Energy Efficient MPTCP Transmission Over Channels with a Common Bottleneck

Michał Morawski(✉) and Przemysław Ignaciuk

Institute of Information Technology, Lodz University of Technology,
215 Wólczańska St., 90-924 Łódź, Poland
{michal.morawski,przemyslaw.ignaciuk}@p.lodz.pl

Abstract. The majority of networking devices currently available are equipped with a few communication interfaces. However, the design of the fundamental transport protocol – TCP – prohibits using them simultaneously. Recently, a variant of TCP – Multipath TCP (MPTCP) – that allows for parallel transmission over different paths has been developed. The protocol itself does not define any particular way of splitting the application data stream. Taking into account the proliferation of mobile terminals, a good quality indicator is the amount of energy dissipated by the communicating device. The previous works in the field assume that the paths established by the MPTCP stack are independent. This paper discusses the theoretical and practical aspects of stream splitting when the paths influence each other through a common bottleneck. A comparison with the reference solution is provided. The tests conducted in the public networking environment show a substantial improvement in energy economy.

Keywords: Multipath TCP · Load balancing · Energy efficiency

1 Introduction

Nowadays, the networking devices, like phones, laptops, or servers, have installed multiple communication interfaces (NICs), e.g., Ethernet, WiFi, or cellular. However, the design restrictions of the protocols created in the past permit only one interface to be actively engaged in serving the application data stream at a time. The other interfaces are incorporated if the active one fails, or if the application dictates a switch according to certain quality measure. Such behavior is no longer satisfactory owing to the widespread use of radio technologies. Trying to maintain the ongoing session within a chosen radio channel, with highly variable temporal characteristics (determined by the distance from the access point, nearby obstacles, node mobility, interferences, etc.), has negative impact on the user's experience [1,2]. Instead, one would opt for a communication solution in which the traffic is dynamically distributed among the available interfaces, with explicit consideration of their current transfer capabilities.

The recent proposals, like Multipath TCP (MPTCP) [3–5] promise to ensure truly parallel and load-balanced traffic distribution. However, the reference implementation of MPTCP [6] covers only the general protocol issues, leaving area for further improvements and adjustments. One of such unresolved challenges – addressed in this work – is formulating a method of MPTCP stream splitting into multiple sub-streams so that selected performance criteria are satisfied. As the popularity and functional spectrum of mobile computers have outgrown the progress in battery design, reducing the energy expenditure becomes a critical factor to consider within the ever more stringent service and environmental constraints (green networking) [7]. Therefore, in the presented approach, the emphasis is placed on improving the end user's experience through better energy economy while handling the transmission at the communicating terminals.

At a mobile device, the amount of dissipated energy depends both on the power efficiency of NICs and application level activity, i.e., turning on and off the display, triggering additional cores, GPU, or external sensors. For instance, one can observe faster battery depletion when the data is sent though a secure connection rather than in a plain-text form. The power drain rate also increases when the information regarding the transferred data is rendered on the screen. Therefore, in order to reduce the energy expenditure while effectuating the network connectivity, one should shorten the time of data transfer and allow the user to switch off the screen (and additional hardware components), or put the application in an idle state. This paper proposes a new load-balancing solution for the MPTCP scheduler, particularly well-suited for battery-powered devices equipped with a few independent interfaces. Although addressing similar problems as in the current literature [8–11], the idea presented here differs both in the objectives and in the way the traffic is distributed among the interfaces. The paper focuses on the situation when the sub-streams directed through different logical interfaces influence each other before reaching the destination, The transmission through independent channels is considered as a special case, only. The way the MPTCP stream is divided into sub-streams by the scheduler is determined as a solution of optimization problem set in a mathematical framework of power dissipation. The performance assessment and comparison with the reference scheduler is not restricted to simulations but involves public networks and real networking devices.

2 Multipath Data Transfer

The system architecture is illustrated in Fig. 1. In the considered communication scenario, the user application seeks to send B bits of data to a remote peer. The data is placed in the buffer held by a local MPTCP controller. The MPTCP controller opens sub-channels and distributes the traffic among the corresponding sub-streams that are left under the control of the ordinary, single path TCP (SPTCP). The SPTCP flow control is executed separately for each sub-channel. The research objective is to design an efficient load-balancing algorithm for the MPTCP traffic scheduler. The operation of SPTCP within the sub-channels, e.g., CUBIC, is not affected.

