TAiM: TCP Assistant-in-the-Middle for Multihop Low-power and Lossy Networks in IoT

Mingyu Park and Jeongyeup Paek

Abstract: **We investigate the performance of TCP over multihop many-to-one wireless low-power and lossy networks (LLNs), and present TCP assistant-in-the-middle (`TAiM`) to improve TCP fairness among LLN endpoints without sacrificing aggregate throughput nor end-to-end backward compatibility. `TAiM` exploits the fact that RTT over LLN is significantly higher, and more variable, than wired or WiFi networks due to unique characteristics of LLN. `TAiM` intervenes in the middle of TCP communication only at the LLN border router (LBR), and delicately manipulates the RTT of the passing flows to achieve its goal, but does not infringe anything of packet nor operation of existing protocols. `TAiM` is adaptive and flexible to network status, and does not require any modifications to the TCP/IPv6 stack at the endpoints. We implement `TAiM` solely in a Linux-based LBR, and perform experiments on two 30-nodes TelosB testbeds running BLIP/TinyRPL stack in TinyOS. Our experiment results show that `TAiM` helps TCP to operate fairly and efficiently while maintaining total throughput and end-to-end compatibility.**

Index Terms: **Fairness, flow control, low-power and lossy wireless networks (LLNs), multihop, RPL, TCP.**

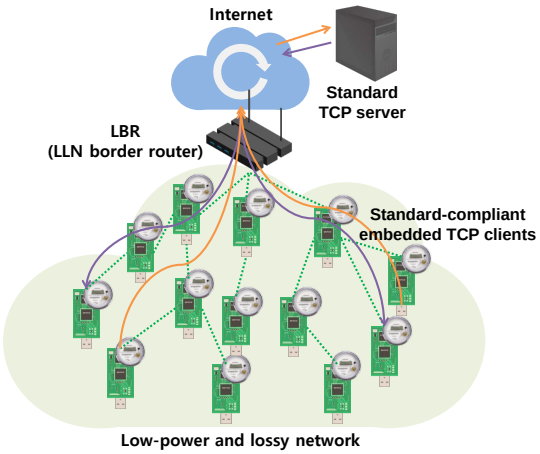


Fig. 1. Illustration of the target IoT LLN scenario.

I. INTRODUCTION

LOW-POWER and lossy wireless networks (LLNs) have been used widely in many areas including smart grid AMIs [1], [2], smart city management [3], health care [4], building and industrial automation [5], [6], and environmental monitoring. Armed with Internet standard IP/IPv6-based protocols such as RPL [7], [8] and 6LoWPAN [9], LLNs have started to integrate as part of the Internet architecture that aims to support IP connectivity on millions of devices over wireless mesh networks. This approach enables LLN to be more inter-operable, scalable, flexible and versatile, leading to the realization of Internet of things (IoT).

Taking smart grid as an example, its network requirements include unprecedented scale and multivendor interoperability over wireless network, which is well-suited for IP-based LLN [2], [10]. Cisco's CG-Mesh system and field area network is a good example of an IP-based commercial solution for smart grid developed over LLN by the industry [11]. It is based on IEEE 802.15.4g/e at the PHY and MAC layer to form an LLN, and allows up to 5000 endpoints to connect to a single LLN border router (LBR). Above the link layer, it uses 6LoWPAN, RPL, and IPv6 to provide inter-networking support for smart metering endpoints. This provides an evidence that the use of IPv6 over RPL/6LoWPAN in an IEEE 802.15.4 network is feasible in large scale LLNs. It is also part of the growing industry effort that invests in LLN solutions to facilitate IoT. We believe that this trend is promising and will continue for the foreseeable future. That is, we believe that industrial LLN solutions will adopt well-tested standard Internet protocols for scalability, compatibility, security, and development cost reasons [10], [12].

Over the past few years, there have been a number of performance evaluation studies of LLN combining IP based protocols (Section V). However in most of those studies, UDP was used as the transport protocol. This is because, considering the fact that LLNs typically have low throughput, high packet loss, and frequent topology changes among other characteristics that pose integration challenges, it was believed that TCP is inappropriate for resource-constrained embedded devices in LLN. This is the main reason why TCP has not been tightly associated with LLN (for example, 6LoWPAN defines header compression for UDP, but not for TCP) even though it is the dominant transport protocol in today's Internet.

However, effective support of TCP is essential for achieving interoperability of the IP stack in LLNs. Since many legacy and already-installed devices currently use TCP, LLN is also required to support TCP maybe not for performance improvement, but for compatibility and cost reduction. For example, smart grid applications support encapsulation of their protocols in TCP/IP (but not UDP), and some protocols even define a dedicated mode of operation over the TCP/IP (e.g., IEC 60870-5-104 variation of the IEC 60870-5-101 SCADA protocol) [13]–[15].

Manuscript received November 18, 2018. This paper is specially handled by EICs with the help of three anonymous reviewers in a fast manner.

This research was supported by the Chung-Ang University Graduate Research Scholarship in 2017, and also by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2017R1D1A1B03031348).

M. Park and J. Paek are with the Chung-Ang University, School of Computer Science and Engineering, Seoul, Republic of Korea, email: {hello0922, jpaek}@cau.ac.kr.

J. Paek is the corresponding author.

Digital Object Identifier: 10.1109/JCN.2019.000017

Thus, TCP is likely to continue to play a significant role in LLNs as a part of IoT.

Unfortunately, it is true and well-known that TCP's congestion control mechanism performs poorly in lossy wireless environments and generates high overhead, especially over wireless multihop tree topology. In particular, consider the target IoT LLN scenario illustrated in Fig. 1. There are 'N' LLN endpoints (e.g., smart electrical meters in AMI system) that form a mesh network rooted at an LBR, and the LBR connects the LLN to the Internet (WAN) at which a server resides. LLN endpoints communicate with each other through IEEE 802.15.4 links, and use RPL to construct IPv6 routes towards the LBR. On top of RPL/IPv6, each node uses TCP connection to the server for data collection. In this scenario, every TCP flow must pass through an LBR which becomes the bottleneck of communication. During this process, packet loss due to collisions, RTT variance due to multihop topology (e.g., different network depth, varying buffering delays for forwarding packets from varying subtree sizes) and link-layer retransmissions, in addition to how TCP's congestion control mechanism reacts to those are the reasons why TCP suffers unfairness problem and operates inefficiently in multihop LLN (as we will show in Section II).

