Improving Performance of Remote TCP in Cognitive Radio Networks

Hyun Yang, Sungrae Cho and Chang Yun Park
School of Computer Science and Engineering, Chung-Ang University, Seoul, Korea
[e-mail:yanghyun@cnlab.cse.cau.ac.kr, {srcho,cypark}@cau.ac.kr]
*Corresponding author: Chang Yun Park

Received April 6, 2012; revised July 19, 2012; accepted August 22, 2012
published September 26, 2012

Abstract

Recent advances in cognitive radio technology have drawn immense attention to higher layer protocols above medium access control, such as transmission control protocol (TCP). Most proposals to improve the TCP performance in cognitive radio (CR) networks have assumed that either all nodes are in CR networks or the TCP sender side is in CR links. In those proposals, lower layer information such as the CR link status could be easily exploited to adjust the congestion window and improve throughput. In this paper, we consider a TCP network in which the TCP sender is located remotely over the Internet while the TCP receiver is connected by a CR link. This topology is more realistic than the earlier proposals, but the lower layer information cannot be exploited. Under this assumption, we propose an enhanced TCP protocol for CR networks called *TCP for cognitive radio* (TCP-CR) to improve the existing TCP by (1) detection of primary user (PU) interference by a remote sender without support from lower layers, (2) delayed congestion control (DCC) based on PU detection when the retransmission timeout (RTO) expires, and (3) exploitation of two separate scales of the congestion window adapted for PU activity. Performance evaluation demonstrated that the proposed TCP-CR achieves up to 255% improvement of the end-to-end throughput. Furthermore, we verified that the proposed TCP does not deteriorate the fairness of existing TCP flows and does not cause congestions.

Keywords: TCP, Congestion control, cognitive radio networks

http://dx.doi.org/10.3837/tiis.2012.09.020

1. Introduction

Along with significant demands on wireless resources, cognitive radio (CR) technology has recently received much attention. In cognitive radio networks (CRN), secondary (or unlicensed) users can periodically search and identify available channels in their spectrum of operation. Based on sensing results, secondary users (SUs) dynamically tune their transceivers to an identified channel to communicate among themselves without disturbing the primary (or licensed) users (PUs). When a SU detects a PU, the SU must release the channel to the PU and continue to use another available channels, if any. This is due to the basic principle that the PUs have the exclusive right to occupy the channel [1].

Recent researches on CRNs have concentrated on performance improvement in physical and MAC layers, such as optimization of channel sensing, efficient channel allocation, and channel sharing, because they have focused primarily on maximizing bandwidth capacity. However, CRN research also requires development of higher layer protocols, such as within the transport layer, in order to support application-level quality of service (QoS).

Transmission control protocol (TCP) is one of the most popular transport layer protocols. One aspect of TCP is that it reduces the transmission rate when a packet is lost, because it regards the packet loss as an indicator of congestion [2]. In CRN, however, loss of a packet might also be due to PU arrival. For example, suppose that a TCP sender is remotely located over the Internet, while the access network for a TCP receiver is based on a CR link. We assume an AP is wired to the Internet and provides wireless access to the TCP receiver. Under this topology, the PU may cause communication to cease between the TCP receiver (SU) and the AP (SU). In this case, the end-to-end TCP transmission rate decreases and the end-to-end delay increases since the TCP starts congestion control. In our experiment, we observed that the end-to-end throughput of the TCP[1] decreases for the following reason. When the PU arrives, the AP or the receiver must release the channel licensed to the PU, and it attempts to find an alternative channel from a candidate specrum set. However, if the attempt is not successful, the AP or the receiver must cease its communication. In this case, the round trip time (RTT) of the TCP becomes longer, which invokes a retransmission timeout (RTO). Thus, the TCP miscomprehends this event as serious congestion and drops the congestion window to 1, resulting in decreased throughput. Segments transmitted from the TCP sender exeeding the AP capacity may be lost, resulting in AP overflow, since the CR link is blocked.

Existing TCP protocols for CRNs have assumed that either all nodes are in CR networks [7][8] or the TCP sender side is in CR links [9][10]. In those proposals, lower layer information such as CR link status could be easily exploited to adjust the congestion window so that the throughput can be maximized. However, this situation does not apply to our problem, in which the TCP sender is located remotely and connected by wire-line over the Internet, while the access network for the TCP receiver is based on a CR link. This topology is a more realistic deployment scenario, in which the lower layer information is not useful. Slingerland et al. [4] also concluded that TCP performance in our scenario is significantly deteriorated. To our best knowledge, our research is the first work targeting TCP performance improvement in this topology scenario.

Exploiting lower layer information is still possible in this topology. However, this study determines not to utilize the lower layer information due to the following two reasons. First,

[1] We used the TCP-SACK [12] for the experiment.

transmitting the CR link status to the remote sender is not always possible without extra components or channels. When a PU appears, the TCP connection is blocked and thus the status information cannot be transmitted through the connection. To be able to always transmit information, there should be an extra mechanism other than the end-to-end TCP connection, where some extra channels or components never blocked by PU activity should be involved. Second, although the status can be always transmitted, incorrectness caused by transmission delay is inevitable. Therefore, it is concluded that a simple approach purely localized to TCP may be less correct but more practical than utilizing the lower layer information.

In this paper, we propose a new TCP protocol called *TCP for cognitive radio* (*TCP-CR*) for CRNs. In TCP-CR, the TCP sender detects PU arrival without support from the lower layer protocols. Based on the PU detection, we propose two techniques: delayed congestion control (DCC) and dual-phase congestion window (DPCW). The DCC scheme is performed whenever the RTO expires. The RTO may occur when the PU arrives. In this case, performing congestion control will degrade the TCP throughput performance, as explained above. If the RTO is caused by congestion, the TCP sender should perform congestion control. Thus, TCP-CR delays perfoming congestion control on the RTO until it can determine if PU interference is involved. Furthermore, we introduce DPCW, since transmitting TCP segments while the PU remains on the channel can be too aggressive. The DPCW scheme uses two separate scales of a congestion window: the window is adjusted to the size of the AP buffer in additive increase and multiplicative decrease (AIMD) fashion when the PU remains on the channel, and the ordinary TCP-SACK is applied when the PU vacates the channel.

Most of the existing TCP protocols for the CRN do not mention how they react on interrupting events not related with PU, such as wireless link errors. In contrast, our proposed protocol takes into account congestion, wireless link errors, and PU arrivals. We validate that TCP-CR responds properly to all those events.

The rest of this paper is organized as follows. Section 2 describes related work. Section 3 considers the characteristics and problems in the existing designs of TCP over CRNs. In Section 4, we propose algorithms of the TCP-CR to improve the end-to-end throughput in the CRN. Extensive performance evaluation of the proposed algorithms is given in Section 5. Finally, in Section 6 we offer conclusions of our work and indications for future research.