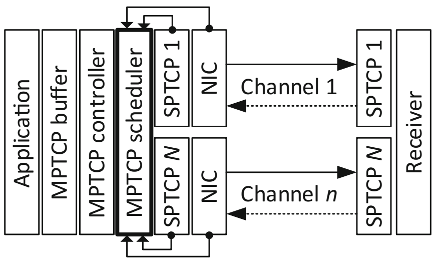

Fig. 1. MPTCP architecture with local feedback – the current power profile of NIC and SPTCP internals. Solid line – data, dashed line – acknowledgments.

The specification of MPTCP does not define how the user's data should be distributed among the sub-channels. Sometimes, it is appropriate to use only one path and switch to another only when the quality of service drops below a predefined level. The primary intention of MPTCP, however, is to employ multiple channels simultaneously. In its reference implementation [6], three schedulers have been defined:

(a) round-robin – evenly distribute subsequent segments among the sub-channels (insignificant in practice, serve as a theoretical baseline),
(b) redundant – send the same segment through all the available sub-channels – results in low latency but leads to increased bandwidth consumption,
(c) default – send the data using the channel with the lowest smoothed round-trip time (SRTT). Smoothing is performed by the filter typical for the selected SPTCP dialect.

Solution (c) – proposed to increase the total effective throughput – is strongly recommended by the authors of implementation [6]. In this paper, this solution is treated as a reference one. Other schedulers described in the current literature, e.g., [8–11], have not been adopted in the formal development works. On the other hand, the default scheduler suffers from a number of disadvantages:

- *sticky* behavior [12] – when the smoothed round-trip time (SRTT) differs a little among the active channels, the one with the shortest SRTT value is used exclusively,
- imprecise SRTT estimates – and thus wrong channel selection – originating from the buffer bloat (different, but generally large, queue lengths in the buffers of intermediate devices on the established paths),
- Karn's algorithm implications,
- spurious acknowledgments – if the channels differ much in their properties, then the retransmission timeout (RTO) at the MPTCP level can be shorter than the variance of SRTT in the 'worse' ones; then, the MPTCP stack mistakenly signals the sender that a segment has been lost;
- head-of-line blocking – if the buffers are too short at the end-points, then some data cannot be temporary sent via the available channel because there are unacknowledged data transmitted previously through the 'worse' ones.

3 Energy Considerations and Optimal Load Distribution

In the mobile devices, the GPU, CPU, and display tend to consume more energy than the network interfaces. The NIC energy expenditure depends on the transmission power, noise level, number of retransmissions, and control plane activity. These factors differ among the interfaces and their mode of operation. In general, the transmit power is highly variable in time [13], but even the median measurements differ much. The power effectiveness for the transmission can be as low as 2 [mW/Mbps] for cable NIC (e.g., Ethernet), and as high as 440 [mW/Mbps] for LTE NIC. Similarly, the power necessary to keep the interface in the operational state can be as low as 90 [mW], and as high as 1440 [mW], for the same interfaces, respectively [10]. Hence, the energy dissipated by the NICs does not have to be comparable with the energy consumed by the operating system and application activities (200–300 mW in low-end systems up to 300 W in high-end ones). Therefore, in order to reduce the overall energy expenditure during the data transfer the following optimization problem may be considered – minimize ϑ given by

$$\vartheta = \sum_{i=1}^{n} \vartheta_i + D \max T_i \tag{1}$$

subject to $T_i > 0$, where

$$\vartheta_i = \int_0^{T_i} p_i(t)dt \tag{2}$$

is the total energy dissipated by interface $i \in [1, n]$, n being the number of interfaces, $p_i(t) > 0$ is the power necessary to transmit the data through this interface at time t, T_i is the total transmission time at this interface, and $D > 0$ is the power consumed by non-NIC system components (display, additional cores, sensors, etc.). D can be obtained as a difference between the power consumed at a given moment and the power drained when the system is in an idle state. Meanwhile, the transmit power $p_i(t)$ [W] is a function of the power efficiency of NIC – $\rho_i(t)$ [W/bit], the number of bits sent through channel i – b_i, and the power necessary to keep the NIC in the operational state – w_i [W]. It can be expressed as

$$p_i(t) = \rho_i(t)b_i + w_i(t), \tag{3}$$

with $\rho_i(t)$ and $w_i(t)$ being functions of:

– the distance from the end-point to the hub (access point, head-end, or BTS),
– the mode of operation, which includes the selection standard ‘b’, ‘g’, ‘n’, ‘ac’, band, RTS, or CCA mode in WiFi and band, GPRS, or the revision of the UMTS class in the case of cellular interfaces,
– interferences level, number of retransmissions at the link layer.

Due to large number of factors affecting $\rho_i(t)$ and $w_i(t)$, these profiles should be directly measured by the communication end-point (e.g., such approach is suggested in [8]). The profile varies with time as a result of node mobility, controller activity, or other terminals behavior. Here, piece-wise constant evolution is assumed.

4 Analytical Background

Often, the paths established by the MPTCP subsystem are independent, i.e., there is no common bottleneck between the communicating peers. An energy-oriented solution for the independent channels case has been presented in [13]. However, such premise is not always justified. The streams may encounter a common bottleneck, or interfere through a third-party connection. The mutual dependency may be persistent or temporary, and may result in network or application performance decrease and is both difficult to predict and not necessarily related to a common protocol framework. In the conducted tests: identical connection end-points, transmission proceeding at the same time, yet using different protocols; the paths differ significantly in SRTT (ipv4: 40–110 ms and ipv6: 60–300 ms) and number of hops (ipv4: 9 hops and ipv6: 17 hops). Moreover, as the ipv6 traffic traverses the potentially congested international backbone, it is more likely to be exposed to throughput fluctuations.

In this paper, extremum seeking control [14] is used as a basis to find a solution to problem (1) when the channels cannot be treated as independent of each other. The idea behind extremum seeking control is to obtain optimal system performance (according to a chosen measure) for an uncertain, possibly non-linear dynamic system by probing it through a known excitation injected into the control input. By observing the system response and the gradient of performance index it is possible to retrieve the unknown plant properties. In the typical implementation, the estimate of performance index gradient is obtained through a periodic excitation applied to linear system representation. Usually, this signal is much faster than the nominal plant dynamics. The response induced by the injected perturbation can then be removed by a low-pass filter (Fig. 2).

Owing to the hard non-linearities and delays, the classical extremum seeking approach cannot be directly applied to optimize the MPTCP transmission. Therefore, in this work, a modified method is elaborated. In the classical approach, by observing the performance index gradient, one estimates the unknown system parameters. In the method proposed here, the system parameters (path properties) are determined with high accuracy and thus assumed known, whereas the gradient measurement is used to decide whether the mutual dependency among the MPTCP sub-channels occurs, or not. The channels are deemed unrelated (independent) if the gradient is zero. Otherwise, an appropriate action needs to be taken, as illustrated in Fig. 2. In the discussed method, in order to assess the extent of mutual dependency of the sub-streams, the gradient of performance index (1) needs to be evaluated. It is assumed that the device establishes two paths only, as is the most common situation in the considered application area. The power efficiency is improved by changing the sub-channels load so

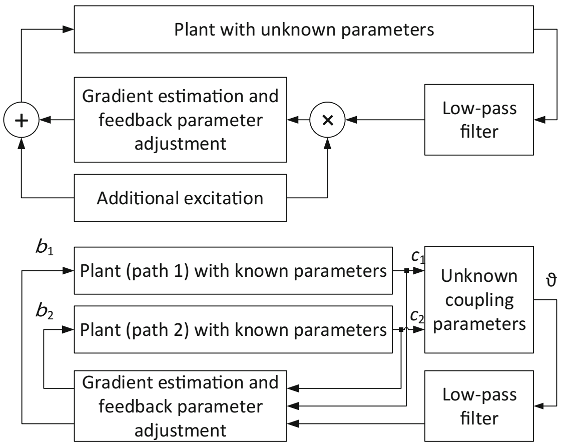
Plant with unknown parameters
Gradient estimation and feedback parameter adjustment
Low-pass filter
Additional excitation
+
×
b_1
b_2
Plant (path 1) with known parameters
Plant (path 2) with known parameters
c_1
c_2
Unknown coupling parameters
ϑ
Gradient estimation and feedback parameter adjustment
Low-pass filter

