness among TCP and UBC flows. Similarly to the previous scenario, the VQ-oCSMA can maintain $vq$ high even in the case of empty actual queue, and hence it can immediately provide high aggressiveness to new TCP packets entering the empty buffer. In this way, our virtual queueing scheme achieves very close performance to UBC-oCSMA.

## V. Conclusion

In this letter, we proposed a simple yet effective virtual queueing scheme for optimal CSMA to improve both throughput and fairness of TCP flows. We demonstrated through testbed-based experiments that our VQ-oCSMA works as designed to address the TCP starvation problem in various contention scenarios.

## Acknowledgment

This research was supported by the KCC (Korea Communications Commission), Korea, under the R&D program supervised by the KCA (Korea Communications Agency).(KCA-2011-09913-04005, KCA-2012-11913-05004).

## References

[1] L. Jiang and J. Walrand, "A distributed CSMA algorithm for throughput and utility maximization in wireless networks," *IEEE/ACM Trans. Netw.*, vol. 18, no. 3, pp. 960–972, June 2010.

[2] J. Liu, Y. Yi, A. Proutiere, M. Chiang, and H. V. Poor, "Towards utility-optimal random access without message passing," *Wiley J. Wireless Commu. Mob. Comp.*, vol. 10, pp. 115–128, Jan. 2010.

[3] S. Rajagopalan, D. Shah, and J. Shin, "Network adiabatic theorem: an efficient randomized protocol for contention resolution," in *Proc. 2009 ACM Sigmetrics*.

[4] S.-Y. Yun, Y. Yi, J. Shin, and D. Y. Eun, "Optimal CSMA: a survey,' in *Proc. 2012 IEEE ICCS*.

[5] J. Lee, J. Lee, Y. Yi, S. Chong, A. Proutiere, and M. Chiang, "Implementing utility-optimal CSMA," in *Proc. 2009 Allerton Conf.*

[6] J. Lee, Y. Yi, S. Chong, B. Nardelli, E. Knightly, and M. Chiang, "Making 802.11 DCF optimal: design, implementation, and evaluation," submitted for publication.

[7] B. Nardelli, J. Lee, K. Lee, Y. Yi, S. Chong, E. Knightly, and M. Chiang, "Experimental evaluation of optimal CSMA," in *Proc. 2011 IEEE Infocom*.

[8] W. Chen, Y. Wang, M. Chen, and S. C. Liew, "On the performance of TCP over throughput-optimal CSMA," in *Proc. 2011 IWQoS*.

[9] J. Lee, J. Lee, K. Lee, and S. Chong, "CommonCode: a code-reuse platform for wireless network experimentation," *IEEE Commun. Mag.*, vol. 50, no. 3, pp. 156–163, Mar. 2012.