International Journal of Internet, Broadcasting and Communication Vol.16 No.1 29-35 (2024)
http://dx.doi.org/10.7236/IJIBC.2024.16.1.29

IJIBC 24-1-4

Improving the TCP Retransmission Timer Adjustment Mechanism for Constrained IoT Networks

Chansook Lim

Professor, Department of Software and Communications Engineering
Hongik University, Sejong, Republic of Korea
chansooklim@hongik.ac.kr

Abstract

TCP is considered as one of the major candidate transport protocols even for constrained IoT networks..In our previous work, we investigated the congestion control mechanism of the uIP TCP. Since the uIP TCP sets the window size to one segment by default, managing the retransmission timer is the primary approach to congestion control. However, the original uIP TCP sets the retransmission timer based on the fixed RTO, it performs poorly when a radio duty cycling mechanism is enabled and the hidden terminal problem is severe. In our previous work, we proposed a TCP retransmission timer adjustment scheme for uIP TCP which adopts the notion of weak RTT estimation of CoCoA, exponential backoffs with variable limits, and dithering. Although our previous work showed that the proposed retransmission timer adjustment scheme can improve performance, we observe that the scheme often causes a node to set the retransmission timer for an excessively too long time period. In this work, we show that slightly modifying the dithering mechanism of the previous scheme is effective for improving TCP fairness.

Keywords: *TCP, IoT Network, Cooja, Congestion Control*

1. Introduction

For the past decade, there have been several studies on TCP for constrained IoT networks. Hurni et al. attempted to improve TCP performance by examining the effect of distributed caching and local retransmission strategies [1]. Their main idea is that each intermediate node caches TCP segments and retransmits a segment whose ACK is considerably delayed, based on RTT estimation. Kim et al. conducted an experimental study on performance of TCP over RPL in an IPv6-based testbed using the TinyOS BLIP stack, one LBR and one Linux-based server [2]. They find that TCP incurs significant throughput unfairness among nodes in multihop LLNs and that RPL may adversely affect TCP performance when RPL is unable to balance traffic load. Park et al. attempted to address the TCP fairness problem among LLN endpoints, by proposing TAiM (TCP assistant in the middle) [3]. TaiM intervenes in the middle of TCP communication only at LBR, and manipulates RTT

Manuscript Received: December. 10, 2023 / Revised: January. 4, 2024 / Accepted: January. 13, 2024
Corresponding Author: chansooklim@hongik.ac.kr
Tel: +82-44-860-2549, Fax: +82-44-865-0460
Professor, Department of Software and Communications Engineering, Hongik University, Sejong, Korea

of the passing flows so that a flow with low throughput can have shorter delay whereas a flow with high throughput may have longer delay. Gomez et al. provide recommendations for lightweight TCP implementation and suitable operation in IoT scenarios [4]. Kumar et al. proposed a full-scale TCP, called TCPlp that can fit well within CPU and memory constraints of modern wireless sensor network(WSN) platforms leveraging the full-featured TCP in FreeBSD OS [5]. The authors deal with the issues related to hidden terminals and radio duty cycling (RDC). To reduce hidden-terminal-induced packet loss for TCP, they propose adding delay between link-layer retransmissions. Also, to overcome poor interaction of TCP's self-clocking with the duty-cycled link, they proposed Adaptive Duty Cycling.

In our previous work, we tried to improve TCP performance in a constrained network environment where an RDC mechanism is used and the hidden terminal problem is severe [7]. We proposed a scheme for managing the retransmission timer which adopts the notion of weak RTT estimation of CoCoA, exponential backoffs with variable limits, and dithering. Although our previous work showed performance improvement of the uIP TCP, we found that the TCP retransmission timer is set too long particularly when retransmission of the same packet fails multiple times. In this work, we attempt to improve the TCP retransmission timer adjustment mechanism by making a slight change to the dithering mechanism in our previous scheme. Simulation results show that this modification can improve TCP fairness without reducing TCP goodput.

2. Background and Issues

2.1 Background

In this section, we describe the congestion control mechanism in the original uIP TCP/IP [8, 9], ContikiMAC duty cycling mechanism [10] and our previous work [7] briefly.

