J Inf Process Syst, Vol.12, No.1, pp.73~82, March 2016
http://dx.doi.org/10.3745/JIPS.03.0041

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

An Experimental Implementation of a Cross-Layer Approach for Improving TCP Performance over Cognitive Radio Networks

Sang-Seon Byun*

Abstract

In cognitive radio networks (CRNs), the performance of the transmission control protocol (TCP) at the secondary user (SU) severely drops due to the mistrigger of congestion control. A long disruption is caused by the transmission of primary user, leading to the mistrigger. In this paper, we propose a cross-layer approach, called a CR-aware scheme that enhances TCP performance at the SU. The scheme is a sender side addition to the standard TCP (i.e., TCP-NewReno), and utilizes an explicit cross-layer signal delivered from a physical (or link) layer and the signal gives an indication of detecting the primary transmission (i.e., transmission of the primary user). We evaluated our scheme by implementing it onto a software radio platform, the Universal Software Radio Peripheral (USRP), where many parts of lower layer operations (i.e., operations in a link or physical layer) run as user processes. In our implementation, we ran our CR-aware scheme over IEEE 802.15.4. Furthermore, for the purpose of comparison, we implemented a selective ACK-based local recovery scheme that helps TCP isolate congestive loss from a random loss in a wireless section.

Keywords

Cognitive Radio Networks, Congestion Control, TCP, USRP

1. Introduction

Ten years has passed since the concept of a cognitive radio network (CRN) [1] was proposed in order to solve the problem of inefficient spectrum utilization. In CRNs, while primary users (PUs) are licensed to access some frequency bands that are inactive, secondary users (SUs) are allowed to transmit over these frequency bands. To this end, SUs should be equipped with a cognitive radio (CR) capability that can sense and occupy an unused spectrum, and release it when the PU that is licensed to access the spectrum starts its transmission.

The majority of the research work related to CRNs have focused on the issues in physical or link layers (i.e., spectrum sensing, dynamic spectrum allocation, interference control, etc.). Issariyakul et al. [2] studied the performance degradation of a transmission control protocol (TCP) in overlay CRNs (In overlay CRNs, SUs can access the portion of the spectrum that is not used by any PUs. As a result, there is virtually no interference with the PUs. Henceforth, CRN implies an overlay CRN in this paper) where an

Manuscript received September 17, 2014; accepted January 21, 2015; onlinefirst December 11, 2015.
Corresponding Author: Sang-Seon Byun (ssbyun@cup.ac.kr)
* Dept. of Computer Engineering, Catholic University of Pusan, Busan, Korea (ssbyun@cup.ac.kr)

SU's transmission is entirely disrupted whenever a PU starts its transmission until the SU handoffs to other unused spectrum or when the PU finishes the transmission. In this situation, the SU's TCP suffers from a severe performance drop due to a long round-trip time (RTT) and consecutive retransmission timeouts (RTOs). Especially, the consecutive RTOs continuously hinder the progress of the congestion window since the congestion window always squeezes to 1 on every RTO. Furthermore, the retransmission timer (RT) is backed off exponentially in the standard TCP (i.e., TCP-NewReno) on every RTO.

In this paper, we propose a cross-layer scheme that resolves the above-mentioned performance degradation in overlay CRNs. Our scheme is only implemented at the secondary transmitter (ST) and does not require any changes at the secondary receiver (SR). We let the lower layer protocol send the overlying TCP an explicit signal indicating the start and stop of the PU's transmission. Then, we modified the TCP-NewReno so that the TCP stops its transmission after saving its current sending state upon receiving the signal indicating the start of the PU's transmission, and resumes transmitting upon receiving the signal indicating that the PU has stopped transmitting.

We evaluated our scheme by implementing it on a software-defined radio platform, the Universal Software Radio Peripheral (USRP). We also measured the performance gain when our scheme is assisted by a link-layer loss recovery scheme, that is, link-layer selective-ACK.