Fig. 2. Classical approach to extremum seeking control [14] (above) and the proposed one (below) with the additional excitation replaced by a legitimate signal from other channels.

that the gradient $\nabla\vartheta = \dot{\vartheta} \leq 0$. Since the performance index $\vartheta(t) = f(c_1(t), c_2(t))$, the chain rule applied to (1) leads to

$$\dot{\vartheta} = \frac{\partial\vartheta}{\partial c_1}\dot{c}_1 + \frac{\partial\vartheta}{\partial c_2}\dot{c}_2 \leq 0. \tag{4}$$

Assuming p_i and D change sufficiently slow, the energy expenditure related to channel i may be approximated as $\vartheta_i = \int_0^{T_i} p_i(t)dt \approx p_i T_i$. Then, rewriting (1) with $T_i \approx b_i/\mathrm{av}(c_i)$ and $\max T_i \approx (b_1 + b_2)/[\mathrm{av}(c_1) + \mathrm{av}(c_2)]$, one obtains

$$\vartheta = \rho_1 \frac{b_1}{\mathrm{av}(c_1)} + \rho_2 \frac{b_2}{\mathrm{av}(c_2)} + D_w \frac{b_1 + b_2}{\mathrm{av}(c_1) + \mathrm{av}(c_2)}, \tag{5}$$

where $\mathrm{av}(c_i) = f_i/\tau_i$ is the average throughput of channel i obtained as the fraction of *in-flight* (i.e., sent but not acknowledged yet) data (f_i) and SRTT (τ_i). D_w equals $D + w_1 + w_2$. The quantities f_i and τ_i are available in any contemporary SPTCP stack.

Applying multivariable chain rule (4) to (5), the gradient is determined as

$$\nabla\vartheta = \dot{\vartheta} = -\rho_1 \frac{b_1}{\mathrm{av}^2(c_1)}\dot{c}_1 - \rho_2 \frac{b_2}{\mathrm{av}^2(c_2)}\dot{c}_2 - D_w \frac{b_1 + b_2}{[\mathrm{av}(c_2) + \mathrm{av}(c_2)]^2}(\dot{c}_1 + \dot{c}_2). \tag{6}$$

The term $\rho_i b_i/\mathrm{av}^2(c_i)$ in (6) can be physically interpreted as the inverse of energy efficiency of path i and $D_w(b_1 + b_2)/[\mathrm{av}(c_1) + \mathrm{av}(c_2)]^2$, as the inverse of energy efficiency of the communicating device as a whole.

Since by definition ρ_i, c_i, b_i, and D_w are all positive and bounded, according to (6), the gradient follows the changes of the channel capacity (with reverse sign) and the energy dissipation decreases with $\left|\dot{\vartheta}\right|$ increasing. Therefore, the minimum energy dissipation – the largest gradient – is achieved by transferring the data as fast as possible, i.e., when $\dot{c}_i \to \infty$. On the other hand, the channels have limited capacity and the derivatives in Eq. (6) cannot be positive all the time. In particular, when the common bottleneck is encountered, then the derivatives $\dot{c}_i$ differ in sign, and the gradient value is determined by the power efficiency of the transmitting device.

Remark 1. In some publications, e.g., [15], the use of correlation is advocated to assess the channel coupling. The correlation-based methods, however, require long measurement history to infer about the channel dependency due to temporal differences (mainly SRTT fluctuations). For the practical reasons, a method with short memory is preferable, as proposed in this work.

