ISSN 1330-3651 (Print), ISSN 1848-6339 (Online) https://doi.org/10.17559/TV-20220814213135
Preliminary communication

Possible Improvements of TCP Protocol with the Use of Heuristic Methods

Afan CECO*, Sasa MRDOVIC

Abstract: This paper describes the possibility of applying heuristic methods for parameter optimization in the TCP protocol. The proposed concept provides for a TCP protocol adjusting its parameters for greater efficiency through testing the network state and adapting accordingly. This can be achieved through careful analysis of the network state both before and during data transfer connections. The proposed solution introduces an innovative approach, incorporating the possibility of self-learning and self-adjusting capabilities. This sophisticated algorithm should define the next parameter values in terms of finding optimal parameter settings. Each TCP connection plays a crucial role as iteration in the process of finding the optimal solution. The concept focuses on calculating TCP parameter values at the network ends in order to optimize network traffic and to maximize the use of network resources. The approach has been tested on a dedicated test platform, validating its potential for verifying the network protocols functionality and for optimizing their parameters. The proposed solution, here referred to as the modified TCP, showed better performance compared to other versions of the TCP protocol. Notably, even under heavy traffic loads on links, the results for the modified TCP consistently outperform the standard TCP, delivering results that are several times better.

Keywords: data; heuristic; network; optimization; protocol; TCP

1 INTRODUCTION

The Transmission Control Protocol/Internet Protocol (TCP/IP) serves as a standard suite of protocols that underpins the functioning of the Internet. Transmission Control Protocol (TCP), in addition to Internet Protocol (IP), is one of two major protocols in the TCP/IP protocol suite. It facilitates reliable, connection-oriented data transfer. Before data transmission, two interacting hosts engage in a synchronization process to establish a virtual connection, enabling them to prepare for data transmission and determine initial sequence numbers. This process, known as the three-way handshake, consisting of three steps, enables a virtual connection between the two devices.

The original TCP transmits multiple segments into the network, limited to the receiver's advertised window size. Although the above-mentioned claim has been acceptable when the two hosts are on the same LAN (Local Area Network), issues may arise when routers and slower links between transmitter and receiver are involved. Intermediary routers may create queues with packets, leading to the router's potential memory space constraints for temporary placement of the packets.

The Slow Start algorithm has been introduced to address this issue. It monitors the rate at which new packets enter the network, ensuring it aligns with the speed of receiving acknowledgments from the other side.

Congestion may occur when data arrive from a fast LAN, but need to be sent to a slower WAN (Wide Area Network). It can also occur when a router's output capacity is insufficient (less than the sum of all inputs) to handle multiple incoming flows.

Congestion Avoidance is introduced to overcome the packets loss issue. Fast Retransmit is the improvement of the TCP that reduces time before retransmitting lost segments.

Once the Fast Retransmit sends the missing segment, the Congestion Avoidance takes place, not the Slow Start. This is called the Fast Recovery algorithm.

To achieve dynamic optimization of TCP parameters, heuristic algorithms can be used, as it is described in this paper. This paper explores possible TCP protocol improvements, such as identifying optimal values for the initial congestion window, finding the minimum value of the congestion window and determining appropriate values for increasing and decreasing of the congestion window.

The first part of this paper provides an overview of the current situation in the subject area. The second part of the paper introduces the proposed solution of TCP parameter optimization using the evolutionary strategy, as part of heuristic methods. The last part of this paper presents the simulation results obtained by parameter optimization.

2 RELATED WORK

2.1 Loss-Based, Delay-Based and Combined/Hybrid Algorithms

Traditionally, TCP algorithms relied on detecting congestion through packet losses, implementing the AIMD (Additive Increase/Multiplicative Decrease) principle to adjust data transmission rate by increasing or decreasing the transmission window. Noteworthy algorithms in this category include Tahoe TCP [1], Reno TCP [2], New Reno TCP [3], SACK TCP [4], and others.

Furthermore, TCP algorithms that use delays as congestion indicators have been developed, such as for example Vegas TCP [5], Fast TCP [6], or recently introduced BBR [7].

In addition, combined TCP algorithms have emerged, leveraging both loss and delay as indicators of congestion, including Africa TCP [8], Yeah TCP [9] and Compound TCP [10]. These algorithms are classified as hybrid algorithms, acting as more aggressive TCP algorithms when no congestion is detected and transitioning to standard (Reno) TCP algorithms when congestion is detected.

2.2 Fast/High Speed Algorithms, Fairness Principle and Specific Applications

Fast speed TCP algorithms have a more aggressive window increase, determining the protocol operational speed and its link capacity utilization. Notable algorithms

include Fast TCP [6], HS TCP [11], Scalable TCP [12], BIC [13], CUBIC [14] and others.

The fairness principle is demonstrated in algorithms such as Africa and Yeah TCP, referring to the ability of fast speed TCP to adapt to slower TCP streams, such as Reno TCP. When such flows occur on the link or the congestion is detected, they act as if they were Reno TCP. Additionally, in [15] the author proposes its own Coexistent TCP, further elaborating on the fairness principle.

There are also versions of TCP protocols designed for specific applications. For data centre applications, there is DCTCP (Data Centre TCP) [16], or optimized for wireless media, WTCP (Wireless TCP) has been developed [17].

2.3 Increasing or Abolition of the Initial Congestion Window

In [18], there is a proposition to increase the initial congestion window to the size of 10 maximum segments. In [19], the author proposes an automatic increase of the initial congestion window based on measuring the number of connections with errors (packet losses). This adaptive approach allows for network feedback over long timescales, avoiding values that consistently lead to network problems.

Another approach proposed in [20] is to repeal a predetermined fixed initial congestion window. Instead, the application can select the size of the initial congestion window, with specific restrictions: the three-way handshake must complete without a retransmission, initial windows of more than 10 segments must be evenly paced across the first round-trip time as measured during the three-way handshake, and the initial congestion window (cwnd) must be bounded by the receiver's advertised window.

2.4 Changing Values of Increasing and Decreasing the Congestion Window

There are many different ways to vary the window size. Chiu and Jain [21] demonstrated that among linear methods, additive increase/multiplicative decrease (AIMD) converges to high utilization and a fair allocation of throughput, under some simplifying assumptions (long-running connections with synchronized and instantaneous feedback).

2.5 TCP Parameters Optimization

In [22], the authors conduct a comprehensive evaluation of the impact of greedy end-point behaviour in TCP using a game-theoretic analysis. They introduce the concept of a "TCP Game" where each flow seeks to maximize its throughput by modifying its congestion control behaviour. Through a combination of analysis and simulation, they determine the Nash Equilibrium of this game.

NUM methods (Network Utility Maximization) are powerful techniques for optimizing network traffic (or the utilization of network infrastructure), treating the complete layered stack with all protocols as one unique optimization problem. NUM methods are further explored in [23, 24].

In [25] the authors present a completely offline optimization method called Remy TCP. This approach enables the generation of a comprehensive congestion control algorithm within a short time. Once such an algorithm is generated, it is distributed to end-users, replacing their existing algorithms. Subsequently, the optimized algorithm is used without the possibility of additional learning. Every time a new connection is initiated, the optimized algorithm starts from the beginning, mimicking the behaviour of standard TCP and its slow start process. Unlike some other methods, the algorithm in this solution does not cache the congestion state or transfer knowledge between connections.

In contrast, [26] addresses the issues of dynamic adjustment of Initial Window (IW) and Congestion Control (CC) schemes based on knowledge acquired from previous connections. For short streams, it adjusts the IW, while for long streams, it intelligently selects the most suitable CC scheme based on the network's current conditions.

Some other TCP optimization proposals are for specific use only, for example mobile (5G) [27] or IoT networks [28].

2.6 Relation to Proposed Solution

The solution proposed in this article incorporates both loss-based and delay-based principle, with the implementation of the delay-based principle reserved for future research. Additionally, the proposed solution embraces a dual principle, utilizing more aggressive algorithms when maximum performance is required, while still maintaining the ability to switch to the classic Reno TCP to ensure fairness towards other TCP flows.

One of the key strengths of the proposed solution lies in its versatility, as it is designed to be adaptable for various purposes. While the solution shares certain similarities with NUM methods, it differs by focusing its optimization efforts on a single layer of the network stack. The fundamental difference between the solution presented in this article and previously introduced methods such as Remy is the fact that they are completely offline optimization methods, while this solution represents an online optimization method.

A different approach to network congestion issue is presented in [29], with proposed solution in content caching on the network edge, by predicting content popularity.

3 POSSIBLE IMPROVEMENTS OF TCP PROTOCOL USING HEURISTIC METHODS FOR PARAMETER OPTIMIZATION

In this section, we propose an innovative approach to improve the functionality of the TCP protocol. Our proposal introduces two significant differences from the concepts presented in the previous section. Firstly, in our approach, the parameters used to calculate changes to the congestion window are not fixed but are continuously updated. This dynamic and adjustable function enables the flexible setting of the congestion window, making better use of available bandwidth and adapting more effectively to changing network conditions. Secondly, these parameter values are optimized using heuristic methods. Through

iterative adjustments, we search for parameter values that provide the highest throughput under specific network conditions. The dynamic and constantly changing nature of the network state means that here are no fixed, stationary optimal values of the parameters. Therefore, heuristic methods are well-suited for optimizing in such dynamic conditions.

The proposed approach leverages mathematical methods from the field of Operational Research and Artificial Intelligence to demonstrate the optimization of TCP connections. This method of improving the TCP protocol performance focuses on the mathematical optimization of connection parameters, in accordance with the principle of heuristic algorithms. During the initiation of each subsequent connection, the parameters values are modified, relative to the previous values, while seeking an optimal solution based on heuristic strategy. The underlying idea behind this proposal is to harness the frequent initiation of new TCP connections by subsequently probing the network condition, and to continuously optimize TCP connection parameters for the best network performance.

