Discrete-Time MPTCP Flow Control for Channels with Diverse Delays and Uncertain Capacity

Przemysław Ignaciuk, *Senior Member, IEEE*, and Michał Morawski
Institute of Information Technology
Lodz University of Technology
215 Wólczańska St., 90-924 Łódź, Poland
{przemyslaw.ignaciuk, michal.morawski}@p.lodz.pl

***Abstract*—The TCP protocol provides the control framework for most of the today Internet traffic. Since its development over forty years ago, however, the end-points have not benefited from multiple interfaces installed for redundancy purposes in the underlying equipment. In the case of failure, the logical connection has been broken or stalled. In order to remedy this situation, a multipath version of TCP – MPTCP – has recently been elaborated. It allows for simultaneous use of a set of transmission paths and interfaces, yet requires appropriate means of coordination. This paper presents a discrete-time model of data exchange process subject to MPTCP master control and provides a new flow control algorithm for the paths with diverse parameters, e.g., different delays. The algorithm establishes a feasible (non-negative and bounded) input signal and finite data queue length despite uncertain networking conditions. The properties of the designed control system are analyzed formally and illustrated by numerical tests.**

***Keywords*—telecommunication networks; discrete-time control; time-delay systems; MPTCP.**

I. INTRODUCTION

Undoubtedly, the current Internet communication cannot exist without the TCP protocol transporting about 95% of all the exchanged data. The TCP versions used today differ significantly from the initial ones. The protocol evolves, being continuously adjusted to the encountered challenges. The majority of TCP modifications have focused on improving its efficiency, leaving the fundamental disadvantage intact – the protocol suffered from the single point of failure drawback. Contrary to the networking equipment, designed with the risk of failure in mind, the end-point devices (servers, laptops, smartphones) use only one interface even if they have a few physically installed (e.g., Ethernet, WiFi, LTE). A couple of years ago, this restriction has been relaxed when the multipath version of TCP – MPTCP [1]–[4] – has been developed. This new protocol is transparent both for the communication equipment and applications. Therefore, it is likely to substitute the legacy TCP, soon. Concurrently, a new research domain has emerged: how to shape the multipath data transfer in order to answer the needs of modern communication services? This paper presents a dynamic data flow controller for the MPTCP protocol so that efficient and balanced network performance in the face of uncertainty is obtained.

The MPTCP functional framework, organized into a number of modules, is illustrated in Fig. 1. In response to the request from the user application for opening a network connection the path manger attempts to establish a set of channels (paths). The path supervision is performed during the connection life-time. When the application fills the buffer with data, the master controller acts to deliver them according to the selected policy, e.g., as fast as possible, considering fairness objectives, conserving the buffer space, etc. It passes the data into the scheduler module, which distributes the stream among the currently available channels. Technically, the data is placed into the Singlepath TCP (SPTCP) buffers and then transmitted according to the SPTCP rules. Since the MPTCP master controller does not need to replicate the SPTCP tasks, it can be designed in a different, more flexible way. For instance, one can address the general aspects of handling the application stream, i.e., to increase throughput, reduce buffer occupancy, or shorten latency [5][6].

The paper addresses the design of data flow algorithm for the master controller so that high efficiency, understood as expedient data delivery, is obtained. In the literature, and in the reference protocol stack, one can find proposals of various master controller algorithms, e.g., LIA [2], OLIA [7], BALIA

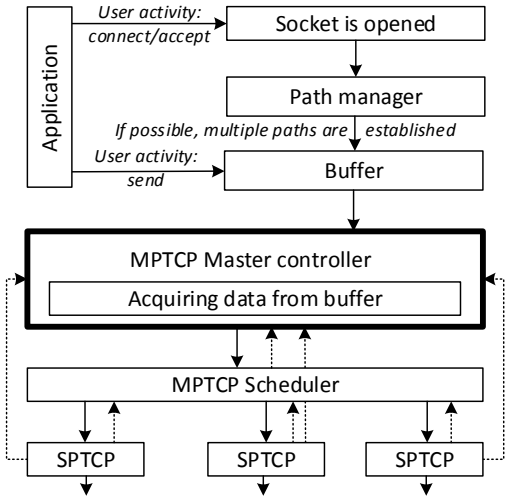

Fig 1. MPTCP architecture. Solid arrows represent the flow of user data, dashed ones – internal information exchange.