The uIP TCP/IP stack is a small implementation of the TCP/IP protocol suite for embedded systems. The uIP TCP allows only a single TCP segment per connection to be unacknowledged at any given time. Even when each node generates low-rate traffic, congestion can occur at areas where traffic is concentrated. The uIP TCP needs congestion control by RTO management. Contiki OS supports TCP using uIP TCP/IP stack.

RDC is a major tool to save battery power in wireless sensor networks. Contiki OS that we use for this study employs ContikiMAC as the default RDC mechanism. To switch off the RDC mechanisms, Contiki OS provides the NullRDC that leaves the radio always on. ContikiMAC is a sender-initiated asynchronous RDC mechanism that does not use signaling messages and additional packet headers. ContikiMAC uses a power efficient wake-up mechanism with a set of timing constraints to allow devices to keep their transceivers off. To give an indication of radio activity on the channel, ContikiMAC wake-ups use a Clear Channel Assessment (CCA) mechanism that uses Received Signal Strength Indicator (RSSI) of radio transceiver. If a receiver detects a packet transmission during a wake-up, the receiver keeps on to be able to receive the packet. However, it has been pointed out that asynchronous RDC mechanisms may result in long delays that accumulate over multihop path [12, 13].

The Cooja network simulator is a Java-based application which was designed for simulation in the domain of wireless sensor networks [14, 15]. One of the main features of Cooja is the capability to emulate various motes constituting the wireless sensor networks. The emulation mechanism allows Cooja to perform fine-grained, precise and low-level simulations. Cooja supports a GUI interface to design, run, and analyze WSN simulations.

Our previous work showed that the original uIP TCP performs poorly due to its fixed RTO when an RDC mechanism is enabled and the hidden terminal problem is severe [7]. The original uIP TCP uses the fixed

RTO(3sec) for retransmissions after the first transmission. When RDC and hidden terminals jointly cause large RTT variations, the fixed RTO does not work well. To address the problem, we proposed a scheme for managing the retransmission timer which adopts the notion of weak RTT estimation of CoCoA, exponential backoffs with variable limits, and dithering. Weak RTT estimation means RTT estimation using retransmitted messages as well. The previous work showed effectiveness of the scheme through Cooja simulation varying the network size, the random loss rate, the link layer queue size [7]. In our another work, we investigated the effect of the traffic intensity, the degree of the hidden terminal as well by varying the offered load, the transmission range, and the RDC channel check rate [16].

2.2 Issues to be Handled

Although our previous work demonstrated that TCP performance can be improved through weak RTT estimation, exponential backoffs with variable limits and dithering, we observe that the previous scheme often makes a node wait too long before timeout. When a node repeatedly retransmits and loses the same packet, the node sets the retransmission timer for an excessively long period of time, which makes the node suffer severe degradation of performance. Figure 1 shows such cases. Figure 1 shows the change of RTT, RTO, the actual retransmission timer and the retransmission counter for two nodes. In Figure(a), the node is 4 hops away from the LBR, and in Figure 1(b), the node is within the transmission range of the LBR. In Figure 1(a), we can see that the node performs 6 retransmissions for the same packet and therefore cannot send a new packet for 200 seconds. Figure 1(b) shows that even a node which is near the LBR cannot send a new packet while performing 5 retransmissions at least for 110 seconds. These examples imply that we need some mechanism that can prevent setting the TCP retransmission timer for an excessively long period of time.