The proposed method also includes the concept of learning. During the learning phase, the algorithm defines the next parameter values to find the optimal solution. Therefore, it can be considered heuristic as passing through the learning phase or self-adjusting. As the algorithm transitions to the operational (working) state, it can preserve the calculated parameter values for the next connection. This repetitive process with each new connection start-up allows for long-term learning. Subsequently, this approach enables the algorithm to gain valuable insights into the network status and preserve such knowledge for future use. Fig. 1 visually illustrates the flow chart of the proposed solution.

3.1 Congestion Window Calculation

The proposed algorithm for congestion window calculation is a variation of TCP Reno. It adheres to the fundamental principle of incrementally increasing the congestion window until congestion is detected. Then, the congestion window is decreased to alleviate congestion. Initial congestion window size and formulas for its increase and decrease are somewhat different. We decided to use Additive-Increase Additive-Decrease (AIAD) approach instead of traditional Reno style setting (AIMD). The AIAD approach can be fine-tuned and is not so drastic in reducing window size when facing congestion.

The proposed approach introduces a fundamental change to the traditional TCP congestion mechanism by abandoning the slow start mechanism, which involves exponential growth of the congestion window, and instead adopting congestion avoidance, employing linear growth of the congestion window. To make the window grow at a faster rate, a larger initial window size is used compared to standard TCP protocol versions, similar to the solution suggested in [30].

As it was mentioned in the beginning of this section, the algorithm parameters are not fixed, but are determined by using an evolutionary strategy, and new values are applied for each new connection. The evolutionary strategy is based on the experience gained from previous connections. Specifically, parameter a is responsible for defining the starting and minimum values of the congestion window, while parameters b and c are used to determine the value of increasing or decreasing of the congestion window. The optimization of the parameters is further elaborated in the next subsection.

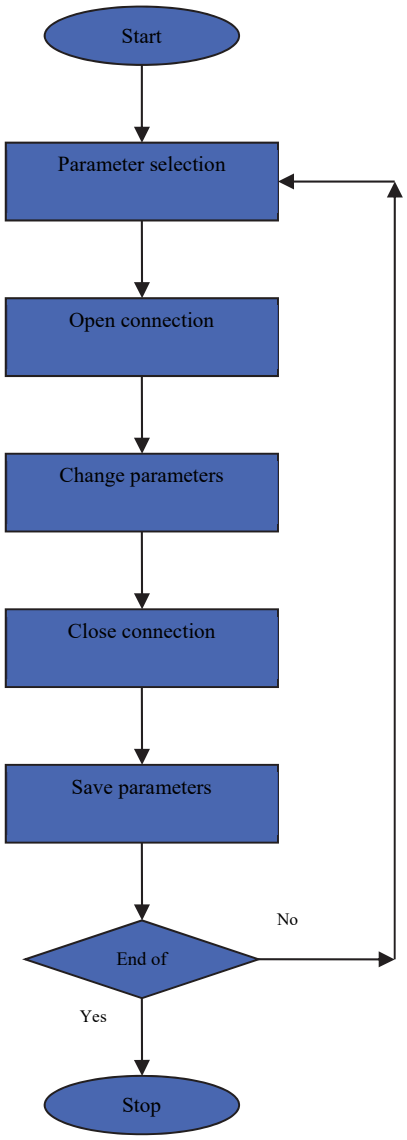


Figure 1 Flow chart of the proposed solution

The algorithm operates as follows:

First, the initial value of the congestion window is determined as: cwnd start = a × cwnd default, where cwnd default is 1 MSS (Maximum Segment Size).

Data segments are sent up to the value allowed by the size of the window.

If there are no reported congestion issues, the congestion window value is increased as follows:

cwnd = cwnd + b × MSS.

If a congestion issue is reported, the congestion window value is reduced as follows:

cwnd = cwnd − c × MSS.

At the same time, the congestion window value cannot decrease to less than cwnd min = a × cwnd default.

The above-described algorithm is shown in Fig. 2.

Certainly, when there is no reported congestion on the network path, the algorithm incrementally increases the

congestion window value in each subsequent transmission cycle. However, if congestion issues are detected, the algorithm successively decrements the congestion window value each time a congestion indication is received.

This congestion indication may occur either with the loss of a data packet or based on an increase in the round-trip time (RTT) during data transmission. In this paper, congestion indications are solely based on packet losses, while the increase in RTT as a congestion indication is left for future exploration.

The presented approach can be applied to optimize the parameters of any TCP algorithm involved in congestion window calculation. The presented algorithm, as described here, has been proposed and implemented.

3.2 Heuristic Methods

Heuristic methods are applied when seeking approximate solutions where conventional methods fail to find the exact solution. The Evolutionary Strategy (ES) used in the proposed approach belongs to a wider group of heuristic methods. ES is an optimization technique that draws inspiration from adaptation and evolution [31]. It primarily uses mutation and selection as search operators for the solutions space. The ES (1 + 1), in particular, operates with the population size of two: the current point (parent) and the result of its mutation (descendant). If the offspring's fitness is at least as good as the parent's, it becomes the parent for the next generation; otherwise, such a solution is rejected.

More generally, in this ES (1 + λ), a descendant is created through mutations and then competes with its parents. In the ES (1, λ), the best mutant becomes a parent for the next generation, while the current parent gets constantly rejected. In the solution proposed in this paper, the ES (1 + 1) is applied due to its population size of two, enabling efficient evaluation between the old and new solution.

In this approach, each TCP connection represents one iteration in the process of discovering the optimal parameter settings. The parameters a, b and c (a: starting and minimum values of the congestion window; b and c: the increasing and decreasing value of the congestion window) are modified by the mutation operator *lambda* between each connection, applying the following formulas: $a2 = (a1 + lambda)$; $b2 = (b1 + lambda)$; $c2 = (c1 + lambda)$.

If the result (R) of a new connection demonstrates improved network performance, measured either by the time interval for transmitting a predetermined amount of data or by the round-trip time (RTT), then the mutation process continues for the next connection, following the same direction of changing parameters. Specifically, parameters $a1$, $b1$ and $c1$ are updated with new values $a2$, $b2$ and $c2$, respectively, applying the same formulas for calculating new values of parameters a, b and c as follows: $a2 = (a1 + lambda)$; $b2 = (b1 + lambda)$; $c2 = (c1 + lambda)$. In this paper, R represents the time interval for transmitting a predetermined amount of data, while RTT measurement is intended for future research. The selected amount of data can be customized based on the application requirements, allowing for flexibility to use any desired data size, such as 500 MB, 1 GB or 10 GB. For this particular study, we focus on using 10 GB of data to demonstrate the optimization of large file transfers.

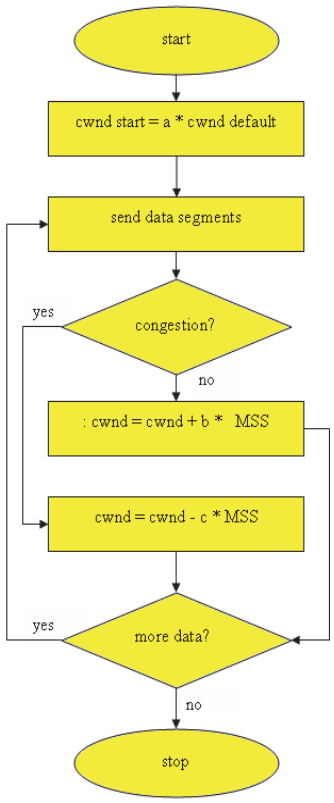


Figure 2 The algorithm of the proposed solution

Otherwise, the resulting parameters are discarded and a new mutation process for the new connection initiated. This involves exploring a different direction of the solution space. Specifically, parameters $a1$, $b1$ and $c1$ are not updated with parameters $a2$, $b2$ and $c2$, and the formulas for calculating new values are reapplied using starting values $a1$, $b1$ and $c1$; $a2 = (a1 + lambda)$; $b2 = (b1 + lambda)$; $c2 = (c1 + lambda)$. For this iteration, a different *lambda* value is applied, signifying the exploration of a new direction in the search for solution space. In this particular case, *lambda* values are determined as follows: $t + 4$ with 0.1% probability, +3 with 2.1% probability, +2 with 13.6% probability, +1 or 0 with 34.1% probability. These values are the outcome of a single standard deviation in the process of generating random numbers, ensuring that only positive values or zero are considered. It is crucial to note that these values may vary for different parameters, aligning with the principles of the heuristic algorithm, where each parameter undergoes separate adjustments.

As the system runs for a long period, the parameter changes (both increase and decrease) should remain relatively small, indicating that a stable state has been achieved, and the individual changes have proven to be effective. However, in the event of significant changes in circumstances, the strategy swiftly adapts by either increasing/decreasing the parameters to adequately compensate for the change.

Fig. 3 illustrates the flow chart of the heuristic algorithm. Furthermore, the next section describes the test platform used for validation of the proposed solution.

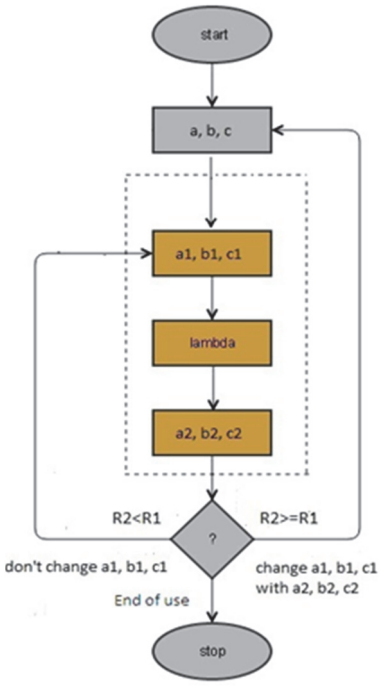


Figure 3 Flow chart of the heuristic algorithm

4 SIMULATION RESULTS

In this section, we present the simulation results performed on the test platform described in [32], obtained by applying the proposed solution for TCP parameter optimization.

The parameters optimized include the initial value of the congestion window, the minimum value of the congestion window, the value of the congestion window increase, and the value of the congestion window decrease.