The rest of this paper is organized as follows: in Section 2, we presented the problems that the TCP has over CRNs and other related research efforts. In Section 3, we present our scheme of a CR-aware TCP via a cross-layer approach. In Section 4, we give the implementation details. In Section 5, we present the results of the evaluations. Finally, we conclude this paper in Section 6.

2. TCP over CRNs

2.1 TCP over Wireless Networks

In the Internet, the TCP provides the application with reliable and ordered packet delivery over unreliable physical media. Therefore, TCP is the most widely used transport layer protocol even in wireless networks, and there is no sign of change occurring in the foreseeable future.

The most popular TCP variant (i.e., TCP-NewReno [3]) regards packet loss as the signal of network congestion. Two different signals are used for noticing the packet loss, namely, triple duplicate acknowledgements (ACKs), and RTOs. Upon receiving triple duplicate ACKs, the TCP performs a fast retransmission and triggers a congestion avoidance mechanism by halving its congestion window size (henceforth, we denote the congestion window size as *cwnd*). On the occurrence of an RTO, it retransmits the unacknowledged packet. At the same time, it reduces the *cwnd* to 1 and triggers the slow start deeming the network overloaded.

There has been extensive research on resolving the problem of a standard TCP over wireless networks; multiple random packet losses within one RTT, due to interference, shadowing, fading, and contention in wireless channels and other factors (such as buffer overflow in the NIC (network interface card), lead to consecutive RTOs and an exponential back-off of the RT. Therefore, the standard TCP suffers from a drastic decrease of its throughput [4]. The main ideas behind the schemes that tackle the above-mentioned problem are 1) to reduce the impact of random packet loss on the TCP by local recovery and 2) to differentiate between random packet loss and congestive packet loss.

2.2 Cross-Layer Implementation

Unlike legacy-layered approaches, the cross-layer technique allows higher layer protocols to react more adaptively and agilely to changes in lower layers. In a similar way, lower layers are able to adjust their parameters using the information given by higher layers. For instance, in CRNs, the TCP in the SU side can respond more agilely to channel availability and capacity via cross-layer signaling from the physical layer and the physical layer adjusts the modulation and coding/decoding methods for providing the TCP with better goodput.

Recently, a few approaches [5-8] have been proposed for improving TCP performance in CRNs through cross-layering. However, all of them are evaluated via simulations with somewhat optimistic and convenient assumptions. For instance, perfect knowledge on the channel state [5], error-free control channel [7,8], statistical knowledge on the behavior of PU's transmission [6-8] and static channel gain during a single RTT [5]. For all of that, it is still challenging to implement cross-layer schemes in general computer systems since it is inevitable to manipulate the TCP stack and device driver in an operating system. In [9,10], the authors have implemented cross-layer approaches to improve TCP performance over multi-hop networks and their approaches were implemented in the Linux kernel and MadWiFi driver.

Lately, several software-defined radio (SDR)-based experimental platforms and testbeds for CRNs have shown up in the research field, and have been paid attention to by researchers as well as practitioners. In [11,12], the authors have reviewed the most popular SDR platforms and introduced exemplary CRNs. Mainly, those SDR-based platforms are designed to carry out the majority of lower layer operations in software with minimal hardware RF front-end. Hence, it is possible to achieve cross-layer implementation on these platforms more conveniently since lower layers, as well as the TCP, can be implemented as user processes, which enables the easy implementation of cross-layer signaling.

In this paper, we propose a cross-layer scheme that enhances TCP performance over CRNs, and provide a practical implementation on an SDR platform, the USRP. In USRP, the main signal processing blocks are implemented with C++ and their flow graphs are designed with Python. In our implementation, we utilized 802.15.4 implementation [13] as a lower layer wireless standard. Then we implemented a standard TCP (i.e., TCP-NewReno) and our enhancement scheme, and coupled them with the 802.15.4 implementation. Additionally, we implemented a link layer recovery scheme (henceforth, referred to as the LR scheme) that improves TCP performance against random packet loss.