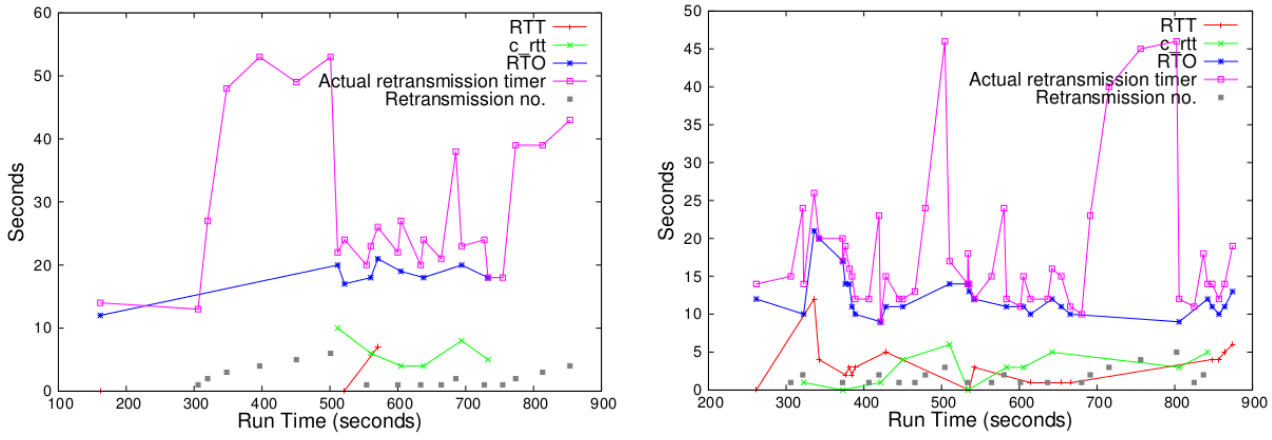

Figure 1. Cases setting the retransmission timer for an excessively long time period

3. Proposed Scheme

We attempt to improve TCP fairness among the nodes without reducing the total goodput by improving the TCP retransmission timer adjustment mechanism. We note that a major cause of an excessively long retransmission timer is the dithering mechanism of the previous work. The dithering mechanism adds a random value to the RTO. Hence the retransmission timer value is always larger than the RTO value. When the RTO

backs off several times, the RTO value becomes large due to exponential backoff. Dithering makes the retransmission timer larger than the RTO value which is already large. To fix the problem, we made a slight modification to the dithering mechanism proposed in [9]. In our proposed scheme, when the number of retransmissions exceeds 2 and the RTO is longer than 15seconds, dithering is performed by subtracting a random value from the backed-off RTO value with the probability of 0.5. Table 1 shows how we modified the previous scheme. The modified part is in the shaded area.

Table 1. Proposed Scheme

Event	Action
Initiation	MDEV = 16 seconds

An RTT sample is taken from the latest transmission	Err = RTT – SRTT
	SRTT = SRTT + 0.125 * Err
	MDEV = MDEV + 0.25 * (|Err| – MDEV)
	RTO = SRTT + 4 * MDEV
	retransmission_timer = RTO
	// dithering
	If (retransmission_timer > 4 seconds)
	  retransmission_timer += random integer in [0, retransmission_timer*0.5)
	else
	  retransmission_timer += random integer in [0, retransmission_timer)

Retransmission timer expires	// variable limits on the maximum backoffs
	If RTO ≤ 3 seconds then
	  n_backoff = max (#retransmissions, 4)
	else if RTO ≤ 6 seconds then
	  n_backoff = max (#retransmissions, 3)
	else if RTO ≤ 12 seconds then
	  n_backoff = max (#retransmissions, 2)
	else
	  n_backoff = max (#retransmissions, 1)
	retransmission_timer = max(RTO << n_backoff, 48 seconds)
	// dithering
	If (retransmission_timer > 12 seconds)
	  d_time = random integer in [0, retransmission_timer*0.25)
	else if (retransmission_timer > 4 seconds)
	  d_time = random integer in [0, retransmission_timer*0.5)
	else
	  d_time = random integer in [0, retransmission_timer]

	// modified in the proposed scheme
	If (#retransmissions > 2) and (RTO > 15) then
	  // with the probability of 0.5
	  retransmission_timer -= d_time
	else
	  retransmission_timer += d_time

4. Simulation Results

We evaluated performance of the proposed scheme using the Cooja network simulator. We simulated a 4x4 grid topology consisting of Sky motes. In the topology, the top leftmost node was designated as the LBR, which is connected to a TCP server using a slip protocol. Since the transmission range and interference range are 50m and 100m, respectively, and the nodes in a same row or column are 40m away from each other, many nodes suffer severe packet losses due to the hidden terminal problem. We set the random loss rate to 0% in order to focus on congestion, so packets are lost only due to packet collision or buffer overflow. The number of packets in the link-layer queue was set to 4. For an RDC mechanism, we used ContikiMAC. Table 2 summarizes the main simulation parameters.