2. Related Work

Recently, several performance evaluations of existing TCP over cognitive radio networks (CRNs) were conducted [4][5][6]. Slingerland *et al*. [4] concluded that the TCP-Reno can work well in the following situation: an unlimited buffer is available at the AP and the TCP receiver employs the selective acknowledgment (SACK) option. In practice, however, the capacity of the AP buffer is limited, and thus AP overflow may occur. Issariyakul *et al*. [5] also illustrated the performance of TCP in the CRN. They found that the TCP in the CRN has to cope with losses caused by PU arrivals; otherwise performance is significantly deteriorated. However, their research did not address wireless link errors such as bit errors, and in fact it is necessary to consider that link errors can be caused by wireless transmission impairments as well as the losses due to PU arrival. Felice *et al*. [6] investigated an extension of the ns2 simulator [11] to support a realistic simulation of the CRN by developing a model of PU activities and SU spectrum management functionality. Moreover, they provided an analysis of TCP performance over the CRN. They discovered that PU behavior and the channel sensing

interval are important factors in deciding the optimal end-to-end performance. However, they did not provide a scheme for performance improvement of TCP over the CRN.

Protocols for improving performance of TCP over the CRN were proposed in [7][8][9][10]. These protocols assumed the underlying technology to be either all CRNs [7][8] or CR links on the TCP sender side [9][10]. Luo *et al*. [7] studied an optimal TCP throughput based multi-channel access scheme over the CRN and formulated the channel access process over the CRN as a stochastic system. They improved performance by using a cross layer design approach in which they jointly condered a modulation and coding scheme in the physical layer and frame size in the MAC layer. In [8], the parameters of physical and MAC layers over the CRN were jointly optimized to maximize TCP throughput, modeling the CR system as a partially observable Markov decision process (POMDP). They argued that the design parameters of the CRN significantly impact TCP performance, improving it substantially if low layer parameters of the CRN are optimized jointly. Chowdhury *et al*. [9] proposed a window-based transport protocol which adapts the existing TCP rate control algorithm to interact with the physical and network layers. In this protocol, the TCP sender at the source node maintains information about the network state and adjusts its transmission rate based on updates from the intermediate nodes and feedback from the destination. However, this approach does not cope appropriately with cases in which the intermediate nodes are removed or the routing path is reconfigured. Sarkar *et al*. [10] proposed a protocol for the transport layer of the CRN that uses a cross layer approach to serve delay-tolerant applications and to adjust the congestion window by considering spectrum sensing and bandwidth variations.

As mentioned above, most TCP protocols for the CRN have assumed that either all network devices are connected by CR links or that at least the TCP sender is connected by a CR link. In this scenario, lower layer information such as PU activity could be easily exploited at the TCP sender. However, a more practical deployment of CR technology would be that the TCP sender or server is remotely connected over wire-line Internet while the TCP receiver or client accesses the wire-line Internet via CR technology. In this case, the lower layer information at the TCP receiver is not helpful to the TCP sender due to long RTT. Therefore, the TCP sender needs a different technique to detect PU arrival at the TCP receiver.

3. Observations and Characteristics of Existing TCP in CRN

We assume the following regarding network environments and PU activity:

- The TCP sender is located remotely and connected by wire-line over the Internet, while the access network for the TCP receiver is based on a CR link.
- Traffic is generated from the TCP sender to the TCP receiver (SU).
- Packets from SUs (TCP receivers or APs) may collide with packets from PUs.
- PU activity follows the exponential on/off model.

Fig. 1 shows our scenario in which the SUs (TCP receivers) and SU APs opportunistically transmit their data over a channel that is not used by a PU. When they detect the signal of a PU or a PU base station, the SUs and SU APs must release the channel [8]. Using this method of CR channel access, the TCP receivers can download a large variety of traffic from the remote TCP sender over the Internet. At times, data transmitted between an SU receiver and an SU AP may be blocked by a PU. To account for this, we represent PU behavior with an exponential on/off model in which the "on" duration (corresponding to PU arrival) is exponentially distributed with mean T_{ON}, and the "off" duration (SU communication time) is exponentially

distributed with mean T_{Off} [5]. Furthermore, in contrast to most existing TCP protocols for CR [4][5][6][7][8][9][10], we include wireless link errors in our analysis of the TCP-CR.

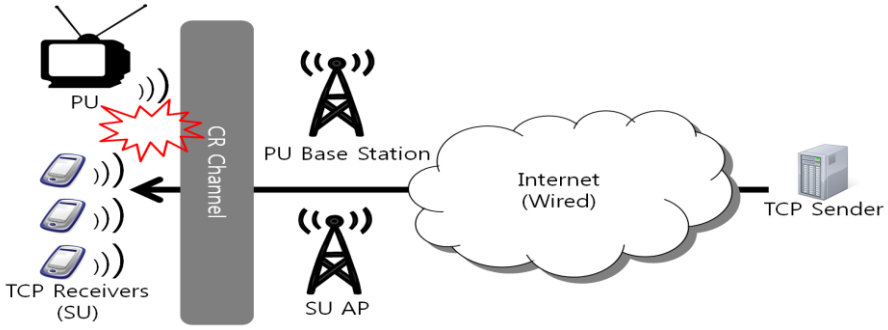

Fig. 1. Experimental environment considered in our scenario

Previous research has studied topologies such as 1) all TCP sender and receiver connections are via CR links [6][7][8] or 2) the CR access network is located on the TCP sender side [5][10]. In those scenarios, the TCP sender can obtain CR link status from the lower layers through cross-layering approaches. However, in our scenario, which is more realistic to the existing Internet, the intermediate nodes cannot relay the CR link status to the TCP sender.

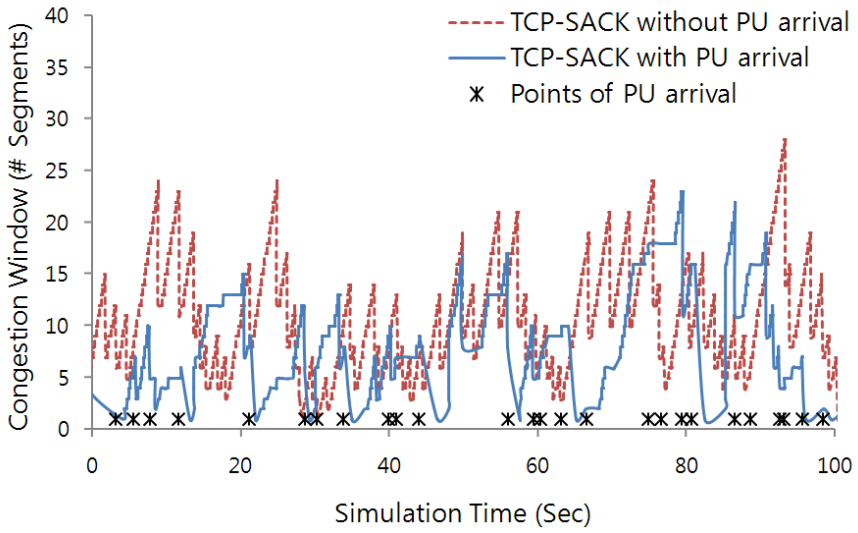

Fig. 2. Congestion windows vs. simulation time with or without PU arrival

(wired link bandwidth=10Mbps; CR link bandwidth=10Mbps; propagation delay of wired link=100msec; propagation delay of CR link=1 μs; size of TCP window=64Kbytes; TCP segment size=1Kbytes; FER=0.1; T_{ON}=1.0; T_{OFF}=1.0)

To analyze the behavior of existing configurations with the remote TCP sender connected via a CR link, we used the ns-2 simulator [11]. We measured the congestion window of the

TCP-SACK [12] with and without PU arrival. **Fig. 2** shows the congestion window versus simulation time. The TCP-SACK without PU arrival (in red) has a mostly higher congestion window than with PU arrival (in blue). In this simulation, we assume FER=0.1. We observe that the congestion window of the TCP-SACK drops to 1 at most occurrences of PU arrival (marked by dots) in **Fig. 2**. This may be an overreaction, since the PU may vacate the channel while the congestion window remains low. Therefore, the existing TCP is not suited to the given scenario.