In this paper, we propose 'TCP assistant-in-the-middle' (TAiM) for multihop LLNs to improve TCP fairness among LLN endpoints without sacrificing total aggregate throughput nor end-to-end backward compatibility. TAiM exploits the fact that RTT over multihop LLN is significantly higher and more variable than wired or WiFi networks, especially due to lower bandwidth, higher loss rate, and multihop topology. Based on this intuition, TAiM delicately manipulates the RTT of the passing flows to achieve its goal. TAiM intervenes in the middle of TCP communication only at the LBR but does not infringe anything of packet nor operation of existing protocols. TAiM is adaptive and flexible to network status, and does not require any modifications to (in fact, oblivious to) the TCP/IP stack at the endpoints.

The contributions of this paper are as follows;

- We propose TAiM, a TCP assistant scheme that can address the TCP unfairness problem among LLN endpoints when communicating with an endpoint outside the LLN (e.g., Internet) without modifying anything at the endpoints nor violating the end-to-end backward compatibility.
- We implement TAiM in, and only in, a Linux-based LBR, and evaluate it on two 30-node indoor LLN testbeds. Each node is a TelosB [16] mote, and use BLIP and TinyRPL in TinyOS [17] as the TCP/IP and RPL implementation.
- Experiment results show that TAiM reduces the throughput difference between the fastest and the slowest node by up to 50% while maintaining the network-wide total aggregate throughput.

The remainder of this paper is organized as follows: In Section II, we define the problem and motivate our work. Section III presents the design of TAiM, and elaborate on its main functional blocks. We then evaluate TAiM in Section IV, and discuss the related work in Section V. Finally, we will summarize our work in Section VI.

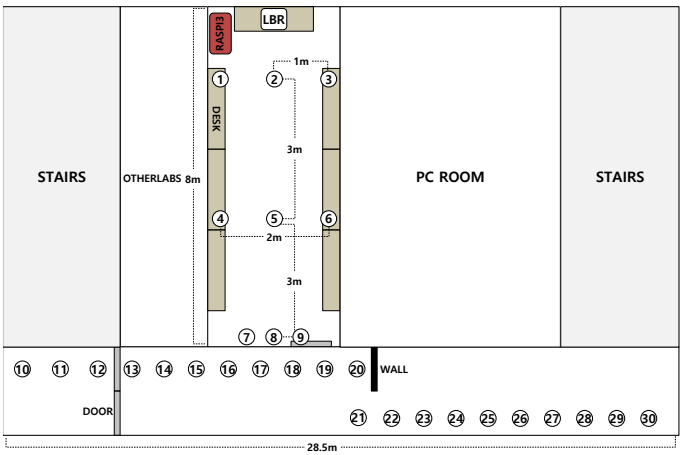


Fig. 2. 'Testbed-1': 30-node LLN testbed that expands from an office room to the corridor. Transmit power set to -12 dBm.

28 bytes	40 bytes	20 bytes	38 bytes
TinyOS message header	IPv6 header	TCP header	Payload

Fig. 3. Structure of a TCP/IPv6 packet in TinyOS/BLIP.

II. PROBLEM – PRELIMINARY STUDY

In this section, we conduct a preliminary testbed experiments to confirm the unfairness problem of TCP in multi-hop LLN, and discuss the main causes of the problem to motivate our work.

A. Scenario and Experiment Setup

We have designed our testbeds and experiments to mimic the target IoT LLN scenario illustrated in Fig. 1. At a high-level, LLN endpoints communicate with each other through IEEE 802.15.4 links, and use RPL to construct IPv6 routes towards the LBR. Once a node founds a path to the LBR through RPL, it opens a TCP connection to the server in WAN. For our experiment purposes, the server will issue a 'start data collection' command to all nodes when all N nodes are connected to the server. Then the clients will transmit TCP data segments to the server as fast as possible. A client disconnects the communication once it finishes transferring 500 application-layer data segments[1], which corresponds to 19000 application-layer data bytes. In this scenario, LBR simply routes and forwards received packets between LLN and WAN without any modification to the packets.

Experiment Setup. We configured two LLN testbeds, each with 30 LLN endpoints and one LBR. The first testbed (named 'testbed-1' hereafter) expands from an office room to the corridor as depicted in Fig. 2, and the second testbed will be shown in Section IV. Each circled number represent an LLN endpoint, and radio transmit power is set to -15 dBm. Each LLN node is a TelosB [16] clone device with an msP430 microcontroller and a CC2420 radio chip, and communicates with each other over IEEE802.15.4 links. We use BLIP and TinyRPL in TinyOS 2.1.2 as the embedded TCP/IP stack and RPL routing protocol

[1]Note that the actual number of TCP packets may differ due to msS, sliding window, congestion window, etc.

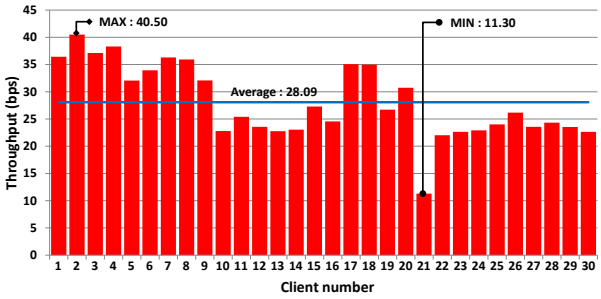


Fig. 4. TCP throughput result from a preliminary experiment on 'testbed-1', a 30 nodes multihop LLN deployed in the corridor. Unfairness can be seen.