The main reason of choosing IEEE 802.15.4 as the target lower layer is that, at first, it works perfectly on our software radio platform (USRP E100), and there are many issues related to the coexistence of IEEE 802.15.4 with other major protocols operating on unlicensed 2.4 GHz ISM band, notably IEEE 802.11 (WLAN) and IEEE 802.15.1 (Bluetooth) [14-16]. Furthermore, it is not unusual to address reliable data transmission on IP-enabled small devices that use IEEE 802.15.4 [17-20].

3. CR-Aware TCP via Cross-Layer Approach

In CRNs, the standard TCP at the SU side suffers from the performance degradation due to not only the random packet loss but also unavailable channel access; SU's transmission is disrupted by the PU's

transmission if both the SU and PU access the same channel. However, the standard TCP refers the disruption to the signal of network congestion. Furthermore, the random loss occurs transiently while the disruption by the PU's transmission lasts for a long while. Therefore, unless the SU handoffs to other unused spectrums, the disruption incurs more consecutive RTOs and an exponential back-off of the RT. In detail, if the disruption incurs RTO at the SU, and if it continues until the next RT expires, the RT is backed off exponentially twice. Thus, it is highly probable that the SU does not proceed with its transmission even after the disruption is over. Our experimental results show that TCP performance is deteriorated more severely by the disruption than the random loss.

We consider an overlay CRN with a single frequency band in this paper. The physical layer in the SU side should have the capability of detecting the PU's transmission, and should cease with its transmission upon the detection in order not to interfere with the PU's transmission. In this situation, we envisaged a cross-layer signal that indicates the detection of the PU's transmission. This cross-layer signal is delivered to the CR-aware module that is a sender side addition to a standard TCP. We call our standard TCP equipped with the CR-aware capability, *CR-aware TCP*. Our scheme requires no modifications at the receiver side.

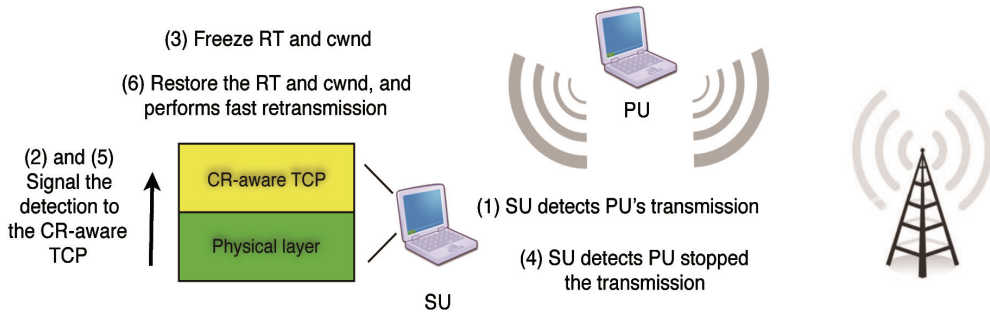

Fig. 1. Illustration of the cross-layer signaling with CR-aware TCP.

The brief behavior of the CR-aware TCP is presented as (also refer to Fig. 1):

1) When the physical layer at the SU detects the PU's transmission, it gives the explicit cross-layer signal to the TCP layer.

2) Upon receiving the cross-layer signal, the TCP layer cancels the RTO and freezes its *cwnd* and RT value.

3) The SU periodically overhears the channel in order to detect whether the PU is still transmitting or not.

4) If no transmission is detected for a certain predefined duration (henceforth, we denote this duration as T_{idle}), the physical layer delivers the cross-layer signal to the TCP.

5) On receiving the cross-layer signal, the TCP layer resets the RTO with the frozen RT value, and resumes transmission with restoring the frozen *cwnd*.

Even though the secondary receiver receives packets successfully, acknowledgements can be lost by the disruption. In this case, the SU's transmission is delayed until the next RTO, even after the disruption ends since only an outstanding acknowledgement can increase the *cwnd*. In order to avoid this situation, the CR-aware TCP immediately retransmits packet whose acknowledgment is pending.