TCP congestion control is originally designed to operate in wired networks, and it is not appropriate for the CRN. In the CRN, the RTO may expire and the congestion window may drop to 1 when a PU arrives. This hyperreactive congestion control causes end-to-end throughput degradation. Since the underlying MAC protocol searches for an alternative channel, the blocking time at the TCP sender might be shorter in the CR link than that of the network-wide congestion in the wired network. Traditional congestion control is not necessary in this case, but rather is harmful to throughput performance. Instead, we propose that TCP throughput can be improved by making TCP congestion control insensitive. More specifically, the TCP may delay adjusting the congestion window until it detects that PU arrivals have actually occured. This may also reduce the TCP transmission rate under PU arrival such that the TCP sender sufficiently fills the AP buffer.

In this paper, we analyze various symptoms considering congestion, wireless link errors, and PU arrivals based on the simulation results. **Table 1** shows these symptoms perceived at the TCP sender along with actual network events. This provides key insight for our approach.

Table 1. CR symptoms and Our Approaches

	Case I	**Case II**	**Case III**	
Actual Events in CR Networks	PU arrival		Link Error	Congestion
	Transmission Blockage or Packet Drop	AP Overflow		
Symptoms perceived at TCP Sender	RTO, Duplicate ACK	RTO	Duplicate ACK	RTO, Duplicate ACK
Our Approach	Delayed Congestion Control (DCC)	Dual-Phase Congestion Window (DPCW)	Existing Congestion Control	
Additional Information for Our Approach	Points of PU Arrival	Points of PU Arrival and Departure	-	

In the CRN, there can be transmission blockages, packet drops, or AP overflow when there is a PU on the CR link. There also occur link errors or congestions as usual. Traditionally, TCP protocols have considered a duplicate ACK as a symptom of wireless link errors, and RTO and duplicate ACKs have been regarded as being casued by congestion [2]. Existing TCP protocols cope well with these events, so our TCP protocol exploits existing mechanisms such as TCP-SACK or TCP Westwood [14] for **Case III** in **Table 1**. However, based on our experiment and [6][9][10], we observe that two types of symptoms arise due to PU arrivals: RTO alone or RTO with duplicate ACK.

Transmission blockage or packet drop can cause RTO and duplicate ACKs to occur as shown for **Case I** in **Table 1**. The transmission blockage is created because the SU cannot transmit ACKs while the PU occupies the channel. Once a PU starts occupying the channel, the packets sent to the SU should remain in the AP buffer because the AP should not use the channel. Then the RTO of the TCP sender expires, and the TCP sender considers it as a packet loss caused by congestion. This situation can be considered to be a premature timeout. In this situation, the end-to-end delay between the TCP sender and receiver suddenly increases, and the ACKs cannot be transmitted until the PU vacates the channel. At that time, the RTO of the TCP sender should be delayed to prevent unnecessary congestion control. On the other hand, packet drop may be caused by collision with packets generated by a PU arrival. This event provokes the RTO, and congestion control is performed, although the network actually is not yet congested. In this case, congestion control is not necessary. To resolve transmission blockage or packet drop, therefore, our approach is to delay congestion control (DCC); the TCP sender immediately retransmits the lost packet(s) but postpones the TCP timeout decision until verifying whether the PU is invloved.

AP overflow entails RTO only for **Case II** in **Table 1**. The AP overflows in the CRN occur when the network is congested or when the packets sent to the SU cannot be released at the AP buffer while the PU occupies the channel. In this case, the existing TCP does not work well because it reacts after the AP overflows occur. To alleviate AP overflows in the CRN, as much traffic should be generated as the AP buffer can hold. Thus, we propose a dual-phase congestion control scheme which uses two separate scales of a window adapted for PU activity and inactivity phases.

The existing TCP cannot distinguish between these events, and so an aggressive congestion control could degrade TCP performance. To resolve this problem, we incorporate a scheme to detect PU activity first. The details of our approach for performance improvements are explained in section 4.2.

4. TCP for Cognitive Radio (TCP-CR)

In this section, we describe the proposed TCP-CR to improve end-to-end throughput based on PU detection, delayed congestion control, and two-phase congestion window.

4.1 PU Detection

For the remote TCP, it may not be possible to precisely detect PU arrival at the CR link; a delay of about one-half of the RTT is inevitable. Unless the PU has some deterministic characteristic, such as a period, precise detection is not feasible without help from the lower layers to distinguish PU arrival from interference (e.g., jitter or congestion) in the network caused by other traffic. However, we have developed a simple and somewhat coarse approximate detection algorithm for PU arrival. The method and its engineering decisions are described below.

When a PU arrives on the channel being used by the TCP, ACKs from the TCP receiver are piled up in the AP buffer at the lower layer. In this situation, the TCP sender suffers from ACK drought because it cannot receive ACKs until the PU vacates the channel. Once the PU leaves, ACKs piled up in the buffer are transmitted in a burst, so the TCP sender suffers from ACK burst. In this manner, in the TCP sender, the ACK receptions become repeatedly droughty and bursty. By recognizing this unique pattern, the TCP sender can detect PU activities. A state diagram of PU activities is shown in **Fig. 3**.

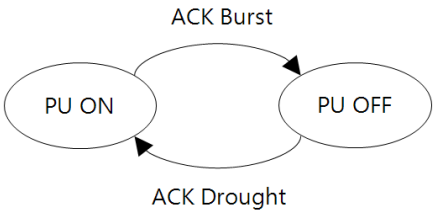

Fig. 3. State diagram of PU activities

In our TCP-CR, there are preconditions for detection of PU activities as follows. First, ACK drought may also happen via a non-CR link, so the remote TCP should be made aware that the other side is a CR link. This study assumes that this knowledge is transmitted in the optional header at the connection setup phase. Second, an ACK drought requires there to be an outstanding segment from the TCP sender, hence any ACK drought without an outstanding segment should be ignored.

There should be a guideline to determine whether the ACK drought is caused by PU arrival, because ACKs can be delayed for some time if user sending requests are sporadic or there is jitter in the network transmission. For this guideline, the TCP sender measures the elapsed time since the last ACK reception (the duration of ACK drought). If it is shorter than a certain threshold $d_{ACK\_Drought}$ (i.e., if an ACK is received within $d_{ACK\_Drought}$), then the TCP sender decides that no PU has arrived on the channel. If the length of ACK drought is longer than $d_{ACK\_Drought}$ (i.e., if an ACK is not received within $d_{ACK\_Drought}$), then the TCP sender decides it has detected a PU arrival. The parameter $d_{ACK\_Drought}$ should be carefully designed. If it is too long, PU interference shorter than $d_{ACK\_Drought}$ cannot be detected (*detection failure*) since ACKs after PU departure can be received before $d_{ACK\_Drought}$. If the threshold is too short, the TCP sender may erroneously decide it has detected a PU (*false detection*).

