Constructing a Green MPTCP Framework for Industrial Internet of Things Applications

Michał Morawski(✉) and Przemysław Ignaciuk

Lodz University of Technology, Lodz, Poland
michal.morawski@p.lodz.pl

Abstract. In the typical distributed applications, the data exchange between the communicating peers proceeds along the transport path established in the connection initialization phase, even if a better one is discovered during the active session. With the recent advancement in the multipath protocol development, e.g., MPTCP, the peers can benefit from a concurrent use of a few channels, thus improving the transmission quality. However, the present approaches to the multipath transfer organization tend to neglect the energy aspects, crucial for resource-constrained Internet of Things (IoT) devices. In this paper, a framework for MPTCP module tuning, targeting the power expenditure, is developed. A new Scheduler and a new Path Manager promoting a conservative energy economy are designed by adopting a formal optimization approach. Moreover, explicit guidelines regarding the TCP variant selection are provided. As confirmed by numerous experiments involving physical devices and real networks, the proposed configuration scheme allows for several percent energy gain with respect to the default one, thus setting a solid framework for green MPTCP-based Industrial IoT communication.

Keywords: Green networking · Industrial Internet of Things · MPTCP

1 Introduction

Industrial control systems have evolved from centralized, through distributed, to Internet of Things (IoT) installations. These changes reflect the ever more challenging requirements of heterogeneity, multi-vendor equipment integration, and management facility. The IoT and Industry 4.0 applications unify various electronic devices in a joint effort to manage an industrial process. While designed to handle different functional aspects, e.g., monitoring, or dynamic adaptation, and created using different tools and technologies, they recur to a common communication platform to realize the intended goals [1]. The industrial applications expect the network to deliver the data within assumed time bounds, which imposes stringent throughput and robustness constraints. When dedicated links are not available, e.g., when the wireless media need to be incorporated, the streams generated by different system components (and different systems) interfere with each other [2, 3]. Then, neither the time, nor throughput requirements cannot be guaranteed, thus downgrading the quality or even compromising the industrial process objectives.

H. Gao et al. (Eds.): BROADNETS 2020, LNICST 355, pp. 22–32, 2021.
https://doi.org/10.1007/978-3-030-68737-3_2

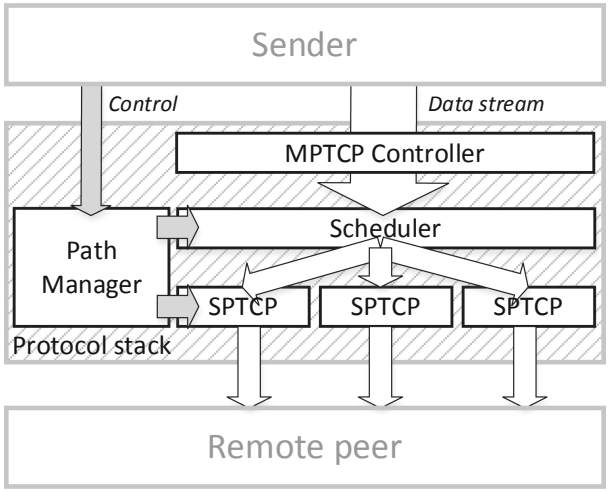

Fig. 1. MPTCP architecture from the sender point of view.

Today, extending the capabilities of a networking device by additional interfaces is relatively inexpensive and easy to perform. Unfortunately, due to the design principles of common communication protocols, the extra interfaces cannot simultaneously direct the data to the same peer.

2 Multipath Communication

A self-imposing solution for obtaining smooth data transfer over the links with variable bandwidth is to simultaneously engage multiple channels. Unfortunately, the most widely network protocol used today – TCP – identify peers using single address-port tuple, thus changing interface leads to the connection break. To overcome this limitation, recently, a multipath version of TCP [4–6] – has become available. The newly standardized Multipath TCP (MPTCP), transparent for the applications and networking devices [4], allows for concurrent data transfer in many channels. From the architectural perspective, MPTCP extends the TCP protocol stack by a few additional modules (Fig. 1). When a client application initiates the transfer of data, the standard TCP handshaking procedure between an arbitrary pair of the logical interfaces at the sender and remote peer is executed. The peers exchange also the information concerning possible paths of data flow. By default, the *Path Manager* module attempts to build as many channels as possible from the pool of available interfaces at the communicating device (*full-mesh* option). As a result of device displacement, noise intensification, or networking failures, a path may need to be closed, or a new one established. The path closure/opening procedure may be executed many times during the communication session, however, the logical MPTCP connection persists until at least one channel is valid. *Path Manager* handles the path creation during the entire MPTCP session.