4. Implementation Details

4.1 The Platform

We implemented our CR-aware TCP on USRP E100, which is a standalone version of USRPs, and its main hardware consists of a TI OMAP Beagle Board and ARM cortex A8 core. It can hold one RF daughter board that plays the role of the RF front-end. Besides, it holds one FPGA chip where the DAC/ADC and interpolator/decimator are installed, and its default operating system is an embedded Linux.

GNURadio and the USRP Hardware Driver (UHD) are widely used radio software for USRP. GNURadio is an open source project (http://www.gnuradio.org), and supports hardware-independent signal processing functionalities. In GNURadio, signal-processing blocks are written in C++, while signal flow interfaces are built using Python. UHD, which was developed by Ettus Research LLC (the manufacturer of the USRP), can work with or replace GNURadio, and its signal processing blocks are exclusively optimized for USRP.

In USRP, most of the signal-processing operations are executed on a general-purpose processor (GPP). Therefore, its performance is much worse than a general radio system where all the signal-processing operations are handled by dedicated DSPs. However, this device sufficiently fits the purpose of evaluating prototypical protocols.

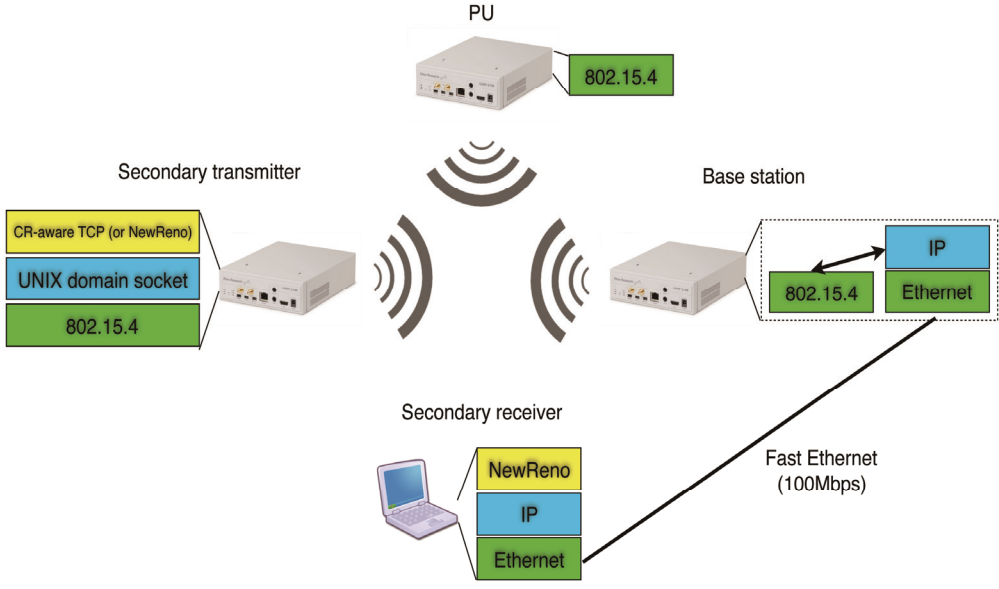

Fig. 2. Experimental topology.

As shown in Fig. 2, we configured experimental topology deploying three USRPs: one for a ST, another for a PU, and the other for a base station (BS). We let the ST and the BS communicate through 802.15.4 with each other, and the BS communicate with the SR through the IP raw socket over the Ethernet since USRP E100 supports Ethernet connection as well. In order to minimize the congestive loss in the wired path, we located both the SR and BS in the same subnet.

4.2 Implementation of CR-Aware TCP

Like other software radio modules implemented for USRP, the 802.15.4 implementation is written in C++ and Python for main blocks and block interfacing, respectively, and runs in a user space as a single Python process. We added the Python code, which implements the functionality of detecting and signaling the start and end of PU's transmission to the 802.15.4 implementation. Additionally, we made the PU perform its transmission using 802.15.4 just as the ST does. Then, by letting the ST know the hardware address of the PU in advance, the ST can detect the PU's transmission simply by decoding the address field of incoming 802.15.4 frames. By doing this, we made it possible for the performance of the TCP to not be affected by the performance of the spectrum sensing mechanism.