Reordering the terms in (6) yields

$$\begin{aligned}&-\rho_1 \tfrac{b_1}{\mathrm{av}^2(c_1)}\dot{c}_1 - \rho_2 \tfrac{b_2}{\mathrm{av}^2(c_2)}\\ &-D_w \tfrac{b_1}{[\mathrm{av}(c_1)+\mathrm{av}(c_2)]^2}\left(\dot{c}_1+\dot{c}_2\right) - D_w \tfrac{b_2}{[\mathrm{av}(c_1)+\mathrm{av}(c_2)]^2}\left(\dot{c}_1+\dot{c}_2\right) \le 0.\end{aligned} \tag{7}$$

Thus,

$$\frac{b_1}{b_2} \ge -\frac{\rho_2 \frac{1}{\mathrm{av}^2(c_2)}\dot{c}_2 + D_w \frac{1}{[\mathrm{av}(c_1)+\mathrm{av}(c_2)]^2}\left(\dot{c}_1+\dot{c}_2\right)}{\rho_1 \frac{1}{\mathrm{av}^2(c_1)}\dot{c}_1 + D_w \frac{1}{[\mathrm{av}(c_1)+\mathrm{av}(c_2)]^2}\left(\dot{c}_1+\dot{c}_2\right)}, \tag{8}$$

if the denominator is positive and

$$\frac{b_1}{b_2} \le -\frac{\rho_2 \frac{1}{\mathrm{av}^2(c_2)}\dot{c}_2 + D_w \frac{1}{[\mathrm{av}(c_1)+\mathrm{av}(c_2)]^2}\left(\dot{c}_1+\dot{c}_2\right)}{\rho_1 \frac{1}{\mathrm{av}^2(c_1)}\dot{c}_1 + D_w \frac{1}{[\mathrm{av}(c_1)+\mathrm{av}(c_2)]^2}\left(\dot{c}_1+\dot{c}_2\right)}, \tag{9}$$

otherwise.

In the steady state, when $\dot{c}_i \to 0$, applying L'Hôpital's rule to (8) (or (9)), one obtains

$$\frac{b_1}{b_2} = -\frac{\rho_2 \frac{1}{\mathrm{av}^2(c_2)} + D_w \frac{1}{[\mathrm{av}(c_1)+\mathrm{av}(c_2)]^2}}{\rho_1 \frac{1}{\mathrm{av}^2(c_1)} + D_w \frac{1}{[\mathrm{av}(c_1)+\mathrm{av}(c_2)]^2}}. \tag{10}$$

Inequalities (8)–(10) define the scheduler for coupled channels. If $b_1 > b_2$ the current segment ought to be transmitted by channel 1, and by channel 2, otherwise.

5 Implementation Issues

For the purpose of verification of the developed load-balancing strategies a new Linux kernel module implementing (8)–(10) has been created. In addition to this new scheduler, the Linux implementation encompasses the standard path-manager that establishes the paths along which the data will be directed and MPTCP master controller that targets the fairness issues. Since the objective of this work is to provide a method of stream division, only the scheduler has been modified with respect to the reference MPTCP implementation [6]. The other modules are left unaltered with the parameters set at their defaults, i.e., the path manager operates in the full-mesh mode (the number of sub-streams is the product of the number of logical interfaces at both peers) and no specific master control is executed. The standard Linux congestion control algorithm (CUBIC) supervises the individual SPTCP sub-streams.

In contrast to the default scheduler, in the proposed approach, the previously selected paths influence the present scheduling decisions. To properly evaluate (8)–(10), the scheduler uses the internal SPTCP variables (f_i and τ_i). In addition, it needs to track variable c_i to obtain $\dot{c}_i$. For that purpose, a recurrent filter (typical for the TCP implementations) is used, namely,

$$\dot{c}_i(k) = (1-\alpha)\,\dot{c}_i(k) + \alpha\,(c_i(k) - c_i(k-1))\,, \tag{11}$$

where k denotes the subsequent time instants of $\dot{c}_i$ evaluation. The averaging coefficient α has been experimentally chosen as 1/16. In order to obtain the allocation decision, scheduler (8)–(10) performs two iterations per MPTCP segment (*mptcp_for_each_sk* loop): the first one provides the sub-stream coefficients ($\dot{c}_i$), the second one finds the best channel for data transfer. The details of the kernel module organization and power measurement one can find in [13], where the case of independent channels has been considered. In the tests described here in Sect. 6, the following artificially injected patterns have been applied: (a) constant, (b) linear increase/decrease (simulates moving away, or getting closer to a link layer hub), (c) stepwise (simulates handover, or getting in or out of a radio shadow), and (d) oscillatory (simulates channel fading). Consequently, in addition to already covering a broad spectrum of practical networking situations, new profiles can be seamlessly introduced into the applied framework for conducting a variety of tests.