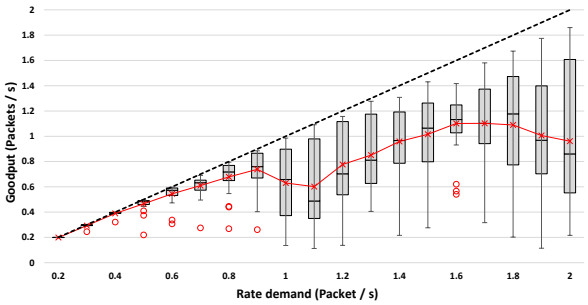


Fig. 6. Average goodput results achieved by UDP on the 'testbed-1'.

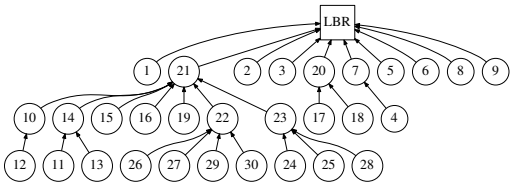


Fig. 5. A snapshot of the routing tree topology constructed by RPL during the preliminary experiment on 'testbed-1'.

implementation, respectively, mostly with their default configurations after correcting a few software bugs in their code with reference to the previous work by Kim *et al.* [18]. On top of that, we installed TCP client application in TinyOS to all LLN endpoints for the experiments. We use Linux Ubuntu 16.04.3 for the LBR, and Raspberry-Pi3 device for the TCP server which is small single-board computers with Linux based Raspbian OS.

One parameter of the experiment is worth noting. The maximum packet size a TelosB (or many IEEE 802.15.4-based devices) can transmit is 127 bytes, and the total header size including TCP/IP adds up to 88 bytes as shown in Fig. 3, leaving only 39 bytes for TCP payload. To minimize the already-significant header overhead, it is better to set the MSS to the maximum possible size. For this reason, embedded TCP implementations (e.g., uIP [19] in ContikiOS or BLIP in TinyOS) often ends up having single outstanding packet without having sliding window behavior.

B. Preliminary Experiment Results

Fig. 4 plots the per-node average TCP throughput on 'testbed-1' where the solid horizontal line represent the average of all nodes. Note that we have not introduced `TAiM` to this experiment yet. As it can be seen, the achieved data throughput of each node is very different even though they are intended to be identical devices within an LLN from the applications perspective. The throughput of the fastest node is about 3.6 times the slowest one. This result exemplifies that there is an unfairness problem in multi-hop LLN.

One interesting observation is that the throughput of several children nodes are significantly higher than their parent nodes. To understand this, consider a parent node which has several children; e.g., node 21 in Fig. 5 which depicts a snapshot of the routing tree topology during this experiment. Note that all TCP clients are attempting to transmit as fast as possible. In this case, there are a lot of packets in the air around the parent node since the parent node needs to receive packet from its children while forwarding and transmitting all of those packets in addition to its own packets to its parent (i.e., parent's parent). Thus, contention will be intensive around a node with large number of children, relatively more significant than that of the children, and accordingly it is more exposed to link losses compared to its children.

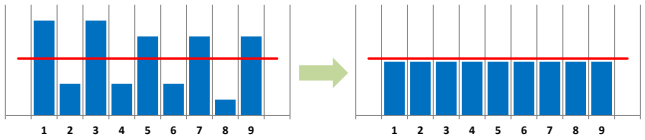
Moreover, a node which loses a packet will notice the loss after TCP retransmission timeout, and thus will not send anything until retransmission timer is fired. In this case, the number of packets in the air will decrease sharply, and thus other nodes can communicate with the server better than before. As a result, the gap between a node exposed to link losses and a node which does not becomes significant. Because the parents are more exposed to link losses than their children in a many-to-one topology, this phenomenon may happen more frequently on the parent nodes. Similarly, throughputs of the clients in the room are much better than those in the corridor because the number of clients in the room is less than the corridor, thus the contention in the room is also less than the corridor, resulting in better use of the medium.

To show how congestion affect link losses, we conducted additional experiment using UDP instead of TCP with various offered traffic load. Fig. 6 shows the result from the UDP experiment. X-axis represents the number of packet sent every second, and y-axis represents the number of packets successfully received at the LBR. The dash x = y line is the reference line to compare the delivery ratio. Each boxplot shows the distribution of each node's goodput. As the figure shows, all nodes can communicate well almost without losses when endpoints transmit packet slowly until 0.4 packets / s. However starting from 0.5 packets / s and up, some nodes suffer losses and the loss rates increase dramatically as the offered load increases. In particular, we can observe a node with significantly low goodput at offered load of 0.5 packets / s and above. The worst node is node 21 which is also the slowest node in the preliminary experiments in Fig. 4. This result explains that a node which has a lot of children is also exposed and sensitive to congestion losses.

Bottom line is that TCP endpoints within multi-hop LLN experiences severe throughput unfairness problem due to link losses from bursty congestion, and this motivates us to design our proposed new scheme, `TAiM`.

III. `TAiM` DESIGN

In this section, we propose `TAiM` to address the unfairness problem in multi-hop LLNs.



(a) (b)

Fig. 7. An example scenario where the number of packets sent by an LLN endpoint within each (small) time window vary significantly. x-axis is conceptual time windows, and y-axis is the number of packets: (a) Burst scenario and (b) distributed scenario.

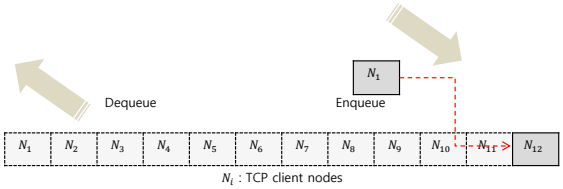


Fig. 8. An example scenario of priority queuing where a second packet from N_1 is forwarded before the first packet from N_{12}.