At first, we implemented TCP-NewReno in C and added the code of the CR-aware scheme. Our TCP implementation also runs in a user space as a single process. For inter-process communication (IPC) between the TCP process and the Python process (i.e., for the deliveries of user data, acknowledgements, and cross-layer signals), we used the Unix Domain Socket. We did not implement a flow control mechanism and IP layer for removing the factors that might influence the performance of the congestion control algorithm. Furthermore, we let the TCP process itself generate user data.

Furthermore, we implemented a selective ACK (SACK)-based LR scheme [21] for improving TCP performance against random loss. Each SACK contains information about up to three non-contiguous data blocks that have been received successfully by the receiver and its starting and ending sequence number describe each block of data. In our implementation, the 802.15.4 module in the BS snoops the sequence number of incoming TCP packets and transmits the SACK to the ST whenever it detects a packet loss. On receiving the SACK, the ST immediately retransmits packets marked as 'lost' in the SACK.

In this paper, we envisage two different implementations of the SACK-based LR scheme according to layers where the retransmission function is implemented. For the following general SACK-based LR scheme, a retransmission function can be added in 802.15.4 implementation (henceforth, we call this LR scheme PHY-LR). It is also possible to provide TCP implementation with a retransmission function (henceforth, we call this LR scheme TCP-LR) and we only implemented the retransmission module in the TCP. The PHY-LR scheme is preferred in terms of fast responsiveness, and the TCP-LR scheme is preferred in terms of less buffer usage since it shares the buffer with a standard TCP.

5. Performance Evaluations

5.1 Platform Settings

We located the USRP of the ST and the USRP of the BS around seven meters away from each other. Moreover, in order to degrade the channel state moderately, we put them out of the line of sight. Then we located the USRP of the PU 5 m away from the ST, but in the line of sight. We let all the USRPs transmit with fixed transmission power, that is, 1 dB. All of the USRPs were allocated with 2 Mbps of the physical bandwidth and 200 kbps of the sampling rate. We let all of the transmissions perform over the central frequency of 2.1 GHz.

The size of MAC protocol data unit (MPDU) is given as 128 bytes, including 19 bytes of header and 2

bytes of tail. Therefore, 107 bytes are reserved for the payload. We allocated 12 bytes for the header of our TCP implementation and 4 bytes and 8 bytes for the sequence number and RTT, respectively. Therefore, each MDPU can carry 95 bytes of user data.

We let the PU generate 802.15.4 packets for an exponentially distributed time with the mean of 4 seconds, and ceased transmission for an exponentially distributed time with the mean of 4 seconds. We let T_{idle} = 1 s. If no transmission from the PU is overheard for at least 1 second, the ST resumes transmission.