To determine $d_{ACK\_Drought}$, our study introduces the notion of the *minimum PU interference time* defined as the length of time during which the TCP sender cannot avoid experiencing transmission blockage due to CR characteristics. In the CR system, spectrum sensing is performed periodically to detect any PU arrivals, and this is perceived as interference from the viewpoint of the TCP sender regardless of PU arrival. Thus, the spectrum sensing time can be seen as the minimum PU interference time. We determined the minimum PU interference time to be 25 msec according to IEEE 802.22 which describes the wireless standard for cognitive radio [13].

Another factor affecting $d_{ACK\_Drought}$ is the end-to-end propagation delay between the TCP sender and receiver. In the remote TCP, a delay of one-half of the RTT is inevitable. If $d_{ACK\_Drought}$ is shorter than one-half of the RTT, there are concerns about false detection because $d_{ACK\_Drought}$ is not long enough for an ACK to arrive. Therefore, $d_{ACK\_Drought}$ is given by

$$d_{ACK\_Drought} = \max\left(\textit{the minimum PU interference time, one-half the RTT}\right) \tag{1}$$

The main condition that detects PU departure is an ACK burst. If the number of ACKs in a burst is larger than a certain threshold n_{ACK_Burst}, then it is decided that the PU has vacated the channel. The range of n_{ACK_Burst} should be [1, *number of outstanding segments*]. Larger values of n_{ACK_Burst} lead to higher probability of successful detection but more time to detect PU departure. We determined n_{ACK_Burst} to be 1 in order to detect quickly. In other words, when an ACK is received at the TCP sender, we decide simply that the PU has left the CR link.

In order to check the appropriateness of our PU detection, the correctness of PU detection is defined in a simple way as follows.

$$Correctness\ of\ Detection = 1 - \left(\frac{\#of\ False\ Detection\ +\ \#of\ Detection\ Failure}{\#of\ PU\ ON} \right) \quad (2)$$

We measure the correctness of detection while varying the variance of transmission delay as shown in **Fig. 4**. When there is little variance in delay, most detection errors come from detection failure; PU interference whose duration time is smaller than one half of delay may not be detected. False detections occur more frequently with larger delay variance because an ACK delayed longer than the average might be misinterpreted as ACK drought. The majority of errors is still detection failure. Overall, the correctness decreases as delay variance increases.

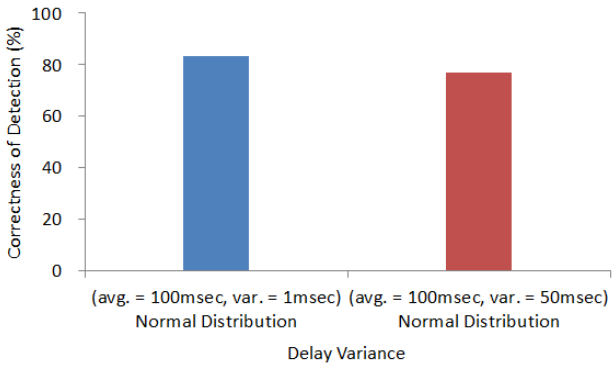

Fig. 4. The correctness of PU detection

It is difficult to understand the side-effect of detection error in detail without knowing the reactions on PU detection, which will be explained later. However, one can understand the principle that detection failure makes TCP-CR behave like the conventional TCPs while false detection may cause some overreactions. This means that detection failure may not do any harm but reduce improvement on TCP performance. False detection may generate some negative effects; for example, a timeout caused by congestion is misinterpreted as PU interference and the congestion window is not decreased. However, this negative effect is limited because false detection is soon corrected on arrival of ACK.

Although 80% of correct detection may look low, one should note that there exists inevitable delay between the CR link and the remote TCP. Our detection method is somewhat conservative in the sense that reducing negative effects is the first concern rather than improving the correctness of detection. Experiments on a lot of scenarios have confirmed that our simple detection method almost always gives better performance than other alternatives with respect to end-to-end throughput and fair network sharing.

4.2 Delayed Congestion Control Scheme

Originally, when the RTO of the TCP sender expires, the existing TCP performs two functions simultaneously: (1) retransmission to recover a lost segment and (2) congestion control to alleviate the congestion. However, if the RTO of the TCP sender is caused not by congestion

but by PU arrival, the second function of the congestion control creates performance degradation. To prevent this problem, our TCP-CR *separates* the above two functions, as shown in **Fig. 5**.

In the proposed *delayed congestion control* (DCC) scheme, when an ACK is received at the TCP-CR sender, an ACK drought timer is set. According to $d_{ACK\_Drought}$, the RTO of the TCP sender can expire before or after the ACK drought timer. When the RTO of the TCP sender expires before the ACK drought timer, the sender cannot identify whether that is caused by congestion or PU arrival. Therefore, the TCP-CR sender retransmits the lost segment immediately and then postpones congestion control until it can determine whether the PU has arrived. If the sender detects PU arrival (i.e., the ACK drought timer expires), it decides that the previous RTO expiration was caused by the PU arrival, and then it avoids congestion control by holding the current congestion window (CW) as the previous one. If an ACK arrives before the ACK drought timer expires, the TCP sender determines that the previous RTO expiration was not caused by a PU, and thus it performs traditional congestion control by setting the current CW to 1. We show the detailed procedure in **Fig. 6**.

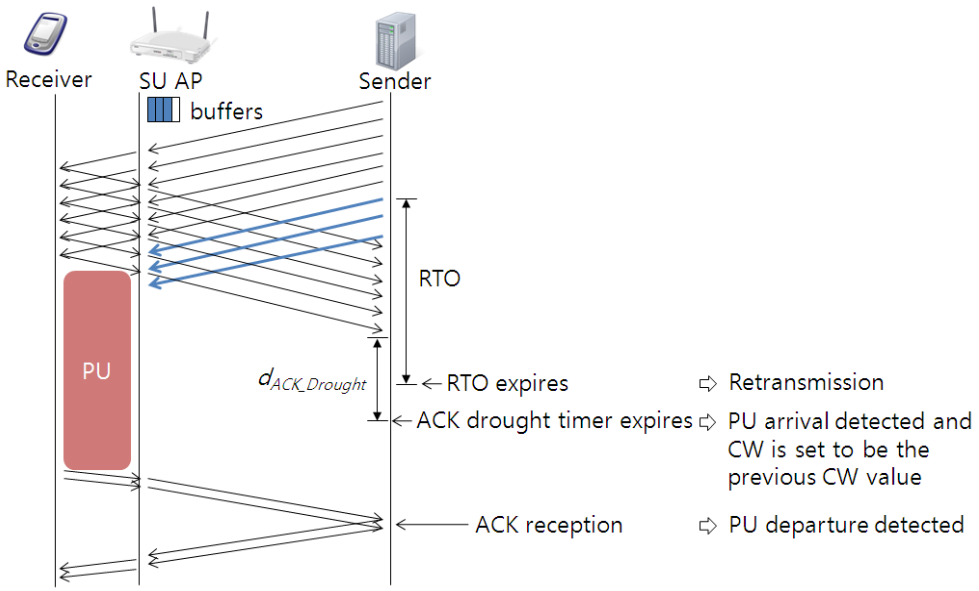

Fig. 5. An example of delayed congestion control scheme

On the other hand, when the RTO of the TCP-CR sender expires after the ACK drought timer expires, it can be determined immediately that the RTO expiration was caused by PU arrival. In this case, the TCP-CR sender retransmits the lost segment and avoids congestion control by holding the current CW as the previous one simultaneously. In this case, the TCP-CR sender does not postpone congestion control.