[8], wVegas [9]. While building on solid theoretical grounds, these algorithms prioritize fairness over efficiency. Moreover, lacking the formal, analytical support of control theory, their dynamical properties are not sufficiently explored.

In this work, a model of MPTCP data transfer, encompassing the effects of delays and finite sampling rate, is constructed. Unlike the typical approach to consider the perturbations only at the output (concerning the *a priori* unknown channel capacity variations [10]), here, also the incoming flow variations (originating from the application) are explicitly taken into account in the design procedure and algorithm evaluation. In order to achieve high efficiency without compromising stability, a delay compensation mechanism is incorporated. By balancing the input-output flow discrepancy, this mechanism allows one to reduce the buffer space expectations while keeping a consistent input signal. The properties of the designed control system are formulated and proved analytically. In particular, it is shown that the established flow control signal is always non-negative and upper-bounded, as required by the analyzed class of applications. It is also demonstrated that the queue length of the TCP segments waiting for acknowledgment in the MPTCP buffer is finite. The buffer volume is explicitly indicated. Finally, numerical simulations validate the design correctness.

II. PROBLEM STATEMENT

As depicted in Fig. 1, the user application seeks to send data to a remote peer. It places them in the MPTCP buffer. The master controller module, which is a standard part of the MPTCP stack, acquires the data from the buffer and transfers them to the remote peer using n connections. Each connection j is assigned a weight p_j, $0 \le p_j \le 1$, reflecting the adopted policy of stream splitting. The weights can be selected arbitrarily, e.g., according to the link monetary cost, reliability, power consumption [11], [12], or other factors [13], [14], as long as the condition $\sum_{j=1}^{n} p_j = 1$ is satisfied.

The objective is to transfer the data (drain the buffer) as soon as possible with the intention to maximize throughput. The obstacle to overcome are the *a priori* unknown variations of the link capacity (output) and incoming data rate generated by the application (input). From the point of view of control theory the buffer is the controlled plant, subject to the perturbation related to the input/output flow imbalance. It is assumed that the data units – TCP segments in the analyzed case – are individually relieved from the buffer as soon as the underlying SPTCP acknowledges reception by the peer, not necessarily in the FIFO order. Nevertheless, since the segments and acknowledgments traverse the network experiencing delay, the perturbation is mismatched. In order to address the robustness issues, without throttling responsiveness, a discrete-time delay compensator is incorporated.

III. SYSTEM MODEL

The system is sampled at a constant rate, at times 0, T_s,…, kT_s,…, $k \in N$. For notational brevity, T_s will be omitted from the time reference.

Once Master controller has established the control signal, the MPTCP stream is split into the SPTCP substreams as

$$u_j(k) = p_j u(k), \tag{1}$$

where $u_j(k)$ is the flow rate in channel j and p_j's are the split coefficients such that $\sum_{j=1}^{n} p_j = 1$.

Let RTT_j define the round-trip time in channel j. RTT_j comprises an integer multiple of T_s, $RTT_j = m_j T_s$, $m_j \in N$, that pass since an attempt has been made to transfer a TCP segment up to the feedback information (acknowledgement) reception from the peer. During this time, the data emitted by the sender occupy the MPTCP buffer along with the data yet to be transmitted. In the networking terminology, the unacknowledged data are commonly referred to as the *in-flight* data. The amount of *in-flight* data in channel j can be calculated by summing all the data emitted so far by the sender – u_j – minus those whose reception has been acknowledged by the remote peer – u_j^a. Mathematically,

$$f_j(k) = \sum_{i=0}^{k-1} u_j(i) - \sum_{i=0}^{k-1} u_j^a(i). \tag{2}$$

Taking into account the uncertain, time-varying networking conditions, function $f_j(k)$, representing the *in-flight* data in channel j, can be represented in the alternative form

$$f_j(k) = \sum_{i=1}^{m_j} u_j(k-i) + \delta_j(k), \tag{3}$$

Function f_j comprises the term related to the nominal operating conditions $\sum_{i=1}^{m_j} u(k-i)$ and (output) perturbation $\delta_j(k)$. Considering all the subchannels, the overall amount of *in-flight* data can be expressed as $f(k) = \sum_{j=1}^{n} f_j(k)$, and the total perturbation at the sender output as $\delta(k) = \sum_{j=1}^{n} \delta_j(k)$.