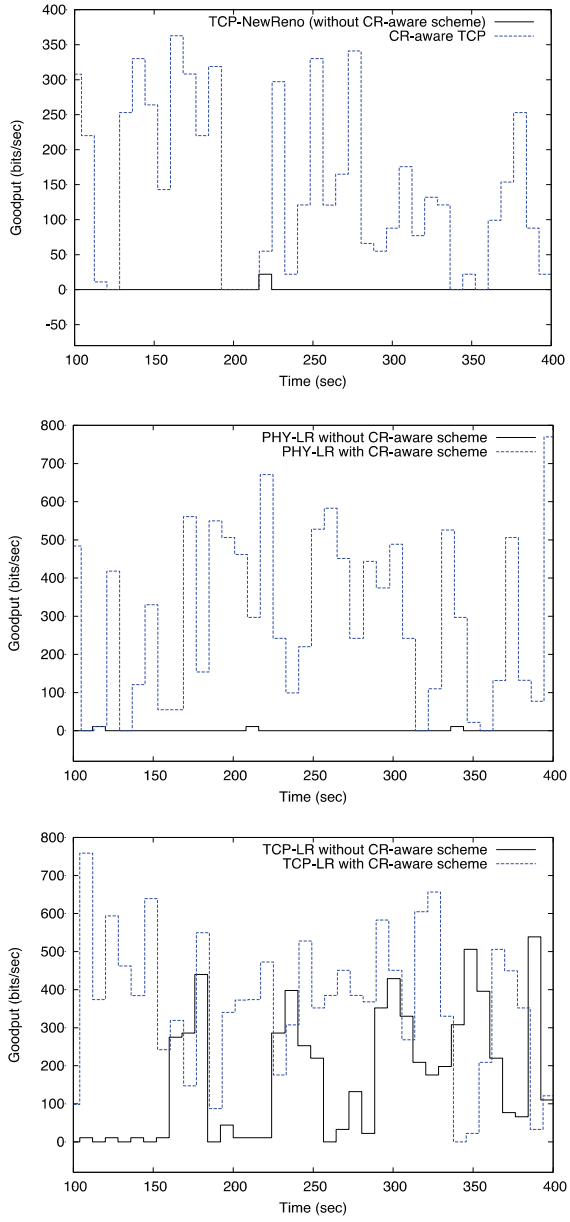

Fig. 3. Comparison of the goodput measured in each TCP implementation. (a) TCP-NewReno, (b) PHY-LR scheme, and (c) TCP-LR scheme.

We evaluated the performance of our CR-aware scheme over three different TCP implementations: TCP-NewReno, PHY-LR scheme, and TCP-LR scheme. The performance of each implementation was measured for 600 seconds.

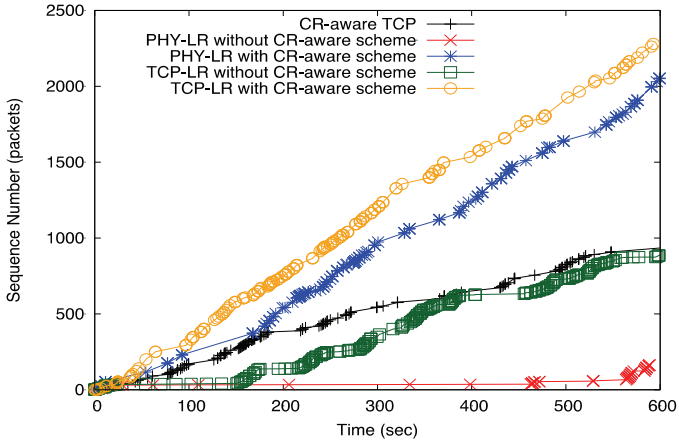

Fig. 4. Trace of the packet sequence.

First, we plotted the goodput of each TCP implementation in Fig. 3. As shown in these graphs, it is observed that significantly better goodput is achieved on every TCP implementation when the CR-aware scheme is deployed together. Especially, we noted that the TCP-NewReno and PHY-LR scheme, where no CR-aware scheme is used, yield very poor performance. Over the course of 600 seconds, they only succeeded in delivering 26 packets (TCP-NewReno) and 163 packets (PHY-LR scheme), respectively. As described earlier, this is due to the fact that TCP-implementations with no CR-aware scheme continue to transmit or retransmit the user data, even though the physical channel is blocked. As a result, they trigger consecutive RTOs (with squeezing *cwnd* to 1) and postpone the transmission until the next RTO, even when the channel becomes available for transmission.

We also observed that all LR schemes, even with a CR-aware scheme, trigger RTOs frequently, which results from the loss of feedback packets (i.e., SACK or TCP acknowledgement), as well as the loss of retransmitted packets. Furthermore, it has been observed that the TCP-LR scheme (with no CR-aware function) yields better goodput than other schemes with no CR-aware scheme. This is due to the fast retransmission of packets marked as 'lost' in SACKs whenever the RTO is triggered. In the implementation of the TCP-LR scheme, we let all lost packets—indicated by SACK and TCP acknowledgment—be retransmitted on every RTO. Finally, we observed poor goodput over all the TCP implementations (at most up to 780 kB/s). This is, as we previously mentioned, due to the poor performance of the USRP.