6 Evaluation

The developed scheduling solution has been verified using the setup based on the general system architecture depicted in Fig. 1. The popular low-end Raspberry Pi device has been chosen as the MPTCP client and a high-end virtual machine in the local data center (in the *cloud*) as the server, which corresponds to the typical use pattern in the MPTCP networking. During all the tests the server runs unmodified, following the standardized MPTCP rules. In all the experiments, the public network (neither Intranet, nor closed simulations) is employed. While in such environment a single measurement is hardly repeatable, a large number of trials allows one to filter the artifacts and gives a sound basis for the protocol assessment in a realistic rather than artificially created lab environment. To minimize the influence of channel temporal variations and give statistically meaningful results, the tests have been repeated multiple times.

6.1 Test Scenarios

Two channel combinations have been tested. In the first one – scenario 1 – the probability of a common bottleneck is low, in the second – scenario 2 – high. The first scenario represents the case of a smartphone, or an IoT device [16,17], communicating with the server, with one interface connected to a cable network and the second inter-face connected to an LTE network. The cable and LTE networks are unrelated. The measured initial SRTT of the cable network channel varies within 70–85 ms and the segments traverse 7 hops, whereas for the LTE connection the SRTT fluctuates within 80–400 ms and the segments encounter 9 hops before reaching the destination. In the second scenario, the same interface is used by the endpoints, yet the channels differ in the applied network protocol. Such paths frequently influence each other. During the tests, the first path consists of 9 hops and the measured SRTT varies in the interval 40–110 ms and the second path covers 17 hops with the SRTT fluctuating in the range 60–300 ms.

6.2 Test Case Conditions

In all the tests, $D_w = 600$ [mW], $\rho_1 = 30$ [mW/Mb], $\rho_2 = 400$ [mW/Mb] (scenario 1) and $\rho_1 = \rho_2 = 140$ [mW/Mb] (scenario 2). A single test case (in both scenarios) involves an *iperf3* generated stream controlled first by the default scheduler, next by the proposed energy-aware one. The tests are performed at different days and times, and using different power profiles. As already mentioned, in order to obtain statistically relevant results the tests are repeated multiple times and are executed at different times and days. During all the tests the gain is defined as

$$\mathrm{G} = \left(\vartheta^{\text{default}} - \vartheta^{\text{proposed}}\right) / \vartheta^{\text{default}}, \tag{12}$$

where $\vartheta^{\text{default}}$ is the energy dissipated when the default scheduler operates, whereas $\vartheta^{\text{proposed}}$ refers to the energy expenditure of scheduler (8)–(10).

6.3 Test Results

Selected test results are presented in Table 1. The advantages of using the proposed power-aware scheduler are highlighted when the amount of fluctuations (congestion incidents, interferences, etc.) in the network grows. The advantages of the discussed solution are particularly visible during rush hours (7–8 PM). In scenario 1, the proposed scheduler outperforms the default one in terms of lower energy expenditure (10–36%), and occasionally in terms of throughput. In an uncommon situation, when the channel with a shorter SRTT dissipates more energy (LTE), it is possible to obtain the gain as high as 90%. In scenario 2, the power-aware scheduler gives 4–10% better results (maximum 12%) with respect to the reference one.

Table 1. Experiment results. T – Time of transmission [s], ϑ – dissipated energy [J], G – gain in applying energy-aware solution [%]. R – rush hours (7–8 PM), E – Early morning (5–6 AM), O – ordinary hours (1 PM–4 PM).

Scenario 1					Scenario 2					When
Default		Proposed			Default		Proposed			
T	ϑ	T	ϑ	G	T	ϑ	T	ϑ	G	
8.58	7.99	7.99	7.99	**8**	9.56	6.45	9.50	6.41	**1**	E
8.20	8.01	8.01	8.01	**11**	9.52	6.37	9.45	6.35	**0**	E
9.35	9.02	9.02	9.02	**3**	10.62	7.16	10.07	6.91	**4**	O
9.82	8.34	8.34	8.34	**14**	10.66	7.19	10.20	6.97	**3**	O
11.59	9.12	9.12	9.12	**21**	9.95	6.60	9.72	6.56	**1**	O
22.55	16.1	16.1	16.1	**23**	13.07	7.71	11.41	7.06	**8**	R
22.82	15.5	15.5	15.5	**36**	12.93	7.55	11.52	7.12	**6**	R

During ordinary hours (12 AM–4 PM), in scenario 1, the gain of using the energy-aware schedulers oscillates between 3% and 25% and differs much in subsequent test runs. Nevertheless, a case when the gain is negative has not been observed. In scenario 2, the power saving is smaller gain is smaller than in rush hours yet still meaningful with respect to the default solution.