On the other hand, at any step k, the user application can supplement the buffer by additional $\beta(k)$ data. From the point of view of the MPTCP controller, since $\beta(k)$ is not known *a priori*, it is treated as an input disturbance.

At time k, the controller checks how many data units have been acknowledged and removes them from the buffer. The free space can be used to accept the new data from the application. Because the copy of *in-flight* data have to be held in the buffer until they are acknowledged to avoid permanent loss, the following condition needs to be satisfied:

$$\beta(k-1) + f(k) \le B, \tag{4}$$

where B is the buffer size. Taking into account the preceding considerations, the buffer occupancy for any $k \geq 0$ can be expressed through

$$y(k+1) = y(k) + \beta(k) - \sum_{j=1}^{n} u_j^a(k). \tag{5}$$

It is assumed that no transmission is performed until $k = 0$, i.e., for any $k < 0$, $u(k) = 0$. Before the transmission commences, the data coming from the application accumulate in the MPTCP buffer. Thus, $\sum_{i=-\infty}^{-1} \beta(i) = y(0)$ and $\beta(0) = 0$. Since no data is emitted for $k < 0$, no acknowledgements are expected, and one has $f(k) = 0$ for $k \leq 0$.

IV. MPTCP CONTROLLER

In order to provide fast, yet stable performance despite uncertain networking conditions, the following control law is proposed:

$$u(k) = y(k) - \sum_{i=0}^{k-1} u(i) + \sum_{i=0}^{k-1} u^a(i). \tag{6}$$

According to (6), the overall MPTCP transfer rate reflects the current buffer occupancy and the history of previous transmission attempts. The properties of thus constructed control system will be formulated as theorems and strictly proved.

Lemma 1. If control law (6) is applied to model (1)–(5), then for any $k > 0$ the control input matches the input perturbation from the previous time instant, i.e.,

$$\forall_{k>0}\ u(k) = \beta(k-1). \tag{7}$$

Proof: It follows from the assumed initial conditions that $u(0) = y(0)$. Applying (5) to (6), one obtains

$$\begin{aligned} u(k) &= y(k-1) + \beta(k-1) \\ &\quad - \sum_{j=1}^{n} u_j^a(k-1) - \sum_{i=0}^{k-1} u(i) + \sum_{i=0}^{k-1} u^a(i). \end{aligned} \tag{8}$$

Taking into account the fact that $\sum_{j=1}^{n} u_j(k) = u(k) \sum_{j=1}^{n} p_j = u(k)$ and $\sum_{j=1}^{n} u_j^a(k) = u^a(k)$, (8) may be rewritten as

$$u(k) = y(k-1) + \beta(k-1) - \sum_{i=0}^{k-1} u(i) + \sum_{i=0}^{k-2} u^a(i). \tag{9}$$

After reordering, (9) simplifies to

$$\begin{aligned} u(k) &= \overbrace{y(k-1) - \sum_{i=0}^{k-2} u(i) + \sum_{i=0}^{k-2} u^a(i)}^{u(k-1)} - u(k-1) + \beta(k-1) \\ &= \beta(k-1). \end{aligned} \tag{10}$$

□

Theorem 2. If control strategy (6) is applied to model (1)–(5), then for any $k \geq 0$, the control signal is non-negative and upper-bounded, i.e.,

$$\forall_{k\geq 0}\ 0 \leq u(k) \leq B. \tag{11}$$

Proof: As $u(0) = y(0)$, the theorem holds at the initial time $k = 0$. Since for any k, $f(k) \geq 0$, one has from (4): $\beta(k-1) \leq B$. Thus, by invoking Lemma 1, one obtains for $k > 0$: $u(k) = \beta(k-1) \leq B$. On the other hand, since the amount of data coming from the application $\beta(k)$ is non-negative, one has $u(k) \geq 0$. □

Theorem 2 shows that the proposed control law is indeed feasible to implement in telecommunication networks. Moreover, as demonstrated in Lemma 1, the control signal depends on the output disturbance, but not on the RTT_j's and p_j's. Therefore, the input established by the controller will not be affected by the changes of those parameters, which further increases robustness of the discussed control system.