The data transfer process is managed by a joint effort of three modules: *MPTCP Controller*, *SPTCP Controller(s)*, and *Scheduler*. *MPTCP Controller* dictates the general properties of the data stream, with respect to the throughput, fairness, and buffer allocation objectives. *Scheduler* splits the stream shaped by *MPTCP Controller* over the paths (channels) established by *Path Manager*. Various splitting strategies can be exercised [7–10], however, neither *Scheduler*, nor *Path Manager*, influences the traffic intensity, directly. This task is delegated to *SPTCP Controllers*, acting in each path separately according to the rules of the standard, Singlepath TCP (SPTCP) and the corresponding congestion control algorithms. Those modules, the sender and receiver applications, and the network interact with each other in a complex way, which poses a serious challenge for establishing an efficient communication platform. The design of such a platform, respecting energy and resource constraints of industrial environment, is the focus of this work.

3 Proposed Green MPTCP Framework

The IoT devices are not destined for constant networking activity. They perform actions after a specified time elapses (time-driven systems), or in response to a trigger (event-driven systems). In either case, the application running on the device may power up additional cores, engage accelerators, or increase the clock frequency. Once the processing is finished, the additional components, including the network interfaces, should be powered down to conserve energy. Meanwhile, the MPTCP modules, possibly except *MPTCP Controller*, are typically tuned for maximum throughput, thus for persistent activity. In this work, a systematic tuning methodology of the MPTCP modules so that a low energy profile is achieved is proposed. Each module will be given a separate treatment in latter sections.

Let $k = 0, 1, 2,\ldots$ mark the subsequent instants within the session time t_D when *Scheduler* takes a decision regarding the path selection. k is also a moment of the acquisition of a new segment from *MPTCP Controller*. At that instant, the device processing unit, and other essential components except the network interfaces, e.g., display, or accelerators, are supplied with power $p_D(k)$. In turn, interface $i \in [1, n]$, n – the number of available interfaces at the device, requires power $p_i(k)$ to send (or receive) the data. Both $p_D(k)$ and $p_i(k)$ are measurable, thus, assumed known in the mathematical model. The energy dissipated to handle the MPTCP stream in a single session can be expressed as

$$E = \sum_{i=1}^{n} \sum_{k=0}^{t_i} p_i(k)\Delta\tau(k) + \sum_{k=0}^{t_D} p_D(k)\Delta\tau(k), \tag{1}$$

where t_i is the instant of the last *Scheduler* decision concerning channel i, t_D is the instant of the last decision in the entire session, and $\Delta\tau(k)$ is the time between instants k and $k+1$. $\tau_i = \sum_{k=0}^{t_i} \Delta\tau(k)$ is the time of sending data through interface i, covering all the link layer activities, and $\tau_D = \max_i \tau_i$ is the overall session time.

The objective is to adjust the MPTCP modules so that E is minimized. A major challenge to overcome is the fact that once the data exchange session is initiated one

cannot determine precisely how long it will take to process the data and effectuate the transfer. The session duration depends on the fluctuating networking conditions on the paths, the volume of processed data, and the physical interface specifics.

3.1 Path Manager

The effort necessary to establish multiple paths is not justified when the communicating peers exchange only short messages. However, the decision about setting an additional path needs to be taken in real-time, because the length of the data stream coming from the application in not known in advance by *Path Manager*. In the typical networking scenarios, the initial data burst is limited by the slow-start threshold, set as 10 segments by default. Therefore, if the amount of data in the sender buffer does not exceed this initial burst, additional channels need not be used to avoid energy dissipation for setting or tearing down the paths. In other words, irrespective of the direction of data flow, it is justified to engage a secondary channel only if the data exceeds the initial slow-start threshold. Note that the primary channel (interface) used to establish the MPTCP connection usually promises the best performance. The operation of the proposed green *Path Manager* can thus be summarized as:

Rule 1: Delay opening new channels until the IoT device effectuates the transfer of the initial burst of data (14–15 kB).