When the network traffic is stable and low (e.g. in the early morning – 5–6 AM) the proposed scheduler provides limited gain. The allocation decisions are highly influenced by the buffer-bloat phenomenon (BB). While other network users are absent, large buffers in the intermediate devices prohibit the *cwnd* reduction at the end-points and SRTT increases. When the default scheduler controls the segment-to-path assignment, the energy greedy interface (LTE) is erroneously chosen over the more economic one if the SRTT, elongated by BB, exceeds the transfer time in the expensive one. The proposed scheduler is immune to the BB effects – see also [18]. When the throughput does not undergo significant variations ($\dot{c}_i$'s are close to zero), then the proposed scheduler operates in the steady-state conditions according to (10). Except for the segment ordering there are no substantial differences comparing to the reference solution.

6.4 Detailed Analysis

An example run of scheduler (8)–(10) under scenario 2 is illustrated in Fig. 3. The established paths have similar initial SRTT and they are exposed to BB (SRTT grows from the initial 52 ms for channel 1 and 69 ms for channel 2, respectively, up to 400 ms as a result of buffering in the intermediate routers). At the beginning of the transmission, the scheduler is in the steady state (the split ratio is constant) and it allocates the channels according to (10). At $t \approx 4$ s, stream 2 experiences a congestion incident. Both the *in-flight* data (f_2) and throughput (c_2) are reduced, thus $\dot{c}_2 < 0$. Then, according to rule (9),

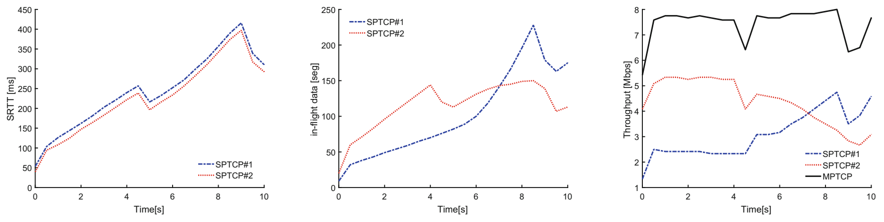

Fig. 3. Performance of the proposed scheduler in scenario 2, sampling period 0.5 s: SRTT (left), in-flight data (middle), throughput (right).

the transfer speed in stream 2 is throttled, and in stream 1 it accelerates. Since the sum $\mathrm{av}(c_1) + \mathrm{av}(c_2)$ is approximately constant, the stream 2 cannot increase its speed and the transmission proceeds mainly in channel 1.

6.5 Beneficial Side Effects

As a beneficial side effect, it has been noticed that even if the power coefficients are not considered in the allocation decisions ($\rho_1 = \rho_2 = 1$, $D = D_w = 0$ in (8)–(10)), the energy-aware scheduler gives on average a few-percent improvement over the default one. This gain increases with the throughput fluctuations. It is attributed to the BB reduction (as noticed also in the case of SPTCP in [18]). Moreover, without any tricky heuristics, or link layer dependency [6], the jitter is constrained to 2–4 segments and fast response to the channel variability is obtained. In consequence, the redundant scheduler from the standard implementation [6] need not be applied, low buffer occupancy is observed, and the risk of head-of-line blocking is limited. All these favorable characteristics are obtained at a small computational cost.

7 Summary and Conclusions

The paper presents a new stream splitting algorithm for the MPTCP traffic scheduler for the paths encountering a common bottleneck. In the proposed solution, the load over the available network interfaces is distributed so as to achieve low energy consumption. Thus, it is particularly well-suited for mobile, battery-powered devices, like tablets, or smartphones. The algorithm is formulated on the basis of a modified extremum seeking approach. By incorporating broader information about the channel dynamic characteristics, the energy efficiency as compared with the default, recommended solution, is improved. Numerous experiments (conducted in the open, public Internet, as opposed to the commonly-applied simulated environments) show several percent reduction in the energy expenditure. In many of the analyzed cases, higher throughput is also attained. As a result, the energy economy of the multipath transfer in a dynamic networking environment can be enhanced.