TAiM resides and operates only at the LBR, and uses RTT adjustment, burst distribution, and packet re-ordering techniques to enhance throughput fairness between each LLN endpoint (TCP client) without total aggregate throughput loss. While doing this, TAiM intervenes in the middle of TCP communication, but does not violate or disrupt end-to-end compatibility because TAiM does not infringe on any packet or existing operation of TCP/IP protocols. TAiM simply measures throughput of each client via packet sniffing, and controls traffic flow by adjusting the timing of forwarding.

TAiM is based on the intuition that RTT over multihop LLN is significantly higher and more variable than wired or WiFi networks, especially due to lower bandwidth, higher loss rate, and multihop topology. Key idea is to buffer packets to control RTT of each node, and allow all endpoints within the LLN to experience similar RTT at the TCP layer, both among nodes and across traffic load variations. This can mitigate the effect of highly variable RTT, and also disperse traffic bursts to achieve stable and continuous traffic flows. While doing this, TAiM does not drop any packet, unlike the well-known random early detectino (RED) technique. TAiM is adaptive and flexible to network status, and does not require any modifications to (in fact, oblivious to) the TCP/IP stack at the endpoints.

A. Burst Distribution

The first technique in TAiM is burst distribution. Even if all clients transmit data packets periodically at a regular rate, the actual number of packets in transit in the LLN within a small time window can be irregular. For example, consider Fig. 7(a). Imagine that the solid horizontal line represent the throughput threshold which a link can endure. The x-axis is conceptual time windows, and y-axis is the number of packets per each time window. In this case, collision probability will increase dramatically when the number of packets is larger than threshold. To avoid this situation, we try to distribute these packets evenly along the time domain. With this motif, we let TAiM, which resides and operates at the LBR, buffer packets and intentionally add time delay between the forwarding of consecutive packets when it receives packets from either side of the TCP connection.

Algorithm 1 Delay time decision algorithm.

1: **while** until the end of experiment, at 1 minute interval **do**
2: **if** $throughput_{\max} > 2 \times throughput_{\min}$ **then**
3: $delaytime \leftarrow delaytime \times (1 + \alpha)$
4: $\overline{ERTO} \leftarrow \text{MEAN}(ERTO)$
5: **if** $delaytime > \overline{ERTO}/num\_clients\_in\_q$ **then**
6: $delaytime \leftarrow \overline{ERTO}/num\_clients\_in\_q$
7: **end if**
8: **else if** $throughput_{\max} < 2 \times throughput_{\min}$ **then**
9: $delaytime \leftarrow delaytime \times (1 - \beta)$
10: **end if**
11: **for all** clients **do**
12: $throughput_i \leftarrow \text{EWMA}(throughput_i)$
13: **end for**
14: $num\_clients\_in\_q \leftarrow \text{EWMA}(num\_clients\_in\_q)$
15: **end while**

Said differently, TAiM puts received packets into its priority queue and forwards the packets to the server (or clients) after a little delay between packets to reduce bursts of packets in the air. Note that buffering is possible at the LBR since, unlike resource-constrained LLN endpoints, LBR is a more resource-rich Linux device. One may think that adding a little delay to packets may slow-down the whole system. It does not. In fact, by doing this, the number of packets in the air will be distributed evenly as shown in Fig. 7(b). As a result, the collision probability will decrease, thus reducing the average RTT and improving overall throughput of those nodes that used to experience packet collisions.

B. Packet Re-ordering

The second technique in TAiM is packet re-ordering. Since TAiM can buffer received TCP packets, it can also decide the priority order in which to forward those buffered packets. To this end, TAiM estimates each node's accumulative throughput per time window, and uses that for prioritizing each node. When TAiM intentionally delays packets before every forwarding, a node which has the lower throughput can have shorter delay, and the nodes with higher throughput can have longer delay.

As a simple example, suppose that there are 30 nodes, the round trip time (RTT) of every node is 500 ms, and the inter-packet interval is always 50 ms for simplicity of illustration. Also assume that the estimated throughput of every node is distinct. If the LBR receives packets from at least 12 nodes almost simultaneously, the packet from the lowest throughput node will be sent first, and the packet having the 11th priority will be sent after 500 ms. Because the RTT is 500 ms, at the same time that the 11th packet is transmitted, the next packet of the lowest throughput node will arrive and the packet will be enqueued. At this time, 12th packet is not transmitted yet, and the lowest throughput node's packet has a higher priority than 12th packet, thus the lowest node can send a packet one more time compared to 12th nodes. Fig. 8 illustrates this operation concisely. This packet re-ordering is implemented using per-flow queues at the LBR, and is a key component of TAiM that achieves flow control for throughput fairness without dropping (e.g., RED) or tagging (e.g., ECN) packets.

C. Delay Time Decision for RTT Adjustment

The remaining key challenge is how TAiM can decide the ap-

propriate delay time. In order to give sufficient time for dispersing packet bursts with a goal of collision-free packet transmission, TAiM selects a default delay time of D_{init} at start when there are no TCP flows passing through the LBR. The purpose of D_{init} is to avoid burst transmissions by providing minimum spacing between packets, and it is selected by observing bursty behavior at the USB/Serial link of LBR interface[2]. Then, once TCP flows start to pass through the LBR, TAiM will recalculate every minute to find an appropriate delay time for each flow and adapt to the network status.

For this purpose, TAiM estimates the throughput of each node every minute by peeking the TCP sequence number in the received packets, smoothes the estimation using exponentially weighted moving average (EWMA) filter, and calculates the maximum and minimum of throughputs for all TCP flows. TAiM also estimates the number of concurrent TCP flows passing through the LBR at any given time using EWMA as well. Using these values, TAiM determines the delay time according to the following 'Delay Time Decision Algorithm' shown in Algorithm 1. The basic idea of the algorithm is that, if sufficient fairness is achieved in the current time window, TAiM tries to forward packets faster. In the opposite case, if unfairness is detected, TAiM forwards slower. It should be noted that the decrement (β, 5% in our implementation) of delay time is bigger than the increments (α, 2% in our implementation). This is to aggressively seek for better throughput when fairness is achieved because, even though fairness is one of our goal, we do not want to sacrifice throughput while achieving fairness.