In order to study the contribution of a CR-aware scheme to TCP performance more clearly, we traced the packet sequence of each implementation and plotted it in Fig. 4. We did not plot the trace for the TCP-NewReno with any CR-aware scheme since its trace cannot be discerned on the graph, due to its poor performance. As shown in the graph, the TCP-LR with the CR-aware scheme yields the best performance, and the PHY-LR with the CR-aware scheme achieves the second best performance. It is also worth addressing that the TCP-NewReno with the CR-aware scheme slightly outperforms the TCP-LR (with no CR-aware scheme), which implies disruption. This is due to the PU's transmission

that makes the TCP perform more poorly performance poorer than random packet loss does (This is also dependent on the frequency and duration of the PU's transmission). Therefore, it is necessary to consider the disruption that occurs due to the PU's transmission when designing the TCP over CRNs.

6. Concluding Remarks

In this paper, we considered a cross-layer approach to enhance the TCP performance of the SU in CRNs. We provided cross-layer signals between the physical layer and the TCP. The signals indicate the start or stop of the PU's transmission. Upon receiving the cross-layer signal indicating the start of the PU's transmission, the CR-aware TCP freezes its RTO and *cwnd*, and stops the transmission. Right after the signal is given that the PU has stopped transmitting, the CR-aware TCP resumes the transmission by restoring the frozen RTO and *cwnd*. We implemented our cross-layer scheme on an SDR platform, USRP, where lower layer protocols and the TCP are implemented using high-level programming languages (i.e., C/C++ and Python) and are run as user processes. Therefore, the cross-layering can be realized by simply providing an IPC mechanism. Furthermore, we implemented a selective ACK-based LR-scheme to enhance the performance over a random loss in a wireless channel. We deployed the 802.15.4 implementation as a lower layer protocol. After a series of experiments, we reached the conclusion that in CRNs it is essential to design the TCP at the SU side by giving more consideration to the impact of the PU's transmission than to the impact of random loss.

References

[1] J. Mitola and G. Q. Maguire, "Cognitive radio: making software radios more personal," *IEEE Personal Communications*, vol. 6, no. 4, pp. 13-18, 1999.

[2] T. Issariyakul, L. S. Pillutla, and V. Krishnamurthy, "Tuning radio resource in an overlay cognitive radio network for TCP: greed isn't good," *IEEE Communications Magazine*, vol. 47, no. 7, pp. 57-63, 2009.

[3] S, Floyd and T. Henderson, "The NewReno modification to TCP's fast recovery algorithm," RFC 2582, 1999; https://tools.ietf.org/html/rfc2582.

[4] Y. Tian, K. Xu, and N. Ansari, "TCP in wireless environments: problems and solutions," *IEEE Communications Magazine*, vol. 43, no. 3, pp. S27-S32, 2005.

[5] D. Chen, H. Ji, and V. Leung, "Distributed optimal relay selection for improving TCP throughput over cognitive radio networks: a cross-layer design approach," in *Proceedings of 2011 IEEE International Conference on Communications (ICC),* Kyoto, Japan, 2011, pp. 1-5.

[6] C. Luo, F. R. Yu, H. Ji, and V. Leung, "Cross-layer design for TCP performance improvement in cognitive radio networks," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 5, pp. 2485-2495, 2010.

[7] K. R. Chowdhury, M. Di Felice, and I. F. Akyildiz, "TP-CRAHN: a transport protocol for cognitive radio ad-hoc networks," in *Proceedings of IEEE Conference on Computer Communication (INFOCOM)*, Rio de Janeiro, Brazil, 2009, pp. 2482-2490.

[8] K. R. Chowdhury, M. Di Felice, and I. F. Akyildiz, "TCP CRAHN: a transport control protocol for cognitive radio ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 12, no. 4, pp. 790-803, 2013.