Theorem 3. If control strategy (6) is applied to model (1)–(5), then for any $k \geq 0$ the buffer occupancy $y(k)$ fulfills the inequalities:

$$0 \leq y(k) \leq B. \tag{12}$$

Proof: The assumed initial conditions indicate that $u(0) = y(0)$. Therefore, for $k = 0$, $y(k)$ is within bounds (12). For $k > 0$, one has from (5):

$$\begin{aligned} y(1) &= y(0) + \beta(0) - u^a(0), \\ y(2) &= y(1) + \beta(1) - u^a(1) = y(0) + \sum_{i=0}^{1} \beta(i) - \sum_{i=0}^{1} u^a(i), \\ &\vdots \\ y(k) &= y(0) + \sum_{i=0}^{k-1} \beta(i) - \sum_{i=0}^{k-1} u^a(i). \end{aligned} \tag{13}$$

Using (2) in the last equation in (13), one obtains

$$y(k) = y(0) + \sum_{i=0}^{k-1} \beta(i) + f(k) - \sum_{i=0}^{k-1} u(i). \tag{14}$$

Then, taking into account the relations $u(0) = y(0)$ and $\beta(0) = 0$, (14) can be simplified as

$$y(k) = y(0) + \beta(0) + \sum_{i=1}^{k-1} \beta(i) + f(k) - u(0) - \sum_{i=1}^{k-1} u(i) = \sum_{i=1}^{k-1} \beta(i) + f(k) - \sum_{i=1}^{k-1} u(i). \quad (15)$$

Applying (7) to (15) leads to

$$\begin{aligned} y(k) &= \sum_{i=1}^{k-1} \beta(i) + f(k) - \sum_{i=1}^{k-1} \beta(i-1) \\ &= \sum_{i=1}^{k-1} \beta(i) + f(k) - \sum_{i=0}^{k-2} \beta(i) \\ &= \beta(k-1) + \sum_{i=1}^{k-2} \beta(i) - \sum_{i=1}^{k-2} \beta(i) + f(k) - \overbrace{\beta(0)}^{=0} \\ &= \beta(k-1) + f(k). \end{aligned} \quad (16)$$

Since for any $k > 0, f(k) \geq 0$, then it follows from (16) that

$$y(k) \geq \beta(k-1) \geq 0. \quad (17)$$

On the other hand, taking into account (4), one obtains from (16):

$$y(k) \leq B. \quad (18)$$

□

Relations (16)–(18) have a practical implication for the actual use of controller (6) in the networks modelled according to (1)–(5). First of all, the control system operates correctly for any buffer size B, irrespective of the delay range, or the applied method of MPTCP stream splitting. It follows from (16) that the amount of data accepted from the application (and permitted into the network) relates directly to *in-flight* data. On the one hand, a larger buffer size allows one to inject more data into the network, giving perspective for faster transmission accomplishment when looking at the MPTCP controller as a transport service provider for the application. On the other hand, allowing the controller to send a large amount of data increases the risk of congestion (thus losses) and wasted resources due to retransmissions. As is typical in the network protocol implementations, it is recommended to set the buffer size commensurate with the nominal delay. Taking into account the flow separation, $B = \sum_{j=1}^{n} RTT_j$ is advised.

V. Numerical Example

Consider an MPTCP flow directed through three SPTCP channels, as depicted in Fig. 1. The acknowledgements in the SPTCP subflows are delivered with RTT_j equal to 3, 5, and 7 sampling periods, respectively. The nominal split ratio p_i for these subflows is set as 0.3, 0.3, and 0.4. In addition to the nominal case (a), the robustness of controller (6) to uncertain, time-varying networking conditions is evaluated. Two such cases are examined: (b) the nominal *RTT*s randomly elongated by $0–4T_s$ and (c) the split ratio exhibiting additional, random fluctuations (sketched in Fig. 7). The transfer rate in any of the SPTCP subflows does not exceed its *cwnd* (congestion window) and follows the rules of TCP Reno. The maximum attainable throughput of each flow is presented in Fig. 2. The buffer is initially filled by $y(0) = 1000$ data units.

Fig. 3 illustrates the buffer depletion according to the established control signal shown in Fig. 4, and Fig. 5 depicts the related acknowledgements. The flow imbalance $\delta(k)$ is shown in Fig. 6. It follows from the $y(k)$ curve analysis that the buffer is indeed efficiently drained despite delays and *a priori* unknown flow imbalance, both in the nominal and perturbed models. While one can observe (small) differences between $y(k)$ and $u^a(k)$ in the investigated scenarios (a)–(c), the system is robust. From the perspective of preserving the flow consistency, a significant advantage of the proposed algorithm is its insensitivity to delay and split ratio variations. The generated control signal (Fig. 4) is the same in all three cases (a)–(c).