3.2 Scheduler

By default, *Scheduler* chooses the path of transmission from the pool created by *Path Manager* using solely the information about the low-pass filtered time between sending a segment and acknowledgement reception – *srtt* (Smoothed Round Trip Time). The power necessary to transfer the data is not taken into account at all. In order to design a green *Scheduler*, in this work, a formal optimization procedure with low computational footprint as a target is conducted. The design basis is set in the following theorem.

Theorem 1: The energy dissipated by the device to handle the application stream (1) is minimum, if all the paths are used simultaneously, i.e., if for every $i, j \in [1, n]$, the time of using the paths $t_i = t_j = t_D$.

Proof: Let us order the transmission times through interfaces 1, 2, …, n as $t_1 \leq t_2 \leq \ldots \leq t_n$. Then, (1) can be written as

$$\begin{aligned} E = &\sum_{i=1}^{n} \sum_{k=0}^{t_1} p_i(k)\Delta\tau(k) \\ &+ \sum_{k=t_1+1}^{t_2} p_2(k)\Delta\tau(k) + \sum_{k=t_1+1}^{t_3} p_3(k)\Delta\tau(k) + \ldots \\ &+ \sum_{k=0}^{t_1} p_D(k)\Delta\tau(k) + \sum_{k=t_1+1}^{t_D} p_D(k)\Delta\tau(k). \end{aligned} \tag{2}$$

The elements in the second row in (2) reflect the energy dissipation in the time exceeding the transfer through channel 1. Therefore, taking into account the fact that $p_i(k)$ and $p_D(k)$ are non-negative, E is the lowest, if those terms disappear. This conjecture implies $t_1 = t_2 = \ldots = t_n = t_D = \max_i t_i$. □

In the sequel, using the implications from Theorem 1, a formula for efficient load balancing to be implemented as the green Scheduler will be derived. Let $h_i(k)$ denote the throughput in channel i at instant k, $h_D(k)$ the throughput of the entire MPTCP stream, and $p_i(k) = w_i(k)v_i(k)$, where $w_i(k)$ [W/b] is the power necessary to transfer a unit of data (e.g., a segment) and $v_i(k)$ is the amount of sent data. $w_i(k)$ aggregates the energy expenditures for the MAC procedures and link layer retransmissions. $w_i(k)$ depends on interface manufacturer, engaged technology, distance to the hub, noise level, and it fluctuates. Recall that $p_D(k)$ is the power invested by the device into all the activities not associated directly with the interface procedures, e.g., managing the display. For the sake of consistency, it will be expressed in a similar way as $p_i(k)$: $p_D(k) = w_D(k)v_D(k)$, with $w_D(k)$ determined from the measurements of $p_D(k)$ and $v_D(k)$.

Let $g_i(k) = w_i(k) + w_D(k)/n$, i.e., the power efficiency of channel i augmented by the power dissipation of other components averaged over interfaces. With this notation, after substituting $p_i(k)$, $p_D(k)$, and $h_i(k)$ into (1), one obtains

$$E(k) = \sum_{i=1}^{n} g_i(k) \frac{v_i^2(k)}{h_i(k)}, \tag{3}$$

and for the entire MPTCP stream

$$E = \sum_{k=0}^{t_D} E(k). \tag{4}$$

In the practical implementation, the scheduling decisions need to be taken in real-time. Since at instant k the power profiles are not known for time $l > k$, the minimum energy expenditure is sought by minimizing (3), i.e., by solving $\partial E(k)/\partial v_i(k) = 0$ individually at each moment k. Consequently, the optimal volume of data to be sent through channel i is calculated as ${v_i}^{opt}(k) = 2g_i(k)/h_i(k)$. As *Scheduler* is not destined to manipulate the stream intensity, rather the split ratio, the data distribution strategy among the channels follows

$$\frac{v_i^{opt}(k)}{v_j^{opt}(k)} = \frac{g_i(k)/h_i(k)}{g_j(k)/h_j(k)}, \tag{5}$$

with the current throughput of channel determined as

$$h_i(k) = \frac{inflight_i(k)}{srtt_i(k)}, \tag{6}$$

where $inflight_i(k)$ is the amount of data already sent, but not yet acknowledged, and $srtt_i(k)$ is the current *srtt* of channel i. Both values are readily available in the TCP stack.