Table 2. Simulation parameters

Parameters	Values
Radio model	Unit Disk Graph Medium (UDGM)
Random loss rate	0%
Transmission/Interference range	50m/100m
Radio duty cycling mechanism	ContikiMAC (RDC channel check rate: 8Hz)
RPL objective function	MRHOF
Number of packets in the link-layer queue	4
Length of TCP payload	48 bytes
Simulation duration	10 minutes

In this simulation, the major performance metrics are TCP goodput and fairness. For TCP goodput, we measured the number of received TCP segments. Fairness between TCP flows is quantified using Jain's index. Figure 2 shows the simulation results on the scatter plot in which we can compare performance metrics between the previous scheme and the scheme proposed in this work. In the scatter plot, having the dots in the upper right part means better goodput and better fairness. We can see that the proposed scheme has more dots in the upper right part than the previous scheme.

Figure 2. Simulation results

It is notable that TCP often suffers from pretty low goodput for the previous scheme and the proposed scheme both. We found that when the total TCP goodput was below 500 segments, the major cause of the low performance is usually in the RPL tree. In the other words, when an RPL tree is formed so that the root node (the LBR node) has two children nodes as shown in Figure 3(a), performance was not degraded badly. A typical RPL tree topology degrading TCP performance is illustrated in Figure 3(b). In the topology, node 2, which is located within the transmission range of the LBR, is not a child of the root node. While such an inefficient RPL topology remains unchanged, TCP usually suffers from performance degradation. We will investigate the effect of RPL on TCP performance as our future work.

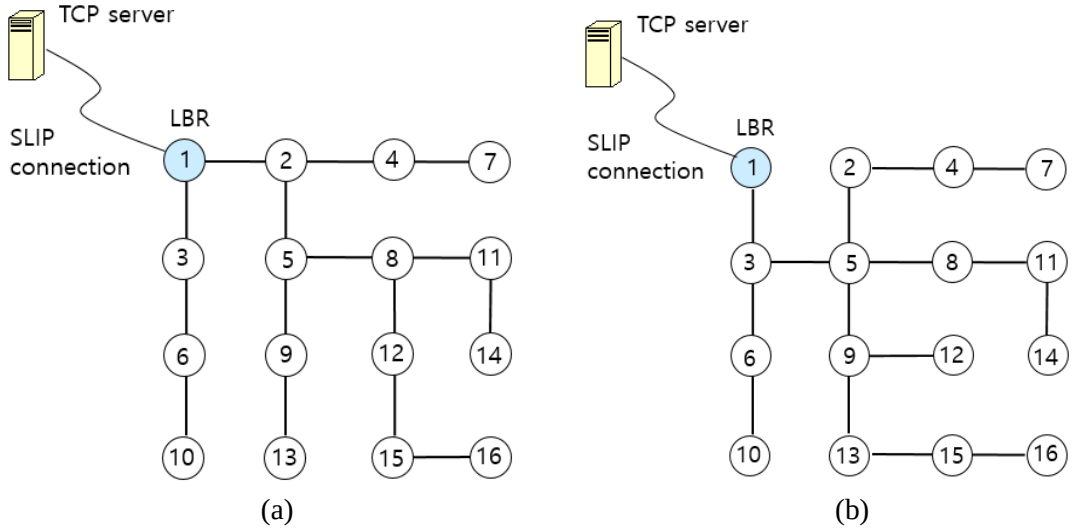

Figure 3. Illustrations of RPL tree topology affecting TCP performance. In the topology shown in (b), TCP performs worse than in the topology shown in (a).