References

1. Ignaciuk, P., Bartoszewicz, A.: Congestion Control in Data Transmission Networks: Sliding Mode and Other Designs. Springer, London (2013). https://doi.org/10.1007/978-1-4471-4147-1
2. Ignaciuk, P., Karbowańczyk, M.: Active queue management with discrete sliding modes in TCP networks. Bull. Polish Acad. Sci. Tech. Sci. **62**(4), 701–711 (2014)
3. Peng, Q., Walid, A., Hwang, J., Low, S.: Multipath TCP: analysis, design, and implementation. IEEE/ACM Trans. Netw. **24**(1) 596–609 (2016)
4. Xu, C., Zhao, J., Muntean, G.: Congestion control design for multipath transport protocols: a survey. IEEE Commun. Surv. Tutor. **18**(4), 2948–2969 (2016)
5. Ford, A., Raiciu, C., Handley, M., Bonaventure, O.: TCP extensions for multipath operation with multiple addresses. Technical report 6824, RFC (2013)
6. Barré, S., Paasch, C.: Multipath TCP Linux Kernel implementation. http://www.multipath-tcp.org
7. Kwak, J., Choi, O., Chong, S., Mohapatra, P.: Processor-network speed scaling for energy-delay tradeoff in smartphone applications. IEEE Trans. Netw. **24**(3), 1647–1660 (2016)
8. Deng, S., Netravali, R., Sivaraman, A., Balakrishnan, H.: WiFi, LTE, or both? measuring multi-homed wireless internet performance. In: Proceedings on ACM IMC, Vancouver, Canada, pp. 181–194 (2014)
9. Paasch, C., Ferlin, S., Alay, O., Bonaventure, O.: Experimental evaluation of multipath TCP schedulers. In: Proceedings on ACM SIGCOMM CSWS, Chicago, USA, pp. 27–32 (2014)
10. Hwang, J., Yoo, J.: Packet scheduling for multipath TCP. In: Proceedings on 7th International Conference on Ubiquitous and Future Networks, Sapporo, Japan, pp. 177–179 (2015)
11. Peng, Q., Walid, A., Chen, M., Low, S.: Energy efficient multipath tCP for mobile devices. In: Proceedings on 15th ACM International Symposium on Mobile Ad Hoc Networking and Computing, Philadelphia, Pennsylvania, USA, pp. 257–266 (2014)
12. Arzani, B., Gurney, A., Cheng, S., Guerin, R., Loo, B.: Impact of path characteristics and scheduling policies on MPTCP performance. In: Proceedings on 28th International Conference on Advanced Information Networking and Applications Workshops (AINA), Victoria, BC, Canada, pp. 743–748 (2014)
13. Morawski, M., Ignaciuk, P.: On implementation of energy-aware MPTCP scheduler. In: Borzemski, L., Świątek, J., Wilimowska, Z. (eds.) ISAT 2017. AISC, vol. 655, pp. 242–251. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-67220-5_22
14. Ariyur, K., Krsti, M.: Real-Time Optimization by Extremum-Seeking Control. Wiley, Hoboken (2003)
15. Kou, L., Liu, J., Hu, M.: A multiple-path TCP congestion control algorithm based on sub flow correlation matrix. In: Proceedings on International Conference on Cloud Computing and Internet of Things, Changchun, China, pp. 140–144 (2014)
16. Morawski, M., Ignaciuk, P.: Reducing impact of network induced perturbations in remote control systems. Control Eng. Pract. **44**(10), 127–138 (2016)
17. Morawski, M., Ignaciuk, P.: Network nodes play a game - a routing alternative in multihop ad-hoc environments. Comput. Netw. **122**, 96–104 (2017)
18. Cardwell, N., Cheng, Y., Gunn, C., Yeganeh, S., Jacobson, V.: BBR: congestion-based congestion control. Commun. ACM **60**(2), 58 (2017)