In addition, TAiM checks whether the delay time is larger than average expected RTO (ERTO). This is because, if the delay time is too long, a TCP client that has timed out by RTO will retransmit a duplicate packet although the packet has already arrived at the LBR successfully. These duplicate retransmission packets in the path of communication will cause unnecessary energy consumption and increase congestion. Since TAiM can sniff all packets, it can estimate the RTT for each flow. That means TAiM can also calculate ERTO of each node. To this end, we can use this value as an upper limit of the delay time. Specifically, TAiM never sets the delay time to be bigger than a value which is the average ERTO divided by the number of clients. Through this decision process, TAiM is adaptive to network condition.

D. TAiM Process All Together

To put everything together, Fig. 9 shows the overall functional diagram and components of TAiM. When TAiM within an LBR receives a packet, it checks whether it is a TCP segment passing between LLN and WAN through the 'packet checker'. If not, TAiM will ignore and forward the received packet directly since our scheme aims at assisting TCP flows in multihop LLN. If the packet is a TCP segment flowing between LLN and WAN, it is passed to the 'throughput/RTT estimator' component which estimates throughput, RTT, and ERTO for each TCP flow. Then, the estimated throughput is used for assigning priority to the received packets, and the RTT is used for setting an upper limit of delay time. After the estimation and configuration of the delay time, the estimator inserts the packet into per-flow priority queue. Additionally, a scheduler takes one packet per each delay time and forwards the packet to its destination through an appropriate interface. Finally, one limitation of TAiM is that all of above estimations assume a relatively static network such as smart grid AMI network, and the estimations may be inaccurate for mobile devices that move between different LLNs/LBRs [20]. We hope to address the mobile devices in our future work.

[2]This D_{init} may sound insignificant or unnecessary theoretically, but is critical in real implementations due to bursty behaviour of USB/serial links.

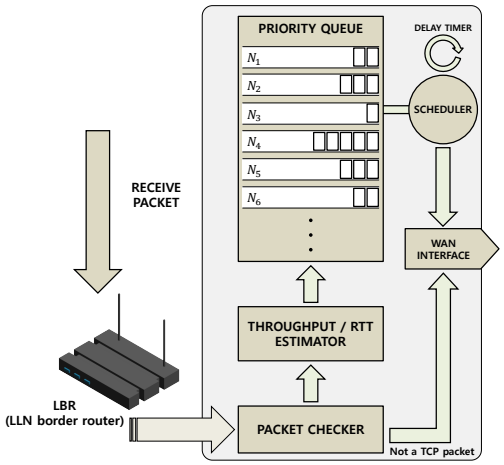


Fig. 9. TAiM's main functional components.

IV. EVALUATION

To evaluate the effectiveness of TAiM and show how TCP performance is improved compared to the result in Fig. 4, we conduct TCP experiments with TAiM using the scenario and 'testbed-1' setup (Fig. 2) described in Section II.A. All node/routing configurations and parameters are identical to the preliminary experiment since TAiM operates at and only at the LBR without any modification to the endpoints.

Fig. 10 plots the per-node throughput from the experiment when we apply TAiM. We can visually observe at a glance that the overall throughput fairness is much better than the case without TAiM in the Fig. 4. Specifically, the nodes that used to suffer high contention and exhibited relatively lower throughput have improved their throughput significantly compared to the experiment without TAiM. The main reason for this improvement is because TAiM reduces congestion by distributing packet bursts over time. Moreover, the total aggregate throughput is not reduced compared to the case without TAiM although TAiM intentionally delays most packet transmissions. This means that the added delay offset did not negatively impact the throughput nor the average RTT of the TCP clients, which in turn implies that the throughput achieved by TCP in LLN without TAiM is far less than what can be inferred from the RTT due to the lossy and multihop characteristics of wireless LLN.

To compare the fairness of both results quantitatively, we use Jain's fairness index [21] as well as the difference between the max and min throughput nodes. First of all, Jain's fairness index has improved from 0.947 without TAiM to 0.994 with TAiM. The increment may look small numerically, but the improvement is clear and consistent over multiple experiments.

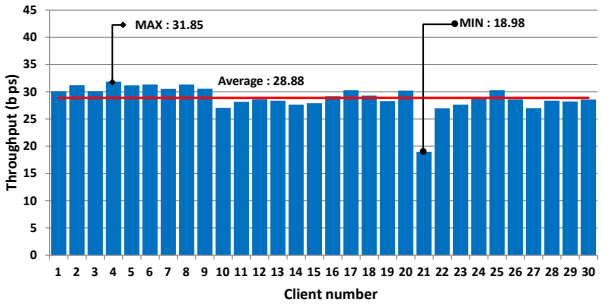


Fig. 10. TCP data throughput for each node with TAiM on the 'testbed-1'. Fairness has improved compared to Fig. 4.

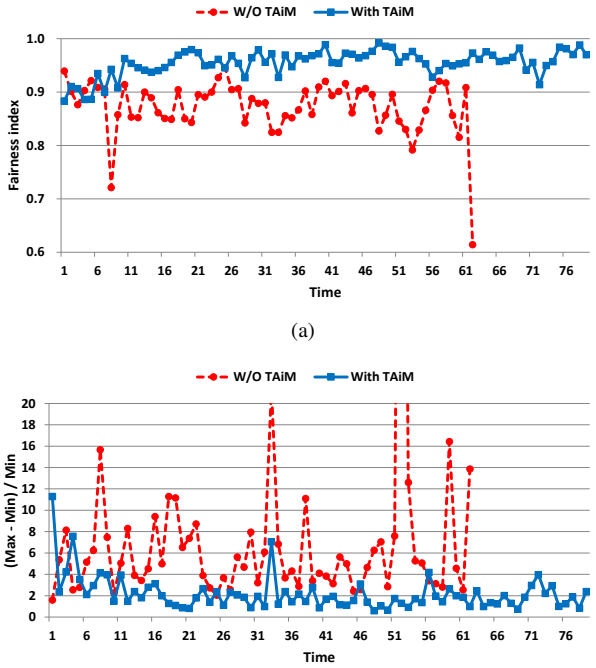


Fig. 11. Fairness of per-node throughput during the progress of the experiment per-minute, with and without TAiM on the 'testbed-1': (a) Jain's fairness index and (b) max-min difference ratio.