Meanwhile, if duplicate ACKs are received at the TCP-CR sender as soon as the PU vacates the channel, it means that there was ~~a~~ previous packet drop(s) due to the PU arrival (collision among PU- and SU-generated packets). In this situation, the TCP-CR sender retransmits the lost segment but does not perform congestion control, because the previous packet drop was not caused by congestion.

In this approach, side effects from congestion might be of concern. The TCP-CR does not perform congestion control promptly, holding the current CW as the previous one when the

RTO expires. This aggressive approach may deteriorate congestion if it occurs. To resolve this concern, we employ the dual-phase congestion window scheme. This technique is explained in detail in section 4.3.

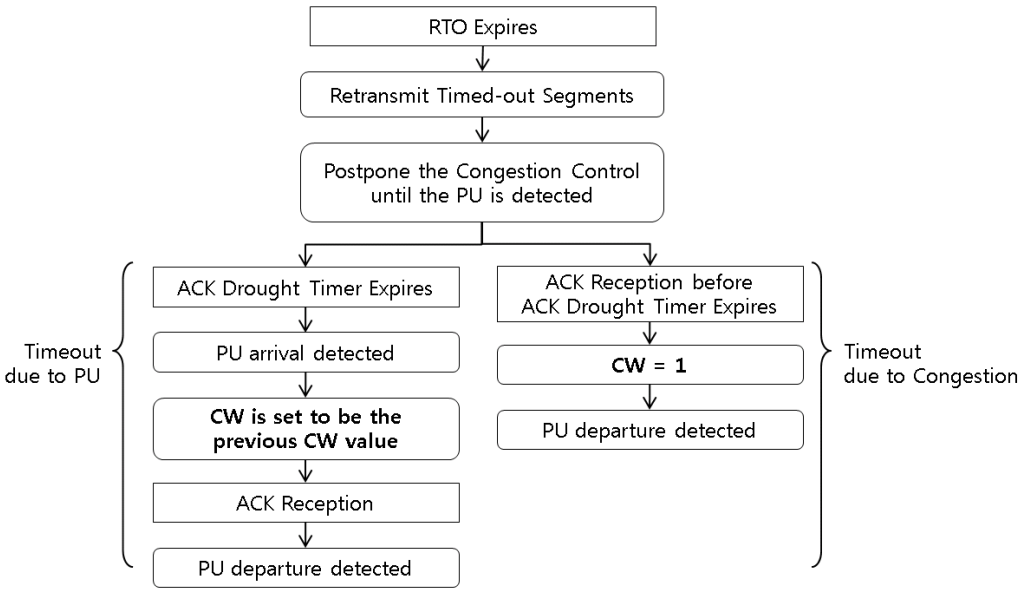

Fig. 6. Flow chart for delayed congestion control scheme

4.3 Dual-Phase Congestion Window

Traditional TCP congestion control operates independently from PU activities; the TCP sender generates data continuously according to the congestion window (CW) size, even if the PU arrives on the channel and the AP[2] cannot send any buffered frames. Eventually, this may cause overflow at the AP. To avoid AP overflows, the TCP sender should generate as much data as the AP buffer can hold. However, it should also fully utilize the available bandwidth when there is no PU interference. To manage this, the TCP-CR uses the *dual-phase congestion window* (DPCW) approach.

The rationale behind this approach is that the behaviors of a CR link vary significantly depending on PU presence on the channel. Thus, it is insufficient to estimate capacity based on only one scale of the congestion window. The packets sent to the SU (i.e., TCP-CR receiver) pass through the CR link while the PU vacates the channel. This is a typical issue of estimating available bandwidth of a link, and it has been well addressed by the existing TCP congestion control [2]. On the other hand, the packets sent to the SU are accumulated in the AP buffer while the PU occupies the channel. The issue is how to estimate the buffer size at the AP. After the PU vacates the channel, the link behavior returns *not smoothly but rather abruptly* to the previous mode without the PU. Therefore, the TCP-CR controls the transmission rate with two separate scales of the window, adapted to the PU activity.

The DPCW scheme introduces a new window called the PU window (PW), as shown in **Fig. 7**, in addition to the existing CW. While the PU vacates the channel, the existing CW is used to estimate a combined share of the link and the AP buffer. When the PU occupies the

[2] Refer to SU AP in Fig. 1.

channel, the PW is utilized instead of the CW to maintain the number of outstanding to fill the AP buffer without overflow. The PW also operates by using an additive increase multiplicative decrease (AIMD) approach, like the CW. The TCP-CR adjusts the transmission rate to the CW when a PU does not exist on the channel, and it adjusts the rate to the PW when a PU does exist on the channel.

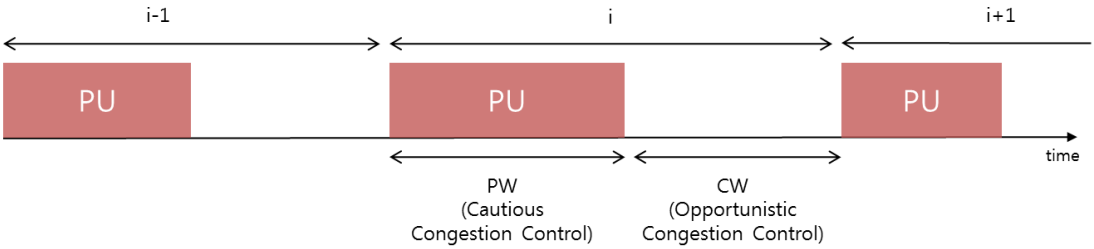

Fig. 7. Dual phase congestion window (PW: PU window)

As mentioned in section 4.2, this approach plays a major role in resolving the concerns about the aggressiveness of the DCC scheme. In the DPCW technique, the value of the PW is usually lower than the value of the CW, since the TCP-CR sender maintains the PW as the AP buffer fill level. This cautious approach prevents the DCC scheme from aggravating congestion or AP overflows.

Remarks: As mentioned, our detection algorithm can generate errors due to false detection and detection failure. False detection error may exacerbate the congestion because it causes the TCP-CR sender to keep the transmission rate by holding the CW. However, this detection error is not continuous, because it can be corrected after receiving an ACK. Therefore, the side effects for false detection error are not fatal.

Detection failure occurs when the TCP sender cannot detect the arrival of a PU in the CR link. This detection error may obstruct TCP performance improvement because it causes the TCP-CR sender to perform congestion control, when there is no actual congestion. This can occur if the length of PU occupancy on the CR link is shorter than $d_{ACK\_Drought}$.