The operation of the proposed green *Scheduler* can be summarized as:

Rule 2: at each instant k, split the MPTCP stream according to (5) with the instantaneous throughput calculated from (6).

3.3 SPTCP Algorithm Selection

In order to satisfy the premises of green networking, one should limit the transfer of superfluous packets. The associated energy expenditure is related both to the direct effort of sending the data, and to the wasted channel bandwidth that could have been used for other purposes. Therefore, green congestion control algorithms, while increasing throughput, should limit the number of transmitted packets.

Unfortunately, by definition, Scheduler does not analyze dropped and retransmitted packets. In the implementation of green Scheduler, this inopportune situation is partly remedied by reading the aggregate in-flight data count $inflight_i(k)$. Nevertheless, the energy economy can further be enhanced by a judicious choice of the SPTCP congestion control algorithm governing the flow rate in the individual channels.

Note that in a common TCP transmission scenario, many packets are dropped, because the internal congestion/bandwidth sensing procedures use loses to shape the transfer profile [11]. There exist, however, TCP variants that incorporate other congestion indicators, e.g., Westwood+ (time between acknowledgments) [12], or BBR (probe pattern timing) [13]. Therefore, for green MPTCP implementation, it is proposed to

Rule 3: Employ an SPTCP congestion control algorithm, which evaluates the traffic conditions using methods not based on drops. As the performed experiments indicate, a particularly good candidate is BBR – it is as effective energetically as Westwood, but tends to attain a higher throughput.

3.4 MPTCP Controller

MPTCP Controller shapes the general properties of the data stream. Typically, it is set to maintain fairness at all costs, i.e., to prevent the MPTCP stream from dominating legacy TCP ones [14]. In the industrial IoT applications, other policies might be more appropriate, e.g., robust controller [15]. However, while choosing the MPTCP congestion control algorithm, one should also consider its impact on the other entities in the network and their energy profiles. The problem of *MPTCP Controller* selection for green networking applications, owing to its multidimensionality, deserves a separate, thorough line of research. In this work, it is left at the default configuration recommended for general networking [5].

4 Experimental Environment

In order to assess the impact of the proposed green MPTCP framework on the actual transmission properties, the test setup illustrated in Fig. 2 has been prepared. It reflects a common scenario in the IoT communication, i.e., agent data collection by a control server [2]. The agent is represented by a low-performance device (here, Raspberry Pi), equipped with two wireless interfaces. As an IIoT data sink, a high-end server placed in the multi-access university data center is considered. Both the client and the server operate under Linux ver. 4.19 with the MPTCP stack ver. 0.95. The agent is modified according to *Rules 1–3*, stipulated in Sect. 3. The uncertain networking conditions of the

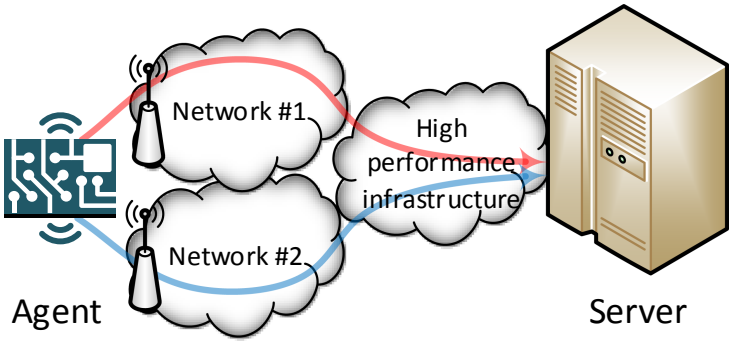

Fig. 2. Experimental setup. Upper path (red) – primary, lower path (blue) – secondary. First mile – wireless links operating in different bands. (Color figure online)

industrial environment are recreated by using crowded, public Internet Service Provider (ISP) network. The ISPs are unrelated, thus the corresponding paths have no common bottleneck – the common infrastructure has capacity ~100 times higher than that of ISPs. The primary path incorporates 8 hops, with the measured throughput 8.7–8.9 Mbps and the initial *srtt* 25–60 ms, whereas the secondary one – 19 hops, 6.0–12.5 Mbps throughput, and 30–120 ms initial *srtt*, respectively. The popular *iperf3* utility is used as the data source.