The network transmission with the MPTCP protocol involved does not imply the number of paths is kept constant during the data exchange. In fact, the path manager continuously tries to create new channels and respond to the errors in active ones. Figs. 8–11 illustrate the algorithm behavior in the situation where a single initial path is supplemented by two additional ones at $10T_s$ and $20T_s$, respectively, and afterwards, the initial path is closed at $30T_s$. All the *in-flight* data from the first stream are lost. In this scenario, $p_i = 1/n$, with n time-varying. Despite *a priori* unknown number of channels, $u(k)$ determined according to (6) is identical to the one depicted in Fig. 4.

VI. Conclusion

The paper presents a discrete-time model of MPTCP communication system and an algorithm governing the operation of the MPTCP master flow controller. The model encompasses the effects of loop delay and finite sampling rate related to key networking events. Unlike the typical solutions proposed so far in the field, the proposed algorithm does not require manual parameter adjustment. It can be flexibly incorporated into the MPTCP architecture with no need to modify the other components, e.g., the scheduler, or the path manager. It works in the same way irrespective of the number of established paths, their quality, and parameter variations. The properties of the obtained control system are formally proved and illustrated by simulations. It is shown that the controller always generates a feasible, i.e., non-negative and bounded input, and it efficiently drains the allocated buffer. The resulting system is demonstrated robust with respect to modelling inaccuracies and channel perturbations.

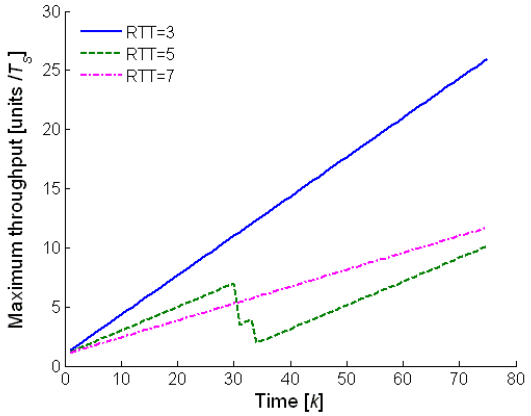

Fig. 2. Congestion window *cwnd* (maximum attainable throughput) of SPTCP subflows.

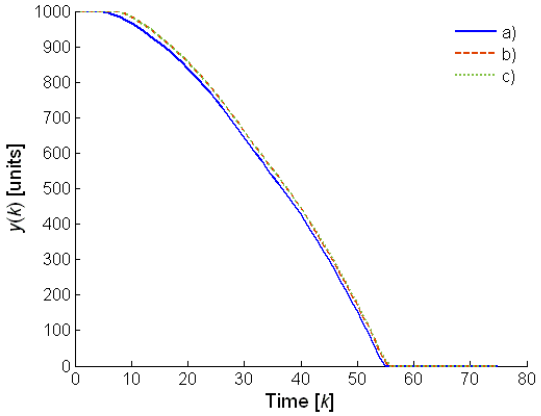

Fig. 3. Buffer occupancy: a) nominal case, b) random delays, c) random delays and random split ratio.

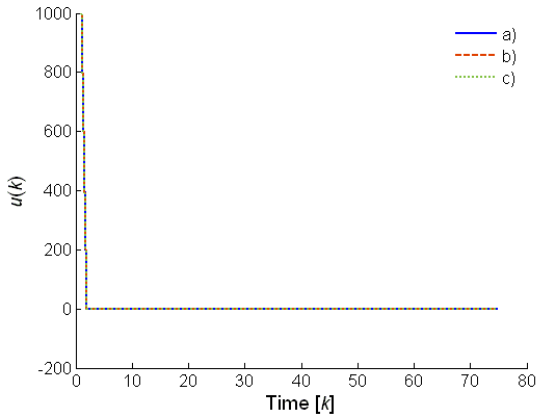

Fig. 4. Transfer rate established according to (6): a) nominal case, b) random delays, c) random delays and random split ratio.