We begin by presenting the simulation results for a test topology consisting of one transmitter and one receiver. These are followed by the simulation results for a test topology with two transmitters and two receivers, and then by simulation results for a test topology with ten transmitters and ten receivers. Initially, the links between routers were set to DS3 (45 Mbps). Then, the links are reduced to DS1 (1.54 Mbps). All the time, the links from end nodes to routers were maintained at 100 Mbps Ethernet. During the simulations, client computers downloaded files of 10 GB from servers using the FTP protocol. The comparisons were made between two protocol scenarios: one with the standard (unmodified) Reno TCP protocol and the other with the modified (proposed solution) TCP protocol. We compared the results between the standard TCP protocol and the proposed solution. Additionally, we compared the results obtained from our proposed solution to other well-known existing solutions.

4.1 One Transmitter and One Receiver

Fig. 5 displays the simulation results for this particular topology. On the left side, we observe the initial parameter values, while on the right side, we see the parameter values obtained by the evolutionary algorithm. The graph in Fig. 5 illustrates the curve that shows the throughput in the time domains for the client computer under three different scenarios:

- scenario 1 - 0% packet loss (shown in blue),
- scenario 2 - 1% packet loss (in red),
- scenario 3 - 5% packet loss (in green).

On the left side of Fig. 5 is a graph representing the initial parameter values, while on the right side, there is a graph displaying the parameter values obtained by the evolutionary algorithm.

The initial parameter values are (a = 2, b = 1, c = 1), whereas the evolutionary algorithm yielded the following parameter values for different scenarios: (a = 7, b = 13, c = 14) for scenario 1, (a = 2, b = 7, c = 8) for scenario 2, and (a = 2, b = 7, c = 1) for scenario 3. In this context, a represents the initial and minimal congestion window, b is the value of the congestion window increase, and c is the value of the congestion window decrease.

An important observation emerges when comparing the outcomes of the initial parameters to those obtained through the evolutionary algorithm. This primarily refers to the duration of transmission. In the first scenario, the duration is totalling 64 sec for the first case (from 100 sec to 164 sec), while it is totalling 60 sec for the second case (from 100 sec to 160 sec). It is an improvement of only around 5%, underscoring the fact that this improvement is realized under ideal conditions, with no packet losses along the transmission path. The network topology is displayed in Fig. 4.

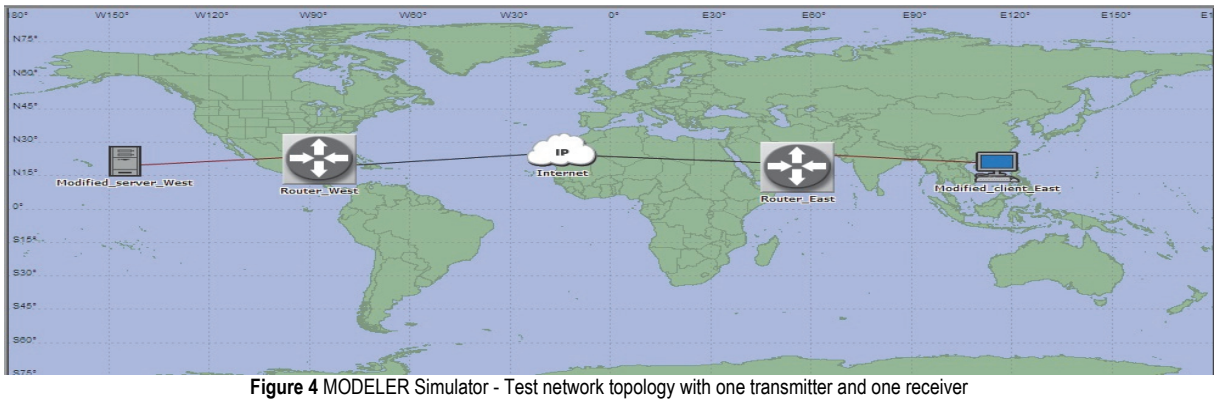


Figure 4 MODELER Simulator - Test network topology with one transmitter and one receiver

In the context of the second scenario, it is noteworthy that the transmission duration exhibits distinct disparities. For the initial parameters, the transmission time extends to 224 sec (a leap from 100 sec to 324 sec), while it totals to 72 sec for the parameters obtained by the evolutionary algorithm (from 100 sec to 172 sec). It represents a

difference of 152 sec, signifying an impressive improvement of approximately 70%.

For the third scenario, the transmission duration spans 502 sec for the initial parameters in the first case (from 100 sec to 602 sec), while it totals to 271 sec in the second case (from 100 sec to 371 sec). It represents a difference of 231 sec, which marks an improvement of approximately 45%.

Tab. 1 provided a comprehensive overview of the epochs within the evolutionary algorithm for simulations involving DS3 links, whereas z signifies the vector of parameters a, b and c, while t denotes the time needed for data transmission.

Table 1 Epochs of the evolutionary algorithm

Scenario 1 - 0% packet loss	Scenario 2 - 1% packet loss	Scenario 3 - 5% packet loss
z = 2, 1, 1; t = 164 s;	z = 2, 1, 1; t = 183 s;	z = 2, 1, 1; t = 514 s;
z = 3, 3, 0; t = 164 s;	z = 3, 3, 0; t = 184 s;	z = 1, 3, 2; t = 402 s;
z = 4, 3, 0; t = 164 s;	z = 2, 1, 1; t = 183 s;	z = 2, 2, 2; t = 395 s;
z = 4, 3, 4; t = 164 s;	z = 3, 1, 0; t = 183 s;	z = 1, 2, 1; t = 389 s;
z = 7, 2, 7; t = 164 s;	z = 3, 1, 4; t = 183 s;	z = 0, 1, 1; t = 389 s;
z = 8, 2, 8; t = 164 s;	z = 6, 0, 7; t = 183 s;	z = 0, 2, 1; t = 389 s;
z = 8, 2, 9; t = 164 s;	z = 7, 0, 8; t = 183 s;	z = 1, 2, 1; t = 389 s;
z = 9, 3, 10; t = 164 s;	z = 7, 0, 9; t = 183 s;	z = 0, 1, 1; t = 389 s;
z = 8, 4, 12; t = 164 s;	z = 8, 1, 10; t = 185 s;	z = 0, 2, 1; t = 389 s;
z = 8, 5, 13; t = 164 s;	z = 7, 0, 9; t = 183 s;	z = 1, 2, 2; t = 491 s;
z = 8, 5, 12; t = 164 s;	z = 6, 1, 11; t = 185 s;	z = 0, 2, 1; t = 491 s;
z = 9, 4, 11; t = 164 s;	z = 6, 2, 12; t = 182 s;	z = 0, 1, 0; t = 491 s;
z = 8, 1, 12; t = 164 s;	z = 6, 2, 11; t = 183 s;	z = 0, 2, 0; t = 491 s;
z = 8, 0, 13; t = 164 s;	z = 6, 2, 12; t = 182 s;	z = 0, 3, 0; t = 491 s;
z = 6, 0, 13; t = 164 s;	z = 7, 1, 11; t = 190 s;	z = 2, 3, 2; t = 407 s;
z = 7, 13, 14; t = 160 s	z = 2, 7, 8; t = 172 s;	z = 2, 7, 1; t = 351 s;

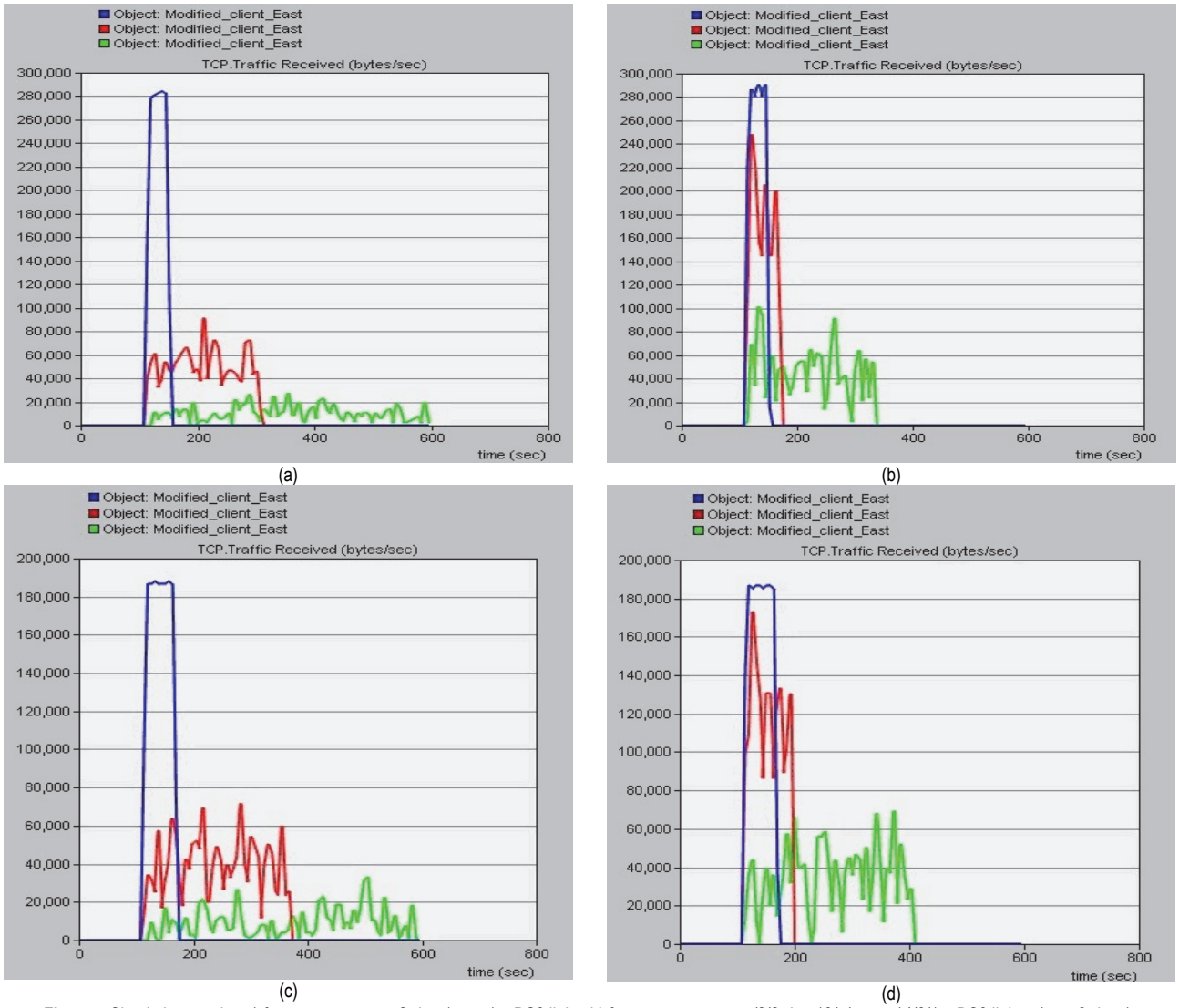


Figure 5 Simulation results: a) for parameters a = 2, b = 1, c = 1 – DS3 links; b) for parameters a = 7/2/2, b = 13/7/7, c = 14/8/1 – DS3 links; c) a = 2, b = 1, c = 1 – DS1 links; d) a = 7/2/2, b = 13/7/7, c = 14/8/1 – DS1 links