Two scenarios – reflecting short message exchange and persistent data transfer – are considered. The experiments are repeatable in the sense that they provide similar outcomes in the subsequent runs under a comparable traffic load.

4.1 Scenario 1

The multipath transmission is usually not recommended for short-lived communication due to the energy expenditure concerns [16]. In the IIoT applications, however, the agent-server exchange of the measurement and control data proceeds in short bursts. Therefore, the purpose of the first group of tests is to evaluate the proposed method of protocol tuning so that MPTCP can be used also in the IIoT systems. Consequently, the stream (message) length is constrained to 10 kB. As neither *Scheduler*, nor *SPTCP* congestion control algorithm impact the power efficiency in the case of short streams, Scenario 1 emphasizes the performance of green *Path Manager*.

4.2 Scenario 2

In order to evaluate the green MPTCP framework in a composite way, in the experiments conducted under Scenario 2 streaming data transfer is considered. The test time has been set as 10 s. Such interval is long enough to eliminate measurement bias related to specific networking events, e.g., a noise surge on a wireless link. The following cases concerning the device power profiles w_1: w_2: w_D have been considered: a) 1:2:0 – interface 2 requires twice as much power as interface 1, power for other components disregarded (not measured); b) 1:3:10 – balanced energy dissipation; and c) 1:1:1 – equal

power demands of interfaces and other components – throughput optimization case, i.e., shortening the time of transmission.

The framework settings are the same for either scenario, i.e., the proposed method ensures self-adaptation of the MPTCP components to both short and long-lived transmission in real-time. No *a priori* knowledge of the stream type is required.

Table 1. Measured transmission properties

Parameter	Case	Green setting	Default setting
Throughput [Mbps]	a	15.8–16.3	11.7–14.0
	b	13.6–14.8	10.3–10.9
	c	16.1–18.0	7.28–9.05
Primary path data volume [MB]	a	10.29–10.30	8.19–10.59
	b	10.06–10.25	3.37–7.51
	c	10.11–10.17	5.52–9.52
Secondary path data volume [MB]	a	11.16–12.02	6.44–11.00
	b	8.38–10.11	7.88–11.34
	c	11.83–14.44	2.44–9.54
Primary path drop count	a	4–13	7–70
	b	5–14	7–424
	c	21–41	40–151
Secondary path drop count	a	4–9	108–119
	b	5–11	7–218
	c	16–37	106–366
Dissipated energy [J]	a	4.55–4.67	5.22–6.27
	b	6.79–7-11	8.47–8.60
	c	4.23–4.55	8.42–10.10
Energy gain – green vs. Default [%]	a	11–18	
	b	14–20	
	c	**49–55**	

5 Results and Discussion

The proposed green tuning framework has been compared with the default MPTCP configuration [4] in numerous experiments. Each test was executed 30 times. The results have been summarized in Table 1, with an example case (b) selected for closer examination in Fig. 3.

The measurements obtained in Scenario 1 indicate lower energy dissipation in the range 3–10% of the green MPTCP with respect to the default setting. Although the actual savings depend on the w_i ratio and channel $srtt_i$, and vary according to the current network capabilities, the gain of applying the green MPTCP components, notably the green *Path Manager*, is always observed. On average, it amounts to ~5% when short data exchange in the IIoT environment takes place.

In Scenario 2, a number of additional benefits can be observed, as documented in Table 1. First of all, thanks to the application of a modern congestion control algorithm (BBR), the drop count indeed diminished, which implies no energy expenditure for spurious retransmissions. Secondly, although tuned independently from other modules, the green *Scheduler* happens to cooperate in synergy with the SPTCP algorithms, boosting further the power efficiency. It occurs that using BBR as the SPTCP congestion control algorithm helps the *Scheduler* evaluate the current throughput $h_i(k)$ in a more reliable way and, hence, choose a better channel for data transfer. In the default MPTCP setting, the channel selection is based on the *srtt* value, only, which leads to energetically inefficient load balancing.