However, since Jain's fairness index indicates fairness between all clients, it may not well represent unfairness between significantly faster and slower nodes when there are also many nodes closer to the average. So we use a second indicator, *'max-min ratio'*, which is defined as the throughput difference between the nodes with maximum throughput and minimum throughput, divided by the minimum throughput. Using this metric, basic TCP's max-min ratio is 3.6 which means that the fastest node's throughput is about 3.6 times the slowest node's throughput. On the other hand, when we use TAiM, max-min ratio is only 1.7. In addition, the differences between throughputs in the room and those in the corridor are approximately twice without TAiM, except for one slowest node which has large number of children. However when we use TAiM, the differences becomes 1.1x only. Thus the fairness has improved by over 50% when we use TAiM.

Above results represent only the final value averaged over the whole duration of the experiment (which was roughly 1.5∼2 hours). To take a deeper look into the details, we analyze the per-node throughput over time at 1 minute intervals based on the time-scale that TAiM recalculates the delay time. Fig. 11(a) and Fig. 11(b) present the per-minute Jain's fairness index and max-min throughput ratio during the experiment, with and without TAiM, over time. Dotted line with circles present the case without TAiM, and the solid line with squares is for the case with TAiM. At the beginning of the experiment, fairness may be similar for both cases with and without TAiM. However, as time progresses, TAiM clearly achieves better throughput fairness (higher fairness index, lower max-min ratio) throughout the entire experiment after around 10 minutes from the beginning of the experiment. Furthermore, note that TAiM finishes later than without TAiM. This is because we plot the fairness index only when all 30 connections are active, and TAiM is fairer than the case without TAiM. That is, the fastest node finishes earlier without TAiM since its throughput (the fastest among all nodes) is higher than the case with TAiM. Said differently, TAiM makes the fastest node slower to let the slowest node become faster.

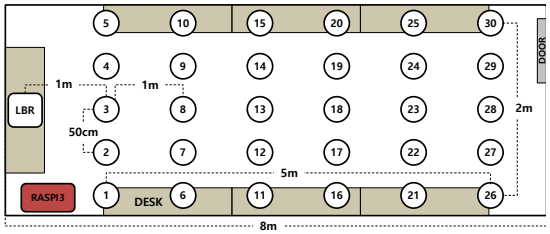


Fig. 12. 'Testbed-2': 30-node LLN testbed in an office room. Transmit power is set to -25 dBm.

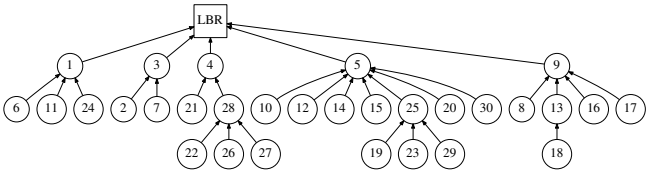
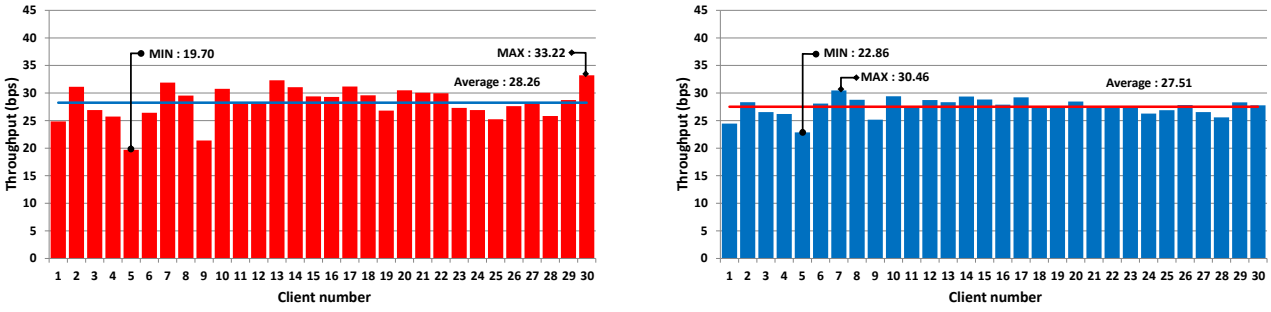


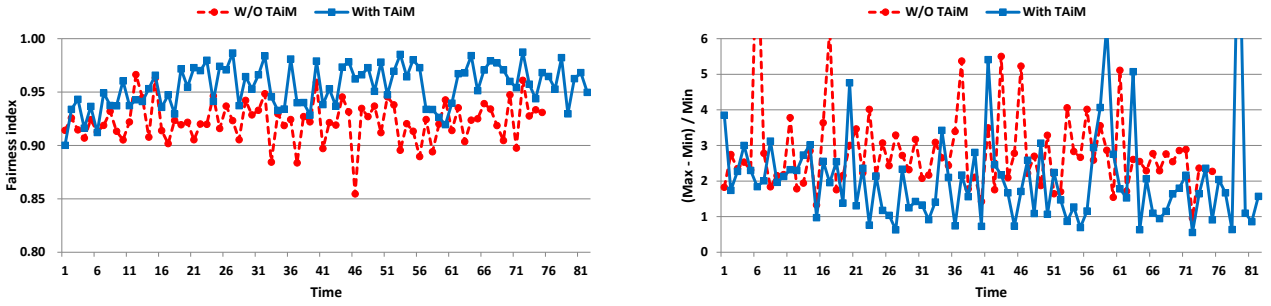
Fig. 13. A snapshot of the routing tree topology constructed by RPL during TCP experiment on the 'testbed-2'.