[9] A. Warrier, S. Janakiraman, S. Ha, and I. Rhee, "DiffQ: Practical differential backlog congestion control for wireless networks," in *Proceedings of IEEE Conference on Computer Communication (INFOCOM)*, Rio de Janeiro, Brazil, 2009, pp. 262-270.

[10] S. M. ElRakabawy and C. Lindemann, "A practical adaptive pacing scheme for TCP in multihop wireless networks," *IEEE/ACM Transactions on Networking*, vol. 19, no. 4, pp. 975-988, 2011.

[11] K. R. Chowdhury and T. Melodia, "Platforms and testbeds for experimental evaluation of cognitive ad hoc networks," *IEEE Communications Magazine,* vol. 48, no. 9, pp. 96-104, 2010.

[12] P. Pawelczak, K. Nolan, L. Doyle, S. W. Oh, and D. Cabric, "Cognitive radio: ten years of experimentation and development," *IEEE Communications Magazine*, vol. 49, no. 3, pp. 90-100, 2011.

[13] T. Schmid, "GNU radio 802.15.4 en- and decoding," 2006; http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.179.31.

[14] J. H. Hauer, V. Handziski, and A. Wolisz, "Experimental study of the impact of WLAN interference on IEEE 802.15. 4 body area networks," in *Wireless Sensor Networks*. Heidelberg: Springer, 2009, pp. 17-32.

[15] S. P. Chepuri, R. De Francisco, and G. Leus, "Performance evaluation of an IEEE 802.15. 4 cognitive radio link in the 2360-2400 MHz band," in *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC),* Cancun, Mexico, 2011, pp. 2155-2160.

[16] M. Timmers, S. Pollin, A. Dejonghe, L. Van der Perre, and F. Catthoor, "Exploring vs exploiting: enhanced distributed cognitive coexistence of 802.15. 4 with 802.11," in *Proceedings of IEEE Sensors,* Lecce, Italy, 2008, pp. 613-616.

[17] A. Ayadi, P. Maillé, and D. Ros, "TCP over low-power and lossy networks: tuning the segment size to minimize energy consumption," in *Proceedings of 2011 4th IFIP International Conference on New Technologies, Mobility and Security (NTMS),* Paris, 2011, pp. 1-5.

[18] A. Dunkels, T. Voigt, and J. Alonso, "Making TCP/IP viable for wireless sensor networks," in *Proceedings of 1st European Workshop on Wireless Sensor Networks (EWSN) work-in-progress session*, Berlin, Germany, 2004, pp. 1-4.

[19] A. Dunkels, "Full TCP/IP for 8-bit architectures," in *Proceedings of the 1st International Conference on Mobile Systems, Applications and Services (MobiSys'03),* San Francisco, CA, 2003, pp. 85-98

[20] F. L. Piccolo, D. Battaglino, L. Bracciale, A. Bragagnini, M. S. Turolla, and N. B. Melazzi, "On the IP support in IEEE 802.15. 4 LR-WPANs: self-configuring solutions for real application scenarios," in *Proceedings of 2010 the 9th IFIP Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net),* Juan Les Pins, France, 2010, pp. 1-10.

[21] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 756-769, 1997.

Sang-Seon Byun http://orcid.org/0000-0002-5579-1378

He received B.S., M.S., and Ph.D. degrees in Computer Science from Korea University, Seoul, South Korea in 1996, 2002, and 2007, respectively. He was an assistant research professor of the Graduate School of Embedded Software, Korea University, in 2007. From 2007 to 2012, he has served as postdoctoral researcher and research scientist of the Department of Electronics and Telecommunications, Norwegian University of Science and Technology, Trondheim, Norway. He has also worked as a research professor in the School of Information and Communications, Gwanju Institute of Science and Technology (GIST), Gwangju, South Korea, and a senior researcher of Deagu-Gyeongbuk Medical Innovation Foundation (DGMIF), Daegu, South Korea. Currently, he is an assistant professor of Department of Computer Engineering, Catholic University of Pusan, Busan, South Korea. His research interests include the fields of cognitive radio networks and software-defined radio architecture.