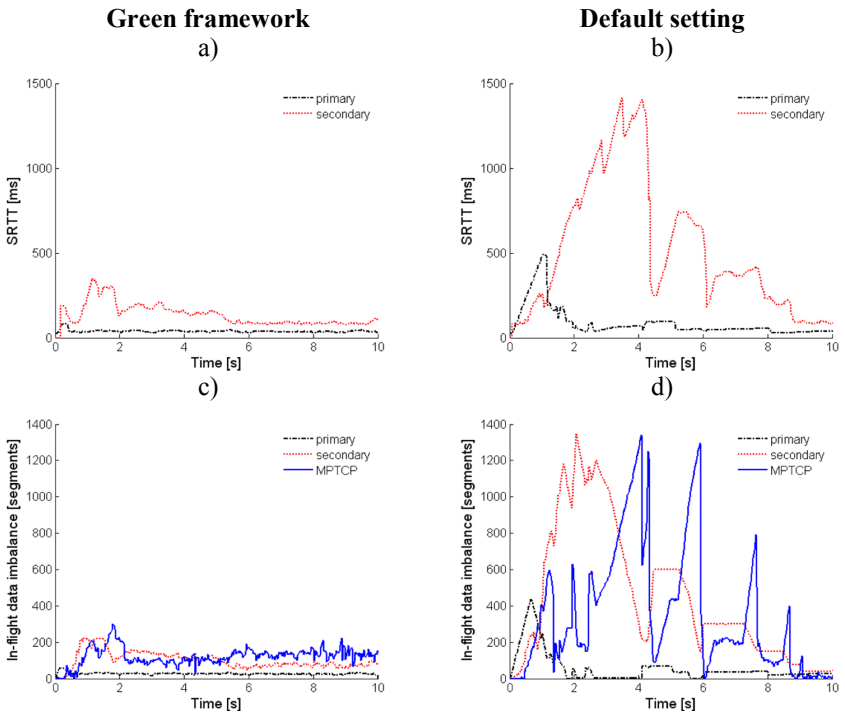

Fig. 3. Transmission properties (Scenario 2, case b): a), b) *srtt* evolution, c), d) *inflight* data.

Probably, the biggest obstacle in achieving the desired power conservation in multipath data transfer is Head-of-Line (HoL) blocking phenomenon, i.e., the situation when the stream reconstruction at the receiver is stalled due to missing segments on certain

paths. HoL prolongs the transmission time and might bring down the entire multipath transfer system. A good, quantitative measure of this inopportune phenomenon is flow imbalance, calculated as the difference of the in-flight data seen at the MPTCP level with respect to the path level $inflight_D(k) - \sum_{i=0}^{n} inflight_i(k)$. In a properly operating multipath system it should be as low as possible. As depicted in Fig. 3(c, d), in a typical traffic scenario, the flow imbalance in the green MPTCP setting is nearly an order of magnitude lower than in the case of the default configuration. It relates, again, to the improper (from the energy point of view) estimation of the temporary channel conditions by the MPTCP components in their default setting.

The default MPTCP *Scheduler*, using the initial *srtt* readings (Fig. 3b, 0–1 s), overestimates the capabilities on the secondary path and directs too many data segments into that channel. It results in a large *srtt* increase (Fig. 3b, 2–4 s). As a consequence, the primary path becomes temporarily suspended, as MPTCP needs to wait for the data sent through the sluggish, secondary channel (though it initially promised a lower *srtt*). The described situation is particularly important for IIoT applications, as the majority of data exchange is likely to be concluded in the first couple of seconds. The green *Scheduler* does not fall for this bait of a "lower-*srtt* channel". Conversely, it quickly corrects the initial erroneous bandwidth estimation, and the data goes through the primary, uncongested channel.

6 Conclusions

MPTCP, intended as a replacement of the single-path TCP, promises better performance and higher reliability in the general networking context. However, which is crucial in the IIoT systems, it does not come up to the energy efficiency expectations of its predecessor. Moreover, if improperly tuned, MPTCP may actually arrive at worse transmission properties due to HoL blocking.