4.2 Two Transmitters and Two Receivers

This section delves into the presentation of simulation results encompassing a scenario involving two transmitters and two receivers. The network topology, shown in Fig. 6, mirrors the same configuration as in [32]. In this context, Client 1 downloads a 10 GB file from Server 1, while simultaneously, Client 2 downloads a 10 GB file from Server 2, using the FTP protocol. The initial scenario employs DS3 links (45 Mbps) between routers, then transitioning to DS1 (1.54 Mbps) capacity links. The links connecting end nodes to routers maintain a capacity of 100 Mbps Ethernet.

The first simulation involves unmodified settings for both servers and clients, specifically employing Reno TCP protocol. In the second simulation, Server 1 and Client 1 adhere to the standard Reno TCP protocol, while Server 2 and Client 2 engage with the modified TCP, in accordance with the proposed solution outlined in this paper.

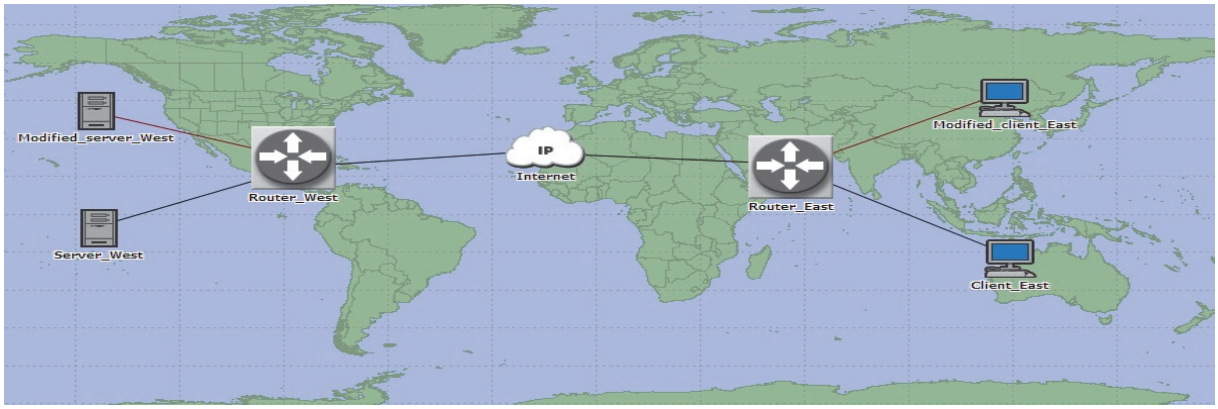
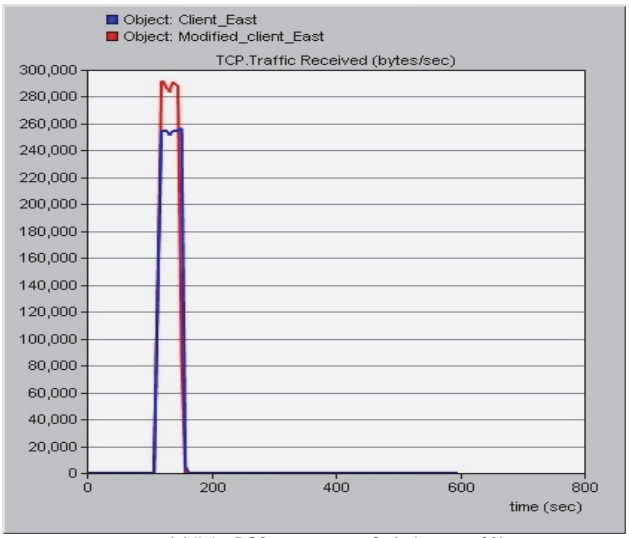
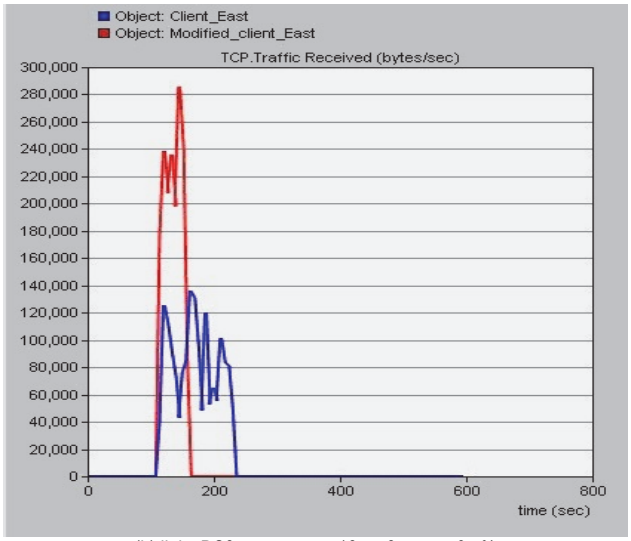


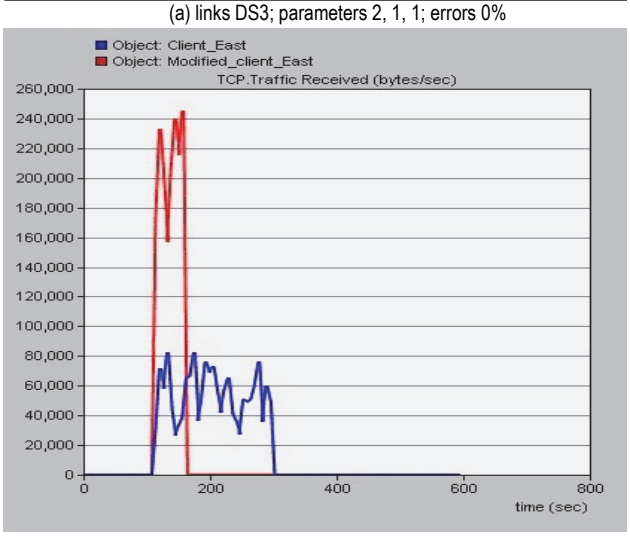
Figure 6 MODELER Simulator - Test network topology with two servers and two clients



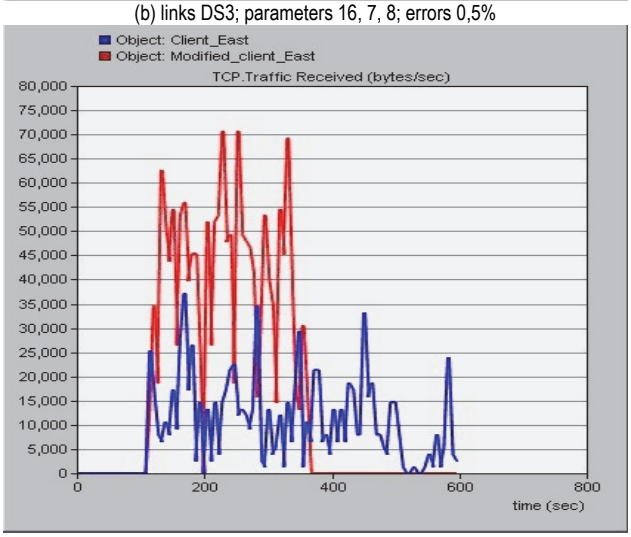
(a) links DS3; parameters 2, 1, 1; errors 0%