'Testbed-2' Experiment. To ascertain that TAiM is adaptive to other topologies as well, we conduct another experiment in an office room where the nodes are deployed in a regular lattice form as shown in Fig. 12 (named 'testbed-2' hereafter). Since the physical size of the experimental environment is smaller, we decrease transmit power to -25 dBm for creating a multihop topology within the room. Other parameters are exactly equal to the parameters of the experiment conducted on testbed-1. The results of the experiments with and without TAiM are plotted in Figs. 14(b) and 14(a), respectively. Fig. 13 depicts a snapshot of the routing tree topology. As expected, node 5 and 9 are the slowest nodes due to having a large number of children nodes. In the results, the aggregate throughput decreases slightly by about 2.65% compared to the experiment without TAiM. However Jain's fairness index has improved from 0.989 to 0.997. In addition, the difference between maximum throughput and minimum decreases from 1.7 times to 1.3 times by 20.98%. Figs. 15(a) and 15(b) plots the changes of fairness index and



(a) (b)

Fig. 14. TCP data throughput for each node on 'testbed-2' with and without TAiM: (a) Without TAiM and (b) with TAiM.



(a) (b)

Fig. 15. Fairness of per-node throughput during the progress of the experiment per-minute, with and without TAiM on the 'testbed-2': (a) Jain's fairness index and (b) max-min difference ratio.

max-min ratio of per-node throughput over time at 1 minute intervals. Overall, both the fairness index and max-min fairness improved. This improvement may look small, but this is because the overall fairness was pretty good even without TAiM due to the regular lattice pattern deployment of the nodes in testbed-2. In other words, this is the worst case for TAiM where there was no room for improvement to begin with. These results show that TAiM is adaptive to other topologies as well.

To summarize, the results all together show that TAiM improves throughput fairness by reducing the throughput difference between the fastest node and the slowest node without sacrificing total aggregate throughput. All we manipulate is the timing of forwarding (and possibly RTT) to mitigate the collision and contention due to bursty transmissions, and we do it minimally and delicately to maintain total aggregate throughput and avoid firing of RTO. Added delay offset does not increase the average RTT of nodes since added delay to the fastest node provides opportunities for other slower nodes. TAiM is based on the intuition that RTT over multihop LLN is significantly higher and more variable than wired or WiFi networks, especially due to lower bandwidth, higher loss rate, and multihop topology. End-to-end backward compatibility comes for free since we do not modify anything in the packets nor protocol behavior.

V. RELATED WORK - TCP IN MULTIHOP LLN

There are vast amount of work on TCP fairness over the past 20∼30 years in the Internet or WiFi context (e.g., RED [22]), which is too many to list here. However, there are not many in the LLN context. In this section, we limit our scope to related work on TCP/IPv6 over multihop LLN.

Several prior work have investigated the performance of IPv6/RPL on IEEE 802.15.4 based LLN. Ko *et al.* evaluated the performance of RPL and 6LoWPAN using TinyRPL and BLIP implementations in TinyOS [23], and shown that performance is similar to the de facto data collection protocol, CTP in TinyOS, while having the benefits of an IPv6-based architecture. They also compared it against the performance of ContikiRPL over uIPv6 in ContikiOS [24], and showed that the two embedded IP stack implementations are interoperable but parameter selection and implementation details have significant effect on the performance. However, both of these work neglected TCP in their evaluations.

Duquennoy *et al.* presented TCP experiment results on LLN using the burst forwarding scheme that they proposed to enhance data throughput [25]. However, their LLN setup was a single PC-LLN-PC line topology testing a single stream of data. In contrast, our work uses embedded TCP on one side of the connection, uses multihop tree topology with multiple children nodes per parent, and tests 30 streams of data simultaneously in a many-to-one LLN scenario.

Kim *et al.* [18] presented a measurement study of TCP over RPL in an IPv6 and IEEE 802.15.4-based LLN. Similar to our preliminary findings, they have also identified that TCP has unfairness problem in multihop LLN. Zheng *et al.* [26] have also shown that TCP operates poorly in LLN. By measuring retransmission times, transfer duration and energy consumption, they show that TCP is very sensitive to congestion in LLN. However, unlike our work, neither of these work have proposed any mechanism to resolve those problems. In contrast, we propose a mechanism to address the unfairness problem of TCP over mul-

tihop wireless LLN without requiring modifications to (in fact, oblivious to) the TCP/IP stack at the endpoints nor sacrificing end-to-end backward compatibility.

One of the most notable work in the WiFi context is *Snoop* by Balakrishnan *et al.* [27] which proposed a simple and effective mechanism to improve TCP throughput over wireless links. Similar to TAiM, *Snoop* operates on the basestation without any end-to-end modifications. However, *Snoop* is experimented with AT&T Wavelan which has much higher bandwidth and operations in a 1-hop star topology. In addition, their goal was to improve throughput and did not mention fairness. In contrast, TAiM focuses on low-bandwidth high-RTT multihop LLN with a goal of achieving fairness among several endpoints despite variable RTT and congestion levels. Finally, compared to the well-known RED [22] idea, our work differs in that we do not drop packets within the network.

VI. CONCLUSION

In this paper, we proposed TAiM, *'TCP Assistant-in-the-Middle'*. It uses 'burst distribution' and 'packet re-ordering' techniques to address the unfairness problem of TCP over multihop wireless LLNs. Key idea is to balance the RTT that each LLN endpoint experiences by adjusting the timing of forwarding at the LBR based on the achieved throughput. TAiM is adaptive and flexible to network status, and does not require any modifications to (in fact, oblivious to) the TCP/IP stack at the endpoints. We implemented TAiM solely in a Linux-based LBR, and performed real experiments on two 30-nodes TelosB testbeds running BLIP/TinyRPL stack in TinyOS. Our results show that TAiM reduces the throughput difference between the fastest node and the lowest node by up to 50% without sacrificing total throughput nor end-to-end backward compatibility.