In this paper, a systematic tuning procedure for improving the energy efficiency of MPTCP data transfer is proposed. Precise guidelines for all major components of the protocol stack are provided and verified in strenuous tests involving physical devices and shared networks. For *Path Manager*, contrary to the default full-mesh option, it is recommended to refrain from opening spurious paths. *Scheduler*, in turn, should split the application stream taking into account both the power needed to transfer the data at the interfaces and the power needed to operate the device, as well as the current throughput at each path. The splitting rule for the green *Scheduler* has been established here through a formal optimization procedure. Finally, *SPTCP Controllers* should use the algorithms that minimize the number of drops, e.g., BBR, so that resources are not wasted on retransmissions. Although treated separately, the proposed adjustments happen to reinforce each other in increasing the overall energy gain. As stated in the measurement summary in Table 1, one can achieve an energy boost in the range of several percent in the case of a common IoT scenario. When throughput maximization is the primary objective, the gain from applying the proposed green tuning framework is even bigger – it can be as high as 50% with respect to the default configuration (marked bottom row in Table 1.

All the presented rules are straightforward to implement at a physical device, with minor code complexity and negligible increase of computational power. In this way, the fundamental principles of IIoT system communication are fulfilled.

References

1. Koutsiamanis, R.-A., Papadopoulos, G.Z., Fafoutis, X., Del Fiore, J.M., Thubert, P., Montavont, N.: From best effort to deterministic packet delivery for wireless Industrial IoT networks. IEEE Trans. Ind. Inf. **14**(10), 4468–4480 (2018)
2. Xu, L., He, W., Li, S.: Internet of Things in industries: a survey. IEEE Trans. Ind. Inf. **10**(4), 2233–2243 (2014)
3. Willig, A.: Recent and emerging topics in wireless industrial communications: a selection. IEEE Trans. Ind. Inf. **4**(2), 102–124 (2008)
4. Barré, S., Paasch, C., Bonaventure, O.: MultiPath TCP: from theory to practice. Technical report, Université Catholique de Louvain (2011)
5. Ford, A., Raiciu, C., Handley, M., Bonaventure, O., Paasch, C.: TCP extensions for multipath operation with multiple addresses. RFC 8684 (2020)
6. Barré, S., Paasch, C.: MultiPath TCP – Linux kernel implementation. https://www.multipath-tcp.org
7. Paasch, C., Ferlin, S., Alay, O., Bonaventure, O.: Experimental evaluation of multipath TCP schedulers. In: Proceedings of the ACM SIGCOMM CSWS, Chicago, USA, pp. 27–32 (2014)
8. Ferlin, S., Alay, Ö., Mehani, O., Boreli, R.: BLEST: blocking estimation-based MPTCP scheduler for heterogeneous networks. In: Proceedings of the IFIP Networking Conference Workshops, Vienna, Austria, pp. 431–439 (2016)
9. Frommgen, A., Erbshäußer, T., Buchmann, A., Zimmermann, T., Wehrle, K.: ReMP TCP: low latency multipath TCP. In: 2016 IEEE International Conference on Communications (ICC), Kuala Lumpur, pp. 1–7 (2016)
10. Ferlin, S., Kucera, S., Claussen, H., Alay, Ö.: MPTCP meets FEC: supporting latency-sensitive applications over heterogeneous networks. IEEE/ACM Trans. Netw. **26**(5), 2005–2018 (2018)
11. Afanasyev, A., Tilley, N., Reiher, P., Kleinrock, L.: Host-to-host congestion control for TCP. IEEE Commun. Surv. Tutor. **12**(3), 304–342 (2010). 3Q
12. Mascolo, S., Casetti, C., Gerla, M., Sanadid, M.Y., Wang, R.: TCP westwood: bandwidth estimation for enhanced transport over wireless links. In: Proceedings of the ACM Mobicom, Rome, Italy (2001)
13. Cardwell, N., Cheng, Y., Gunn, C.S., Yeganeh, S.H., Jacobson, V.: BBR: congestion-based congestion control. ACM Queue **14**(5), 20–53 (2016)
14. Xu, C., Zhao, J., Muntean, G.: Congestion control design for multipath transport protocols: a survey. IEEE Commun. Surv. Tutor. **18**(4), 2948–2969 (2016)
15. Ignaciuk, P., Morawski, M.: Discrete-time sliding-mode controllers for MPTCP networks. IEEE Trans. Syst. Man Cybern. Syst. **50**(2) (2020, in press)
16. Deng, S., Netravali, R., Sivaraman, A., Balakrishnan, H.: WiFi, LTE, or both? Measuring multi-homed wireless internet performance. In: Proceedings of the ACM IMC, Vancouver, Canada, pp. 181–194 (2014)