(b) links DS3; parameters 16, 7, 8; errors 0,5%

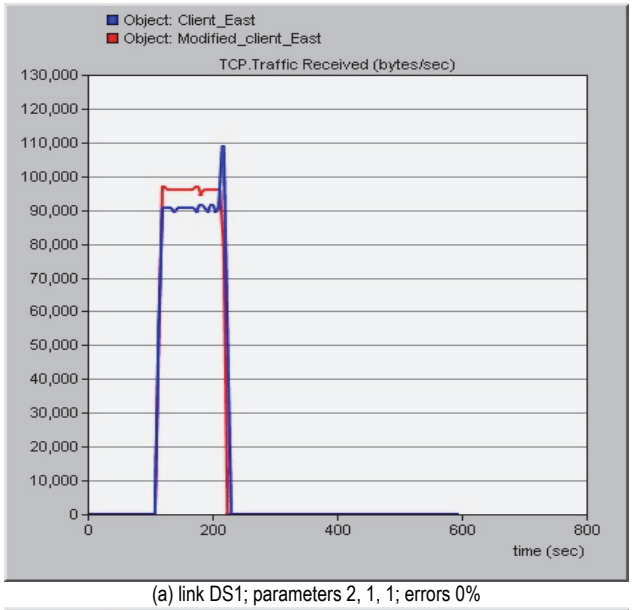


(c) links DS3; parameters 16, 7, 8; errors 1%

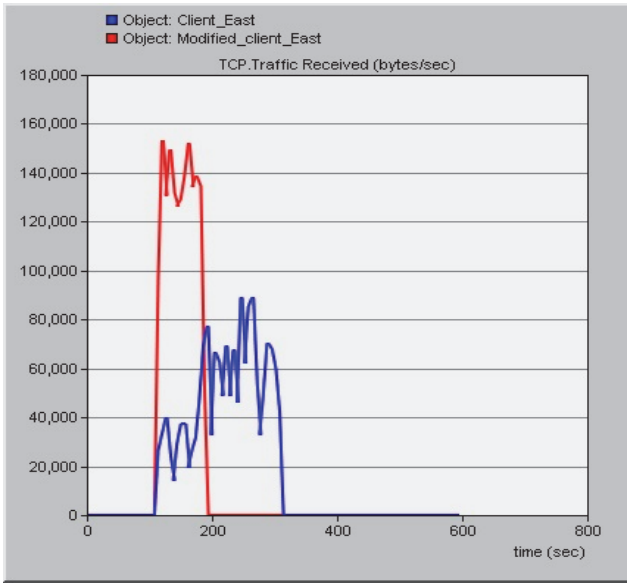


(d) links DS3; parameters 16, 7, 8; errors 5%

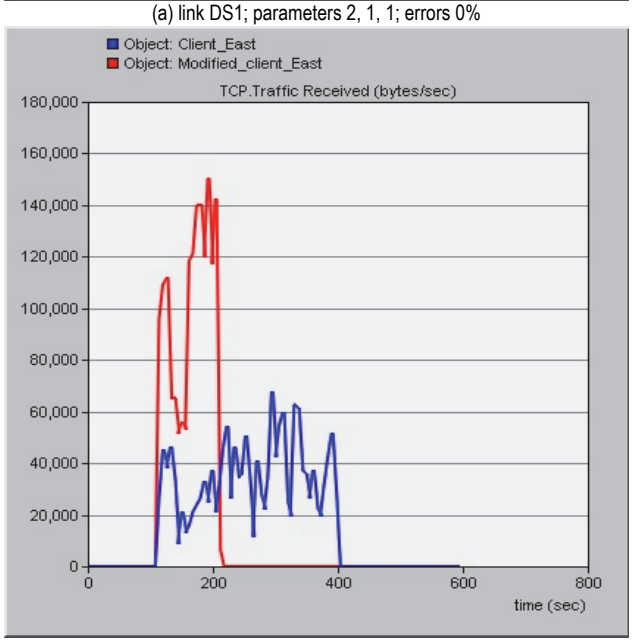
Figure 7 Simulation results: a) parameters 2/1/1, 0% packet loss, without modifications; b) parameters 16/7/8, 0,5% packet loss, with modification on Client 2 (in red); c) 1% packet loss; d) 5% packet loss



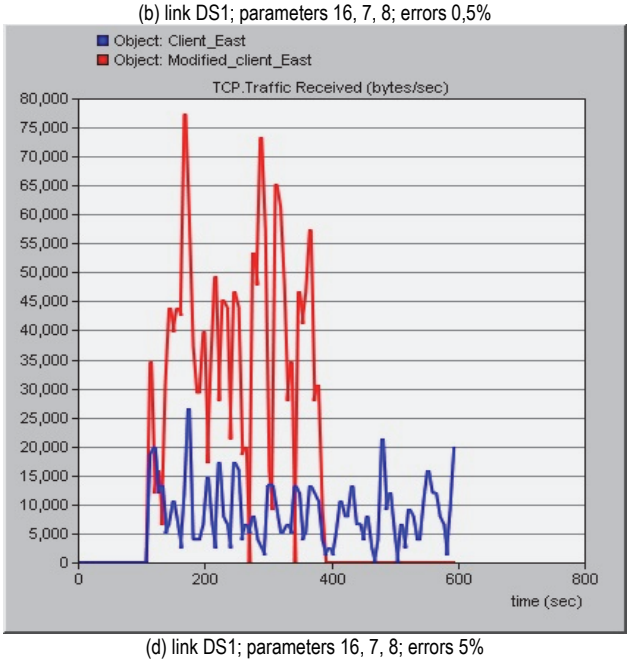
(a) link DS1; parameters 2, 1, 1; errors 0%



(b) link DS1; parameters 16, 7, 8; errors 0,5%



(c) link DS1; parameters 16, 7, 8; errors 1%



(d) link DS1; parameters 16, 7, 8; errors 5%

Figure 8 Simulation results: a) parameters 2/1/1, 0% packet loss, without modifications; b) parameters 16/7/8, 0,5% packet loss, with modification on Client 2 (in red); c) 1% packet loss; d) 5% packet loss

Fig. 7 illustrates the simulation results for scenarios involving high-bandwidth links between routers (DS3 - 45 Mbps), whereas Fig. 8 shows the simulation results for scenarios with low-bandwidth links between routers (DS1 - 1.54 Mbps).

Both Fig. 7a and Fig. 8a present the simulation results for the parameter values $a = 2$, $b = 1$, $c = 1$, whereas a represents the initial and minimal congestion window, b signifies the value of the congestion window increase, and c means the value of the congestion window decrease.

These initial parameters ($a = 2, b = 1, c = 1$) correspond to the application of an unaltered TCP protocol, namely the Reno TCP protocol, which is applied to both servers and clients.

Fig. 7b, Fig. 7c, Fig. 7d, Fig. 8b, Fig. .c and Fig. 8d present the simulation results for the parameters $a = 16$, $b = 7$, $c = 8$, which are applied to Server 2 and Client 2.

These parameter values were derived from the evolutionary algorithm as the optimal outcomes in the experiments discussed earlier. Meanwhile, Server 1 and Client 1 continue to use the unaltered Reno TCP protocol.

4.3 Ten Transmitters and Ten Receivers

The network topology, shown in Fig. 9 showcases ten nodes on one and ten nodes on the other side, constituting ten transmitters and ten receivers, along with the inter-network between them. Among the ten transmitters, one is modified, while the remainder remain unmodified. Similarly, one of the ten clients is modified, while the other nine remain unmodified. These clients engage in downloading 10 GB files from servers using the FTP protocol. The initial link configuration featured DS3 (45 Mbps) connections, subsequently transitioning to DS1 (1.54 Mbps) links. Furthermore, the links from end nodes to routers sustain a bandwidth of 100 Mbps Ethernet.

Fig. 10a to Fig. 17a show a comparison between the throughput of the first unmodified client and the modified client. Notably, the modified client displays higher

throughput, accompanied by shorter transmission duration. Similarly, Fig. 10b to Fig. 17b extend the comparison, encompassing the throughput of both the first and the last unmodified client in contrast to the modified client. Evidently, the modified client showcases greater throughput and shorter transmission time.

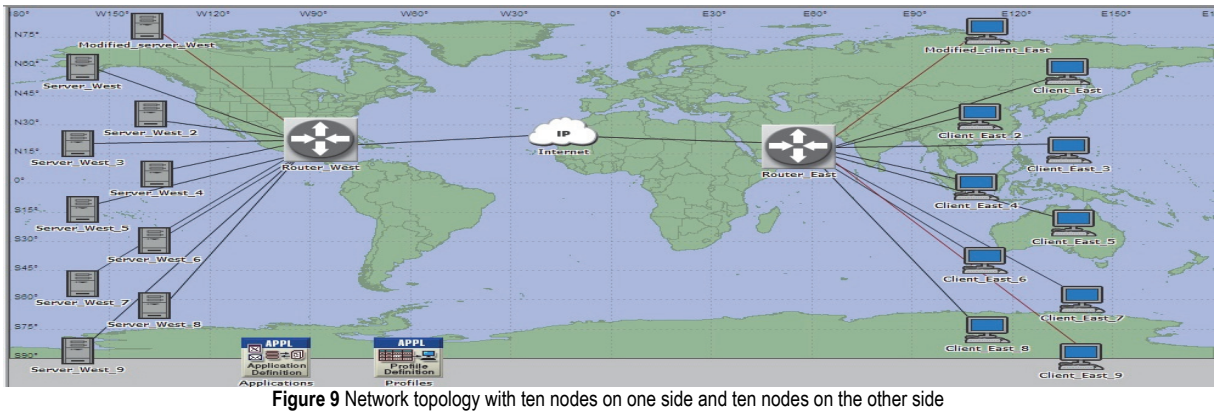
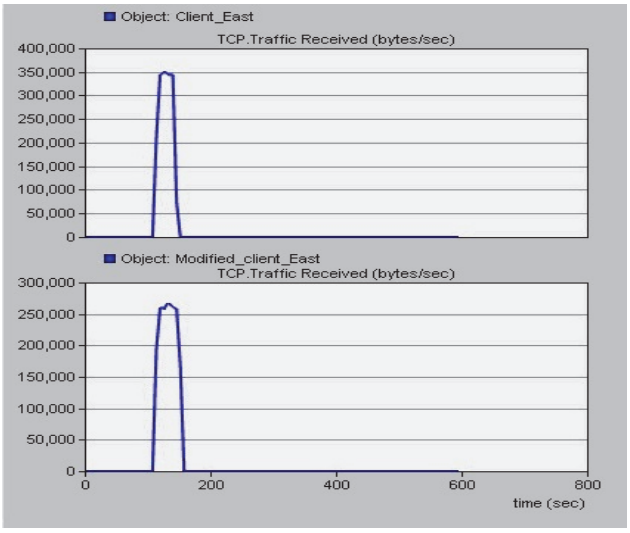
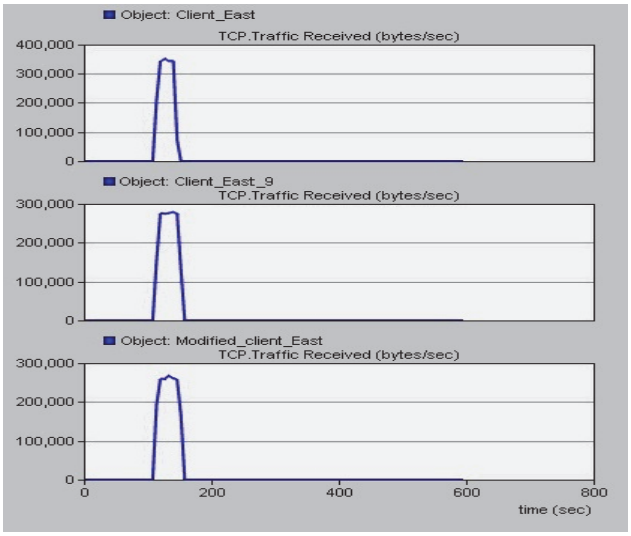


Figure 9 Network topology with ten nodes on one side and ten nodes on the other side

A noteworthy observation emerges from the results that the performance of the modified TCP consistently surpasses that of the standard TCP, even under conditions of heavy traffic load on links, mirroring the challenges encountered in real network congestion scenarios.