5. Performance Evaluation

In this section, we evaluate performance of the proposed scheme compared with the TCP-SACK [12] under the experiment environment shown in **Fig. 1**. The simulation was performed using the ns2 simulator with the simulation parameters listed in **Table 2**. We assumed that PU activity followed the exponential on/off model described in section 3. The PU occupation rate, defined by T_{ON} / (T_{ON} + T_{OFF}), was set variably to 10-90%. We measured the end-to-end throughput of TCP and the TCP retransmission rate, defined as follows:

- **End-to-end throughput of TCP**: the goodput measured at the TCP sender
- **TCP retransmission rate**: the number of retransmitted segments per total transmitted segments at the TCP sender

5.1 The End-to-End Throughput and Retransmission Rate

In this subsection, we measure the end-to-end throughput and the TCP retransmission rate while varying the frame error rate (FER) (**Fig. 8**) and the PU occupation rate (**Fig. 9**). **Fig. 8** shows the end-to-end throughput and the TCP retransmission rate when the PU occupancy rate

is 50%. We observe that the DCC scheme has higher end-to-end throughputs than the TCP-SACK because it can keep larger CW values by avoiding unnecessary CW cut downs (for FER = 0% and 1%). However, this aggressive scheme increases the TCP retransmission rate and aggravates network congestion. Thus, we used the DPCW scheme to reduce the TCP retransmission rate by alleviating AP overflows. As seen in **Fig. 8**, the TCP-CR with integrated DCC and DPCW schemes shows the highest end-to-end throughput and also reduces the network congestion; this can be done simultaneously because the schemes interact orthogonally.

Table 2. Simulation Parameters

Parameter		Value
Link Bandwidth	Wired Link	10 Mbps
	CR Link	10 Mbps
Propagation Delay	Wired Link	100 ms
	CR Link	1μs
Size of TCP Window		64 Kbytes
TCP Segment Size		1 Kbytes
Frame Error Rate (FER)		Variable
Application Protocol		FTP
PU Occupation Rate		Variable

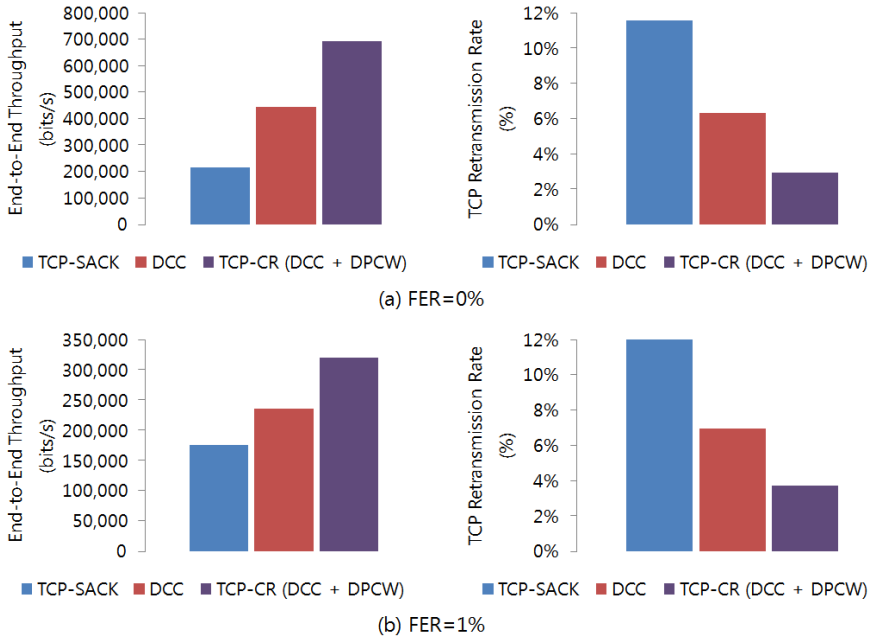

Fig. 8. The end-to-end throughputs and TCP retransmission rate (PU occupancy rate = 50%)

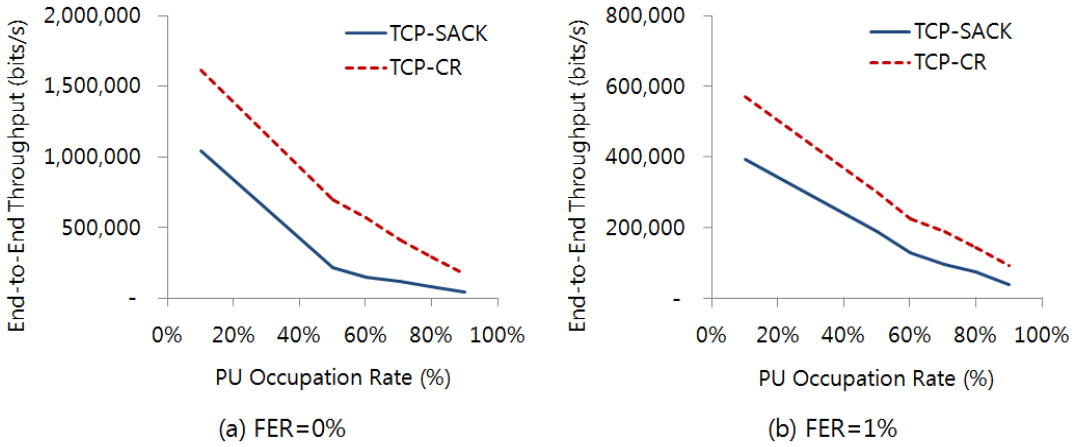

Fig. 9. The end-to-end throughputs vs. the PU occupation rate

Fig. 9 shows the end-to-end throughput versus the PU occupancy rate. The end-to-end throughputs of the TCP-CR are higher than those of the TCP-SACK because the TCP-CR prevents unnecessary congestion control and alleviates AP overflows. Although the difference between their end-to-end throughputs becomes smaller as the PU occupancy rate increases, the results demonstrate that the proposed scheme achieves substantial throughput enhancement; for example, compared to the TCP-SACK, the TCP-CR achieves 397% throughput enhancement when the PU occupancy rate is 90%, and it achieves 255% throughput enhancement on average.

5.2 Link Bandwidth Sharing

In this subsection, we consider the scenario that the bandwidth is shared by TCP-CR and TCP-SACK flows, in order to verify that the TCP-CR does not degrade the performance of existing TCPs. This is a concern because the TCP-CR is opportunistic and somewhat aggressive. We chose the experimental condition of uncongested environments. The experimental setup is shown in **Fig. 10**. **Fig. 10 (a)** shows the case in which two TCP-SACK senders (TCP-SACK$_A$ and TCP-SACK$_B$) share the bandwidth, while **Fig. 10 (b)** shows the case in which the bandwidth is shared by TCP-CR and TCP-SACK$_A$. This simulation setup is intended to determine how TCP-SACK$_A$ in **Fig. 10** experiences bandwidth sharing.

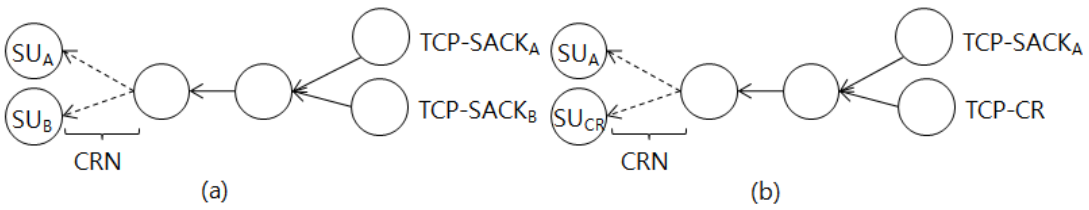

Fig. 10. Two different experiment setups for bandwidth sharing

The results are shown in **Fig. 11**. We observe similar end-to-end throughputs of TCP-SACK$_A$ (solid line) for the two cases (**Fig. 11 (a)** and **Fig. 11 (b)** for FER=1%). This means that the performance of the TCP-SACK is not affected by the presence of the TCP-CR. The TCP-CR provides better end-to-end throughput than the existing TCP in both cases, consistent with the

results of section 5.1. Therefore we see that the TCP-CR improves its performance not by preempting other TCP's bandwidth but by utilizing extra bandwidth unused by the TCP-SACK.