5. Conclusion

In this work, we show that the variant of uIP TCP which adopts weak RTT estimation, exponential backoffs with variable limits, and dithering still needs enhancement in adjusting the TCP retransmission timer. Observing that the previous scheme often makes a node wait too long before timeout, we propose modifying the dithering mechanism of the previous scheme to prevent setting the retransmission timer for an excessively long time period. Unlike the previous dithering mechanism which only adds a random value to the RTO value regardless of the RTO length, the newly proposed scheme allows the dithering mechanism to subtract a random value from the RTO value with the probability of 0.5 when the number of retransmissions exceeds 2 and the RTO is longer than 15 seconds. Simulation results show that the new dithering mechanism results in fairness improvement without reducing TCP goodput. Also, we found that depending on the RPL tree topology, TCP performance significantly degrades. We will examine the effect of RPL on TCP performance as our future work.

Acknowledgement

This work was supported by 2021 Hongik University Research Fund.

References

[1] P. Hurni, U. Bürgi, M. Anwander, and T. Braun, "TCP performance optimizations for wireless sensor networks," 2012, pp. 17-32. DOI: https://doi.org/10.1007/978-3-642-28169-3_2

[2] H. Kim, H. Im, M. Lee, J. Paek, and S. Bahk, "A measurement study of TCP over RPL in low-power and lossy networks," Journal of Communications and Networks, Vol. 17, 6 2015, pp. 647-655. DOI: https://*doi*.org/*10.1109*/*JCN*.2015.000111

[3] M. Park and J. Paek, "TAiM: TCP assistant-in-the-middle for multihop low-power and lossy networks in IoT," Journal of Communications and Networks, Vol. 21, 2 2019, pp. 192-199. DOI: https://doi.org /10.1109/JCN.2019.000017

[4] C. Gomez, A. Arcia-Moret, and J. Crowcroft, "TCP in the Internet of Things: from ostracism to prominence," IEEE Internet Computing, Vol. 22, 1 2018, pp. 29-41. DOI: https://*doi*.org/*10.1109*/*MIC.2018.112102200*

[5] S. Kumar, M.P. Andersen, H. Kim, and D.E. Culler, "Performant TCP for Low-Power Wireless Networks," 2020, pp. 911-932.

[6] J. Sakamoto, D. Nobayashi, K. Tsukamoto, T. Ikenaga, G. Sato, and K. Takizawa, "Poster: Implementation and Performance Evaluation of TCP/IP Communication over Private LoRa," 2022, pp. 1-2. DOI: https://doi.org/10.1109/ICNP55882.2022.9940334

[7] C. Lim, "Improving Congestion Control of TCP for Constrained IoT Networks," Sensors, Vol. 20, 17 2020, pp. 4774. DOI: https://doi.org/10.3390/s20174774

[8] A. Dunkels, "Full TCP/IP for 8-bit architectures," 2003, pp. 85-98. DOI: https://doi.org/10.1145/1066116.1066118

[9] A. Dunkels, "uIP-A free small TCP/IP stack," The uIP, Vol. 1 2002.

[10] A. Dunkels, "The contikimac radio duty cycling protocol," Swedish Institute of Computer Science, 2011.

[11] R.C. Carrano, D. Passos, L.C. Magalhaes, and C.V. Albuquerque, "A comprehensive analysis on the use of schedule-based asynchronous duty cycling in wireless sensor networks," Ad Hoc Networks, Vol. 16 2014, pp. 142-164. DOI: https://doi.org/10.1016/j.adhoc.2013.12.009

[12] W. Ye, J. Heidemann, D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in: Proc. of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), vol. 3, pp. 1567–1576, 2002. DOI: https://doi.org/ 10.1109/INFCOM.2002.1019408

[13] G. Lu, B. Krishnamachari, C. Raghavendra, "An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks," in: Proc. of the 18th International Parallel and Distributed Processing Symposium, p. 224, 2004. https://doi.org/10.1109/IPDPS.2004.1303264

[14] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with cooja," 2006, pp. 641-648. DOI: https://doi.org/10.1109/LCN.2006.322172

[15] K. Roussel and O. Zendra, "Using Cooja for WSN simulations: Some new uses and limits," 2016, pp. 319–324.

[16] C. Lim, "A Simulation study on TCP performance for constrained IoT networks," International Journal of Internet, Broadcasting and Communication Vol.15 No.1 1-7 (2023) DOI: http://dx.doi.org/10.7236/IJIBC.2023.15.1.1