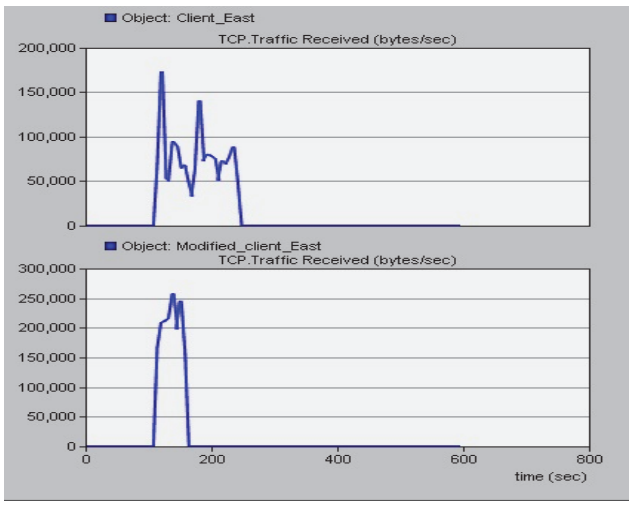


(a)

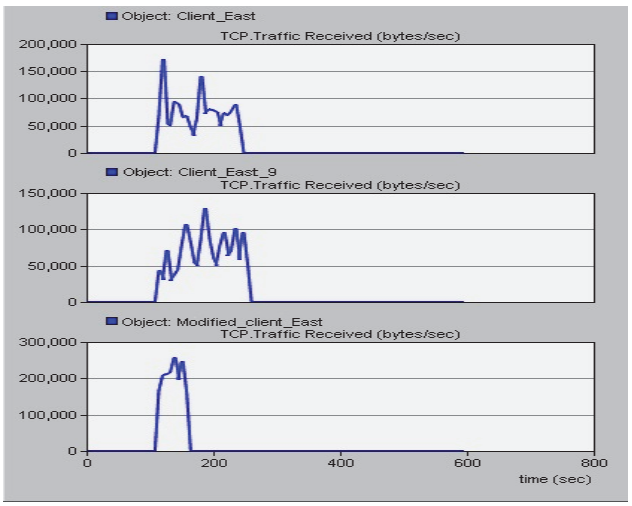


(b)

Figure 10 Simulation results for a 0% loss rate: a) unmodified client vs. modified client, high bandwidth links; b) first and last unmodified client vs. modified client, high bandwidth links (DS3)

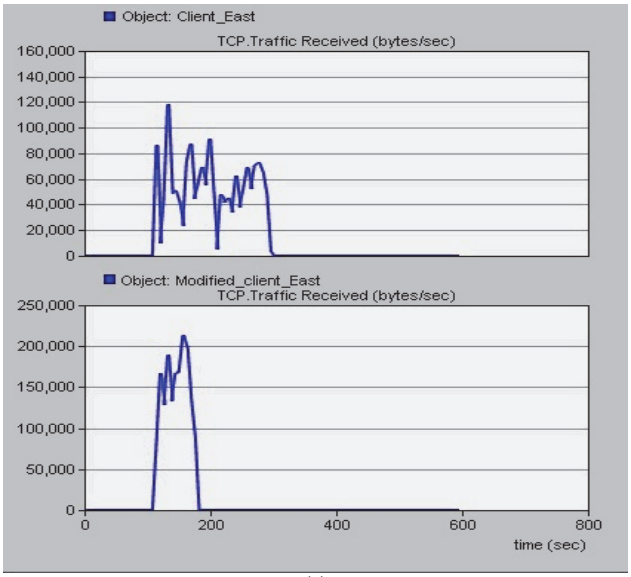


(a)

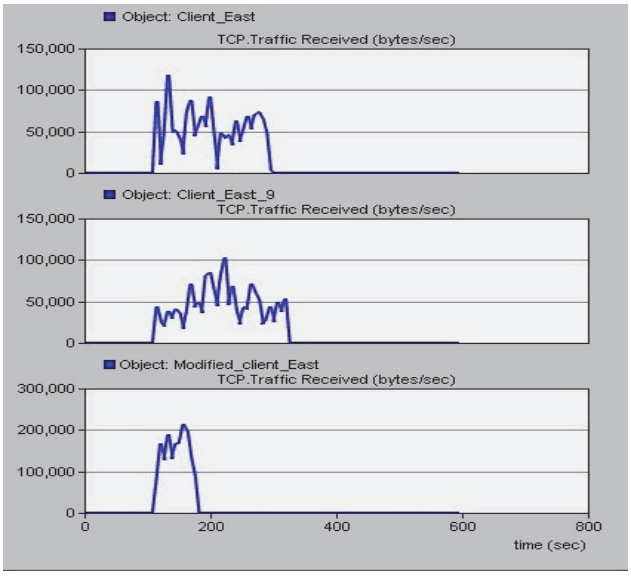


(b)

Figure 11 Simulation results for a 0.5% loss rate: a) unmodified client vs. modified client, high bandwidth links; b) first and last unmodified client vs. modified client, high bandwidth links (DS3)

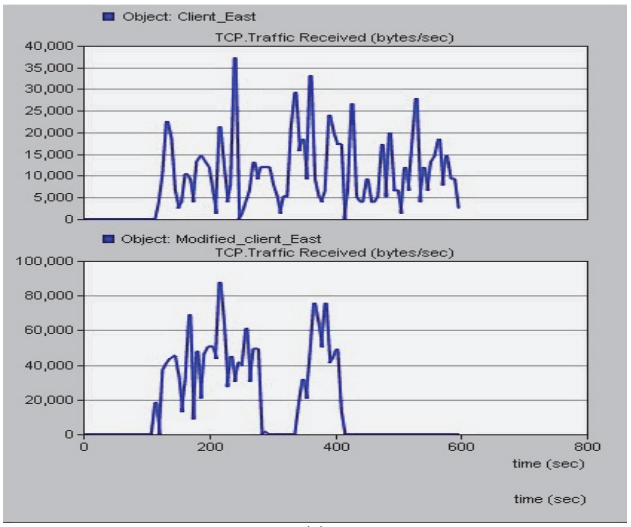


(a)

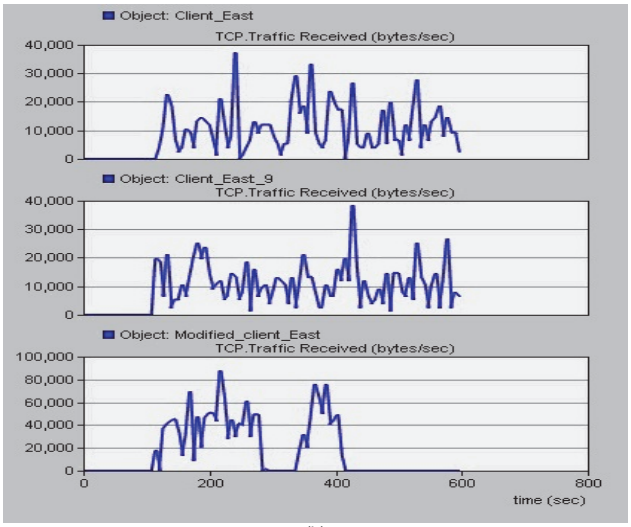


(b)

Figure 12 Simulation results for a 1% loss rate: a) unmodified client vs. modified client, high bandwidth links; b) first and last unmodified client vs. modified client, high bandwidth links (DS3)

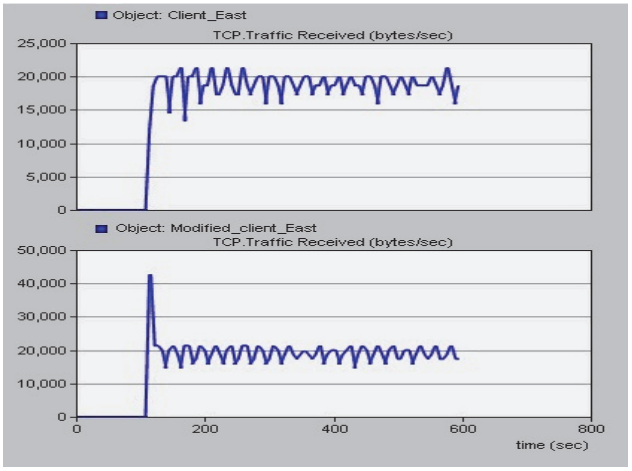


(a)

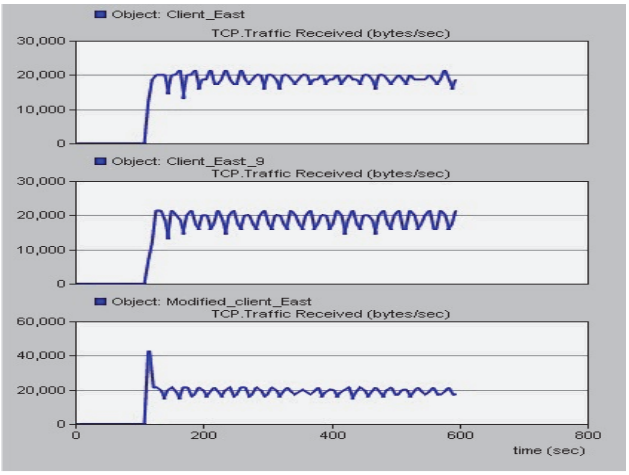


(b)