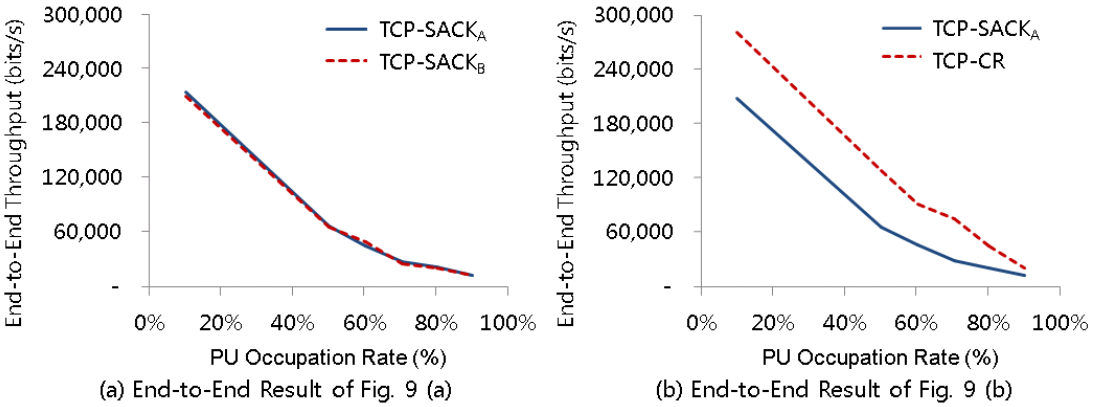

Fig. 11. Bandwidth sharing between TCP-CR and TCP-SACK (FER=1%)

5.3 Congestion Control

In this subsection, we investigate how the TCP-CR reacts when the network is congestive. To construct congested environments, we deployed additional TCP-SACK senders to cause congestion, as shown in **Fig. 12**. We added 4 pairs of TCP-SACK nodes at C(*S*) and C(*R*) to create congestion in the wired link. In this setup, the TCPs send traffic from C(*S*) to C(*R*) and perform congestion control of the TCP-SACK. We set the FER to 1% in the CRN and the bandwidth of the intermediate link to 1 Mb/s. We considered two cases as in section 5.2; i.e., the bandwidth was shared by two TCP-SACK senders (TCP-SACK$_A$ and TCP-SACK$_B$) or by TCP-CR and TCP-SACK$_A$ senders.

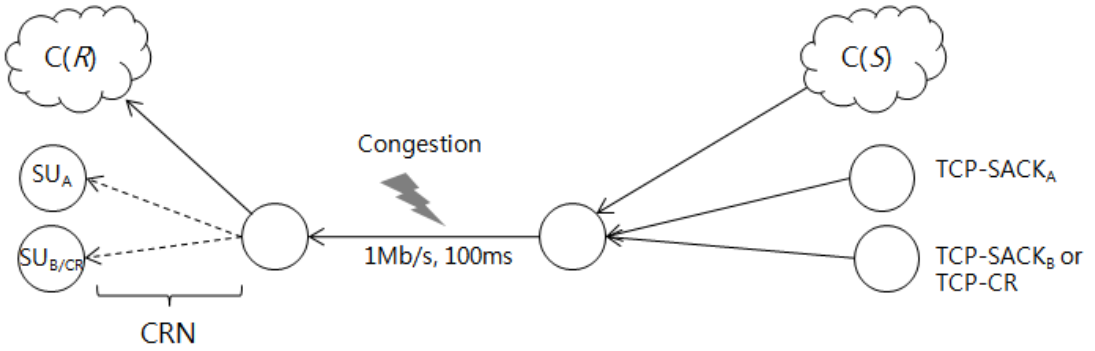

Fig. 12. Topology for fairness experiments in congested environments

The results are shown in **Table 3**. The average end-to-end throughputs of C(*S*) are similar in the two cases during the congested period. This result means that the TCP-CR does not aggravate congestion by cutting down CW in a congested period.

Table 3. Congestion Control of TCP-CR in Congested Environments (FER=1%)

Scenarios	Throughputs of TCP-CR and Existing TCP(bits/s)	Average Throughput of C(*S*) (bits/s)
$TCP\text{-}SACK_A \rightarrow SU_A$ $TCP\text{-}SACK_B \rightarrow SU_B$	($TCP\text{-}SACK_A$, $TCP\text{-}SACK_B$) (16,329, 16,825)	114,989
$TCP\text{-}SACK_A \rightarrow SU_A$ $TCP\text{-}CR \rightarrow SU_{CR}$	($TCP\text{-}SACK_A$, TCP-CR) (16,717, 17,003)	115,277

To analyze the TCP-CR congestion control mechanism in detail, we observe the variations of the CW shown in **Fig. 13**. This shows traces of the CW values of the TCP-CR and $TCP\text{-}SACK_A$. The experiment first starts the uncongested status when C(*S*) is inactive. After 300 seconds, C(*S*) generates traffic and creates congestion for 500 seconds, which causes the CWs of the TCP-CR and $TCP\text{-}SACK_A$ to decrease from about 300 to 800 seconds. This is because the TCP-CR decreases the transmission rate when the network is congested. On the other hand, the TCP-CR increases the CW and thus has a higher CW than the $TCP\text{-}SACK_A$ during uncongested periods. This can improve the end-to-end throughput of the TCP-CR compared with the TCP-SACK, enabling the TCP-CR to dynamically adapt to different network environments.

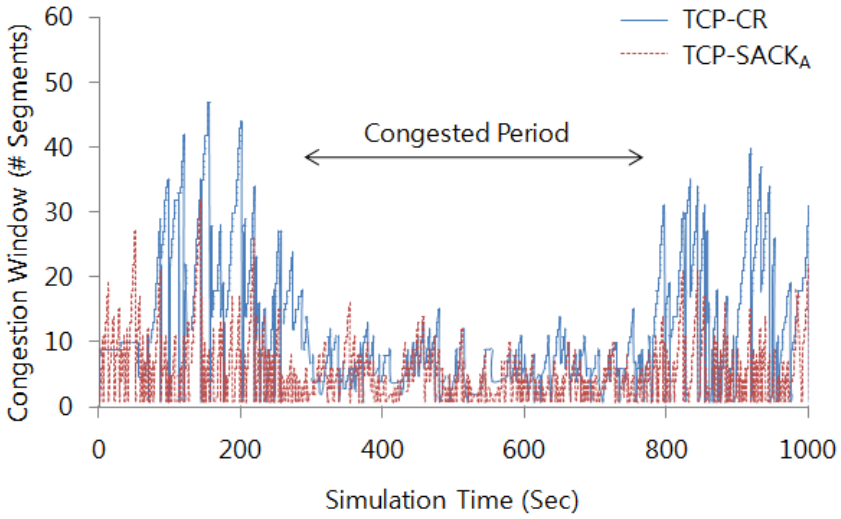

Fig. 13. Variations of congestion window with time (FER: 1%, simulation time: 1000 seconds)

6. Conclusion

Most existing TCP protocols for the CRN have assumed that either all network devices are connected by CR links or that at least the TCP sender is connected via a CR link. In this situation, lower layer information such as PU activity was easily exploited by the TCP sender. However, in more practical deployments of CR technology, the TCP sender or server is remotely located over wire-line Internet, while the TCP receiver or client accesses the wire-line Internet via CR technology. In this case, lower layer information at the TCP receiver is not helpful to the TCP sender due to the long RTT.