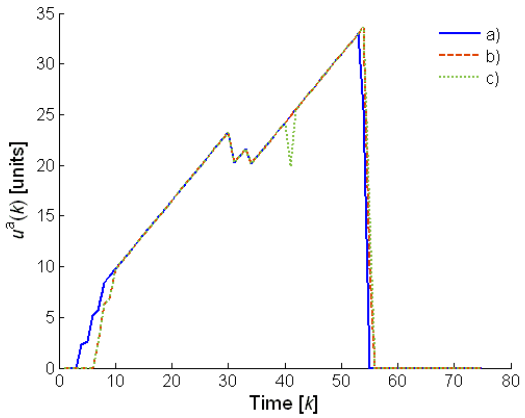

Fig. 5. Number of acknowledged data units: a) nominal case, b) random delays, c) random delays and random split ratio.

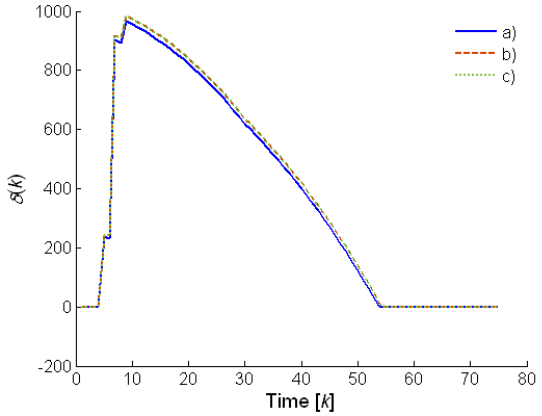

Fig. 6. Disturbance determined according to (3): a) nominal case, b) random delays, c) random delays and random split ratio.

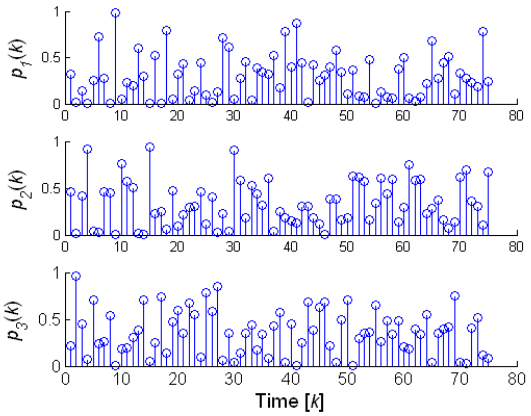

Fig. 7. Random split ratio in test c).

REFERENCES

[1] C. Xu, J. Zhao, and G. Muntean, "Congestion control design for multipath transport protocols: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2948–2969, 2016.

[2] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "TCP extensions for multipath operation with multiple addresses," RFC 6824, 2013.

[3] S. Barré, C. Paasch, and O. Bonaventure, *MultiPath TCP: From Theory to Practice*, Université Catholique de Louvain, 2011.

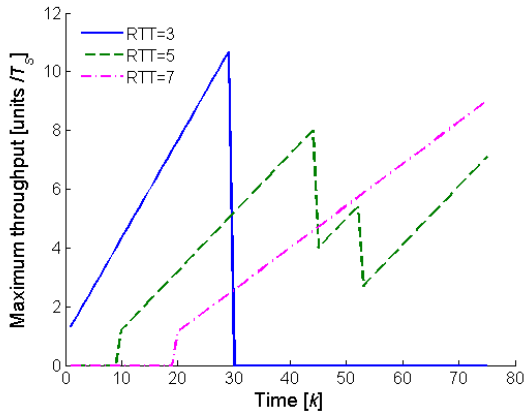

Fig. 8. Congestion window *cwnd* (maximum attainable throughput) of SPTCP subflows in the variable number of streams scenario.

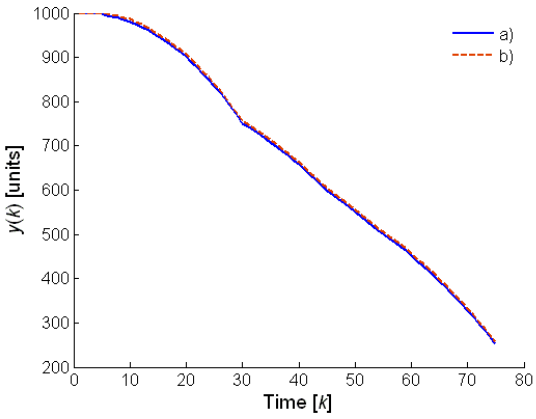

Fig. 9. Buffer occupancy: a) nominal case, b) random delays in the variable number of streams scenario.