Figure 13 Simulation results for a 5% loss rate: a) unmodified client vs. modified client, high bandwidth links; b) first and last unmodified client vs. modified client, high bandwidth links (DS3)

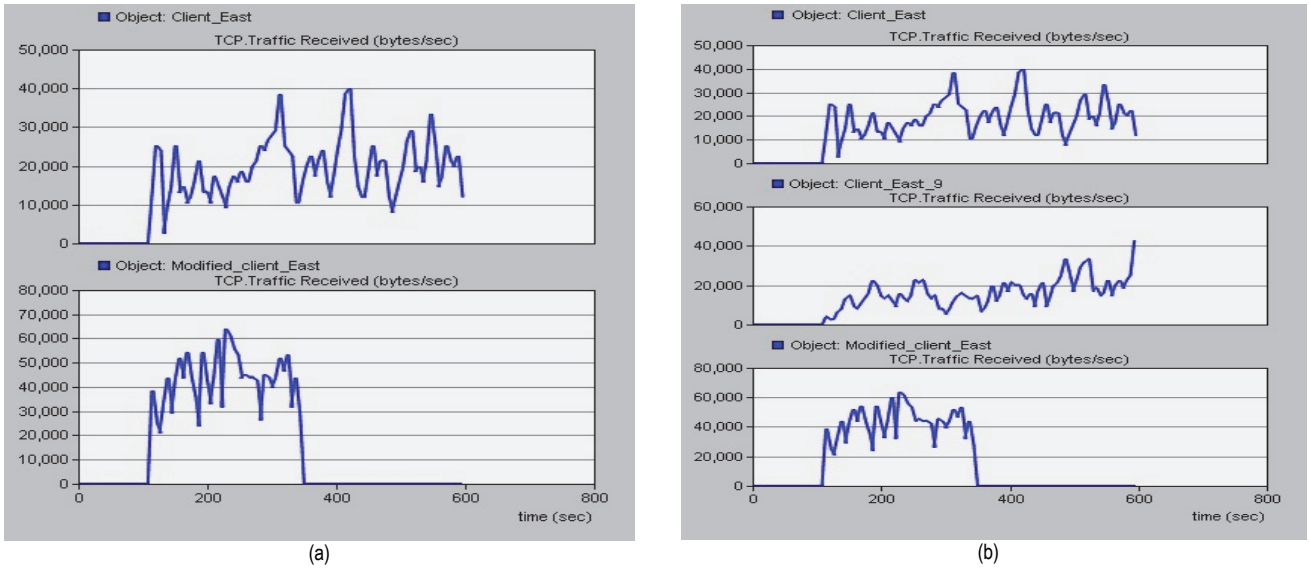


(a)



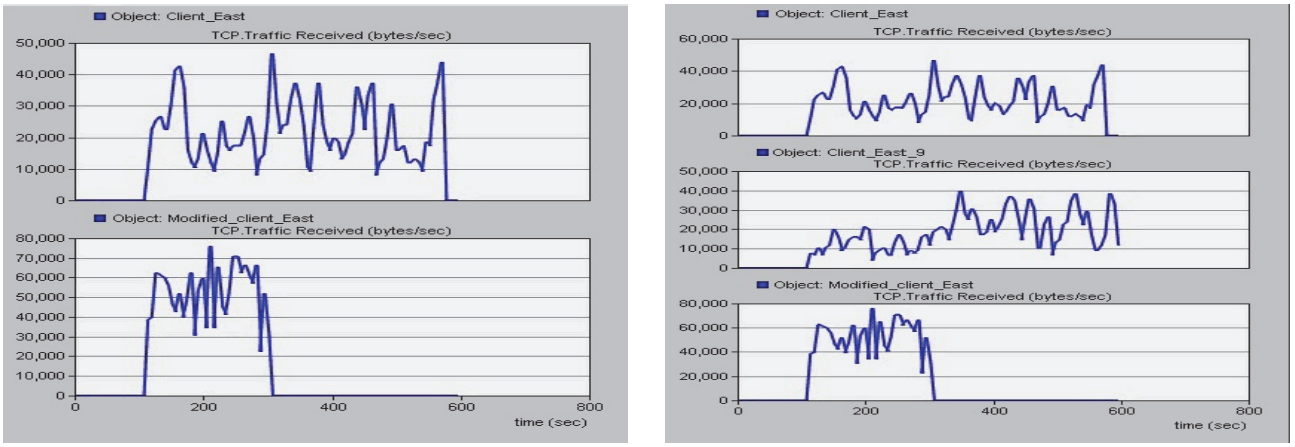
(b)

Figure 14 Simulation results for a 0% loss rate: a) unmodified client vs. modified client, low bandwidth links; b) first and last unmodified client vs. modified client, low bandwidth links (DS1)



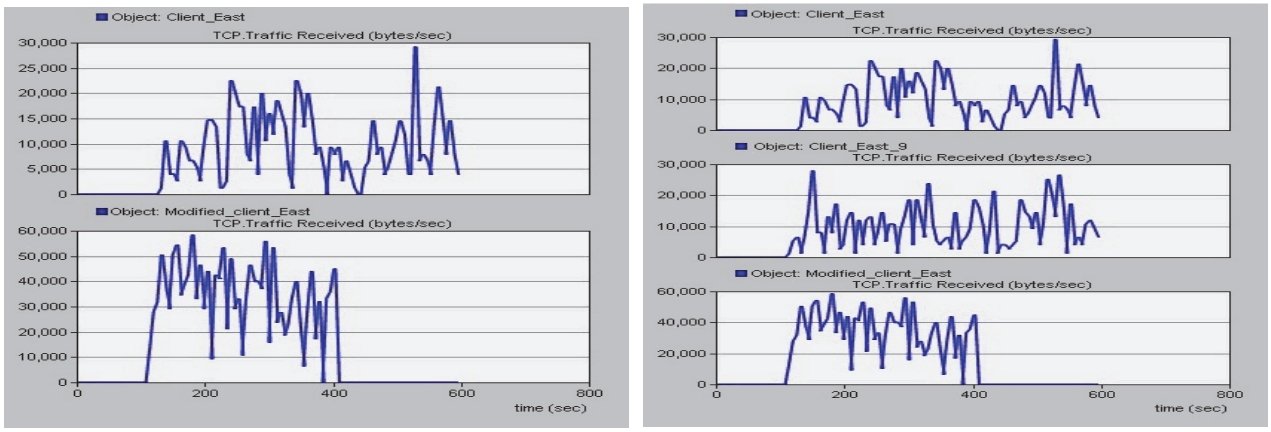
(a) (b)

Figure 15 Simulation results for a 0.5% loss rate: a) unmodified client vs. modified client, low bandwidth links; b) first and last unmodified client vs. modified client, low bandwidth links (DS1)



(a) (b)

Figure 16 Simulation results for a 1% loss rate: a) unmodified client vs. modified client, low bandwidth links; b) first and last unmodified client vs. modified client, low bandwidth links (DS1)



(a) (b)

Figure 17 Simulation results for a 5% loss rate: a) unmodified client vs. modified client, low bandwidth links; b) first and last unmodified client vs. modified client, low bandwidth links (DS1)

4.4 Comparison with Other Solutions

The proposed solution is compared with other well-known TCP protocol variants, such as TCP SACK, HS TCP, STCP and H-TCP. This comparison is executed within the same network setting as in [33], comprised of two transmitters and two receivers. The bottleneck segment of the network consists of routers and link between them. It is the same network topology employed in the previous section, with two transmitters and two receivers.

When contrasting the proposed solution, the modified TCP, with the Reno TCP enabling the SACK option,

compelling observations come to light. As presented in Fig. 18a, the modified TCP showcases no negative impact on the Reno TCP flow, maintaining identical transmission duration for both cases (120 sec). Notably, as illustrated in Fig. 18b, the results for the optimized parameter-driven modified TCP, display a performance improvement of up to 10 times than the results for the SACK TCP.

These experiments replicate the simulations conditions of [33], employing two transmitters and two receivers. Client 1 downloads a 10 GB file from Server 1, while Client 2 accesses a 10 GB file from Server 2 using the FTP protocol. While the standard Reno TCP with the SACK option activated runs on Server 1 and Client 1 (SACK TCP), the modified TCP, as described in this paper, operates on Client 2 and Server 2.

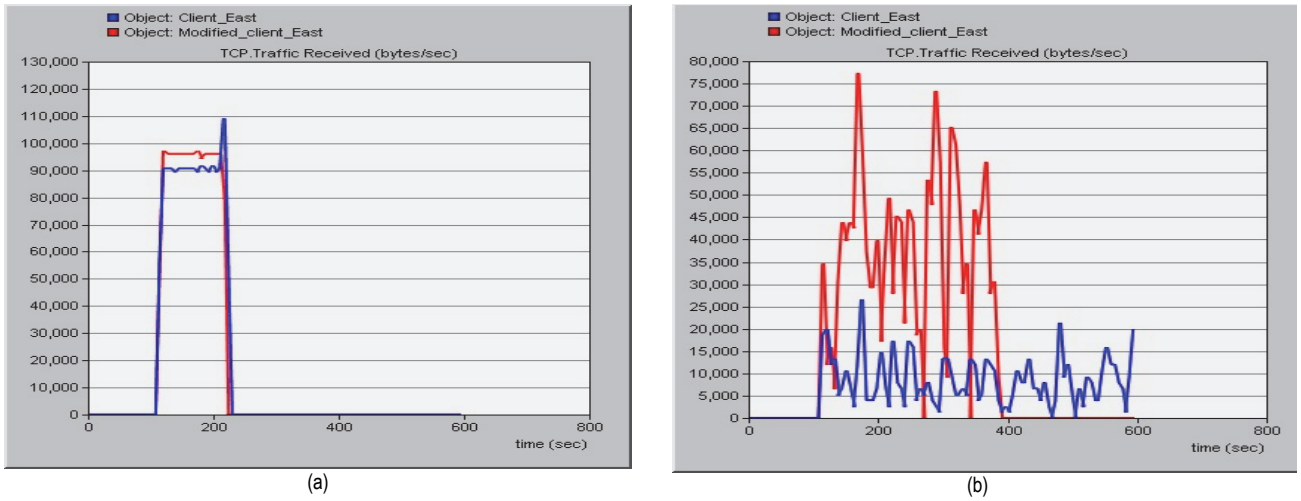


Figure 18 Simulation results: a) parameters 2/1/1, 0% packet loss; b) parameters 16/7/8, 5% packet loss; with modification on Client 2 (in red)

Fig. 18a shows the simulation results for the parameters $a = 2$, $b = 1$, $c = 1$, where a denotes the initial and minimal congestion window, b signifies the value of the congestion window increase, and c means the value of the congestion window decrease.