REFERENCES

[1] E. Ancillotti, R. Bruno, and M. Conti, "The role of the rpl routing protocol for smart grid communications," *IEEE. Commun. Mag.*, vol. 51, pp. 75–83, Jan. 2013.

[2] D. Popa, M. Gillmore, L. Toutain, J. Hui, R. Ruben, and K. Monden, "Applicability statement for the routing protocol for low power and lossy networks (RPL) in AMI networks," *draft-ietf-roll-applicability-ami-10*, Jan. 2015.

[3] T. Heo *et al.*, "Escaping from ancient rome! Applications and challenges for designing smart cities," *Trans. Emerging Telecommun. Technol.*, vol. 25, no. 1, pp. 109–119, 2014.

[4] J. Park *et al.*, "Glasses for the third eye: Improving clinical data analysis with motion sensor-based filtering," in *Proc. ACM SenSys*, Nov. 2017.

[5] A. Brandt, J. Buron, and G. Porcu, "Home automation routing requirements in low-power and lossy networks," *RFC 5826*, Apr. 2010.

[6] E. J. Martocci, P. D. Mil, N. Riou, and W. Vermeylen, "Building automation routing requirements in low-power and lossy networks," *RFC 5867*, June 2010.

[7] T. Winter *et al.*, "RPL: IPv6 routing protocol for low-power and lossy networks," *RFC 6550*, Mar. 2012.

[8] H.-S. Kim, J. Ko, D. E. Culler, and J. Paek, "Challenging the IPv6 routing protocol for low-power and lossy networks (RPL): A Survey," *IEEE Commun. Surveys Tuts.*, vol. 19, pp. 2502–2525, Sept. 2017.

[9] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," *RFC 4944*, Sept. 2007.

[10] J. Wang and V. Leung, "A survey of technical requirements and consumer application standards for IP-based smart grid AMI network," in *Proc. ICOIN*, 2011.

[11] Cisco, "Connected grid networks for smart grid - Field area network." http://www.cisco.com/web/strategy/energy/field_area_network.html.

[12] Z. Fan *et al.*, "Smart grid communications: Overview of research challenges, solutions, and standardization activities," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 1, pp. 21–38, 2013.

[13] T. Sauter and M. Lobashov, "End-to-end communication architecture for smart grids," *IEEE Trans. Ind. Electron.*, vol. 58, pp. 1218–1228, Apr. 2011.

[14] V. Altmann, J. Skodzik, F. Golatowski, and D. Timmermann, "Investigation of the use of embedded web services in smart metering applications," in *Proc. IEEE IECON*, Oct. 2012.

[15] N. B. Priyantha, A. Kansal, M. Goraczko, and F. Zhao, "Tiny web services: Design and implementation of interoperable and evolvable sensor networks," in *Proc. ACM SenSys*, 2008.

[16] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling ultra-low power wireless research," in *Proc. IPSN/SPOTS*, Apr. 2005.

[17] J. Hill *et al.*, "System architecture directions for network sensors," in *Proc. ASPLOS*, Nov. 2000.

[18] H.-S. Kim, H. Im, M.-S. Lee, J. Paek, and S. Bahk, "A measurement study of TCP over RPL in low-power and lossy networks," *J. Commun. Netw.*, vol. 17, pp. 647–655, Dec. 2015.

[19] A. Dunkels, "Full TCP/IP for 8-bit architectures," in *Proc. ACM MobiSys*, 2003.

[20] S. Jeong *et al.*, "MAPLE: Mobility support using asymmetric transmit power in low-power and lossy networks," *J. Commun. Netw.*, vol. 20, no. 4, pp. 414–424, 2018.

[21] R. K. Jain, D.-M. W. Chiu, and W. R. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," *ACM Trans. Comput. Syst.*, 1984.

[22] X. Kaixin, G. Mario, Q. Lantao, and S. Yantai, "Enhancing TCP fairness in ad hoc wireless networks using neighborhood RED," in *Proc. ACM MobiCom*, pp. 16–28, Sept. 2003.

[23] J. Ko, S. Dawson-Haggerty, O. Gnawali, D. Culler, and A. Terzis, "Evaluating the performance of RPL and 6LoWPAN in TinyOS," in *Proc. IP+SN*, Apr. 2011.

[24] J. Ko *et al.*,"Beyond interoperability: Pushing the performance of sensor network IP stacks," in *Proc. ACM SenSys*, 2011.

[25] S. Duquennoy, F. Österlind, and A. Dunkels, "Lossy links, low power, high throughput," in *Proc. ACM SenSys*, 2011, pp. 12–25.

[26] T. Zheng, A. Ayadi, and X. Jiang, "TCP over 6LoWPAN for industrial applications: An experimental study," in *Proc. IFIP NTMS*, 2011.

[27] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz, "Improving TCP/IP performance over wireless networks," in *Proc. ACM MobiCom*, 1995, pp. 2–11.

Mingyu Park received his B.S. degree in Computer and Information Communications Engineering from Hongik University in 2017. He is currently a graduate student in Computer Science and Engineering, Chung-Ang University, Seoul, Republic of Korea. He is also a Research Assistant at the Networked Systems Laboratory (NSL) led by Dr. Jeongyeup Paek.

Jeongyeup Paek received his B.S. degree from Seoul National University in 2003 and his M.S. degree from University of Southern California in 2005, both in Electrical Engineering. He then received his Ph.D. degree in Computer Science from the University of Southern California (USC) in 2010 where he was a member of the Networked Systems Laboratory (NSL) led by Dr. Ramesh Govindan. He worked at Deutsche Telekom Inc. R&D Labs USA as a research intern in 2010, and then joined Cisco Systems Inc. in 2011 where he was a Technical Leader in the Internet of Things Group (IoTG), Connected Energy Networks Business Unit (CENBU, formerly the Smart Grid BU). In 2014, he was with the Hongik University, Department of Computer Information Communication as an assistant professor. Jeongyeup Paek is currently an Associate Professor at the School of Computer Science and Engineering, Chung-Ang University, Seoul, Republic of Korea.