In this paper, we have considered a TCP network in which the TCP sender is located remotely over the Internet while the TCP receiver is connected via a CR link. For this scenario, we proposed an enhanced TCP protocol for CR networks called *TCP for cognitive radio* (TCP-CR). In the TCP-CR, the TCP sender detects PU arrival without support from lower layer protocols. Based on PU detection, we proposed two techniques: delayed congestion control (DCC) and a dual-phase congestion window (DPCW).

Performance evaluation demonstrated that the proposed TCP-CR achieves 255% improvement in end-to-end throughput on average. We also validated that the proposed TCP-CR does not deteriorate fairness, maintains existing TCP flows, and does not increase congestion.

As a first step to address the CR link problem at the remote TCP, this study has validated the solution approach through simulation experiments. Some mathematical results based on performance analysis will give more precise understanding of this problem. As a more practical approach, exploiting the behavior pattern of PU will be promising. Most CR links have some periodic properties such as spectrum sensing. Using these properties, a somewhat proactive congestion control might be possible.

References

[1] Woongsoo Na and Sungrae Cho, "Log-based Admission Control Scheme for Dynamic Spectrum Access Networks," *IEICI Transactions on Communications*, vol.E94-B, no.10, pp.2933-2936 2011. Article (CrossRef Link)

[2] Jitendra Padhye, Victor Firoiu, Donald F. Towsley, and James F. Kurose, "Modeling TCP Reno performance: A simple model and its empirical validation," *IEEE/ACM Networking*, vo.8, no.2, pp.133-145, 2000. Article (CrossRef Link)

[3] Ian F. Akyildiz, Won-Yeol Lee, and Kaushik R. Chowdhury, "CRAHNs: Cognitive radio ad hoc networks," *Elsevier Ad Hoc Networks*, vo.7, no.5, pp.810–836, 2009. Article (CrossRef Link)

[4] Alex M. R. Slingerland, Przemysław Pawełczak, R. Venkatesha Prasad, Anthony Lo, and Ramin Hekmat, "Performance of Transport Control Protocol over Dynamic Spectrum Access Links," in *Proc. of IEEE Symposium on Dynamic Spectrum Access Networks*, pp.486-495, 2007. Article (CrossRef Link)

[5] Teerawat Issariyakul, Laxminarayana S. Pillutla, and Vikram Krishnamurthy, "Tuning Radio Resource in an Overlay Cognitive Radio Network for TCP: Greed Isn't Good," *IEEE Communications Magazine*, vo.47, no.7, pp.57-63, 2009. Article (CrossRef Link)

[6] Marco Di Felice, Kaushik Roy Chowdhury, and Luciano Bononi, "Modeling and Performance Evaluation of Transmission Control Protocol over Cognitive Radio Ad Hoc Networks," in *Proc. of ACM Conf. on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pp.4-12, 2009. Article (CrossRef Link)

[7] Changqing Luo, F. Richard Yu, Hong Ji and Victor C.M. Leung, "Optimal Channel Access for TCP Performance Improvement in Cognitive Radio Networks: A Cross-Layer Design Approach," *Proc. of IEEE Conf. on Global telecommunications*, pp.2618-2623, 2009.Article (CrossRef Link)

[8] Changqing Luo, F. Richard Yu, Hong Ji, and Victor C.M. Leung, "Cross-Layer Design for TCP Performance Improvement in Cognitive Radio Networks," *IEEE Vehicular Technology*, vo.59, no.5, pp.2485-2495, 2009.Article (CrossRef Link)

[9] Kaushik R. Chowdhury, Marco Di Felice, and Ian F. Akyildiz, "TP-CRAHN: A Transport Protocol for Cognitive Radio Ad-hoc Networks," in *Proc. of IEEE Conf. on Computer Communications*, pp.2482-2490, 2009.Article (CrossRef Link)

[10] Dilip Sarkar and Harendra Naray, "Transport Layer Protocols for Cognitive Networks," *Proc. of IEEE Conf. on Computer Communications*, pp.1-6, 2010. Article (CrossRef Link)

[11] Kevin Fall and Kannan Varadhan, The ns Manual, The VINT Project.

[12] Shu Lin, Daniel J. Costello Jr., and Michael J. Miller, "Automatic-repeatrequest error-control schemes," *IEEE Communications Magazine*, vo.22, no.12, pp.5-17, 1994.
Article (CrossRef Link)
[13] Carlos Cordeiro, Kiran Challapali, and Dagnachew Birru, "IEEE 802.22: An Introduction to the First Wireless Standard based on Cognitive Radios," *Communications*, vol.1, no.1, pp.38-47, 2006.
Article (CrossRef Link)
[14] Saverio Mascolo, Claudio Casetti, Mario Gerla, M. Y. Sanadidi, and Ren Wang, "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links," *in Proc. of ACM Conf. on Mobile Computing and Networking (MOBICOM)*, pp.287-297, 2001.
Article (CrossRef Link)

Hyun Yang received the B.S., M.S., and Ph.D. degrees in computer engineering from Chung-Ang University, Seoul, Korea in 2006, 2008, and 2012, respectively. He is working at the Korea Institute of Ocean Science and Technology (KIOST). His research interests include cognitive radio networks, wireless sensor networks, satellite networks, reliable wireless multicast, and energy-efficient communication."

Sungrae Cho received his B.S. and M.S. degrees in Electronics Engineering from Korea University, Seoul, Korea, and his Ph.D. degree in Electrical and Computer Engineering from the Georgia Institute of Technology, Atlanta, GA in 1992, 1994, and 2002, respectively. He is currently a professor in Computer Science and Engineering at Chung-Ang University, Seoul, Korea. Prior to joining Chung-Ang University, he was an assistant professor in Computer Science at Georgia Southern University, Statesboro, GA from 2003 to 2006. In 2003, he was a principal research engineer at Samsung Advance Institute of Technology, Keung, Korea. From 1994 to 1996, he was a member of research staff at Electronics and Telecommunications Research Institute, Daejon, Korea. He is a editor of Elsevier Ad Hoc Networks Journal, Journal of Korea Information and Community Society, and Journal of Korean Institute of Information Scientists and Engineers. He also served an organizing committee member or a TPC member of numerous international conferences including IEEE SECON 2012, ICOIN 2010, 2011, 2012, ICUFN 2010, 2011, 2012, ICTC 2011, 2012, IEEE MASS2009, CSIE 2009, IEEE Tridentcom 2007, and IEEE MADWH 2005.

Chang Y. Park received the B.S. and M.S. degrees in computer engineering from Seoul National University, Seoul, Korea in 1984 and 1986, respectively, and received the Ph.D. in computer science from the University of Washington, Seattle in 1992. He is currently a Professor in the School of Computer Science and Engineering, ChungAng University, Seoul, Korea. His research interests include computer networks, and real-time systems.