Fig. 18b, conversely, presents the simulation results for the parameters $a = 16$, $b = 7$, $c = 8$, obtained by the evolutionary algorithm as the optimal values from the experiments detailed above.

Comparing the simulation results in this study, as shown in Fig. 18, with the findings presented in [33], an intriguing observation emerges. The results of the modified TCP, proposed in this paper, align with or even surpass the performance of HS TCP and H-TCP, or even approach the STCP performance.

5 CONCLUSION

The presented method stands as an effective means to optimize TCP connection parameters. Thereby, the optimization approach harnesses heuristic methods, such as evolutionary strategies, to optimize the parameters. Each TCP connection represents one iteration in the process of seeking the optimal parameter settings.

Notably, the evolutionary algorithm has demonstrated its aptitude for identifying parameter values that lead to substantially improved TCP connection performance over a series of iterations. Also, it can be concluded that the evolutionary algorithm exhibits remarkable adaptability to the changes in network conditions.

In essence, leveraging heuristic methods to fine-tune the TCP protocol emerges as a promising solution. This paper anticipates the ongoing research of seeking new mechanisms and creating new versions of the TCP protocol. Finally, the prospect of applying heuristic methods for determining optimal TCP connection parameters holds substantial promise.

6 REFERENCES

[1] Jacobson, V. (1988). Congestion Avoidance and Control. *SIGCOMM Symposium on Communication Architecture and Protocols*. https://doi.org/10.1145/52324.52356

[2] Jacobson, V. (1990). Modified TCP Congestion Control and Avoidance Algorithms. *Technical Report 30.*

[3] Floyd, S. & Henderson, T. (1990). The New-Reno Modification to TCP's Fast Recovery Algorithm. *RFC 2582.*

[4] Fall, K. & Floyd, S. (1996). Simulation Based Comparison of Tahoe, Reno and SACK TCP. *ACM SIGCOMM Computer Communication Review Homepage archive*, *26*(3), 5-21. https://doi.org/10.1145/235160.235162

[5] Brakmo, L. S. & Peterson, L. L. (1995). TCP Vegas: End to End Congestion Avoidance on a Global Internet. *IEEE Journal on Selected Areas in Communication*, *13*, 1465-1490. https://doi.org/10.1109/49.464716

[6] Wei, D., Jin, C., Low, S., & Hegde, S. (2006). Fast TCP: Motivation, Architecture, Algorithms, Performance. *IEEE/ACM Transactions on Networking*, *14*(6). https://doi.org/10.1109/TNET.2006.886335

[7] Neal, C., Yuchung, C., Stephen, G. C., Soheil H. Y., & Jacobson V. (2016). BBR: Congestion-Based Congestion Control-Measuring bottleneck bandwidth and round-trip propagation time. *ACM Networks*, *14*(5). https://doi.org/10.1145/3012426.3022184

[8] King, R., Baraniuk, R., & Riedi, R. (2005). TCP-Africa: An Adaptive and Fair Rapid Increase Rule for Scalable TCP. Departments of Electrical and Computer Engineering and of Statistics, Rice University, Houston Texas. *IEEE INFOCOM.*

[9] Baiocchi, A., Castellani, A. P., & Vacirca, F. (2007). YeAH TCP: Yet Another Highspeed TCP. *IEEE INFOCOM.* University of Roma "Sapienza", Roma, Italy, Fifth International Workshop on Protocols for Fast Long-Distance Networks *PFLDnet.*

[10] Tan, K., Song, J., Zhang, Q., & Sridharan, M. (2006). A Compound TCP Approach for High-speed and Long Distance Networks. *IEEE INFOCOM.* https://doi.org/ 10.1109/INFOCOM.2006.188
[11] Floyd, S. (2003). HighSpeed TCP for Large Congestion Windows. https://doi.org/10.17487/RFC3649
[12] Kelly, T. (2003). Scalable TCP: Improving Performance in Highspeed Wide Area Networks. First International *Workshop on Protocols for Fast Long-Distance Networks, CERN, Geneva, Switzerland*. https://doi.org/10.1145/956981.956989
[13] Xu, L., Harfoush, K., & Rhee, I. (2004). Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks. *IEEE INFOCOM*. https://doi.org/10.1109/INFCOM.2004.1354672
[14] Ha, S., Rhee, I., & Xu, L. (2008). CUBIC: A New TCP Friendly High-Speed TCP Variant. *ACM SIGOPS Operating System Review*, *42*(5), 64-74. https://doi.org/10.1145/1400097.1400105
[15] Budzisz, L., Stanojevic, R., Schlote, A., Baker, F., & Shorten, R. (2011). On the Fair Coexistence of Loss and Delay-Based TCP. *IEEE/Acm Transactions on Networking*, *19*(6). https://doi.org/10.1109/TNET.2011.2159736
[16] Alizadeh, M., Greenberg, A., Maltz, D. A., Padhye, J., Patel, P., Prabhakar, B., Sengupta, S., & Sridharan, M. (2010). Data Center TCP (DCTCP). *SIGCOMM*. https://doi.org/10.1145/1851275.1851192
[17] Karunaharan, R. & Matta, I. (2002). WTCP: An Efficient Mechanism for Improving Wireless Access to TCP Services. https://doi.org/10.1002/dac.579
[18] Chu, J., Dukkipati, N., Cheng, Y., & Mathis, M. (2013). Increasing TCP's Initial Window. https://doi.org/10.17487/RFC6928
[19] Touch, J. (2012). Automating the Initial Window in TCP.
[20] Allman, M. (2015). Removing TCP's Initial Congestion Window. ICSI. *Internet Engineering Task Force.*
[21] Chiu, D. M., & Jain, R. (1989). Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks. *Computer Networks and ISDN Systems*, *17*, 1-14. https://doi.org/10.1016/0169-7552(89)90019-6
[22] Akella, A., Seshan, S., Karp, R., Shenker, S., & Papadimitriou, C. (2002). Selfish Behavior and Stability of the Internet: A Game-Theoretic Analysis of TCP. https://doi.org/10.1145/964725.633037
[23] Bethanabhotla, D., Caire, G., Neely, M. J., & Hsieh, M. (2013). Utility Optimal Scheduling and Admission Control for Adaptive Video Streaming in Small Cell Networks. *University of Southern California.* https://doi.org/10.1109/ISIT.2013.6620565
[24] Bethanabhotla, D., Caire, G., & Neely, M. J. (2012). Joint Transmission Scheduling and Congestion Control for Adaptive Streaming in Wireless Device-to-Device Networks. *University of Southern California.* https://doi.org/10.1109/ACSSC.2012.6489207
[25] Winstein, K., & Balakrishnan, H. (2013). TCP ex Machina: Computer-Generated Congestion Control. *Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge.* https://doi.org/10.1145/2486001.2486020
[26] Xiaohui, N., Youjian, Z., Zhihan, L., Guo, C., Kaixin, S., Jiyang, Z., Zijie, Y., & Dan, P. (2019). Dynamic TCP Initial Windows and Congestion Control Schemes Through Reinforcement Learning. *IEEE Journal on Selected Areas in Communications*, *37*(6), 1231-1247. https:// doi.org/10.1109/JSAC.2019.2904350
[27] Wang, Y., Li, H. L., Fei, Z. X., & Zhao, H. Y. (2019). Research and Prospect of TCP Optimization in 5G Multi-ACCESS Networks. *Journal of Beijing University of Posts and Telecommunications.*
[28] Srinidhi, N. N., Kumar, S. M., & Venugopa, K. R. (2018). Network optimization in the Internet of Things: A review. *Engineering Science and Technology, an International Journal.* https://doi.org/10.1016/j.jestch.2018.09.003
[29] Sun, S., Jiang, W., Feng, G., Qin, S., & Yuan, Y. (2019). Cooperative Caching with Content Popularity Prediction for Mobile Edge Caching. *Tehnicki Vjesnik*, *Croatia*.
[30] See https://buildbot.tools.ietf.org/html/rfc6928
[31] See http://www.scholarpedia.org/article/ Evolution_strategies
[32] Ceco, A. & Mrdovic, S. (2019). Test bed for network protocols optimization. *Telfor Journal*, *11*(1), https://doi.org/10.5937/telfor1901014C
[33] Cho, S. & Bettati, R. (2005). Adaptive TCP for Effective and Fair Utilization of High Bandwidth-Delay Product Networks. *Department of Computer Science, Texas A&M University, College Station, TX 77843 USA*

Contact information:

Afan CECO
(Corresponding author)
Faculty of Electrical Engineering, University of Sarajevo,
Zmaja od Bosne bb, 71 000 Sarajevo, Bosnia & Herzegovina
E-mail: afan.ceco@etf.unsa.ba

Sasa MRDOVIC
Faculty of Electrical Engineering, University of Sarajevo,
Zmaja od Bosne bb, 71 000 Sarajevo, Bosnia & Herzegovina
E-mail: sasa.mrdovic@etf.unsa.ba