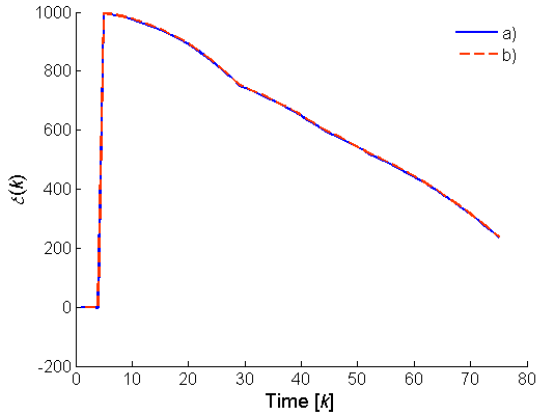

Fig. 10. Disturbance determined according to (3): a) nominal case, b) random delays in the variable number of streams scenario.

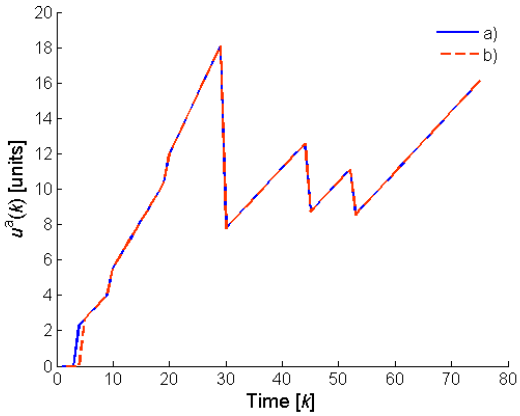

Fig. 11. Number of acknowledged data units: a) nominal case, b) the random delays in variable number of streams scenario.

[4] Q. Peng, A. Walid, J. Hwang, and S. H. Low, "Multipath TCP: Analysis, Design, and Implementation," *IEEE/ACM Transactions on Networking*, vol. 24, no. 1, pp. 596–609, 2016.

[5] M. Morawski and P. Ignaciuk, "Reducing impact of network induced perturbations in remote control systems," *Control Engineering Practice*, vol 55, no. 10, pp. 127–138, 2016.

[6] M. Morawski and P. Ignaciuk, "Network nodes play a game – a routing alternative in multihop ad-hoc environments," *Computer Networks*, vol. 122, pp. 96–104, 2017.

[7] R. Khalili, N. Gast, M. Popovic, and J.-Y. Le Boudec, "MPTCP is not Pareto-optimal: Performance issues and a possible solution," *IEEE/ACM Transactions on Networking*, vol. 21, no. 5, pp. 1651–1665, 2013.

[8] Q. Peng, A. Walid, J. Hwang, and S. H. Low, "Multipath TCP: Analysis, design, and implementation," *IEEE/ACM Transactions on Networking*, vol. 24, no 1, pp. 596–609, 2016.

[9] Y. Cao, M. Xu, and X. Fu, "Delay-based congestion control for multipath TCP," In *Proc. 20th IEEE Int. Conf. on Network Protocols (ICNP)*, Austin, TX, USA, pp. 1–10, 2012.

[10] P. Ignaciuk and A. Bartoszewicz, *Congestion Control in Data Transmission Networks. Sliding Mode and Other Designs*, LNCS, Springer, 2013.

[11] M. Morawski and P. Ignaciuk, "On implementation of energy-aware MPTCP scheduler," In: *Proc. 38th Int. Conf. on Information Systems Architecture and Technology* (*ISAT*), pp. 242–251, Karpacz, Poland, 2017.

[12] M. Morawski and P. Ignaciuk, "Energy efficient MPTCP transmission – Scheduler implementation and evaluation," In *Proc. 21th International Conference on System Theory, Control and Computing – ICSTCC'2017*, pp. 654–659, Sinaia, Romania, 2017

[13] C. Paasch, and S. Ferlin, O. Alay, and O. Bonaventure, "Experimental evaluation of multipath TCP schedulers," In: *Proc. ACM SIGCOMM CSWS*, Chicago (IL), USA, pp. 27–32, 2014.

[14] B. Arzani, A. Gurney, S. Cheng, R. Guerin, and B. T. Loo, "Impact of path characteristics and scheduling policies on MPTCP performance," in *Proc. IEEE AINA Workshop*, pp. 743–748, 2014.